

١٢ / ٢ / ١٣٨٥

بسم الله الرحمن الرحيم

١٢٧٤١



وزارت علوم، تحقیقات و فناوری

پژوهشگاه اطلاعات و مدارک علمی ایران

Iranian Research Institute
for Scientific Information
and Documentation
(IRANDOC)

تجلی

شماره: NO. ۲۱۰۵

تاریخ: ۱۳۸۵/۱۲/۱۸

یوت: Ref.

دبیرخانه شورای پژوهش

پژوهشگاه اطلاعات و مدارک علمی ایران

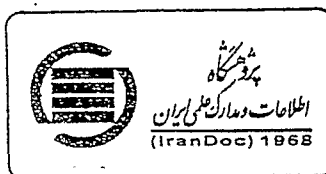
۱۱۸۵ / ۲ / ۱۳۸۵

بدینوسیله انجام طرح پژوهشی بهینه
سازی اطلاعات علمی شیمی و مهندسی
شیمی (سمینارها) توسط آقای رجبی به عنوان
مجری در پژوهشگاه اطلاعات و مدارک علمی ایران
و گزارش نهایی این طرح بر اساس نظر آقای دکتر
پورجوادی به عنوان داور تایید می گردد.
لازم می داند مراتب قدردانی و سپاس خود را به
کلیه دست اندرکاران این طرح تقدیم دارد.

توفیق از اوست

حسین غریبی

رئیس مرکز اطلاعات و مدارک علمی ایران



تهران - خیابان انقلاب، چهارراه فلسطین، شماره ۱۱۸۸ صندوق پستی: ۱۳۷۱-۱۳۱۸۵

تلفن: ۶۶۹۵۱۳۳۰ (خط ۱۰) - ۶۶۴۹۴۹۸۰ (خط ۵) دورنگار: ۶۶۴۶۲۲۵۴ تلفن گویا: ۶۶۴۹۴۹۵۴

1188, Enqelab Ave, Tehran, Iran, P.O. Box: 13185 - 1371 Tel: (+9821) 6649 4955 - 6646 2548

Fax: (+9821) 6646 2254 Email: info@irandoc.ac.ir

www.irandoc.ac.ir

۲۲۷۶۱

وزارت علوم، تحقیقات و فناوری
پژوهشگاه اطلاعات و مدارک علمی ایران
گروه پژوهشی زبان‌شناسی

بهینه‌سازی اطلاعات علمی شیمی و مهندسی شیمی (سمینارها) در بانک
جامع اطلاعات پژوهشگاه اطلاعات و مدارک علمی ایران و طراحی نرم افزار
بانک ترکیبات شیمیایی

گزارش طرح پژوهش

مجری: تقی رجبی

ناظر:

دکتر حسین غریبی

همکاران:

حسین غریبی

امیر پارسی اصفهانی

مهرداد نوروزی اقبالی

فهرست مطالب

صفحه	عنوان
۱	چکیده
۲	مقدمه
۳	سوالات پژوهش
	فصل اول: بررسی وضعیت موجود (مشاهدات و استنتاجات)
۵	۱- بررسی کمی بانک جامع اطلاعات پژوهشگاه از نظر تعداد و موضوع رکوردهای اطلاعاتی سمینارهای شیمی و مهندسی شیمی
۵	۱-۱- سمینارهای موجود
۵	۱-۲- سمینارهای غیرموجود
۶	۲- مشکلات کاربرگه‌های موجود و ضرورت طراحی کاربرگه‌های جدید
۶	۳- مشکلات نرم افزاری و ضرورت طراحی نرم افزار جدید
	فصل دوم: به سوی وضعیت مطلوب
۹	۱- تکمیل اطلاعات موجود
۹	۱-۱- گردآوری سمینارهای شیمی و مهندسی شیمی غایب در بانک پژوهشگاه
۹	۱-۲- استخراج اطلاعات از بانک جامع پژوهشگاه و نوع فرمت آن
۱۱	۱-۳- دسته بندی اطلاعات
۱۳	۲- پردازش اطلاعات
۱۳	۲-۱- ویرایش و نمایه‌سازی مجدد (تصحیح، حذف و افزایش کلیدواژه‌ها)
۱۳	۲-۲- یکسان‌سازی و استاندارد سازی (عناوین، کلیدواژه‌ها و اسامی ترکیبات شیمیایی)
۱۳	۲-۳- معادل‌یابی فارسی و لاتین
۱۳	۲-۴- ارجاع‌دهی و استخراج کوتاه‌نوشته‌ها
۱۳	۲-۵- استخراج اسامی ترکیبات شیمیایی
۱۳	۲-۶- تصحیح اسامی ترکیبات شیمیایی و ورود اطلاعات
۱۳	۲-۷- فرمول‌دهی به اسامی ترکیبات شیمیایی و ورود اطلاعات به ماشین
۱۳	۲-۸- ایجاد نمایه‌نامه فرمولی
۱۳	۲-۹- ایجاد نمایه‌نامه اسامی ترکیبات شیمیایی
۱۳	۲-۱۰- یکسان‌سازی اطلاعات کلیه فیلدهای مدرک شناختی

۱۳	۲-۱۱- تبدیل فرمت کلیه اطلاعات پردازش شده از سی.دی.اس به زرنگار یا وورد
۱۳	۲-۱۲- اختصاص دادن شماره ثبت شیمیایی به اغلب ترکیبات شیمیایی و ساخت نمایه‌نامه شماره‌های ثبت شیمیایی
۱۳	۲-۱۳- صفحه آرای و ویرایش نهایی
۱۴	۳- طراحی کاربرگه اطلاعات شیمی
۱۴	۳-۱- فیلدهای مورد نیاز و قابل بازیابی در کاربرگه
۱۶	۳-۲- نمونه کاربرگه (فرم خام)
۱۹	۳-۳- نمونه کاربرگه (فرم پر شده)
۲۴	۴- طراحی نرم افزار بانک ترکیبات شیمیایی
۲۴	۴-۱- مرور ادبیات نرم افزاری
۲۶	۴-۲- طراحی بخش الگوهای شیمیایی
۲۷	۴-۳- طراحی بخش ذخیره سازی اطلاعات
۲۹	۴-۴- آزمایش نهایی بر روی نرم افزار مورد نظر و رفع مشکلات احتمالی
	فصل سوم: اهداف، خلاصه و نتیجه گیری
۳۱	۱- اهداف و خلاصه پژوهش
۳۱	۱-۱- آموزش نیروی انسانی (دو نفر)
۳۱	۱-۲- طراحی کاربرگه استاندارد ورود اطلاعات ترکیبات شیمیایی
۳۱	۱-۳- طراحی بانک نرم افزار ترکیبات شیمیایی
۳۲	۱-۴- چاپ کتاب سمینارهای شیمی و مهندسی شیمی ایران
۳۳	۲- نتایج پژوهش
۳۳	۳- مشکلات پژوهش
۳۵	مراجع
۴۲	کد اصلی نرم افزار
	پرسشنامه طرح پژوهش
	نظر داور طرح

چکیده

در این طرح اطلاعات علمی ۲۷۱۴ رکورد اطلاعاتی مربوط به سمینارهای شیمی و مهندسی شیمی به صورت یک مجموعه واحد و دارای نمایه‌نامه‌های تخصصی به صورت چاپی و نیز نرم افزاری ارائه شده است. این سمینارها در فاصله زمانی سالهای ۱۳۶۸ تا ۱۳۷۹ هجری شمسی در کشور ایران برگزار شده‌اند. در گذشته اطلاعات برخی از این سمینارها در قالب مجموعه مقالاتی در پژوهشگاه اطلاعات و مدارک علمی ایران گردآوری، نمایه و چکیده نویسی شده و به شکل غیر موضوعی در مجلدهای جداگانه‌ای به چاپ رسیده‌اند. در این طرح اطلاعات غایب در بانک جمع آوری و وارد چرخه نمایه‌سازی شدند و سپس کلیه این اطلاعات مورد بازبینی، ویرایش و بازپردازش تخصصی قرار گرفتند. بخصوص کلیدواژه‌ها که بعضاً بصورت غیراستاندارد گزینش شده بودند و نیز بصورت غیریکسان و گاه بی‌معنی به لحاظ تخصصی وارد بانک اطلاعات جامع پژوهشگاه شده و نیز به چاپ رسیده بودند، مورد استاندارد کردن، یکسان‌سازی، ارجاع دهی و معادل‌یابی قرار گرفتند و در نهایت در یک مجلد به چاپ رسیدند، تا مورد استفاده دانش پژوهان و محققان در بخش صنعت و دانشگاه جهت تعیین اولویتهای تحقیقاتی و جلوگیری از دوباره کاری قرار گیرد. در این طرح کنترل نامگذاری‌ها با توجه به استانداردهای ایوپاک انجام شده است. همچنین شیوه بازیابی اطلاعات شیمی افزون بر جستجوی موضوعی با استفاده از فرمول‌ها، اسامی ترکیبات شیمیایی و نیز شماره‌های ثبت شیمیایی عملی شده است. به منظور تقویت و اصلاح در ساختار تشکیلات نمایه‌سازی و چکیده نویسی در پژوهشگاه، دو نفر کارشناس شیمی مورد آموزش قرار گرفتند که هم اکنون در بخش مدیریت اسناد و اطلاعات مشغول فعالیت می‌باشند. طراحی نرم افزار تخصصی شیمی در ثبت، ذخیره و بازیابی اطلاعات ترکیبات شیمیایی، ایجاد نمایه‌نامه‌های تخصصی شیمی، طراحی کاربرگ‌های استاندارد ورود اطلاعات شیمی و استاندارد کردن شیوه‌های ذخیره و بازیابی اطلاعات شیمی از موارد مورد توجه در این طرح است. همچنین فهرستهای تخصصی و واژه‌های یکدست شده در این طرح می‌توانند توسط دانشجویان، اساتید، محققان، رساله نویسان، نمایه سازان، مترجمان و فرهنگ نویسان مورد استفاده قرار گیرند.

مقدمه:

برای شروع به کار جهت باز پردازش اطلاعات سمینارهای شیمی و مهندسی شیمی ایران ابتدا بانک جامع اطلاعات پژوهشگاه به لحاظ کمی مورد بررسی قرار گرفت تا مشخص شود چه تعدادی از سمینارهای برگزار شده در کشور در این بانک موجود است. با مشخص شدن نواقص بانک از نظر کمی و مکاتبه با سازمانهای مربوطه مانند انجمن شیمی و مهندسی شیمی ایران به بخش عمده‌ای از چکیده‌های مورد نظر دسترسی پیدا کردیم. این اطلاعات پس از طی مراحل نمایه‌سازی و ورود اطلاعات در بخش مدیریت اسناد و اطلاعات، به بانکهای پژوهشگاه افزوده شد. سپس با انتخاب کلید واژه‌های مناسب کلیه رکوردهای مربوط به سمینارهای شیمی و مهندسی شیمی از بانک جامع اطلاعات پژوهشگاه استخراج گردید. پس از آن با استفاده از امکانات نرم افزار وین‌آسیس، این اطلاعات دسته‌بندی شده و عملیات باز پردازش تخصصی روی آنها صورت گرفت.

در گزارش طرح پژوهشی پیشین، بصورت مشروح به مشکلات عدیده موجود در بازیابی اطلاعات علمی شیمی و مهندسی شیمی در بانک جامع اطلاعات پژوهشگاه اشاره نمودیم و ویژگیهای خاص اصطلاحات شیمی و اسامی ترکیبات شیمیایی و نحوه رفتار با آنها را در ایجاد پایگاه‌ها و نمایه‌نامه‌های تخصصی شیمی بر شمردیم [۱۸]. در آنجا اشاره کردیم که عدم وجود کاربرگه‌های استاندارد و متناسب با اطلاعات علمی شیمی یکی از مشکلات اساسی در ایجاد پایگاه‌های اطلاعاتی شیمی و رشته‌های وابسته به آن است. نیز وعده کرده بودیم که به طراحی و پیشنهاد کاربرگه‌های ثبت اطلاعات علمی شیمی اقدام نماییم. کاربرگه‌های طراحی شده در این طرح، به نمایه‌سازان و محققان در ثبت استاندارد اطلاعات کمک شایانی خواهند نمود و نیز پردازش تخصصی اطلاعات را بصورت سهولتی عملی خواهند کرد و در واقع گامی در جهت عملیاتی شدن فرآیند پردازش ماشینی اطلاعات علمی شیمی و مهندسی شیمی و نیز تبادل اطلاعات از راه دور خواهند بود. این کاربرگه‌ها به نمایه ساز و ماشین نویس امکان ورود اطلاعات به شکل تخصصی و تفکیک شده را می‌دهد و همچنین کاربر را قادر می‌سازد با استفاده از نمایه‌نامه‌های مورد نظر خود به جستجوی اطلاعات بپردازد.

در طرح پژوهشی پیشین به تفصیل مشکلات نرم‌افزار سی.دی.اس آسیس را در ثبت، نمایش و بازیابی علائم و ساختارهای شیمیایی بر شمردیم [۱۸]. در ویرایش تحت ویندوز این نرم افزار یعنی وین‌آسیس نیز این مشکلات مرتفع نشده است. در این طرح، نرم‌افزاری به منظور ذخیره و بازیابی اطلاعات ترکیبات شیمیایی طراحی و کلیه فیلدهای موجود در کاربرگه‌های طراحی شده در

آن منعکس گردیده است. این نرم افزار قادر به رسم، نمایش، ذخیره و بازیابی اغلب اشکال و انواع اطلاعات و علائم مورد استفاده در متون شیمی می باشد.

سوالات پژوهش:

۱. چگونه می توان اطلاعات سمینارهای شیمی و مهندسی شیمی را از مجموعه بانک جامع اطلاعات پژوهشگاه استخراج نمود؟
۲. چگونه می توان اطلاعات سمینارهای برگزار شده و غیر موجود در بانک جامع اطلاعات را به این بانک اضافه نمود؟
۳. آیا کاربرگه های موجود ثبت اطلاعات علمی شیمی در انواع مدارک مناسب و استاندارد است؟
۴. شکل مناسب کاربرگه استاندارد ثبت اطلاعات شیمی چگونه است؟
۵. چرا بازیافت اطلاعات علمی شیمی در بانک جامع اطلاعات مرکز به نحو مطلوب ممکن نیست؟
۶. آیا نرم افزار سی.دی.اس آیسس تحت داس یا ویندوز به شکل فعلی برای نمایه سازی و ذخیره سازی و بازیافت اطلاعات علمی شیمی در انواع مدارک مناسب و استاندارد است؟
۷. آیا می توان نرم افزاری طراحی کرد به نحوی که قادر باشد ساختارها و علائم شیمیایی را نمایش دهد و مانند یک پایگاه اطلاعاتی عمل نموده و در ذخیره سازی و بازیافت اطلاعات با نمایه نامه های تخصصی مانند نمایه نامه فرمولی، اسامی ترکیبات شیمیایی، ساختارهای حلقوی و یا شماره های ثبت شیمیایی موفق باشد؟
۸. آیا می توان مجموعه اطلاعات سمینارهای شیمی برگزار شده در کشور را بصورت یک مجلد با نمایه نامه های تخصصی ایجاد نمود؟ چگونه؟
۹. آیا می توان دو نفر از کارشناسان شیمی را مورد آموزش قرار داد تا بتوانند از نرم افزار وین آیسس، نرم افزارهای تخصصی شیمی و اینترنت برای نمایه سازی و چکیده نویسی تخصصی شیمی بهره ببرند؟ چگونه؟

فصل اول

مشاهدات و استنتاجات
(بررسي وضعيت موجود)

۱- بررسی کمی بانک جامع اطلاعات پژوهشگاه از نظر تعداد و موضوع رکوردهای اطلاعاتی
سمینارهای شیمی و مهندسی شیمی
وضعیت بانک جامع اطلاعات مرکز اطلاعات و مدارک علمی ایران از نظر تعداد و عناوین
سمینارهای شیمی و مهندسی شیمی ایران قبل از افزایش ناقصی ها و در زمان شروع طرح بصورت
زیر گزارش می شود.

۱-۱ - سمینارهای موجود:

- سمینار شیمی فیزیک ایران، دومین
- سمینار تخصصی شیمی آلی ایران، سومین ۱
- سمینار شیمی تجزیه ایران، هفتمین
- کنگره شیمی و مهندسی شیمی ایران، اولین، دومین، چهارمین، پنجمین، ششمین، نهمین و یازدهمین
- کنگره بین المللی شیمی و مهندسی شیمی ایران، اولین
- سمینار دوسالانه الکتروشیمی ایران، اولین
- سمپوزیم متالورژی ایران، اولین
- کنگره ملی مهندسی شیمی ایران، اولین
- کنگره بیوشیمی، دومین
- کنگره ملی خوردگی، چهارمین
- سمپوزیم سراسری خوردگی در صنایع ایران، اولین

۱-۲ - سمینارهای غیرموجود:

- سمینار شیمی فیزیک ایران، اولین و سومین
- سمینار تخصصی شیمی آلی ایران، اولین و چهارمین
- سمینار شیمی تجزیه ایران، اولین، دومین، سومین، چهارمین، پنجمین، ششمین
- کنگره بین المللی شیمی و مهندسی شیمی ایران، اولین و دومین
- کنگره انجمن شیمی و مهندسی شیمی ایران، سومین، هفتمین و دهمین
- کنگره ملی شیمی و مهندسی شیمی ایران، دوازدهمین

۲- مشکلات کاربرگه‌های موجود و ضرورت طراحی کاربرگه‌های جدید

در کاربرگه‌های موجود تعدادی از فیلدها برای درج اطلاعات مدرک شناختی مدارک در نظر گرفته شده است که در اینجا مورد بحث ما نیستند و ما آنها را بدون هیچ تغییری باقی خواهیم گذاشت. در بین کلیه فیلدها در کاربرگه‌های موجود، سه فیلد عنوان، کلید واژه و چکیده فیلدهای اصلی شناخته شده برای قرار گیری اطلاعات علمی موجود در مدارک می باشند. همانطوری که در گزارش طرح پژوهشی پیشین آورده‌ایم برای ثبت اطلاعات علمی مدارک شیمی به فیلدهای بیشتری نیاز داریم تا مستند سازی مدارک علمی شیمی به روش استاندارد صورت پذیرد [۱۸].

هر ماده شیمیایی را می‌توان علاوه بر نمایش آن توسط نام ترکیب، با فرمول ساختاری، شماره ثبت شیمیایی، خواص فیزیکوشیمیایی، خواص طیفی و خواص تقارنی نیز معرفی نمود. از آنجا که متون شیمی بخصوص در حوزه شیمی آلی و شیمی معدنی عموماً حاوی این گونه نمایش‌ها هستند، پایگاه‌های اطلاعاتی شیمی، نیز باید قابلیت ثبت، نمایش و بازیابی این نوع نمایش‌ها را داشته باشند [۲، ۳، ۴، ۱۸].

در پایگاه‌های اطلاعاتی معروف شیمی علاوه بر اینکه انواع و اقسام نمایشهای شیمیایی قابل مشاهده هستند امکان بازیابی اطلاعات بر اساس آن نمایشها وجود دارد [۳۷، ۳۸]. بدیهی است چنانکه نمایه‌ساز و چکیده نویس مدارک شیمی، کاربرگه مناسبی که حاوی فیلدهای خاص ترکیبات شیمیایی است را در اختیار نداشته باشد، مستند سازی مدارک شیمی به شیوه مطلوبی صورت نخواهد پذیرفت و طبیعتاً بازیابی اطلاعات به نحو مناسب عملی نخواهد بود.

۳- مشکلات نرم افزاری و ضرورت طراحی نرم افزار جدید

در طرح پژوهشی پیشین به تفصیل مشکلات نرم افزار سی.دی.اس آیسس را در ثبت، نمایش و بازیابی علائم و ساختارهای شیمیایی برشمردیم [۱۸]. در ویرایش تحت ویندوز این نرم افزار یعنی وین آیسس نیز این مشکلات مرتفع نشده است [۹۸]. همچنین این مشکلات در ساختار بانک اطلاعات پژوهشگاه اطلاعات و مدارک علمی ایران که بر روی شبکه جهانی اینترنت منتشر می‌شود نیز وجود دارد.

برنامه نویسی به منظورافزایش فیلدهای اسامی ترکیبات شیمیایی، فرمولهای بسته شیمیایی و شماره‌های ثبت شیمیایی به عنوان فیلدهای قابل بازیابی در نرم افزار وین آیسس ضروری است.

ایجاد پایگاه اطلاعات شیمی و مهندسی شیمی در پایگاه اطلاعات جامع مرکز به صورتی که قابلیت جستجوی تخصصی با استفاده از نمایه‌نامه‌های فرمولی و اسامی ترکیبات شیمیایی را داشته باشد نیز اقدام مهمی است که باید انجام شود [۱۸].

فصل دوم

به سوي وضعیت مطلوب

۱-۱- گردآوری سمینارهای شیمی و مهندسی شیمی غایب در بانک پژوهشگاه

پس از مشخص شدن مجموعه سمینارهای شیمی و مهندسی شیمی موجود و غیرموجود در بانک جامع اطلاعات پژوهشگاه اطلاعات و مدارک علمی ایران، با ارسال نامه‌ای به دبیرخانه انجمن شیمی و مهندسی شیمی ایران درخواست مجموعه‌های غیرموجود را نمودیم. و انجمن پس از دریافت این درخواست مجموعه‌های زیر را برای پژوهشگاه ارسال نمود:

۱. سمینار شیمی تجزیه ایران، اولین، دومین، سومین، چهارمین، پنجمین و ششمین

۲. سمینار تخصصی شیمی آلی ایران، اولین و چهارمین

۳. سمینار شیمی فیزیک ایران، اولین و سومین

۴. کنگره بین‌المللی شیمی و مهندسی شیمی ایران، اولین و سومین

۵. کنگره انجمن شیمی و مهندسی شیمی ایران، سومین، هفتمین و دهمین

۶. کنگره ملی شیمی و مهندسی شیمی ایران، دوازدهمین

سپس این مجموعه‌ها به بخش مدیریت اسناد و اطلاعات منتقل گردید و مورد نمایه سازی و چکیده نویسی قرار گرفت و اطلاعات آن وارد بانک جامع اطلاعات پژوهشگاه گردید.

۱-۲- استخراج اطلاعات از بانک جامع پژوهشگاه و نوع فرمت آن

در این مرحله با انتخاب کلید واژه‌های مناسب به جستجوی کلیه سمینارهای شیمی و مهندسی شیمی موجود در بانک جامع اطلاعات پژوهشگاه (اطلاعات قبلی و اطلاعات اضافه شده) پرداختیم. این کلید واژه‌ها را در فیلد عنوان سمینارها جستجو کردیم. کلید واژه‌ها طوری انتخاب شدند تا در حد امکان همه سمینارهای محتمل موجود در رشته شیمی، مهندسی شیمی و گرایشهای وابسته به آن را پوشش دهند. کلید واژه‌ها عبارت بودند از:

شیمی؛ مهندسی شیمی؛ الکتروشیمی؛ خوردگی؛ بیوشیمی؛ شیمی آلی؛ شیمی فیزیک؛ شیمی تجزیه؛ شیمی معدنی؛ شیمی کاربردی؛ مهندسی متالورژی و مواد؛ مهندسی متالورژی؛ مهندسی مواد؛ مهندسی نفت؛ پلیمر؛ نفت؛ پتروشیمی؛ شیمی هسته‌ای؛ صنایع شیمیایی؛ نفت و گاز؛ نفت؛ گاز؛ پالایش؛ شیمی حیاتی؛ زیست شیمی؛ متالورژی؛ رنگ؛ طراحی فرایندها و سرمایه‌یک.

در مجموع ۳۷۰۲ رکورد بصورت زیر بدست آمد، و این اطلاعات با فرمت فارسی مایکروسافت ذخیره گردید به صورتی که قابل استفاده در نرم افزار وین‌آیسیس باشد.

عنوان سمینار	تعداد رکورد
سمپوزیم سراسری خوردگی در صنایع ایران، اولین	۱۶
سمپوزیم متالورژی ایران، اولین	۴۷
سمینار اثرات جنگهای شیمیایی بیولوژیک بر انسان، محیط زیست و جامعه	۱۷
سمینار ایمنی و صنعت نفت	۱۸
سمینار بهداشت و ایمنی محیط کار	۱۱
سمینار تخصصی شیمی آلی ایران، سومین	۵۴
سمینار تخصصی شیمی آلی ایران، ششمین	۱۰۱
سمینار تخصصی شیمی فیزیک ایران، اولین	۲۱
سمینار خودکفایی - تحقیقاتی صنعت سرامیک	۱۲
سمینار شیمی آلی ایران، اولین	۳۹
سمینار شیمی آلی ایران، چهارمین	۸۴
سمینار شیمی تجزیه ایران	۹۲
سمینار شیمی تجزیه ایران، پنجمین	۵۹
سمینار شیمی تجزیه ایران، چهارمین	۳۴
سمینار شیمی تجزیه ایران، دومین	۲۳
سمینار شیمی تجزیه ایران، سومین	۳۰
سمینار شیمی تجزیه ایران، ششمین	۶۷
سمینار شیمی فیزیک ایران، دومین	۳۱
سمینار شیمی فیزیک ایران، سومین	۴۴
سمینار علوم و تکنولوژی پلیمرها	۳۴
سمینار متالورژی	۲۴
سمینار متالورژی سطح و تریبولوژی ایران، اولین	۳۷
سمینار متالورژی فلزات غیر آهنی ایران، اولین	۵۹
سمینار نفت و گاز و پترو شیمی	۱۱
سمینار هسته خودکفایی - تحقیقاتی صنایع رنگ و رزین، اولین	۱
سمینار و کارگاه لیزر و کاربردهای آن در پلیمرها، اولین	۱۴

۱۱	کنفرانس آموزش شیمی ایران، اولین
۹۰	کنگره بین المللی پزشکی گازهای شیمیایی جنگی در ایران،
۱۸۵	کنگره بین المللی شیمی و مهندسی شیمی ایران، اولین
۲۴۷	کنگره بین المللی (دومین) و ملی (دوازدهمین) شیمی و مهندسی شیمی ایران
۲۶	کنگره پژوهشی نفت، گاز و پتروشیمی
۵۴	کنگره سالانه انجمن مهندسين متالورژی، اولین
۱۵۵	کنگره سراسری بیوشیمی جمهوری اسلامی ایران، اولین
۵۶	کنگره سراسری نفت و توسعه پایدار
۱۰۶۲	کنگره شیمی و مهندسی شیمی ایران، ۱، ۲، ۳، ۴، ۵، ۶، ۹، ۱۰، ۱۱ و ۱۳
۱۲۶	کنگره ملی مهندسی شیمی ایران، اولین
۲۲۵	کنگره ملی مهندسی شیمی ایران، دومین
۱۹۲	کنگره ملی مهندسی شیمی ایران، سومین
۳۴	گردهمایی بیوشیمی فیزیک ایران، چهارمین
۲۷	گردهمایی بیوشیمی فیزیک ایران، سومین
۲۶	همایش سالانه بررسی عوارض مزمن گازهای شیمیایی جنگی، ششمین
۱۹	همایش شیمی و صنایع شوینده، اولین
۱۱۳	همایش نفت، گاز و پتروشیمی، نهمین
۷۴	همایش نفت، گاز و پتروشیمی، هفتمین

۱-۳- دسته بندی اطلاعات

در پرسشنامه مصوب طرح پژوهش حاضر، باز پردازش تعداد ۲۷۰۰ رکورد از سمینارهای شیمی و مهندسی شیمی ایران پیش بینی شده بود. بر این اساس و نیز با توجه به عنوان این طرح پژوهشی یعنی "بهبود سازی اطلاعات علمی شیمی و مهندسی شیمی ایران (سمینارها)" سمینارهای صفحه بعد بر اساس گرایش های تخصصی انتخاب و بصورت زیر برای انجام عملیات باز پردازش تخصصی دسته بندی شدند، مجموع تعداد رکوردهای سمینارهای زیر ۲۷۱۵ رکورد می باشد:

عنوان سمینار	تعداد رکورد
سمینار شیمی آلی ایران، اولین	۳۹
سمینار شیمی آلی ایران، سومین	۵۴
سمینار شیمی آلی ایران، چهارمین	۸۴
سمینار شیمی آلی ایران، ششمین	۱۰۱
سمینار شیمی تجزیه ایران، اولین	۹۲
سمینار شیمی تجزیه ایران، دومین	۲۳
سمینار شیمی تجزیه ایران، سومین	۳۰
سمینار شیمی تجزیه ایران، چهارمین	۳۴
سمینار شیمی تجزیه ایران، پنجمین	۵۹
سمینار شیمی تجزیه ایران، ششمین	۶۷
سمینار شیمی فیزیک ایران، اولین	۲۱
سمینار شیمی فیزیک ایران، دومین	۳۱
سمینار شیمی فیزیک ایران، سومین	۴۴
کنگره شیمی و مهندسی شیمی ایران، اولین	۶۹
کنگره شیمی و مهندسی شیمی ایران، دومین	۴۳
کنگره شیمی و مهندسی شیمی ایران، سومین	۷۰
کنگره شیمی و مهندسی شیمی ایران، چهارمین	۱۳۸
کنگره شیمی و مهندسی شیمی ایران، پنجمین	۷۴
کنگره شیمی و مهندسی شیمی ایران، ششمین	۲۳
کنگره شیمی و مهندسی شیمی ایران، نهمین	۱۴۵
کنگره شیمی و مهندسی شیمی ایران، دهمین	۱۴
کنگره شیمی و مهندسی شیمی ایران، یازدهمین	۱۲۹
کنگره شیمی و مهندسی شیمی ایران، سیزدهمین	۳۵۷
کنگره ملی مهندسی شیمی ایران، اولین	۱۲۶
کنگره ملی مهندسی شیمی ایران، دومین	۲۲۵
کنگره ملی مهندسی شیمی ایران، سومین	۱۹۲
کنگره بین المللی شیمی و مهندسی شیمی ایران، اولین	۱۸۵

۲ - پردازش اطلاعات

- ۱-۲- ویرایش و نمایه سازی مجدد (تصحیح؛ حذف و افزایش کلیدواژه ها)
- ۲-۲- یکسان سازی و استاندارد سازی (عناوین؛ کلیدواژه ها و اسامی ترکیبات شیمیایی)
- ۳-۲- معادل یابی فارسی و لاتین
- ۴-۲- ارجاع دهی و استخراج کوتاه نوشته ها
- ۵-۲- استخراج اسامی ترکیبات شیمیایی
- ۶-۲- تصحیح اسامی ترکیبات شیمیایی و ورود اطلاعات
- ۷-۲- فرمول دهی به اسامی ترکیبات شیمیایی و ورود اطلاعات به ماشین
- ۸-۲- ایجاد نمایه نامه فرمولی
- ۹-۲- ایجاد نمایه نامه اسامی ترکیبات شیمیایی
- ۱۰-۲- یکسان سازی اطلاعات کلیه فیلدهای مدرک شناختی
- ۱۱-۲- تبدیل فرمت کلیه اطلاعات پردازش شده از سی.دی.اس به زرنگار یا وورد
- ۱۲-۲- صفحه آرایی و ویرایش نهایی

جزئیات موارد دوازده گانه فوق به تفصیل در گزارش طرح پژوهشی پیشین آمده است [۱۸]. در این طرح تمامی فرایندهای پردازش اطلاعات با استفاده از نرم افزار وین آیسس صورت گرفته است و این تفاوت اصلی موارد فوق در مقایسه با طرح پیشین است. در آنجا در محیط سی.دی.اس آیسس تحت داس کار می شد. بدیهی است کار کردن در محیط تحت ویندوز به مراتب راحتتر و برخوردار از قابلیت های بالاتری است [۹،۸].

۲-۱۳- اختصاص دادن شماره ثبت شیمیایی به اغلب ترکیبات شیمیایی و ساخت نمایه نامه

شماره های ثبت شیمیایی

به هر ماده شیمیایی شماره ثبت منحصر به فردی اختصاص داده می شود. برای آن که هر جسم به طور اختصاصی تعریف شود از یک الگوریتم رایانه ای، که ماهیت ماده شیمیایی را بر اساس توصیف روشنی از ساختار مولکولی آن می شناسد، استفاده می شود. مشخصات فضاشیمی و سایر اطلاعات شناساننده به سوابق رایانه ای ماده شیمیایی ضمیمه می شود. شماره ثبت به خودی خود حائز هیچ گونه اهمیت شیمیایی نیست، اما به عنوان یک برچسب یا عامل شناسایی منحصر به فرد ماده به کار می رود [۴،۷،۹].

به کلیه مواد خاصی که از سال ۱۹۶۵ در متون دست اول برای نمایه‌سازی در چکیده‌نامه‌های شیمی شناسایی شده‌اند، شماره ثبت اختصاص داده شده است. شماره‌های ثبت در هشتمین نمایه‌نامه جامع وارد شده‌اند و قسمتی از مدخل هر ماده در نمایه‌نامه مواد شیمیایی، نمایه‌نامه فرمولی و راهنمای نمایه‌ای در خلال نهمین دوره جامع را تشکیل می‌دهند. شماره‌های ثبت اداره چکیده‌نامه‌های شیمی در بسیاری از نشریات دست اول و نوشتار تلخیص شده نیز چاپ می‌شود [۴۹، ۴۸].

کتاب راهنمای ثبت - بخش عددی (CAS Registry Handbook-Number Section) فهرست مفصلی از چند میلیون ماده شیمیایی منحصر به فرد در هفت مجلد است. این کتاب راهنما نام نمایه‌ای و فرمول مولکولی هر یک از شماره‌های ثبت شده را به ترتیب صعودی شماره‌های ثبت، فهرست می‌کند. پیوستهای کتاب راهنمای ثبت - بخش عددی همین اطلاعات را برای تمام مواد شیمیایی که در خلال دوره‌های زمانی بعد به ثبت رسیده‌اند، فراهم می‌سازد [۴].

با در دست داشتن شماره ثبت یک ماده شیمیایی، می‌توان با استفاده از این کتاب راهنما و پیوستهای آن نام نمایه‌ای و فرمول مولکولی را بدست آورد. نام نمایه‌ای و فرمول مولکولی، نقاط مدخلی به نمایه‌نامه‌های مواد شیمیایی و نمایه‌نامه فرمولی می‌باشند و این نمایه‌نامه‌ها شخص را به چکیده‌های سندهای حاوی اطلاعات در مورد جسم ارجاع می‌دهند [۴].

شماره ثبت انحصاری، به ویژه در نمایه‌نامه‌های فرمولی، رابط نامهای سیستماتیک متفاوتی است که در دوره‌های مختلف نمایه‌نامه‌های جامع به کار می‌روند. نام مواد را می‌توان از طریق شماره‌های ثبت و بدون درک پیچیدگیهای موجود در نامگذاری شیمیایی مواد، معین کرد. با استفاده از امکانات نرم افزار وین آیسس، فیلد مخصوصی برای شماره‌های ثبت شیمیایی تعبیه شد و به بسیاری از ترکیبات شیمیایی موجود در رکوردها شماره‌های ثبت شیمیایی اختصاص داده شد. این شماره‌های ثبت شیمیایی با جستجوی این ترکیبات در وب سایتهای معروف شیمی بدست آمد [۳۷، ۳۸].

۳ - طراحی کاربرگه اطلاعات شیمی

۳-۱- فیلدهای مورد نیاز و قابل بازیابی در کاربرگه

همانطوری که در گزارش طرح پژوهشی پیشین آورده‌ایم برای ثبت اطلاعات علمی مدارک شیمی به فیلدهای بیشتری نیاز داریم تا مستند سازی مدارک علمی شیمی به روش استاندارد صورت پذیرد [۱۸].

هر ماده شیمیایی را می‌توان علاوه بر نمایش آن توسط نام ترکیب، با فرمول ساختاری، شماره ثبت شیمیایی، خواص فیزیکوشیمیایی، خواص طیفی و خواص تقارنی نیز معرفی نمود. از آنجا که متون شیمی بخصوص در حوزه شیمی آلی و شیمی معدنی عموماً حاوی این گونه نمایش‌ها هستند، پایگاه‌های اطلاعاتی شیمی، نیز باید قابلیت ثبت، نمایش و بازیابی این نوع نمایش‌ها را داشته باشند [۴۹، ۱۸].

در پایگاه‌های اطلاعاتی معروف شیمی علاوه بر اینکه انواع و اقسام نمایشهای شیمیایی قابل مشاهده هستند امکان بازیابی اطلاعات بر اساس آن نمایشها وجود دارد [۳۷، ۳۸].

بدیهی است چنانکه نمایه‌ساز و چکیده‌نویس مدارک شیمی، کاربرگه مناسبی که حاوی فیلدهای خاص ترکیبات شیمیایی است را در اختیار نداشته باشد، مستندسازی مدارک شیمی به شیوه مطلوبی صورت نخواهد پذیرفت و طبیعتاً بازیابی اطلاعات به نحو مناسب عملی نخواهد بود.

همانطوری که گفته شد در مستندسازی مدارک علمی شیمی به غیر از سه فیلد عنوان، چکیده و کلیدواژه، فیلدهای دیگری نیز مورد نیاز است. پس از بررسی چندین پایگاه اطلاعاتی بر روی اینترنت و چند دستینه (handbook) و کتب مرجع شیمی و نیز بر اساس تجربیات بدست آمده در طرح پژوهشی پیشین و همچنین بررسی چند نرم افزار معروف و پرکاربرد شیمی مانند: آیسس دراو (isisdraw) و کم اسکچ (chemsketch)، فیلدهایی را انتخاب نمودیم تا بر اساس این فیلدها در بخشهای بعدی طرح که شامل طراحی پایگاه ترکیبات شیمیایی و نیز طراحی نرم افزار ویژه ثبت و ذخیره و بازیابی اطلاعات علمی شیمی است، از آنها بعنوان فیلدهای استاندارد مستندسازی مدارک شیمی و رشته‌های وابسته به آن استفاده شود.

علاوه بر نمایه‌سازان که بصورت ماشینی نیز می‌توانند از این کاربرگه‌ها استفاده کنند، محققان و رساله‌نویسان نیز با در اختیار داشتن چنین کاربرگه‌هایی می‌توانند نقش بزرگی در استاندارد سازی و نهایتاً در بازیابی کارای اطلاعات علمی شیمی ایفا نمایند. در ادامه گزارش نرم افزار طراحی شده را معرفی خواهیم نمود که امکان ثبت و نمایش ساختارها و علائم مورد استفاده در متون شیمی را برای ما فراهم می‌کند ضمن اینکه از آن برای ایجاد پایگاه تخصصی اطلاعات شیمی نیز می‌توان استفاده نمود.

Chemical Compound Name	نام ترکیب شیمیایی (تکرار پذیر)
IUPAC Name	نام ایوپاک
Systematic Name	نام سیستماتیک
Trivial (common) Names	نامهای معمولی
1	1
2	2
3	3
Trade (commercial) Names	نامهای تجاری
1	1
2	2
3	3

Molecular Formula	فرمول مولکولی (تکرار پذیر)

Chemical Registry Number	شماره ثبت شیمیایی (تکرار پذیر)

Structural Formula

فرمول ساختاری (تکرار پذیر)

Physicochemical

خواص فیزیکوشیمیایی (تکرار پذیر)

Properties

Boiling Point	نقطه جوش
Composition	ترکیب در صد
Color	رنگ
Crystalline form	شکل بلوری
Density	چگالی
Dielectric Constant	ثابت دی الکتریک
Dipole moment	ممان دوقطبی
Evaporation Rate	سرعت تبخیر
Flash Point	نقطه اشتعال
Refractive Index	ضریب شکست
Melting Point	نقطه ذوب
Molecular Weight	وزن مولکولی
Molar Refractivity	قابلیت شکست مولی
Molar Volume	حجم مولی
Odor	بو
Pka	پی . ا . ک
Physical state	حالت فیزیکی
Polarizability	قطبش پذیری
Surface Tension	کشش سطحی

Thermodynamic Standard Properties

$$\Delta H_f^\circ = \quad S^\circ =$$

$$\Delta G_f^\circ = \quad C_p^\circ =$$

Vapor Density	خواص ترمودینامیکی استاندارد
Vapor Pressure	چگالی بخار
Viscosity	فشار بخار
Water Solubility	گرانروی
	حلالیت در آب

Spectral Properties

خواص طیفی (تکرار پذیر)

Electronic

ESR

Infrared

Microwave

Mossbauer

Nuclear Magnetic Resonance

Photoelectron

Raman

Ultraviolet Visible

X-RAY Diffraction

MASS

Symmetric Properties

خواص تقارنی (تکرار پذیر)

Point group:

Symmetry Elements are:

Comment and further Information

توضیح و اطلاعات بیشتر (تکرار پذیر)

۲- ۳- نمونه کاربرگه (فرم پر شده)

Chemical Compound Name

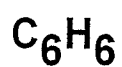
نام ترکیب شیمیایی (تکرار پذیر)

IUPAC Name	benzene
Systematic Name	cyclohexatriene
Trivial (common) Names	
1	phene
2	benzol
3	annulene
Trade (commercial) Names	
1	benzene
2	benzol
3	annulene

بنزن	نام ایوپاک
سیکلو هگزا تریان	نام سیستماتیک
	نامهای معمولی
	1 فن
	2 بنزول
	3 آنولن
	نامهای تجاری
	1 بنزن
	2 بنزول
	3 آنولن

Molecular Formula

فرمول مولکولی (تکرار پذیر)



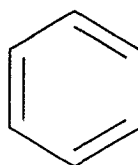
Chemical Registry Number

شماره ثبت شیمیایی (تکرار پذیر)

[71-43-2]

Structural Formula

فرمول ساختاری (تکرار پذیر)



Physicochemical

خواص فیزیکی و شیمیایی (تکرار پذیر)

Properties

Boiling Point	80.1
Composition	C(92/26%)H(7/74%)
Color	Colorless liquid
Crystalline form	-
Density	0/873 +_ 0/06 g/cm3
Dielectric Constant	2/36 +_ 0/1
Dipole moment	-
Evaporation Rate	-
Flash Point	-11
Refractive Index	1/498 +_ 0/02
Melting Point	5/5
Molecular Weight	78/112
Molar Refractivity	26/25 +_ 0/3 cm3
Molar Volume	89/4 +_ 3 cm3
Odor	paint-thinner-like
Pka	
Physical state	liquid
Polarizability	10/40 +_ 0/5exp(-24) cm3
Surface Tension	28/8 +_ 0/3 dyne/cm
Thermodynamic Standard Properties	

$$\Delta H_f^0 = 11/71 \quad S^0 = 41/41$$

$$\Delta G_f^0 = 29/72 \quad C_p^0 = 19/52$$

Vapor Density	2/77
Vapor Pressure	12.7 kPa (25 C)
Viscosity	-

نقطه جوش
ترکیب در صد
رنگ
شکل بلوری
چگالی
ثابت دی الکتریک
ممان دو قطبی
سرعت تبخیر
نقطه اشتعال
ضریب شکست
نقطه ذوب
وزن مولکولی
قابلیت شکست مولی
حجم مولی
بو
پی . ک . ا
حالت فیزیکی
قطبش پذیری
کشش سطحی

خواص ترمودینامیکی استاندارد

چگالی بخار

فشار بخار

گرانروی

Spectral Properties

خواص طیفی (تکرار پذیر)

Electronic

ESR

Infrared

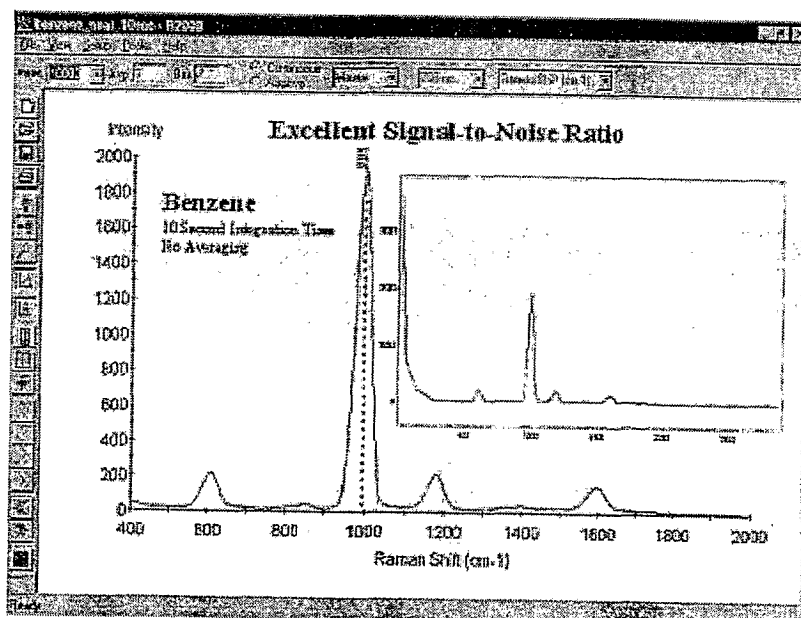
Microwave

Mossbauer

Nuclear Magnetic Resonance

Photoelectron

Raman



Ultraviolet Visible

X-RAY Diffraction

MASS

Symmetric Properties

خواص تقارنی (تکرار پذیر)

Point group: **D_{6h}**

Symmetry Elements are:

One i , Center of inversion

One C_6 rotation axis perpendicular to the molecular plane

One C_3 rotation axis perpendicular to the molecular plane.

One C_2 rotation axis perpendicular to the molecular plane.

Three C_2' rotation axes in the molecular plane.

Three C_2'' rotation axes in the molecular plane.

One σ_h reflection plane corresponding to the molecular plane.

Three σ_v reflection planes containing the principal axis (C_6).

Three σ_d reflection planes containing the principal axis (C_6)

One S_6 improper axis coincident with the C_6 axis.

One S_3 improper axis coincident with the C_3 axis.

Comment and further Information

توضیح و اطلاعات بیشتر (تکرار پذیر)

Benzene is an aromatic hydrocarbon that is produced by the burning of natural products. It is also a component of products derived from coal and petroleum. It is found in gasoline and other fuels, and is used in the manufacture of plastics, detergents, pesticides, and other chemicals.

Research has shown benzene to be a carcinogen (cancer causing). With exposures from less than 5 years to more than 30 years, individuals have developed, and died from, leukemia. Long-term exposure may affect bone marrow and blood production.

Short-term exposure to high levels of benzene can cause drowsiness, dizziness, unconsciousness, and death. The current permissible exposure level is 1 part per million (ppm) in air for an 8 hour average with a short-term exposure limit of 5 ppm. Benzene can also be absorbed through the skin.

۴- طراحی نرم افزار بانک ترکیبات شیمیایی

۴-۱- مرور ادبیات نرم افزاری

در طی مراحل انجام طرح ضمن بحثهای کارشناسی با حضور مجری، ناظر محترم و متخصصین بخش رایانه تصمیم گرفته شد که پایگاه اطلاعات جدیدی به پایگاههای موجود افزوده نشود و فقط امکان ذخیره و بازیابی با استفاده از فرمولهای مولکولی و اسامی ترکیبات شیمیایی به بانکهای موجود داده شود که البته این مطلب بصورت اصلاحیه در شورای پژوهش پژوهشگاه مطرح و مورد تصویب واقع گردید. و بدین ترتیب بحث ایجاد نرم افزار بانک ترکیبات شیمیایی بصورتی که قابلیت صادرات و واردات اطلاعات با بانکهای پژوهشگاه را داشته باشد مطرح گردید که به این مساله در ادامه خواهیم پرداخت.

در طرح پژوهشی پیشین به تفصیل مشکلات نرم افزار اسی.دی.اس آیسس در ثبت، نمایش و بازیابی علائم و ساختارهای شیمیایی را برشمردیم. این نرم افزار قادر به نمایش ساختارهای شیمیایی و علائم مورد استفاده در متون شیمی نمی باشد. در ویرایش تحت ویندوز این نرم افزار یعنی وین آیسس نیز این مشکلات مرتفع نشده است. همانطوری که قبلاً اشاره نمودیم در متون علمی شیمی (در اکثرگرایشها) علائم فراوانی بکار می رود مانند: علائم مربوط به نمایش واکنشهای شیمیایی، علائم ریاضی، حروف یونانی، نمایش ساختارهای گسترده شیمیایی و ... بنا بر این بدیهی است نرم افزارهایی که برای نمایه سازی مدارک شیمی و ایجاد پایگاه های اطلاعاتی شیمی بکار میروند باید قابلیت ورود، نمایش، ذخیره و بازیابی این علائم و اشکال را داشته باشند.

در گزارش طرح پژوهشی پیشین، بصورت مشروح به مشکلات عدیده موجود در بازیابی اطلاعات علمی شیمی و مهندسی شیمی در بانک جامع اطلاعات پژوهشگاه اشاره نمودیم و ویژگیهای خاص اصطلاحات شیمی و اسامی ترکیبات شیمیایی و نحوه رفتار با آنها را در ایجاد پایگاه ها و نمایه نامه های تخصصی شیمی بر شمرده ایم [۱۸]. در آنجا اشاره کرده ایم که عدم وجود کاربرگه های استاندارد و متناسب با اطلاعات علمی شیمی یکی از مشکلات اساسی در ایجاد پایگاه های اطلاعاتی شیمی و رشته های وابسته به آن است و همانطوری که ملاحظه شد در این طرح کاربرگه های جدیدی طراحی و پیشنهاد گردید. کاربرگه های طراحی شده، به نمایه سازان و محققان در ثبت استاندارد اطلاعات کمک شایانی خواهند نمود و نیز پردازش تخصصی اطلاعات را بصورت سهلتری عملی خواهند کرد و در واقع گامی در جهت عملیاتی شدن فرآیند پردازش ماشینی اطلاعات علمی شیمی و مهندسی شیمی خواهند بود. این کاربرگه ها به نمایه ساز و ماشین نویس امکان ورود اطلاعات به شکل تخصصی و تفکیک شده را می دهد و همچنین کاربر را قادر می سازد با استفاده از نمایه نامه های مورد نظر خود به جستجوی اطلاعات بپردازد. کاربرگه های

طراحی شده که گزارش روند تهیه آنها در بخش ۳ آمد به متخصص برنامه‌نویسی رایانه‌ای تحویل شد و کلیه فیلدهای موجود در کاربرگه در یک نرم افزار بانک ترکیبات شیمیایی منعکس گردید که در ادامه به توضیح آن خواهیم پرداخت.

نرم افزارهای متعددی برای رسم و نمایش ساختارها و علائم و حتی محاسبات ساده و پیچیده شیمیایی وجود دارد. بسیاری از این نرم افزارها بصورت رایگان بر روی سایتهای اینترنتی شیمی وجود دارند. سه تا از معروف ترین آنها عبارتند از:

1. Chem.sketch
2. Isis.Draw
3. Chem.Office (Chem.Draw)

کم/اسکیچ نرم افزاری بسیار قوی با قدرت گرافیک بالا است. توسط این نرم افزار به راحتی و با قدرت مانور بسیار بالا می‌توان هر نوع ساختار شیمیایی^۴ را ترسیم نمود و نمایش‌های سه بعدی را نیز مشاهده کرد. با این نرم افزار به دوشکل وصل خط (online) و برون خط (offline) می‌توان کار کرد. در حالت وصل خط با اتصال به پایگاه‌های اطلاعاتی موجود بر روی سایت اینترنتی <http://www.acdlabs.com> قابلیت‌های نرم افزار افزایش می‌یابد. این نرم افزار بسیاری از ساختارهای شیمیایی را به صورت آماده در اختیار می‌گذارد، همچنین گروه‌های عاملی و جدول تناوبی بسیار کارآیی را ارائه می‌کند. از این نرم افزار می‌توان در نمایه سازی مدارک شیمی استفاده‌های فراوانی برد. اطلاعات مربوط به بسیاری از فیلدهای پیش‌بینی شده در کاربرگه‌های جدید توسط این نرم افزار قابل استحصال است. در حال حاضر ویرایش هشتم، آخرین ویرایش این نرم افزار می‌باشد.

آیسیس‌درا که نرم افزار شناخته شده‌تر و قدیمی‌تری است نیز کارآیی‌های تقریباً مشابه ولی در سطح پایین‌تری دارد که فقط بصورت برون خط کار می‌کند. با نصب یک نوع plug in به نام Autonomomenclature بر روی آن، این نرم‌افزار نام آیوپاک ساختارهای ترسیم شده را بدست می‌دهد.

نرم افزار Chem.Draw از مجموعه Chem.Office نیز تواناییهای مشابهی با نرم‌افزارهای فوق دارد. کم‌درا طوری طراحی شده است که با دو نرم افزار دیگر از مجموعه کم آفیس یعنی Chem.finder و Chem.3D بصورت هماهنگ عمل می‌کند. این نرم افزار توسط سایت کمبریج پشتیبانی می‌شود.

به غیر از نرم افزارهای فوق، یک plug in به نام Chemi وجود دارد که به مرورگرهای اینترنت اکسپلور و نت اسکپ امکان می‌دهد تا بتوانند ساختارهای سه بعدی شیمیایی را نمایش دهند و حتی نام آیوپاک ترکیبات شیمیایی را بصورت وصل خط بدست آورند.

آنچه که در بالا ذکر گردید مروری کوتاه بر برخی از نرم افزارهای شیمی بود که هیچ کدام آنها قابلیت‌های یک نرم افزار پایگاه اطلاعاتی مانند وین‌آیسیس را ارائه نمی‌کنند و از طرفی نرم‌افزارهای بانک اطلاعاتی مانند وین‌آیسیس از امکانات مورد نیاز برای نمایه کردن مدارک شیمی برخوردار نیستند. از اینرو در این طرح پژوهشی تصمیم گرفته شد تا نرم افزاری طراحی شود تا هر دو قابلیت نرم افزارهای شیمی و بانک‌های اطلاعاتی را توأمان داشته باشد.

۴-۲- طراحی بخش الگوهای شیمیایی

این بخش در حقیقت می‌تواند به عنوان یک نرم افزار کامل گرافیکی برای رسم اشکال شیمیایی بکار رود. در ترسیم الگوهای شیمیایی، این نرم افزار از قابلیت‌های زیر برخوردار است:

الف - انواع اشکال

- خط ساده
- خط شکسته
- کمان گذرنده از سه نقطه
- چهارگوش
- دایره
- چند ضلعی (از سه تا نه ضلعی) به صورت آماده و بیشتر از آن با رسم ترکیبی
- پیوند خارج از صفحه رو به بیرون
- پیوند خارج از صفحه رو به داخل
- فلش یک طرفه ساده
- فلش کمانی یک طرفه ساده
- فلش دو طرفه ساده
- فلش کمانی دو طرفه ساده
- فلش یک طرفه دوبر
- پیوند دو گانه
- پیوند سه گانه
- زنجیره کربنی
- نوشتن فرمول‌ها و علائم شیمیایی و متن ساده

ب - امکانات بخش طرحی اشکال :

- انتقال اشیا
- چرخش اشیا
- حذف اشیا
- تبدیل پیوندهای ساده ، دو گانه و سه گانه به یکدیگر

۴-۳- طراحی بخش ذخیره سازی اطلاعات

این نرم افزار در بخش ذخیره سازی ، نمایش و بازیابی اطلاعات از قابلیت های کلی زیر برخوردار می باشد:

قابلیت های کلی:

- ساختار اصلی فایل ذخیره سازی binary tree می باشد.
- سرعت بسیار زیاد برای جستجوی اطلاعات بر اساس کلید رکورد.
- قابلیت ذخیره سازی اطلاعات غیر اسکی از جمله تصاویر و الگوهای شیمیایی.
- قابلیت دسته بندی اطلاعات با استفاده از خاصیت Tree subtree در فایل ذخیره سازی بدون نیاز به index.
- قابلیت ایجاد ایندکس های خود کار.
- دارا بودن حدود ۷۰ فیلد با دسته بندی های زیر
 - مشخصات اصلی
 - فرمول ساختاری

اطلاعات ذرات می باشد
قسمت ذرات

الف - امکان طراحی فرمول با استفاده از قسمت طراحی

ب - امکان نشان دادن تصویر فرمول که از قبل بصورت بیت مپ درآمده است

- خواص فیزیکی شیمیایی
- خواص طیفی
- با این قابلیت که تعداد نامحدودی طیف را می تواند نمایش دهد
 - خواص تقارنی
 - توضیحات بیشتر
- طراحی نرم افزار به گونه ای است که کلیه قسمت های ایجاد رکورد جدید ، ویرایش رکورد ، نمایش نتایج جستجو و نمایش محتوای رکوردها به ساده ترین صورت ممکن همگی در یک قسمت انجام می شوند.

- برای قسمتهایی که می‌توانند تصویر قبول کنند، تصویرهایی با فرمت‌های بی.ام.پی و جی.پی قابل دریافت هستند.
- از ویژگی‌های این نرم افزار این است که کل اطلاعات بانک، همگی در یک فایل می‌باشند و بنابراین هیچگونه پراکندگی ایجاد نخواهد شد و حداقل تا ۲ گیگا بایت اطلاعات نیز قابل ذخیره سازی می‌باشد.

جستجوی اطلاعات:

جستجوی اطلاعات بر اساس فیلدهای زیر صورت می‌گیرد:

- نام آیوپاک
- فرمول مولکولی
- شماره ثبت
- نامهای معمولی و تجاری
- نقطه جوش
- چگالی
- شماره دات
- نقطه اشتعال
- نقطه ذوب
- حالت فیزیکی
- چگالی بخار
- گروه نقطه‌ای
- امکان جستجو در کل بانک
- امکان جستجو در رکوردهای انتخاب شده
- امکان جستجو در نتایج جستجوی قبلی

ذخیره اطلاعات:

- امکان ذخیره کل بانک با یک نام جدید
- امکان ذخیره رکوردهای انتخاب شده با یک نام جدید
- امکان ذخیره حاصل نتایج جستجو با یک نام جدید

خروجی:

- خروجی فیلدهای غیر تصویری بصورت ایزو و بر اساس تگ های مارک

نمایش لیست دسترسی بر اساس:

- نام آیوپاک
- نام معمولی
- نام تجاری
- فرمول مولکولی
- شماره ثبت شیمیایی

۴-۴- آزمایش نهایی بر روی نرم افزار مورد نظر و رفع مشکلات احتمالی

با ورود اطلاعات مربوط به تعدادی از ترکیبات شیمیایی، قابلیت‌های مختلف این نرم افزار با موفقیت مورد آزمایش قرار گرفت. در ادامه تصاویری از محیط نرم افزار ارائه می‌شود و راهنمای کاربران نرم افزار متعاقباً ارائه خواهد شد.

فایل نمایش جستجو اینترنت

نامک اطلاعات ترکیبات شیمیایی - [TEST.CC] نرم افزار تست شیمی طرح شیمی ۲

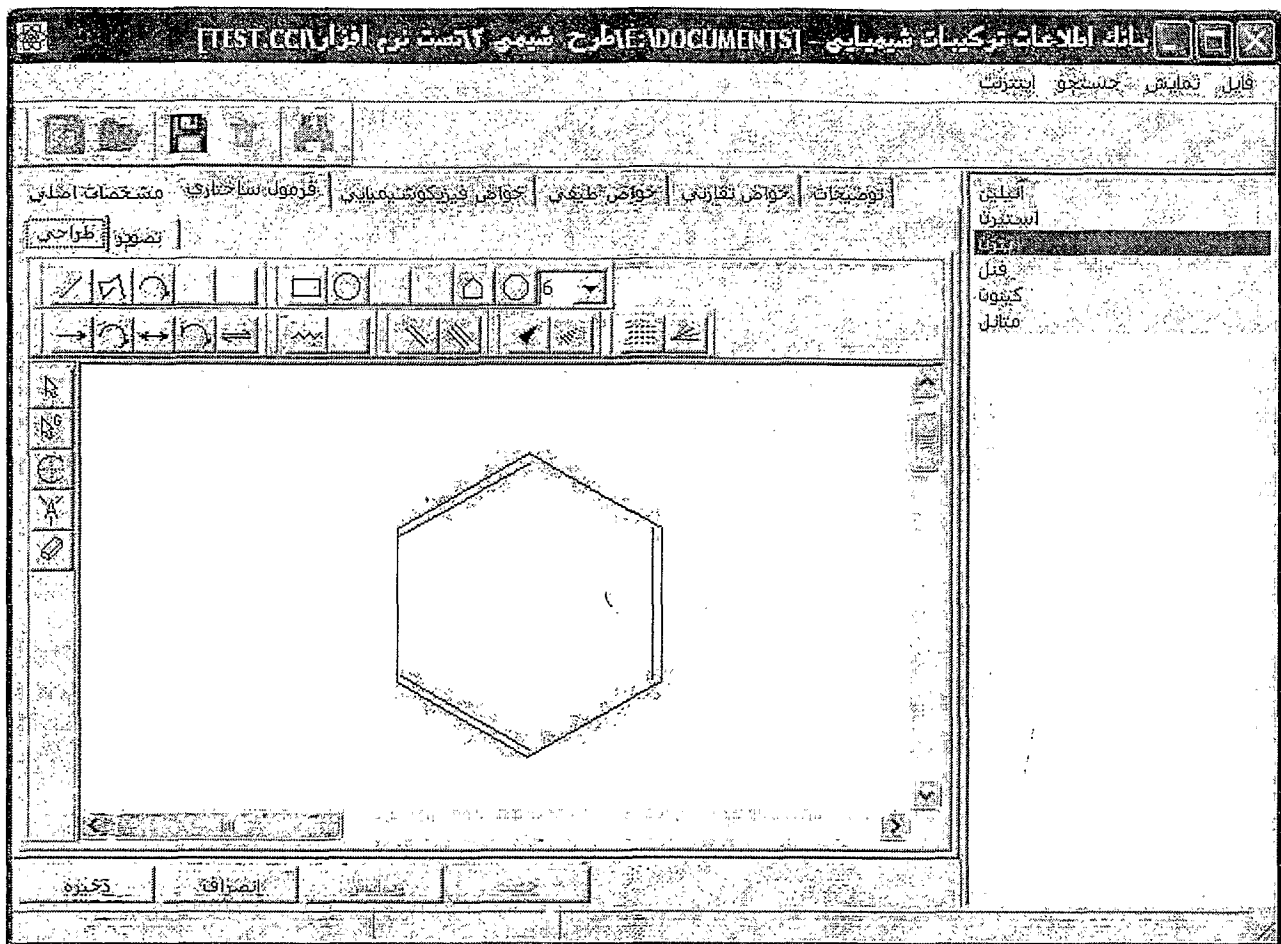
توضیحات | خواص تقارنی | خواص طبیعی | خواص فیزیکوشیمیایی | فرمول ساختاری | مشخصات اصلی

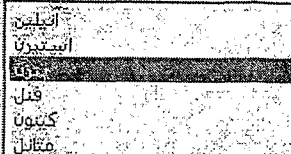
بنزن نام آیوپاک
Benzene ترجمه نام آیوپاک
C₆H₆ فرمول مولکولی
۲-۲۲-۷۱ شماره ثبت شیمیایی
۷۸,۱۱۲۲-۰۲۲۲-۰۲۱۲۵ وزن مولکولی

نام‌های تجاری

نام	ترجمه	نام	ترجمه
بن	بن	بن	بن
annulene(۶)	(۶) آنولن		
سیکلو هگزاتریان			

ذخیره | انصراف | ریزایش | حذف





فصل سوم

اهداف، خلاصه و نتیجه گیری

۱ - اهداف و خلاصه پژوهش:

۱-۱- آموزش نیروی انسانی (دو نفر)

- ۱-۱-۱- کسب مهارت‌های تخصصی در نمایه‌سازی شیمی.
- ۱-۱-۲- آشنایی با اصطلاحنامه‌های تخصصی شیمی (چاپی، الکترونیکی).
- ۱-۱-۳- استفاده از اینترنت (بانک‌های اطلاعاتی شیمی و سایت‌های تخصصی شیمی) در نمایه‌سازی متون شیمی.

۱-۱-۴- آشنایی با نرم افزار وین‌آیسیس.

۱-۱-۴-۱- ایجاد بانک، ورک شیت و فیلدهای مورد نظر.

۱-۱-۴-۲- پردازش و ویرایش اطلاعات.

۱-۱-۴-۲-۱- بر روی هر رکورد.

۱-۱-۴-۲-۲- از طریق ساخت نمایه‌نامه.

۱-۱-۴-۳- ایجاد نمایه‌نامه‌های تخصصی.

- ۱-۱-۵- آشنایی با نرم‌افزارهای تخصصی شیمی مانند آیسیس‌درا و کم‌اسکیچ و فرهنگ‌های الکترونیک شیمی.

۲-۱- طراحی کاربرگه استاندارد ورود اطلاعات ترکیبات شیمیایی

۱-۲-۱- انتخاب فیلدهای موجود در بانک‌های اطلاعاتی ترکیبات شیمیایی از

مراجع معتبر و مختلف و چینش آنها بر حسب اولویت‌های مطرح شده در طرح.

۱-۲-۲- استفاده از این کاربرگه در طراحی نرم افزار بانک ترکیبات شیمیایی.

۳-۱- طراحی بانک نرم افزار ترکیبات شیمیایی

۱-۳-۱- ورود اطلاعات بر حسب فیلدهای تعریف شده در کاربرگه به نرم

افزار (خواص فیزیکوشیمیایی، خواص طیفی، خواص تقارنی و...).

۱-۳-۲- امکان طراحی ساختارهای گسترده ترکیبات شیمیایی و نیز وارد کردن

تصویری از ساختار ترکیب مورد نظر.

۱-۳-۳- امکان جستجوی بسیار توانمند اطلاعات.

۱-۳-۴- امکان ذخیره اطلاعات با فرمت ایزو.

۱-۳-۵- استفاده از نرم افزار بعنوان ابزاری تخصصی در نمایه‌سازی مدارک شیمی

همچون یک دائرةالمعارف الکترونیک شیمی که تدریجاً تکمیل می‌شود و در

هنگام مراجعه مجدد از دوباره‌کاری جلوگیری می‌کند.

۱-۳-۶- امکان الگوبرداری از نرم افزار در طراحی بانک اطلاعات شیمیایی در وبسایت پژوهشگاه اطلاعات و مدارک علمی ایران.

۱-۳-۷- امکان استفاده از نرم افزار به منظور دریافت اطلاعات از راه دور با فرمتی استاندارد.

۱-۳-۸- امکان ارتباط با بانکهای شیمیایی اینترنتی با استفاده از فیلدهای مشترک استاندارد مانند شماره‌های ثبت شیمیایی.

۱-۴- چاپ کتاب سمینارهای شیمی و مهندسی شیمی ایران و ارائه گزارش نهایی طرح

۱-۴-۱- پردازش اطلاعات

۱-۴-۱-۱- ویرایش و نمایه‌سازی مجدد (تصحیح؛ حذف و افزایش کلیدواژه‌ها)

۱-۴-۱-۲- یکسان‌سازی و استاندارد سازی (عناوین؛ کلیدواژه‌ها و اسامی ترکیبات شیمیایی)

۱-۴-۱-۲- معادل‌یابی فارسی و لاتین

۱-۴-۱-۳- تهیه ارجاع و استخراج کوتاه‌نوشته‌ها

۱-۴-۱-۴- استخراج اسامی ترکیبات شیمیایی

۱-۴-۱-۵- تصحیح اسامی ترکیبات شیمیایی و ورود اطلاعات

۱-۴-۱-۶- فرمول دهی به اسامی ترکیبات شیمیایی و ورود اطلاعات به ماشین

۱-۴-۱-۷- ایجاد نمایه‌نامه فرمولی

۱-۴-۱-۸- ایجاد نمایه‌نامه اسامی ترکیبات شیمیایی

۱-۴-۱-۹- اختصاص دادن شماره ثبت شیمیایی به اغلب ترکیبات شیمیایی و ساخت نمایه‌نامه شماره‌های ثبت شیمیایی

۱-۴-۱-۱۰- یکسان سازی اطلاعات کلیه فیلدهای مدرک شناختی

۱-۴-۲- چاپ کتاب سمینارهای شیمی ایران

۱-۴-۲-۱- تبدیل فرمت کلیه اطلاعات پردازش شده از سی.دی.اس به

زرنگار یا ورود

۱-۴-۲-۲- صفحه آرایی و ویرایش نهایی

۲- نتایج پژوهش:

- ۱-۲- نمایه‌سازی مدارک شیمی را متخصصین موضوعی آشنا به مباحث اطلاع‌رسانی و نمایه‌سازی می‌توانند انجام دهند. به این منظور دو نفر کارشناس شیمی مورد آموزش قرار گرفتند.
 - ۲-۲- کاربرگه‌های ثبت اطلاعات موجود برای انواع مدارک در رشته شیمی از فیلدهای لازم و کافی برخوردار نیست و کاربرگه‌های جدید طراحی شد.
 - ۳-۲- برای نمایه‌سازی متون شیمی نرم افزار سی‌دی‌اس / آی‌سیس تحت داس و یا ویندوز کافی نمی‌باشد و نیاز به تقویت شدن دارند و به این منظور نرم افزاری تهیه گردید.
 - ۴-۲- سازگاری بین نرم افزار وین‌آسیس و یا سی‌دی‌اس / آی‌سیس با نرم افزار مایکروسافت وورد از نظر استاندارد بودن خروجی‌ها ضروری است.
 - ۵-۲- جهت بازیابی اطلاعات در یک بانک اطلاعات شیمی، نمایه‌نامه‌های موجود کافی نمی‌باشند و نمایه‌نامه‌های تخصصی شیمی مانند نمایه‌نامه فرمولی و نمایه‌نامه شماره‌های ثبت ترکیبات شیمیایی ایجاد شدند.
 - ۶-۲- ایجاد نمایه‌نامه‌های تخصصی فوق نیاز به طراحی برنامه‌های کامپیوتری، محیطهای نرم‌افزاری شیمی، نیروهای متخصص شیمی، واژگان‌ها و اصطلاحنامه‌های تخصصی شیمی و مهندسی شیمی و ارتباط با مراکز بزرگ نمایه‌سازی در دنیا مانند Chemical Abstract دارد.
 - ۷-۲- بدلیل وجود نمایه‌نامه‌های تخصصی مانند نمایه‌نامه سیستم‌های حلقه‌ای، نمایه‌نامه فرمولی و نمایه‌نامه اسامی ترکیبات شیمیایی، روش بازیافت اطلاعات شیمی روشی خاص است و این ویژگی ضرورت ایجاد بانک تخصصی موضوعی شیمی را دو چندان می‌کند.
 - ۸-۲- نمایه‌نامه‌های ایجاد شده توسط متخصصین غیرموضوعی به لحاظ حرف‌نگاری، معادل‌گذاری، ارجاع‌دهی، ارزش کلیدواژه‌ای و صحت نامگذاری به شدت مورد تردید است.
 - ۹-۲- تغییر در نظام نمایه‌سازی و گردش کار در بخش مدیریت اسناد و اطلاعات به منظور مرتفع ساختن معضلات مطرح شده در این پژوهش ضروری است.
- ## ۳- مشکلات پژوهش:

- ۱-۳- کامل نبودن مجموعه سمینارهای شیمی و مهندسی شیمی در بانک جامع پژوهشگاه لازم است در پوشش زمانی مورد نظر یک بررسی آماری به لحاظ کمیت گردآوری سمینارهای شیمی و مهندسی شیمی از دانشگاه‌ها و سایر مراکز مرتبط در سراسر کشور صورت گیرد تا مشخص شود مجموعه سمینارهای شیمی و مهندسی شیمی گردآوری شده در پژوهشگاه چند درصد کل سمینارهای شیمی و مهندسی شیمی برگزار شده در کشور را تشکیل می‌دهد.

۳-۲- عدم همخوانی کامل بین برنامه مایکروسافت ورد و نرم افزار وین آیسس بدلیل عدم وجود سازگاری کامل بین برنامه‌های مایکروسافت ورد و نرم افزار وین آیسس به هنگام تبدیل اطلاعات پردازش شده از محیط وین آیسس به محیط مایکروسافت ورد، نیاز به استانداردسازی وجود خواهد داشت که در روند اجرای کار وقفه بزرگی پدیده آورده از دقت و صحت کار می‌کاهد. بنابراین به نظر می‌رسد حل این مشکل از طرف متخصصان کامپیوتر ضروری است.

۳-۳- افزایش رکوردها و لزوم روزآمد کردن مجموعه اطلاعات هم اکنون حدود ۲۷۱۴ رکورد از سمینارهای شیمی و مهندسی شیمی در یک مجموعه واحد با ویژگیهایی که شرح آن در فصل دوم آورده شد آماده چاپ شده است، در گذشته نیز حدود ۱۳۰۰ پایان‌نامه شیمی بصورت یک مجموعه تک جلدی به چاپ رسید و قرار است در آینده مجموعه‌های مشابهی از طرح‌های پژوهشی شیمی و مهندسی شیمی و گزارش های دولتی شیمی ایران بصورت مجلدهای جداگانه به چاپ برسند. در سالهای آینده هر یک از این چهار مجلد به دلیل افزایش تعداد رکوردها، نیاز به روزآمد شدن دارند و شماره‌های بعدی این مجلدها نیز می‌بایست به چاپ رسند، لذا حفظ و نگهداری این اطلاعات پردازش یافته به صورت یک پایگاه ویژه اطلاعات شیمی و مهندسی شیمی که قابلیت توسعه و روزآمدی را داشته باشد ضروری به نظر می‌رسد.

مراجع:

۱. ابراهیم‌زاده، حسن و همکاران. (۱۳۷۱) واژه‌نامه زیست‌شناسی، تهران: انتشارات علوی.
۲. اتحادیه بین‌المللی شیمی محض و کاربردی «ایوپاک». (۱۳۶۴) نامگذاری در شیمی معدنی، ترجمه منصور عابدینی و همکاران، تهران: مرکز نشر دانشگاهی.
۳. اتحادیه بین‌المللی شیمی محض و کاربردی «ایوپاک». (۱۳۷۳) نامگذاری در شیمی آلی، ترجمه شروین اسفندیاری و همکاران، تهران: مرکز نشر دانشگاهی.
۴. اداره چکیده نامه‌های شیمی، انجمن شیمی آمریکا. (۱۳۷۵) راهنمای استفاده از چکیده‌نامه‌های شیمی. ترجمه مسعود حسن‌پور، تهران: مرکز نشر دانشگاهی.
۵. آقابخشی، علی. (۱۳۷۲) نمایه‌سازی همارا، تهران: مرکز اطلاعات و مدارک علمی ایران.
۶. آقاپور مقدم، سیدرضا. (۱۳۶۳) فرهنگ مصور شیمی، تهران: انتشارات اطلس.
۷. انجمن سلطنتی شیمی، گروه فنون آموزشی. (۱۳۷۰) روش استفاده از چکیده‌نامه‌های شیمی. ترجمه مسعود حسن‌پور، تهران: مرکز نشر دانشگاهی.
۸. باکستون، اندرو. و هاپکینسون، آلن. (۱۳۷۷) راهنمای کاربرد و آموزش نظام بازیابی اطلاعات CDS/ISIS، ترجمه علی‌اکبر پورسرباز، تبریز: نعمتی.
۹. پارسی اصفهانی، امیر. (۱۳۷۷) راهنمای مرجع کاربران سی.دی.اس آیسس تحت ویندوز، تهران: مرکز اطلاعات و مدارک علمی ایران.
۱۰. پارکر، س.پ. (۱۳۷۹) فرهنگ تشریحی علوم مهندسی مک گرو هیل، ترجمه محمد رضا افضلی، تهران: دانشیار.
۱۱. پارکر، س.پ. (۱۳۸۲) فرهنگ تشریحی شیمی، ترجمه تقی رجبی و همکاران، تهران: دانشیار.
۱۲. پورجوادی، علی. (۱۳۷۰) واژه‌نامه شیمی، ویرایش سوم، تهران: مرکز نشر دانشگاهی.
۱۳. پورجوادی، علی. (۱۳۷۹) واژگان شیمی و مهندسی شیمی، ویرایش دوم، تهران: مرکز نشر دانشگاهی.
۱۴. حق‌شناس، علی محمد. (۱۳۷۴) آواشناسی، تهران: انتشارات آگاه.
۱۵. راولی، جنیفر. (۱۳۷۴) نمایه‌سازی و چکیده‌نویسی، ترجمه دکتر جعفر مهران، تهران: سازمان مدارک فرهنگی انقلاب اسلامی.
۱۶. رجبی، تقی. (۱۳۸۰) اصطلاحنامه شیمی، علوم اطلاع رسانی، مرکز اطلاعات و مدارک علمی ایران، دوره ۱۶، شماره ۳ و ۴، بهار و تابستان ۱۳۸۰.

۱۷. رجبی، تقی. و غربی، حسین. (۱۳۷۸) اطلاعات علمی شیمی و مهندسی شیمی ایران، جلد اول (پایان نامه ها)، تهران: مرکز اطلاعات و مدارک علمی ایران.
۱۸. رجبی، تقی. و همکاران (۱۳۷۹) بررسی اطلاعات علمی شیمی و مهندسی شیمی ایران، گزارش طرح پژوهش، تهران: مرکز اطلاعات و مدارک علمی ایران.
۱۹. رها دوست، فاطمه. (۱۳۷۲) سرعنوانهای موضوعی پزشکی، تهران: دانشگاه علوم پزشکی ایران.
۲۰. سلطانیپور، زهرا. (۱۳۷۴) واژگان متالورژی (مواد)، تهران: مرکز نشر دانشگاهی.
۲۱. سمائی، سید مهدی. و رجبی، تقی. مشکلات یکسان نویسی اسامی ترکیبهای شیمیایی آلی در زبان فارسی، نخستین هم اندیشی مسائل واژه گزینی و اصطلاح شناسی، تهران: فرهنگستان زبان و ادب فارسی.
۲۲. شارب، دیوید ویلیام آرتور. (۱۳۷۵) فرهنگ شیمی، ترجمه عیسی یآوری، تهران: انتشارات فاطمی.
۲۳. صدیق بهزادی، ماندانا. (۱۳۷۵) شیوه نامه ضبط اعلام انگلیسی در فارسی، تهران: مرکز نشر دانشگاهی.
۲۴. فرهنگستان زبان و ادب فارسی. (۱۳۸۱) واژه های شیمی ۱ مصوب فرهنگستان زبان و ادب فارسی، ویرایش دوم، تهران: فرهنگستان زبان و ادب فارسی.
۲۵. فرهنگستان زبان و ادب فارسی. (۱۳۸۱) واژه های شیمی ۲ مصوب فرهنگستان زبان و ادب فارسی، ویرایش دوم، تهران: فرهنگستان زبان و ادب فارسی.
۲۶. گروه فیزیک. (۱۳۷۰) واژه نامه فیزیک، تهران: مرکز نشر دانشگاهی.
۲۷. مرکز نشر دانشگاهی. شیوه نامه، ویرایش دوم، تهران: مرکز نشر دانشگاهی.
۲۸. منصوری، رضا و همکاران (۱۳۷۳) واژه نامه علوم و تکنولوژی، تهران: مرکز نشر دانشگاهی.
۲۹. مولا پرست، فیروزه. (۱۳۶۳) اصطلاح شناسی و کاربرد آن در فعالیتهای اطلاع رسانی، نشریه فنی مرکز اسناد و مدارک علمی، شماره ۱، دوره ۸، ۹ - ۶۶.
۳۰. نجفی، ابوالحسن. (۱۳۷۰) غلط نویسیم، چاپ چهارم، تهران: مرکز نشر دانشگاهی.
۳۱. واژه نامه ریاضی، مرکز نشر دانشگاهی
۳۲. یاحقی، محمدجعفر. و ناصح، محمد مهدی. (۱۳۷۱) راهنمای نگارش و ویرایش، مشهد: مؤسسه چاپ و انتشارات آستان قدس رضوی.

۳۳. یونسکو. (۱۳۷۷) اصطلاحنامه نما، ترجمه ملوک السادات بهشتی، جلد اول و دوم، ویرایش دوم، تهران: مرکز اطلاعات و مدارک علمی ایران.

34. Ash, I. and Ash, M. (1992) Industrial Chemical Thesaurus, Second edition, Volume 1 & 2, VCH.
35. Ash, I. and Ash, M. (1994) Gardner's Chemical Synonyms and Trade Names, tenth edition, Gower Publishing limited.
36. Bottle, R.T. and Rowland, J.F.B. (1993) Information Sources in chemistry, Bowker - Saur Ltd.
37. ChemClub - from CambridgeSoft, www.chemfinder.com.
38. ChemExper Chemical Directory, <http://www.chemexper.com/>
39. Chemical Abstracts Service, a division of the American Chemical www.cas.org
40. Crystal, D. (1992) An Encyclopedic Dictionary of Language and Languages, Blackwell.
41. DEAN, J.A. (1985) Lange's Handbook of Chemistry, thirteenth edition, McGraw-Hill Book Company.
42. Grant, J. (1972) HACKH'S Chemical Dictionary, McGraw-Hill Book Company.
43. Hamped, C.A. & Hawley, G.G. (1982) Glossary of Chemical Term, Second edition, VNR.
44. Hornby, A.S. (1996) Oxford Advanced Learner's dictionary, fifth ed., oxford university press.
45. <http://etoh.niaaa.nih.gov/AODVol1/aodhqz.htm> .
46. International Atomic Energy Agency, (1996) INIS Thesaurus, -Vienna.
47. NIH Publication, (1988) Medical Subject Headings.
48. Parker, S.P. (1997) McGraw-Hill dictionary of chemistry, McGraw-Hill Book Company.
49. Rhodes, P.H. (1995) The Organic Chemist's Desk Reference, Chapman & hall.
50. Royal Society of chemistry, (1994) Further advances In chemical Information.
51. SHARP, D.W.A. (1983) The Penguin dictionary of chemistry, penguin books.
52. US Department of Defense, Office of Naval Research, (1967) Thesaurus of Engineering and Scientific Terms.
53. Wolman, Y. (1988) Chemical Information, John Wiley & Sons.

كد اصلي نرم افزار

```
1:  //-----
2:
3:  #include <vcl.h>
4:  #pragma hdrstop
5:
6:  #include "Jpeg.hpp"
7:  #include <Math.h>
8:
9:  #include "CompoundStruct.h"
10: #include "Functions.h"
11: #include "Warning2.h"
12: #include "AddEditItem.h"
13: #include "Main.h"
14: #include "SearchForm.h"
15: #include "ISO2709.h"
16: #include "DeleteFrom.h"
17:  //-----
18: #pragma package(smart_init)
19: #pragma link "ThemeMgr"
20: #pragma resource "*.dfm"
21: TMainForm *MainForm;
22: char ApplicationDatabaseProvider[50];
23: float ApplicationVersion;
24:
25:  //-----
26:
27:  __fastcall TMainForm::TMainForm(TComponent* Owner)
28:      : TForm(Owner)
29:  {
30:      ChromatographyActiveComponent = NULL;
31:      SpectralActiveComponent = NULL;
32:
33:      SetItemsActivity(NoRecord);
34:
35:      Data.DataFileName = "";
36:      ListItemsKind = IUPAC;
37:      ShowBlankListItems = true;
38:      BlankList = new TStringList;
39:      StrCopy(BlankTitle, "[BLANK IUPAC NAME]");
40:
41:      PaintTool = new PaintTools(Image);
42:      PaintTool->RotationFixed = PaintTool->SizeFixed = false;
43:      PolygonSideComboBox->Items->Add("3");
44:      PolygonSideComboBox->Items->Add("4");
45:      PolygonSideComboBox->Items->Add("5");
46:      PolygonSideComboBox->Items->Add("6");
47:      PolygonSideComboBox->Items->Add("7");
48:      PolygonSideComboBox->Items->Add("8");
49:      PolygonSideComboBox->Items->Add("9");
50:      PolygonSideComboBox->ItemIndex = 3;
51:
52:      MainProgressBar = new TProgressBar(StatusBar1);
53:      MainProgressBar->Parent = StatusBar1;
54:
55:      MainProgressBar->Left = StatusBar1->Panels->Items[0]->Width;
56:      MainProgressBar->Top = 2; //StatusBar1->Panels->Items[0]->Width;
57:      MainProgressBar->Width = StatusBar1->Panels->Items[1]->Width - 4;
58:      MainProgressBar->Height = 16; //StatusBar1->Panels->Items[1]->Width;
59:
60:      StrCopy(ApplicationDatabaseProvider, "Chemical Compounds Information Data Bank");
61:      ApplicationVersion = 1.0;
62:  }
63:  //-----
64:
65:  __fastcall TMainForm::~TMainForm()
66:  {
67:      delete MainProgressBar;
68:      delete BlankList;
69:      delete PaintTool;
70:  }
71:  //-----
72:
73:  void __fastcall TMainForm::Splitter2Moved(TObject *Sender)
74:  {
75:      Label4->Left = Splitter2->Left - Label4->Width - 2;
```

```
76: }
77: //-----
78:
79: void __fastcall TMainForm::ExitClick(TObject *Sender)
80: {
81:     Close();
82: }
83: //-----
84:
85: void __fastcall TMainForm::CheckSpeedButtons(TObject *Sender)
86: {
87:     if(PaintTool->CurrentActivation == ATOMTEXT)
88:         PaintTool->HideAtomTextEdit();
89:
90:     if(Sender != SpeedButton1)
91:         SpeedButton1->Down = false;
92:     else
93:         PaintTool->CurrentActivation = DRAWLINE;
94:
95:     if(Sender != SpeedButton2)
96:         SpeedButton2->Down = false;
97:     else
98:         PaintTool->CurrentActivation = DRAWCONTINUOUSLINE;
99:
100:    if(Sender != SpeedButton3)
101:        SpeedButton3->Down = false;
102:    else
103:        PaintTool->CurrentActivation = DRAWCIRCULARARC;
104:
105:    if(Sender != SpeedButton4)
106:        SpeedButton4->Down = false;
107:    if(Sender != SpeedButton5)
108:        SpeedButton5->Down = false;
109:
110:    if(Sender != SpeedButton6)
111:        SpeedButton6->Down = false;
112:    else
113:        PaintTool->CurrentActivation = DRAWRECTANGLE;
114:
115:    if(Sender != SpeedButton7)
116:        SpeedButton7->Down = false;
117:    else
118:        PaintTool->CurrentActivation = DRAWCIRCLE;
119:
120:    if(Sender != SpeedButton8)
121:        SpeedButton8->Down = false;
122:    if(Sender != SpeedButton9)
123:        SpeedButton9->Down = false;
124:
125:    if(Sender != SpeedButton10)
126:        SpeedButton10->Down = false;
127:    else
128:        PaintTool->CurrentActivation = DRAWCYCLOPENTANE;
129:
130:    if(Sender != SpeedButton11)
131:        SpeedButton11->Down = false;
132:    else
133:        PaintTool->CurrentActivation = DRAWPOLYGON;
134:
135:    if(Sender != SpeedButton12)
136:        SpeedButton12->Down = false;
137:    else
138:        PaintTool->CurrentActivation = DRAWCHAIN;
139:
140:    if(Sender != SpeedButton13)
141:        SpeedButton13->Down = false;
142:
143:    if(Sender != SpeedButton14)
144:        SpeedButton14->Down = false;
145:    else
146:        PaintTool->CurrentActivation = DRAWARROW;
147:
148:    if(Sender != SpeedButton15)
149:        SpeedButton15->Down = false;
150:    else
```

```
151:         PaintTool->CurrentActivation = DRAWCIRCULARARROW;
152:
153:     if(Sender != SpeedButton16)
154:         SpeedButton16->Down = false;
155:     else
156:         PaintTool->CurrentActivation = DRAWARROW2SIDE;
157:
158:     if(Sender != SpeedButton17)
159:         SpeedButton17->Down = false;
160:     else
161:         PaintTool->CurrentActivation = DRAWCIRCULARARROW2SIDE;
162:
163:     if(Sender != SpeedButton18)
164:         SpeedButton18->Down = false;
165:     else
166:         PaintTool->CurrentActivation = DRAWDDOUBLEBOND;
167:
168:     if(Sender != SpeedButton19)
169:         SpeedButton19->Down = false;
170:     else
171:         PaintTool->CurrentActivation = DRAWTRIPLEBOND;
172:
173:     if(Sender != SpeedButton20)
174:         SpeedButton20->Down = false;
175:     else
176:         PaintTool->CurrentActivation = DRAWUPWEDGE;
177:
178:     if(Sender != SpeedButton21)
179:         SpeedButton21->Down = false;
180:     else
181:         PaintTool->CurrentActivation = DRAWDOWNWEDGE;
182:
183:     if(Sender != SpeedButton24)
184:         SpeedButton24->Down = false;
185:     else
186:         PaintTool->CurrentActivation = DRAWARROW2SIDESEP;
187:
188:     if(Sender != ToolButton1)
189:         ToolButton1->Down = false;
190:     else
191:     {
192:         ToolButton1->Down = true;
193:         PaintTool->CurrentActivation = SHAPESELECTION;
194:     }
195:
196:     if(Sender != ToolButton2)
197:         ToolButton2->Down = false;
198:     else
199:     {
200:         ToolButton2->Down = true;
201:         PaintTool->CurrentActivation = GROUPSELECTION;
202:     }
203:
204:     if(Sender != ToolButton3)
205:         ToolButton3->Down = false;
206:     else
207:     {
208:         ToolButton3->Down = true;
209:         PaintTool->CurrentActivation = ROTATION;
210:     }
211:
212:     if(Sender != ToolButton4)
213:         ToolButton4->Down = false;
214:     else
215:     {
216:         ToolButton4->Down = true;
217:         PaintTool->CurrentActivation = ATOMTEXT;
218:     }
219:
220:     if(Sender != ToolButton5)
221:         ToolButton5->Down = false;
222:     else
223:     {
224:         ToolButton5->Down = true;
225:         PaintTool->CurrentActivation = DELETESHAPE;
```

```
226:     }
227: }
228: //-----
229:
230: void __fastcall TMainForm::PolygonSideComboBoxChange(TObject *Sender)
231: {
232:     PaintTool->ActivePolygonSideNumber =
233:         StrToInt(PolygonSideComboBox->Items->Strings[PolygonSideComboBox->ItemIndex]);
234: //-----
235:
236: void __fastcall TMainForm::SpeedButton22Click(TObject *Sender)
237: {
238:     PaintTool->SizeFixed = SpeedButton22->Down;
239: }
240: //-----
241:
242: void __fastcall TMainForm::SpeedButton23Click(TObject *Sender)
243: {
244:     PaintTool->RotationFixed = SpeedButton23->Down;
245: }
246: //-----
247:
248:
249: void __fastcall TMainForm::FormulaForm1Click(TObject *Sender)
250: {
251:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.AtomText.TextForm = false;
252:     PaintTool->RefreshAllImage();
253: }
254: //-----
255:
256: void __fastcall TMainForm::TextForm1Click(TObject *Sender)
257: {
258:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.AtomText.TextForm = true;
259:     PaintTool->RefreshAllImage();
260: }
261: //-----
262:
263: void __fastcall TMainForm::RefreshAll1Click(TObject *Sender)
264: {
265:     PaintTool->RefreshAllImage();
266: }
267: //-----
268:
269: void __fastcall TMainForm::SingleBond1Click(TObject *Sender)
270: {
271:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.Line.BondNumber = 1;
272:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.Line.LeftSideBond = false;
273:     PaintTool->RefreshAllImage();
274: }
275: //-----
276:
277: void __fastcall TMainForm::DoubleBond1Click(TObject *Sender)
278: {
279:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.Line.BondNumber = 2;
280:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.Line.LeftSideBond = false;
281:     PaintTool->RefreshAllImage();
282: }
283: //-----
284:
285: void __fastcall TMainForm::TripleBond1Click(TObject *Sender)
286: {
287:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.Line.BondNumber = 3;
288:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.Line.LeftSideBond = false;
289:     PaintTool->RefreshAllImage();
290: }
291: //-----
292:
293: void __fastcall TMainForm::FlipBond1Click(TObject *Sender)
294: {
295:     PaintTool->GData[PaintTool->OnShapeNumber]->Shape.Line.LeftSideBond =
296:         PaintTool->GData[PaintTool->OnShapeNumber]->Shape.Line.LeftSideBond == false ? true :
297:         false;
298:     PaintTool->RefreshAllImage();
299: }
```

```

298: //-----
299:
300: void __fastcall TMainForm::ImageMouseDown(TObject *Sender,
301:     TMouseButton Button, TShiftState Shift, int X, int Y)
302: {
303:     TPoint CurrentPoint = Point(X, Y);
304:
305:     if(PaintTool->CurrentActivation == ATOMTEXT)
306:         PaintTool->HideAtomTextEdit();
307:
308:     TabSheet3->SetFocus();
309:
310:     if(Button == mbLeft)
311:     {
312:         if(PaintTool->CurrentActivation == DELETESHAPE)
313:         {
314:             int ShapeToBeDelete;
315:
316:             if(PaintTool->IsMouseOnVertex(CurrentPoint.x, CurrentPoint.y))
317:             {
318:                 ShapeToBeDelete =
319: PaintTool->FirstShapeContainPoint(&PaintTool->OnVertexPoint);
320:                 while(ShapeToBeDelete > -1)
321:                 {
322:                     PaintTool->DeleteShape(ShapeToBeDelete);
323:                     PaintTool->IsMouseOnVertex(CurrentPoint.x, CurrentPoint.y);
324:                     ShapeToBeDelete =
325: PaintTool->FirstShapeContainPoint(&PaintTool->OnVertexPoint);
326:                 }
327:                 PaintTool->RefreshAllImage();
328:             }
329:             else if(PaintTool->IsMouseOnShape(CurrentPoint.x, CurrentPoint.y, false))
330:             {
331:                 PaintTool->DeleteShape(PaintTool->OnShapeNumber);
332:                 PaintTool->RefreshAllImage();
333:             }
334:             PaintTool->DetermineResultantPoint();
335:         }
336:         else if(!PaintTool->Drawing)
337:         {
338:             PaintTool->Drawing = True;
339:             if(PaintTool->IsMouseOnVertex(CurrentPoint.x, CurrentPoint.y))
340:             {
341:                 PaintTool->StartPoint = PaintTool->OnVertexPoint;
342:                 if(PaintTool->CurrentActivation == SHAPESELECTION ||
343: PaintTool->CurrentActivation == GROUPSELECTION)
344:                 {
345:                     PaintTool->MoveSelVertex = true;
346:                     PaintTool->MoveSelShape = false;
347:                     if(!PaintTool->IsVertexSelected(&PaintTool->StartPoint))
348:                         PaintTool->SelectVertexOnPoint(&PaintTool->StartPoint,
349: Shift.Contains(ssShift) ? false : true);
350:                     else if(Shift.Contains(ssShift))
351:                         PaintTool->DeselectVertexOnPoint(&PaintTool->StartPoint);
352:                     else
353:                         PaintTool->SelectVertexOnPoint(&PaintTool->StartPoint);
354:
355:                     PaintTool->SelectingArea = TRect(-1, -1, -1, -1);
356:                 }
357:             }
358:             else if(PaintTool->CurrentActivation == ROTATION)
359:             {
360:                 PaintTool->MoveSelVertex = true;
361:                 PaintTool->MoveSelShape = false;
362:             }
363:             else if(PaintTool->CurrentActivation == ATOMTEXT)
364:                 PaintTool->ShowAtomTextEdit(Mouse->CursorPos.x,
365: Mouse->CursorPos.y);
366:
367:             PaintTool->DrawShapes();
368:         }
369:     }
370:     else if(PaintTool->IsMouseOnShape(CurrentPoint.x, CurrentPoint.y, false))
371:     {
372:         PaintTool->StartPoint = CurrentPoint;
373:         if(PaintTool->CurrentActivation == SHAPESELECTION)

```

```
368:         {
369:             PaintTool->MoveSelShape = true;
370:             PaintTool->MoveSelVertex = false;
371:             if(!PaintTool->IsShapeSelected(PaintTool->OnShapeNumber))
372:                 PaintTool->SelectShape(PaintTool->OnShapeNumber,
Shift.Contains(ssShift) ? false : true, false);
373:             else if(Shift.Contains(ssShift))
374:                 PaintTool->DeselectShape(PaintTool->OnShapeNumber, false);
375:             else
376:                 PaintTool->SelectShape(PaintTool->OnShapeNumber, true,
false);
377:         }
378:         else if(PaintTool->CurrentActivation == GROUPSELECTION ||
PaintTool->CurrentActivation == ROTATION)
379:         {
380:             PaintTool->MoveSelShape = true;
381:             PaintTool->MoveSelVertex = false;
382:             if(!PaintTool->IsShapeSelected(PaintTool->OnShapeNumber))
383:
PaintTool->SelectGroup(PaintTool->GData[PaintTool->OnShapeNumber]->GroupNumber,
Shift.Contains(ssShift) ? false : true, false);
384:             else if(Shift.Contains(ssShift))
385:
PaintTool->DeselectGroup(PaintTool->GData[PaintTool->OnShapeNumber]->GroupNumber,
false);
386:             else
387:
PaintTool->SelectGroup(PaintTool->GData[PaintTool->OnShapeNumber]->GroupNumber, true,
false);
388:         }
389:         else if(PaintTool->CurrentActivation == ATOMTEXT)
390:             PaintTool->ShowAtomTextEdit(Mouse->CursorPos.x,
Mouse->CursorPos.y);
391:     }
392:     else
393:     {
394:         PaintTool->MoveSelShape = false;
395:         PaintTool->MoveSelVertex = false;
396:         PaintTool->StartPoint = CurrentPoint;
397:
398:         if(PaintTool->CurrentActivation == SHAPESELECTION ||
PaintTool->CurrentActivation == GROUPSELECTION ||
PaintTool->CurrentActivation == ROTATION)
399:         {
400:             if(!Shift.Contains(ssShift))
401:                 PaintTool->SelectShape(-2, true, true); //Deselecting all
402:             shapes
403:
404:         }
405:         else if(PaintTool->CurrentActivation == ATOMTEXT)
406:             PaintTool->ShowAtomTextEdit(Mouse->CursorPos.x,
Mouse->CursorPos.y);
407:     }
408:     PaintTool->MovePoint = PaintTool->StartPoint;
409:
410:     PaintTool->DetermineResultantPoint();
411: }
412: }
413: }
414: //-----
415:
416: void __fastcall TMainForm::ImageMouseMove(TObject *Sender,
417:     TShiftState Shift, int X, int Y)
418: {
419:     TPoint CurrentPoint = Point(X, Y);
420:
421:     if(PaintTool->Drawing)
422:     {
423:         if(PaintTool->MoveSelVertex || PaintTool->MoveSelShape)
424:         {
425:             TPoint DistancePoint = TPoint(CurrentPoint.x - PaintTool->StartPoint.x,
CurrentPoint.y - PaintTool->StartPoint.y);
426:             PaintTool->DrawShapes(true, false);
427:
428:             if(PaintTool->CurrentActivation == ROTATION)
429:                 PaintTool->RotateSelectedVertex(&DistancePoint);
```

```
430:         else
431:             PaintTool->MoveSelectedVertex(&DistancePoint);
432:
433:         PaintTool->StartPoint = CurrentPoint;
434:         PaintTool->DrawShapes(false);
435:     }
436:     else
437:     {
438:         if(PaintTool->CurrentActivation == SHAPESELECTION ||
PaintTool->CurrentActivation == GROUPELECTION)
439:         {
440:             PaintTool->PenMode = pmNotXor;
441:             PaintTool->Image->Canvas->Pen->Mode = PaintTool->PenMode;
442:             PaintTool->Image->Canvas->Pen->Style = psDot;
443:             PaintTool->Image->Canvas->Pen->Color = (TColor)NORMALCOLOR;
444:             PaintTool->DrawRectangle(&PaintTool->StartPoint,
&TPoint(PaintTool->StartPoint.x, PaintTool->MovePoint.y),
&TPoint(PaintTool->MovePoint.x, PaintTool->StartPoint.y), &PaintTool->MovePoint);
445:             PaintTool->MovePoint = CurrentPoint;
446:             PaintTool->DrawRectangle(&PaintTool->StartPoint,
&TPoint(PaintTool->StartPoint.x, PaintTool->MovePoint.y),
&TPoint(PaintTool->MovePoint.x, PaintTool->StartPoint.y), &PaintTool->MovePoint);
447:             PaintTool->Image->Canvas->Pen->Style = psSolid;
448:             PaintTool->SelectingArea = TRect(PaintTool->StartPoint,
PaintTool->MovePoint);
449:             if(PaintTool->SelectingArea.Right < PaintTool->SelectingArea.Left)
450:                 SWAPINTEGER(PaintTool->SelectingArea.Right,
PaintTool->SelectingArea.Left)
451:             if(PaintTool->SelectingArea.Bottom < PaintTool->SelectingArea.Top)
452:                 SWAPINTEGER(PaintTool->SelectingArea.Bottom,
PaintTool->SelectingArea.Top)
453:         }
454:         else
455:         {
456:             PaintTool->PenMode = pmNotXor;
457:             PaintTool->DrawShape();
458:             if(PaintTool->IsMouseOnVertex(CurrentPoint.x, CurrentPoint.y))
459:                 PaintTool->MovePoint = PaintTool->OnVertexPoint;
460:             else
461:             {
462:                 if(PaintTool->RotationFixed)
463:                     CurrentPoint =
PaintTool->FindNearestFixedRotationPoint(&PaintTool->StartPoint, &CurrentPoint,
FIXROTATIONSIZE);
464:
465:                 if(PaintTool->SizeFixed)
466:                     PaintTool->MovePoint =
PaintTool->FindNearestFixedPoint(&PaintTool->StartPoint, &CurrentPoint, FIXSIZE);
467:                 else
468:                     PaintTool->MovePoint = CurrentPoint;
469:             }
470:
471:             if(PaintTool->CurrentActivation == DRAWCIRCULARARC ||
PaintTool->CurrentActivation == DRAWCIRCULARARROW || PaintTool->CurrentActivation ==
DRAWCIRCULARARROW2SIDE)
472:             {
473:                 PaintTool->MidPoint.x = (PaintTool->MovePoint.y -
PaintTool->StartPoint.y + PaintTool->MovePoint.x + PaintTool->StartPoint.x) / 2;
474:                 PaintTool->MidPoint.y = (PaintTool->StartPoint.x -
PaintTool->MovePoint.x + PaintTool->MovePoint.y + PaintTool->StartPoint.y) / 2;
475:             }
476:
477:             PaintTool->DrawShape();
478:         }
479:     }
480: }
481: else
482:     if(!PaintTool->IsMouseOnVertex(CurrentPoint.x, CurrentPoint.y))
483:         PaintTool->IsMouseOnShape(CurrentPoint.x, CurrentPoint.y);
484: }
485: //-----
486:
487: void __fastcall TMainForm::ImageMouseUp(TObject *Sender,
488:     TMouseButton Button, TShiftState Shift, int X, int Y)
489: {
```

```
490:     TPoint CurrentPoint = Point(X, Y);
491:
492:     if (PaintTool->Drawing)
493:     {
494:         if(PaintTool->CurrentActivation != SHAPESELECTION &&
PaintTool->CurrentActivation != GROUPSELECTION)
495:         {
496:             PaintTool->PenMode = pmCopy;
497:
498:             if(PaintTool->RotationFixed)
499:                 CurrentPoint =
PaintTool->FindNearestFixedRotationPoint(&PaintTool->StartPoint, &CurrentPoint,
FIXROTATIONSIZE);
500:
501:             if(PaintTool->SizeFixed)
502:                 PaintTool->MovePoint =
PaintTool->FindNearestFixedPoint(&PaintTool->StartPoint, &CurrentPoint, FIXSIZE);
503:             else
504:                 PaintTool->MovePoint = CurrentPoint;
505:
506:             PaintTool->DrawShape();
507:         }
508:         else if(!PaintTool->MoveSelVertex && !PaintTool->MoveSelShape)
509:         {
510:             PaintTool->PenMode = pmNotXor;
511:             PaintTool->Image->Canvas->Pen->Mode = PaintTool->PenMode;
512:             PaintTool->Image->Canvas->Pen->Style = psDot;
513:             PaintTool->Image->Canvas->Pen->Color = (TColor)NORMALCOLOR;
514:             PaintTool->DrawRectangle(&PaintTool->StartPoint,
&TPoint(PaintTool->StartPoint.x, PaintTool->MovePoint.y),
&TPoint(PaintTool->MovePoint.x, PaintTool->StartPoint.y), &PaintTool->MovePoint);
515:             PaintTool->MovePoint = CurrentPoint;
516:             PaintTool->Image->Canvas->Pen->Style = psSolid;
517:             PaintTool->PenMode = pmCopy;
518:             for(register i = 0; i < PaintTool->ShapeCounter; i++)
519:             {
520:                 if(PaintTool->IsShapeInSelectedArea(i))
521:                     PaintTool->SelectShape(i, false, false);
522:             }
523:
524:             PaintTool->DrawShapes();
525:         }
526:
527:         if(PaintTool->MoveSelVertex || PaintTool->MoveSelShape)
528:             PaintTool->MoveSelVertex = PaintTool->MoveSelShape = PaintTool->Drawing
false;
529:
530:         switch(PaintTool->CurrentActivation)
531:         {
532:             case ROTATION:
533:                 PaintTool->FixRotatedPoints();
534:             case SHAPESELECTION:
535:             case DELETESHAPE:
536:             case GROUPSELECTION:
537:             case ATOMTEXT:
538:                 PaintTool->Drawing = false;
539:                 break;
540:
541:             case DRAWLINE:
542:             case DRAWRECTANGLE:
543:             case DRAWCIRCULARARC:
544:             case DRAWCIRCULARARROW:
545:             case DRAWARROW:
546:             case DRAWCIRCULARARROW2SIDE:
547:             case DRAWARROW2SIDE:
548:             case DRAWARROW2SIDESEP:
549:             case DRAWCIRCLE:
550:             case DRAWDOUBLEBOND:
551:             case DRAWTRIPLEBOND:
552:             case DRAWUPWEDGE:
553:             case DRAWDOWNWEDGE:
554:                 PaintTool->AddShapeData(PaintTool->CurrentActivation, true);
555:                 PaintTool->Drawing = false;
556:                 PaintTool->DrawShapes();
557:                 PaintTool->GroupCounter++;
```

```
558:         break;
559:
560:     case DRAWCONTINUOUSLINE:
561:         if(Button == mbRight)
562:         {
563:             PaintTool->AddShapeData(PaintTool->CurrentActivation, true);
564:             PaintTool->SelectGroup(PaintTool->GroupCounter, true, false);
565:             PaintTool->DrawShapes();
566:             PaintTool->Drawing = false;
567:             PaintTool->GroupCounter++;
568:         }
569:     else
570:     {
571:         PaintTool->AddShapeData(PaintTool->CurrentActivation, true);
572:         PaintTool->StartPoint = PaintTool->MovePoint;
573:     }
574:     break;
575:
576: case DRAWCHAIN:
577: case DRAWPOLYGON:
578: case DRAWCYCLOPENTANE:
579:     PaintTool->AddShapeData(PaintTool->CurrentActivation, true);
580:     PaintTool->SelectGroup(PaintTool->GroupCounter, true, false);
581:     PaintTool->DrawShapes();
582:     PaintTool->Drawing = false;
583:     PaintTool->GroupCounter++;
584:     break;
585: }
586: }
587: }
588: //-----
589:
590: void __fastcall TMainForm::SetItemsActivity(int Activity)
591: {
592:     TControl *ChildControl;
593:     TClass ClassRef;
594:     AnsiString ClassNameStr;
595:     register i;
596:
597:     switch(Activity)
598:     {
599:         case NewRecord:
600:         case EditRecord:
601:             ListView1->Enabled = false;
602:
603:             Panel4->Enabled = true;
604:             ValueListEditor2->OnKeyDown = ListBox3KeyDown;
605:             ValueListEditor2->PopupMenu = NameListMenu;
606:             ValueListEditor1->OnKeyDown = ListBox3KeyDown;
607:             ValueListEditor1->PopupMenu = NameListMenu;
608:
609:             TabSheet4->Enabled = true;
610:             Button4->Enabled = true;
611:             Button5->Enabled = true;
612:             ToolBar8->Visible = true;
613:             ControlBar2->Visible = true;
614:             Image->Enabled = true;
615:
616:             ComboBox1->Enabled = SpectralActiveComponent == NULL ? false : true;
617:             Image5->Enabled = SpectralActiveComponent == NULL ? false : true;
618:             Memo1->Enabled = SpectralActiveComponent == NULL ? false : true;
619:             ScrollBox2->PopupMenu = PictureListMenu;
620:
621:             TabSheet7->Enabled = true;
622:
623:             Image6->Enabled = ChoromatographyActiveComponent == NULL ? false : true;
624:             Memo3->Enabled = ChoromatographyActiveComponent == NULL ? false : true;
625:             ScrollBox3->PopupMenu = PictureListMenu;
626:
627:             TabSheet9->Enabled = true;
628:
629:             for (i = 0; i < ScrollBox4->ControlCount; i++)
630:             {
631:                 ChildControl = ScrollBox4->Controls[i];
632:                 ClassRef = ChildControl->ClassType();
```

```
633:         ClassNameStr = String(ClassRef->ClassName());
634:         if(ClassNameStr == "TLabelEdit" ||
635:            ClassNameStr == "TGroupBox" ||
636:            ClassNameStr == "TEdit")
637:             ChildControl->Enabled = true;
638:     }
639:
640:     for (i = 0; i < ScrollBox2->ControlCount; i++)
641:     {
642:         ChildControl = ScrollBox2->Controls[i];
643:         ClassRef = ChildControl->ClassType();
644:         ClassNameStr = String(ClassRef->ClassName());
645:         if(ClassNameStr == "TPanel")
646:             static_cast<TPanel *>(ChildControl)->PopupMenu =
PictureListItemMenu;
647:     }
648:
649:     for (i = 0; i < ScrollBox3->ControlCount; i++)
650:     {
651:         ChildControl = ScrollBox3->Controls[i];
652:         ClassRef = ChildControl->ClassType();
653:         ClassNameStr = String(ClassRef->ClassName());
654:         if(ClassNameStr == "TPanel")
655:             static_cast<TPanel *>(ChildControl)->PopupMenu =
PictureListItemMenu;
656:     }
657:
658:     SaveButton->Enabled = true;
659:     CancelButton->Enabled = true;
660:     EditButton->Enabled = false;
661:     NewButton->Enabled = false;
662:
663:     New->Enabled = false;
664:     ToolButton10->Enabled = false;
665:     Open->Enabled = false;
666:     ToolButton6->Enabled = false;
667:     Save_As->Enabled = false;
668:     ISO_2709->Enabled = false;
669:     Optimize->Enabled = false;
670:     Find->Enabled = false;
671:     ToolButton9->Enabled = false;
672:     ToolButton11->Enabled = false;
673:     ListView->Enabled = false;
674:     ToolButton7->Enabled = true;
675:     break;
676:
677:     case ShowRecord:
678:     case NoRecord:
679:         ListView1->Enabled = true;
680:
681:         Panel4->Enabled = false;
682:         ValueListEditor2->OnKeyDown = NULL;
683:         ValueListEditor2->PopupMenu = NULL;
684:         ValueListEditor1->OnKeyDown = NULL;
685:         ValueListEditor1->PopupMenu = NULL;
686:
687:         TabSheet4->Enabled = false;
688:         Button4->Enabled = false;
689:         Button5->Enabled = false;
690:         ToolBar8->Visible = false;
691:         ControlBar2->Visible = false;
692:         Image->Enabled = false;
693:
694:         ComboBox1->Enabled = false;
695:         Memo1->Enabled = false;
696:         Image5->Enabled = false;
697:         ScrollBox2->PopupMenu = NULL;
698:
699:         TabSheet7->Enabled = false;
700:
701:         Image6->Enabled = false;
702:         Memo3->Enabled = false;
703:         ScrollBox3->PopupMenu = NULL;
704:
705:         TabSheet9->Enabled = false;
```

```
706:
707:     for (i = 0; i < ScrollBox4->ControlCount; i++)
708:     {
709:         ChildControl = ScrollBox4->Controls[i];
710:         ClassRef = ChildControl->ClassType();
711:         ClassNameStr = String(ClassRef->ClassName());
712:         if(ClassNameStr == "TLabelEdit" ||
713:            ClassNameStr == "TGroupBox" ||
714:            ClassNameStr == "TEdit")
715:             ChildControl->Enabled = false;
716:     }
717:
718:     for (i = 0; i < ScrollBox2->ControlCount; i++)
719:     {
720:         ChildControl = ScrollBox2->Controls[i];
721:         ClassRef = ChildControl->ClassType();
722:         ClassNameStr = String(ClassRef->ClassName());
723:         if(ClassNameStr == "TPanel")
724:             static_cast<TPanel *>(ChildControl)->PopupMenu = NULL;
725:     }
726:
727:     for (i = 0; i < ScrollBox3->ControlCount; i++)
728:     {
729:         ChildControl = ScrollBox3->Controls[i];
730:         ClassRef = ChildControl->ClassType();
731:         ClassNameStr = String(ClassRef->ClassName());
732:         if(ClassNameStr == "TPanel")
733:             static_cast<TPanel *>(ChildControl)->PopupMenu = NULL;
734:     }
735:
736:     SaveButton->Enabled = false;
737:     CancelButton->Enabled = false;
738:     EditButton->Enabled = Activity == NoRecord ? false : true;
739:     NewButton->Enabled = true;
740:
741:     New->Enabled = true;
742:     ToolButton10->Enabled = true;
743:     Open->Enabled = true;
744:     ToolButton6->Enabled = true;
745:     Save_As->Enabled = true;
746:     ISO_2709->Enabled = true;
747:     Optimize->Enabled = true;
748:     Find->Enabled = true;
749:     ToolButton9->Enabled = true;
750:     ToolButton11->Enabled = true;
751:     ListView->Enabled = true;
752:     ToolButton7->Enabled = false;
753:     break;
754: }
755:
756: ScrollBox1->Enabled = true; //ScrollBoxes must not disable
757: ScrollBox2->Enabled = true;
758: ScrollBox3->Enabled = true;
759: }
760: //-----
761:
762: void __fastcall TMainForm::ListView1Click(TObject *Sender)
763: {
764:     ClearRecordValues();
765:     if(ListView1->SelCount < 1 || ListView1->ItemIndex < 0)
766:     {
767:         CurrentActivity = NoRecord;
768:         SetItemsActivity(NoRecord);
769:     }
770:     else
771:     {
772:         CurrentActivity = ShowRecord;
773:         SetItemsActivity(ShowRecord);
774:         Data.Load(GetItemNodeAddress());
775:         Data.SetMainFormData();
776:     }
777: }
778: //-----
779:
780: void __fastcall TMainForm::ListView1MouseDown(TObject *Sender,
```

```
781:         TMouseButton Button, TShiftState Shift, int X, int Y)
782: {
783:     if(!ListView1->GetItemAt(X, Y))
784:     {
785:         ListView1->ItemIndex = -1;
786:         ListView1->ClearSelection();
787:         ListView1Click(Sender);
788:     }
789: }
790: //-----
791:
792:
793:
794: void __fastcall TMainForm::NewButtonClick(TObject *Sender)
795: {
796:     if(Data.DataFileName.IsEmpty())
797:         if(OpenDialog2->Execute())
798:         {
799:             if(CheckIfOverWriteAccepted(OpenDialog2->FileName.c_str()))
800:             {
801:                 Data.DataFileName = OpenDialog2->FileName;
802:                 DeleteFile(Data.DataFileName);
803:                 Data.OpenDatabase();
804:                 Data.DataFile->SetHeaderProviderVersion(ApplicationDatabaseProvider
ApplicationVersion);
805:
806:                 Application->Title = "بانك اطلاعات تركيبات شيميايي - [" +
Data.DataFileName + "];
807:                 MainForm->Caption = Application->Title;
808:             }
809:         }
810:
811:     if(!Data.DataFileName.IsEmpty())
812:     {
813:         CurrentActivity = NewRecord;
814:         SetItemsActivity(CurrentActivity);
815:         ClearRecordValues();
816:     }
817: }
818: //-----
819:
820: void __fastcall TMainForm::EditButtonClick(TObject *Sender)
821: {
822:     CurrentActivity = EditRecord;
823:     SetItemsActivity(CurrentActivity);
824: }
825: //-----
826:
827: void __fastcall TMainForm::CancelButtonClick(TObject *Sender)
828: {
829:     ListView1Click(Sender);
830: }
831: //-----
832:
833: void __fastcall TMainForm::SaveButtonClick(TObject *Sender)
834: {
835:     int SearchIndex = 0;
836:     int CurrentIndex = ListView1->ItemIndex;
837:     int CurrentAddress = GetItemNodeAddress(CurrentIndex);
838:
839:     Data.GetMainFormData();
840:     if(Data.Save(CurrentActivity == EditRecord ? true : false))
841:     {
842:         if(CurrentActivity == EditRecord)
843:             ListView1->Items->Delete(CurrentIndex);
844:
845:         if(Data.RepeatNodeAddress)
846:             while((SearchIndex = SearchListForAddress(Data.RepeatNodeAddress,
SearchIndex)) >= 0)
847:                 ListView1->Items->Delete(SearchIndex);
848:
849:         SearchIndex = 0;
850:         if(ListItemsKind == Trade || ListItemsKind == Trivial)
851:             while((SearchIndex = SearchListForAddress(CurrentAddress, SearchIndex))
```

```
852:             ListView1->Items->Delete(SearchIndex);
853:
854:
855:             AddListItems(true);
856:
857:             CurrentActivity = ShowRecord;
858:             SetItemsActivity(CurrentActivity);
859:         }
860:     }
861: //-----
862:
863:
864: void __fastcall TMainForm::DeleteListClick(TObject *Sender)
865: {
866:     if(ListView1->SelCount > 0)
867:     {
868:         int DeleteKind = DeleteSelection->ShowDeleteSelection();
869:
870:         if(DeleteKind == DeleteFromDatabase)
871:         {
872:             if(Warning->ShowWarning("اخطار", "آیا مطمئن هستید؟ می‌خواهید رکوردهای", "انتخاب شده حذف شوند؟", "بله", "خیر", "", 2) == 1)
873:             {
874:                 while(ListView1->Selected)
875:                 {
876:                     Data.DeleteRecordAt(GetItemNodeAddress(ListView1->Selected));
877:                     ListView1->Selected->Delete();
878:                 }
879:
880:                 DeleteAllListItems(true);
881:             }
882:         }
883:         else if(DeleteKind == DeleteFromList)
884:         {
885:             DeleteAllListItems(true);
886:         }
887:     }
888: }
889: //-----
890:
891: void __fastcall TMainForm::N18Click(TObject *Sender)
892: {
893:     ListView1->SelectAll();
894: }
895: //-----
896:
897: void __fastcall TMainForm::N12Click(TObject *Sender)
898: {
899:     AnsiString Value;
900:
901:     if(NameListMenu->PopupComponent == ValueListEditor2)
902:         Value = AddEditItemForm->ShowAddEditItem("نام تجاری جدید", "");
903:     else if(NameListMenu->PopupComponent == ValueListEditor1)
904:         Value = AddEditItemForm->ShowAddEditItem("نام معمولی جدید", "");
905:
906:     if(!Value.IsEmpty())
907:         ((TValueListEditor *)NameListMenu->PopupComponent)->Strings->Add(Value);
908: }
909: //-----
910:
911: void __fastcall TMainForm::N13Click(TObject *Sender)
912: {
913:     if(((TValueListEditor *)NameListMenu->PopupComponent)->RowCount > 1)
914:         if(Warning->ShowWarning("اخطار", "آیا مطمئن هستید؟ می‌خواهید گزینه‌های انتخاب", "شده حذف شوند؟", "بله", "خیر", "", 2) == 1)
915:             ((TValueListEditor *)NameListMenu->PopupComponent)->DeleteRow(((TValueListEditor *)NameListMenu->PopupComponent)->Row);
916: }
917: //-----
918:
919: void __fastcall TMainForm::N14Click(TObject *Sender)
920: {
921:     if(((TValueListEditor *)NameListMenu->PopupComponent)->Row > 0)
922:     {
```

```

923:         if(NameListMenu->PopupComponent == ValueListEditor2)
924:             ValueListEditor2->Strings->Strings[ValueListEditor2->Row - 1] =
AddEditItemForm->ShowAddEditItem("ویرایش نام تجاری" ,
ValueListEditor2->Strings->Strings[ValueListEditor2->Row - 1]);
925:         else if(NameListMenu->PopupComponent == ValueListEditor1)
926:             ValueListEditor1->Strings->Strings[ValueListEditor1->Row - 1] =
AddEditItemForm->ShowAddEditItem("ویرایش نام معمولی" ,
ValueListEditor1->Strings->Strings[ValueListEditor1->Row - 1]);
927:     }
928: }
929: //-----
930:
931:
932: void __fastcall TMainForm::ListBox3KeyDown(TObject *Sender, WORD &Key,
933:     TShiftState Shift)
934: {
935:     if(Key == VK_RETURN)
936:     {
937:         AnsiString Value;
938:
939:         if(((TValueListEditor *)Sender)->Row > -1)
940:         {
941:             if(NameListMenu->PopupComponent == ValueListEditor2)
942:                 ValueListEditor2->Strings->Strings[ValueListEditor2->Row] =
AddEditItemForm->ShowAddEditItem("ویرایش نام تجاری" ,
ValueListEditor2->Strings->Strings[ValueListEditor2->Row]);
943:             else if(NameListMenu->PopupComponent == ValueListEditor1)
944:                 ValueListEditor1->Strings->Strings[ValueListEditor1->Row] =
AddEditItemForm->ShowAddEditItem("ویرایش نام معمولی" ,
ValueListEditor1->Strings->Strings[ValueListEditor1->Row]);
945:         }
946:     }
947:     else if(Key == VK_INSERT)
948:     {
949:         AnsiString Value;
950:
951:         if(Sender == ValueListEditor2)
952:             Value = AddEditItemForm->ShowAddEditItem("نام تجاری جدید" , "");
953:         else if(Sender == ValueListEditor1)
954:             Value = AddEditItemForm->ShowAddEditItem("نام معمولی جدید" , "");
955:
956:         if(!Value.IsEmpty())
957:             ((TValueListEditor *)Sender)->Strings->Add(Value);
958:     }
959:     else if(Key == VK_DELETE)
960:     {
961:         if(((TValueListEditor *)Sender)->RowCount > 1)
962:             if(Warning->ShowWarning("اخطار" , "آیا مطمئن هستید؟ می‌خواهید گزینه‌های" , "اخطار" , "بله" , "خیر" , "", 2) == 1)
963:                 ((TValueListEditor *)Sender)->DeleteRow( ((TValueListEditor
*)Sender)->Row );
964:     }
965: }
966: //-----
967:
968: void __fastcall TMainForm::Button5Click(TObject *Sender)
969: {
970:     LoadImage(Image7);
971: }
972: //-----
973:
974: void __fastcall TMainForm::LoadImage(TImage *ImageBox)
975: {
976:     TJPEGImage *jpg = new TJPEGImage();
977:
978:     if(OpenDialog1->Execute())
979:     {
980:         Screen->Cursor = crHourGlass;
981:
982:         AnsiString FileName = OpenDialog1->FileName;
983:         AnsiString Ext = ExtractFileExt(FileName);
984:
985:         if(!Ext.AnsiCompareIC(".BMP"))
986:         {
987:             ImageBox->Picture->LoadFromFile(FileName);

```

```
988:         }
989:     else
990:     {
991:         jp->LoadFromFile(FileName);
992:         ImageBox->Picture->Bitmap->Assign(jp);
993:     }
994:
995:     Screen->Cursor = crDefault;
996: }
997:
998: delete jp;
999: }
1000: //-----
1001:
1002: void __fastcall TMainForm::Button4Click(TObject *Sender)
1003: {
1004:     ClearImage(Image7);
1005:     //Image7->Picture->Bitmap->FreeImage();
1006:     //Image7->Picture = NULL;
1007: }
1008: //-----
1009:
1010: void __fastcall TMainForm::N17Click(TObject *Sender)
1011: {
1012:     int ComponentNumber = ((TScrollBox
1013: *)PictureListMenu->PopupComponent)->ComponentCount;
1014:
1015:     Screen->Cursor = crHourGlass;
1016:
1017:     TPanel *Panel = new TPanel(((TScrollBox *)PictureListMenu->PopupComponent));
1018:     Panel->Parent = ((TScrollBox *)PictureListMenu->PopupComponent);
1019:     Panel->Top = 5;
1020:     Panel->Left = 5 + (ComponentNumber * 80);
1021:     Panel->Height = 75;
1022:     Panel->Width = 60;
1023:     Panel->BevelOuter = bvLowered;
1024:     Panel->PopupMenu = PictureListItemMenu;
1025:
1026:     TImage *ActiveImage = new TImage(Panel);
1027:     ActiveImage->Parent = Panel;
1028:     ActiveImage->Height = 58;
1029:     ActiveImage->Width = 58;
1030:     ActiveImage->Top = 1;
1031:     ActiveImage->Left = 1;
1032:     ActiveImage->Stretch = true;
1033:     ActiveImage->Canvas->Pen->Color = clWhite;
1034:     ActiveImage->Canvas->FillRect(TRect(0, 0, ActiveImage->Height, ActiveImage->Width))
1035:
1036:     TLabel *Label = new TLabel(Panel);
1037:     Label->Parent = Panel;
1038:     Label->Height = 14;
1039:     Label->Width = 58;
1040:     Label->Alignment = taCenter;
1041:     Label->AutoSize = false;
1042:     Label->Caption = "شناسه";
1043:     Label->Top = 59;
1044:     Label->Left = 1;
1045:     Label->Hint = "Label";
1046:
1047:     TLabel *MemoLabel = new TLabel(Panel);
1048:     MemoLabel->Parent = Panel;
1049:     MemoLabel->Hint = "Memo";
1050:     MemoLabel->Visible = false;
1051:
1052:     TLabel *CategoryLabel = new TLabel(Panel);
1053:     CategoryLabel->Parent = Panel;
1054:     CategoryLabel->Hint = "Category";
1055:     CategoryLabel->Visible = false;
1056:
1057:     Panel->OnClick = PictureListPanelClick;
1058:     ActiveImage->OnClick = PictureListPanelChildsClick;
1059:     Label->OnClick = PictureListPanelChildsClick;
1060:
1061:     if((TScrollBox *)PictureListMenu->PopupComponent == ScrollBox2)
```

```

1062: {
1063:     SpectralActiveComponent = ChangeActivePictureItem(Panel,
SpectralActiveComponent);
1064:     SetPictureImage(Image5, SpectralActiveComponent);
1065:     SetDescriptionFromPictureList(Memo1, SpectralActiveComponent);
1066:     SetSpectralCategoryFromPictureList(ComboBox1, SpectralActiveComponent);
1067: }
1068: else if((TScrollBar *)PictureListMenu->PopupComponent == ScrollBox3)
1069: {
1070:     ChoromatographyActiveComponent = ChangeActivePictureItem(Panel,
ChoromatographyActiveComponent);
1071:     SetPictureImage(Image6, ChoromatographyActiveComponent);
1072:     SetDescriptionFromPictureList(Memo3, ChoromatographyActiveComponent);
1073: }
1074:
1075: SetItemsActivity(CurrentActivity);
1076: Screen->Cursor = crDefault;
1077: }
1078: //-----
1079:
1080: TComponent * __fastcall TMainForm::ChangeActivePictureItem(TComponent *New, TComponent
*Old)
1081: {
1082:     if(Old)
1083:     {
1084:         ((TPanel *)Old)->Color = clRed;
1085:     }
1086:
1087:     ((TPanel *)New)->Color = clYellow;
1088:
1089:     return New;
1090: }
1091: //-----
1092:
1093: void __fastcall TMainForm::ClearRecordValues(void)
1094: {
1095:     for(register i = ScrollBox2->ComponentCount - 1; i >= 0; i--)
1096:         if(String(ScrollBox2->Components[i]->ClassName()) == "TPanel")
1097:             delete ((TPanel *)ScrollBox2->Components[i]);
1098:
1099:     for(register i = ScrollBox3->ComponentCount - 1; i >= 0; i--)
1100:         if(String(ScrollBox3->Components[i]->ClassName()) == "TPanel")
1101:             delete ((TPanel *)ScrollBox3->Components[i]);
1102:
1103:     ChoromatographyActiveComponent = NULL;
1104:     SpectralActiveComponent = NULL;
1105:
1106:     ClearImage(Image5);
1107:     ClearImage(Image6);
1108:     ClearImage(Image7);
1109:     PaintTool->DeleteAllShapes();
1110:
1111:     ValueListEditor1->Strings->Clear();
1112:     ValueListEditor2->Strings->Clear();
1113:
1114:     ClearComponentsValue(PageControl1);
1115: }
1116: //-----
1117:
1118: void __fastcall TMainForm::SetDescriptionFromPictureList(TMemo *Memo, TComponent
*ActiveComponent)
1119: {
1120:     for(register i = 0; i < ActiveComponent->ComponentCount; i++)
1121:         if(String(ActiveComponent->Components[i]->ClassName()) == "TLabel")
1122:         {
1123:             if(((TLabel *)ActiveComponent->Components[i])->Hint == "Memo")
1124:             {
1125:                 Memo->Text = ((TLabel *)ActiveComponent->Components[i])->Caption;
1126:                 break;
1127:             }
1128:         }
1129: }
1130: //-----
1131:
1132: void __fastcall TMainForm::SetSpectralCategoryFromPictureList(TComboBox *Combo,

```

```

TComponent *ActiveComponent)
1133: {
1134:     for(register i = 0; i < ActiveComponent->ComponentCount; i++)
1135:         if(String(ActiveComponent->Components[i]->ClassName()) == "TLabel")
1136:         {
1137:             if(((TLabel *)ActiveComponent->Components[i])->Hint == "Category")
1138:             {
1139:                 Combo->Text = ((TLabel *)ActiveComponent->Components[i])->Caption;
1140:                 break;
1141:             }
1142:         }
1143: }
1144: //-----
1145:
1146: void __fastcall TMainForm::PictureListPanelClick(TObject *Sender)
1147: {
1148:     if(((TWinControl *)Sender)->Parent == ScrollBox2)
1149:     {
1150:         SpectralActiveComponent = ChangeActivePictureItem((TWinControl *)Sender,
SpectralActiveComponent);
1151:         SetPictureImage(Image5, SpectralActiveComponent);
1152:         SetDescriptionFromPictureList(Mem01, SpectralActiveComponent);
1153:         SetSpectralCategoryFromPictureList(ComboBox1, SpectralActiveComponent);
1154:     }
1155:     else if(((TWinControl *)Sender)->Parent == ScrollBox3)
1156:     {
1157:         ChoromatographyActiveComponent = ChangeActivePictureItem((TWinControl *)Sender,
ChoromatographyActiveComponent);
1158:         SetPictureImage(Image6, ChoromatographyActiveComponent);
1159:         SetDescriptionFromPictureList(Memo3, ChoromatographyActiveComponent);
1160:     }
1161: }
1162: //-----
1163:
1164: void __fastcall TMainForm::PictureListPanelChildsClick(TObject *Sender)
1165: {
1166:     if(((TWinControl *)Sender)->Parent->Parent == ScrollBox2)
1167:     {
1168:         SpectralActiveComponent = ChangeActivePictureItem(((TWinControl
*)Sender)->Parent, SpectralActiveComponent);
1169:         SetPictureImage(Image5, SpectralActiveComponent);
1170:         SetDescriptionFromPictureList(Mem01, SpectralActiveComponent);
1171:         SetSpectralCategoryFromPictureList(ComboBox1, SpectralActiveComponent);
1172:     }
1173:     else if(((TWinControl *)Sender)->Parent->Parent == ScrollBox3)
1174:     {
1175:         ChoromatographyActiveComponent = ChangeActivePictureItem(((TWinControl
*)Sender)->Parent, ChoromatographyActiveComponent);
1176:         SetPictureImage(Image6, ChoromatographyActiveComponent);
1177:         SetDescriptionFromPictureList(Memo3, ChoromatographyActiveComponent);
1178:     }
1179: }
1180: }
1181: //-----
1182:
1183:
1184: void __fastcall TMainForm::N19Click(TObject *Sender)
1185: {
1186:     TScrollBar *ParentScrollBar = (TScrollBar *)((TPanel
*)PictureListItemMenu->PopupComponent)->Parent;
1187:     int Index = ((TPanel *)PictureListItemMenu->PopupComponent)->ComponentIndex;
1188:
1189:     delete (TPanel *)PictureListItemMenu->PopupComponent;
1190:     RearrangeScrollBarPanels(ParentScrollBar);
1191:
1192:     if(!ParentScrollBar->ComponentCount)
1193:     {
1194:         if(ParentScrollBar == ScrollBox2)
1195:         {
1196:             SpectralActiveComponent = NULL;
1197:             Image5->Picture = NULL;
1198:             Mem01->Clear();
1199:         }
1200:         else if(ParentScrollBar == ScrollBox3)
1201:         {

```

```
1202:         ChoromatographyActiveComponent = NULL;
1203:         Image6->Picture = NULL;
1204:         Memo3->Clear();
1205:     }
1206: }
1207: else
1208:     if (ParentScrollBar == ScrollBox2)
1209:     {
1210:         if (Index < ParentScrollBar->ComponentCount)
1211:             SpectralActiveComponent =
1212:             ChangeActivePictureItem(ParentScrollBar->Components[Index], NULL);
1213:         else
1214:             SpectralActiveComponent =
1215:             ChangeActivePictureItem(ParentScrollBar->Components[Index - 1], NULL);
1216:         SetPictureImage(Image5, SpectralActiveComponent);
1217:         SetDescriptionFromPictureList(Memo1, SpectralActiveComponent);
1218:         SetSpectralCategoryFromPictureList(ComboBox1, SpectralActiveComponent);
1219:     }
1220:     else if (ParentScrollBar == ScrollBox3)
1221:     {
1222:         if (Index < ParentScrollBar->ComponentCount)
1223:             ChoromatographyActiveComponent =
1224:             ChangeActivePictureItem(ParentScrollBar->Components[Index], NULL);
1225:         else
1226:             ChoromatographyActiveComponent =
1227:             ChangeActivePictureItem(ParentScrollBar->Components[Index - 1], NULL);
1228:         SetPictureImage(Image6, ChoromatographyActiveComponent);
1229:         SetDescriptionFromPictureList(Memo3, ChoromatographyActiveComponent);
1230:     }
1231: SetItemsActivity(CurrentActivity);
1232: //-----
1233:
1234: void __fastcall TMainForm::RearrangeScrollBarPanels(TScrollBar *ParentScroll)
1235: {
1236:     for (register i = ParentScroll->ControlCount - 1; i >= 0; i--)
1237:         if (String(ParentScroll->Controls[i]->ClassName()) == "TPanel")
1238:             ((TPanel *)ParentScroll->Controls[i])->Left = 5 + (i * 80);
1239:     ParentScroll->Refresh();
1240: }
1241: //-----
1242:
1243:
1244: void __fastcall TMainForm::PictureListItemMenuPopup(TObject *Sender)
1245: {
1246:     PictureListPanelClick((TWinControl *)PictureListItemMenu->PopupComponent);
1247: }
1248: //-----
1249:
1250: void __fastcall TMainForm::N15Click(TObject *Sender)
1251: {
1252:     LoadImage((TImage *)PictureMenu->PopupComponent);
1253:
1254:     if ((TImage *)PictureMenu->PopupComponent == Image5)
1255:         SetPictureListImage(SpectralActiveComponent, Image5);
1256:     else if ((TImage *)PictureMenu->PopupComponent == Image6)
1257:         SetPictureListImage(ChoromatographyActiveComponent, Image6);
1258: }
1259: //-----
1260:
1261: void __fastcall TMainForm::SetPictureListImage(TComponent *Dest, TComponent *Source)
1262: {
1263:     for (register i = 0; i < Dest->ComponentCount; i++)
1264:         if (String(Dest->Components[i]->ClassName()) == "TImage")
1265:         {
1266:             ((TImage *)Dest->Components[i])->Picture->Assign(((TImage
1267: *)Source)->Picture);
1268:             break;
1269:         }
1270: }
1271: //-----
```

```
1272: void __fastcall TMainForm::SetPictureImage(TComponent *Dest, TComponent *Source)
1273: {
1274:     for(register i = 0; i < Source->ComponentCount; i++)
1275:         if(String(Source->Components[i]->ClassName()) == "TImage")
1276:         {
1277:             ((TImage *)Dest)->Picture->Assign(((TImage
*)Source->Components[i])->Picture);
1278:             break;
1279:         }
1280: }
1281: //-----
1282:
1283: void __fastcall TMainForm::N16Click(TObject *Sender)
1284: {
1285:     ClearImage((TImage *)PictureMenu->PopupComponent);
1286:
1287:     if((TImage *)PictureMenu->PopupComponent == Image5)
1288:         RemovePictureListImage(SpectralActiveComponent);
1289:     else
1290:         RemovePictureListImage(ChoromatographyActiveComponent);
1291: }
1292: //-----
1293:
1294: void __fastcall TMainForm::RemovePictureListImage(TComponent *Dest)
1295: {
1296:     for(register i = 0; i < Dest->ComponentCount; i++)
1297:         if(String(Dest->Components[i]->ClassName()) == "TImage")
1298:         {
1299:             ((TImage *)Dest->Components[i])->Picture = NULL;
1300:             ((TImage *)Dest->Components[i])->Canvas->Pen->Color = clWhite;
1301:             ((TImage *)Dest->Components[i])->Canvas->FillRect(TRect(0, 0, ((TImage
*)Dest->Components[i])->Height, ((TImage *)Dest->Components[i])->Width));
1302:             break;
1303:         }
1304: }
1305: //-----
1306:
1307: void __fastcall TMainForm::RemovePictureImage(TComponent *Dest)
1308: {
1309:     ((TImage *)Dest)->Picture = NULL;
1310: }
1311: //-----
1312:
1313: void __fastcall TMainForm::Memo1Change(TObject *Sender)
1314: {
1315:     for(register i = 0; i < SpectralActiveComponent->ComponentCount; i++)
1316:         if(String(SpectralActiveComponent->Components[i]->ClassName()) == "TLabel")
1317:         {
1318:             if(((TLabel *)SpectralActiveComponent->Components[i])->Hint == "Memo")
1319:             {
1320:                 ((TLabel *)SpectralActiveComponent->Components[i])->Caption =
Memo1->Text;
1321:                 break;
1322:             }
1323:         }
1324: }
1325: //-----
1326:
1327: void __fastcall TMainForm::Memo3Change(TObject *Sender)
1328: {
1329:     for(register i = 0; i < ChoromatographyActiveComponent->ComponentCount; i++)
1330:         if(String(ChoromatographyActiveComponent->Components[i]->ClassName()) ==
"TLabel")
1331:         {
1332:             if(((TLabel *)ChoromatographyActiveComponent->Components[i])->Hint ==
"Memo")
1333:             {
1334:                 ((TLabel *)ChoromatographyActiveComponent->Components[i])->Caption
Memo3->Text;
1335:                 break;
1336:             }
1337:         }
1338: }
1339: //-----
1340:
```

```
1341: void __fastcall TMainForm::ComboBox1Change(TObject *Sender)
1342: {
1343:     for(register i = 0; i < SpectralActiveComponent->ComponentCount; i++)
1344:         if(String(SpectralActiveComponent->Components[i]->ClassName()) == "TLabel")
1345:         {
1346:             if(((TLabel *)SpectralActiveComponent->Components[i])->Hint == "Category")
1347:             {
1348:                 ((TLabel *)SpectralActiveComponent->Components[i])->Caption =
ComboBox1->Text;
1349:                 break;
1350:             }
1351:         }
1352: }
1353: //-----
1354:
1355: void __fastcall TMainForm::N20Click(TObject *Sender)
1356: {
1357:     TScrollBar *ParentPanel = (TScrollBar *)((TPanel
*)PictureListItemMenu->PopupComponent);
1358:
1359:     for(register i = 0; i < ParentPanel->ComponentCount; i++)
1360:         if(String(ParentPanel->Components[i]->ClassName()) == "TLabel")
1361:         {
1362:             if(((TLabel *)ParentPanel->Components[i])->Hint == "Label")
1363:             {
1364:                 ((TLabel *)ParentPanel->Components[i])->Caption =
AddEditItemForm->ShowAddEditItem("تغییر شناسه" , ((TLabel
*)ParentPanel->Components[i])->Caption, false);
1365:                 break;
1366:             }
1367:         }
1368: }
1369: //-----
1370:
1371: void __fastcall TMainForm::AddListItems(bool Select)
1372: {
1373:     register i;
1374:     TStringList *List = new TStringList;
1375:     int CountRepeat = 0;
1376:
1377:     switch(ListItemsKind)
1378:     {
1379:         case IUPAC:
1380:             if(StrLen(Data.MainFields.IUPACName))
1381:             {
1382:                 ListView1->AddItem(Data.MainFields.IUPACName, (TObject
*)Data.CurrentNodeAddress);
1383:             }
1384:             else
1385:             {
1386:                 if(ShowBlankListItems)
1387:                 {
1388:                     ListView1->AddItem(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1389:                 }
1390:                 else
1391:                     BlankList->AddObject(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1392:             }
1393:             break;
1394:
1395:         case Trivial:
1396:             if(!Data.TrivialNames.IsEmpty())
1397:             {
1398:                 List->Text = Data.TrivialNames;
1399:                 for(i = 0; i < List->Count; i++)
1400:                 {
1401:                     if(!List->Strings[i].IsEmpty())
1402:                     {
1403:                         ListView1->AddItem(List->Strings[i], CountRepeat ? (TObject
*)Data.CurrentNodeAddress) : (TObject *)Data.CurrentNodeAddress);
1404:                         CountRepeat++;
1405:                     }
1406:                 }
1407:             }
```

```
1408:         else
1409:         {
1410:             if (ShowBlankListItems)
1411:             {
1412:                 ListView1->AddItem(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1413:             }
1414:             else
1415:                 BlankList->AddObject(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1416:         }
1417:         break;
1418:
1419:     case Trade:
1420:         if (!Data.TradeNames.IsEmpty())
1421:         {
1422:             List->Text = Data.TradeNames;
1423:             for (i = 0; i < List->Count; i++)
1424:             {
1425:                 if (!List->Strings[i].IsEmpty())
1426:                 {
1427:                     ListView1->AddItem(List->Strings[i], CountRepeat ? (TObject
*)Data.CurrentNodeAddress) : (TObject *)Data.CurrentNodeAddress);
1428:                     CountRepeat++;
1429:                 }
1430:             }
1431:         }
1432:         else
1433:         {
1434:             if (ShowBlankListItems)
1435:             {
1436:                 ListView1->AddItem(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1437:             }
1438:             else
1439:                 BlankList->AddObject(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1440:         }
1441:         break;
1442:
1443:     case MolecularFormula:
1444:         if (StrLen(Data.MainFields.MolecularFormula))
1445:         {
1446:             ListView1->AddItem(Data.MainFields.MolecularFormula, (TObject
*)Data.CurrentNodeAddress);
1447:         }
1448:         else
1449:         {
1450:             if (ShowBlankListItems)
1451:             {
1452:                 ListView1->AddItem(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1453:             }
1454:             else
1455:                 BlankList->AddObject(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1456:         }
1457:         break;
1458:
1459:     case ChemicalRegistryNumber:
1460:         if (StrLen(Data.MainFields.ChemicalRegistryNumber))
1461:         {
1462:             ListView1->AddItem(Data.MainFields.ChemicalRegistryNumber, (TObject
*)Data.CurrentNodeAddress);
1463:         }
1464:         else
1465:         {
1466:             if (ShowBlankListItems)
1467:             {
1468:                 ListView1->AddItem(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
1469:             }
1470:             else
1471:                 BlankList->AddObject(BlankTitle, (TObject
*)Data.CurrentNodeAddress);
```

```
1472:         }
1473:         break;
1474:     }
1475:
1476:     if(Select)
1477:     {
1478:         int SearchResult = 0;
1479:         bool FirstResult = true;
1480:
1481:         SearchResult = SearchListForAddress(Data.CurrentNodeAddress, SearchResult);
1482:         while(SearchResult != -1)
1483:         {
1484:             ListView1->Items->Item[SearchResult]->Selected = true;
1485:
1486:             if(FirstResult)
1487:             {
1488:                 ListView1->ItemIndex = SearchResult;
1489:                 FirstResult = false;
1490:             }
1491:
1492:             SearchResult = SearchListForAddress(Data.CurrentNodeAddress, SearchResult
+ 1);
1493:         }
1494:     }
1495:
1496:     delete List;
1497: }
1498:
1499: //-----
1500: void __fastcall TMainForm::ShowBlankListClick(TObject *Sender)
1501: {
1502:     if>ShowBlankListItems)
1503:     {
1504:         MoveBlankItemsFromList();
1505:         ShowBlankListItems = false;
1506:     }
1507:     else
1508:     {
1509:         MoveBlankItemsToList();
1510:         ShowBlankListItems = true;
1511:     }
1512:
1513:     ShowBlankList->Checked = ShowBlankListItems;
1514:     ShowBlankListP->Checked = ShowBlankListItems;
1515: }
1516: //-----
1517:
1518: void __fastcall TMainForm::MoveBlankItemsToList(void)
1519: {
1520:     Screen->Cursor = crHourGlass;
1521:
1522:     MainProgressBar->Min = 0;
1523:     MainProgressBar->Max = BlankList->Count;
1524:     MainProgressBar->Position = 0;
1525:
1526:     for(register i = BlankList->Count - 1; i >= 0; MainProgressBar->Position++ ,i--)
1527:         ListView1->AddItem(BlankTitle, (TObject *) (int) (BlankList->Objects[i]));
1528:
1529:     BlankList->Clear();
1530:     MainProgressBar->Position = 0;
1531:     Screen->Cursor = crDefault;
1532: }
1533: //-----
1534:
1535: void __fastcall TMainForm::MoveBlankItemsFromList(void)
1536: {
1537:     Screen->Cursor = crHourGlass;
1538:
1539:     MainProgressBar->Min = 0;
1540:     MainProgressBar->Max = ListView1->Items->Count;
1541:     MainProgressBar->Position = 0;
1542:
1543:     for(register i = ListView1->Items->Count - 1; i >= 0; MainProgressBar->Position++,
i--)
1544:         if(ListView1->Items->Item[i]->Caption == BlankTitle)
```

```
1545:         {
1546:             BlankList->AddObject(BlankTitle, (TObject
*) (int) (ListView1->Items->Item[i]->Data));
1547:             ListView1->Items->Delete(i);
1548:         }
1549:
1550:     MainProgressBar->Position = 0;
1551:     Screen->Cursor = crDefault;
1552: }
1553: //-----
1554:
1555: void __fastcall TMainForm::OpenClick(TObject *Sender)
1556: {
1557:     bool CheckVersion = false;
1558:
1559:     if(OpenDialog2->Execute())
1560:     {
1561:         Data.CloseDatabase();
1562:         Data.DataFileName = OpenDialog2->FileName;
1563:         Data.OpenDatabase();
1564:         if(!(CheckVersion = Data.CheckForCorrectVersion()))
1565:         {
1566:             Data.CloseDatabase();
1567:             Warning->ShowWarning("خطا", "اطلاعات مربوط به فایل مورد نظر قابل بازیابی", "تایید", "غیر باشد");
1568:         }
1569:     }
1570:
1571:     if(!Data.DataFileName.IsEmpty())
1572:     {
1573:         DeleteAllListItems();
1574:         if(CheckVersion)
1575:             LoadAll();
1576:
1577:         CurrentActivity = NoRecord;
1578:         SetItemsActivity(CurrentActivity);
1579:         ClearRecordValues();
1580:     }
1581: }
1582: //-----
1583:
1584: void __fastcall TMainForm::LoadAll(void)
1585: {
1586:     void *Result;
1587:
1588:     if(!Data.DataFileName.IsEmpty())
1589:     {
1590:         Screen->Cursor = crHourGlass;
1591:         Data.DataFile->Go(TOP);
1592:         Result = Data.DataFile->FreeSearch(1, NULL, NULL, NULL, NULL);
1593:         while(Result)
1594:         {
1595:             Data.Load();
1596:             AddListItems();
1597:             Result = Data.DataFile->FreeSearch(1, NULL, NULL, NULL, NULL, NEXT);
1598:         }
1599:
1600:         Application->Title = "بانک اطلاعات ترکیبات شیمیایی - [" + Data.DataFileName +
"]";
1601:         MainForm->Caption = Application->Title;
1602:         Screen->Cursor = crDefault;
1603:     }
1604: }
1605: //-----
1606:
1607:
1608: void __fastcall TMainForm::IUPACListClick(TObject *Sender)
1609: {
1610:     IUPACList->Checked = true;
1611:     TrivialList->Checked = false;
1612:     TradeList->Checked = false;
1613:     MolecularFormulaList->Checked = false;
1614:     ChemicalRegistryNumberList->Checked = false;
1615:
1616:     IUPACListP->Checked = true;
```

```
1617: TrivialListP->Checked = false;
1618: TradeListP->Checked = false;
1619: MolecularFormulaListP->Checked = false;
1620: ChemicalRegistryNumberListP->Checked = false;
1621:
1622: ClearRecordValues();
1623: DeleteAllListItems();
1624: ListItemsKind = IUPAC;
1625: StrCopy(BlankTitle, "[BLANK IUPAC NAME]");
1626: LoadAll();
1627: }
1628: //-----
1629:
1630: void __fastcall TMainForm::TrivialListClick(TObject *Sender)
1631: {
1632:     IUPACList->Checked = false;
1633:     TrivialList->Checked = true;
1634:     TradeList->Checked = false;
1635:     MolecularFormulaList->Checked = false;
1636:     ChemicalRegistryNumberList->Checked = false;
1637:
1638:     IUPACListP->Checked = false;
1639:     TrivialListP->Checked = true;
1640:     TradeListP->Checked = false;
1641:     MolecularFormulaListP->Checked = false;
1642:     ChemicalRegistryNumberListP->Checked = false;
1643:
1644:     ClearRecordValues();
1645:     DeleteAllListItems();
1646:     ListItemsKind = Trivial;
1647:     StrCopy(BlankTitle, "[BLANK TRIVIAL NAME]");
1648:     LoadAll();
1649: }
1650: //-----
1651:
1652: void __fastcall TMainForm::TradeListClick(TObject *Sender)
1653: {
1654:     IUPACList->Checked = false;
1655:     TrivialList->Checked = false;
1656:     TradeList->Checked = true;
1657:     MolecularFormulaList->Checked = false;
1658:     ChemicalRegistryNumberList->Checked = false;
1659:
1660:     IUPACListP->Checked = false;
1661:     TrivialListP->Checked = false;
1662:     TradeListP->Checked = true;
1663:     MolecularFormulaListP->Checked = false;
1664:     ChemicalRegistryNumberListP->Checked = false;
1665:
1666:     ClearRecordValues();
1667:     DeleteAllListItems();
1668:     ListItemsKind = Trade;
1669:     StrCopy(BlankTitle, "[BLANK TRADE NAME]");
1670:     LoadAll();
1671: }
1672: //-----
1673:
1674: void __fastcall TMainForm::MolecularFormulaListClick(TObject *Sender)
1675: {
1676:     IUPACList->Checked = false;
1677:     TrivialList->Checked = false;
1678:     TradeList->Checked = false;
1679:     MolecularFormulaList->Checked = true;
1680:     ChemicalRegistryNumberList->Checked = false;
1681:
1682:     IUPACListP->Checked = false;
1683:     TrivialListP->Checked = false;
1684:     TradeListP->Checked = false;
1685:     MolecularFormulaListP->Checked = true;
1686:     ChemicalRegistryNumberListP->Checked = false;
1687:
1688:     ClearRecordValues();
1689:     DeleteAllListItems();
1690:     ListItemsKind = MolecularFormula;
1691:     StrCopy(BlankTitle, "[BLANK MOLECULAR FORMULA]");
```

```
1692:     LoadAll();
1693: }
1694: //-----
1695:
1696: void __fastcall TMainForm::ChemicalRegistryNumberListClick(TObject *Sender)
1697: {
1698:     IUPACList->Checked = false;
1699:     TrivialList->Checked = false;
1700:     TradeList->Checked = false;
1701:     MolecularFormulaList->Checked = false;
1702:     ChemicalRegistryNumberList->Checked = true;
1703:
1704:     IUPACListP->Checked = false;
1705:     TrivialListP->Checked = false;
1706:     TradeListP->Checked = false;
1707:     MolecularFormulaListP->Checked = false;
1708:     ChemicalRegistryNumberListP->Checked = true;
1709:
1710:     ClearRecordValues();
1711:     DeleteAllListItems();
1712:     ListItemsKind = ChemicalRegistryNumber;
1713:     StrCopy(BlankTitle, "[BLANK CHEMICAL REGISTRY NUMBER]");
1714:     LoadAll();
1715: }
1716: //-----
1717:
1718: int __fastcall TMainForm::GetItemNodeAddress(int Index, bool OnlyPositive)
1719: {
1720:     int ReturnValue = 0;
1721:
1722:     if(Index < 0)
1723:         Index = ListView1->ItemIndex;
1724:
1725:     if(Index >= 0)
1726:         ReturnValue = (int)(ListView1->Items->Item[Index]->Data);
1727:
1728:     if(ReturnValue < 0 && OnlyPositive)
1729:         ReturnValue = -ReturnValue;
1730:
1731:     return ReturnValue;
1732: }
1733: //-----
1734:
1735: int __fastcall TMainForm::GetItemNodeAddress(TListItem *Item, bool OnlyPositive)
1736: {
1737:     int ReturnValue = (int)(Item->Data);
1738:     if(ReturnValue < 0 && OnlyPositive)
1739:         ReturnValue = -ReturnValue;
1740:
1741:     return ReturnValue;
1742: }
1743: //-----
1744:
1745: void __fastcall TMainForm::OptimizeClick(TObject *Sender)
1746: {
1747:     if(!Data.DataFileName.IsEmpty())
1748:     {
1749:         Screen->Cursor = crHourGlass;
1750:         Data.DataFile->Optimize();
1751:         Screen->Cursor = crDefault;
1752:         Warning->ShowWarning("اطلاعات باید دوباره خوانده شوند", "پیام", "تایید", "");
1753:         DeleteAllListItems();
1754:         LoadAll();
1755:
1756:         CurrentActivity = ShowRecord;
1757:         SetItemsActivity(CurrentActivity);
1758:         ClearRecordValues();
1759:     }
1760: }
1761: //-----
1762:
1763: void __fastcall TMainForm::DeleteListItem(int ItemIndex)
1764: {
1765:     if(ItemIndex < ListView1->Items->Count)
```

```
1766: {
1767:     ListView1->Items->Delete(ItemIndex);
1768: }
1769: }
1770: //-----
1771:
1772: void __fastcall TMainForm::DeleteAllListItems(bool OnlySelected)
1773: {
1774:     if(OnlySelected)
1775:         ListView1->DeleteSelected();
1776:     else
1777:     {
1778:         ListView1->Clear();
1779:         BlankList->Clear();
1780:     }
1781: }
1782: //-----
1783:
1784: int __fastcall TMainForm::SearchListForAddress(int Address, int StartItem)
1785: {
1786:     int ReturnValue = -1;
1787:     int CurrentAddress;
1788:
1789:     Screen->Cursor = crHourGlass;
1790:
1791:     MainProgressBar->Min = 0;
1792:     MainProgressBar->Max = ListView1->Items->Count;
1793:     MainProgressBar->Position = StartItem;
1794:
1795:     for(register i = StartItem; i < ListView1->Items->Count;
MainProgressBar->Position++, i++)
1796:     {
1797:         CurrentAddress = GetItemNodeAddress(ListView1->Items->Item[i]);
1798:         if(CurrentAddress == Address)
1799:         {
1800:             ReturnValue = i;
1801:             break;
1802:         }
1803:     }
1804:     Screen->Cursor = crDefault;
1805:
1806:     MainProgressBar->Position = 0;
1807:     return ReturnValue;
1808: }
1809: //-----
1810:
1811: void __fastcall TMainForm::ListView1Db1Click(TObject *Sender)
1812: {
1813:     if(ListView1->ItemIndex >= 0)
1814:     {
1815:         CurrentActivity = EditRecord;
1816:         SetItemsActivity(CurrentActivity);
1817:     }
1818: }
1819: //-----
1820:
1821: void __fastcall TMainForm::FindClick(TObject *Sender)
1822: {
1823:     Search->Show();
1824: }
1825: //-----
1826:
1827: void __fastcall TMainForm::Edit40KeyPress(TObject *Sender, char &Key)
1828: {
1829:     FloatEditorsKeyPress(Sender, Key);
1830: }
1831: //-----
1832:
1833: void __fastcall TMainForm::N41Click(TObject *Sender)
1834: {
1835:     ExportAll = true;
1836:     MakeISO2709();
1837: }
1838: //-----
1839:
```

```
1840: void __fastcall TMainForm::N40Click(TObject *Sender)
1841: {
1842:     ExportAll = false;
1843:     MakeISO2709();
1844: }
1845: //-----
1846:
1847: void __fastcall TMainForm::MakeISO2709(void)
1848: {
1849:     if(ListView1->Items->Count)
1850:         if(OpenDialog3->Execute())
1851:         {
1852:             ISO2709 ExportISO;
1853:             TStringList *List = new TStringList;
1854:             int LoadAddress;
1855:
1856:             if(CheckIfOverWriteAccepted(OpenDialog3->FileName.c_str()))
1857:             {
1858:                 Screen->Cursor = crHourGlass;
1859:
1860:                 ExportISO.DataFileName = OpenDialog3->FileName;
1861:                 DeleteFile(ExportISO.DataFileName);
1862:                 ExportISO.OpenDatabase();
1863:
1864:                 MainProgressBar->Min = 0;
1865:                 MainProgressBar->Max = ListView1->Items->Count;
1866:                 MainProgressBar->Position = 0;
1867:
1868:                 for(register i = 0; i < ListView1->Items->Count;
MainProgressBar->Position++, i++)
1869:                 {
1870:                     LoadAddress = GetItemNodeAddress(ListView1->Items->Item[i],
false);
1871:
1872:                     if(LoadAddress > 0 && (ExportAll ||
ListView1->Items->Item[i]->Selected))
1873:                     {
1874:                         Data.Load(LoadAddress);
1875:
1876:                         ExportISO.ResetRecord();
1877:
1878:                         ExportISO.AddField(1, Data.MainFields.IUPACName);
1879:                         ExportISO.AddField(40, Data.MainFields.IUPACTranslate);
1880:
1881:                         List->Text = Data.TrivialNames;
1882:                         for(register j = 0; j < List->Count; j++)
1883:                             ExportISO.AddField(2, List->Strings[j]);
1884:
1885:                         List->Text = Data.TradeNames;
1886:                         for(register j = 0; j < List->Count; j++)
1887:                             ExportISO.AddField(3, List->Strings[j]);
1888:
1889:                         ExportISO.AddField(4, Data.MainFields.MolecularFormula);
1890:                         ExportISO.AddField(5,
Data.MainFields.ChemicalRegistryNumber);
1891:                         ExportISO.AddField(6,
Data.PhysicochemicalProperties.ACXNumber);
1892:                         ExportISO.AddField(7,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.BoilingPoint));
1893:                         ExportISO.AddField(8,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.Density));
1894:                         ExportISO.AddField(9,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.DielectricConstant));
1895:                         ExportISO.AddField(10,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.DipoleMoment));
1896:                         ExportISO.AddField(11,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.EvaporationRate));
1897:                         ExportISO.AddField(12,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.FlashPoint));
1898:                         ExportISO.AddField(13,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.RefractiveIndex));
1899:                         ExportISO.AddField(14,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.MeltingPoint));
1900:                         ExportISO.AddField(15,
FloatToString(Data.MainFields.FixPhysicochemicalProperties.MolecularWeight));
```

```
1901:             ExportISO.AddField(16,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.MolarRefractivity));  
1902:             ExportISO.AddField(17,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.MolarVolume));  
1903:             ExportISO.AddField(18,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.PKA));  
1904:             ExportISO.AddField(19,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.Polarizability));  
1905:             ExportISO.AddField(20,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.RTECS));  
1906:             ExportISO.AddField(21,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.SurfaceTension));  
1907:             ExportISO.AddField(22,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.VaporDensity));  
1908:             ExportISO.AddField(23,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.VaporPressure));  
1909:             ExportISO.AddField(24,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.Viscosity));  
1910:             ExportISO.AddField(25,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.WaterSolubility));  
1911:             ExportISO.AddField(26,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.  
ThermodynamicStandarProperties.H));  
1912:             ExportISO.AddField(27,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.  
ThermodynamicStandarProperties.S));  
1913:             ExportISO.AddField(28,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.  
ThermodynamicStandarProperties.G));  
1914:             ExportISO.AddField(29,  
FloatToString(Data.MainFields.FixPhysicochemicalProperties.  
ThermodynamicStandarProperties.C));  
1915:             ExportISO.AddField(30,  
Data.PhysicochemicalProperties.Composition);  
1916:             ExportISO.AddField(31,  
Data.PhysicochemicalProperties.Color);  
1917:             ExportISO.AddField(32,  
Data.PhysicochemicalProperties.CrystallineForm);  
1918:             ExportISO.AddField(33,  
Data.PhysicochemicalProperties.EPACode);  
1919:             ExportISO.AddField(34,  
Data.PhysicochemicalProperties.Odor);  
1920:             ExportISO.AddField(35,  
Data.PhysicochemicalProperties.DOTNumber);  
1921:             ExportISO.AddField(36,  
Data.PhysicochemicalProperties.PhysicalState);  
1922:             ExportISO.AddField(37,  
Data.SymmetricProperties.PointGroup);  
1923:             ExportISO.AddField(38,  
Data.SymmetricProperties.SymmetryElements);  
1924:             ExportISO.AddField(39, Data.Comments);  
1925:  
1926:             ExportISO.AddRecord();  
1927:         }  
1928:     }  
1929:  
1930:     Screen->Cursor = crDefault;  
1931: }  
1932:  
1933:     delete List;  
1934: }  
1935:  
1936:     MainProgressBar->Position = 0;  
1937: }  
1938: //-----  
1939:  
1940: void __fastcall TMainForm::SaveAs(int WhichRecords)  
1941: {  
1942:     if(!Data.DataFileName.IsEmpty())  
1943:     {  
1944:         if(OpenDialog2->Execute())  
1945:         {  
1946:             AnsiString SaveAsFileName = OpenDialog2->FileName;  
1947:  
1948:             if(SaveAsFileName != Data.DataFileName)
```

```
1949:         {
1950:             if(CheckIfOverWriteAccepted(SaveAsFileName.c_str()))
1951:             {
1952:                 void *Result;
1953:                 int LoadAddress;
1954:
1955:                 switch(WhichRecords)
1956:                 {
1957:                     case saAllDatabase:
1958:                     {
1959:                         TFileStream *SaveFile = new
TFileStream(SaveAsFileName, fmCreate);
1960:                         Screen->Cursor = crHourGlass;
1961:                         SaveFile->CopyFrom(Data.DataFile->File, 0);
1962:                         Screen->Cursor = crDefault;
1963:                         delete SaveFile;
1964:
1965:                         Data.CloseDatabase();
1966:                         Data.DataFileName = SaveAsFileName;
1967:                         Data.OpenDatabase();
1968:                         DeleteAllListItems();
1969:                         LoadAll();
1970:                         CurrentActivity = ShowRecord;
1971:                         SetItemsActivity(CurrentActivity);
1972:                         ClearRecordValues();
1973:                     }
1974:                     break;
1975:
1976:                     case saSelected:
1977:                     case saCurrentList:
1978:                     {
1979:                         CompoundData SaveAsData;
1980:
1981:                         Screen->Cursor = crHourGlass;
1982:                         SaveAsData.DataFileName = SaveAsFileName;
1983:                         DeleteFile(SaveAsData.DataFileName);
1984:                         SaveAsData.OpenDatabase();
1985:
1986:                         Data.DataFile->SetHeaderProviderVersion(ApplicationDatabaseProvider,
ApplicationVersion);
1987:
1988:                         MainProgressBar->Min = 0;
1989:                         MainProgressBar->Max = ListView1->Items->Count;
1990:                         MainProgressBar->Position = 0;
1991:
1992:                         for(register i = 0; i < ListView1->Items->Count
MainProgressBar->Position++, i += 100)
1993:                         {
1994:                             LoadAddress =
GetItemNodeAddress(ListView1->Items->Item[i], false);
1995:
1996:                             if(LoadAddress > 0 &&
(ListView1->Items->Item[i]->Selected || WhichRecords == saCurrentList))
1997:                             {
1998:                                 Data.Load(LoadAddress);
1999:                                 SaveAsData.Assign(&Data);
2000:                                 SaveAsData.Save(false);
2001:                             }
2002:                         }
2003:
2004:                         for(register i = 1; i < ListView1->Items->Count
MainProgressBar->Position++, i++)
2005:                         {
2006:                             LoadAddress =
GetItemNodeAddress(ListView1->Items->Item[i], false);
2007:
2008:                             if(!i%100)
2009:                                 MainProgressBar->Position--;
2010:
2011:                             if(LoadAddress > 0 && (i%100) &&
(ListView1->Items->Item[i]->Selected || WhichRecords == saCurrentList))
2012:                             {
2013:                                 Data.Load(LoadAddress);
2014:                                 SaveAsData.Assign(&Data);
2015:                                 SaveAsData.Save(false);
2016:                             }
2017:                         }
2018:                     }
2019:                 }
2020:             }
2021:         }
2022:     }
2023: }
```

```

2015:         }
2016:     }
2017:
2018:     SaveAsData.CloseDatabase();
2019:
2020:     Data.CloseDatabase();
2021:     Data.DataFileName = SaveAsFileName;
2022:     Data.OpenDatabase();
2023:
2024:     Screen->Cursor = crDefault;
2025:
2026:     DeleteAllListItems();
2027:     LoadAll();
2028:     CurrentActivity = ShowRecord;
2029:     SetItemsActivity(CurrentActivity);
2030:     ClearRecordValues();
2031:     }
2032:     break;
2033: }
2034:
2035:     }
2036: }
2037: else
2038:     Warning->ShowWarning("خطا", "نام بانک اطلاعات جدید نمی‌تواند همنام نام بانک اطلاعات فعلی باشد", "تایید", "", "");
2039: }
2040: }
2041: MainProgressBar->Position = 0;
2042: }
2043: //-----
2044:
2045: void __fastcall TMainForm::N42Click(TObject *Sender)
2046: {
2047:     SaveAs(saAllDatabase);
2048: }
2049: //-----
2050:
2051: void __fastcall TMainForm::N43Click(TObject *Sender)
2052: {
2053:     SaveAs(saSelected);
2054: }
2055: //-----
2056:
2057: void __fastcall TMainForm::N44Click(TObject *Sender)
2058: {
2059:     SaveAs(saCurrentList);
2060: }
2061: //-----
2062:
2063:
2064: void __fastcall TMainForm::N10Click(TObject *Sender)
2065: {
2066:     ShellExecute(Handle, "open", "http://chemfinder.cambridgesoft.com", NULL, NULL,
        SW_SHOWNORMAL);
2067: }
2068: //-----
2069:
2070: void __fastcall TMainForm::N11Click(TObject *Sender)
2071: {
2072:     MainProgressBar->Min = 0;
2073:     MainProgressBar->Max = ListView1->Items->Count;
2074:     MainProgressBar->Position = 0;
2075:
2076:     for(register i = 0; i < ListView1->Items->Count; MainProgressBar->Position++, i++)
2077:         ListView1->Items->Item[i]->Selected = ListView1->Items->Item[i]->Selected ?
        false : true;
2078:
2079:     MainProgressBar->Position = 0;
2080: }
2081: //-----
2082:
2083: void __fastcall TMainForm::FormShow(TObject *Sender)
2084: {
2085:     bool CheckVersion = false;
2086:

```

```

2087: Application->ProcessMessages();
2088:
2089: if(ParamCount() > 1)
2090: {
2091:     if(FileExists(ParamStr(1)))
2092:     {
2093:         Data.CloseDatabase();
2094:         Data.DataFileName = ParamStr(1);
2095:         Data.OpenDatabase();
2096:         if(!(CheckVersion = Data.CheckForCorrectVersion()))
2097:         {
2098:             Data.CloseDatabase();
2099:             Warning->ShowWarning("خطا", "اطلاعات مربوط به فایل مورد نظر قابل", "تایید", "", "");
2100:             , "بازیابی نمی‌باشد"
2101:         }
2102:     }
2103:     if(!Data.DataFileName.IsEmpty())
2104:     {
2105:         DeleteAllListItems();
2106:         if(CheckVersion)
2107:             LoadAll();
2108:
2109:         CurrentActivity = NoRecord;
2110:         SetItemsActivity(CurrentActivity);
2111:         ClearRecordValues();
2112:     }
2113: }
2114: }
2115: //-----
2116:
2117: void __fastcall TMainForm::FormKeyDown(TObject *Sender, WORD &Key,
2118:     TShiftState Shift)
2119: {
2120:     if(Key == VK_DELETE)
2121:     {
2122:         if(TabSheet3->Focused())
2123:             PaintTool->DeleteAllShapes(true);
2124:     }
2125: }
2126: //-----
2127:
2128: void __fastcall TMainForm::Timer1Timer(TObject *Sender)
2129: {
2130:     Application->ProcessMessages();
2131:     Timer1->Interval = 1500;
2132: }
2133: //-----
2134:
2135: void __fastcall TMainForm::NewClick(TObject *Sender)
2136: {
2137:     if(OpenDialog2->Execute())
2138:     {
2139:         if(CheckIfOverWriteAccepted(OpenDialog2->FileName.c_str()))
2140:         {
2141:             Data.CloseDatabase();
2142:             Data.DataFileName = OpenDialog2->FileName;
2143:             DeleteFile(Data.DataFileName);
2144:             Data.OpenDatabase();
2145:             Data.DataFile->SetHeaderProviderVersion(ApplicationDatabaseProvider,
ApplicationVersion);
2146:
2147:             Application->Title = " - بانک اطلاعات ترکیبات شیمیایی" +
Data.DataFileName + ".txt";
2148:             MainForm->Caption = Application->Title;
2149:         }
2150:     }
2151:
2152:     if(!Data.DataFileName.IsEmpty())
2153:     {
2154:         DeleteAllListItems();
2155:         CurrentActivity = NoRecord;
2156:         SetItemsActivity(CurrentActivity);
2157:         ClearRecordValues();
2158:     }

```

```
2159: }
2160: //-----
2161:
2162: void __fastcall TMainForm::ToolButton7Click(TObject *Sender)
2163: {
2164:     SaveButtonClick(Sender);
2165: }
2166: //-----
2167:
2168: void __fastcall TMainForm::ToolButton11Click(TObject *Sender)
2169: {
2170:     DeleteListClick(Sender);
2171: }
2172: //-----
2173:
2174: void __fastcall TMainForm::httpwwwchemexpercom1Click(TObject *Sender)
2175: {
2176:     ShellExecute(Handle, "open", "http://www.chemexper.com", NULL, NULL, SW_SHOWNORMAL);
2177: }
2178: //-----
2179:
2180: void __fastcall TMainForm::httpwwwcascom1Click(TObject *Sender)
2181: {
2182:     ShellExecute(Handle, "open", "http://www.cas.com", NULL, NULL, SW_SHOWNORMAL);
2183: }
2184: //-----
2185:
2186: void __fastcall TMainForm::httpwwwchemwebcom1Click(TObject *Sender)
2187: {
2188:     ShellExecute(Handle, "open", "http://www.chemweb.com", NULL, NULL, SW_SHOWNORMAL);
2189: }
2190: //-----
2191:
2192: void __fastcall TMainForm::httpwwwacdlabscom1Click(TObject *Sender)
2193: {
2194:     ShellExecute(Handle, "open", "http://www.acdlabs.com", NULL, NULL, SW_SHOWNORMAL);
2195: }
2196: //-----
2197:
2198:
```

```
1: //-----
2:
3: #include <vcl.h>
4: #pragma hdrstop
5:
6: #include "Main.h"
7: #include "PaintTools.h"
8: #include "AtomTextEditForm.h"
9: //-----
10: #pragma package(smart_init)
11: #pragma resource "*.dfm"
12: TForm1 *Form1;
13:
14: template<class T>
15: int AsInt(T Param)
16: {
17:     return static_cast<int>(Param);
18: }
19: //-----
20:
21: __fastcall TCharsetObject::TCharsetObject(int FCharset): TObject()
22: {
23:     Charset = FCharset;
24: }
25: //-----
26:
27: __fastcall TForm1::TForm1(TComponent* Owner)
28:     : TForm(Owner)
29: {
30:     GetFontNames();
31:     FontName->Text = MainForm->Font->Name;
32:     FontSize->Text = "12";
33: }
34: //-----
35: void __fastcall TForm1::AtomTextEditKeyPress(TObject *Sender, char &Key)
36: {
37:     if(Key == VK_RETURN)
38:     {
39:         MainForm->PaintTool->HideAtomTextEdit();
40:     }
41:
42:     if(Key == VK_ESCAPE)
43:         MainForm->PaintTool->HideAtomTextEdit(false);
44: }
45: //-----
46:
47: int __stdcall EnumFontsProc(TLogFontA &LogFont, TTextMetricA & /*TextMetric*/,
48:                             int /*FontType*/, Pointer Data)
49: {
50:     TCharsetObject *FontCharset;
51:     TStrings *Strings = static_cast<TStrings *>(Data);
52:     FontCharset = new TCharsetObject(AsInt(LogFont.lfCharSet));
53:     Strings->AddObject(LogFont.lfFaceName, FontCharset);
54:
55:     return 1;
56: }
57: //-----
58:
59: void __fastcall TForm1::GetFontNames(void)
60: {
61:     HDC hDC = GetDC(0);
62:     void * cTmp = static_cast<void *>(Form1->FontName->Items);
63:     EnumFonts(hDC, NULL, (FONTENUMPROC) EnumFontsProc, (long) cTmp );
64:     ReleaseDC(0,hDC);
65:     Form1->FontName->Sorted = True;
66: }
67: //-----
68:
69: void __fastcall TForm1::FormDestroy(TObject *Sender)
70: {
71:     for(int n = 0;n < Form1->FontName->Items->Count;n++)
72:         if(Form1->FontName->Items->Objects[n])
73:         {
74:             delete Form1->FontName->Items->Objects[n];
75:             Form1->FontName->Items->Objects[n] = NULL;
```

```
76:     }
77: }
78: //-----
79: void __fastcall TForm1::FontNameChange(TObject *Sender)
80: {
81:     TCharsetObject* ChasrsetObject;
82:
83:     int FontIndex = Form1->FontName->ItemIndex;
84:
85:     if((FontIndex > -1) && (FontIndex < Form1->FontName->Items->Count))
86:     {
87:         ChasrsetObject =
dynamic_cast<TCharsetObject*>(Form1->FontName->Items->Objects[Form1->FontName->ItemIndex]);
88:         MainForm->PaintTool->AtomTextFont->Charset = ChasrsetObject->Charset;
89:         MainForm->PaintTool->AtomTextFont->Name =
Form1->FontName->Items->Strings[Form1->FontName->ItemIndex];
90:     }
91:
92:     MainForm->PaintTool->AtomTextFont->Size = StrToInt(Form1->FontSize->Text);
93:
94:     if(SpeedButton1->Down)
95:         MainForm->PaintTool->AtomTextFont->Style =
MainForm->PaintTool->AtomTextFont->Style << fsBold;
96:     else
97:         MainForm->PaintTool->AtomTextFont->Style =
MainForm->PaintTool->AtomTextFont->Style >> fsBold;
98:
99:     if(SpeedButton2->Down)
100:         MainForm->PaintTool->AtomTextFont->Style =
MainForm->PaintTool->AtomTextFont->Style << fsItalic;
101:     else
102:         MainForm->PaintTool->AtomTextFont->Style =
MainForm->PaintTool->AtomTextFont->Style >> fsItalic;
103:
104:     if(SpeedButton3->Down)
105:         MainForm->PaintTool->AtomTextFont->Style =
MainForm->PaintTool->AtomTextFont->Style << fsUnderline;
106:     else
107:         MainForm->PaintTool->AtomTextFont->Style =
MainForm->PaintTool->AtomTextFont->Style >> fsUnderline;
108:
109:     AtomTextEdit->Font->Assign(MainForm->PaintTool->AtomTextFont);
110:     AtomTextEdit->Font->Size = 10;
111:     Panel2->Font->Assign(MainForm->PaintTool->AtomTextFont);
112: }
113: //-----
114:
115: void __fastcall TForm1::FontSizeKeyPress(TObject *Sender, char &Key)
116: {
117:     if((Key < 48 || Key > 57) && Key != VK_BACK && Key != '.')
118:         Key = 0;
119:     else
120:         AtomTextEditKeyPress(Sender, Key);
121: }
122: //-----
123:
124: void __fastcall TForm1::Button1Click(TObject *Sender)
125: {
126:     MainForm->PaintTool->HideAtomTextEdit();
127: }
128: //-----
129:
130: void __fastcall TForm1::FormShow(TObject *Sender)
131: {
132:     FontName->Text = MainForm->PaintTool->AtomTextFont->Name;
133:     FontSize->Text = IntToStr(MainForm->PaintTool->AtomTextFont->Size);
134:
135:     SpeedButton1->Down = MainForm->PaintTool->AtomTextFont->Style.Contains(fsBold);
136:     SpeedButton2->Down = MainForm->PaintTool->AtomTextFont->Style.Contains(fsItalic);
137:     SpeedButton3->Down = MainForm->PaintTool->AtomTextFont->Style.Contains(fsUnderline);
138: }
139: //-----
140:
141: void __fastcall TForm1::AtomTextEditChange(TObject *Sender)
```

```
142: {  
143:     Panel2->Caption = AtomTextEdit->Text;  
144: }  
145: //-----  
146:
```

```

1: #include "math.h"
2:
3:
4: #include <Main.h>
5: #include <PaintTools.h>
6: #include <AtomTextEditForm.h>
7:
8: //-----
9:
10: __fastcall PaintTools::PaintTools(TImage *ImageWorking)
11: {
12:     CurrentActivation = SHAPESELECTION;
13:     Image = ImageWorking;
14:     Drawing = false;
15:     MoveSelVertex = false;
16:     MoveSelShape = false;
17:     PenMode = pmCopy;
18:     ShapeCounter = GroupCounter = 0;
19:     AtomTextFont = new TFont;
20:     ActivePolygonSideNumber = 6;
21:
22:     AtomTextFont->Assign(MainForm->Font);
23:
24:     Image->ControlStyle = Image->ControlStyle << csOpaque;
25:     Image->Canvas->MoveTo(0, 0);
26:     Image->Canvas->Brush->Style = bsClear;
27: }
28: //-----
29:
30: __fastcall PaintTools::~PaintTools(void)
31: {
32:     DeleteAllShapes();
33:     delete AtomTextFont;
34: }
35: //-----
36:
37: void __fastcall PaintTools::HideAtomTextEdit(bool AddData)
38: {
39:     if(AddData && !Form1->AtomTextEdit->Text.IsEmpty())
40:         AddAtomText();
41:
42:     Drawing = false;
43:     Form1->Close();
44: }
45: //-----
46:
47: void __fastcall PaintTools::AddAtomText(void)
48: {
49:     int SizeBak;
50:
51:     GData[ShapeCounter] = new GraphData;
52:
53:     GData[ShapeCounter]->GroupNumber = GroupCounter;
54:     GData[ShapeCounter]->GroupKind = ATOMTEXT;
55:
56:     GData[ShapeCounter]->ShapeKind = ATOMTEXT;
57:     GData[ShapeCounter]->Shape.AtomText.Position.x = StartPoint.x;
58:     GData[ShapeCounter]->Shape.AtomText.Position.y = StartPoint.y;
59:     GData[ShapeCounter]->Shape.AtomText.Position.Angle = 0;
60:     GData[ShapeCounter]->Pen = PenAssign(Image->Canvas->Pen);
61:
62:     GData[ShapeCounter]->Shape.AtomText.Width = DrawAtomText(AtomTextFont,
63:                                                             &TPoint(StartPoint.x,
64: StartPoint.y),
65:                                                             Form1->AtomTextEdit->Text
66: false);
67:
68:     SizeBak = AtomTextFont->Size;
69:     Image->Canvas->Font->Assign(AtomTextFont);
70:     GData[ShapeCounter]->Shape.AtomText.Height =
71: Image->Canvas->TextHeight(Form1->AtomTextEdit->Text);
72:     Image->Canvas->Font->Size -= Image->Canvas->Font->Size * 3 / 10;
73:     GData[ShapeCounter]->Shape.AtomText.Height +=
74: Image->Canvas->TextHeight(Form1->AtomTextEdit->Text) / 3;
75:     StrCopy(GData[ShapeCounter]->Shape.AtomText.Text,

```

```
Form1->AtomTextEdit->Text.c_str());
72:   Form1->AtomTextEdit->Text = "";
73:   Image->Canvas->Font->Size = SizeBak;
74:   GData[ShapeCounter]->Shape.AtomText.TextForm = false;
75:   StrCopy(GData[ShapeCounter]->Shape.AtomText.Font.Name, AtomTextFont->Name.c_str())
76:   GData[ShapeCounter]->Shape.AtomText.Font.Color = AtomTextFont->Color;
77:   GData[ShapeCounter]->Shape.AtomText.Font.CharSet = AtomTextFont->Charset;
78:   GData[ShapeCounter]->Shape.AtomText.Font.Height = AtomTextFont->Height;
79:   GData[ShapeCounter]->Shape.AtomText.Font.Pitch = AtomTextFont->Pitch;
80:   GData[ShapeCounter]->Shape.AtomText.Font.Size = AtomTextFont->Size;
81:   GData[ShapeCounter]->Shape.AtomText.Font.Style = 0;
82:   GData[ShapeCounter]->Shape.AtomText.Font.Style |=
AtomTextFont->Style.Contains(fsBold) ? 1 : 0;
83:   GData[ShapeCounter]->Shape.AtomText.Font.Style |=
AtomTextFont->Style.Contains(fsItalic) ? 2 : 0;
84:   GData[ShapeCounter]->Shape.AtomText.Font.Style |=
AtomTextFont->Style.Contains(fsUnderline) ? 4 : 0;
85:
86:   SelectShape(ShapeCounter, true);
87:
88:   ShapeCounter++;
89:   GroupCounter++;
90: }
91: //-----
92:
93: int __fastcall PaintTools::DrawAtomText(TFont *AtomFont, TPoint *Position, AnsiString
Text, bool TextForm)
94: {
95:   int Pos = Position->x;
96:   int Character;
97:   int Length = Text.Length();
98:   int HeightDistance;
99:   AnsiString Str;
100:  int NormalSize, SmallSize;
101:
102:  Image->Canvas->Font->Assign(AtomFont);
103:  HeightDistance = Image->Canvas->TextHeight(Text) * 2 / 3;
104:  NormalSize = Image->Canvas->Font->Size;
105:  SmallSize = Image->Canvas->Font->Size - Image->Canvas->Font->Size * 3 / 10;
106:
107:  for(register i = 0; i < Length; i++)
108:  {
109:    Character = Text.c_str()[i];
110:    Str = Text.SubString(i + 1, 1);
111:
112:    if(iswdigit(Character) && !TextForm)
113:    {
114:      Image->Canvas->Font->Size = SmallSize;
115:      Image->Canvas->TextOut(Pos, Position->y + HeightDistance, Str);
116:      Pos += Image->Canvas->TextWidth(Str);
117:    }
118:    else
119:    {
120:      Image->Canvas->Font->Size = NormalSize;
121:      Image->Canvas->TextOut(Pos, Position->y, Str);
122:      Pos += Image->Canvas->TextWidth(Str);
123:    }
124:  }
125:  return Pos - Position->x;
126: }
127: //-----
128:
129: PenSpec __fastcall PaintTools::PenAssign(TPen *Pen)
130: {
131:   PenSpec PenSpecification = {Pen->Color, Pen->Mode, Pen->Style, Pen->Width};
132:   return PenSpecification;
133: }
134: //-----
135:
136: BrushSpec __fastcall PaintTools::BrushAssign(TBrush *Brush)
137: {
138:   BrushSpec BrushSpecification = {Brush->Color, Brush->Style};
139:   return BrushSpecification;
140: }
141: //-----
```

```
142:
143: void __fastcall PaintTools::SelectShape(int ShapeNum, bool DeSelectOther, bool ReDraw)
144: {
145:     if(ShapeNum > -1)
146:     {
147:         switch(GData[ShapeNum]->ShapeKind)
148:         {
149:             case DRAWCONTINUOUSLINE:
150:             case DRAWLINE:
151:             case DRAWCHAIN:
152:             case DRAWPOLYGON:
153:             case DRAWCYCLOPENTANE:
154:             case DRAWARROW:
155:             case DRAWARROW2SIDE:
156:             case DRAWARROW2SIDESEP:
157:             case DRAWDUBLEBOND:
158:             case DRAWTRIPLEBOND:
159:             case DRAWUPWEDGE:
160:             case DRAWDOWNWEDGE:
161:                 GData[ShapeNum]->Shape.Line.Start.Selected = true;
162:                 GData[ShapeNum]->Shape.Line.End.Selected = true;
163:                 break;
164:
165:             case ATOMTEXT:
166:                 GData[ShapeNum]->Shape.AtomText.Position.Selected = true;
167:                 break;
168:
169:             case DRAWRECTANGLE:
170:                 GData[ShapeNum]->Shape.Rectangle.TopLeft.Selected = true;
171:                 GData[ShapeNum]->Shape.Rectangle.BottomRight.Selected = true;
172:                 GData[ShapeNum]->Shape.Rectangle.TopRight.Selected = true;
173:                 GData[ShapeNum]->Shape.Rectangle.BottomLeft.Selected = true;
174:                 break;
175:
176:             case DRAWCIRCULARARC:
177:             case DRAWCIRCULARARROW:
178:             case DRAWCIRCULARARROW2SIDE:
179:                 GData[ShapeNum]->Shape.CircularArc.Start.Selected = true;
180:                 GData[ShapeNum]->Shape.CircularArc.End.Selected = true;
181:                 GData[ShapeNum]->Shape.CircularArc.Mid.Selected = true;
182:                 break;
183:         }
184:         GData[ShapeNum]->Pen.Color = (TColor)SELECTCOLOR;
185:     }
186:
187:     if(ReDraw)
188:         DrawShape(ShapeNum);
189:
190:     if(DeSelectOther)
191:     {
192:         for(register i = 0; i < ShapeCounter; i++)
193:         {
194:             if(i != ShapeNum)
195:                 DeselectShape(i, false);
196:         }
197:         if(ReDraw)
198:             DrawShapes();
199:     }
200:
201:     if(ShapeNum > -1)
202:     {
203:         switch(GData[ShapeNum]->ShapeKind)
204:         {
205:             case DRAWCONTINUOUSLINE:
206:             case DRAWCHAIN:
207:             case DRAWPOLYGON:
208:             case DRAWCYCLOPENTANE:
209:                 for(register i = 0; i < ShapeCounter; i++)
210:                 {
211:                     if(GData[i]->GroupNumber == GData[ShapeNum]->GroupNumber)
212:                     {
213:                         if(ComparePoints(&TPoint(GData[i]->Shape.Line.Start.x,
214: GData[i]->Shape.Line.Start.y),
215:                                     &TPoint(GData[ShapeNum]->Shape.Line.End.x,
216: GData[ShapeNum]->Shape.Line.End.y)) ||
```

```

215:         ComparePoints(&TPoint(GData[i]->Shape.Line.Start.x,
GData[i]->Shape.Line.Start.y),
216:         &TPoint(GData[ShapeNum]->Shape.Line.Start.x,
GData[ShapeNum]->Shape.Line.Start.y)))
217:         GData[i]->Shape.Line.Start.Selected = true;
218:
219:         if(ComparePoints(&TPoint(GData[i]->Shape.Line.End.x,
GData[i]->Shape.Line.End.y),
220:         &TPoint(GData[ShapeNum]->Shape.Line.Start.x,
GData[ShapeNum]->Shape.Line.Start.y)) ||
221:         ComparePoints(&TPoint(GData[i]->Shape.Line.End.x,
GData[i]->Shape.Line.End.y),
222:         &TPoint(GData[ShapeNum]->Shape.Line.End.x,
GData[ShapeNum]->Shape.Line.End.y)))
223:         GData[i]->Shape.Line.End.Selected = true;
224:     }
225: }
226: }
227: }
228: }
229: //-----
230:
231: void __fastcall PaintTools::DeselectShape(int ShapeNum, bool ReDraw)
232: {
233:     if(ReDraw)
234:     {
235:         PenMode = pmNotXor;
236:         DrawShape(ShapeNum);
237:     }
238:
239:     GData[ShapeNum]->Pen.Color = (TColor)NORMALCOLOR;
240:
241:     switch(GData[ShapeNum]->ShapeKind)
242:     {
243:         case DRAWCONTINUOUSLINE:
244:         case DRAWCHAIN:
245:         case DRAWPOLYGON:
246:         case DRAWCYCLOPENTANE:
247:             for(register i = 0; i < ShapeCounter; i++)
248:                 if(GData[i]->GroupNumber == GData[ShapeNum]->GroupNumber &&
!IsShapeSelected(i))
249:                 {
250:                     GData[i]->Shape.Line.Start.Selected = false;
251:                     GData[i]->Shape.Line.End.Selected = false;
252:                 }
253:         case DRAWLINE:
254:         case DRAWARROW:
255:         case DRAWARROW2SIDE:
256:         case DRAWARROW2SIDESEP:
257:         case DRAWCIRCLE:
258:         case DRAWDOUBLEBOND:
259:         case DRAWTRIPLEBOND:
260:         case DRAWUPWEDGE:
261:         case DRAWDOWNWEDGE:
262:             GData[ShapeNum]->Shape.Line.Start.Selected = false;
263:             GData[ShapeNum]->Shape.Line.End.Selected = false;
264:             break;
265:
266:         case ATOMTEXT:
267:             GData[ShapeNum]->Shape.AtomText.Position.Selected = false;
268:             break;
269:
270:         case DRAWRECTANGLE:
271:             GData[ShapeNum]->Shape.Rectangle.TopLeft.Selected = false;
272:             GData[ShapeNum]->Shape.Rectangle.BottomRight.Selected = false;
273:             GData[ShapeNum]->Shape.Rectangle.TopRight.Selected = false;
274:             GData[ShapeNum]->Shape.Rectangle.BottomLeft.Selected = false;
275:             break;
276:
277:         case DRAWCIRCULARARC:
278:         case DRAWCIRCULARARROW:
279:         case DRAWCIRCULARARROW2SIDE:
280:             GData[ShapeNum]->Shape.CircularArc.Start.Selected = false;
281:             GData[ShapeNum]->Shape.CircularArc.End.Selected = false;
282:             GData[ShapeNum]->Shape.CircularArc.Mid.Selected = false;

```

```
283:         break;
284:     }
285:
286:     if (ReDraw)
287:     {
288:         PenMode = pmCopy;
289:         DrawShape(ShapeNum);
290:     }
291: }
292: //-----
293:
294: void __fastcall PaintTools::DrawShape(int ShapeNum)
295: {
296:     int StartX = StartPoint.x, StartY = StartPoint.y;
297:     int EndX = MovePoint.x, EndY = MovePoint.y;
298:     int MidX;
299:     int MidY;
300:     int ShapeKind = ShapeNum > -1 ? GData[ShapeNum]->ShapeKind : CurrentActivation;
301:     bool StartSelected = false;
302:     bool EndSelected = false;
303:     bool MidSelected = false;
304:     Image->Canvas->Pen->Mode = PenMode;
305:     Image->Canvas->Pen->Color = (TColor)NORMALCOLOR;
306:
307:     switch(ShapeKind)
308:     {
309:         case DRAWLINE:
310:         case DRAWCONTINUOUSLINE:
311:         case DRAWCHAIN:
312:         case DRAWPOLYGON:
313:         case DRAWCYCLOPENTANE:
314:         case DRAWCIRCLE:
315:         case DRAWARROW:
316:         case DRAWARROW2SIDE:
317:         case DRAWARROW2SIDESEP:
318:         case DRAWDUBLEBOND:
319:         case DRAWTRIPLEBOND:
320:         case DRAWUPWEDGE:
321:         case DRAWDOWNWEDGE:
322:             if(ShapeNum > -1)
323:             {
324:                 StartX = GData[ShapeNum]->Shape.Line.Start.x;
325:                 StartY = GData[ShapeNum]->Shape.Line.Start.y;
326:                 EndX = GData[ShapeNum]->Shape.Line.End.x;
327:                 EndY = GData[ShapeNum]->Shape.Line.End.y;
328:                 StartSelected = GData[ShapeNum]->Shape.Line.Start.Selected;
329:                 EndSelected = GData[ShapeNum]->Shape.Line.End.Selected;
330:             }
331:             if(StartSelected && EndSelected)
332:                 Image->Canvas->Pen->Color = GData[ShapeNum]->Pen.Color;
333:
334:             if(ShapeKind == DRAWCHAIN && ShapeNum < 0)
335:                 DrawChain(&TPoint(StartX, StartY), &TPoint(EndX, EndY));
336:             else if((ShapeKind == DRAWPOLYGON || ShapeKind == DRAWCYCLOPENTANE) &&
ShapeNum < 0)
337:                 DrawPolygon(&TPoint(StartX, StartY), &TPoint(EndX, EndY), ShapeKind);
338:             else if(ShapeKind == DRAWCIRCLE)
339:                 DrawCircle(&TPoint(StartX, StartY), &TPoint(EndX, EndY));
340:             else if(ShapeKind == DRAWUPWEDGE || ShapeKind == DRAWDOWNWEDGE)
341:                 DrawWedge(&TPoint(StartX, StartY), &TPoint(EndX, EndY), ShapeKind);
342:             else if(ShapeKind == DRAWARROW2SIDESEP)
343:             {
344:                 DrawBond(&TPoint(StartX, StartY), &TPoint(EndX, EndY), 2, false);
345:                 DrawArrow(&ArrayPoints[3], &ArrayPoints[2], true);
346:                 DrawArrow(&TPoint(StartX, StartY), &TPoint(EndX, EndY), true);
347:             }
348:             else
349:             {
350:                 if(ShapeKind == DRAWARROW || ShapeKind == DRAWARROW2SIDE)
351:                     DrawArrow(&TPoint(StartX, StartY), &TPoint(EndX, EndY));
352:                 if(ShapeKind == DRAWARROW2SIDE)
353:                     DrawArrow(&TPoint(EndX, EndY), &TPoint(StartX, StartY));
354:
355:                 DrawBond(&TPoint(StartX, StartY), &TPoint(EndX, EndY),
ShapeNum > -1 ? GData[ShapeNum]->Shape.Line.BondNumber :
```

```
ShapeKind == DRAWDOUBLEBOND ? 2 : ShapeKind == DRAWTRIPLEBOND ? 3 : 1,
357: ShapeNum > -1 ? GData[ShapeNum]->Shape.Line.LeftSideBond :
358: false);
359: }
360: if(StartSelected)
361:     DrawVertex(TPoint(StartX, StartY));
362: if(EndSelected)
363:     DrawVertex(TPoint(EndX, EndY));
364: break;
365:
366: case DRAWRECTANGLE:
367:     if(ShapeNum > -1)
368:     {
369:         StartX = GData[ShapeNum]->Shape.Line.Start.x;
370:         StartY = GData[ShapeNum]->Shape.Line.Start.y;
371:         EndX = GData[ShapeNum]->Shape.Line.End.x;
372:         EndY = GData[ShapeNum]->Shape.Line.End.y;
373:         StartSelected = GData[ShapeNum]->Shape.Line.Start.Selected;
374:         EndSelected = GData[ShapeNum]->Shape.Line.End.Selected;
375:     }
376:
377:     if(StartSelected && EndSelected)
378:         Image->Canvas->Pen->Color = GData[ShapeNum]->Pen.Color;
379:
380:     DrawRectangle(&TPoint(StartX, StartY), &TPoint(StartX, EndY),
381: &TPoint(EndX, StartY), &TPoint(EndX, EndY));
382:     if(StartSelected && EndSelected)
383:     {
384:         DrawVertex(TPoint(StartX, StartY));
385:         DrawVertex(TPoint(EndX, EndY));
386:         DrawVertex(TPoint(StartX, EndY));
387:         DrawVertex(TPoint(EndX, StartY));
388:     }
389:     break;
390:
391: case DRAWCIRCULARARC:
392: case DRAWCIRCULARARROW:
393: case DRAWCIRCULARARROW2SIDE:
394:     if(ShapeNum > -1)
395:     {
396:         StartX = GData[ShapeNum]->Shape.CircularArc.Start.x;
397:         StartY = GData[ShapeNum]->Shape.CircularArc.Start.y;
398:         EndX = GData[ShapeNum]->Shape.CircularArc.End.x;
399:         EndY = GData[ShapeNum]->Shape.CircularArc.End.y;
400:         MidX = GData[ShapeNum]->Shape.CircularArc.Mid.x;
401:         MidY = GData[ShapeNum]->Shape.CircularArc.Mid.y;
402:         StartSelected = GData[ShapeNum]->Shape.CircularArc.Start.Selected;
403:         EndSelected = GData[ShapeNum]->Shape.CircularArc.End.Selected;
404:         MidSelected = GData[ShapeNum]->Shape.CircularArc.Mid.Selected;
405:     }
406:     else
407:     {
408:         MidX = (EndY - StartY + EndX + StartX) / 2;
409:         MidY = (StartX - EndX + EndY + StartY) / 2;
410:     }
411:     if(StartSelected && EndSelected && MidSelected)
412:         Image->Canvas->Pen->Color = GData[ShapeNum]->Pen.Color;
413:
414:     DrawCircularArc(&TPoint(StartX, StartY), &TPoint(EndX, EndY),
415: &TPoint(MidX, MidY), ShapeKind);
416:
417:     if(StartSelected)
418:         DrawVertex(TPoint(StartX, StartY));
419:     if(EndSelected)
420:         DrawVertex(TPoint(EndX, EndY));
421:     if(MidSelected)
422:         DrawVertex(TPoint(MidX, MidY));
423:     break;
424: }
425:
426: void __fastcall PaintTools::DrawShapes(bool Clear, bool ReDraw)
427: {
```

```
428:     bool StartSelected;
429:     bool EndSelected;
430:     bool MidSelected;
431:     int StartX, StartY;
432:     int EndX, EndY;
433:     int MidX, MidY;
434:
435:     bool TopLeftSelected;
436:     bool TopRightSelected;
437:     bool BottomLeftSelected;
438:     bool BottomRightSelected;
439:     int TopLeftX, TopLeftY;
440:     int TopRightX, TopRightY;
441:     int BottomLeftX, BottomLeftY;
442:     int BottomRightX, BottomRightY;
443:     float RotationAngle;
444:
445:     TPoint Point;
446:
447:     Image->Canvas->Pen->Mode = PenMode;
448:
449:     for(register j = (Clear ? 0 : 1); j < (ReDraw ? 2 : 1); j++)
450:     {
451:         if(!j)
452:             Image->Canvas->Pen->Color = Image->Canvas->Brush->Color;
453:
454:         for(register i = 0; i < ShapeCounter; i++)
455:         {
456:             if(j)
457:                 Image->Canvas->Pen->Color = GData[i]->Pen.Color;
458:
459:             switch(GData[i]->ShapeKind)
460:             {
461:                 case DRAWLINE:
462:                 case DRAWCONTINUOUSLINE:
463:                 case DRAWCHAIN:
464:                 case DRAWPOLYGON:
465:                 case DRAWCYCLOPENTANE:
466:                 case DRAWCIRCLE:
467:                 case DRAWARROW:
468:                 case DRAWARROW2SIDE:
469:                 case DRAWARROW2SIDESEP:
470:                 case DRAWDDOUBLEBOND:
471:                 case DRAWTRIPLEBOND:
472:                 case DRAWUPWEDGE:
473:                 case DRAWDOWNWEDGE:
474:                     StartX = GData[i]->Shape.Line.Start.x;
475:                     StartY = GData[i]->Shape.Line.Start.y;
476:                     RotationAngle = RotationSize(GData[i]->Shape.Line.Start.Angle);
477:                     if(RotationAngle != 0.0)
478:                     {
479:                         Point = TPoint(StartX, StartY);
480:                         RotatePoint(&Point, &ResultantPoint, RotationAngle);
481:                         StartX = Point.x;
482:                         StartY = Point.y;
483:                     }
484:
485:                     EndX = GData[i]->Shape.Line.End.x;
486:                     EndY = GData[i]->Shape.Line.End.y;
487:                     RotationAngle = RotationSize(GData[i]->Shape.Line.End.Angle);
488:                     if(RotationAngle != 0.0)
489:                     {
490:                         Point = TPoint(EndX, EndY);
491:                         RotatePoint(&Point, &ResultantPoint, RotationAngle);
492:                         EndX = Point.x;
493:                         EndY = Point.y;
494:                     }
495:
496:                     StartSelected = GData[i]->Shape.Line.Start.Selected;
497:                     EndSelected = GData[i]->Shape.Line.End.Selected;
498:
499:                     if(GData[i]->ShapeKind == DRAWCIRCLE)
500:                         DrawCircle(&TPoint(StartX, StartY), &TPoint(EndX, EndY));
501:                     else if(GData[i]->ShapeKind == DRAWUPWEDGE ||
GData[i]->ShapeKind == DRAWDOWNWEDGE/* && ShapeNum < 0 */)

```

```
502:             DrawWedge(&TPoint(StartX, StartY), &TPoint(EndX, EndY),
GData[i]->ShapeKind);
503:             else if(GData[i]->ShapeKind == DRAWARROW2SIDESEP)
504:             {
505:                 DrawBond(&TPoint(StartX, StartY), &TPoint(EndX, EndY), 2,
false);
506:                 DrawArrow(&ArrayPoints[3], &ArrayPoints[2], true);
507:                 DrawArrow(&TPoint(StartX, StartY), &TPoint(EndX, EndY),
true);
508:             }
509:             else
510:             {
511:                 if(GData[i]->ShapeKind == DRAWARROW || GData[i]->ShapeKind
== DRAWARROW2SIDE)
512:                     DrawArrow(&TPoint(StartX, StartY), &TPoint(EndX,
EndY));
513:                 if(GData[i]->ShapeKind == DRAWARROW2SIDE)
514:                     DrawArrow(&TPoint(EndX, EndY), &TPoint(StartX,
StartY));
515:                 DrawBond(&TPoint(StartX, StartY), &TPoint(EndX, EndY),
GData[i]->Shape.Line.BondNumber,
518:                     GData[i]->Shape.Line.LeftSideBond);
519:             }
520:             if(StartSelected || !j)
521:                 DrawVertex(TPoint(StartX, StartY));
522:             if(EndSelected || !j)
523:                 DrawVertex(TPoint(EndX, EndY));
524:             break;
525:             ;
526:             ;
527:         case DRAWRECTANGLE:
528:             TopLeftX = GData[i]->Shape.Rectangle.TopLeft.x;
529:             TopLeftY = GData[i]->Shape.Rectangle.TopLeft.y;
530:             RotationAngle =
RotationSize(GData[i]->Shape.Rectangle.TopLeft.Angle);
531:             if(RotationAngle != 0.0)
532:             {
533:                 Point = TPoint(TopLeftX, TopLeftY);
534:                 RotatePoint(&Point, &ResultantPoint, RotationAngle);
535:                 TopLeftX = Point.x;
536:                 TopLeftY = Point.y;
537:             }
538:             TopRightX = GData[i]->Shape.Rectangle.TopRight.x;
539:             TopRightY = GData[i]->Shape.Rectangle.TopRight.y;
540:             RotationAngle =
RotationSize(GData[i]->Shape.Rectangle.TopRight.Angle);
541:             if(RotationAngle != 0.0)
542:             {
543:                 Point = TPoint(TopRightX, TopRightY);
544:                 RotatePoint(&Point, &ResultantPoint, RotationAngle);
545:                 TopRightX = Point.x;
546:                 TopRightY = Point.y;
547:             }
548:             BottomRightX = GData[i]->Shape.Rectangle.BottomRight.x;
549:             BottomRightY = GData[i]->Shape.Rectangle.BottomRight.y;
550:             RotationAngle =
RotationSize(GData[i]->Shape.Rectangle.BottomRight.Angle);
551:             if(RotationAngle != 0.0)
552:             {
553:                 Point = TPoint(BottomRightX, BottomRightY);
554:                 RotatePoint(&Point, &ResultantPoint, RotationAngle);
555:                 BottomRightX = Point.x;
556:                 BottomRightY = Point.y;
557:             }
558:             BottomLeftX = GData[i]->Shape.Rectangle.BottomLeft.x;
559:             BottomLeftY = GData[i]->Shape.Rectangle.BottomLeft.y;
560:             RotationAngle =
RotationSize(GData[i]->Shape.Rectangle.BottomLeft.Angle);
561:             if(RotationAngle != 0.0)
562:             {
563:                 Point = TPoint(BottomLeftX, BottomLeftY);
```

```
567:         RotatePoint(&Point, &ResultantPoint, RotationAngle);
568:         BottomLeftX = Point.x;
569:         BottomLeftY = Point.y;
570:     }
571:
572:     TopLeftSelected = GData[i]->Shape.Rectangle.TopLeft.Selected;
573:     BottomRightSelected =
GData[i]->Shape.Rectangle.BottomRight.Selected;
574:     TopRightSelected = GData[i]->Shape.Rectangle.TopRight.Selected;
575:     BottomLeftSelected =
GData[i]->Shape.Rectangle.BottomLeft.Selected;
576:
577:     DrawRectangle(&TPoint(TopLeftX, TopLeftY), &TPoint(BottomLeftX,
BottomLeftY), &TPoint(TopRightX, TopRightY), &TPoint(BottomRightX, BottomRightY));
578:
579:     if(TopLeftSelected || !j)
580:         DrawVertex(TPoint(TopLeftX, TopLeftY));
581:     if(BottomRightSelected || !j)
582:         DrawVertex(TPoint(BottomRightX, BottomRightY));
583:     if(TopRightSelected || !j)
584:         DrawVertex(TPoint(TopRightX, TopRightY));
585:     if(BottomLeftSelected || !j)
586:         DrawVertex(TPoint(BottomLeftX, BottomLeftY));
587:     break;
588:
589:     case DRAWCIRCULARARC:
590:     case DRAWCIRCULARARROW:
591:     case DRAWCIRCULARARROW2SIDE:
592:         StartX = GData[i]->Shape.CircularArc.Start.x;
593:         StartY = GData[i]->Shape.CircularArc.Start.y;
594:         RotationAngle =
RotationSize(GData[i]->Shape.CircularArc.Start.Angle);
595:         if(RotationAngle != 0.0)
596:         {
597:             Point = TPoint(StartX, StartY);
598:             RotatePoint(&Point, &ResultantPoint, RotationAngle);
599:             StartX = Point.x;
600:             StartY = Point.y;
601:         }
602:
603:         EndX = GData[i]->Shape.CircularArc.End.x;
604:         EndY = GData[i]->Shape.CircularArc.End.y;
605:         RotationAngle =
RotationSize(GData[i]->Shape.CircularArc.End.Angle);
606:         if(RotationAngle != 0.0)
607:         {
608:             Point = TPoint(EndX, EndY);
609:             RotatePoint(&Point, &ResultantPoint, RotationAngle);
610:             EndX = Point.x;
611:             EndY = Point.y;
612:         }
613:
614:         MidX = GData[i]->Shape.CircularArc.Mid.x;
615:         MidY = GData[i]->Shape.CircularArc.Mid.y;
616:         RotationAngle =
RotationSize(GData[i]->Shape.CircularArc.Mid.Angle);
617:         if(RotationAngle != 0.0)
618:         {
619:             Point = TPoint(MidX, MidY);
620:             RotatePoint(&Point, &ResultantPoint, RotationAngle);
621:             MidX = Point.x;
622:             MidY = Point.y;
623:         }
624:         StartSelected = GData[i]->Shape.CircularArc.Start.Selected;
625:         EndSelected = GData[i]->Shape.CircularArc.End.Selected;
626:         MidSelected = GData[i]->Shape.CircularArc.Mid.Selected;
627:
628:         DrawCircularArc(&TPoint(StartX, StartY), &TPoint(EndX, EndY),
&TPoint(MidX, MidY), GData[i]->ShapeKind);
629:
630:         if(StartSelected || !j)
631:             DrawVertex(TPoint(StartX, StartY));
632:         if(EndSelected || !j)
633:             DrawVertex(TPoint(EndX, EndY));
634:         if(MidSelected || !j)
```

```

635:         DrawVertex(TPoint(MidX, MidY));
636:         break;
637:
638:     case ATOMTEXT:
639:         StartX = GData[i]->Shape.AtomText.Position.x;
640:         StartY = GData[i]->Shape.AtomText.Position.y;
641:         RotationAngle =
RotationSize(GData[i]->Shape.AtomText.Position.Angle);
642:         if(RotationAngle != 0.0)
643:         {
644:             Point = TPoint(StartX, StartY);
645:             RotatePoint(&Point, &ResultantPoint, RotationAngle);
646:             StartX = Point.x;
647:             StartY = Point.y;
648:         }
649:
650:         StartSelected = GData[i]->Shape.AtomText.Position.Selected;
651:
652:         AtomTextFont->Name = GData[i]->Shape.AtomText.Font.Name;
653:         AtomTextFont->Color = Image->Canvas->Pen->Color;
654:         AtomTextFont->Charset = GData[i]->Shape.AtomText.Font.CharSet;
655:         AtomTextFont->Height = GData[i]->Shape.AtomText.Font.Height;
656:         AtomTextFont->Pitch = GData[i]->Shape.AtomText.Font.Pitch;
657:         AtomTextFont->Size = GData[i]->Shape.AtomText.Font.Size;
658:         AtomTextFont->Style >> fsBold >> fsItalic >> fsUnderline;
659:
660:         if(GData[i]->Shape.AtomText.Font.Style&1)
661:             AtomTextFont->Style = AtomTextFont->Style << fsBold;
662:         else
663:             AtomTextFont->Style = AtomTextFont->Style >> fsBold;
664:
665:         if(GData[i]->Shape.AtomText.Font.Style&2)
666:             AtomTextFont->Style = AtomTextFont->Style << fsItalic;
667:         else
668:             AtomTextFont->Style = AtomTextFont->Style >> fsItalic;
669:
670:         if(GData[i]->Shape.AtomText.Font.Style&4)
671:             AtomTextFont->Style = AtomTextFont->Style << fsUnderline;
672:         else
673:             AtomTextFont->Style = AtomTextFont->Style >> fsUnderline;
674:
675:         DrawAtomText(AtomTextFont, &TPoint(StartX, StartY),
GData[i]->Shape.AtomText.Text, GData[i]->Shape.AtomText.TextForm);
676:
677:         if(StartSelected || !j)
678:             DrawVertex(TPoint(StartX, StartY));
679:         break;
680:     }
681: }
682: }
683: }
684: //-----
685:
686: void __fastcall PaintTools::DrawChain(TPoint *Start, TPoint *End)
687: {
688:     int PointCounter = DetermineChainArrayPoints(Start, End);
689:
690:     Image->Canvas->MoveTo(Start->x, Start->y);
691:     for(register i = 0; i < PointCounter; i++)
692:         Image->Canvas->LineTo(ArrayPoints[i].x, ArrayPoints[i].y);
693: }
694: //-----
695:
696: int __fastcall PaintTools::DetermineChainArrayPoints(TPoint *Start, TPoint *End)
697: {
698:     int ReturnValue = 0;
699:     int BigY = Start->y > End->y ? Start->y : End->y;
700:     float Slope, Angle;
701:     TPoint StartBak = TPoint(Start->x, -(Start->y - BigY)),
702:         EndBak = TPoint(End->x, -(End->y - BigY));
703:
704:     if(abs(Start->x - End->x) >= abs(Start->y - End->y) && End->x != Start->x)
705:     {
706:         Slope = float(EndBak.y - StartBak.y) / float(EndBak.x - StartBak.x);
707:         Angle = atan(Slope);

```

```

708:
709:     RotatePoint(&StartBak, &TPoint(0, 0), -Angle, false);
710:     RotatePoint(&EndBak, &TPoint(0, 0), -Angle, false);
711:
712:     if(StartBak.x < EndBak.x)
713:     {
714:         for(register i = StartBak.x + CHAINWIDTH / 2; i < EndBak.x && ReturnValue
715: < MAXARRAYPOINTS; i += CHAINWIDTH)
716:         {
717:             ArrayPoints[ReturnValue].x = i;
718:             ArrayPoints[ReturnValue].y = StartBak.y + CHAINHEIGHT;
719:             RotatePoint(&ArrayPoints[ReturnValue], &TPoint(0, 0), Angle, false);
720:             ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
721:             ReturnValue++;
722:
723:             if(i < EndBak.x - CHAINWIDTH / 2)
724:             {
725:                 ArrayPoints[ReturnValue].x = i + CHAINWIDTH / 2;
726:                 ArrayPoints[ReturnValue].y = StartBak.y;
727:                 RotatePoint(&ArrayPoints[ReturnValue], &TPoint(0, 0), Angle,
728: false);
729:                 ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
730:                 ReturnValue++;
731:             }
732:         }
733:     }
734:     else
735:     {
736:         for(register i = StartBak.x - CHAINWIDTH / 2; i > EndBak.x && ReturnValue
737: < MAXARRAYPOINTS; i -= CHAINWIDTH)
738:         {
739:             ArrayPoints[ReturnValue].x = i;
740:             ArrayPoints[ReturnValue].y = StartBak.y - CHAINHEIGHT;
741:             RotatePoint(&ArrayPoints[ReturnValue], &TPoint(0, 0), Angle, false);
742:             ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
743:             ReturnValue++;
744:
745:             if(i > EndBak.x + CHAINWIDTH / 2)
746:             {
747:                 ArrayPoints[ReturnValue].x = i - CHAINWIDTH / 2;
748:                 ArrayPoints[ReturnValue].y = StartBak.y;
749:                 RotatePoint(&ArrayPoints[ReturnValue], &TPoint(0, 0), Angle,
750: false);
751:                 ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
752:                 ReturnValue++;
753:             }
754:         }
755:     }
756:     else if(End->y != Start->y)
757:     {
758:         Slope = float(EndBak.x - StartBak.x) / float(EndBak.y - StartBak.y);
759:         Angle = atan(Slope);
760:
761:         RotatePoint(&StartBak, &TPoint(0, 0), Angle, false);
762:         RotatePoint(&EndBak, &TPoint(0, 0), Angle, false);
763:         if(StartBak.y < EndBak.y)
764:         {
765:             for(register i = StartBak.y + CHAINWIDTH / 2; i < EndBak.y && ReturnValue
766: < MAXARRAYPOINTS; i += CHAINWIDTH)
767:             {
768:                 ArrayPoints[ReturnValue].y = i;
769:                 ArrayPoints[ReturnValue].x = StartBak.x - CHAINHEIGHT;
770:                 RotatePoint(&ArrayPoints[ReturnValue], &TPoint(0, 0), -Angle, false);
771:                 ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
772:                 ReturnValue++;
773:
774:                 if(i < EndBak.y - CHAINWIDTH / 2)
775:                 {
776:                     ArrayPoints[ReturnValue].y = i + CHAINWIDTH / 2;
777:                     ArrayPoints[ReturnValue].x = StartBak.x;
778:                     RotatePoint(&ArrayPoints[ReturnValue], &TPoint(0, 0), -Angle,
779: false);
780:                     ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
781:                     ReturnValue++;

```

```

777:         }
778:     }
779: }
780: else
781: {
782:     for(register i = StartBak.y - CHAINWIDTH / 2; i > EndBak.y && ReturnValue
< MAXARRAYPOINTS; i -= CHAINWIDTH)
783:     {
784:         ArrayPoints[ReturnValue].y = i;
785:         ArrayPoints[ReturnValue].x = StartBak.x + CHAINHEIGHT;
786:         RotatePoint(&ArrayPoints[ReturnValue], &TPoint(0, 0), -Angle, false);
787:         ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
788:         ReturnValue++;
789:
790:         if(i > EndBak.y + CHAINWIDTH / 2)
791:         {
792:             ArrayPoints[ReturnValue].y = i - CHAINWIDTH / 2;
793:             ArrayPoints[ReturnValue].x = StartBak.x;
794:             RotatePoint(&ArrayPoints[ReturnValue], &TPoint(0, 0), -Angle,
false);
795:             ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
796:             ReturnValue++;
797:         }
798:     }
799: }
800: }
801:
802: return ReturnValue;
803: }
804: //-----
805:
806: void __fastcall PaintTools::DrawPolygon(TPoint *Start, TPoint *End, int ShapeKind)
807: {
808:     int PointCounter = DeterminePolygonArrayPoints(Start, End, ShapeKind);
809:     Image->Canvas->MoveTo(ArrayPoints[0].x, ArrayPoints[0].y);
810:     for(register i = 0; i < PointCounter; i++)
811:         Image->Canvas->LineTo(ArrayPoints[i].x, ArrayPoints[i].y);
812:     Image->Canvas->LineTo(ArrayPoints[0].x, ArrayPoints[0].y);
813: }
814: //-----
815:
816: int __fastcall PaintTools::DeterminePolygonArrayPoints(TPoint *Start, TPoint *End, int
ShapeKind)
817: {
818:     int ReturnValue = 0;
819:     float Slope, Angle, PartAngle;
820:     int SideNumber = ShapeKind == DRAWCYCLOPENTANE ? 6 : ActivePolygonSideNumber;
821:     float Distance;
822:     int BigY = Start->y > End->y ? Start->y : End->y;
823:     TPoint StartBak = TPoint(Start->x, -(Start->y - BigY));
824:     EndBak = TPoint(End->x, -(End->y - BigY));
825:
826:     if(abs(Start->x - End->x) >= abs(Start->y - End->y) && End->x != Start->x)
827:     {
828:         Slope = float(EndBak.y - StartBak.y) / float(EndBak.x - StartBak.x);
829:         Angle = atan(Slope);
830:
831:         if(Start->x - End->x > 0)
832:             Angle += M_PI;
833:
834:         PartAngle = M_PI * 2 / SideNumber;
835:         Distance = sqrt(pow(StartBak.x - EndBak.x, 2) + pow(StartBak.y - EndBak.y, 2));
836:
837:         for(register i = 0; i < SideNumber; i++)
838:         {
839:             if(ShapeKind != DRAWCYCLOPENTANE || i != 3)
840:             {
841:                 ArrayPoints[ReturnValue].x = Distance * cos(Angle + PartAngle * i)
StartBak.x;
842:                 ArrayPoints[ReturnValue].y = Distance * sin(Angle + PartAngle * i)
StartBak.y;
843:                 ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
844:                 ReturnValue++;
845:             }
846:         }

```

```

847:     }
848:     else if(End->y != Start->y)
849:     {
850:         Slope = float(EndBak.x - StartBak.x) / float(EndBak.y - StartBak.y);
851:         Angle = atan(Slope);
852:
853:         if(Start->y - End->y < 0)
854:             Angle += M_PI;
855:
856:         PartAngle = M_PI * 2 / SideNumber;
857:         Distance = sqrt(pow(StartBak.x - EndBak.x, 2) + pow(StartBak.y - EndBak.y, 2));
858:
859:         for(register i = 0; i < SideNumber; i++)
860:         {
861:             if(ShapeKind != DRAWCYCLOPENTANE || i != 3)
862:             {
863:                 ArrayPoints[ReturnValue].y = Distance * cos(Angle + PartAngle * i)
StartBak.y;
864:                 ArrayPoints[ReturnValue].x = Distance * sin(Angle + PartAngle * i)
StartBak.x;
865:                 ArrayPoints[ReturnValue].y = BigY - ArrayPoints[ReturnValue].y;
866:                 ReturnValue++;
867:             }
868:         }
869:     }
870:
871:     return ReturnValue;
872: }
873: //-----
874:
875: void __fastcall PaintTools::DrawCircle(TPoint *Start, TPoint *End)
876: {
877:     float Redius = sqrt(pow(Start->x - End->x, 2) + pow(Start->y - End->y, 2));
878:
879:     Image->Canvas->Ellipse(Start->x - Redius, Start->y - Redius, Start->x + Redius,
Start->y + Redius);
880: }
881: //-----
882:
883: void __fastcall PaintTools::DrawWedge(TPoint *Start, TPoint *End, int ShapeKind)
884: {
885:     int BigY = Start->y > End->y ? Start->y : End->y;
886:     float Slope, Angle;
887:     bool DoDraw = false;
888:     TPoint StartBak = TPoint(Start->x, -(Start->y - BigY)),
889:           EndBak = TPoint(End->x, -(End->y - BigY));
890:
891:     int Distance = sqrt(sqrt(pow(StartBak.x - EndBak.x, 2) + pow(StartBak.y - EndBak.y
2))) / 2;
892:
893:     if(abs(Start->x - End->x) >= abs(Start->y - End->y) && End->x != Start->x)
894:     {
895:         Slope = float(EndBak.y - StartBak.y) / float(EndBak.x - StartBak.x);
896:         Angle = atan(Slope);
897:
898:         RotatePoint(&StartBak, &TPoint(0, 0), -Angle, false);
899:
900:         ArrayPoints[0].x = ArrayPoints[1].x = StartBak.x;
901:         ArrayPoints[0].y = StartBak.y + Distance;
902:         ArrayPoints[1].y = StartBak.y - Distance;
903:         RotatePoint(&ArrayPoints[0], &TPoint(0, 0), Angle, false);
904:         RotatePoint(&ArrayPoints[1], &TPoint(0, 0), Angle, false);
905:         ArrayPoints[0].y = BigY - ArrayPoints[0].y;
906:         ArrayPoints[1].y = BigY - ArrayPoints[1].y;
907:         DoDraw = true;
908:     }
909:     else if(End->y != Start->y)
910:     {
911:         Slope = float(EndBak.x - StartBak.x) / float(EndBak.y - StartBak.y);
912:         Angle = atan(Slope);
913:
914:         RotatePoint(&StartBak, &TPoint(0, 0), Angle, false);
915:
916:         ArrayPoints[0].y = ArrayPoints[1].y = StartBak.y;
917:         ArrayPoints[0].x = StartBak.x + Distance;

```

```

918:         ArrayPoints[1].x = StartBak.x - Distance;
919:         RotatePoint(&ArrayPoints[0], &TPoint(0, 0), -Angle, false);
920:         RotatePoint(&ArrayPoints[1], &TPoint(0, 0), -Angle, false);
921:         ArrayPoints[0].y = BigY - ArrayPoints[0].y;
922:         ArrayPoints[1].y = BigY - ArrayPoints[1].y;
923:         DoDraw = true;
924:     }
925:
926:     if(DoDraw)
927:     {
928:         TBrushStyle Style = Image->Canvas->Brush->Style;
929:         TColor Color = Image->Canvas->Brush->Color;
930:
931:         if(!Drawing)
932:         {
933:             Image->Canvas->Brush->Style = ShapeKind == DRAWDOWNWEDGE ? bsDiagCross :
bsSolid;
934:             Image->Canvas->Brush->Color = Image->Canvas->Pen->Color;
935:         }
936:
937:         ArrayPoints[2] = *End;
938:         Image->Canvas->Polygon(ArrayPoints, 2);
939:
940:         if(!Drawing)
941:         {
942:             Image->Canvas->Brush->Color = Color;
943:             Image->Canvas->Brush->Style = Style;
944:         }
945:     }
946: }
947: //-----
948:
949: void __fastcall PaintTools::DrawCircularArc(TPoint *Start, TPoint *End, TPoint *Mid, int
ShapeKind)
950: {
951:     TPoint StartBak = TPoint(Start->x - Mid->x, -(Start->y - Mid->y)),
952:         EndBak = TPoint(End->x - Mid->x, -(End->y - Mid->y)),
953:         StartArrowPoint,
954:         EndArrowPoint;
955:
956:     int UnderFraction = 2 * (StartBak.x * EndBak.y - EndBak.x * StartBak.y);
957:     if(UnderFraction)
958:     {
959:         float CenterX = (EndBak.y * (pow(StartBak.x, 2) + pow(StartBak.y, 2)) -
StartBak.y * (pow(EndBak.x, 2) + pow(EndBak.y, 2))) /
960:             UnderFraction;
961:         float CenterY = (StartBak.x * (pow(EndBak.x, 2) + pow(EndBak.y, 2)) - EndBak
* (pow(StartBak.x, 2) + pow(StartBak.y, 2))) /
962:             UnderFraction;
963:
964:         StartArrowPoint = TPoint(CenterX, CenterY);
965:         EndArrowPoint = TPoint(CenterX, CenterY);
966:
967:         CenterX += Mid->x;
968:         CenterY = -(CenterY - Mid->y);
969:
970:         int Radius = sqrt(pow(CenterX - Start->x, 2) + pow(CenterY - Start->y, 2));
971:
972:         if(IsPixelAtTheRightOfTheLine(&StartBak, &EndBak, &TPoint(0, 0)))
973:         {
974:             Image->Canvas->Arc(CenterX - Radius, CenterY - Radius, CenterX + Radius,
CenterY + Radius, Start->x, Start->y, End->x, End->y);
975:             RotatePoint(&StartArrowPoint, &EndBak, -M_PI_2, true);
976:             RotatePoint(&EndArrowPoint, &StartBak, M_PI_2, true);
977:         }
978:         else
979:         {
980:             Image->Canvas->Arc(CenterX - Radius, CenterY - Radius, CenterX + Radius,
CenterY + Radius, End->x, End->y, Start->x, Start->y);
981:             RotatePoint(&StartArrowPoint, &EndBak, M_PI_2, true);
982:             RotatePoint(&EndArrowPoint, &StartBak, -M_PI_2, true);
983:         }
984:
985:         StartArrowPoint.x += Mid->x;
986:         StartArrowPoint.y = -(StartArrowPoint.y - Mid->y);

```

```
987:
988:     EndArrowPoint.x += Mid->x;
989:     EndArrowPoint.y = -(EndArrowPoint.y - Mid->y);
990:
991:     if(ShapeKind == DRAWCIRCULARARROW || ShapeKind == DRAWCIRCULARARROW2SIDE)
992:     {
993:         DrawArrow(&StartArrowPoint, End);
994:     }
995:     if(ShapeKind == DRAWCIRCULARARROW2SIDE)
996:     {
997:         DrawArrow(&EndArrowPoint, Start);
998:     }
999: }
1000: }
1001: //-----
1002:
1003: void __fastcall PaintTools::DrawRectangle(TPoint *TopLeft, TPoint *BottomLeft, TPoint
    *TopRight, TPoint *BottomRight)
1004: {
1005:     Image->Canvas->MoveTo(TopLeft->x, TopLeft->y);
1006:     Image->Canvas->LineTo(TopRight->x, TopRight->y);
1007:     Image->Canvas->LineTo(BottomRight->x, BottomRight->y);
1008:     Image->Canvas->LineTo(BottomLeft->x, BottomLeft->y);
1009:     Image->Canvas->LineTo(TopLeft->x, TopLeft->y);
1010: }
1011: //-----
1012:
1013: void __fastcall PaintTools::DrawArrow(TPoint *Start, TPoint *End, bool HalfArrow)
1014: {
1015:     int BigY = Start->y > End->y ? Start->y : End->y;
1016:     float Slope, Angle;
1017:     TPoint StartBak = TPoint(Start->x, -(Start->y - BigY));
1018:     EndBak = TPoint(End->x, -(End->y - BigY));
1019:     TPoint Points[3];
1020:     bool DoDraw = true;
1021:
1022:     if(abs(Start->x - End->x) >= abs(Start->y - End->y) && End->x != Start->x)
1023:     {
1024:         Slope = float(EndBak.y - StartBak.y) / float(EndBak.x - StartBak.x);
1025:         Angle = atan(Slope);
1026:
1027:         if(Angle < M_PI_4 && Angle > -M_PI_4 && EndBak.x - StartBak.x <= 0)
1028:             Angle += M_PI;
1029:
1030:         RotatePoint(&StartBak, &TPoint(0, 0), -Angle, false);
1031:         RotatePoint(&EndBak, &TPoint(0, 0), -Angle, false);
1032:
1033:         Points[0].y = EndBak.y - (HalfArrow ? 0 : 3);
1034:         Points[1].y = EndBak.y + 3;
1035:
1036:         if(StartBak.x < EndBak.x)
1037:             Points[0].x = Points[1].x = EndBak.x - 15;
1038:         else
1039:             Points[0].x = Points[1].x = EndBak.x + 15;
1040:
1041:         RotatePoint(&Points[0], &TPoint(0, 0), Angle, false);
1042:         RotatePoint(&Points[1], &TPoint(0, 0), Angle, false);
1043:
1044:         Points[0].y = BigY - Points[0].y;
1045:         Points[1].y = BigY - Points[1].y;
1046:
1047:     }
1048:     else if(End->y != Start->y)
1049:     {
1050:         Slope = float(EndBak.x - StartBak.x) / float(EndBak.y - StartBak.y);
1051:         Angle = atan(Slope);
1052:
1053:         if(Angle < M_PI_4 && Angle > -M_PI_4 && EndBak.y - StartBak.y >= 0)
1054:             Angle += M_PI;
1055:
1056:         RotatePoint(&StartBak, &TPoint(0, 0), Angle, false);
1057:         RotatePoint(&EndBak, &TPoint(0, 0), Angle, false);
1058:
1059:         Points[0].x = EndBak.x - (HalfArrow ? 0 : 3);
1060:         Points[1].x = EndBak.x + 3;
```

```
1061:
1062:     if(StartBak.y < EndBak.y)
1063:         Points[0].y = Points[1].y = EndBak.y - 15;
1064:     else
1065:         Points[0].y = Points[1].y = EndBak.y + 15;
1066:
1067:     RotatePoint(&Points[0], &TPoint(0, 0), -Angle, false);
1068:     RotatePoint(&Points[1], &TPoint(0, 0), -Angle, false);
1069:
1070:     Points[0].y = BigY - Points[0].y;
1071:     Points[1].y = BigY - Points[1].y;
1072: }
1073: else
1074:     DoDraw = false;
1075:
1076:
1077: if(DoDraw)
1078: {
1079:     TBrushStyle Style = Image->Canvas->Brush->Style;
1080:     TColor Color = Image->Canvas->Brush->Color;
1081:
1082:     if(!Drawing)
1083:     {
1084:         Image->Canvas->Brush->Style = bsSolid;
1085:         Image->Canvas->Brush->Color = Image->Canvas->Pen->Color;
1086:     }
1087:
1088:     Points[2] = *End;
1089:     Image->Canvas->Polygon(Points, 2);
1090:
1091:     if(!Drawing)
1092:     {
1093:         Image->Canvas->Brush->Color = Color;
1094:         Image->Canvas->Brush->Style = Style;
1095:     }
1096: }
1097: }
1098: //-----
1100: void __fastcall PaintTools::DrawBond(TPoint *Start, TPoint *End, int BondNumber, bool
    IsLeftSideBond)
1101: {
1102:     int BigY = Start->y > End->y ? Start->y : End->y;
1103:     float Slope, Angle;
1104:     TPoint StartBak = TPoint(Start->x, -(Start->y - BigY));
1105:     EndBak = TPoint(End->x, -(End->y - BigY));
1106:
1107:     if(BondNumber > 1)
1108:     {
1109:         if(abs(Start->x - End->x) >= abs(Start->y - End->y) && End->x != Start->x)
1110:         {
1111:             Slope = float(EndBak.y - StartBak.y) / float(EndBak.x - StartBak.x);
1112:             Angle = atan(Slope);
1113:
1114:             if(Angle < M_PI_4 && Angle > -M_PI_4 && EndBak.x - StartBak.x <= 0)
1115:                 Angle += M_PI;
1116:
1117:             RotatePoint(&StartBak, &TPoint(0, 0), -Angle, false);
1118:             RotatePoint(&EndBak, &TPoint(0, 0), -Angle, false);
1119:
1120:             ArrayPoints[0].x = ArrayPoints[2].x = StartBak.x;
1121:             ArrayPoints[1].x = ArrayPoints[3].x = EndBak.x;
1122:             ArrayPoints[0].y = ArrayPoints[1].y = StartBak.y + 5;
1123:             ArrayPoints[2].y = ArrayPoints[3].y = StartBak.y - 5;
1124:             RotatePoint(&ArrayPoints[0], &TPoint(0, 0), Angle, false);
1125:             RotatePoint(&ArrayPoints[1], &TPoint(0, 0), Angle, false);
1126:             RotatePoint(&ArrayPoints[2], &TPoint(0, 0), Angle, false);
1127:             RotatePoint(&ArrayPoints[3], &TPoint(0, 0), Angle, false);
1128:             ArrayPoints[0].y = BigY - ArrayPoints[0].y;
1129:             ArrayPoints[1].y = BigY - ArrayPoints[1].y;
1130:             ArrayPoints[2].y = BigY - ArrayPoints[2].y;
1131:             ArrayPoints[3].y = BigY - ArrayPoints[3].y;
1132:         }
1133:         else if(End->y != Start->y)
1134:         {
```

```

1135:         Slope = float(EndBak.x - StartBak.x) / float(EndBak.y - StartBak.y);
1136:         Angle = atan(Slope);
1137:
1138:         if(Angle < M_PI_4 && Angle > -M_PI_4 && EndBak.y - StartBak.y >= 0)
1139:             Angle += M_PI;
1140:
1141:         RotatePoint(&StartBak, &TPoint(0, 0), Angle, false);
1142:         RotatePoint(&EndBak, &TPoint(0, 0), Angle, false);
1143:
1144:         ArrayPoints[0].y = ArrayPoints[2].y = StartBak.y;
1145:         ArrayPoints[1].y = ArrayPoints[3].y = EndBak.y;
1146:         ArrayPoints[0].x = ArrayPoints[1].x = StartBak.x + 5;
1147:         ArrayPoints[2].x = ArrayPoints[3].x = EndBak.x - 5;
1148:         RotatePoint(&ArrayPoints[0], &TPoint(0, 0), -Angle, false);
1149:         RotatePoint(&ArrayPoints[1], &TPoint(0, 0), -Angle, false);
1150:         RotatePoint(&ArrayPoints[2], &TPoint(0, 0), -Angle, false);
1151:         RotatePoint(&ArrayPoints[3], &TPoint(0, 0), -Angle, false);
1152:         ArrayPoints[0].y = BigY - ArrayPoints[0].y;
1153:         ArrayPoints[1].y = BigY - ArrayPoints[1].y;
1154:         ArrayPoints[2].y = BigY - ArrayPoints[2].y;
1155:         ArrayPoints[3].y = BigY - ArrayPoints[3].y;
1156:     }
1157:
1158:     if(BondNumber > 2 || IsLeftSideBond)
1159:     {
1160:         Image->Canvas->MoveTo(ArrayPoints[0].x, ArrayPoints[0].y);
1161:         Image->Canvas->LineTo(ArrayPoints[1].x, ArrayPoints[1].y);
1162:     }
1163:
1164:     if(BondNumber > 2 || !IsLeftSideBond)
1165:     {
1166:         Image->Canvas->MoveTo(ArrayPoints[2].x, ArrayPoints[2].y);
1167:         Image->Canvas->LineTo(ArrayPoints[3].x, ArrayPoints[3].y);
1168:     }
1169: }
1170:
1171: Image->Canvas->MoveTo(Start->x, Start->y);
1172: Image->Canvas->LineTo(End->x, End->y);
1173: }
1174: //-----
1175:
1176: void __fastcall PaintTools::DrawVertex(TPoint VertexPoint)
1177: {
1178:     Image->Canvas->Rectangle(VertexPoint.x - 2, VertexPoint.y - 2, VertexPoint.x + 2,
1179:         VertexPoint.y + 2);
1180: }
1181: //-----
1182: bool __fastcall PaintTools::IsShapeSelected(int ShapeNum)
1183: {
1184:     bool ReturnValue = false;
1185:
1186:     switch(GData[ShapeNum]->ShapeKind)
1187:     {
1188:         case DRAWLINE:
1189:         case DRAWARROW:
1190:         case DRAWARROW2SIDE:
1191:         case DRAWARROW2SIDESEP:
1192:         case DRAWCONTINUOUSLINE:
1193:         case DRAWCHAIN:
1194:         case DRAWPOLYGON:
1195:         case DRAWCYCLOPENTANE:
1196:         case DRAWCIRCLE:
1197:         case DRAWDUBLEBOND:
1198:         case DRAWTRIPLEBOND:
1199:         case DRAWUPWEDGE:
1200:         case DRAWDOWNWEDGE:
1201:             if(GData[ShapeNum]->Shape.Line.Start.Selected &&
1202:                 GData[ShapeNum]->Shape.Line.End.Selected)
1203:                 ReturnValue = true;
1204:             break;
1205:
1206:         case ATOMTEXT:
1207:             if(GData[ShapeNum]->Shape.AtomText.Position.Selected)
1208:                 ReturnValue = true;

```

```
1209:         break;
1210:
1211:     case DRAWRECTANGLE:
1212:         if (GData[ShapeNum] -> Shape.Rectangle.TopLeft.Selected &&
1213:             GData[ShapeNum] -> Shape.Rectangle.BottomRight.Selected &&
1214:             GData[ShapeNum] -> Shape.Rectangle.TopRight.Selected &&
1215:             GData[ShapeNum] -> Shape.Rectangle.BottomLeft.Selected)
1216:             ReturnValue = true;
1217:         break;
1218:
1219:     case DRAWCIRCULARARC:
1220:     case DRAWCIRCULARARROW:
1221:     case DRAWCIRCULARARROW2SIDE:
1222:         if (GData[ShapeNum] -> Shape.CircularArc.Start.Selected &&
1223:             GData[ShapeNum] -> Shape.CircularArc.End.Selected &&
1224:             GData[ShapeNum] -> Shape.CircularArc.Mid.Selected)
1225:             ReturnValue = true;
1226:         break;
1227:     }
1228:
1229:     return ReturnValue;
1230: }
1231: //-----
1232:
1233: bool __fastcall PaintTools::IsVertexSelected(TPoint *VertexPosition)
1234: {
1235:     bool ReturnValue = false;
1236:
1237:     for(register i = 0; i < ShapeCounter && !ReturnValue; i++)
1238:     {
1239:         switch(GData[i] -> ShapeKind)
1240:         {
1241:             case DRAWLINE:
1242:             case DRAWARROW:
1243:             case DRAWARROW2SIDE:
1244:             case DRAWARROW2SIDESEP:
1245:             case DRAWCONTINUOUSLINE:
1246:             case DRAWCHAIN:
1247:             case DRAWPOLYGON:
1248:             case DRAWCYCLOPENTANE:
1249:             case DRAWCIRCLE:
1250:             case DRAWDUBLEBOND:
1251:             case DRAWTRIPLEBOND:
1252:             case DRAWUPWEDGE:
1253:             case DRAWDOWNWEDGE:
1254:                 if(
1255:                     (ComparePoints(VertexPosition,
1256: &TPoint(GData[i] -> Shape.Line.Start.x, GData[i] -> Shape.Line.Start.y)) &&
1257:                     GData[i] -> Shape.Line.Start.Selected) ||
1258:                     (ComparePoints(VertexPosition, &TPoint(GData[i] -> Shape.Line.End
1259: GData[i] -> Shape.Line.End.y)) &&
1260:                     GData[i] -> Shape.Line.End.Selected)
1261:                 )
1262:                     ReturnValue = true;
1263:                 break;
1264:
1265:             case ATOMTEXT:
1266:                 if(
1267:                     (ComparePoints(VertexPosition,
1268: &TPoint(GData[i] -> Shape.AtomText.Position.x, GData[i] -> Shape.AtomText.Position.y)) &&
1269:                     GData[i] -> Shape.AtomText.Position.Selected)
1270:                 )
1271:                     ReturnValue = true;
1272:                 break;
1273:
1274:             case DRAWRECTANGLE:
1275:                 if(
1276:                     (ComparePoints(VertexPosition,
1277: &TPoint(GData[i] -> Shape.Rectangle.TopLeft.x, GData[i] -> Shape.Rectangle.TopLeft.y)) &&
1278:                     GData[i] -> Shape.Rectangle.TopLeft.Selected) ||
1279:                     (ComparePoints(VertexPosition,
1280: &TPoint(GData[i] -> Shape.Rectangle.TopRight.x, GData[i] -> Shape.Rectangle.TopRight.y)) &&
1281:                     GData[i] -> Shape.Rectangle.TopRight.Selected) ||
1282:                     (ComparePoints(VertexPosition,
1283: &TPoint(GData[i] -> Shape.Rectangle.BottomLeft.x, GData[i] -> Shape.Rectangle.BottomLeft.y)) &&
1284:                     GData[i] -> Shape.Rectangle.BottomLeft.Selected) ||
1285:                     (ComparePoints(VertexPosition,
1286: &TPoint(GData[i] -> Shape.Rectangle.BottomRight.x, GData[i] -> Shape.Rectangle.BottomRight.y)) &&
1287:                     GData[i] -> Shape.Rectangle.BottomRight.Selected)
1288:                 )
1289:                     ReturnValue = true;
1290:                 break;
1291:         }
1292:     }
1293:     return ReturnValue;
1294: }
```

```

1278:         GData[i]->Shape.Rectangle.BottomLeft.Selected) ||
1279:         (ComparePoints(VertexPosition,
1280:         &TPoint(GData[i]->Shape.Rectangle.BottomRight.x,
1281:         GData[i]->Shape.Rectangle.BottomRight.y)) &&
1282:         GData[i]->Shape.Rectangle.BottomRight.Selected)
1283:         )
1284:         ReturnValue = true;
1285:         break;
1286:     case DRAWCIRCULARARC:
1287:     case DRAWCIRCULARARROW:
1288:     case DRAWCIRCULARARROW2SIDE:
1289:         if(
1290:             (ComparePoints(VertexPosition,
1291:             &TPoint(GData[i]->Shape.CircularArc.Start.x, GData[i]->Shape.CircularArc.Start.y)) &&
1292:             GData[i]->Shape.CircularArc.Start.Selected) ||
1293:             (ComparePoints(VertexPosition,
1294:             &TPoint(GData[i]->Shape.CircularArc.End.x, GData[i]->Shape.CircularArc.End.y)) &&
1295:             GData[i]->Shape.CircularArc.End.Selected) ||
1296:             (ComparePoints(VertexPosition,
1297:             &TPoint(GData[i]->Shape.CircularArc.Mid.x, GData[i]->Shape.CircularArc.Mid.y)) &&
1298:             GData[i]->Shape.CircularArc.Mid.Selected)
1299:             )
1300:             ReturnValue = true;
1301:             break;
1302:         }
1303:     }
1304:     return ReturnValue;
1305: }
1306: //-----
1307: void __fastcall PaintTools::RotatePoint(TPoint *Point, TPoint *RotationPoint, float
1308: Angle, bool ChangeAxisToRotationPoint)
1309: {
1310:     float X;
1311:     struct
1312:     {
1313:         float x, y;
1314:     }PointBak = {Point->x, Point->y};
1315:     if(ChangeAxisToRotationPoint)
1316:     {
1317:         PointBak.x = PointBak.x - RotationPoint->x;
1318:         PointBak.y = RotationPoint->y - PointBak.y;
1319:     }
1320:     X = PointBak.x;
1321:     PointBak.x = X * cos(Angle) - PointBak.y * sin(Angle);
1322:     PointBak.y = X * sin(Angle) + PointBak.y * cos(Angle);
1323:     if(ChangeAxisToRotationPoint)
1324:     {
1325:         PointBak.x = PointBak.x + (float)RotationPoint->x;
1326:         PointBak.y = (float)RotationPoint->y - PointBak.y;
1327:     }
1328:     *Point = TPoint(PointBak.x, PointBak.y);
1329: }
1330: //-----
1331: float __fastcall PaintTools::RotationSize(float Angle)
1332: {
1333:     float ReturnValue = Angle;
1334:     if(RotationFixed)
1335:     {
1336:         int DistanceAngle = Angle/FIXROTATIONSIZE;
1337:         ReturnValue = DistanceAngle * FIXROTATIONSIZE;
1338:     }
1339:     return ReturnValue;
1340: }
1341: //-----

```

```

1346:
1347: bool __fastcall PaintTools::IsPixelAtTheRightOfTheLine(TPoint *LineStart, TPoint
    *LineEnd, TPoint *Point)
1348: {
1349:     bool ReturnValue = false;
1350:     float Slope;
1351:     float Distance;
1352:
1353:     if(abs(LineStart->x - LineEnd->x) > abs(LineStart->y - LineEnd->y))
1354:     {
1355:         Slope = float(LineEnd->y - LineStart->y) / float(LineEnd->x - LineStart->x);
1356:         Distance = Slope * LineStart->x - LineStart->y;
1357:         if(LineStart->x > LineEnd->x)
1358:         {
1359:             if(Point->y < Slope * Point->x + Distance)
1360:                 ReturnValue = true;
1361:         }
1362:         else
1363:         {
1364:             if(Point->y > Slope * Point->x + Distance)
1365:                 ReturnValue = true;
1366:         }
1367:     }
1368:     else
1369:     {
1370:         Slope = float(LineEnd->x - LineStart->x) / float(LineEnd->y - LineStart->y);
1371:         Distance = Slope * LineStart->y - LineStart->x;
1372:         if(LineStart->y > LineEnd->y)
1373:         {
1374:             if(Point->x > Slope * Point->y + Distance)
1375:                 ReturnValue = true;
1376:         }
1377:         else
1378:         {
1379:             if(Point->x < Slope * Point->y + Distance)
1380:                 ReturnValue = true;
1381:         }
1382:     }
1383:
1384:     return ReturnValue;
1385: }
1386: //-----
1387:
1388: bool __fastcall PaintTools::ComparePoints(TPoint *Point1, TPoint *Point2)
1389: {
1390:     bool ReturnValue = true;
1391:
1392:     if(Point1->x != Point2->x || Point1->y != Point2->y)
1393:         ReturnValue = false;
1394:
1395:     return ReturnValue;
1396: }
1397: //-----
1398:
1399: void __fastcall PaintTools::RefreshAllImage(void)
1400: {
1401:     Image->Canvas->Brush->Style = bsSolid;
1402:     Image->Canvas->FillRect(Image->Canvas->ClipRect);
1403:     Image->Canvas->Brush->Style = bsClear;
1404:
1405:     PenMode = pmCopy;
1406:     DrawShapes();
1407: }
1408: //-----
1409:
1410: int __fastcall PaintTools::FirstShapeContainPoint(TPoint *VertexPosition)
1411: {
1412:     int ReturnValue = -1;
1413:
1414:     for(register i = 0; i < ShapeCounter && ReturnValue == -1; i++)
1415:     {
1416:         switch(GData[i]->ShapeKind)
1417:         {
1418:             case DRAWLINE:
1419:                 case DRAWARROW:

```

```
1420:         case DRAWARROW2SIDE:
1421:         case DRAWARROW2SIDESEP:
1422:         case DRAWCONTINUOUSLINE:
1423:         case DRAWCHAIN:
1424:         case DRAWPOLYGON:
1425:         case DRAWCYCLOPENTANE:
1426:         case DRAWCIRCLE:
1427:         case DRAWDDOUBLEBOND:
1428:         case DRAWTRIPLEBOND:
1429:         case DRAWUPWEDGE:
1430:         case DRAWDOWNWEDGE:
1431:             if(
1432:                 ComparePoints(VertexPosition,
1433:                     &TPoint(GData[i]->Shape.Line.Start.x, GData[i]->Shape.Line.Start.y)) ||
1434:                 ComparePoints(VertexPosition, &TPoint(GData[i]->Shape.Line.End.x,
1435:                     GData[i]->Shape.Line.End.y))
1436:             )
1437:                 ReturnValue = i;
1438:             break;
1439:         case ATOMTEXT:
1440:             if(
1441:                 ComparePoints(VertexPosition,
1442:                     &TPoint(GData[i]->Shape.AtomText.Position.x, GData[i]->Shape.AtomText.Position.y))
1443:             )
1444:                 ReturnValue = i;
1445:             break;
1446:         case DRAWRECTANGLE:
1447:             if(
1448:                 ComparePoints(VertexPosition,
1449:                     &TPoint(GData[i]->Shape.Rectangle.TopLeft.x, GData[i]->Shape.Rectangle.TopLeft.y)) ||
1450:                 ComparePoints(VertexPosition,
1451:                     &TPoint(GData[i]->Shape.Rectangle.TopRight.x, GData[i]->Shape.Rectangle.TopRight.y)) ||
1452:                 ComparePoints(VertexPosition,
1453:                     &TPoint(GData[i]->Shape.Rectangle.BottomLeft.x, GData[i]->Shape.Rectangle.BottomLeft.y)) ||
1454:                 ComparePoints(VertexPosition,
1455:                     &TPoint(GData[i]->Shape.Rectangle.BottomRight.x,
1456:                         GData[i]->Shape.Rectangle.BottomRight.y))
1457:             )
1458:                 ReturnValue = i;
1459:             break;
1460:         case DRAWCIRCULARARC:
1461:         case DRAWCIRCULARARROW:
1462:         case DRAWCIRCULARARROW2SIDE:
1463:             if(
1464:                 ComparePoints(VertexPosition,
1465:                     &TPoint(GData[i]->Shape.CircularArc.Start.x, GData[i]->Shape.CircularArc.Start.y)) ||
1466:                 ComparePoints(VertexPosition,
1467:                     &TPoint(GData[i]->Shape.CircularArc.End.x, GData[i]->Shape.CircularArc.End.y)) ||
1468:                 ComparePoints(VertexPosition,
1469:                     &TPoint(GData[i]->Shape.CircularArc.Mid.x, GData[i]->Shape.CircularArc.Mid.y))
1470:             )
1471:                 ReturnValue = i;
1472:             break;
1473:     }
1474:     return ReturnValue;
1475: }
1476: //-----
1477: void __fastcall PaintTools::DeleteShape(int ShapeNumber)
1478: {
1479:     if(ShapeCounter)
1480:     {
1481:         GraphData *GDataPointer = GData[ShapeNumber];
1482:         for(register i = ShapeNumber; i < ShapeCounter; i++)
1483:             GData[i] = GData[i + 1];
1484:         delete GDataPointer;
1485:         ShapeCounter--;
1486:     }
1487: }
```

```
1483:     }
1484: }
1485: //-----
1486:
1487: void __fastcall PaintTools::DeleteAllShapes(bool OnlySelected)
1488: {
1489:     for(register i = ShapeCounter - 1; i >= 0; i--)
1490:         if(OnlySelected == false || IsShapeSelected(i))
1491:             DeleteShape(i);
1492:
1493:     RefreshAllImage();
1494: }
1495: //-----
1496:
1497: bool __fastcall PaintTools::IsMouseOnVertex(int X, int Y, bool ReDraw)
1498: {
1499:     TPoint CurrentOnVertexPoint = TPoint(-1, -1);
1500:     bool Status = false;
1501:
1502:     for(register i = 0; i < ShapeCounter && !Status; i++)
1503:     {
1504:         switch(GData[i]->ShapeKind)
1505:         {
1506:             case DRAWLINE:
1507:             case DRAWCONTINUOUSLINE:
1508:             case DRAWCHAIN:
1509:             case DRAWPOLYGON:
1510:             case DRAWCYCLOPENTANE:
1511:             case DRAWCIRCLE:
1512:             case DRAWARROW:
1513:             case DRAWARROW2SIDE:
1514:             case DRAWARROW2SIDESEP:
1515:             case DRAWDUBLEBOND:
1516:             case DRAWTRIPLEBOND:
1517:             case DRAWUPWEDGE:
1518:             case DRAWDOWNWEDGE:
1519:                 if(X >= GData[i]->Shape.Line.Start.x - 2 &&
1520:                    X <= GData[i]->Shape.Line.Start.x + 2 &&
1521:                    Y >= GData[i]->Shape.Line.Start.y - 2 &&
1522:                    Y <= GData[i]->Shape.Line.Start.y + 2)
1523:                 {
1524:                     CurrentOnVertexPoint = TPoint(GData[i]->Shape.Line.Start.x,
1525: GData[i]->Shape.Line.Start.y);
1526:                     Status = true;
1527:                 }
1528:                 else if(X >= GData[i]->Shape.Line.End.x - 2 &&
1529:                    X <= GData[i]->Shape.Line.End.x + 2 &&
1530:                    Y >= GData[i]->Shape.Line.End.y - 2 &&
1531:                    Y <= GData[i]->Shape.Line.End.y + 2)
1532:                 {
1533:                     CurrentOnVertexPoint = TPoint(GData[i]->Shape.Line.End.x,
1534: GData[i]->Shape.Line.End.y);
1535:                     Status = true;
1536:                 }
1537:                 break;
1538:             case ATOMTEXT:
1539:                 if(X >= GData[i]->Shape.AtomText.Position.x - 2 &&
1540:                    X <= GData[i]->Shape.AtomText.Position.x + 2 &&
1541:                    Y >= GData[i]->Shape.AtomText.Position.y - 2 &&
1542:                    Y <= GData[i]->Shape.AtomText.Position.y + 2)
1543:                 {
1544:                     CurrentOnVertexPoint =
1545: TPoint(GData[i]->Shape.AtomText.Position.x, GData[i]->Shape.AtomText.Position.y);
1546:                     Status = true;
1547:                 }
1548:                 break;
1549:             case DRAWRECTANGLE:
1550:                 if(X >= GData[i]->Shape.Rectangle.TopLeft.x - 2 &&
1551:                    X <= GData[i]->Shape.Rectangle.TopLeft.x + 2 &&
1552:                    Y >= GData[i]->Shape.Rectangle.TopLeft.y - 2 &&
1553:                    Y <= GData[i]->Shape.Rectangle.TopLeft.y + 2)
1554:                 {
1555:                     CurrentOnVertexPoint =
```

```

TPoint(GData[i]->Shape.Rectangle.TopLeft.x, GData[i]->Shape.Rectangle.TopLeft.y);
1555:         Status = true;
1556:     }
1557:     else if(X >= GData[i]->Shape.Rectangle.BottomRight.x - 2 &&
1558:             X <= GData[i]->Shape.Rectangle.BottomRight.x + 2 &&
1559:             Y >= GData[i]->Shape.Rectangle.BottomRight.y - 2 &&
1560:             Y <= GData[i]->Shape.Rectangle.BottomRight.y + 2)
1561:     {
1562:         CurrentOnVertexPoint =
TPoint(GData[i]->Shape.Rectangle.BottomRight.x,
GData[i]->Shape.Rectangle.BottomRight.y);
1563:         Status = true;
1564:     }
1565:     else if(X >= GData[i]->Shape.Rectangle.TopRight.x - 2 &&
1566:             X <= GData[i]->Shape.Rectangle.TopRight.x + 2 &&
1567:             Y >= GData[i]->Shape.Rectangle.TopRight.y - 2 &&
1568:             Y <= GData[i]->Shape.Rectangle.TopRight.y + 2)
1569:     {
1570:         CurrentOnVertexPoint =
TPoint(GData[i]->Shape.Rectangle.TopRight.x, GData[i]->Shape.Rectangle.TopRight.y);
1571:         Status = true;
1572:     }
1573:     else if(X >= GData[i]->Shape.Rectangle.BottomLeft.x - 2 &&
1574:             X <= GData[i]->Shape.Rectangle.BottomLeft.x + 2 &&
1575:             Y >= GData[i]->Shape.Rectangle.BottomLeft.y - 2 &&
1576:             Y <= GData[i]->Shape.Rectangle.BottomLeft.y + 2)
1577:     {
1578:         CurrentOnVertexPoint =
TPoint(GData[i]->Shape.Rectangle.BottomLeft.x, GData[i]->Shape.Rectangle.BottomLeft.y);
1579:         Status = true;
1580:     }
1581:     break;
1582:
1583:     case DRAWCIRCULARARC:
1584:     case DRAWCIRCULARARROW:
1585:     case DRAWCIRCULARARROW2SIDE:
1586:         if(X >= GData[i]->Shape.CircularArc.Start.x - 2 &&
1587:            X <= GData[i]->Shape.CircularArc.Start.x + 2 &&
1588:            Y >= GData[i]->Shape.CircularArc.Start.y - 2 &&
1589:            Y <= GData[i]->Shape.CircularArc.Start.y + 2)
1590:         {
1591:             CurrentOnVertexPoint =
TPoint(GData[i]->Shape.CircularArc.Start.x, GData[i]->Shape.CircularArc.Start.y);
1592:             Status = true;
1593:         }
1594:         else if(X >= GData[i]->Shape.CircularArc.End.x - 2 &&
1595:                 X <= GData[i]->Shape.CircularArc.End.x + 2 &&
1596:                 Y >= GData[i]->Shape.CircularArc.End.y - 2 &&
1597:                 Y <= GData[i]->Shape.CircularArc.End.y + 2)
1598:         {
1599:             CurrentOnVertexPoint = TPoint(GData[i]->Shape.CircularArc.End.
GData[i]->Shape.CircularArc.End.y);
1600:             Status = true;
1601:         }
1602:         else if(X >= GData[i]->Shape.CircularArc.Mid.x - 2 &&
1603:                 X <= GData[i]->Shape.CircularArc.Mid.x + 2 &&
1604:                 Y >= GData[i]->Shape.CircularArc.Mid.y - 2 &&
1605:                 Y <= GData[i]->Shape.CircularArc.Mid.y + 2)
1606:         {
1607:             CurrentOnVertexPoint = TPoint(GData[i]->Shape.CircularArc.Mid.
GData[i]->Shape.CircularArc.Mid.y);
1608:             Status = true;
1609:         }
1610:         break;
1611:     }
1612: }
1613:
1614: if(!ComparePoints(&CurrentOnVertexPoint, &OnVertexPoint))
1615: {
1616:     TColor ColorBackup = Image->Canvas->Pen->Color;
1617:     TPenMode ModeBackup = Image->Canvas->Pen->Mode;
1618:
1619:     if(ReDraw)
1620:     {
1621:         Image->Canvas->Pen->Mode = pmCopy;

```

```
1622:
1623:         Image->Canvas->Pen->Color = Image->Canvas->Brush->Color;
1624:         Image->Canvas->Rectangle(OnVertexPoint.x - 4, OnVertexPoint.y - 4,
OnVertexPoint.x + 4, OnVertexPoint.y + 4);
1625:     }
1626:
1627:     OnVertexPoint = CurrentOnVertexPoint;
1628:
1629:     if(ReDraw)
1630:     {
1631:         if(Status)
1632:         {
1633:             Image->Canvas->Pen->Color = clRed;
1634:             Image->Canvas->Rectangle(OnVertexPoint.x - 4, OnVertexPoint.y - 4,
OnVertexPoint.x + 4, OnVertexPoint.y + 4);
1635:         }
1636:
1637:         Image->Canvas->Pen->Mode = ModeBackup;
1638:         Image->Canvas->Pen->Color = ColorBackup;
1639:     }
1640: }
1641:
1642: return Status;
1643: }
1644: //-----
1645:
1646: void __fastcall PaintTools::DeselectVertexOnPoint(TPoint *Point)
1647: {
1648:     for(register i = 0; i < ShapeCounter; i++)
1649:     {
1650:         switch(GData[i]->ShapeKind)
1651:         {
1652:             case DRAWLINE:
1653:             case DRAWARROW:
1654:             case DRAWARROW2SIDE:
1655:             case DRAWARROW2SIDESEP:
1656:             case DRAWCONTINUOUSLINE:
1657:             case DRAWCHAIN:
1658:             case DRAWPOLYGON:
1659:             case DRAWCYCLOPENTANE:
1660:             case DRAWCIRCLE:
1661:             case DRAWDDOUBLEBOND:
1662:             case DRAWTRIPLEBOND:
1663:             case DRAWUPWEDGE:
1664:             case DRAWDOWNWEDGE:
1665:                 if(ComparePoints(Point, &TPoint(GData[i]->Shape.Line.Start.x,
GData[i]->Shape.Line.Start.y)))
1666:                     GData[i]->Shape.Line.Start.Selected = false;
1667:                 if(ComparePoints(Point, &TPoint(GData[i]->Shape.Line.End.x,
GData[i]->Shape.Line.End.y)))
1668:                     GData[i]->Shape.Line.End.Selected = false;
1669:                 break;
1670:
1671:             case ATOMTEXT:
1672:                 if(ComparePoints(Point, &TPoint(GData[i]->Shape.AtomText.Position.x,
GData[i]->Shape.AtomText.Position.y)))
1673:                     GData[i]->Shape.AtomText.Position.Selected = false;
1674:                 break;
1675:
1676:             case DRAWRECTANGLE:
1677:                 if(ComparePoints(Point, &TPoint(GData[i]->Shape.Rectangle.TopLeft.x,
GData[i]->Shape.Rectangle.TopLeft.y)))
1678:                     GData[i]->Shape.Rectangle.TopLeft.Selected = false;
1679:                 if(ComparePoints(Point, &TPoint(GData[i]->Shape.Rectangle.TopRight.x,
GData[i]->Shape.Rectangle.TopRight.y)))
1680:                     GData[i]->Shape.Rectangle.TopRight.Selected = false;
1681:                 if(ComparePoints(Point,
&TPoint(GData[i]->Shape.Rectangle.BottomLeft.x,
GData[i]->Shape.Rectangle.BottomLeft.y)))
1682:                     GData[i]->Shape.Rectangle.BottomLeft.Selected = false;
1683:                 if(ComparePoints(Point,
&TPoint(GData[i]->Shape.Rectangle.BottomRight.x,
GData[i]->Shape.Rectangle.BottomRight.y)))
1684:                     GData[i]->Shape.Rectangle.BottomRight.Selected = false;
1685:                 break;
```

```
1686:
1687:         case DRAWCIRCULARARC:
1688:         case DRAWCIRCULARARROW:
1689:         case DRAWCIRCULARARROW2SIDE:
1690:             if(ComparePoints(Point, &TPoint(GData[i]->Shape.CircularArc.Start.x,
GData[i]->Shape.CircularArc.Start.y)))
1691:                 GData[i]->Shape.CircularArc.Start.Selected = false;
1692:             if(ComparePoints(Point, &TPoint(GData[i]->Shape.CircularArc.End.x,
GData[i]->Shape.CircularArc.End.y)))
1693:                 GData[i]->Shape.CircularArc.End.Selected = false;
1694:             if(ComparePoints(Point, &TPoint(GData[i]->Shape.CircularArc.Mid.x,
GData[i]->Shape.CircularArc.Mid.y)))
1695:                 GData[i]->Shape.CircularArc.Mid.Selected = false;
1696:             break;
1697:         }
1698:     }
1699: }
1700: //-----
1701:
1702: void __fastcall PaintTools::SelectVertexOnPoint(TPoint *Point, bool DeSelectOther)
1703: {
1704:     for(register i = 0; i < ShapeCounter; i++)
1705:     {
1706:         switch(GData[i]->ShapeKind)
1707:         {
1708:             case DRAWLINE:
1709:             case DRAWCONTINUOUSLINE:
1710:             case DRAWCHAIN:
1711:             case DRAWPOLYGON:
1712:             case DRAWCYCLOPENTANE:
1713:             case DRAWARROW:
1714:             case DRAWARROW2SIDE:
1715:             case DRAWARROW2SIDESEP:
1716:             case DRAWCIRCLE:
1717:             case DRAWDUBLEBOND:
1718:             case DRAWTRIPLEBOND:
1719:             case DRAWUPWEDGE:
1720:             case DRAWDOWNWEDGE:
1721:                 if(ComparePoints(&TPoint(GData[i]->Shape.Line.Start.x,
GData[i]->Shape.Line.Start.y), Point))
1722:                     GData[i]->Shape.Line.Start.Selected = true;
1723:                 else if(DeSelectOther)
1724:                     GData[i]->Shape.Line.Start.Selected = false;
1725:
1726:                 if(ComparePoints(&TPoint(GData[i]->Shape.Line.End.x,
GData[i]->Shape.Line.End.y), Point))
1727:                     GData[i]->Shape.Line.End.Selected = true;
1728:                 else if(DeSelectOther)
1729:                     GData[i]->Shape.Line.End.Selected = false;
1730:                 break;
1731:
1732:             case ATOMTEXT:
1733:                 if(ComparePoints(&TPoint(GData[i]->Shape.AtomText.Position.x,
GData[i]->Shape.AtomText.Position.y), Point))
1734:                     GData[i]->Shape.AtomText.Position.Selected = true;
1735:                 else if(DeSelectOther)
1736:                     GData[i]->Shape.AtomText.Position.Selected = false;
1737:                 break;
1738:
1739:             case DRAWRECTANGLE:
1740:                 if(ComparePoints(&TPoint(GData[i]->Shape.Rectangle.TopLeft.x,
GData[i]->Shape.Rectangle.TopLeft.y), Point))
1741:                     GData[i]->Shape.Rectangle.TopLeft.Selected = true;
1742:                 else if(DeSelectOther)
1743:                     GData[i]->Shape.Rectangle.TopLeft.Selected = false;
1744:
1745:                 if(ComparePoints(&TPoint(GData[i]->Shape.Rectangle.TopRight.x,
GData[i]->Shape.Rectangle.TopRight.y), Point))
1746:                     GData[i]->Shape.Rectangle.TopRight.Selected = true;
1747:                 else if(DeSelectOther)
1748:                     GData[i]->Shape.Rectangle.TopRight.Selected = false;
1749:
1750:                 if(ComparePoints(&TPoint(GData[i]->Shape.Rectangle.BottomRight.x,
GData[i]->Shape.Rectangle.BottomRight.y), Point))
1751:                     GData[i]->Shape.Rectangle.BottomRight.Selected = true;
```

```
1752:         else if(DeSelectOther)
1753:             GData[i]->Shape.Rectangle.BottomRight.Selected = false;
1754:
1755:             if(ComparePoints(&TPoint(GData[i]->Shape.Rectangle.BottomLeft.x,
GData[i]->Shape.Rectangle.BottomLeft.y), Point))
1756:                 GData[i]->Shape.Rectangle.BottomLeft.Selected = true;
1757:             else if(DeSelectOther)
1758:                 GData[i]->Shape.Rectangle.BottomLeft.Selected = false;
1759:             break;
1760:
1761:         case DRAWCIRCULARARC:
1762:         case DRAWCIRCULARARROW:
1763:         case DRAWCIRCULARARROW2SIDE:
1764:             if(ComparePoints(&TPoint(GData[i]->Shape.CircularArc.Start.x,
GData[i]->Shape.CircularArc.Start.y), Point))
1765:                 GData[i]->Shape.CircularArc.Start.Selected = true;
1766:             else if(DeSelectOther)
1767:                 GData[i]->Shape.CircularArc.Start.Selected = false;
1768:
1769:             if(ComparePoints(&TPoint(GData[i]->Shape.CircularArc.End.x,
GData[i]->Shape.CircularArc.End.y), Point))
1770:                 GData[i]->Shape.CircularArc.End.Selected = true;
1771:             else if(DeSelectOther)
1772:                 GData[i]->Shape.CircularArc.End.Selected = false;
1773:
1774:             if(ComparePoints(&TPoint(GData[i]->Shape.CircularArc.Mid.x,
GData[i]->Shape.CircularArc.Mid.y), Point))
1775:                 GData[i]->Shape.CircularArc.Mid.Selected = true;
1776:             else if(DeSelectOther)
1777:                 GData[i]->Shape.CircularArc.Mid.Selected = false;
1778:             break;
1779:         }
1780:     }
1781: }
1782: //-----
1783:
1784: bool __fastcall PaintTools::IsMouseOnShape(int X, int Y, bool ReDraw)
1785: {
1786:     int CalculatedOnShapeX;
1787:     int CalculatedOnShapeY;
1788:     bool Status = false;
1789:     bool ReDrawSign = false;
1790:     bool OnThisShape;
1791:     int StatusShapeNumber = -1;
1792:     int DistanceX, DistanceY, DistanceCurrent;
1793:     int LineStartX, LineStartY;
1794:     int LineEndX, LineEndY;
1795:     TPoint Start, End;
1796:
1797:     for(register i = 0; i < ShapeCounter; i++)
1798:     {
1799:         OnThisShape = false;
1800:
1801:         switch(GData[i]->ShapeKind)
1802:         {
1803:             case DRAWLINE:
1804:             case DRAWCONTINUOUSLINE:
1805:             case DRAWCHAIN:
1806:             case DRAWPOLYGON:
1807:             case DRAWCYCLOPENTANE:
1808:             case DRAWDOUBLEBOND:
1809:             case DRAWTRIPLEBOND:
1810:             case DRAWARROW:
1811:             case DRAWARROW2SIDE:
1812:             case DRAWARROW2SIDESEP:
1813:             case DRAWUPWEDGE:
1814:             case DRAWDOWNWEDGE:
1815:                 Start = TPoint(GData[i]->Shape.Line.Start.x,
GData[i]->Shape.Line.Start.y);
1816:                 End = TPoint(GData[i]->Shape.Line.End.x, GData[i]->Shape.Line.End.y);
1817:
1818:                 DistanceX = End.x - Start.x;
1819:
1820:                 if(DistanceX < 0)
1821:                 {
```

```

1822:         DistanceX *= -1;
1823:         SWAPPOINT(Start, End);
1824:     }
1825:
1826:     if(X >= Start.x - 1 &&
1827:        X <= End.x + 1)
1828:     {
1829:         DistanceY = End.y - Start.y;
1830:
1831:         if(DistanceX != 0 && abs(DistanceX) > abs(DistanceY))
1832:         {
1833:             DistanceCurrent = X - Start.x;
1834:
1835:             CalculatedOnShapeY = DistanceCurrent * DistanceY /
DistanceX + Start.y;
1836:
1837:             if(Y <= CalculatedOnShapeY + 1 && Y >= CalculatedOnShapeY - 1)
1838:             {
1839:                 OnThisShape = true;
1840:                 if(StatusShapeNumber < 0)
1841:                     StatusShapeNumber = i;
1842:             }
1843:         }
1844:         else if(DistanceY != 0)
1845:         {
1846:             DistanceCurrent = Y - Start.y;
1847:             CalculatedOnShapeX = DistanceCurrent * DistanceX /
DistanceY + Start.x;
1848:
1849:             if(X <= CalculatedOnShapeX + 1 && X >= CalculatedOnShapeX - 1)
1850:             {
1851:                 OnThisShape = true;
1852:                 if(StatusShapeNumber < 0)
1853:                     StatusShapeNumber = i;
1854:             }
1855:         }
1856:     }
1857:     break;
1858:
1859: case DRAWRECTANGLE:
1860:     for(register j = 0; j < 4; j++)
1861:     {
1862:         switch(j)
1863:         {
1864:             case 0:
1865:                 LineStartX = GData[i]->Shape.Rectangle.TopLeft.x;
1866:                 LineStartY = GData[i]->Shape.Rectangle.TopLeft.y;
1867:                 LineEndX = GData[i]->Shape.Rectangle.TopRight.x;
1868:                 LineEndY = GData[i]->Shape.Rectangle.TopRight.y;
1869:                 break;
1870:
1871:             case 1:
1872:                 LineStartX = GData[i]->Shape.Rectangle.TopRight.x;
1873:                 LineStartY = GData[i]->Shape.Rectangle.TopRight.y;
1874:                 LineEndX = GData[i]->Shape.Rectangle.BottomRight.x;
1875:                 LineEndY = GData[i]->Shape.Rectangle.BottomRight.y;
1876:                 break;
1877:
1878:             case 2:
1879:                 LineStartX = GData[i]->Shape.Rectangle.BottomRight.x;
1880:                 LineStartY = GData[i]->Shape.Rectangle.BottomRight.y;
1881:                 LineEndX = GData[i]->Shape.Rectangle.BottomLeft.x;
1882:                 LineEndY = GData[i]->Shape.Rectangle.BottomLeft.y;
1883:                 break;
1884:
1885:             case 3:
1886:                 LineStartX = GData[i]->Shape.Rectangle.BottomLeft.x;
1887:                 LineStartY = GData[i]->Shape.Rectangle.BottomLeft.y;
1888:                 LineEndX = GData[i]->Shape.Rectangle.TopLeft.x;
1889:                 LineEndY = GData[i]->Shape.Rectangle.TopLeft.y;
1890:                 break;
1891:         }
1892:     }

```

```

1893:         DistanceX = LineEndX - LineStartX;
1894:         DistanceY = LineEndY - LineStartY;
1895:
1896:         if(DistanceX != 0 && abs(DistanceX) > abs(DistanceY))
1897:         {
1898:             if(LineStartX > LineEndX)
1899:             {
1900:                 SWAPINTEGER(LineStartX, LineEndX);
1901:                 SWAPINTEGER(LineStartY, LineEndY);
1902:             }
1903:
1904:             if(X >= LineStartX - 1 &&
1905:                X <= LineEndX + 1)
1906:             {
1907:                 DistanceCurrent = X - LineStartX;
1908:                 CalculatedOnShapeY = DistanceCurrent * DistanceY /
DistanceX + LineStartY;
1909:
1910:                 if(Y <= CalculatedOnShapeY + 1 && Y >=
CalculatedOnShapeY - 1)
1911:                 {
1912:                     OnThisShape = true;
1913:                     if(StatusShapeNumber < 0)
1914:                         StatusShapeNumber = i;
1915:                     break;
1916:                 }
1917:             }
1918:         }
1919:         else if(DistanceY != 0)
1920:         {
1921:             if(LineStartY > LineEndY)
1922:             {
1923:                 SWAPINTEGER(LineStartX, LineEndX);
1924:                 SWAPINTEGER(LineStartY, LineEndY);
1925:             }
1926:
1927:             if(Y >= LineStartY - 1 &&
1928:                Y <= LineEndY + 1)
1929:             {
1930:                 DistanceCurrent = Y - LineStartY;
1931:                 CalculatedOnShapeX = DistanceCurrent * DistanceX /
DistanceY + LineStartX;
1932:
1933:                 if(X <= CalculatedOnShapeX + 1 && X >=
CalculatedOnShapeX - 1)
1934:                 {
1935:                     OnThisShape = true;
1936:                     if(StatusShapeNumber < 0)
1937:                         StatusShapeNumber = i;
1938:                     break;
1939:                 }
1940:             }
1941:         }
1942:         break;
1943:
1944:     case DRAWCIRCLE:
1945:     {
1946:         int Radius = sqrt(pow(GData[i]->Shape.Line.Start.x -
GData[i]->Shape.Line.End.x, 2) + pow(GData[i]->Shape.Line.Start.y -
GData[i]->Shape.Line.End.y, 2));
1947:
1948:         int CurrentDistance = sqrt(pow(GData[i]->Shape.Line.Start.x -
2) + pow(GData[i]->Shape.Line.Start.y - Y, 2));
1949:
1950:         if(CurrentDistance <= Radius + 1 && CurrentDistance >= Radius
1)
1951:         {
1952:             OnThisShape = true;
1953:             if(StatusShapeNumber < 0)
1954:                 StatusShapeNumber = i;
1955:         }
1956:     }
1957:     break;
1958:
1959:     case ATOMTEXT:

```

```
1960:         if(X > GData[i]->Shape.AtomText.Position.x &&
1961:             X < GData[i]->Shape.AtomText.Position.x +
            GData[i]->Shape.AtomText.Width &&
1962:             Y > GData[i]->Shape.AtomText.Position.y &&
1963:             Y < GData[i]->Shape.AtomText.Position.y +
            GData[i]->Shape.AtomText.Height)
1964:         {
1965:             OnThisShape = true;
1966:             if(StatusShapeNumber < 0)
1967:                 StatusShapeNumber = i;
1968:         }
1969:         break;
1970:
1971:     case DRAWCIRCULARARC:
1972:     case DRAWCIRCULARARROW:
1973:     case DRAWCIRCULARARROW2SIDE:
1974:         TPoint StartBak = TPoint(GData[i]->Shape.CircularArc.Start.x -
            GData[i]->Shape.CircularArc.Mid.x, -(GData[i]->Shape.CircularArc.Start.y -
            GData[i]->Shape.CircularArc.Mid.y)),
1975:         EndBak = TPoint(GData[i]->Shape.CircularArc.End.x -
            GData[i]->Shape.CircularArc.Mid.x, -(GData[i]->Shape.CircularArc.End.y -
            GData[i]->Shape.CircularArc.Mid.y));
1976:
1977:         int UnderFraction = 2 * (StartBak.x * EndBak.y - EndBak.x *
            StartBak.y);
1978:         if(UnderFraction)
1979:         {
1980:             int CenterX = (EndBak.y * (pow(StartBak.x, 2) + pow(StartBak.y,
1981: 2)) - StartBak.y * (pow(EndBak.x, 2) + pow(EndBak.y, 2))) /
                UnderFraction;
1982:             int CenterY = (StartBak.x * (pow(EndBak.x, 2) + pow(EndBak.y,
1983: 2)) - EndBak.x * (pow(StartBak.x, 2) + pow(StartBak.y, 2))) /
                UnderFraction;
1984:
1985:             CenterX += GData[i]->Shape.CircularArc.Mid.x;
1986:             CenterY = -(CenterY - GData[i]->Shape.CircularArc.Mid.y);
1987:
1988:             int Radius = sqrt(pow(CenterX -
            GData[i]->Shape.CircularArc.Start.x, 2) + pow(CenterY -
            GData[i]->Shape.CircularArc.Start.y, 2));
1989:             int CurrentDistance = sqrt(pow(CenterX - X, 2) + pow(CenterY -
            Y, 2));
1990:
1991:             if(CurrentDistance <= Radius + 1 && CurrentDistance >= Radius -
            1)
1992:             {
1993:                 OnThisShape = true;
1994:                 if(StatusShapeNumber < 0)
1995:                     StatusShapeNumber = i;
1996:             }
1997:         }
1998:         break;
1999:     }
2000:     if(GData[i]->Pen.Color == ONSHAPECOLOR && !OnThisShape || (StatusShapeNumber
            -1 && Status))
2001:     {
2002:         GData[i]->Pen.Color = (TColor)NORMALCOLOR;
2003:         ReDrawSign = true;
2004:     }
2005:     else if(OnThisShape && !Status)
2006:     {
2007:         GData[i]->Pen.Color = (TColor)ONSHAPECOLOR;
2008:         ReDrawSign = true;
2009:         OnShapeNumber = i;
2010:     }
2011:
2012:     if(StatusShapeNumber > -1)
2013:         Status = true;
2014: }
2015:
2016: if(ReDrawSign)
2017: {
2018:     DrawShapes();
2019: }
2020:
```

```
2021:     if(Status)
2022:         switch(GData[OnShapeNumber]->ShapeKind)
2023:         {
2024:             case DRAWLINE:
2025:             case DRAWCONTINUOUSLINE:
2026:             case DRAWCHAIN:
2027:             case DRAWPOLYGON:
2028:             case DRAWCYCLOPENTANE:
2029:             case DRAWDDOUBLEBOND:
2030:             case DRAWTRIPLEBOND:
2031:                 MainForm->Image->PopupMenu = MainForm->GraphMenu1;
2032:                 break;
2033:
2034:             case ATOMTEXT:
2035:                 MainForm->Image->PopupMenu = MainForm->GraphMenu3;
2036:                 break;
2037:
2038:             default:
2039:                 MainForm->Image->PopupMenu = NULL;
2040:                 break;
2041:         }
2042:     else
2043:         MainForm->Image->PopupMenu = MainForm->GraphMenu2;
2044:
2045:     return Status;
2046: }
2047: //-----
2048:
2049: void __fastcall PaintTools::DeselectGroup(int GroupNum, bool ReDraw)
2050: {
2051:     for(register i = 0; i < ShapeCounter; i++)
2052:         if(GData[i]->GroupNumber == GroupNum)
2053:             DeselectShape(i, false);
2054:
2055:     if(ReDraw)
2056:         DrawShapes();
2057: }
2058: //-----
2059:
2060: void __fastcall PaintTools::SelectGroup(int GroupNum, bool DeSelectOther, bool ReDraw)
2061: {
2062:     for(register i = 0; i < ShapeCounter; i++)
2063:         if(GData[i]->GroupNumber == GroupNum)
2064:             SelectShape(i, false, false);
2065:         else if(DeSelectOther)
2066:             DeselectShape(i, false);
2067:
2068:     if(ReDraw)
2069:         DrawShapes();
2070: }
2071: //-----
2072:
2073: void __fastcall PaintTools::ShowAtomTextEdit(int X, int Y)
2074: {
2075:     Form1->AtomTextEdit->Text = "";
2076:     Form1->Left = X;
2077:     Form1->Top = Y;
2078:     Form1->Show();
2079:
2080:     Form1->AtomTextEdit->SetFocus();
2081: }
2082: //-----
2083:
2084: void __fastcall PaintTools::DetermineResultantPoint(void)
2085: {
2086:     SelectedArea = TRect(Image->Width, Image->Height, -1, -1);
2087:
2088:     for(register i = 0; i < ShapeCounter; i++)
2089:     {
2090:         switch(GData[i]->ShapeKind)
2091:         {
2092:             case DRAWLINE:
2093:             case DRAWARROW:
2094:             case DRAWARROW2SIDE:
2095:             case DRAWARROW2SIDESEP:
```

```

2096:         case DRAWCONTINUOUSLINE:
2097:         case DRAWCHAIN:
2098:         case DRAWPOLYGON:
2099:         case DRAWCYCLOPENTANE:
2100:         case DRAWCIRCLE:
2101:         case DRAWDOUBLEBOND:
2102:         case DRAWTRIPLEBOND:
2103:         case DRAWUPWEDGE:
2104:         case DRAWDOWNWEDGE:
2105:             if (GData[i] -> Shape.Line.Start.Selected)
2106:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.Line.Start.x,
GData[i] -> Shape.Line.Start.y));
2107:             if (GData[i] -> Shape.Line.End.Selected)
2108:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.Line.End.x,
GData[i] -> Shape.Line.End.y));
2109:             break;
2110:
2111:         case ATOMTEXT:
2112:             if (GData[i] -> Shape.AtomText.Position.Selected)
2113:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.AtomText.Position
GData[i] -> Shape.AtomText.Position.y));
2114:             break;
2115:
2116:         case DRAWRECTANGLE:
2117:             if (GData[i] -> Shape.Rectangle.TopLeft.Selected)
2118:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.Rectangle.TopLeft
GData[i] -> Shape.Rectangle.TopLeft.y));
2119:
2120:             if (GData[i] -> Shape.Rectangle.TopRight.Selected)
2121:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.Rectangle.TopRight.x,
GData[i] -> Shape.Rectangle.TopRight.y));
2122:
2123:             if (GData[i] -> Shape.Rectangle.BottomRight.Selected)
2124:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.Rectangle.BottomRight.x,
GData[i] -> Shape.Rectangle.BottomRight.y));
2125:
2126:             if (GData[i] -> Shape.Rectangle.BottomLeft.Selected)
2127:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.Rectangle.BottomLeft.x,
GData[i] -> Shape.Rectangle.BottomLeft.y));
2128:             break;
2129:
2130:         case DRAWCIRCULARARC:
2131:         case DRAWCIRCULARARROW:
2132:         case DRAWCIRCULARARROW2SIDE:
2133:             if (GData[i] -> Shape.CircularArc.Start.Selected)
2134:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.CircularArc.Start
GData[i] -> Shape.CircularArc.Start.y));
2135:             if (GData[i] -> Shape.CircularArc.End.Selected)
2136:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.CircularArc.End.x,
GData[i] -> Shape.CircularArc.End.y));
2137:             if (GData[i] -> Shape.CircularArc.Mid.Selected)
2138:                 SelectedAreaContain(&TPoint(GData[i] -> Shape.CircularArc.Mid.x,
GData[i] -> Shape.CircularArc.Mid.y));
2139:             break;
2140:     }
2141: }
2142:
2143: ResultantPoint.x = (SelectedArea.Left + SelectedArea.Right) / 2;
2144: ResultantPoint.y = (SelectedArea.Top + SelectedArea.Bottom) / 2;
2145: }
2146: //-----
2147:
2148: void __fastcall PaintTools::RotateSelectedVertex(TPoint *NewPoint)
2149: {
2150:     TPoint Point;
2151:     float Angle = M_PI * (NewPoint->x + NewPoint->y) / 180;
2152:
2153:     for(register i = 0; i < ShapeCounter; i++)
2154:     {
2155:         switch(GData[i] -> ShapeKind)
2156:         {
2157:             case DRAWLINE:

```

```
2158:         case DRAWARROW:
2159:         case DRAWARROW2SIDE:
2160:         case DRAWARROW2SIDESEP:
2161:         case DRAWCONTINUOUSLINE:
2162:         case DRAWCHAIN:
2163:         case DRAWPOLYGON:
2164:         case DRAWCYCLOPENTANE:
2165:         case DRAWCIRCLE:
2166:         case DRAWDOUBLEBOND:
2167:         case DRAWTRIPLEBOND:
2168:         case DRAWUPWEDGE:
2169:         case DRAWDOWNWEDGE:
2170:             if (GData[i]->Shape.Line.Start.Selected)
2171:             {
2172:                 GData[i]->Shape.Line.Start.Angle += Angle;
2173:             }
2174:
2175:             if (GData[i]->Shape.Line.End.Selected)
2176:             {
2177:                 GData[i]->Shape.Line.End.Angle += Angle;
2178:             }
2179:             break;
2180:
2181:         case ATOMTEXT:
2182:             if (GData[i]->Shape.AtomText.Position.Selected)
2183:                 GData[i]->Shape.AtomText.Position.Angle += Angle;
2184:             break;
2185:
2186:         case DRAWRECTANGLE:
2187:             if (GData[i]->Shape.Rectangle.TopLeft.Selected ||
2188:                 GData[i]->Shape.Rectangle.TopRight.Selected ||
2189:                 GData[i]->Shape.Rectangle.BottomRight.Selected ||
2190:                 GData[i]->Shape.Rectangle.BottomLeft.Selected)
2191:             {
2192:                 GData[i]->Shape.Rectangle.TopLeft.Angle += Angle;
2193:                 GData[i]->Shape.Rectangle.BottomLeft.Angle += Angle;
2194:                 GData[i]->Shape.Rectangle.TopRight.Angle += Angle;
2195:                 GData[i]->Shape.Rectangle.BottomRight.Angle += Angle;
2196:             }
2197:             break;
2198:
2199:         case DRAWCIRCULARARC:
2200:         case DRAWCIRCULARARROW:
2201:         case DRAWCIRCULARARROW2SIDE:
2202:             if (GData[i]->Shape.CircularArc.Start.Selected)
2203:             {
2204:                 GData[i]->Shape.CircularArc.Start.Angle += Angle;
2205:             }
2206:
2207:             if (GData[i]->Shape.CircularArc.End.Selected)
2208:             {
2209:                 GData[i]->Shape.CircularArc.End.Angle += Angle;
2210:             }
2211:
2212:             if (GData[i]->Shape.CircularArc.Mid.Selected)
2213:             {
2214:                 GData[i]->Shape.CircularArc.Mid.Angle += Angle;
2215:             }
2216:             break;
2217:     }
2218: }
2219: }
2220: //-----
2221:
2222: void __fastcall PaintTools::MoveSelectedVertex(TPoint *NewPoint)
2223: {
2224:     for(register i = 0; i < ShapeCounter; i++)
2225:     {
2226:         switch(GData[i]->ShapeKind)
2227:         {
2228:             case DRAWLINE:
2229:             case DRAWARROW:
2230:             case DRAWARROW2SIDE:
2231:             case DRAWARROW2SIDESEP:
2232:             case DRAWCONTINUOUSLINE:
```

```
2233:         case DRAWCHAIN:
2234:         case DRAWPOLYGON:
2235:         case DRAWCYCLOPENTANE:
2236:         case DRAWCIRCLE:
2237:         case DRAWDOUBLEBOND:
2238:         case DRAWTRIPLEBOND:
2239:         case DRAWUPWEDGE:
2240:         case DRAWDOWNWEDGE:
2241:             if (GData[i]->Shape.Line.Start.Selected)
2242:             {
2243:                 GData[i]->Shape.Line.Start.x += NewPoint->x;
2244:                 GData[i]->Shape.Line.Start.y += NewPoint->y;
2245:             }
2246:
2247:             if (GData[i]->Shape.Line.End.Selected)
2248:             {
2249:                 GData[i]->Shape.Line.End.x += NewPoint->x;
2250:                 GData[i]->Shape.Line.End.y += NewPoint->y;
2251:             }
2252:             break;
2253:
2254:         case ATOMTEXT:
2255:             if (GData[i]->Shape.AtomText.Position.Selected)
2256:             {
2257:                 GData[i]->Shape.AtomText.Position.x += NewPoint->x;
2258:                 GData[i]->Shape.AtomText.Position.y += NewPoint->y;
2259:             }
2260:             break;
2261:
2262:         case DRAWRECTANGLE:
2263:             if (MoveSelShape)
2264:             {
2265:                 if (GData[i]->Shape.Rectangle.TopLeft.Selected)
2266:                 {
2267:                     GData[i]->Shape.Rectangle.TopLeft.x += NewPoint->x;
2268:                     GData[i]->Shape.Rectangle.BottomLeft.x += NewPoint->x;
2269:                     GData[i]->Shape.Rectangle.BottomRight.x += NewPoint->x;
2270:                     GData[i]->Shape.Rectangle.TopRight.x += NewPoint->x;
2271:
2272:                     GData[i]->Shape.Rectangle.TopLeft.y += NewPoint->y;
2273:                     GData[i]->Shape.Rectangle.BottomLeft.y += NewPoint->y;
2274:                     GData[i]->Shape.Rectangle.BottomRight.y += NewPoint->y;
2275:                     GData[i]->Shape.Rectangle.TopRight.y += NewPoint->y;
2276:                 }
2277:             }
2278:             else
2279:             {
2280:                 if (GData[i]->Shape.Rectangle.TopLeft.Selected)
2281:                 {
2282:                     GData[i]->Shape.Rectangle.TopLeft.x += NewPoint->x;
2283:                     GData[i]->Shape.Rectangle.BottomLeft.x += NewPoint->x;
2284:                     GData[i]->Shape.Rectangle.TopLeft.y += NewPoint->y;
2285:                     GData[i]->Shape.Rectangle.TopRight.y += NewPoint->y;
2286:                 }
2287:
2288:                 if (GData[i]->Shape.Rectangle.TopRight.Selected)
2289:                 {
2290:                     GData[i]->Shape.Rectangle.TopRight.x += NewPoint->x;
2291:                     GData[i]->Shape.Rectangle.BottomRight.x += NewPoint->x;
2292:                     GData[i]->Shape.Rectangle.TopRight.y += NewPoint->y;
2293:                     GData[i]->Shape.Rectangle.TopLeft.y += NewPoint->y;
2294:                 }
2295:
2296:                 if (GData[i]->Shape.Rectangle.BottomRight.Selected)
2297:                 {
2298:                     GData[i]->Shape.Rectangle.BottomRight.x += NewPoint->x;
2299:                     GData[i]->Shape.Rectangle.TopRight.x += NewPoint->x;
2300:                     GData[i]->Shape.Rectangle.BottomRight.y += NewPoint->y;
2301:                     GData[i]->Shape.Rectangle.BottomLeft.y += NewPoint->y;
2302:                 }
2303:
2304:                 if (GData[i]->Shape.Rectangle.BottomLeft.Selected)
2305:                 {
2306:                     GData[i]->Shape.Rectangle.BottomLeft.x += NewPoint->x;
2307:                     GData[i]->Shape.Rectangle.TopLeft.x += NewPoint->x;
```

```
2308:             GData[i]->Shape.Rectangle.BottomLeft.y += NewPoint->y;
2309:             GData[i]->Shape.Rectangle.BottomRight.y += NewPoint->y;
2310:         }
2311:     }
2312:     break;
2313:
2314:     case DRAWCIRCULARARC:
2315:     case DRAWCIRCULARARROW:
2316:     case DRAWCIRCULARARROW2SIDE:
2317:         if (GData[i]->Shape.CircularArc.Start.Selected)
2318:         {
2319:             GData[i]->Shape.CircularArc.Start.x += NewPoint->x;
2320:             GData[i]->Shape.CircularArc.Start.y += NewPoint->y;
2321:         }
2322:
2323:         if (GData[i]->Shape.CircularArc.End.Selected)
2324:         {
2325:             GData[i]->Shape.CircularArc.End.x += NewPoint->x;
2326:             GData[i]->Shape.CircularArc.End.y += NewPoint->y;
2327:         }
2328:
2329:         if (GData[i]->Shape.CircularArc.Mid.Selected)
2330:         {
2331:             GData[i]->Shape.CircularArc.Mid.x += NewPoint->x;
2332:             GData[i]->Shape.CircularArc.Mid.y += NewPoint->y;
2333:         }
2334:         break;
2335:     }
2336: }
2337: }
2338: //-----
2339:
2340: void __fastcall PaintTools::SelectedAreaContain(TPoint *Point)
2341: {
2342:     if (Point->x < SelectedArea.Left)
2343:         SelectedArea.Left = Point->x;
2344:     if (Point->x > SelectedArea.Right)
2345:         SelectedArea.Right = Point->x;
2346:     if (Point->y < SelectedArea.Top)
2347:         SelectedArea.Top = Point->y;
2348:     if (Point->y > SelectedArea.Bottom)
2349:         SelectedArea.Bottom = Point->y;
2350: }
2351: //-----
2352:
2353: TPoint __fastcall PaintTools::FindNearestFixedPoint(TPoint *Start, TPoint *End, int Size)
2354: {
2355:     int BigY = Start->y > End->y ? Start->y : End->y;
2356:     float Slope, Angle;
2357:     TPoint StartBak = TPoint(Start->x, -(Start->y - BigY)),
2358:           EndBak = TPoint(End->x, -(End->y - BigY));
2359:     TPoint ReturnValue = *Start;
2360:
2361:     int LineLength = sqrt(pow(StartBak.x - EndBak.x, 2) + pow(StartBak.y - EndBak.y,
2362: 2));
2363:
2364:     if (abs(Start->x - End->x) >= abs(Start->y - End->y) && End->x != Start->x)
2365:     {
2366:         Slope = float(EndBak.y - StartBak.y) / float(EndBak.x - StartBak.x);
2367:         Angle = atan(Slope);
2368:
2369:         RotatePoint(&StartBak, &TPoint(0, 0), -Angle, false);
2370:         RotatePoint(&EndBak, &TPoint(0, 0), -Angle, false);
2371:
2372:         ReturnValue.y = StartBak.y;
2373:         ReturnValue.x = EndBak.x + (StartBak.x < EndBak.x ? -(LineLength%Size) :
2374: (LineLength%Size));
2375:         RotatePoint(&ReturnValue, &TPoint(0, 0), Angle, false);
2376:         ReturnValue.y = BigY - ReturnValue.y;
2377:     }
2378:     else if (End->y != Start->y)
2379:     {
2380:         Slope = float(EndBak.x - StartBak.x) / float(EndBak.y - StartBak.y);
2381:         Angle = atan(Slope);
```

```
2381:         RotatePoint(&StartBak, &TPoint(0, 0), Angle, false);
2382:         RotatePoint(&EndBak, &TPoint(0, 0), Angle, false);
2383:
2384:         ReturnValue.x = StartBak.x;
2385:         ReturnValue.y = EndBak.y + (StartBak.y < EndBak.y ? -(LineLength%Size) :
(LineLength%Size));
2386:         RotatePoint(&ReturnValue, &TPoint(0, 0), -Angle, false);
2387:         ReturnValue.y = BigY - ReturnValue.y;
2388:     }
2389:
2390:     return ReturnValue;
2391: }
2392: //-----
2393:
2394: TPoint __fastcall PaintTools::FindNearestFixedRotationPoint(TPoint *Start, TPoint *End
float FixRotationAngle)
2395: {
2396:     //int BigY = Start->y > End->y ? Start->y : End->y;
2397:     int BigY = Start->y;
2398:     float Slope, Angle;
2399:     TPoint StartBak = TPoint(0, -(Start->y - BigY)),
2400:         EndBak = TPoint(End->x - Start->x, -(End->y - BigY));
2401:     TPoint ReturnValue = *Start;
2402:     int DistanceAngle;
2403:
2404:     if(abs(Start->x - End->x) >= abs(Start->y - End->y) && End->x != Start->x)
2405:     {
2406:         Slope = float(EndBak.y - StartBak.y) / float(EndBak.x - StartBak.x);
2407:         Angle = atan(Slope);
2408:
2409:         DistanceAngle = int(Angle/(FixRotationAngle/2));
2410:         if(Angle > 0)
2411:             DistanceAngle++;
2412:
2413:         RotatePoint(&StartBak, &TPoint(0, 0), -Angle, false);
2414:         RotatePoint(&EndBak, &TPoint(0, 0), -Angle, false);
2415:
2416:         if(DistanceAngle%2)
2417:             Angle = (DistanceAngle - 1) / 2 * FixRotationAngle;
2418:         else
2419:             Angle = DistanceAngle / 2 * FixRotationAngle;
2420:
2421:         ReturnValue = EndBak;
2422:         RotatePoint(&ReturnValue, &TPoint(0, 0), Angle, false);
2423:         ReturnValue.x += Start->x;
2424:         ReturnValue.y = BigY - ReturnValue.y;
2425:     }
2426:     else if(End->y != Start->y)
2427:     {
2428:         Slope = float(EndBak.x - StartBak.x) / float(EndBak.y - StartBak.y);
2429:         Angle = atan(Slope);
2430:
2431:         DistanceAngle = int(Angle/(FixRotationAngle/2));
2432:         if(Angle > 0)
2433:             DistanceAngle++;
2434:
2435:         RotatePoint(&StartBak, &TPoint(0, 0), Angle, false);
2436:         RotatePoint(&EndBak, &TPoint(0, 0), Angle, false);
2437:
2438:         if(DistanceAngle%2)
2439:             Angle = (DistanceAngle - 1) / 2 * FixRotationAngle;
2440:         else
2441:             Angle = DistanceAngle / 2 * FixRotationAngle;
2442:
2443:         ReturnValue = EndBak;
2444:         RotatePoint(&ReturnValue, &TPoint(0, 0), -Angle, false);
2445:         ReturnValue.x += Start->x;
2446:         ReturnValue.y = BigY - ReturnValue.y;
2447:     }
2448:
2449:     return ReturnValue;
2450: }
2451: //-----
2452:
2453: bool __fastcall PaintTools::IsShapeInSelectedArea(int ShapeNum)
```

```
2454: {
2455:     bool ReturnValue = false;
2456:
2457:     switch(GData[ShapeNum]->ShapeKind)
2458:     {
2459:         case DRAWLINE:
2460:         case DRAWARROW:
2461:         case DRAWARROW2SIDE:
2462:         case DRAWARROW2SIDESEP:
2463:         case DRAWCONTINUOUSLINE:
2464:         case DRAWCHAIN:
2465:         case DRAWPOLYGON:
2466:         case DRAWCYCLOPENTANE:
2467:         case DRAWCIRCLE:
2468:         case DRAWDOUBLEBOND:
2469:         case DRAWTRIPLEBOND:
2470:         case DRAWUPWEDGE:
2471:         case DRAWDOWNWEDGE:
2472:             if (GData[ShapeNum]->Shape.Line.Start.x < SelectingArea.Right &&
2473:                 GData[ShapeNum]->Shape.Line.Start.x > SelectingArea.Left &&
2474:                 GData[ShapeNum]->Shape.Line.Start.y < SelectingArea.Bottom &&
2475:                 GData[ShapeNum]->Shape.Line.Start.y > SelectingArea.Top &&
2476:                 GData[ShapeNum]->Shape.Line.End.x < SelectingArea.Right &&
2477:                 GData[ShapeNum]->Shape.Line.End.x > SelectingArea.Left &&
2478:                 GData[ShapeNum]->Shape.Line.End.y < SelectingArea.Bottom &&
2479:                 GData[ShapeNum]->Shape.Line.End.y > SelectingArea.Top)
2480:                 ReturnValue = true;
2481:             break;
2482:
2483:         case ATOMTEXT:
2484:             if (GData[ShapeNum]->Shape.AtomText.Position.x < SelectingArea.Right &&
2485:                 GData[ShapeNum]->Shape.AtomText.Position.x > SelectingArea.Left &&
2486:                 GData[ShapeNum]->Shape.AtomText.Position.y < SelectingArea.Bottom &&
2487:                 GData[ShapeNum]->Shape.AtomText.Position.y > SelectingArea.Top)
2488:                 ReturnValue = true;
2489:             break;
2490:
2491:         case DRAWRECTANGLE:
2492:             if (GData[ShapeNum]->Shape.Rectangle.TopLeft.x < SelectingArea.Right &&
2493:                 GData[ShapeNum]->Shape.Rectangle.TopLeft.x > SelectingArea.Left &&
2494:                 GData[ShapeNum]->Shape.Rectangle.TopLeft.y < SelectingArea.Bottom &&
2495:                 GData[ShapeNum]->Shape.Rectangle.TopLeft.y > SelectingArea.Top &&
2496:                 GData[ShapeNum]->Shape.Rectangle.BottomRight.x < SelectingArea.Right &&
2497:                 GData[ShapeNum]->Shape.Rectangle.BottomRight.x > SelectingArea.Left &&
2498:                 GData[ShapeNum]->Shape.Rectangle.BottomRight.y < SelectingArea.Bottom &&
2499:                 GData[ShapeNum]->Shape.Rectangle.BottomRight.y > SelectingArea.Top &&
2500:                 GData[ShapeNum]->Shape.Rectangle.TopRight.x < SelectingArea.Right &&
2501:                 GData[ShapeNum]->Shape.Rectangle.TopRight.x > SelectingArea.Left &&
2502:                 GData[ShapeNum]->Shape.Rectangle.TopRight.y < SelectingArea.Bottom &&
2503:                 GData[ShapeNum]->Shape.Rectangle.TopRight.y > SelectingArea.Top &&
2504:                 GData[ShapeNum]->Shape.Rectangle.BottomLeft.x < SelectingArea.Right &&
2505:                 GData[ShapeNum]->Shape.Rectangle.BottomLeft.x > SelectingArea.Left &&
2506:                 GData[ShapeNum]->Shape.Rectangle.BottomLeft.y < SelectingArea.Bottom &&
2507:                 GData[ShapeNum]->Shape.Rectangle.BottomLeft.y > SelectingArea.Top)
2508:                 ReturnValue = true;
2509:             break;
2510:
2511:         case DRAWCIRCULARARC:
2512:         case DRAWCIRCULARARROW:
2513:         case DRAWCIRCULARARROW2SIDE:
2514:             if (GData[ShapeNum]->Shape.CircularArc.Start.x < SelectingArea.Right &&
2515:                 GData[ShapeNum]->Shape.CircularArc.Start.x > SelectingArea.Left &&
2516:                 GData[ShapeNum]->Shape.CircularArc.Start.y < SelectingArea.Bottom &&
2517:                 GData[ShapeNum]->Shape.CircularArc.Start.y > SelectingArea.Top &&
2518:                 GData[ShapeNum]->Shape.CircularArc.End.x < SelectingArea.Right &&
2519:                 GData[ShapeNum]->Shape.CircularArc.End.x > SelectingArea.Left &&
2520:                 GData[ShapeNum]->Shape.CircularArc.End.y < SelectingArea.Bottom &&
2521:                 GData[ShapeNum]->Shape.CircularArc.End.y > SelectingArea.Top &&
2522:                 GData[ShapeNum]->Shape.CircularArc.Mid.x < SelectingArea.Right &&
2523:                 GData[ShapeNum]->Shape.CircularArc.Mid.x > SelectingArea.Left &&
2524:                 GData[ShapeNum]->Shape.CircularArc.Mid.y < SelectingArea.Bottom &&
2525:                 GData[ShapeNum]->Shape.CircularArc.Mid.y > SelectingArea.Top)
2526:                 ReturnValue = true;
2527:             break;
```

```

2528:     }
2529:
2530:     return ReturnValue;
2531: }
2532: //-----
2533:
2534: void __fastcall PaintTools::FixRotatedPoints(void)
2535: {
2536:     TPoint Point;
2537:     float RotationAngle;
2538:
2539:     for(register i = 0; i < ShapeCounter; i++)
2540:     {
2541:         switch(GData[i]->ShapeKind)
2542:         {
2543:             case DRAWLINE:
2544:             case DRAWARROW:
2545:             case DRAWARROW2SIDE:
2546:             case DRAWARROW2SIDESEP:
2547:             case DRAWCONTINUOUSLINE:
2548:             case DRAWCHAIN:
2549:             case DRAWPOLYGON:
2550:             case DRAWCYCLOPENTANE:
2551:             case DRAWCIRCLE:
2552:             case DRAWDOUBLEBOND:
2553:             case DRAWTRIPLEBOND:
2554:             case DRAWUPWEDGE:
2555:             case DRAWDOWNWEDGE:
2556:                 RotationAngle = RotationSize(GData[i]->Shape.Line.Start.Angle);
2557:                 if(RotationAngle != 0.0)
2558:                 {
2559:                     Point = TPoint(GData[i]->Shape.Line.Start.x,
GData[i]->Shape.Line.Start.y);
2560:                     RotatePoint(&Point, &ResultantPoint, RotationAngle);
2561:                     GData[i]->Shape.Line.Start.x = Point.x;
2562:                     GData[i]->Shape.Line.Start.y = Point.y;
2563:                     GData[i]->Shape.Line.Start.Angle = 0;
2564:                 }
2565:
2566:                 RotationAngle = RotationSize(GData[i]->Shape.Line.End.Angle);
2567:                 if(RotationAngle != 0.0)
2568:                 {
2569:                     Point = TPoint(GData[i]->Shape.Line.End.x,
GData[i]->Shape.Line.End.y);
2570:                     RotatePoint(&Point, &ResultantPoint, RotationAngle);
2571:                     GData[i]->Shape.Line.End.x = Point.x;
2572:                     GData[i]->Shape.Line.End.y = Point.y;
2573:                     GData[i]->Shape.Line.End.Angle = 0;
2574:                 }
2575:                 break;
2576:
2577:             case ATOMTEXT:
2578:                 RotationAngle =
RotationSize(GData[i]->Shape.AtomText.Position.Angle);
2579:                 if(RotationAngle != 0.0)
2580:                 {
2581:                     Point = TPoint(GData[i]->Shape.AtomText.Position.x,
GData[i]->Shape.AtomText.Position.y);
2582:                     RotatePoint(&Point, &ResultantPoint, RotationAngle);
2583:                     GData[i]->Shape.AtomText.Position.x = Point.x;
2584:                     GData[i]->Shape.AtomText.Position.y = Point.y;
2585:                     GData[i]->Shape.AtomText.Position.Angle = 0;
2586:                 }
2587:                 break;
2588:
2589:             case DRAWRECTANGLE:
2590:                 RotationAngle =
RotationSize(GData[i]->Shape.Rectangle.TopLeft.Angle);
2591:                 if(RotationAngle != 0.0)
2592:                 {
2593:                     Point = TPoint(GData[i]->Shape.Rectangle.TopLeft.x,
GData[i]->Shape.Rectangle.TopLeft.y);
2594:                     RotatePoint(&Point, &ResultantPoint, RotationAngle);
2595:                     GData[i]->Shape.Rectangle.TopLeft.x = Point.x;
2596:                     GData[i]->Shape.Rectangle.TopLeft.y = Point.y;

```

```
2597:         GData[i]->Shape.Rectangle.TopLeft.Angle = 0;
2598:     }
2599:
2600:     RotationAngle =
RotationSize(GData[i]->Shape.Rectangle.TopRight.Angle);
2601:     if(RotationAngle != 0.0)
2602:     {
2603:         Point = TPoint(GData[i]->Shape.Rectangle.TopRight.x,
GData[i]->Shape.Rectangle.TopRight.y);
2604:         RotatePoint(&Point, &ResultantPoint, RotationAngle);
2605:         GData[i]->Shape.Rectangle.TopRight.x = Point.x;
2606:         GData[i]->Shape.Rectangle.TopRight.y = Point.y;
2607:         GData[i]->Shape.Rectangle.TopRight.Angle = 0;
2608:     }
2609:
2610:     RotationAngle =
RotationSize(GData[i]->Shape.Rectangle.BottomRight.Angle);
2611:     if(RotationAngle != 0.0)
2612:     {
2613:         Point = TPoint(GData[i]->Shape.Rectangle.BottomRight.x,
GData[i]->Shape.Rectangle.BottomRight.y);
2614:         RotatePoint(&Point, &ResultantPoint, RotationAngle);
2615:         GData[i]->Shape.Rectangle.BottomRight.x = Point.x;
2616:         GData[i]->Shape.Rectangle.BottomRight.y = Point.y;
2617:         GData[i]->Shape.Rectangle.BottomRight.Angle = 0;
2618:     }
2619:
2620:     RotationAngle =
RotationSize(GData[i]->Shape.Rectangle.BottomLeft.Angle);
2621:     if(RotationAngle != 0.0)
2622:     {
2623:         Point = TPoint(GData[i]->Shape.Rectangle.BottomLeft.x,
GData[i]->Shape.Rectangle.BottomLeft.y);
2624:         RotatePoint(&Point, &ResultantPoint, RotationAngle);
2625:         GData[i]->Shape.Rectangle.BottomLeft.x = Point.x;
2626:         GData[i]->Shape.Rectangle.BottomLeft.y = Point.y;
2627:         GData[i]->Shape.Rectangle.BottomLeft.Angle = 0;
2628:     }
2629:     break;
2630:
2631:     case DRAWCIRCULARARC:
2632:     case DRAWCIRCULARARROW:
2633:     case DRAWCIRCULARARROW2SIDE:
2634:         RotationAngle =
RotationSize(GData[i]->Shape.CircularArc.Start.Angle);
2635:         if(RotationAngle != 0.0)
2636:         {
2637:             Point = TPoint(GData[i]->Shape.CircularArc.Start.x,
GData[i]->Shape.CircularArc.Start.y);
2638:             RotatePoint(&Point, &ResultantPoint, RotationAngle);
2639:             GData[i]->Shape.CircularArc.Start.x = Point.x;
2640:             GData[i]->Shape.CircularArc.Start.y = Point.y;
2641:             GData[i]->Shape.CircularArc.Start.Angle = 0;
2642:         }
2643:
2644:         RotationAngle = RotationSize(GData[i]->Shape.CircularArc.End.Angle);
2645:         if(RotationAngle != 0.0)
2646:         {
2647:             Point = TPoint(GData[i]->Shape.CircularArc.End.x,
GData[i]->Shape.CircularArc.End.y);
2648:             RotatePoint(&Point, &ResultantPoint, RotationAngle);
2649:             GData[i]->Shape.CircularArc.End.x = Point.x;
2650:             GData[i]->Shape.CircularArc.End.y = Point.y;
2651:             GData[i]->Shape.CircularArc.End.Angle = 0;
2652:         }
2653:
2654:         RotationAngle = RotationSize(GData[i]->Shape.CircularArc.Mid.Angle);
2655:         if(RotationAngle != 0.0)
2656:         {
2657:             Point = TPoint(GData[i]->Shape.CircularArc.Mid.x,
GData[i]->Shape.CircularArc.Mid.y);
2658:             RotatePoint(&Point, &ResultantPoint, RotationAngle);
2659:             GData[i]->Shape.CircularArc.Mid.x = Point.x;
2660:             GData[i]->Shape.CircularArc.Mid.y = Point.y;
2661:             GData[i]->Shape.CircularArc.Mid.Angle = 0;
```

```
2662:         }
2663:         break;
2664:     }
2665: }
2666: }
2667: //-----
2668:
2669: void __fastcall PaintTools::AddShapeData(int Shape, bool Select)
2670: {
2671:     int PointCounter;
2672:     int ShapeCounterBak;
2673:     TPoint PreviousPoint;
2674:     GData[ShapeCounter] = new GraphData;
2675:
2676:     GData[ShapeCounter]->GroupNumber = GroupCounter;
2677:     GData[ShapeCounter]->GroupKind = Shape;
2678:
2679:     switch(Shape)
2680:     {
2681:         case DRAWDOUBLEBOND:
2682:         case DRAWTRIPLEBOND:
2683:         case DRAWLINE:
2684:         case DRAWCONTINUOUSLINE:
2685:         case DRAWARROW:
2686:         case DRAWARROW2SIDE:
2687:         case DRAWARROW2SIDESEP:
2688:         case DRAWCIRCLE:
2689:         case DRAWUPWEDGE:
2690:         case DRAWDOWNWEDGE:
2691:             GData[ShapeCounter]->ShapeKind = Shape;
2692:             GData[ShapeCounter]->Shape.Line.Start.x = StartPoint.x;
2693:             GData[ShapeCounter]->Shape.Line.Start.y = StartPoint.y;
2694:             GData[ShapeCounter]->Shape.Line.Start.Angle = 0;
2695:             GData[ShapeCounter]->Shape.Line.End.x = MovePoint.x;
2696:             GData[ShapeCounter]->Shape.Line.End.y = MovePoint.y;
2697:             GData[ShapeCounter]->Shape.Line.End.Angle = 0;
2698:             GData[ShapeCounter]->Shape.Line.LeftSideBond = false;
2699:             GData[ShapeCounter]->Shape.Line.BondNumber = 1;
2700:             if(Shape == DRAWDOUBLEBOND)
2701:                 GData[ShapeCounter]->Shape.Line.BondNumber = 2;
2702:             else if(Shape == DRAWTRIPLEBOND)
2703:                 GData[ShapeCounter]->Shape.Line.BondNumber = 3;
2704:             GData[ShapeCounter]->Pen = PenAssign(Image->Canvas->Pen);
2705:
2706:             if(Select)
2707:                 SelectShape(ShapeCounter, true);
2708:             break;
2709:
2710:         case DRAWRECTANGLE:
2711:             if(StartPoint.x > MovePoint.x)
2712:                 SWAPINTEGER(StartPoint.x, MovePoint.x);
2713:             if(StartPoint.y > MovePoint.y)
2714:                 SWAPINTEGER(StartPoint.y, MovePoint.y);
2715:
2716:             GData[ShapeCounter]->ShapeKind = DRAWRECTANGLE;
2717:             GData[ShapeCounter]->Shape.Rectangle.TopLeft.x = StartPoint.x;
2718:             GData[ShapeCounter]->Shape.Rectangle.TopLeft.y = StartPoint.y;
2719:             GData[ShapeCounter]->Shape.Rectangle.TopLeft.Angle = 0;
2720:             GData[ShapeCounter]->Shape.Rectangle.BottomRight.x = MovePoint.x;
2721:             GData[ShapeCounter]->Shape.Rectangle.BottomRight.y = MovePoint.y;
2722:             GData[ShapeCounter]->Shape.Rectangle.BottomRight.Angle = 0;
2723:             GData[ShapeCounter]->Shape.Rectangle.TopRight.x = MovePoint.x;
2724:             GData[ShapeCounter]->Shape.Rectangle.TopRight.y = StartPoint.y;
2725:             GData[ShapeCounter]->Shape.Rectangle.TopRight.Angle = 0;
2726:             GData[ShapeCounter]->Shape.Rectangle.BottomLeft.x = StartPoint.x;
2727:             GData[ShapeCounter]->Shape.Rectangle.BottomLeft.y = MovePoint.y;
2728:             GData[ShapeCounter]->Shape.Rectangle.BottomLeft.Angle = 0;
2729:             GData[ShapeCounter]->Pen = PenAssign(Image->Canvas->Pen);
2730:             GData[ShapeCounter]->Brush = BrushAssign(Image->Canvas->Brush);
2731:
2732:             if(Select)
2733:                 SelectShape(ShapeCounter, true);
2734:             break;
2735:
2736:         case DRAWCIRCULARARC:
```

```
2737:         case DRAWCIRCULARARROW:
2738:         case DRAWCIRCULARARROW2SIDE:
2739:             GData[ShapeCounter]->ShapeKind = Shape;
2740:             GData[ShapeCounter]->Shape.CircularArc.Start.x = StartPoint.x;
2741:             GData[ShapeCounter]->Shape.CircularArc.Start.y = StartPoint.y;
2742:             GData[ShapeCounter]->Shape.CircularArc.Start.Angle = 0;
2743:             GData[ShapeCounter]->Shape.CircularArc.End.x = MovePoint.x;
2744:             GData[ShapeCounter]->Shape.CircularArc.End.y = MovePoint.y;
2745:             GData[ShapeCounter]->Shape.CircularArc.End.Angle = 0;
2746:             GData[ShapeCounter]->Shape.CircularArc.Mid.x = (MovePoint.y - StartPoint
+ MovePoint.x + StartPoint.x) / 2;
2747:             GData[ShapeCounter]->Shape.CircularArc.Mid.y = (StartPoint.x - MovePoint
+ MovePoint.y + StartPoint.y) / 2;
2748:             GData[ShapeCounter]->Shape.CircularArc.Mid.Angle = 0;
2749:             GData[ShapeCounter]->Pen = PenAssign(Image->Canvas->Pen);
2750:
2751:             if(Select)
2752:                 SelectShape(ShapeCounter, true);
2753:             break;
2754:
2755:         case DRAWCHAIN:
2756:             PointCounter = DetermineChainArrayPoints(&StartPoint, &MovePoint);
2757:             PreviousPoint = StartPoint;
2758:
2759:             for(register i = 0; i < PointCounter; i++)
2760:             {
2761:                 if(i)
2762:                 {
2763:                     GData[ShapeCounter] = new GraphData;
2764:                     GData[ShapeCounter]->GroupNumber = GroupCounter;
2765:                     GData[ShapeCounter]->GroupKind = Shape;
2766:                 }
2767:
2768:                 GData[ShapeCounter]->ShapeKind = DRAWCHAIN; //DRAWCONTINUOUSLINE;
2769:                 GData[ShapeCounter]->Shape.Line.Start.x = PreviousPoint.x;
2770:                 GData[ShapeCounter]->Shape.Line.Start.y = PreviousPoint.y;
2771:                 GData[ShapeCounter]->Shape.Line.Start.Angle = 0;
2772:                 GData[ShapeCounter]->Shape.Line.End.x = ArrayPoints[i].x;
2773:                 GData[ShapeCounter]->Shape.Line.End.y = ArrayPoints[i].y;
2774:                 GData[ShapeCounter]->Shape.Line.End.Angle = 0;
2775:                 GData[ShapeCounter]->Shape.Line.LeftSideBond = false;
2776:                 GData[ShapeCounter]->Shape.Line.BondNumber = 1;
2777:                 GData[ShapeCounter]->Pen = PenAssign(Image->Canvas->Pen);
2778:
2779:                 if(Select)
2780:                     SelectShape(ShapeCounter, true);
2781:
2782:                 PreviousPoint = ArrayPoints[i];
2783:
2784:                 ShapeCounter++;
2785:             }
2786:
2787:             if(!PointCounter)
2788:                 delete GData[ShapeCounter];
2789:
2790:             ShapeCounter--;
2791:             break;
2792:
2793:         case DRAWPOLYGON:
2794:         case DRAWCYCLOPENTANE:
2795:             PointCounter = DeterminePolygonArrayPoints(&StartPoint, &MovePoint,
Shape);
2796:             PreviousPoint = ArrayPoints[PointCounter - 1];
2797:
2798:             ShapeCounterBak = ShapeCounter;
2799:
2800:             for(register i = 0; i < PointCounter; i++)
2801:             {
2802:                 if(i)
2803:                 {
2804:                     GData[ShapeCounter] = new GraphData;
2805:                     GData[ShapeCounter]->GroupNumber = GroupCounter;
2806:                     GData[ShapeCounter]->GroupKind = Shape;
2807:                 }
2808:
```

```
2809:          GData[ShapeCounter]->ShapeKind =
Shape; //DRAWCHAIN; //DRAWCONTINUOUSLINE;
2810:          GData[ShapeCounter]->Shape.Line.Start.x = PreviousPoint.x;
2811:          GData[ShapeCounter]->Shape.Line.Start.y = PreviousPoint.y;
2812:          GData[ShapeCounter]->Shape.Line.Start.Angle = 0;
2813:          GData[ShapeCounter]->Shape.Line.End.x = ArrayPoints[i].x;
2814:          GData[ShapeCounter]->Shape.Line.End.y = ArrayPoints[i].y;
2815:          GData[ShapeCounter]->Shape.Line.End.Angle = 0;
2816:          GData[ShapeCounter]->Shape.Line.LeftSideBond = false;
2817:          GData[ShapeCounter]->Shape.Line.BondNumber = 1;
2818:          GData[ShapeCounter]->Pen = PenAssign(Image->Canvas->Pen);
2819:
2820:          PreviousPoint = ArrayPoints[i];
2821:
2822:          ShapeCounter++;
2823:      }
2824:
2825:      if(Select)
2826:          for(register i = 0; i < PointCounter; i++)
2827:              SelectShape(ShapeCounterBak + i, true);
2828:
2829:      ShapeCounter--;
2830:      break;
2831:  }
2832:
2833:      ShapeCounter++;
2834:  }
2835:  //-----
2836:
```

```
1: //-----
2:
3: #include <vcl.h>
4: #pragma hdrstop
5:
6: #include "Warning2.h"
7: //-----
8: #pragma package(smart_init)
9: #pragma resource "*.dfm"
10: TWarning *Warning;
11: //-----
12: __fastcall TWarning::TWarning(TComponent* Owner)
13:     : TForm(Owner)
14: {
15:     PushedButtonNumber = 0;
16: }
17: //-----
18: void __fastcall TWarning::Button1Click(TObject *Sender)
19: {
20:     PushedButtonNumber = 1;
21:     Close();
22: }
23: //-----
24: void __fastcall TWarning::Button2Click(TObject *Sender)
25: {
26:     PushedButtonNumber = 2;
27:     Close();
28: }
29: //-----
30: void __fastcall TWarning::Button3Click(TObject *Sender)
31: {
32:     PushedButtonNumber = 3;
33:     Close();
34: }
35: //-----
36:
37: int __fastcall TWarning::ShowWarning(AnsiString FormCaption, AnsiString String, AnsiString
    B1Caption, AnsiString B2Caption, AnsiString B3Caption, int DefaultButton)
38: {
39:     Caption = FormCaption;
40:     Label1->Caption = String;
41:
42:     if(!B1Caption.IsEmpty())
43:         Button1->Caption = B1Caption;
44:
45:     if(!B2Caption.IsEmpty())
46:     {
47:         Button2->Caption = B2Caption;
48:         Button2->Visible = true;
49:     }
50:     else
51:         Button2->Visible = false;
52:
53:     if(!B3Caption.IsEmpty())
54:     {
55:         Button3->Caption = B3Caption;
56:         Button3->Visible = true;
57:     }
58:     else
59:         Button3->Visible = false;
60:
61:     switch(DefaultButton)
62:     {
63:     case 1:
64:         Button1->TabOrder = 1;
65:         Button2->TabOrder = 2;
66:         Button3->TabOrder = 3;
67:         break;
68:
69:     case 2:
70:         Button1->TabOrder = 3;
71:         Button2->TabOrder = 1;
72:         Button3->TabOrder = 2;
73:         break;
74:     }
```

```
75:         case 3:
76:             Button1->TabOrder = 2;
77:             Button2->TabOrder = 3;
78:             Button3->TabOrder = 1;
79:             break;
80:     }
81:
82:     ShowModal();
83:     return PushedButtonNumber;
84: }
85: //-----
86:
87: void __fastcall TWarning::FormKeyPress(TObject *Sender, char &Key)
88: {
89:     if(Key == VK_ESCAPE)
90:         Close();
91: }
92: //-----
93:
94: void __fastcall TWarning::FormShow(TObject *Sender)
95: {
96:     PushedButtonNumber = 0;
97: }
98: //-----
99:
```

```
1:  //-----
2:
3:  #include <vcl.h>
4:  #pragma hdrstop
5:
6:  #include "AddEditItem.h"
7:  //-----
8:  #pragma package(smart_init)
9:  #pragma resource "*.dfm"
10: TAddEditItemForm *AddEditItemForm;
11: //-----
12: __fastcall TAddEditItemForm::TAddEditItemForm(TComponent* Owner)
13:     : TForm(Owner)
14: {
15:     PushedButtonNumber = 0;
16: }
17: //-----
18: void __fastcall TAddEditItemForm::Button2Click(TObject *Sender)
19: {
20:     Close();
21:     PushedButtonNumber = 0;
22: }
23: //-----
24:
25: void __fastcall TAddEditItemForm::Button1Click(TObject *Sender)
26: {
27:     Close();
28:     PushedButtonNumber = 1;
29: }
30: //-----
31:
32: AnsiString __fastcall TAddEditItemForm::ShowAddEditItem(AnsiString FormCaption, AnsiString
    OldValue, bool AddTarjomeh)
33: {
34:     TStringList *List = new TStringList;
35:
36:     List->Add(OldValue);
37:     LabeledEdit1->Text = List->Names[0];
38:     LabeledEdit2->Text = (AddTarjomeh == true ? String(List->Values[0]) : String(""));
39:
40:     Caption = FormCaption;
41:
42:     LabeledEdit2->Visible = AddTarjomeh;
43:
44:     ShowModal();
45:
46:     if(PushedButtonNumber)
47:     if(AddTarjomeh)
48:         OldValue = LabeledEdit1->Text + "=" + LabeledEdit2->Text;
49:     else
50:         OldValue = LabeledEdit1->Text;
51:
52:     delete List;
53:     return OldValue;
54: }
55: //-----
56: void __fastcall TAddEditItemForm::FormKeyPress(TObject *Sender, char &Key)
57: {
58:     if(Key == VK_ESCAPE)
59:         Close();
60: }
61: //-----
62:
63:
```

```
1: //-----
2:
3:
4: #pragma hdrstop
5:
6: #include "Functions.h"
7: #include "Warning2.h"
8:
9: //-----
10:
11: #pragma package(smart_init)
12:
13:
14: void __fastcall ClearComponentsValue(TWinControl *Parent)
15: {
16:     TControl *ChildControl;
17:
18:     for(register i = 0; i < Parent->ControlCount; i++)
19:     {
20:         ChildControl = Parent->Controls[i];
21:
22:         if(!ChildControl)
23:             continue;
24:
25:         if(ChildControl->InheritsFrom(__classid(TEdit)))
26:             static_cast<TEdit *>(ChildControl)->Text = "";
27:         else if(ChildControl->InheritsFrom(__classid(TMemo)))
28:             static_cast<TMemo *>(ChildControl)->Clear();
29:         else if(ChildControl->InheritsFrom(__classid(TLabeledEdit)))
30:             static_cast<TLabeledEdit *>(ChildControl)->Text = "";
31:         else if(ChildControl->InheritsFrom(__classid(TListBox)))
32:             static_cast<TListBox *>(ChildControl)->Clear();
33:
34:         if(ChildControl->ControlStyle.Contains(csAcceptsControls))
35:         {
36:             TWinControl *WinControl = dynamic_cast<TWinControl*>(ChildControl);
37:             if(WinControl)
38:                 ClearComponentsValue(WinControl);
39:         }
40:     }
41: }
42: //-----
43:
44: void __fastcall ClearImage(TComponent *Sender)
45: {
46:     ((TImage *)Sender)->Picture->Bitmap->FreeImage();
47:     ((TImage *)Sender)->Picture = NULL;
48: }
49: //-----
50:
51: float __fastcall StringToFloat(AnsiString Str)
52: {
53:     float ReturnValue = 0;
54:     if(!Str.IsEmpty())
55:         ReturnValue = StrToFloat(Str);
56:
57:     return ReturnValue;
58: }
59: //-----
60:
61: AnsiString __fastcall FloatToString(float Number)
62: {
63:     AnsiString ReturnValue = "";
64:
65:     if(Number != 0)
66:     {
67:         TFloatRec FRec;
68:
69:         FloatToStrF(Number, ffExponent, 15, 2);
70:         ReturnValue = Number;
71:     }
72:     return ReturnValue;
73: }
74: //-----
75:
```

```

76: char * __fastcall AnsiStrPosIC(char * Str, char * SubStr)
77: {
78:     char *Str2 = new char[StrLen(Str) + 1], *SubStr2 = new char[StrLen(SubStr) + 1];
79:     char *Pos;
80:     char *ReturnValue = NULL;
81:
82:     StrCopy(Str2, Str);
83:     StrCopy(SubStr2, SubStr);
84:
85:     StrUpper(Str2);
86:     StrUpper(SubStr2);
87:
88:     Pos = AnsiStrPos(Str2, SubStr2);
89:
90:     if(Pos)
91:         ReturnValue = Str + int(Pos) - int(Str2);
92:
93:     delete[] SubStr2;
94:     delete[] Str2;
95:
96:     return ReturnValue;
97: }
98: //-----
99:
100: int __fastcall SearchStringList(AnsiString List, AnsiString SubStr, bool WholeWord)
101: {
102:     TStringList *StrList = new TStringList;
103:     int ReturnValue;
104:
105:     StrList->Text = List;
106:     ReturnValue = SearchStringList(StrList, SubStr, WholeWord);
107:     delete StrList;
108:
109:     return ReturnValue;
110: }
111: //-----
112:
113: int __fastcall SearchStringList(TStringList *List, AnsiString SubStr, bool WholeWord)
114: {
115:     int ReturnValue = -1;
116:
117:     for(register i = List->Count - 1; i >= 0; i--)
118:     {
119:         if(WholeWord)
120:         {
121:             if(!List->Strings[i].AnsiCompareIC(SubStr))
122:             {
123:                 ReturnValue = i;
124:                 break;
125:             }
126:         }
127:         else
128:         {
129:             if(AnsiStrPosIC(List->Strings[i].c_str(), SubStr.c_str()))
130:             {
131:                 ReturnValue = i;
132:                 break;
133:             }
134:         }
135:     }
136:     Screen->Cursor = crHourGlass;
137:
138:     return ReturnValue;
139: }
140: //-----
141:
142: bool __fastcall IsInNumericRange(AnsiString LowValueStr, AnsiString HiValueStr, float
    CurrentValue, float CurrentValuePM)
143: {
144:     bool ReturnValue = false;
145:     float Low, High;
146:     float Swap;
147:
148:     Low = StringToFloat(LowValueStr);
149:     High = StringToFloat(HiValueStr);

```

```
150:
151:     if(High < Low && HiValueStr.IsEmpty() && LowValueStr.IsEmpty())
152:     {
153:         Swap = High;
154:         High = Low;
155:         Low = Swap;
156:     }
157:
158:     if((CurrentValue - CurrentValuePM <= High || HiValueStr.IsEmpty()) &&
159:        (CurrentValue + CurrentValuePM >= Low || LowValueStr.IsEmpty()))
160:         ReturnValue = true;
161:
162:     return ReturnValue;
163: }
164: //-----
165:
166: void __fastcall FloatEditorsKeyPress(TObject *Sender, char &Key)
167: {
168:     if((Key < 48 || Key > 57) && Key != VK_RETURN && Key != VK_BACK && Key != '.' && Key
169:        != '+' && Key != '-' && Key != 'E')
170:         Key = NULL;
171: }
172: //-----
173: bool __fastcall CheckIfOverWriteAccepted(char *FileName)
174: {
175:     bool ReturnValue = true;
176:
177:     if(FileExists(FileName))
178:     {
179:         if(Warning->ShowWarning("اخطار", "آیا بازنویسی است. آیا از قبل موجود است. اگر نه،  

180:            "بله", "خیر", "گرد؟") != 2)
181:             ReturnValue = false;
182:     }
183:     return ReturnValue;
184: }
185: //-----
186:
```

```
1: //-----
2:
3: #include <vcl.h>
4: #pragma hdrstop
5:
6: #include "SearchForm.h"
7: #include <CompoundStruct.h>
8: #include <Main.h>
9: #include <Functions.h>
10: //-----
11: #pragma package(smart_init)
12: #pragma resource "*.dfm"
13: TSearch *Search;
14: //-----
15: __fastcall TSearch::TSearch(TComponent* Owner)
16:     : TForm(Owner)
17: {
18: }
19: //-----
20:
21: void __fastcall TSearch::Button2Click(TObject *Sender)
22: {
23:     Close();
24: }
25: //-----
26:
27: void __fastcall TSearch::Button1Click(TObject *Sender)
28: {
29:     int SearchRecord;
30:
31:     Screen->Cursor = crHourGlass;
32:     SearchRecord = FindNextActiveRecord(true);
33:     while(SearchRecord != -1)
34:     {
35:         AddAsResult(SearchRecord, IsRequestedResult());
36:
37:         SearchRecord = FindNextActiveRecord();
38:     }
39:     Screen->Cursor = crDefault;
40: }
41: //-----
42:
43: int __fastcall TSearch::FindNextActiveRecord(bool FindFirstActiveRecord)
44: {
45:     int ReturnValue = -1;
46:     void *Result;
47:     CompoundData *Data = &MainForm->Data;
48:     static int LastResult = -1;
49:
50:     if(FindFirstActiveRecord)
51:     {
52:         LastResult = -1;
53:
54:         if(RadioButton3->Checked)
55:         {
56:             MainForm->DeleteAllListItems();
57:             Data->DataFile->Go(TOP);
58:             Result = Data->DataFile->FreeSearch(3, NULL, NULL, NULL, NULL);
59:             if(Result)
60:             {
61:                 LastResult = ReturnValue = Data->DataFile->CurrentNodeAddress;
62:                 Data->Load(ReturnValue);
63:             }
64:         }
65:         else if(RadioButton4->Checked)
66:         {
67:             /*TItemStates NextItemKind = TItemStates() << isSelected;
68:             TListItem *Item = ListView1->Selected;
69:
70:             while(Item)
71:             {
72:                 LastResult = ReturnValue = Item->Index;
73:                 Data->Load(MainForm->GetItemNodeAddress(Item));
74:
75:                 Item = ListView1->GetNextItem(Item, sdAll, NextItemKind);
```

```
76:         } */
77:
78:         for(register i = MainForm->ListView1->Items->Count - 1; i >= 0; i--)
79:             if(MainForm->ListView1->Items->Item[i]->Selected)
80:             {
81:                 LastResult = ReturnValue = i;
82:
83:                 Data->Load(MainForm->GetItemNodeAddress(MainForm->ListView1->Items->Item[i]));
84:                 if(CheckBox1->Checked)
85:                     MainForm->ListView1->Items->Item[i]->Selected = false;
86:                 break;
87:             }
88:         else if(RadioButton5->Checked)
89:         {
90:             for(register i = MainForm->ListView1->Items->Count - 1; i >= 0; i--)
91:                 if(!MainForm->ListView1->Items->Item[i]->Selected)
92:                 {
93:                     LastResult = ReturnValue = i;
94:
95:                     Data->Load(MainForm->GetItemNodeAddress(MainForm->ListView1->Items->Item[i]));
96:                     break;
97:                 }
98:             else
99:                 if(CheckBox1->Checked)
100:                     MainForm->ListView1->Items->Item[i]->Selected = false;
101:         }
102:         else if(RadioButton6->Checked)
103:         {
104:             if(MainForm->ListView1->Items->Count)
105:             {
106:                 LastResult = ReturnValue = MainForm->ListView1->Items->Count - 1;
107:                 Data->Load(MainForm->GetItemNodeAddress(ReturnValue));
108:                 if(CheckBox1->Checked)
109:                     MainForm->ListView1->ClearSelection();
110:             }
111:         }
112:         else
113:         {
114:             if(RadioButton3->Checked)
115:             {
116:                 Result = Data->DataFile->FreeSearch(3, NULL, NULL, NULL, NULL, NEXT);
117:                 if(Result)
118:                 {
119:                     LastResult = ReturnValue = Data->DataFile->CurrentNodeAddress;
120:                     Data->Load(ReturnValue);
121:                 }
122:             }
123:             else if(RadioButton4->Checked)
124:             {
125:                 for(register i = LastResult - 1; i >= 0; i--)
126:                     if(MainForm->ListView1->Items->Item[i]->Selected)
127:                     {
128:                         LastResult = ReturnValue = i;
129:
130:                         Data->Load(MainForm->GetItemNodeAddress(MainForm->ListView1->Items->Item[i]));
131:                         if(CheckBox1->Checked)
132:                             MainForm->ListView1->Items->Item[i]->Selected = false;
133:                         break;
134:                     }
135:             }
136:             else if(RadioButton5->Checked)
137:             {
138:                 for(register i = LastResult - 1; i >= 0; i--)
139:                     if(!MainForm->ListView1->Items->Item[i]->Selected)
140:                     {
141:                         LastResult = ReturnValue = i;
142:
143:                         Data->Load(MainForm->GetItemNodeAddress(MainForm->ListView1->Items->Item[i]));
144:                         break;
145:                     }
146:             }
147:             else
148:                 if(CheckBox1->Checked)
149:                     MainForm->ListView1->Items->Item[i]->Selected = false;
```

```
147:         }
148:     else if(RadioButton6->Checked)
149:     {
150:         if(LastResult > 0)
151:         {
152:             ReturnValue = --LastResult;
153:             Data->Load(MainForm->GetItemNodeAddress(ReturnValue));
154:         }
155:     }
156: }
157:
158: return ReturnValue;
159: }
160: //-----
161:
162: void __fastcall TSearch::CheckForEnabledComponents(void)
163: {
164:     if(RadioButton3->Checked)
165:     {
166:         CheckBox1->Enabled = false;
167:         CheckBox1->Checked = true;
168:         RadioButton1->Enabled = false;
169:         RadioButton7->Enabled = false;
170:         RadioButton2->Checked = true;
171:     }
172:     else
173:     {
174:         CheckBox1->Enabled = true;
175:         RadioButton1->Enabled = true;
176:         RadioButton7->Enabled = true;
177:
178:         if(RadioButton4->Checked && !CheckBox1->Checked)
179:         {
180:             RadioButton1->Enabled = false;
181:             RadioButton7->Checked = true;
182:         }
183:         else
184:         {
185:             RadioButton1->Enabled = true;
186:         }
187:
188:         if(RadioButton5->Checked)
189:         {
190:             RadioButton7->Enabled = false;
191:             RadioButton1->Checked = true;
192:         }
193:         else
194:         {
195:             RadioButton7->Enabled = true;
196:         }
197:     }
198:
199:
200:     if(LabeledEdit1->Text.IsEmpty() &&
201:        LabeledEdit2->Text.IsEmpty() &&
202:        LabeledEdit3->Text.IsEmpty() &&
203:        LabeledEdit4->Text.IsEmpty() &&
204:        LabeledEdit5->Text.IsEmpty() &&
205:        LabeledEdit6->Text.IsEmpty() &&
206:        LabeledEdit7->Text.IsEmpty() &&
207:        LabeledEdit8->Text.IsEmpty() &&
208:        LabeledEdit9->Text.IsEmpty() &&
209:        LabeledEdit10->Text.IsEmpty() &&
210:        LabeledEdit11->Text.IsEmpty() &&
211:        LabeledEdit12->Text.IsEmpty() &&
212:        LabeledEdit13->Text.IsEmpty() &&
213:        LabeledEdit14->Text.IsEmpty() &&
214:        LabeledEdit15->Text.IsEmpty() &&
215:        LabeledEdit16->Text.IsEmpty() &&
216:        LabeledEdit17->Text.IsEmpty() &&
217:        LabeledEdit18->Text.IsEmpty())
218:     {
219:         Button1->Enabled = false;
220:     }
221:     else
```

```
222:         Button1->Enabled = true;
223:     }
224: //-----
225: void __fastcall TSearch::RadioButton3Click(TObject *Sender)
226: {
227:     CheckForEnabledComponents();
228: }
229: //-----
230:
231: void __fastcall TSearch::FormShow(TObject *Sender)
232: {
233:     CheckForEnabledComponents();
234: }
235: //-----
236:
237: bool __fastcall TSearch::IsRequestedResult(void)
238: {
239:     bool ReturnValue = true;
240:     CompoundData *Data = &MainForm->Data;
241:
242:     if(!LabeledEdit1->Text.IsEmpty())
243:     {
244:         if(SearchStringList(Data->MainFields.IUPACName, LabeledEdit1->Text,
245:             CheckBox2->Checked) == -1)
246:             ReturnValue = false;
247:     }
248:     if(ReturnValue && !LabeledEdit2->Text.IsEmpty())
249:     {
250:         if(SearchStringList(Data->TrivialNames, LabeledEdit2->Text, CheckBox2->Checked
251:             == -1)
252:             ReturnValue = false;
253:     }
254:     if(ReturnValue && !LabeledEdit3->Text.IsEmpty())
255:     {
256:         if(SearchStringList(Data->TradeNames, LabeledEdit3->Text, CheckBox2->Checked)
257:             == -1)
258:             ReturnValue = false;
259:     }
260:     if(ReturnValue && !LabeledEdit4->Text.IsEmpty())
261:     {
262:         if(SearchStringList(Data->MainFields.MolecularFormula, LabeledEdit4->Text,
263:             CheckBox2->Checked) == -1)
264:             ReturnValue = false;
265:     }
266:     if(ReturnValue && !LabeledEdit5->Text.IsEmpty())
267:     {
268:         if(SearchStringList(Data->MainFields.ChemicalRegistryNumber, LabeledEdit5->Text,
269:             CheckBox2->Checked) == -1)
270:             ReturnValue = false;
271:     }
272:     if(ReturnValue && !LabeledEdit6->Text.IsEmpty())
273:     {
274:         if(SearchStringList(Data->PhysicochemicalProperties.DOTNumber,
275:             LabeledEdit6->Text, CheckBox2->Checked) == -1)
276:             ReturnValue = false;
277:     }
278:     if(ReturnValue && !LabeledEdit7->Text.IsEmpty())
279:     {
280:         if(SearchStringList(Data->PhysicochemicalProperties.PhysicalState,
281:             LabeledEdit7->Text, CheckBox2->Checked) == -1)
282:             ReturnValue = false;
283:     }
284:     if(ReturnValue && !LabeledEdit8->Text.IsEmpty())
285:     {
286:         if(SearchStringList(Data->SymmetricProperties.PointGroup, LabeledEdit8->Text,
287:             CheckBox2->Checked) == -1)
288:             ReturnValue = false;
289:     }
290: }
```

```
289:
290:     if(ReturnValue && (!LabeledEdit9->Text.IsEmpty() || !LabeledEdit10->Text.IsEmpty()))
291:     {
292:         if(!IsInNumericRange(LabeledEdit9->Text, LabeledEdit10->Text,
Data->MainFields.FixPhysicochemicalProperties.BoilingPoint,
Data->MainFields.FixPhysicochemicalProperties.BoilingPointPM))
293:             ReturnValue = false;
294:     }
295:
296:     if(ReturnValue && (!LabeledEdit11->Text.IsEmpty() || !LabeledEdit12->Text.IsEmpty()))
297:     {
298:         if(!IsInNumericRange(LabeledEdit11->Text, LabeledEdit12->Text,
Data->MainFields.FixPhysicochemicalProperties.Density,
Data->MainFields.FixPhysicochemicalProperties.DensityPM))
299:             ReturnValue = false;
300:     }
301:
302:     if(ReturnValue && (!LabeledEdit13->Text.IsEmpty() || !LabeledEdit14->Text.IsEmpty()))
303:     {
304:         if(!IsInNumericRange(LabeledEdit13->Text, LabeledEdit14->Text,
Data->MainFields.FixPhysicochemicalProperties.FlashPoint,
Data->MainFields.FixPhysicochemicalProperties.FlashPointPM))
305:             ReturnValue = false;
306:     }
307:
308:     if(ReturnValue && (!LabeledEdit15->Text.IsEmpty() || !LabeledEdit16->Text.IsEmpty()))
309:     {
310:         if(!IsInNumericRange(LabeledEdit15->Text, LabeledEdit16->Text,
Data->MainFields.FixPhysicochemicalProperties.MeltingPoint,
Data->MainFields.FixPhysicochemicalProperties.MeltingPointPM))
311:             ReturnValue = false;
312:     }
313:
314:     if(ReturnValue && (!LabeledEdit17->Text.IsEmpty() || !LabeledEdit18->Text.IsEmpty()))
315:     {
316:         if(!IsInNumericRange(LabeledEdit17->Text, LabeledEdit18->Text,
Data->MainFields.FixPhysicochemicalProperties.VaporDensity,
Data->MainFields.FixPhysicochemicalProperties.VaporDensityPM))
317:             ReturnValue = false;
318:     }
319:
320:     return ReturnValue;
321: }
322: //-----
323: void __fastcall TSearch::LabeledEdit1Change(TObject *Sender)
324: {
325:     CheckForEnabledComponents();
326: }
327: //-----
328:
329: void __fastcall TSearch::AddAsResult(int ListItem, bool IsResult)
330: {
331:     if(RadioButton3->Checked)
332:     {
333:         if(IsResult)
334:             MainForm->AddListItems();
335:     }
336:     else if(RadioButton4->Checked)
337:     {
338:         if(!IsResult)
339:         {
340:             if(RadioButton1->Checked)
341:                 MainForm->ListView1->Items->Item[ListItem]->Selected = false;
342:             else if(RadioButton2->Checked)
343:                 MainForm->DeleteListItem(ListItem);
344:         }
345:         else
346:         {
347:             if(RadioButton1->Checked)
348:                 MainForm->ListView1->Items->Item[ListItem]->Selected = true;
349:             else if(RadioButton7->Checked)
350:                 MainForm->ListView1->Items->Item[ListItem]->Selected = false;
351:         }
352:     }
353:     else if(RadioButton5->Checked)
```

```
354:     {
355:         if(IsResult)
356:         {
357:             if(RadioButton1->Checked)
358:                 MainForm->ListView1->Items->Item[ListItem]->Selected = true;
359:         }
360:         else if(RadioButton2->Checked)
361:             MainForm->DeleteListItem(ListItem);
362:     }
363:     else if(RadioButton6->Checked)
364:     {
365:         if(IsResult)
366:         {
367:             if(RadioButton1->Checked)
368:                 MainForm->ListView1->Items->Item[ListItem]->Selected = true;
369:             else if(RadioButton7->Checked)
370:                 MainForm->ListView1->Items->Item[ListItem]->Selected = false;
371:         }
372:         else
373:             if(RadioButton2->Checked)
374:                 MainForm->DeleteListItem(ListItem);
375:     }
376: }
377: //-----
378:
379: void __fastcall TSearch::ClearPreviousResults(void)
380: {
381:     if(RadioButton3->Checked)
382:     {
383:         MainForm->DeleteAllListItems();
384:     }
385:     else if(RadioButton4->Checked || RadioButton5->Checked || RadioButton6->Checked)
386:     {
387:         MainForm->ListView1->ClearSelection();
388:     }
389: }
390: //-----
391: void __fastcall TSearch::FormKeyPress(TObject *Sender, char &Key)
392: {
393:     if(Key == VK_ESCAPE)
394:         Close();
395: }
396: //-----
397:
398: void __fastcall TSearch::LabeledEdit9KeyPress(TObject *Sender, char &Key)
399: {
400:     FloatEditorsKeyPress(Sender, Key);
401: }
402: //-----
403:
```

```
1: //-----
2:
3:
4: #pragma hdrstop
5:
6: #include "ISO2709.h"
7:
8: //-----
9:
10: __fastcall ISO2709::ISO2709(void)
11: {
12:     Directory = new TMemoryStream;
13:     FieldsData = new TMemoryStream;
14:
15:     DataFileName = "";
16:     File = NULL;
17:
18:     Leader.RecordStatus = '0';
19:     memset(Leader.ImplementationCodes, '0', 4);
20:     Leader.IndicatorLength = '0';
21:     Leader.SubfieldIdentifierLength = '0';
22:     memset(Leader.ForUserSystems, '0', 3);
23:     memset(Leader.Reserved, '0', 2);
24:     Leader.LengthOfEntryLength = LENGTHOFENTRY + '0';
25:     Leader.StartingCharacterPositionLength = LENGTHOFSTARTINGCHARACTERPOSITION + '0';
26: }
27: //-----
28:
29: __fastcall ISO2709::~~ISO2709(void)
30: {
31:     if(FieldsData)
32:         delete FieldsData;
33:
34:     if(Directory)
35:         delete Directory;
36:
37:     CloseDatabase();
38: }
39: //-----
40:
41: bool __fastcall ISO2709::OpenDatabase(void)
42: {
43:     bool ReturnValue = false;
44:
45:     if(!DataFileName.IsEmpty())
46:     {
47:         File = new TFileStream(DataFileName, fmCreate);
48:         ReturnValue = true;
49:     }
50:
51:     return ReturnValue;
52: }
53: //-----
54:
55: void __fastcall ISO2709::CloseDatabase(void)
56: {
57:     if(File)
58:         delete File;
59:     File = NULL;
60:     DataFileName = "";
61: }
62: //-----
63:
64: void __fastcall ISO2709::ResetRecord(void)
65: {
66:     Directory->Size = 0;
67:     FieldsData->Size = 0;
68: }
69: //-----
70:
71: void __fastcall ISO2709::AddField(int Tag, AnsiString FieldString)
72: {
73:     if(!FieldString.IsEmpty())
74:     {
75:         AnsiString TagStr = IntToStr(Tag);
```

```
76:      AnsiString FieldLength = IntToStr(FieldString.Length() + 1);
77:      AnsiString FieldPosition = IntToStr(FieldsData->Position);
78:      ISO2709Directory DirectoryBuffer;
79:
80:      if(TagStr.Length() < 3)
81:      {
82:          do
83:          {
84:              TagStr.Insert("0", 1);
85:          }while(TagStr.Length() < 3);
86:      }
87:      else
88:          TagStr.SetLength(3);
89:
90:      if(FieldLength.Length() < LENGTHOFENTRY)
91:      {
92:          do
93:          {
94:              FieldLength.Insert("0", 1);
95:          }while(FieldLength.Length() < LENGTHOFENTRY);
96:      }
97:      else
98:          FieldLength.SetLength(LENGTHOFENTRY);
99:
100:     if(FieldPosition.Length() < LENGTHOFSTARTINGCHARACTERPOSITION)
101:     {
102:         do
103:         {
104:             FieldPosition.Insert("0", 1);
105:         }while(FieldPosition.Length() < LENGTHOFSTARTINGCHARACTERPOSITION);
106:     }
107:     else
108:         FieldPosition.SetLength(LENGTHOFSTARTINGCHARACTERPOSITION);
109:
110:     StrCopy(DirectoryBuffer.Tag, TagStr.c_str());
111:     StrCopy(DirectoryBuffer.FieldLength, FieldLength.c_str());
112:     StrCopy(DirectoryBuffer.FieldLocation, FieldPosition.c_str());
113:
114:     Directory->Write(&DirectoryBuffer, sizeof(ISO2709Directory));
115:
116:     FieldString += FIELDSEPERATOR;
117:     FieldsData->Write(FieldString.c_str(), FieldString.Length());
118: }
119: }
120: //-----
121:
122: void __fastcall ISO2709::AddRecord(void)
123: {
124:     if(Directory->Size)
125:     {
126:         Directory->Write(FIELDSEPERATOR, 1);
127:         FieldsData->Write(RECORDSEPERATOR, 1);
128:
129:         int RecordLength = sizeof(ISO2709Leader) + Directory->Size + FieldsData->Size;
130:         AnsiString LengthStr = IntToStr(RecordLength);
131:         AnsiString ControlStr = IntToStr(sizeof(ISO2709Leader) + Directory->Size);
132:         char *Buffer;
133:         int Size;
134:
135:         if(LengthStr.Length() < 5)
136:         {
137:             do
138:             {
139:                 LengthStr.Insert("0", 1);
140:             }while(LengthStr.Length() < 5);
141:         }
142:         else
143:             LengthStr.SetLength(5);
144:
145:         if(ControlStr.Length() < 5)
146:         {
147:             do
148:             {
149:                 ControlStr.Insert("0", 1);
150:             }while(ControlStr.Length() < 5);
```

```
151:         }
152:     else
153:         ControlStr.SetLength(5);
154:
155:     memmove(Leader.RecordLength, LengthStr.c_str(), 5);
156:     memmove(Leader.BaseAddressOfData, ControlStr.c_str(), 5);
157:
158:     Buffer = new char[RecordLength + 1];
159:
160:     Directory->Position = 0;
161:     FieldsData->Position = 0;
162:
163:     memmove(Buffer, &Leader, sizeof(ISO2709Leader));
164:     Directory->Read(&Buffer[sizeof(ISO2709Leader)], Directory->Size);
165:     FieldsData->Read(&Buffer[sizeof(ISO2709Leader) + (int)Directory->Size],
FieldsData->Size);
166:
167:     Size = File->Size;
168:     for(int i = 0; i < RecordLength; ++i)
169:     {
170:         if(!(File->Size%80))
171:             File->Write("\r\n", 2);
172:         File->Write(Buffer + i, 1);
173:     }
174:     delete[] Buffer;
175: }
176: }
177: //-----
178:
179: #pragma package(smart_init)
```

```
1: #pragma hdrstop
2:
3: #include <Main.h>
4: #include <CompoundStruct.h>
5: #include <PaintTools.h>
6: #include "Jpeg.hpp"
7: #include "Fbtree.h"
8:
9: #include <Warning2.h>
10: #include <Functions.h>
11:
12: #pragma package(smart_init)
13:
14: __fastcall CompoundData::CompoundData(void)
15: {
16:     StructuralFormulaPic.PictureBuffer = NULL;
17:     StructuralFormula = NULL;
18:     SpectralProperties = NULL;
19:     CAnalysis = NULL;
20:
21:     MainFields.PaintToolShapeCount = 0;
22:     DataFile = NULL;
23:     DataFileName = "";
24:     *(int *)EndOfBuffer = -1;
25:     CurrentNodeAddress = 0;
26:
27:     StrCopy(SpectralBranchName, "Spectral Property Branch");
28:     StrCopy(ChoromatographyBranchName, "Choromatography Property Branch");
29: }
30: //-----
31:
32: __fastcall CompoundData::~CompoundData(void)
33: {
34:     CloseDatabase();
35:     FreeMemory();
36: }
37: //-----
38:
39: void __fastcall CompoundData::Assign(CompoundData *Source)
40: {
41:     FreeMemory();
42:
43:     StrCopy(MainFields.IUPACName, Source->MainFields.IUPACName);
44:     StrCopy(MainFields.IUPACTranslate, Source->MainFields.IUPACTranslate);
45:     StrCopy(MainFields.MolecularFormula, Source->MainFields.MolecularFormula);
46:     StrCopy(MainFields.ChemicalRegistryNumber,
47:             Source->MainFields.ChemicalRegistryNumber);
48:
49:     MainFields.FixPhysicochemicalProperties =
50:         Source->MainFields.FixPhysicochemicalProperties;
51:
52:     TrivialNames = Source->TrivialNames;
53:     TradeNames = Source->TradeNames;
54:
55:     MainFields.PaintToolShapeCount = Source->MainFields.PaintToolShapeCount;
56:     if(MainFields.PaintToolShapeCount)
57:         StructuralFormula = new GraphData[MainFields.PaintToolShapeCount];
58:     for(register i = 0; i < MainFields.PaintToolShapeCount; i++)
59:         StructuralFormula[i] = Source->StructuralFormula[i];
60:
61:     PhysicochemicalProperties = Source->PhysicochemicalProperties;
62:
63:     StructuralFormulaPic.PictureSize = Source->StructuralFormulaPic.PictureSize;
64:     if(StructuralFormulaPic.PictureSize)
65:         StructuralFormulaPic.PictureBuffer = new
66:             char[StructuralFormulaPic.PictureSize];
67:     memmove(StructuralFormulaPic.PictureBuffer,
68:             Source->StructuralFormulaPic.PictureBuffer, StructuralFormulaPic.PictureSize);
69:
70:     MainFields.SpectralCount = Source->MainFields.SpectralCount;
71:     if(MainFields.SpectralCount)
72:         SpectralProperties = new SpectralProperty[MainFields.SpectralCount];
73:     for(register i = 0; i < MainFields.SpectralCount; i++)
74:     {
75:         SpectralProperties[i].PictureSize = Source->SpectralProperties[i].PictureSize;
```

```
72:         SpectralProperties[i].Label = Source->SpectralProperties[i].Label;
73:         SpectralProperties[i].Category = Source->SpectralProperties[i].Category;
74:         SpectralProperties[i].Description = Source->SpectralProperties[i].Description;
75:         SpectralProperties[i].PictureBuffer = new
char[SpectralProperties[i].PictureSize];
76:         memmove(SpectralProperties[i].PictureBuffer,
Source->SpectralProperties[i].PictureBuffer, SpectralProperties[i].PictureSize);
77:     }
78:
79:     MainFields.ChoromatographyCount = Source->MainFields.ChoromatographyCount;
80:     if(MainFields.ChoromatographyCount)
81:         CAnalysis = new ChoromatographyAnalysis[MainFields.ChoromatographyCount];
82:     for(register i = 0; i < MainFields.ChoromatographyCount; i++)
83:     {
84:         CAnalysis[i].PictureSize = Source->CAnalysis[i].PictureSize;
85:         CAnalysis[i].Label = Source->CAnalysis[i].Label;
86:         CAnalysis[i].Description = Source->CAnalysis[i].Description;
87:         CAnalysis[i].PictureBuffer = new char[CAnalysis[i].PictureSize];
88:         memmove(CAnalysis[i].PictureBuffer, Source->CAnalysis[i].PictureBuffer,
CAnalysis[i].PictureSize);
89:     }
90:
91:     SymmetricProperties = Source->SymmetricProperties;
92:     Comments = Source->Comments;
93: }
94: //-----
95:
96: void __fastcall CompoundData::GetMainFormData(void)
97: {
98:     FreeMemory();
99:
100:     TMemoryStream *MemStream = new TMemoryStream;
101:     TJPEGImage *jp = new TJPEGImage();
102:
103:     StrCopy(MainFields.IUPACName, MainForm->Edit1->Text.c_str());
104:     StrCopy(MainFields.IUPACTranslate, MainForm->LabeledEdit2->Text.c_str());
105:     StrCopy(MainFields.MolecularFormula, MainForm->Edit3->Text.c_str());
106:     StrCopy(MainFields.ChemicalRegistryNumber, MainForm->Edit4->Text.c_str());
107:
108:     PhysicochemicalProperties.ACXNumber = MainForm->Edit50->Text;
109:     MainFields.FixPhysicochemicalProperties.BoilingPoint =
StringToFloat(MainForm->Edit5->Text);
110:     MainFields.FixPhysicochemicalProperties.BoilingPointPM =
StringToFloat(MainForm->Edit32->Text);
111:     MainFields.FixPhysicochemicalProperties.Density =
StringToFloat(MainForm->Edit9->Text);
112:     MainFields.FixPhysicochemicalProperties.DensityPM =
StringToFloat(MainForm->Edit33->Text);
113:     MainFields.FixPhysicochemicalProperties.DielectricConstant =
StringToFloat(MainForm->Edit10->Text);
114:     MainFields.FixPhysicochemicalProperties.DielectricConstantPM =
StringToFloat(MainForm->Edit34->Text);
115:     MainFields.FixPhysicochemicalProperties.DipoleMoment =
StringToFloat(MainForm->Edit11->Text);
116:     MainFields.FixPhysicochemicalProperties.DipoleMomentPM =
StringToFloat(MainForm->Edit35->Text);
117:     MainFields.FixPhysicochemicalProperties.EvaporationRate =
StringToFloat(MainForm->Edit12->Text);
118:     MainFields.FixPhysicochemicalProperties.EvaporationRatePM =
StringToFloat(MainForm->Edit36->Text);
119:     MainFields.FixPhysicochemicalProperties.FlashPoint =
StringToFloat(MainForm->Edit13->Text);
120:     MainFields.FixPhysicochemicalProperties.FlashPointPM =
StringToFloat(MainForm->Edit37->Text);
121:     MainFields.FixPhysicochemicalProperties.RefractiveIndex =
StringToFloat(MainForm->Edit14->Text);
122:     MainFields.FixPhysicochemicalProperties.RefractiveIndexPM =
StringToFloat(MainForm->Edit38->Text);
123:     MainFields.FixPhysicochemicalProperties.MeltingPoint =
StringToFloat(MainForm->Edit15->Text);
124:     MainFields.FixPhysicochemicalProperties.MeltingPointPM =
StringToFloat(MainForm->Edit39->Text);
125:     MainFields.FixPhysicochemicalProperties.MolecularWeight =
StringToFloat(MainForm->Edit16->Text);
126:     MainFields.FixPhysicochemicalProperties.MolarRefractivity =
```

```
StringToFloat(MainForm->Edit17->Text);
127:     MainFields.FixPhysicochemicalProperties.MolarRefractivityPM =
StringToFloat(MainForm->Edit41->Text);
128:     MainFields.FixPhysicochemicalProperties.MolarVolume =
StringToFloat(MainForm->Edit18->Text);
129:     MainFields.FixPhysicochemicalProperties.MolarVolumePM =
StringToFloat(MainForm->Edit42->Text);
130:     MainFields.FixPhysicochemicalProperties.PKA = StringToFloat(MainForm->Edit20->Text);
131:     MainFields.FixPhysicochemicalProperties.PKAPM =
StringToFloat(MainForm->Edit43->Text);
132:     MainFields.FixPhysicochemicalProperties.Polarizability =
StringToFloat(MainForm->Edit22->Text);
133:     MainFields.FixPhysicochemicalProperties.PolarizabilityPM =
StringToFloat(MainForm->Edit44->Text);
134:     MainFields.FixPhysicochemicalProperties.RTECS =
StringToFloat(MainForm->Edit53->Text);
135:     MainFields.FixPhysicochemicalProperties.RTECSPM =
StringToFloat(MainForm->Edit52->Text);
136:     MainFields.FixPhysicochemicalProperties.SurfaceTension =
StringToFloat(MainForm->Edit23->Text);
137:     MainFields.FixPhysicochemicalProperties.SurfaceTensionPM =
StringToFloat(MainForm->Edit45->Text);
138:     MainFields.FixPhysicochemicalProperties.VaporDensity =
StringToFloat(MainForm->Edit24->Text);
139:     MainFields.FixPhysicochemicalProperties.VaporDensityPM =
StringToFloat(MainForm->Edit46->Text);
140:     MainFields.FixPhysicochemicalProperties.VaporPressure =
StringToFloat(MainForm->Edit25->Text);
141:     MainFields.FixPhysicochemicalProperties.VaporPressurePM =
StringToFloat(MainForm->Edit47->Text);
142:     MainFields.FixPhysicochemicalProperties.Viscosity =
StringToFloat(MainForm->Edit26->Text);
143:     MainFields.FixPhysicochemicalProperties.ViscosityPM =
StringToFloat(MainForm->Edit48->Text);
144:     MainFields.FixPhysicochemicalProperties.WaterSolubility =
StringToFloat(MainForm->Edit27->Text);
145:     MainFields.FixPhysicochemicalProperties.WaterSolubilityPM =
StringToFloat(MainForm->Edit49->Text);
146:     MainFields.FixPhysicochemicalProperties.ThermodynamicStandarProperties.H =
StringToFloat(MainForm->Edit30->Text);
147:     MainFields.FixPhysicochemicalProperties.ThermodynamicStandarProperties.C =
StringToFloat(MainForm->Edit28->Text);
148:     MainFields.FixPhysicochemicalProperties.ThermodynamicStandarProperties.G =
StringToFloat(MainForm->Edit31->Text);
149:     MainFields.FixPhysicochemicalProperties.ThermodynamicStandarProperties.S =
StringToFloat(MainForm->Edit29->Text);
150:
151:     TrivialNames = MainForm->ValueListEditor1->Strings->Text;
152:     TradeNames = MainForm->ValueListEditor2->Strings->Text;
153:
154:     MainFields.PaintToolShapeCount = MainForm->PaintTool->ShapeCounter;
155:     if(MainForm->PaintTool->ShapeCounter)
156:         StructuralFormula = new GraphData[MainForm->PaintTool->ShapeCounter];
157:     for(register i = 0; i < MainForm->PaintTool->ShapeCounter; i++)
158:         StructuralFormula[i] = *MainForm->PaintTool->GData[i];
159:
160:     PhysicochemicalProperties.Composition = MainForm->Edit6->Text;
161:     PhysicochemicalProperties.Color = MainForm->Edit7->Text;
162:     PhysicochemicalProperties.CrystallineForm = MainForm->Edit8->Text;
163:     PhysicochemicalProperties.EPACode = MainForm->Edit54->Text;
164:     PhysicochemicalProperties.Odor = MainForm->Edit19->Text;
165:     PhysicochemicalProperties.DOTNumber = MainForm->Edit51->Text;
166:     PhysicochemicalProperties.PhysicalState = MainForm->Edit21->Text;
167:
168:     jp->Assign(MainForm->Image7->Picture->Bitmap);
169:     jp->CompressionQuality = 80;
170:     jp->SaveToStream(MemStream);
171:     StructuralFormulaPic.PictureSize = MemStream->Position;
172:     MemStream->Position = 0;
173:     if(StructuralFormulaPic.PictureSize)
174:         StructuralFormulaPic.PictureBuffer = new
char[StructuralFormulaPic.PictureSize];
175:     MemStream->Read(StructuralFormulaPic.PictureBuffer,
StructuralFormulaPic.PictureSize);
176:
```

```
177:     MainFields.SpectralCount = MainForm->ScrollBar2->ComponentCount;
178:     if(MainForm->ScrollBar2->ComponentCount)
179:         SpectralProperties = new SpectralProperty[MainFields.SpectralCount];
180:     for(register i = 0; i < MainFields.SpectralCount; i++)
181:         SetSpectralProperty(&SpectralProperties[i], (TPanel
*)MainForm->ScrollBar2->Components[i]);
182:
183:     MainFields.ChoromatographyCount = MainForm->ScrollBar3->ComponentCount;
184:     if(MainForm->ScrollBar3->ComponentCount)
185:         CAnalysis = new ChoromatographyAnalysis[MainFields.ChoromatographyCount];
186:     for(register i = 0; i < MainFields.ChoromatographyCount; i++)
187:         SetChoromatographyAnalysis(&CAnalysis[i], (TPanel
*)MainForm->ScrollBar3->Components[i]);
188:
189:     SymmetricProperties.PointGroup = MainForm->LabeledEdit1->Text;
190:     SymmetricProperties.SymmetryElements = MainForm->Memo2->Text;
191:
192:     Comments = MainForm->Memo4->Text;
193:
194:     delete jp;
195:     delete MemStream;
196: }
197: //-----
198:
199: void __fastcall CompoundData::SetMainFormData(void)
200: {
201:     TMemoryStream *MemStream = new TMemoryStream;
202:     TJPEGImage *jp = new TJPEGImage();
203:
204:     MainForm->Edit1->Text = MainFields.IUPACName;
205:     MainForm->LabeledEdit2->Text = MainFields.IUPACTranslate;
206:     MainForm->Edit3->Text = MainFields.MolecularFormula;
207:     MainForm->Edit4->Text = MainFields.ChemicalRegistryNumber;
208:
209:     MainForm->Edit50->Text = PhysicochemicalProperties.ACXNumber;
210:     MainForm->Edit5->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.BoilingPoint);
211:     MainForm->Edit32->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.BoilingPointPM);
212:     MainForm->Edit9->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.Density);
213:     MainForm->Edit33->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.DensityPM);
214:     MainForm->Edit10->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.DielectricConstant);
215:     MainForm->Edit34->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.DielectricConstantPM);
216:     MainForm->Edit11->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.DipoleMoment);
217:     MainForm->Edit35->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.DipoleMomentPM);
218:     MainForm->Edit12->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.EvaporationRate);
219:     MainForm->Edit36->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.EvaporationRatePM);
220:     MainForm->Edit13->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.FlashPoint);
221:     MainForm->Edit37->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.FlashPointPM);
222:     MainForm->Edit14->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.RefractiveIndex);
223:     MainForm->Edit38->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.RefractiveIndexPM);
224:     MainForm->Edit15->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.MeltingPoint);
225:     MainForm->Edit39->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.MeltingPointPM);
226:     MainForm->Edit16->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.MolecularWeight);
227:     MainForm->Edit17->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.MolarRefractivity);
228:     MainForm->Edit41->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.MolarRefractivityPM);
229:     MainForm->Edit18->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.MolarVolume);
```

```
230:     MainForm->Edit42->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.MolarVolumePM);
231:     MainForm->Edit20->Text = FloatToString(MainFields.FixPhysicochemicalProperties.PK2
232:     MainForm->Edit43->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.PKAPM);
233:     MainForm->Edit22->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.Polarizibility);
234:     MainForm->Edit44->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.PolarizibilityPM);
235:     MainForm->Edit53->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.RTECS);
236:     MainForm->Edit52->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.RTECSPM);
237:     MainForm->Edit23->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.SurfaceTension);
238:     MainForm->Edit45->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.SurfaceTensionPM);
239:     MainForm->Edit24->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.VaporDensity);
240:     MainForm->Edit46->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.VaporDensityPM);
241:     MainForm->Edit25->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.VaporPressure);
242:     MainForm->Edit47->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.VaporPressurePM);
243:     MainForm->Edit26->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.Viscosity);
244:     MainForm->Edit48->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.ViscosityPM);
245:     MainForm->Edit27->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.WaterSolubility);
246:     MainForm->Edit49->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.WaterSolubilityPM);
247:     MainForm->Edit30->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.ThermodynamicStandarProperties.F
248:     MainForm->Edit28->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.ThermodynamicStandarProperties.C
249:     MainForm->Edit31->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.ThermodynamicStandarProperties.C
250:     MainForm->Edit29->Text =
FloatToString(MainFields.FixPhysicochemicalProperties.ThermodynamicStandarProperties.S
251:
252:     MainForm->ValueListEditor1->Strings->Text = TrivialNames;
253:     MainForm->ValueListEditor2->Strings->Text = TradeNames;
254:
255:     MainForm->PaintTool->DeleteAllShapes();
256:     MainForm->PaintTool->ShapeCounter = MainFields.PaintToolShapeCount;
257:     for(register i = 0; i < MainForm->PaintTool->ShapeCounter; i++)
258:     {
259:         MainForm->PaintTool->GData[i] = new GraphData;
260:         *MainForm->PaintTool->GData[i] = StructuralFormula[i];
261:         if(MainForm->PaintTool->GData[i]->GroupNumber >
MainForm->PaintTool->GroupCounter)
262:             MainForm->PaintTool->GroupCounter =
MainForm->PaintTool->GData[i]->GroupNumber;
263:     }
264:     MainForm->PaintTool->RefreshAllImage();
265:
266:     MainForm->Edit6->Text = PhysicochemicalProperties.Composition;
267:     MainForm->Edit7->Text = PhysicochemicalProperties.Color;
268:     MainForm->Edit8->Text = PhysicochemicalProperties.CrystallineForm;
269:     MainForm->Edit54->Text = PhysicochemicalProperties.EPACode;
270:     MainForm->Edit19->Text = PhysicochemicalProperties.Odor;
271:     MainForm->Edit51->Text = PhysicochemicalProperties.DOTNumber;
272:     MainForm->Edit21->Text = PhysicochemicalProperties.PhysicalState;
273:
274:     MemStream->Clear();
275:     if(StructuralFormulaPic.PictureSize)
276:     {
277:         MemStream->Write(StructuralFormulaPic.PictureBuffer,
StructuralFormulaPic.PictureSize);
278:         MemStream->Position = 0;
279:         jp->LoadFromStream(MemStream);
280:         MainForm->Image7->Picture->Bitmap->Assign(jp);
281:     }
```

```
282:
283:     for(register i = 0; i < MainFields.SpectralCount; i++)
284:     {
285:         TPanel *Panel = new TPanel(MainForm->ScrollBar2);
286:         Panel->Parent = (MainForm->ScrollBar2);
287:         Panel->Top = 5;
288:         Panel->Left = 5 + (i * 80);
289:         Panel->Height = 75;
290:         Panel->Width = 60;
291:         Panel->BevelOuter = bvLowered;
292:         Panel->PopupMenu = MainForm->PictureListItemMenu;
293:
294:         TImage *ActiveImage = new TImage(Panel);
295:         ActiveImage->Parent = Panel;
296:         ActiveImage->Height = 58;
297:         ActiveImage->Width = 58;
298:         ActiveImage->Top = 1;
299:         ActiveImage->Left = 1;
300:         ActiveImage->Stretch = true;
301:         ActiveImage->Canvas->Pen->Color = clWhite;
302:         ActiveImage->Canvas->FillRect(TRect(0, 0, ActiveImage->Height,
ActiveImage->Width));
303:
304:         TLabel *Label = new TLabel(Panel);
305:         Label->Parent = Panel;
306:         Label->Height = 14;
307:         Label->Width = 58;
308:         Label->Alignment = taCenter;
309:         Label->AutoSize = false;
310:         Label->Caption = SpectralProperties[i].Label; //"شماله";
311:         Label->Top = 59;
312:         Label->Left = 1;
313:         Label->Hint = "Label";
314:
315:         TLabel *MemoLabel = new TLabel(Panel);
316:         MemoLabel->Parent = Panel;
317:         MemoLabel->Hint = "Memo";
318:         MemoLabel->Caption = SpectralProperties[i].Description;
319:         MemoLabel->Visible = false;
320:
321:         TLabel *CategoryLabel = new TLabel(Panel);
322:         CategoryLabel->Parent = Panel;
323:         CategoryLabel->Hint = "Category";
324:         CategoryLabel->Caption = SpectralProperties[i].Category;
325:         CategoryLabel->Visible = false;
326:
327:         Panel->OnClick = MainForm->PictureListPanelClick;
328:         ActiveImage->OnClick = MainForm->PictureListPanelChildsClick;
329:         Label->OnClick = MainForm->PictureListPanelChildsClick;
330:
331:         MemStream->Clear();
332:         MemStream->Write(SpectralProperties[i].PictureBuffer,
SpectralProperties[i].PictureSize);
333:         MemStream->Position = 0;
334:         jp->LoadFromStream(MemStream);
335:         ActiveImage->Picture->Bitmap->Assign(jp);
336:
337:         MainForm->SpectralActiveComponent = MainForm->ChangeActivePictureItem(Panel,
MainForm->SpectralActiveComponent);
338:         MainForm->SetPictureImage(MainForm->Image5, MainForm->SpectralActiveComponent);
339:         MainForm->SetDescriptionFromPictureList(MainForm->Memo1,
MainForm->SpectralActiveComponent);
340:         MainForm->SetSpectralCategoryFromPictureList(MainForm->ComboBox1,
MainForm->SpectralActiveComponent);
341:
342:         MainForm->SetItemsActivity(MainForm->CurrentActivity);
343:     }
344:
345:     for(register i = 0; i < MainFields.ChoromatographyCount; i++)
346:     {
347:         TPanel *Panel = new TPanel(MainForm->ScrollBar3);
348:         Panel->Parent = (MainForm->ScrollBar3);
349:         Panel->Top = 5;
350:         Panel->Left = 5 + (i * 80);
351:         Panel->Height = 75;
```

```

352:         Panel->Width = 60;
353:         Panel->BevelOuter = bvLowered;
354:         Panel->PopupMenu = MainForm->PictureListItemMenu;
355:
356:         TImage *ActiveImage = new TImage(Panel);
357:         ActiveImage->Parent = Panel;
358:         ActiveImage->Height = 58;
359:         ActiveImage->Width = 58;
360:         ActiveImage->Top = 1;
361:         ActiveImage->Left = 1;
362:         ActiveImage->Stretch = true;
363:         ActiveImage->Canvas->Pen->Color = clWhite;
364:         ActiveImage->Canvas->FillRect(TRect(0, 0, ActiveImage->Height,
ActiveImage->Width));
365:
366:         TLabel *Label = new TLabel(Panel);
367:         Label->Parent = Panel;
368:         Label->Height = 14;
369:         Label->Width = 58;
370:         Label->Alignment = taCenter;
371:         Label->AutoSize = false;
372:         Label->Caption = CAnalysis[i].Label;//"ω L L";
373:         Label->Top = 59;
374:         Label->Left = 1;
375:         Label->Hint = "Label";
376:
377:         TLabel *MemoLabel = new TLabel(Panel);
378:         MemoLabel->Parent = Panel;
379:         MemoLabel->Hint = "Memo";
380:         MemoLabel->Caption = CAnalysis[i].Description;
381:         MemoLabel->Visible = false;
382:
383:         Panel->OnClick = MainForm->PictureListPanelClick;
384:         ActiveImage->OnClick = MainForm->PictureListPanelChildsClick;
385:         Label->OnClick = MainForm->PictureListPanelChildsClick;
386:
387:         MemStream->Clear();
388:         MemStream->Write(CAnalysis[i].PictureBuffer, CAnalysis[i].PictureSize);
389:         MemStream->Position = 0;
390:         jp->LoadFromStream(MemStream);
391:         ActiveImage->Picture->Bitmap->Assign(jp);
392:
393:         MainForm->ChoromatographyActiveComponent =
MainForm->ChangeActivePictureItem(Panel, MainForm->ChoromatographyActiveComponent);
394:         MainForm->SetPictureImage(MainForm->Image6,
MainForm->ChoromatographyActiveComponent);
395:         MainForm->SetDescriptionFromPictureList(MainForm->Memo3,
MainForm->ChoromatographyActiveComponent);
396:
397:         MainForm->SetItemsActivity(MainForm->CurrentActivity);
398:     }
399:
400:     MainForm->LabeledEdit1->Text = SymmetricProperties.PointGroup;
401:     MainForm->Memo2->Text = SymmetricProperties.SymmetryElements;
402:
403:     MainForm->Memo4->Text = Comments;
404:
405:     delete jp;
406:     delete MemStream;
407: }
408: //-----
409:
410: void __fastcall CompoundData::SetSpectralProperty(SpectralProperty *SP, TPanel *Panel)
411: {
412:     for(register i = 0; i < Panel->ComponentCount; i++)
413:     {
414:         if(String(Panel->Components[i]->ClassName()) == "TImage")
415:         {
416:             TMemoryStream *MemStream = new TMemoryStream;
417:             TJPEGImage *jp = new TJPEGImage();
418:
419:             jp->Assign((TImage *)Panel->Components[i]->Picture->Bitmap);
420:             jp->CompressionQuality = 80;
421:             jp->SaveToStream(MemStream);
422:             SP->PictureSize = MemStream->Position;

```

```
423:
424:         MemStream->Position = 0;
425:
426:         SP->PictureBuffer = new char[SP->PictureSize];
427:         MemStream->Read(SP->PictureBuffer, SP->PictureSize);
428:
429:         delete jp;
430:         delete MemStream;
431:     }
432:     else if(String(Panel->Components[i]->ClassName()) == "TLabel")
433:     {
434:         if(((TLabel *)Panel->Components[i])->Hint == "Label")
435:             SP->Label = ((TLabel *)Panel->Components[i])->Caption;
436:         else if(((TLabel *)Panel->Components[i])->Hint == "Category")
437:             SP->Category = ((TLabel *)Panel->Components[i])->Caption;
438:         else if(((TLabel *)Panel->Components[i])->Hint == "Memo")
439:             SP->Description = ((TLabel *)Panel->Components[i])->Caption;
440:     }
441: }
442: }
443: //-----
444:
445: void __fastcall CompoundData::SetChoromatographyAnalysis(ChoromatographyAnalysis *CA,
    TPanel *Panel)
446: {
447:     for(register i = 0; i < Panel->ComponentCount; i++)
448:     {
449:         if(String(Panel->Components[i]->ClassName()) == "TImage")
450:         {
451:             TMemoryStream *MemStream = new TMemoryStream;
452:             TJPEGImage *jp = new TJPEGImage();
453:
454:             jp->Assign(((TImage *)Panel->Components[i])->Picture->Bitmap);
455:             jp->CompressionQuality = 80;
456:             jp->SaveToStream(MemStream);
457:             CA->PictureSize = MemStream->Position;
458:
459:             MemStream->Position = 0;
460:
461:             CA->PictureBuffer = new char[CA->PictureSize];
462:             MemStream->Read(CA->PictureBuffer, CA->PictureSize);
463:
464:             delete jp;
465:             delete MemStream;
466:         }
467:         else if(String(Panel->Components[i]->ClassName()) == "TLabel")
468:         {
469:             if(((TLabel *)Panel->Components[i])->Hint == "Label")
470:                 CA->Label = ((TLabel *)Panel->Components[i])->Caption;
471:             else if(((TLabel *)Panel->Components[i])->Hint == "Memo")
472:                 CA->Description = ((TLabel *)Panel->Components[i])->Caption;
473:         }
474:     }
475: }
476: //-----
477:
478: int __fastcall CompoundData::Load(int NodeAddress)
479: {
480:     if(!NodeAddress)
481:         NodeAddress = DataFile->CurrentNodeAddress;
482:
483:     if(NodeAddress)
484:     {
485:         char *Buffer;
486:         char *Pos;
487:         int DataLength;
488:         int FieldCounter = 1;
489:
490:         Clear();
491:
492:         DataFile->Go(NodeAddress);
493:         Buffer = (char *)DataFile->AllocGetCurrentNode(Buffer);
494:
495:         memmove(&MainFields, Buffer, sizeof(FixFields));
496:     }
```

```
497:         Pos = Buffer + sizeof(FixFields);
498:
499:         DataLength = *(int *)Pos;
500:         while(DataLength != -1)
501:         {
502:             DataLength = *(int *)Pos;
503:             Pos += sizeof(int);
504:
505:             switch(FieldCounter)
506:             {
507:                 case 1:
508:                     TrivialNames = Pos;
509:                     TrivialNames.SetLength(DataLength);
510:                     break;
511:
512:                 case 2:
513:                     TradeNames = Pos;
514:                     TradeNames.SetLength(DataLength);
515:                     break;
516:
517:                 case 3:
518:                     Comments = Pos;
519:                     Comments.SetLength(DataLength);
520:                     break;
521:
522:                 case 4:
523:                     if(MainFields.PaintToolShapeCount)
524:                         StructuralFormula = new
GraphData[MainFields.PaintToolShapeCount];
525:                     memmove(StructuralFormula, Pos, sizeof(GraphData) *
MainFields.PaintToolShapeCount);
526:
527:                 case 5:
528:                     PhysicochemicalProperties.Composition = Pos;
529:                     PhysicochemicalProperties.Composition.SetLength(DataLength);
530:                     break;
531:
532:                 case 6:
533:                     PhysicochemicalProperties.Color = Pos;
534:                     PhysicochemicalProperties.Color.SetLength(DataLength);
535:                     break;
536:
537:                 case 7:
538:                     PhysicochemicalProperties.CrystallineForm = Pos;
539:                     PhysicochemicalProperties.CrystallineForm.SetLength(DataLength);
540:                     break;
541:
542:                 case 8:
543:                     PhysicochemicalProperties.EPACode = Pos;
544:                     PhysicochemicalProperties.EPACode.SetLength(DataLength);
545:                     break;
546:
547:                 case 9:
548:                     PhysicochemicalProperties.Odor = Pos;
549:                     PhysicochemicalProperties.Odor.SetLength(DataLength);
550:                     break;
551:
552:                 case 10:
553:                     SymmetricProperties.PointGroup = Pos;
554:                     SymmetricProperties.PointGroup.SetLength(DataLength);
555:                     break;
556:
557:                 case 11:
558:                     SymmetricProperties.SymmetryElements = Pos;
559:                     SymmetricProperties.SymmetryElements.SetLength(DataLength);
560:                     break;
561:
562:                 case 12:
563:                     StructuralFormulaPic.PictureSize = DataLength;
564:                     if(DataLength)
565:                         StructuralFormulaPic.PictureBuffer = new char[DataLength];
566:                     memmove(StructuralFormulaPic.PictureBuffer, Pos, DataLength);
567:                     break;
568:
569:                 case 13:
```

```
570:         PhysicochemicalProperties.DOTNumber = Pos;
571:         PhysicochemicalProperties.DOTNumber.SetLength(DataLength);
572:         break;
573:
574:     case 14:
575:         PhysicochemicalProperties.PhysicalState = Pos;
576:         PhysicochemicalProperties.PhysicalState.SetLength(DataLength);
577:         break;
578:
579:     case 15:
580:         PhysicochemicalProperties.ACXNumber = Pos;
581:         PhysicochemicalProperties.ACXNumber.SetLength(DataLength);
582:         break;
583:     }
584:
585:     Pos += DataLength;
586:     FieldCounter++;
587: }
588:
589: if(Buffer)
590:     delete[] Buffer;
591:
592: LoadSpectralProperty();
593: LoadChoromatographyAnalysis();
594: }
595:
596: CurrentNodeAddress = NodeAddress;
597: return NodeAddress;
598: }
599: //-----
600:
601: int __fastcall CompoundData::Save(bool ReplaceNode)
602: {
603:     bool DoAdd = true;
604:     int ReturnValue = 0;
605:     TMemoryStream *MemBuffer = new TMemoryStream;
606:     int ReplaceNodeAddress = ReplaceNode ? CurrentNodeAddress : 0;
607:     int DataLength;
608:
609:     RepeatNodeAddress = 0;
610:
611:     DataFile->Go(TOP);
612:     if(StrLen(MainFields.IUPACName))
613:     {
614:         if(DataFile->BSearch(MainFields.IUPACName))
615:         {
616:             if((CurrentNodeAddress != DataFile->CurrentNodeAddress && ReplaceNode) |
!ReplaceNode)
617:             {
618:                 if(Warning->ShowWarning("اخطار", "آیا بازنویسی", "رکورد تکراری است. آیا بازنویسی", "بله", "خیر", "غردد؟") != 2)
619:                     DoAdd = false;
620:                 else
621:                     RepeatNodeAddress = DataFile->CurrentNodeAddress;
622:             }
623:         }
624:     }
625:
626:     if(DoAdd)
627:     {
628:         MemBuffer->Write(&MainFields, sizeof(FixFields));
629:
630:         DataLength = TrivialNames.Length();
631:         MemBuffer->Write(&DataLength, sizeof(int));
632:         MemBuffer->Write(TrivialNames.c_str(), DataLength);
633:
634:         DataLength = TradeNames.Length();
635:         MemBuffer->Write(&DataLength, sizeof(int));
636:         MemBuffer->Write(TradeNames.c_str(), DataLength);
637:
638:         DataLength = Comments.Length();
639:         MemBuffer->Write(&DataLength, sizeof(int));
640:         MemBuffer->Write(Comments.c_str(), DataLength);
641:
642:         DataLength = MainFields.PaintToolShapeCount * sizeof(GraphData);
```

```
643:         MemBuffer->Write(&DataLength, sizeof(int));
644:         MemBuffer->Write(StructuralFormula, DataLength);
645:
646:         DataLength = PhysicochemicalProperties.Composition.Length();
647:         MemBuffer->Write(&DataLength, sizeof(int));
648:         MemBuffer->Write(PhysicochemicalProperties.Composition.c_str(), DataLength);
649:
650:         DataLength = PhysicochemicalProperties.Color.Length();
651:         MemBuffer->Write(&DataLength, sizeof(int));
652:         MemBuffer->Write(PhysicochemicalProperties.Color.c_str(), DataLength);
653:
654:         DataLength = PhysicochemicalProperties.CrystallineForm.Length();
655:         MemBuffer->Write(&DataLength, sizeof(int));
656:         MemBuffer->Write(PhysicochemicalProperties.CrystallineForm.c_str(),
DataLength);
657:
658:         DataLength = PhysicochemicalProperties.EPACode.Length();
659:         MemBuffer->Write(&DataLength, sizeof(int));
660:         MemBuffer->Write(PhysicochemicalProperties.EPACode.c_str(), DataLength);
661:
662:         DataLength = PhysicochemicalProperties.Odor.Length();
663:         MemBuffer->Write(&DataLength, sizeof(int));
664:         MemBuffer->Write(PhysicochemicalProperties.Odor.c_str(), DataLength);
665:
666:         DataLength = SymmetricProperties.PointGroup.Length();
667:         MemBuffer->Write(&DataLength, sizeof(int));
668:         MemBuffer->Write(SymmetricProperties.PointGroup.c_str(), DataLength);
669:
670:         DataLength = SymmetricProperties.SymmetryElements.Length();
671:         MemBuffer->Write(&DataLength, sizeof(int));
672:         MemBuffer->Write(SymmetricProperties.SymmetryElements.c_str(), DataLength);
673:
674:         MemBuffer->Write(&StructuralFormulaPic.PictureSize, sizeof(int));
675:         MemBuffer->Write(StructuralFormulaPic.PictureBuffer,
StructuralFormulaPic.PictureSize);
676:
677:         DataLength = PhysicochemicalProperties.DOTNumber.Length();
678:         MemBuffer->Write(&DataLength, sizeof(int));
679:         MemBuffer->Write(PhysicochemicalProperties.DOTNumber.c_str(), DataLength);
680:
681:         DataLength = PhysicochemicalProperties.PhysicalState.Length();
682:         MemBuffer->Write(&DataLength, sizeof(int));
683:         MemBuffer->Write(PhysicochemicalProperties.PhysicalState.c_str(), DataLength);
684:
685:         DataLength = PhysicochemicalProperties.ACXNumber.Length();
686:         MemBuffer->Write(&DataLength, sizeof(int));
687:         MemBuffer->Write(PhysicochemicalProperties.ACXNumber.c_str(), DataLength);
688:
689:         MemBuffer->Write(EndOfBuffer, sizeof(int)); //Sign of End of Record
690:
691:         if(RepeatNodeAddress)
692:         {
693:             DataFile->Go(RepeatNodeAddress);
694:             DataFile->DeleteNode();
695:         }
696:
697:         if(ReplaceNodeAddress)
698:         {
699:             DataFile->Go(ReplaceNodeAddress);
700:             DataFile->DeleteNode();
701:         }
702:
703:         DataFile->Go(TOP);
704:         DataFile->AddNode(MemBuffer->Memory, MemBuffer->Size, MemBuffer->Memory, 400);
705:
706:         SaveSpectralProperty();
707:         SaveChoromatologyAnalysis();
708:
709:         CurrentNodeAddress = ReturnValue = DataFile->CurrentNodeAddress;
710:     }
711:
712:     delete MemBuffer;
713:     return ReturnValue;
714: }
715: //-----
```

```
716:
717: bool __fastcall CompoundData::OpenDatabase(void)
718: {
719:     bool ReturnValue = false;
720:
721:     if(!DataFileName.IsEmpty())
722:     {
723:         DataFile = new Fbtree(DataFileName.c_str(), YES, NO, NULL, NULL, NO);
724:         DataFile->ProgressBar = MainForm->MainProgressBar;
725:         ReturnValue = true;
726:     }
727:
728:     return ReturnValue;
729: }
730: //-----
731:
732: void __fastcall CompoundData::CloseDatabase(void)
733: {
734:     if(DataFile)
735:         delete DataFile;
736:     DataFile = NULL;
737: }
738: //-----
739:
740: void __fastcall CompoundData::DeleteRecordAt(int NodeAddress)
741: {
742:     if(!NodeAddress)
743:         NodeAddress = CurrentNodeAddress;
744:
745:     if(NodeAddress)
746:     {
747:         DataFile->Go(NodeAddress);
748:         if(!DataFile->DeleteNode())
749:             CurrentNodeAddress = 0;
750:     }
751: }
752: //-----
753:
754: void __fastcall CompoundData::Clear(void)
755: {
756:     FreeMemory();
757:     memset(&MainFields, 0, sizeof(MainFields));
758:
759:     TrivialNames = "";
760:     TradeNames = "";
761:
762:     PhysicochemicalProperties.ACXNumber = "";
763:     PhysicochemicalProperties.Composition = "";
764:     PhysicochemicalProperties.Color = "";
765:     PhysicochemicalProperties.CrystallineForm = "";
766:     PhysicochemicalProperties.EPACode = "";
767:     PhysicochemicalProperties.Odor = "";
768:     PhysicochemicalProperties.PhysicalState = "";
769:
770:     SymmetricProperties.PointGroup = "";
771:     SymmetricProperties.SymmetryElements = "";
772:
773:     Comments = "";
774: }
775: //-----
776:
777: void __fastcall CompoundData::FreeMemory(void)
778: {
779:     if(StructuralFormula)
780:         delete[] StructuralFormula;
781:
782:     if(StructuralFormulaPic.PictureBuffer)
783:         delete[] StructuralFormulaPic.PictureBuffer;
784:
785:
786:     if(SpectralProperties)
787:     {
788:         for(register i = 0; i < MainFields.SpectralCount; i++)
789:             delete[] SpectralProperties[i].PictureBuffer;
790:         delete[] SpectralProperties;
```

```
791:     }
792:
793:     if(CAnalysis)
794:     {
795:         for(register i = 0; i < MainFields.ChoromatographyCount; i++)
796:             delete[] CAnalysis[i].PictureBuffer;
797:         delete[] CAnalysis;
798:     }
799:
800:     StructuralFormulaPic.PictureBuffer = NULL;
801:     StructuralFormula = NULL;
802:     SpectralProperties = NULL;
803:     CAnalysis = NULL;
804: }
805: //-----
806:
807: void __fastcall CompoundData::LoadSpectralProperty(void)
808: {
809:     void *Result;
810:     char *Buffer;
811:     char *Pos;
812:     int DataLength;
813:     int i = 0;
814:     int RecordPosition = DataFile->CurrentNodeAddress;
815:
816:     if(DataFile->Go(SUB))
817:     {
818:         if(DataFile->BSearch(SpectralBranchName))
819:         {
820:             if(DataFile->Go(SUB))
821:             {
822:                 if(MainFields.SpectralCount)
823:                     SpectralProperties = new
SpectralProperty[MainFields.SpectralCount];
824:
825:                 Result = DataFile->FreeSearch(2, NULL, NULL, NULL, NULL);
826:                 while(Result && i < MainFields.SpectralCount)
827:                 {
828:                     Buffer = (char *)DataFile->AllocGetCurrentNode(Buffer);
829:                     Pos = Buffer;
830:
831:                     SpectralProperties[i].PictureSize = *(int *)Pos;
832:                     Pos += sizeof(int);
833:
834:                     SpectralProperties[i].PictureBuffer = new
char[SpectralProperties[i].PictureSize];
835:                     memmove(SpectralProperties[i].PictureBuffer, Pos,
SpectralProperties[i].PictureSize);
836:                     Pos += SpectralProperties[i].PictureSize;
837:
838:                     DataLength = *(int *)Pos;
839:                     Pos += sizeof(int);
840:                     SpectralProperties[i].Label = Pos;
841:                     SpectralProperties[i].Label.SetLength(DataLength);
842:                     Pos += DataLength;
843:
844:                     DataLength = *(int *)Pos;
845:                     Pos += sizeof(int);
846:                     SpectralProperties[i].Category = Pos;
847:                     SpectralProperties[i].Category.SetLength(DataLength);
848:                     Pos += DataLength;
849:
850:                     DataLength = *(int *)Pos;
851:                     Pos += sizeof(int);
852:                     SpectralProperties[i].Description = Pos;
853:                     SpectralProperties[i].Description.SetLength(DataLength);
854:
855:                     delete[] Buffer;
856:                     i++;
857:                     Result = DataFile->FreeSearch(2, NULL, NULL, NULL, NULL, NEXT);
858:                 }
859:             }
860:         }
861:     }
862: }
```

```

863:     DataFile->Go(RecordPosition);
864: }
865: //-----
866:
867: void __fastcall CompoundData::SaveSpectralProperty(void)
868: {
869:     int RecordPosition = DataFile->CurrentNodeAddress;
870:     int DataLength;
871:     TMemoryStream *MemBuffer = new TMemoryStream;
872:
873:     if(DataFile->Go(SUB)) //Making Category Branch
874:     {
875:         if(!DataFile->BSearch(SpectralBranchName))
876:             DataFile->AddNode(SpectralBranchName, StrLen(SpectralBranchName),
SpectralBranchName, 20);
877:     }
878:     else
879:         DataFile->AddSubNode(SpectralBranchName, StrLen(SpectralBranchName),
SpectralBranchName, 20);
880:
881:     for(register i = 0; i < MainFields.SpectralCount; i++)
882:     {
883:         MemBuffer->Clear();
884:
885:         MemBuffer->Write(&SpectralProperties[i].PictureSize , sizeof(int));
886:         MemBuffer->Write(SpectralProperties[i].PictureBuffer,
SpectralProperties[i].PictureSize);
887:
888:         DataLength = SpectralProperties[i].Label.Length();
889:         MemBuffer->Write(&DataLength , sizeof(int));
890:         MemBuffer->Write(SpectralProperties[i].Label.c_str(), DataLength);
891:
892:         DataLength = SpectralProperties[i].Category.Length();
893:         MemBuffer->Write(&DataLength , sizeof(int));
894:         MemBuffer->Write(SpectralProperties[i].Category.c_str(), DataLength);
895:
896:         DataLength = SpectralProperties[i].Description.Length();
897:         MemBuffer->Write(&DataLength , sizeof(int));
898:         MemBuffer->Write(SpectralProperties[i].Description.c_str(), DataLength);
899:
900:         if(!i)
901:             DataFile->AddSubNode(MemBuffer->Memory, MemBuffer->Size,
MemBuffer->Memory, 40);
902:         else
903:         {
904:             DataFile->Go(SUBTOP);
905:             DataFile->AddNode(MemBuffer->Memory, MemBuffer->Size, MemBuffer->Memory,
40);
906:         }
907:     }
908:
909:     DataFile->Go(RecordPosition);
910:     delete MemBuffer;
911: }
912: //-----
913:
914: void __fastcall CompoundData::LoadChoromatographyAnalysis(void)
915: {
916:     void *Result;
917:     char *Buffer;
918:     char *Pos;
919:     int DataLength;
920:     int i = 0;
921:     int RecordPosition = DataFile->CurrentNodeAddress;
922:
923:     if(DataFile->Go(SUB))
924:     {
925:         if(DataFile->BSearch(ChoromatographyBranchName))
926:         {
927:             if(DataFile->Go(SUB))
928:             {
929:                 if(MainFields.ChoromatographyCount)
930:                     CAnalysis = new
ChoromatographyAnalysis[MainFields.ChoromatographyCount];
931:

```

```
932:         Result = DataFile->FreeSearch(2, NULL, NULL, NULL, NULL);
933:         while(Result && i < MainFields.ChoromatographyCount)
934:         {
935:             Buffer = (char *)DataFile->AllocGetCurrentNode(Buffer);
936:             Pos = Buffer;
937:
938:             CAnalysis[i].PictureSize = *(int *)Pos;
939:             Pos += sizeof(int);
940:
941:             CAnalysis[i].PictureBuffer = new char[CAnalysis[i].PictureSize
942:             memmove(CAnalysis[i].PictureBuffer, Pos,
943:             CAnalysis[i].PictureSize);
944:             Pos += CAnalysis[i].PictureSize;
945:
946:             DataLength = *(int *)Pos;
947:             Pos += sizeof(int);
948:             CAnalysis[i].Label = Pos;
949:             CAnalysis[i].Label.SetLength(DataLength);
950:             Pos += DataLength;
951:
952:             DataLength = *(int *)Pos;
953:             Pos += sizeof(int);
954:             CAnalysis[i].Description = Pos;
955:             CAnalysis[i].Description.SetLength(DataLength);
956:
957:             delete[] Buffer;
958:             i++;
959:             Result = DataFile->FreeSearch(2, NULL, NULL, NULL, NULL, NEXT);
960:         }
961:     }
962: }
963:
964: DataFile->Go(RecordPosition);
965: }
966: //-----
967:
968: void __fastcall CompoundData::SaveChoromatographyAnalysis(void)
969: {
970:     int RecordPosition = DataFile->CurrentNodeAddress;
971:     int DataLength;
972:     TMemoryStream *MemBuffer = new TMemoryStream;
973:
974:     if(DataFile->Go(SUB)) //Making Category Branch
975:     {
976:         if(!DataFile->BSearch(ChoromatographyBranchName))
977:             DataFile->AddNode(ChoromatographyBranchName,
978:             StrLen(ChoromatographyBranchName), ChoromatographyBranchName, 20);
979:     }
980:     else
981:         DataFile->AddSubNode(ChoromatographyBranchName,
982:         StrLen(ChoromatographyBranchName), ChoromatographyBranchName, 20);
983:
984:     for(register i = 0; i < MainFields.ChoromatographyCount; i++)
985:     {
986:         MemBuffer->Clear();
987:
988:         MemBuffer->Write(&CAnalysis[i].PictureSize, sizeof(int));
989:         MemBuffer->Write(CAnalysis[i].PictureBuffer, CAnalysis[i].PictureSize);
990:
991:         DataLength = CAnalysis[i].Label.Length();
992:         MemBuffer->Write(&DataLength, sizeof(int));
993:         MemBuffer->Write(CAnalysis[i].Label.c_str(), DataLength);
994:
995:         DataLength = CAnalysis[i].Description.Length();
996:         MemBuffer->Write(&DataLength, sizeof(int));
997:         MemBuffer->Write(CAnalysis[i].Description.c_str(), DataLength);
998:
999:         if(!i)
1000:             DataFile->AddSubNode(MemBuffer->Memory, MemBuffer->Size,
1001:             MemBuffer->Memory, 40);
1002:         else
1003:         {
1004:             DataFile->Go(SUBTOP);
1005:             DataFile->AddNode(MemBuffer->Memory, MemBuffer->Size, MemBuffer->Memory,
```

```
    40);
1003:         }
1004:     }
1005:
1006:     DataFile->Go(RecordPosition);
1007:     delete MemBuffer;
1008: }
1009: //-----
1010:
1011: bool __fastcall CompoundData::CheckForCorrectVersion(void)
1012: {
1013:     bool ReturnValue = true;
1014:
1015:     if(StrComp(DataFile->Header.Provider, "Chemical Compounds Information Data Bank"))
1016:         ReturnValue = false;
1017:
1018:     if(DataFile->Header.Version != 1)
1019:         ReturnValue = false;
1020:
1021:     return ReturnValue;
1022: }
1023: //-----
```

```
1: //-----
2:
3: #include <vcl.h>
4: #pragma hdrstop
5:
6: #include "DeleteFrom.h"
7: //-----
8: #pragma package(smart_init)
9: #pragma resource "*.dfm"
10: TDeleteSelection *DeleteSelection;
11: //-----
12: __fastcall TDeleteSelection::TDeleteSelection(TComponent* Owner)
13:     : TForm(Owner)
14: {
15:     SelectedAction = CancelDelete;
16: }
17: //-----
18:
19: void __fastcall TDeleteSelection::FormKeyPress(TObject *Sender, char &Key)
20: {
21:     if(Key == VK_ESCAPE)
22:         Close();
23: }
24: //-----
25: void __fastcall TDeleteSelection::FormShow(TObject *Sender)
26: {
27:     SelectedAction = CancelDelete;
28: }
29: //-----
30: void __fastcall TDeleteSelection::Button1Click(TObject *Sender)
31: {
32:     SelectedAction = RadioButton1->Checked ? DeleteFromList : DeleteFromDatabase;
33:     Close();
34: }
35: //-----
36:
37: int __fastcall TDeleteSelection::ShowDeleteSelection(void)
38: {
39:     ShowModal();
40:     return SelectedAction;
41: }
42: //-----
43: void __fastcall TDeleteSelection::Button2Click(TObject *Sender)
44: {
45:     Close();
46: }
47: //-----
```

```
1: #include <vcl.h>
2: #pragma hdrstop
3:
4: #include "stdio.h"
5: #include "fbtree.h"
6:
7: #pragma package(smart_init)
8:
9: struct DateSpec
10: {
11:     int year;
12:     char month, day;
13:     char dayofweek;
14: };
15:
16: char fbtree_currentindex_sign;
17: AnsiString ExecutionPath;
18:
19:
20: long date_unique_number(DateSpec *date)
21: {
22:     return (long)date->year * 372 + (date->month - 1) * 31 + (date->day - 1);
23: }
24:
25: bool __fastcall _DeleteFile(const AnsiString FileName, bool UseExecutionPath)
26: {
27:     AnsiString FileNamePath = FileName;
28:     bool returnval;
29:
30:     if(UseExecutionPath)
31:         FileNamePath.Insert(ExecutionPath, 1);
32:
33:     if(FileExists(FileNamePath))
34:         returnval = DeleteFile(FileNamePath);
35:     else
36:         returnval = true;
37:
38:     return returnval;
39: }
40:
41: bool __fastcall _FileExists(const AnsiString FileName, bool UseExecutionPath)
42: {
43:     AnsiString FileNamePath = FileName;
44:     bool returnval;
45:
46:     FileNamePath.Insert(ExecutionPath, 1);
47:     returnval = FileExists(FileNamePath);
48:
49:     return returnval;
50: }
51:
52: Fbtree::Fbtree(char *Filename, char change_header, char casesens, int (*comp)(const char
    *s1, const char *s2, unsigned len), FbtreeIndex *f_index, char use_execpath)
53: {
54:     register i;
55:
56:     UseExecutionPath = use_execpath;
57:     ExecutionPath = UseExecutionPath ? (GetExecutionPath() + "\\") : String("");
58:
59:     record_preview.LeftAddress = record_preview.RightAddress =
    record_preview.SubNodeAddress = record_preview.MotherNodeAddress =
    record_preview.PreviousNodeAddress = NULL;
60:     CaseSensitive = casesens;
61:
62:     for(register i = 0; i <= MAXFBTREELEVEL; i++)
63:     {
64:         LastSearchedAddress[i] = NULL;
65:         Stack[i] = new long[1];
66:         StackPosition[i] = 0;
67:     }
68:
69:     CompareFunction = comp;
70:
71:     FbtreeHeader head = {"HUF Data File", 26, "{æ''-æ", "{æ''-æ", 0, {0}, 0.0, {0},
    {0}};
```

```
72:
73:     if(change_header == YES)
74:         *head.Security1 = *head.Security2 = 0;
75:
76:     Header = head;
77:
78:     StrCopy(FbtreeFilename, Filename);
79:
80:     OptimizeActive = NO;
81:     CurrentFileLevel = 1;
82:     OpenIndexFlag = 0;
83:
84:     Found = _FileExists(Filename, UseExecutionPath);
85:
86:     if(Found)
87:         Mode = fmOpenReadWrite;
88:     else
89:         Mode = fmCreate;
90:
91:     File = new TFileStream(ExecutionPath + Filename, Mode);
92:
93:     if(File->Handle > 0)
94:     {
95:         if(Found == false)
96:             WriteHeader();
97:     }
98:
99:     for(i = 0; i < MAXFBTREEINDEX; i++)
100:         Index[i] = NULL;
101:
102:     fbtree_index = NULL;
103:     IndexCounter = 0;
104:     if(f_index)
105:     {
106:         while(f_index[IndexCounter].Filename && IndexCounter < MAXFBTREEINDEX)
107:             IndexCounter++;
108:
109:         if(IndexCounter)
110:             fbtree_index = new FbtreeIndex[IndexCounter + 1];
111:
112:         for(i = 0; i < IndexCounter; i++)
113:         {
114:             fbtree_index[i].Filename = new char[strlen(f_index[i].Filename) + 1];
115:             StrCopy(fbtree_index[i].Filename, f_index[i].Filename);
116:
117:             fbtree_index[i].Level = f_index[i].Level;
118:             fbtree_index[i].StartKey = f_index[i].StartKey;
119:             fbtree_index[i].Len = f_index[i].Len;
120:             fbtree_index[i].Comp = f_index[i].Comp;
121:             fbtree_index[i].CompareKind = f_index[i].Comp ? fbtFREECOMPARE :
122:             f_index[i].CompareKind;
123:             fbtree_index[i].Sign = f_index[i].Sign;
124:         }
125:         fbtree_index[IndexCounter].Filename = NULL;
126:     }
127:
128:     if(File->Handle > 0)
129:     {
130:         ReadHeader();
131:         GetNextNewNodeAddress();
132:         CurrentNodeAddress = Header.FirstNodeAddress;
133:     }
134:
135:     ProgressBar = NULL;
136: }
137: Fbtree::~Fbtree(void)
138: {
139:     if(IndexCounter)
140:     {
141:         for(register i = IndexCounter - 1; i >= 0; i--)
142:             delete[] fbtree_index[i].Filename;
143:
144:         delete[] fbtree_index;
145:     }
```

```
146:
147:     delete File;
148:
149:     for(register i = MAXFBTREELEVEL; i >= 0; i--)
150:         if(Stack[i])
151:             delete[] Stack[i];
152: }
153:
154: void Fbtree::GetNextNewNodeAddress(void)
155: {
156:     NewNodeAddress = File->Size;
157: }
158:
159: void Fbtree::WriteNewRecord(void *nodestruct)
160: {
161:     if(File->Handle > 0)
162:     {
163:         File->Seek(NewNodeAddress, soFromBeginning);
164:         File->Write(&record_preview, sizeof(RecordPreview));
165:         File->Write(nodestruct, record_preview.RecordLen);
166:         CurrentNodeAddress = NewNodeAddress;
167:
168:         GetNextNewNodeAddress();
169:     }
170: }
171:
172: void Fbtree::WriteHeader(char *source, int start, int size)
173: {
174:     if(File->Handle > 0)
175:     {
176:         if(source)
177:         {
178:             char *ptr = (char *)&Header;
179:
180:             ptr += start;
181:
182:             for(register i = 0; i < size; i++)
183:                 *ptr++ = *(source + i);
184:         }
185:
186:         File->Seek(0, soFromBeginning);
187:         File->Write(&Header, sizeof(FbtreeHeader));
188:     }
189: }
190:
191: void Fbtree::ReadHeader(char *dest, int start, int size)
192: {
193:     if(File->Handle > 0)
194:     {
195:         File->Seek(0, soFromBeginning);
196:         File->Read(&Header, sizeof(FbtreeHeader));
197:
198:         if(dest)
199:         {
200:             char *ptr = (char *)&Header;
201:
202:             ptr += start;
203:
204:             for(register i = 0; i < size; i++)
205:                 *dest++ = *(ptr + i);
206:         }
207:     }
208: }
209:
210: void Fbtree::ReadRecord(void *nodestruct, long nodeaddress)
211: {
212:     if(File->Handle > 0)
213:     {
214:         if(nodeaddress == -1)
215:             nodeaddress = CurrentNodeAddress;
216:
217:         File->Seek(nodeaddress, soFromBeginning);
218:         File->Read(&record_preview, sizeof(RecordPreview));
219:         File->Read(nodestruct, record_preview.RecordLen);
220:     }
```

```
221: }
222:
223: void *Fbtree::AllocGetCurrentNode(void *nodestruct)
224: {
225:     return nodestruct = (char *)AllocReadRecord(nodestruct, CurrentNodeAddress);
226: }
227:
228: void *Fbtree::AllocReadRecord(void *nodestruct, long nodeaddress)
229: {
230:     void *ReturnValue = NULL;
231:
232:     if(File->Handle > 0)
233:     {
234:         if(nodeaddress == -1)
235:             nodeaddress = CurrentNodeAddress;
236:
237:         File->Seek(nodeaddress, soFromBeginning);
238:         File->Read(&record_preview, sizeof(RecordPreview));
239:
240:         ReturnValue = nodestruct = new char[record_preview.RecordLen + 1];
241:
242:         if(nodestruct)
243:             File->Read(nodestruct, record_preview.RecordLen);
244:         *((char *)nodestruct + record_preview.RecordLen) = 0;
245:     }
246:
247:     return ReturnValue;
248: }
249:
250: int Fbtree::Compare(void *comp1, void *comp2, unsigned size, char CompareKind)
251: {
252:     int ReturnValue;
253:
254:     if(CompareKind == fbtFREECOMPARE)
255:         if(CompareFunction)
256:             ReturnValue = CompareFunction((char *)comp1, (char *)comp2, size);
257:         else
258:             CompareKind = fbtSTRING;
259:
260:     if(CompareKind != fbtFREECOMPARE)
261:         switch(CompareKind)
262:         {
263:             case fbtSTRING:
264:                 if(CaseSensitive)
265:                     ReturnValue = strcmp((char *)comp1, (char *)comp2, size);
266:                 else
267:                     ReturnValue = strnicmp((char *)comp1, (char *)comp2, size);
268:                 break;
269:
270:             case fbtCHAR:
271:                 ReturnValue = *(char *)comp1 > *(char *)comp2 ? 1 : *(char *)comp1
*(char *)comp2 ? 0 : -1;
272:                 break;
273:
274:             case fbtINTEGER:
275:                 ReturnValue = *(int *)comp1 > *(int *)comp2 ? 1 : *(int *)comp1 ==
*(int *)comp2 ? 0 : -1;
276:                 break;
277:
278:             case fbtLONG:
279:                 ReturnValue = *(long *)comp1 > *(long *)comp2 ? 1 : *(long *)comp1
*(long *)comp2 ? 0 : -1;
280:                 break;
281:
282:             case fbtFLOAT:
283:                 ReturnValue = *(float *)comp1 > *(float *)comp2 ? 1 : *(float *)com
== *(float *)comp2 ? 0 : -1;
284:                 break;
285:
286:             case fbtDOUBLE:
287:                 ReturnValue = *(double *)comp1 > *(double *)comp2 ? 1 : *(double
*)comp1 == *(double *)comp2 ? 0 : -1;
288:                 break;
289:
290:             case fbtDATE:
```

```
291:         ReturnValue = date_unique_number((DateSpec *)comp1) >
        date_unique_number((DateSpec *)comp2) ? 1 : date_unique_number((DateSpec *)comp1) ==
        date_unique_number((DateSpec *)comp2) ? 0 : -1;
292:         break;
293:     }
294:
295:     return ReturnValue;
296: }
297:
298: char Fbtree::AddNode(void *nodestruct, int nodestructsize, void *startnodekey, int
        keysize, char CompareKind, long nodeaddress)
299: {
300:     char *readnode = NULL, *readnodebak;
301:     char ReturnValue = 0;
302:     int comparevalue;
303:
304:     if(File->Handle > 0)
305:     {
306:         OpenIndexFiles();
307:
308:         record_preview.RecordLen = (unsigned)nodestructsize;
309:         record_preview.KeyStartAddress = (unsigned)((char *)startnodekey - (char
        *)nodestruct);
310:         record_preview.KeyLen = (unsigned)keysizes;
311:         record_preview.CompareKind = CompareKind;
312:
313:         if(nodeaddress > 0)
314:         {
315:             File->Seek(nodeaddress, soFromBeginning);
316:             CurrentNodeAddress = nodeaddress;
317:         }
318:
319:         if(!Header.FirstNodeAddress)
320:         {
321:             record_preview.LeftAddress =
322:             record_preview.RightAddress =
323:             record_preview.SubNodeAddress =
324:             record_preview.MotherNodeAddress =
325:             record_preview.PreviousNodeAddress = 0;
326:
327:             Header.FirstNodeAddress = CurrentNodeAddress = NewNodeAddress;
328:             WriteHeader();
329:             WriteNewRecord(nodestruct);
330:
331:             WriteIndex(nodestruct);
332:         }
333:         else
334:         {
335:             PreviousNodeAddress = CurrentNodeAddress;
336:             if(!nodeaddress)
337:             {
338:                 record_preview.LeftAddress = record_preview.RightAddress =
        record_preview.SubNodeAddress = NULL;
339:                 record_preview.PreviousNodeAddress = PreviousNodeAddress;
340:                 WriteNewRecord(nodestruct);
341:                 WriteIndex(nodestruct);
342:
343:                 File->Seek(PreviousNodeAddress, soFromBeginning);
344:                 File->Read(&record_preview, sizeof(RecordPreview));
345:
346:                 switch(PreviousPath)
347:                 {
348:                     case LEFT:
349:                         record_preview.LeftAddress = CurrentNodeAddress;
350:                         break;
351:
352:                     case RIGHT:
353:                         record_preview.RightAddress = CurrentNodeAddress;
354:                         break;
355:                 }
356:
357:                 File->Seek(PreviousNodeAddress, soFromBeginning);
358:                 File->Write(&record_preview, sizeof(RecordPreview));
359:
360:                 ReturnValue = 1;
```

```
361:         }
362:     else
363:     {
364:         readnode = (char *)AllocReadRecord(readnode, CurrentNodeAddress);
365:         readnodebak = readnode;
366:         readnode += (unsigned)((char *)startnodekey - (char *)nodestruct);
367:
368:         comparevalue = Compare(startnodekey, readnode, keysize, CompareKind);
369:
370:         if(readnode)
371:         {
372:             readnode = readnodebak;
373:             delete[] readnode;
374:         }
375:
376:         if(comparevalue <= 0)
377:         {
378:             PreviousPath = LEFT;
379:             ReturnValue = AddNode(nodestruct, nodestructsize, startnodekey
keysizes, CompareKind, record_preview.LeftAddress);
380:         }
381:     else
382:     {
383:         PreviousPath = RIGHT;
384:         ReturnValue = AddNode(nodestruct, nodestructsize, startnodekey
keysizes, CompareKind, record_preview.RightAddress);
385:     }
386: }
387: }
388:
389: CloseIndexFiles();
390: }
391:
392: return ReturnValue;
393: }
394:
395: char Fbtree::AddSubNode(void *nodestruct, int nodestructsize, void *startnodekey, int
keysizes, char CompareKind, long nodeaddress)
396: {
397:     char *readnode = new char[nodestructsize + 1], *readnodebak = readnode;
398:     char ReturnValue = 0;
399:
400:     if(readnode && File->Handle > 0)
401:     {
402:         if(Header.FirstNodeAddress)
403:         {
404:             CurrentFileLevel++;
405:
406:             OpenIndexFiles();
407:
408:             File->Seek(CurrentNodeAddress, soFromBeginning);
409:             File->Read(&record_preview, sizeof(RecordPreview));
410:
411:             if(record_preview.SubNodeAddress)
412:                 ReturnValue = AddNode(nodestruct, nodestructsize, startnodekey,
keysizes, CompareKind, record_preview.SubNodeAddress);
413:             else
414:             {
415:                 ReturnValue = 1;
416:
417:                 record_preview.RecordLen = (unsigned)nodestructsize;
418:                 record_preview.KeyStartAddress = (unsigned)((char *)startnodekey -
(char *)nodestruct);
419:                 record_preview.KeyLen = (unsigned)keysizes;
420:                 record_preview.CompareKind = CompareKind;
421:
422:                 if(nodeaddress > 0)
423:                 {
424:                     File->Seek(nodeaddress, soFromBeginning);
425:                     CurrentNodeAddress = nodeaddress;
426:                 }
427:
428:                 PreviousNodeAddress = CurrentNodeAddress;
429:
430:                 record_preview.LeftAddress = record_preview.RightAddress =
```

```
record_preview.SubNodeAddress = record_preview.PreviousNodeAddress = NULL;
431:         record_preview.MotherNodeAddress = PreviousNodeAddress;
432:         WriteNewRecord(nodestruct);
433:         WriteIndex(nodestruct);
434:
435:         File->Seek(PreviousNodeAddress, soFromBeginning);
436:         File->Read(&record_preview, sizeof(RecordPreview));
437:
438:         record_preview.SubNodeAddress = CurrentNodeAddress;
439:
440:         File->Seek(PreviousNodeAddress, soFromBeginning);
441:         File->Write(&record_preview, sizeof(RecordPreview));
442:
443:     }
444:
445:     CloseIndexFiles();
446: }
447: else
448:     ReturnValue = AddNode(nodestruct, nodestructsize, startnodekey; keysize,
CompareKind, nodeaddress);
449:
450: }
451:
452: if(readnode)
453: {
454:     readnode = readnodebak;
455:     delete[] readnode;
456: }
457:
458: return ReturnValue;
459: }
460:
461: char Fbtree::GetCurrentNode(void *nodestruct)
462: {
463:     char ReturnValue = 0;
464:
465:     if(Header.FirstNodeAddress)
466:     {
467:         ReadRecord(nodestruct);
468:         ReturnValue = 1;
469:     }
470:
471:     return ReturnValue;
472: }
473:
474: void *Fbtree::BSearch(void *searchstring, long startaddress)
475: {
476:     void *ReturnValue = NULL;
477:     int comparevalue;
478:
479:     if(Header.FirstNodeAddress && File->Handle > 0)
480:     {
481:         if(startaddress == -1)
482:         {
483:             startaddress = CurrentNodeAddress;
484:         }
485:
486:         char *readnode = NULL, *readnodebak;
487:         readnode = (char *)AllocReadRecord(readnode, startaddress);
488:         readnodebak = readnode;
489:         readnode += record_preview.KeyStartAddress;
490:
491:         comparevalue = Compare(searchstring, readnode, record_preview.KeyLen,
record_preview.CompareKind);
492:
493:         if(!comparevalue)
494:         {
495:             CurrentNodeAddress = startaddress;
496:             ReturnValue = readnodebak;
497:         }
498:         else
499:         {
500:             readnode = readnodebak;
501:             delete[] readnode;
502:             readnode = NULL;
```

```
503:
504:         if(comparevalue > 0)
505:         {
506:             if(record_preview.RightAddress)
507:                 ReturnValue = BSearch(searchstring,
record_preview.RightAddress);
508:         }
509:         else
510:         {
511:             if(record_preview.LeftAddress)
512:                 ReturnValue = BSearch(searchstring, record_preview.LeftAddress);
513:         }
514:     }
515:
516:     if(readnode)
517:     {
518:         readnode = readnodebak;
519:         delete[] readnode;
520:     }
521: }
522:
523: return ReturnValue;
524: }
525:
526: void *Fbtree::FreeSearch(char searchlevel, void *searchstring, void *nodestruct, void
*compareaddress, unsigned comparelen, char firstornext, char CompareKind)
527: {
528:     searchlevel--;
529:
530:     void *ReturnValue = NULL;
531:     int comparevalue; // = -1;
532:     char done = 0;
533:     char first_next = firstornext;
534:     long PopAddress, CurrentNodeAddressbak = CurrentNodeAddress;
535:     RecordPreview rpsave;
536:
537:     if(File->Handle > 0 && Header.FirstNodeAddress && (firstornext == FIRST ||
(firstornext == NEXT && LastSearchedAddress[searchlevel])))
538:     {
539:         if(firstornext == NEXT)
540:             Go(LastSearchedAddress[searchlevel]);
541:         else
542:             StackPosition[searchlevel] = 0;
543:
544:         File->Seek(CurrentNodeAddress, soFromBeginning);
545:         File->Read(&record_preview, sizeof(RecordPreview));
546:
547:         char *readnode = NULL, *readnodebak;
548:
549:         readnode = (char *)AllocReadRecord(readnode, CurrentNodeAddress);
550:         readnodebak = readnode;
551:
552:         rpsave = record_preview;
553:
554:         while(!done && readnode)
555:         {
556:             if(rpsave.LeftAddress && first_next == FIRST)
557:             {
558:                 if(!Push(CurrentNodeAddress, searchlevel))
559:                     break;
560:                 Go(LEFT);
561:                 readnode = readnodebak;
562:                 delete[] readnode;
563:                 readnode = (char *)AllocReadRecord(readnode, CurrentNodeAddress);
564:                 readnodebak = readnode;
565:                 rpsave = record_preview;
566:             }
567:             else
568:             {
569:                 if(first_next == FIRST)
570:                 {
571:                     if(searchstring)
572:                     {
573:                         readnode += (unsigned)((char *)compareaddress - (char
*)nodestruct);
```

```
574:                                     comparevalue = Compare(searchstring, readnode, comparelen
CompareKind);
575:                                     }
576:                                     else
577:                                     comparevalue = 0;
578:
579:                                     if(!comparevalue)
580:                                     {
581:                                         ReturnValue = readnodebak;
582:                                         LastSearchedAddress[searchlevel] = CurrentNodeAddress;
583:                                         break;
584:                                     }
585:                                     else
586:                                     {
587:                                         if(Go(RIGHT))
588:                                         {
589:                                             readnode = readnodebak;
590:                                             delete readnode;
591:                                             readnode = (char *)AllocReadRecord(readnode,
CurrentNodeAddress);
592:                                             readnodebak = readnode;
593:                                             rpsave = record_preview;
594:                                         }
595:                                         else
596:                                         {
597:                                             PopAddress = Pop(searchlevel);
598:
599:                                             if(!PopAddress)
600:                                             break;
601:
602:                                             Go(PopAddress);
603:                                             readnode = readnodebak;
604:                                             delete[] readnode;
605:                                             readnode = (char *)AllocReadRecord(readnode,
CurrentNodeAddress);
606:                                             readnodebak = readnode;
607:                                             rpsave = record_preview;
608:                                             rpsave.LeftAddress = 0;
609:                                         }
610:                                     }
611:                                     }
612:                                     else
613:                                     {
614:                                         first_next = FIRST;
615:
616:                                         if(Go(RIGHT))
617:                                         {
618:                                             readnode = readnodebak;
619:                                             delete[] readnode;
620:                                             readnode = (char *)AllocReadRecord(readnode,
CurrentNodeAddress);
621:                                             readnodebak = readnode;
622:                                             rpsave = record_preview;
623:                                         }
624:                                         else
625:                                         {
626:                                             PopAddress = Pop(searchlevel);
627:
628:                                             if(!PopAddress)
629:                                             break;
630:
631:                                             Go(PopAddress);
632:                                             readnode = readnodebak;
633:                                             delete[] readnode;
634:                                             readnode = (char *)AllocReadRecord(readnode,
CurrentNodeAddress);
635:                                             readnodebak = readnode;
636:                                             rpsave = record_preview;
637:                                             rpsave.LeftAddress = 0;
638:                                         }
639:                                     }
640:                                     }
641:                                     }
642:
643:                                     if(readnode)
```

```
644:         {
645:             readnode = readnodebak;
646:             delete[] readnode;
647:         }
648:     }
649:
650:     if(!ReturnValue)
651:         CurrentNodeAddress = CurrentNodeAddressbak;
652:
653:     return ReturnValue;
654: }
655:
656: char Fbtree::Push(long address, char searchlevel)
657: {
658:     char ReturnValue = 1;
659:
660:     StackPosition[searchlevel]++;
661:
662:     StackBak = new long[StackPosition[searchlevel] + 1];
663:     if(!StackBak)
664:         ReturnValue = 0;
665:     else
666:     {
667:         for(register i = 0; i < StackPosition[searchlevel] - 1; i++)
668:             StackBak[i] = Stack[searchlevel][i];
669:
670:         delete[] Stack[searchlevel];
671:         Stack[searchlevel] = StackBak;
672:
673:         StackBak[StackPosition[searchlevel] - 1] = address;
674:     }
675:
676:     return ReturnValue;
677: }
678:
679: long Fbtree::Pop(char searchlevel)
680: {
681:     long ReturnValue = 0;
682:     if(StackPosition[searchlevel])
683:         ReturnValue = Stack[searchlevel][--StackPosition[searchlevel]];
684:
685:     StackBak = new long[StackPosition[searchlevel] + 1];
686:     if(!StackBak)
687:         ReturnValue = 0;
688:     else
689:     {
690:         for(register i = 0; i < StackPosition[searchlevel]; i++)
691:             StackBak[i] = Stack[searchlevel][i];
692:
693:         delete[] Stack[searchlevel];
694:         Stack[searchlevel] = StackBak;
695:     }
696:
697:     return ReturnValue;
698: }
699:
700: char Fbtree::Go(long where)
701: {
702:     char ReturnValue = 0;
703:     int sw = where < 41 ? (int)where : ADDRESS;
704:
705:     if(Header.FirstNodeAddress && File->Handle > 0)
706:     {
707:         File->Seek(CurrentNodeAddress, soFromBeginning);
708:         File->Read(&record_preview, sizeof(RecordPreview));
709:
710:         switch(sw)
711:         {
712:             case LEFT:
713:                 if(record_preview.LeftAddress)
714:                 {
715:                     ReturnValue = 1;
716:                     CurrentNodeAddress = record_preview.LeftAddress;
717:                 }
718:                 break;
```

```
719:
720:     case RIGHT:
721:         if(record_preview.RightAddress)
722:         {
723:             ReturnValue = 1;
724:             CurrentNodeAddress = record_preview.RightAddress;
725:         }
726:         break;
727:
728:     case SUB:
729:         if(record_preview.SubNodeAddress)
730:         {
731:             ReturnValue = 1;
732:             CurrentNodeAddress = record_preview.SubNodeAddress;
733:             CurrentFileLevel++;
734:         }
735:         break;
736:
737:     case PREVIOUS:
738:         if(record_preview.PreviousNodeAddress)
739:         {
740:             ReturnValue = 1;
741:             CurrentNodeAddress = record_preview.PreviousNodeAddress;
742:         }
743:         break;
744:
745:     case MOTHER:
746:         if(record_preview.MotherNodeAddress)
747:         {
748:             ReturnValue = 1;
749:             CurrentNodeAddress = record_preview.MotherNodeAddress;
750:             CurrentFileLevel--;
751:         }
752:         break;
753:
754:     case TOP:
755:         while(record_preview.MotherNodeAddress)
756:         {
757:             ReturnValue = 1;
758:             Go(MOTHER);
759:         }
760:
761:         Go(SUBTOP);
762:         CurrentFileLevel = 1;
763:         break;
764:
765:     case SUBTOP:
766:         while(record_preview.PreviousNodeAddress)
767:         {
768:             ReturnValue = 1;
769:             Go(PREVIOUS);
770:         }
771:         break;
772:
773:     case ADDRESS:
774:         if(where)
775:         {
776:             CurrentNodeAddress = where;
777:
778:             if(IndexCounter)
779:             {
780:                 int counter = 1;
781:
782:                 while(Go(MOTHER))
783:                     counter++;
784:
785:                 CurrentNodeAddress = where;
786:                 CurrentFileLevel = counter;
787:             }
788:
789:             ReturnValue = 1;
790:         }
791:         break;
792:     }
793: }
```

```
794:
795:     return ReturnValue;
796: }
797:
798: char Fbtree::DeleteNode(void)
799: {
800:     RecordPreview rpsave;
801:     long savenodeaddress = CurrentNodeAddress;
802:     long othernoteaddress = 0;
803:     char ReturnValue = 0;
804:
805:     if(Header.FirstNodeAddress && File->Handle > 0)
806:     {
807:         ReturnValue = PREVIOUS;
808:         OpenIndexFiles();
809:
810:         File->Seek(CurrentNodeAddress, soFromBeginning);
811:         File->Read(&rpsave, sizeof(RecordPreview));
812:
813:         void *buffer = AllocReadRecord(buffer, CurrentNodeAddress);
814:         RemoveAddressIndex(CurrentNodeAddress);
815:         delete[] buffer;
816:
817:         if(Go(PREVIOUS))
818:         {
819:             File->Seek(CurrentNodeAddress, soFromBeginning);
820:             File->Read(&record_preview, sizeof(RecordPreview));
821:
822:             if(!rpsave.LeftAddress || !rpsave.RightAddress)
823:             {
824:                 if(!rpsave.LeftAddress)
825:                 {
826:                     if(record_preview.LeftAddress == savenodeaddress)
827:                         record_preview.LeftAddress = rpsave.RightAddress;
828:                     else
829:                         record_preview.RightAddress = rpsave.RightAddress;
830:
831:                     File->Seek(CurrentNodeAddress, soFromBeginning);
832:                     File->Write(&record_preview, sizeof(RecordPreview));
833:
834:                     if(rpsave.RightAddress)
835:                         othernoteaddress = rpsave.RightAddress;
836:                 }
837:                 else
838:                 {
839:                     if(record_preview.LeftAddress == savenodeaddress)
840:                         record_preview.LeftAddress = rpsave.LeftAddress;
841:                     else
842:                         record_preview.RightAddress = rpsave.LeftAddress;
843:
844:                     File->Seek(CurrentNodeAddress, soFromBeginning);
845:                     File->Write(&record_preview, sizeof(RecordPreview));
846:
847:                     if(rpsave.LeftAddress)
848:                         othernoteaddress = rpsave.LeftAddress;
849:                 }
850:
851:                 if(othernoteaddress)
852:                 {
853:                     File->Seek(othernoteaddress, soFromBeginning);
854:                     File->Read(&record_preview, sizeof(RecordPreview));
855:                     record_preview.PreviousNodeAddress = rpsave.PreviousNodeAddress;
856:                     File->Seek(othernoteaddress, soFromBeginning);
857:                     File->Write(&record_preview, sizeof(RecordPreview));
858:                 }
859:             }
860:             else
861:             {
862:                 if(record_preview.LeftAddress == savenodeaddress)
863:                     record_preview.LeftAddress = rpsave.LeftAddress;
864:                 else
865:                     record_preview.RightAddress = rpsave.LeftAddress;
866:
867:                 File->Seek(CurrentNodeAddress, soFromBeginning);
868:                 File->Write(&record_preview, sizeof(RecordPreview));
```

```
869:
870:     File->Seek(rpsave.LeftAddress, soFromBeginning);
871:     File->Read(&record_preview, sizeof(RecordPreview));
872:     record_preview.PreviousNodeAddress = rpsave.PreviousNodeAddress;
873:     File->Seek(rpsave.LeftAddress, soFromBeginning);
874:     File->Write(&record_preview, sizeof(RecordPreview));
875:
876:     CurrentNodeAddress = rpsave.LeftAddress;
877:     while(Go(RIGHT));
878:     File->Seek(CurrentNodeAddress, soFromBeginning);
879:     File->Read(&record_preview, sizeof(RecordPreview));
880:
881:     record_preview.RightAddress = rpsave.RightAddress;
882:     File->Seek(CurrentNodeAddress, soFromBeginning);
883:     File->Write(&record_preview, sizeof(RecordPreview));
884:
885:     savenodeaddress = CurrentNodeAddress;
886:
887:     File->Seek(rpsave.RightAddress, soFromBeginning);
888:     File->Read(&record_preview, sizeof(RecordPreview));
889:     record_preview.PreviousNodeAddress = savenodeaddress;
890:     File->Seek(rpsave.RightAddress, soFromBeginning);
891:     File->Write(&record_preview, sizeof(RecordPreview));
892: }
893:
894: CurrentNodeAddress = rpsave.PreviousNodeAddress;
895: }
896: else
897: {
898:     if(rpsave.LeftAddress || rpsave.RightAddress)
899:     {
900:         if(!rpsave.LeftAddress || !rpsave.RightAddress)
901:         {
902:             if(!rpsave.LeftAddress)
903:             {
904:                 Go(RIGHT);
905:                 ReturnValue = RIGHT;
906:             }
907:             else
908:             {
909:                 Go(LEFT);
910:                 ReturnValue = LEFT;
911:             }
912:
913:             File->Seek(CurrentNodeAddress, soFromBeginning);
914:             File->Read(&record_preview, sizeof(RecordPreview));
915:             record_preview.PreviousNodeAddress = 0;
916:             record_preview.MotherNodeAddress = rpsave.MotherNodeAddress;
917:             File->Seek(CurrentNodeAddress, soFromBeginning);
918:             File->Write(&record_preview, sizeof(RecordPreview));
919:
920:             savenodeaddress = CurrentNodeAddress;
921:
922:             if(Go(MOTHER))
923:             {
924:                 File->Seek(CurrentNodeAddress, soFromBeginning);
925:                 File->Read(&record_preview, sizeof(RecordPreview));
926:                 record_preview.SubNodeAddress = savenodeaddress;
927:                 File->Seek(CurrentNodeAddress, soFromBeginning);
928:                 File->Write(&record_preview, sizeof(RecordPreview));
929:             }
930:             else
931:             {
932:                 Header.FirstNodeAddress = savenodeaddress;
933:                 WriteHeader();
934:             }
935:
936:             CurrentNodeAddress = savenodeaddress;
937:         }
938:     }
939:     else
940:     {
941:         if(Go(MOTHER))
942:         {
943:             File->Seek(CurrentNodeAddress, soFromBeginning);
944:             File->Read(&record_preview, sizeof(RecordPreview));
```

```
944:         record_preview.SubNodeAddress = rpsave.LeftAddress;
945:         File->Seek(CurrentNodeAddress, soFromBeginning);
946:         File->Write(&record_preview, sizeof(RecordPreview));
947:     }
948:     else
949:     {
950:         Header.FirstNodeAddress = rpsave.LeftAddress;
951:         WriteHeader();
952:     }
953:
954:     File->Seek(rpsave.LeftAddress, soFromBeginning);
955:     File->Read(&record_preview, sizeof(RecordPreview));
956:     record_preview.PreviousNodeAddress = rpsave.PreviousNodeAddress;
957:     record_preview.MotherNodeAddress = rpsave.MotherNodeAddress;
958:     File->Seek(rpsave.LeftAddress, soFromBeginning);
959:     File->Write(&record_preview, sizeof(RecordPreview));
960:
961:     CurrentNodeAddress = rpsave.LeftAddress;
962:     while(Go(RIGHT));
963:     File->Seek(CurrentNodeAddress, soFromBeginning);
964:     File->Read(&record_preview, sizeof(RecordPreview));
965:
966:     record_preview.RightAddress = rpsave.RightAddress;
967:     File->Seek(CurrentNodeAddress, soFromBeginning);
968:     File->Write(&record_preview, sizeof(RecordPreview));
969:
970:     savenodeaddress = CurrentNodeAddress;
971:
972:     File->Seek(rpsave.RightAddress, soFromBeginning);
973:     File->Read(&record_preview, sizeof(RecordPreview));
974:     record_preview.PreviousNodeAddress = savenodeaddress;
975:     File->Seek(rpsave.RightAddress, soFromBeginning);
976:     File->Write(&record_preview, sizeof(RecordPreview));
977:
978:     CurrentNodeAddress = rpsave.LeftAddress;
979: }
980: }
981: else
982: {
983:     if(Go(MOTHER))
984:     {
985:         ReturnValue = MOTHER;
986:         File->Seek(CurrentNodeAddress, soFromBeginning);
987:         File->Read(&record_preview, sizeof(RecordPreview));
988:         record_preview.SubNodeAddress = 0;
989:         File->Seek(CurrentNodeAddress, soFromBeginning);
990:         File->Write(&record_preview, sizeof(RecordPreview));
991:     }
992:     else
993:     {
994:         ReturnValue = NULL;
995:         Header.FirstNodeAddress = 0;
996:         WriteHeader();
997:     }
998: }
999: }
1000:
1001:     CloseIndexFiles();
1002: }
1003:
1004: return ReturnValue;
1005: }
1006:
1007: void Fbtree::ReplaceNode(void *nodestruct, int nodestructsize)
1008: {
1009:     RecordPreview rpsave;
1010:     long savenodeaddress;
1011:     char deletevalue;
1012:
1013:     if(Header.FirstNodeAddress && File->Handle > 0)
1014:     {
1015:         File->Seek(CurrentNodeAddress, soFromBeginning);
1016:         File->Read(&rpsave, sizeof(RecordPreview));
1017:
1018:         deletevalue = DeleteNode();
```

```

1019:         if(deletevalue != MOTHER)
1020:         {
1021:             Go(SUBTOP);
1022:             AddNode(nodestruct, nodestructsize, (char *)nodestruct +
rpsave.KeyStartAddress, rpsave.KeyLen, rpsave.CompareKind);
1023:         }
1024:         else
1025:             AddSubNode(nodestruct, nodestructsize, (char *)nodestruct +
rpsave.KeyStartAddress, rpsave.KeyLen, rpsave.CompareKind);
1026:
1027:         File->Seek(CurrentNodeAddress, soFromBeginning);
1028:         File->Read(&record_preview, sizeof(RecordPreview));
1029:         record_preview.SubNodeAddress = rpsave.SubNodeAddress;
1030:         File->Seek(CurrentNodeAddress, soFromBeginning);
1031:         File->Write(&record_preview, sizeof(RecordPreview));
1032:
1033:         if(rpsave.SubNodeAddress)
1034:         {
1035:             long counter = 0;
1036:             void *result;
1037:
1038:             savenodeaddress = CurrentNodeAddress;
1039:             Go(SUB);
1040:             do
1041:             {
1042:                 if(!counter)
1043:                 {
1044:                     result = FreeSearch(MAXFBTREELEVEL, NULL, NULL, NULL, NULL);
1045:                     counter++;
1046:                 }
1047:                 else
1048:                     result = FreeSearch(MAXFBTREELEVEL, NULL, NULL, NULL, NULL,
NEXT);
1049:
1050:                 if(result)
1051:                 {
1052:                     File->Seek(CurrentNodeAddress, soFromBeginning);
1053:                     File->Read(&record_preview, sizeof(RecordPreview));
1054:                     record_preview.MotherNodeAddress = savenodeaddress;
1055:                     File->Seek(CurrentNodeAddress, soFromBeginning);
1056:                     File->Write(&record_preview, sizeof(RecordPreview));
1057:                 }
1058:             }while(result);
1059:
1060:             Go(savenodeaddress);
1061:         }
1062:
1063:         if(deletevalue == MOTHER || (rpsave.MotherNodeAddress && (deletevalue == LEFT
|| deletevalue == RIGHT)))
1064:         {
1065:             Go(SUBTOP);
1066:
1067:             File->Seek(rpsave.MotherNodeAddress, soFromBeginning);
1068:             File->Read(&record_preview, sizeof(RecordPreview));
1069:             record_preview.SubNodeAddress = CurrentNodeAddress;
1070:             File->Seek(rpsave.MotherNodeAddress, soFromBeginning);
1071:             File->Write(&record_preview, sizeof(RecordPreview));
1072:         }
1073:     }
1074: }
1075:
1076: int Fbtree::MoveSubUp(void)
1077: {
1078:     //////////////////////////////////////
1079:     //NOTICE:
1080:     //When the sub node is became move up, it loses
1081:     //its left and right node addresses.
1082:     //If this occurred, ReturnValue sets to 1, otherwise sets to 0;
1083:
1084:     //If current node has no sub node, it will be deleted;
1085:     //////////////////////////////////////
1086:
1087:     int ReturnValue = 0;
1088:     RecordPreview subrpsave = {NULL};
1089:     long savenodeaddress = CurrentNodeAddress;

```

```

1090:     char *readnode = NULL;
1091:
1092:     if(Go(SUB))
1093:     {
1094:         File->Seek(CurrentNodeAddress, soFromBeginning);
1095:         File->Read(&subrpsave, sizeof(RecordPreview));
1096:         readnode = (char *)AllocReadRecord(readnode, CurrentNodeAddress);
1097:     }
1098:
1099:     Go(savenodeaddress);
1100:
1101:     DeleteNode();
1102:     if(subrpsave.MotherNodeAddress)
1103:     {
1104:         Go(SUBTOP);
1105:         AddNode(readnode, subrpsave.RecordLen, readnode + subrpsave.KeyStartAddress,
subrpsave.KeyLen, subrpsave.CompareKind);
1106:
1107:         File->Seek(CurrentNodeAddress, soFromBeginning);
1108:         File->Read(&record_preview, sizeof(RecordPreview));
1109:
1110:         record_preview.SubNodeAddress = subrpsave.SubNodeAddress;
1111:         File->Seek(CurrentNodeAddress, soFromBeginning);
1112:         File->Write(&record_preview, sizeof(RecordPreview));
1113:
1114:         savenodeaddress = CurrentNodeAddress;
1115:         if(Go(SUB))
1116:         {
1117:             File->Seek(CurrentNodeAddress, soFromBeginning);
1118:             File->Read(&record_preview, sizeof(RecordPreview));
1119:             record_preview.MotherNodeAddress = savenodeaddress;
1120:             File->Seek(CurrentNodeAddress, soFromBeginning);
1121:             File->Write(&record_preview, sizeof(RecordPreview));
1122:         }
1123:     }
1124:
1125:     if(readnode)
1126:         delete readnode;
1127:
1128:     Go(CurrentNodeAddress);
1129:
1130:     return ReturnValue;
1131: }
1132:
1133: char Fbtree::Optimize(void)
1134: {
1135:     char optimizefilename[128] = "OPTIMIZE.DAT";
1136:     char ReturnValue = 0;
1137:
1138:     bool del = _DeleteFile(optimizefilename, UseExecutionPath);
1139:
1140:     if(del == true && File->Handle > 0)
1141:     {
1142:         OptimizeLevel = 0;
1143:
1144:         Fbtree *opt_result = new Fbtree(optimizefilename, YES, CaseSensitive,
CompareFunction, fbtree_index, UseExecutionPath);
1145:
1146:         if(opt_result)
1147:         {
1148:             opt_result->OptimizeActive = YES;
1149:
1150:             opt_result->OpenIndexFiles();
1151:
1152:             Go(TOP);
1153:
1154:             if(OptProcess(opt_result))
1155:             {
1156:                 FbtreeHeader oldheader = Header;
1157:
1158:                 File->Size = sizeof(Header);
1159:                 File->Seek(sizeof(Header), soFromBeginning);
1160:                 opt_result->File->Seek(sizeof(Header), soFromBeginning);
1161:
1162:                 File->CopyFrom(opt_result->File, opt_result->File->Size -

```

```
        sizeof(Header));
1163:
1164:         File->Seek(0, soFromBeginning);
1165:         opt_result->File->Seek(0, soFromBeginning);
1166:
1167:         memmove(&Header, &opt_result->Header, (int)(&Header.UserActive[0] -
&Header.Header[0]));
1168:
1169:         StrCopy(Header.Security1, oldheader.Security1);
1170:         StrCopy(Header.Security2, oldheader.Security2);
1171:         StrCopy(Header.Provider, oldheader.Provider);
1172:         Header.Version = oldheader.Version;
1173:
1174:         WriteHeader();
1175:         CurrentNodeAddress = Header.FirstNodeAddress;
1176:         ReturnValue = 1;
1177:     }
1178:
1179:     opt_result->CloseIndexFiles();
1180:     delete opt_result;
1181:
1182:     _DeleteFile(optimizefilename, UseExecutionPath);
1183: }
1184:
1185:     GetNextNewNodeAddress();
1186: }
1187:
1188: if(ProgressBar)
1189:     ProgressBar->Position = ProgressBar->Min;
1190:
1191: return ReturnValue;
1192: }
1193:
1194: char Fbtree::OptProccess(Fbtree *opt_result)
1195: {
1196:     long i;
1197:     char ReturnValue = 0;
1198:     int address_handle;
1199:     long record_counter = 0, address;
1200:     char opt_filename[256];
1201:     char *readnode = NULL;
1202:
1203:     OptimizeLevel++;
1204:
1205:     if(UseExecutionPath)
1206:         sprintf(opt_filename, "%s\\address.%d", ExecutionPath, OptimizeLevel);
1207:     else
1208:         sprintf(opt_filename, "address.%d", OptimizeLevel);
1209:
1210:     TFileStream *AddressFile = new TFileStream(opt_filename, fmCreate);
1211:
1212:     if(AddressFile->Handle > 0)
1213:     {
1214:         ReturnValue = 1;
1215:
1216:         if(FreeSearch(MAXFBTREELEVEL, NULL, NULL, NULL, NULL))
1217:         {
1218:             AddressFile->Write(&CurrentNodeAddress, sizeof(long));
1219:             record_counter++;
1220:
1221:             while(FreeSearch(MAXFBTREELEVEL, NULL, NULL, NULL, NULL, NEXT))
1222:             {
1223:                 AddressFile->Write(&CurrentNodeAddress, sizeof(long));
1224:                 record_counter++;
1225:             }
1226:         }
1227:
1228:         if(ProgressBar && OptimizeLevel == 1)
1229:         {
1230:             ProgressBar->Min = 0;
1231:             ProgressBar->Max = record_counter;
1232:             ProgressBar->Position = 0;
1233:         }
1234:
1235:         if(record_counter > 2)
```

```
1236:     {
1237:         long random1 = 0, random2 = 0;
1238:         long address1, address2;
1239:         unsigned randomblock1, randomblock2, maxblock;
1240:
1241:         randomize();
1242:         maxblock = unsigned(record_counter / 32000);
1243:         for(i = 0; i < record_counter; i++)
1244:         {
1245:             randomblock1 = random(maxblock + 1);
1246:             randomblock2 = random(maxblock + 1);
1247:
1248:             while(!random1)
1249:                 random2 = random1 = randomblock1 == maxblock ?
random((unsigned)(record_counter % 32000)) : random(32000);
1250:
1251:             while(random1 == random2 || !random2)
1252:                 random2 = randomblock2 == maxblock ?
random((unsigned)(record_counter % 32000)) : random(32000);
1253:
1254:             AddressFile->Seek((randomblock1 * 32000 + random1) * sizeof(long),
soFromBeginning);
1255:             AddressFile->Read(&address1, sizeof(long));
1256:             AddressFile->Seek((randomblock2 * 32000 + random2) * sizeof(long),
soFromBeginning);
1257:             AddressFile->Read(&address2, sizeof(long));
1258:
1259:             AddressFile->Seek((randomblock1 * 32000 + random1) * sizeof(long),
soFromBeginning);
1260:             AddressFile->Write(&address2, sizeof(long));
1261:             AddressFile->Seek((randomblock2 * 32000 + random2) * sizeof(long),
soFromBeginning);
1262:             AddressFile->Write(&address1, sizeof(long));
1263:
1264:             random1 = random2 = 0;
1265:         }
1266:     }
1267:
1268:     AddressFile->Seek(0, soFromBeginning);
1269:
1270:     for(i = 0; i < record_counter; i++)
1271:     {
1272:         if(ProgressBar && OptimizeLevel == 1)
1273:         {
1274:             ProgressBar->Position = record_counter;
1275:
1276:             if(ProgressBar->Max > 100)
1277:             {
1278:                 if(!(record_counter%(ProgressBar->Max / 100)))
1279:                     Application->ProcessMessages();
1280:             }
1281:             else
1282:                 Application->ProcessMessages();
1283:         }
1284:
1285:         AddressFile->Read(&address, sizeof(long));
1286:         Go(address);
1287:
1288:         readnode = (char *)AllocReadRecord(readnode, CurrentNodeAddress);
1289:
1290:         if(!i)
1291:             opt_result->AddSubNode(readnode, record_preview.RecordLen, readnode
record_preview.KeyStartAddress, record_preview.KeyLen, record_preview.CompareKind);
1292:         else
1293:         {
1294:             opt_result->Go(SUBTOP);
1295:             opt_result->AddNode(readnode, record_preview.RecordLen, readnode +
record_preview.KeyStartAddress, record_preview.KeyLen, record_preview.CompareKind);
1296:         }
1297:
1298:         if(readnode)
1299:             delete[] readnode;
1300:
1301:         if(record_preview.SubNodeAddress)
1302:         {
```

```
1303:                Go(SUB);
1304:
1305:                ReturnValue = OptProccess(opt_result);
1306:                opt_result->Go(MOTHER);
1307:                Go(MOTHER);
1308:            }
1309:        }
1310:        delete AddressFile;
1311:
1312:        _DeleteFile(opt_filename, false/*UseExecutionPath*/);
1313:    }
1314:
1315:    OptimizeLevel--;
1316:
1317:    return ReturnValue;
1318: }
1319:
1320: int Fbtree::PreIndex(int index_number)
1321: {
1322:     int ReturnValue = index_number;
1323:
1324:     for(register i = 0; i < index_number; i++)
1325:         if(!strcmp(fbtree_index[index_number].Filename, fbtree_index[i].Filename) &&
Index[i])
1326:         {
1327:             ReturnValue = i;
1328:             break;
1329:         }
1330:
1331:     return ReturnValue;
1332: }
1333:
1334: void Fbtree::OpenIndexFiles(void)
1335: {
1336:     register i;
1337:
1338:     if(!OpenIndexFlag)
1339:         for(i = 0; i < IndexCounter; i++)
1340:             if((OptimizeActive == YES || CurrentFileLevel == fbtree_index[i].Level ||
(CurrentFileLevel >= fbtree_index[i].Level && fbtree_index[i].IncludeSubNodes)) &&
PreIndex(i) == i)
1341:             {
1342:                 if(OptimizeActive == YES)
1343:                     _DeleteFile(fbtree_index[i].Filename, UseExecutionPath);
1344:
1345:                 Index[i] = new Fbtree(fbtree_index[i].Filename, YES, CaseSensitive,
fbtree_index[i].Comp, NULL, UseExecutionPath);
1346:             }
1347:
1348:     OpenIndexFlag++;
1349: }
1350:
1351: void Fbtree::CloseIndexFiles(void)
1352: {
1353:     register i;
1354:
1355:     OpenIndexFlag--;
1356:
1357:     if(!OpenIndexFlag)
1358:         for(i = IndexCounter - 1; i >= 0; i--)
1359:             if(Index[i])
1360:             {
1361:                 delete Index[i];
1362:                 Index[i] = NULL;
1363:             }
1364: }
1365:
1366: void Fbtree::WriteIndex(void *nodestruct, unsigned long delete_address)
1367: {
1368:     void *result;
1369:     int PreIndex_number;
1370:
1371:     for(register i = 0; i < IndexCounter; i++)
1372:         if(CurrentFileLevel == fbtree_index[i].Level || (CurrentFileLevel >=
fbtree_index[i].Level && fbtree_index[i].IncludeSubNodes))
```

```
1373:     {
1374:         char ch = 1;
1375:         char addsub = 1;
1376:         char *buffer = new char[fbtree_index[i].Len + 1];
1377:
1378:         PreIndex_number = PreIndex(i);
1379:
1380:         memmove(buffer, (char *)nodestruct + fbtree_index[i].StartKey,
fbtree_index[i].Len);
1381:         buffer[fbtree_index[i].Len] = 0;
1382:
1383:         Index[PreIndex_number]->Go(TOP);
1384:
1385:         fbtree_currentindex_sign = fbtree_index[i].Sign;
1386:
1387:         result = Index[PreIndex_number]->BSearch(&ch);
1388:         if(!result)
1389:             Index[PreIndex_number]->AddNode(&ch, sizeof(char), &ch, sizeof(char)
fbtCHAR);
1390:         else
1391:             if(Index[PreIndex_number]->Go(SUB))
1392:             {
1393:                 result = Index[PreIndex_number]->BSearch(buffer);
1394:                 addsub = 0;
1395:             }
1396:             else
1397:                 result = NULL;
1398:
1399:             if(!result)
1400:             {
1401:                 if(addsub)
1402:                     Index[PreIndex_number]->AddSubNode(buffer, strlen(buffer),
buffer, fbtree_index[i].Len, fbtree_index[i].CompareKind);
1403:                 else
1404:                     Index[PreIndex_number]->AddNode(buffer, strlen(buffer), buffer,
fbtree_index[i].Len, fbtree_index[i].CompareKind);
1405:
1406:                     Index[PreIndex_number]->AddSubNode(&CurrentNodeAddress, sizeof(long)
&CurrentNodeAddress, sizeof(long), fbtLONG);
1407:                 }
1408:                 else
1409:                 {
1410:                     if(Index[PreIndex_number]->Go(SUB))
1411:                     {
1412:                         result = NULL;
1413:
1414:                         if(delete_address)
1415:                             result = Index[PreIndex_number]->BSearch(&delete_address);
1416:
1417:                         if(result)
1418:                         {
1419:                             if(Index[PreIndex_number]->DeleteNode() == MOTHER)
1420:
Index[PreIndex_number]->AddSubNode(&CurrentNodeAddress, sizeof(long),
&CurrentNodeAddress, sizeof(long), fbtLONG);
1421:                             else
1422:                                 Index[PreIndex_number]->AddNode(&CurrentNodeAddress,
sizeof(long), &CurrentNodeAddress, sizeof(long), fbtLONG);
1423:                             }
1424:                             else
1425:                                 Index[PreIndex_number]->AddNode(&CurrentNodeAddress,
sizeof(long), &CurrentNodeAddress, sizeof(long), fbtLONG);
1426:                             }
1427:                             else
1428:                                 Index[PreIndex_number]->AddSubNode(&CurrentNodeAddress,
sizeof(long), &CurrentNodeAddress, sizeof(long), fbtLONG);
1429:                             }
1430:
1431:                             delete[] buffer;
1432:                     }
1433: }
1434:
1435: void Fbtree::RemoveAddressIndex(unsigned long delete_address)
1436: {
1437:     void *result;
```

```
1438:     int PreIndex_number;
1439:     char ch = 0;
1440:
1441:     for(register i = 0; i < IndexCounter; i++)
1442:         if(CurrentFileLevel == fbtree_index[i].Level || (CurrentFileLevel >=
fbtree_index[i].Level && fbtree_index[i].IncludeSubNodes))
1443:         {
1444:             PreIndex_number = PreIndex(i);
1445:
1446:             Index[PreIndex_number]->Go(TOP);
1447:
1448:             fbtree_currentindex_sign = fbtree_index[i].Sign;
1449:
1450:             result = Index[PreIndex_number]->BSearch(&ch);
1451:
1452:             if(!result)
1453:                 Index[PreIndex_number]->AddNode(&ch, sizeof(char), &ch, sizeof(char)
fbtCHAR);
1454:
1455:             Index[PreIndex_number]->AddSubNode(&delete_address, sizeof(long),
&delete_address, sizeof(long), fbtLONG);
1456:         }
1457: }
1458:
1459: unsigned long Fbtree::GoIndexAddress(char *filename, void *searchbuffer, int
occurrence_number)
1460: {
1461:     unsigned long ReturnValue = 0;
1462:     char ch;
1463:
1464:     for(register i = 0; i < IndexCounter; i++)
1465:         if(!strcmp(fbtree_index[i].Filename, filename))
1466:         {
1467:             ch = 1;
1468:             Fbtree *index_file = new Fbtree(filename, YES, CaseSensitive,
fbtree_index[i].Comp, NULL, UseExecutionPath);
1469:
1470:             if(index_file)
1471:             {
1472:                 if(index_file->BSearch(&ch))
1473:                     if(index_file->Go(SUB))
1474:                     {
1475:                         if(index_file->BSearch(searchbuffer))
1476:                         {
1477:                             if(index_file->Go(SUB))
1478:                             {
1479:                                 void *result;
1480:                                 unsigned long find_address = 0;
1481:
1482:                                 result = index_file->FreeSearch(MAXFBTREELEVEL,
NULL, NULL, NULL, NULL);
1483:
1484:                                 for(register j = 1; j < occurrence_number; j++)
1485:                                     result =
index_file->FreeSearch(MAXFBTREELEVEL, NULL, NULL, NULL, NULL, NEXT);
1486:
1487:                                 if(result)
1488:                                 {
1489:                                     index_file->GetCurrentNode(&find_address);
1490:
1491:                                     ReturnValue = find_address;
1492:
1493:                                     index_file->Go(TOP);
1494:                                     ch = 0;
1495:
1496:                                     if(index_file->BSearch(&ch))
1497:                                         if(index_file->Go(SUB))
1498:
1499:                                     if(index_file->BSearch(&find_address))
1500:                                         ReturnValue = find_address =
DELETED_INDEX;
1501:
1502:                                     if(ReturnValue)
1503:                                         Go(find_address);
1504:                                 }
1505:                             }
1506:                         }
1507:                     }
1508:                 }
1509:             }
1510:         }
1511:     }
```

```
1504:         }
1505:     }
1506: }
1507:
1508:     delete index_file;
1509: }
1510: }
1511:
1512:     return ReturnValue;
1513: }
1514:
1515: void Fbtree::SetHeaderProviderVersion(char *Provider, float Version)
1516: {
1517:     StrLCopy(Header.Provider, Provider, 49);
1518:     Header.Version = Version;
1519:     WriteHeader();
1520: }
1521:
1522: bool Fbtree::CheckProviderVersion(char *Provider, float Version)
1523: {
1524:     bool ReturnValue = true;
1525:
1526:     if(StrComp(Header.Provider, Provider))
1527:         ReturnValue = false;
1528:
1529:     if(Header.Version != Version)
1530:         ReturnValue = false;
1531:
1532:     return ReturnValue;
1533: }
1534: //-----
1535:
1536: AnsiString __fastcall Fbtree::GetExecutionPath(void)
1537: {
1538:     return ExtractFileDir(ParamStr(0));
1539: }
1540: //-----
```

بسمه تعالی

مركز اطلاعات و مدارك علمي ايران

فرم ارزشيابي نهايي طرح پژوهش (شماره ۱۴-۱)

روي نرم

نام و نام خانوادگي داود: آبي رستم مهرمهراني

عنوان طرح پژوهش: بهينه سازي اطلاعات علمي و مهندسي (سندرجا) مجري: آبي رستم مهرمهراني

سمت و رتبه علمي داود: كفوشت علمي

تاريخ داوري: ۸۴/۱۲/۸

كده: ۷۸-۱۰

توضيحات و پيشنهادات	غير قابل تأييد	به صورت مشروط مورد تأييد	مورد تأييد	معياري ارزشيابي	
			✓	ميزان دستيابي به اهداف منظور شده در طرح	هدفها
			✓	تناسب تحليل انجام شده با هدفهاي طرح	
			✓	ميزان توجه و تعمق در آثار علمي	بررسي آثار علمي
			✓	مرتبط بودن منابع علمي استفاده شده با موضوع	روش انجام طرح
			✓	روش كلي برخورد با موضوع	
			✓	روشها و تكنيكهاي به كار گرفته شده	نتايج
			✓	دستاوردهاي طرح	
			✓	مستدل و علمي بودن نتيجه گيري	كيفيت
			✓	كيفيت تحليل داده ها	
			✓	كيفيت تدوين گزارش	
			✓	تناسب هزينه و زمان صرف شده	هزينه زمان

امضاء

لطفاً موارد پشت صفحه را نيز تكميل فرماييد.

سایر پیشنهادات

طرح آموزشی پیشنهادی: اعلامیه علمی است و ذهنی است و مورد بررسی قرار گرفت
 و در مجموع از نظر اشکالات مورد تأیید است. امیر است. شیخ طرح در این
 زمینه د. جاسد شنی کشور صفه واقع کرد

بند

مهر امیر

ل. ک. ک. ک.

به طور کلی (با فرض رفع نواقص در صورت وجود)

از ۱۰۰ به آن اختصاص می‌دهم

طرح مورد تأیید است و نمره ۹۵

طرح غیر قابل تأیید است

در صورت نبود جلسه داوری، زمان صرف شده برای ارزیابی طرح را ذکر نمایید: ساعت

بسمه تعالی
مرکز اطلاعات و مدارک علمی ایران
پرسشنامه طرح پژوهش

(۱) عنوان طرح:

بهینه سازی اطلاعات علمی شیمی و مهندسی شیمی ایران (سمینارها)

(۲) نوع طرح: ☐ بنیادی ☐ کاربردی ☒ توسعه‌ای

(۳) مشخصات پیشنهاد دهنده:

(۱-۳) نام و نام خانوادگی: تقی رجبی

(۲-۳) شغل و عنوان محل کار: عضو هیات علمی - گروه پژوهشی اصطلاحنامه - مرکز اطلاعات و مدارک علمی ایران

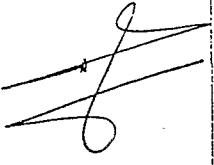
(۳-۳) نشانی محل کار: تهران - خ انقلاب - بعد از چهار راه فلسطین - شماره ۱۱۸۸ - صندوق پستی ۱۳۱۸۵ - ۱۳۷۱

صندوق صوتی: ۶۴۹۴۹۵۴ + ۸ + ۱۳۷
مزد: ۹۳۷۰۵۶

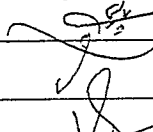
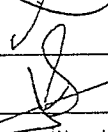
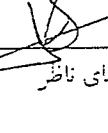
پست الکترونیک: rajabi@irandoc.ac.ir Homepage: <http://www.irandoc.ac.ir/rg/staff-all/rajabi.htm>

(۴-۳) تلفن تماس: محل کار: (۲۰۳) ۶۴۶۲۲۲۳ تاریخ: ۷۹/۵/۳

امضای پیشنهاد دهنده:



(۴) مشخصات همکاران اصلی:

ردیف	نام و نام خانوادگی	میزان تحصیلات	شغل و عنوان محل کار	زمت همکاری	امضا
۱	حسین غریبی	دکتری	استاد دانشگاه تربیت مدرس - رئیس مرکز	نظارت	
۲	امیر پارس اصفهانی	مهندسی کامپیوتر	رئیس اداره خدمات ماشینی مرکز	برنامه نویسی	
۳	صفا علی عسگری	فوق لیسانس	عضو هیات علمی دانشگاه	مشاوره، اجرا	
۴	مهرداد نوروزی	فوق دیپلم کامپیوتر	کارشناس بخش خدمات ماشینی مرکز	برنامه نویسی	

امضای ناظر:



تأیید ناظر (در صورتی که از طرف مرکز برای تدوین پرسشنامه ناظر تعیین شده باشد، این قسمت باید توسط وی تکمیل و امضا شود):

اینجانب

تأیید تعیین شده برای تدوین این پرسشنامه، مفاد آن را از لحاظ صحیحی با رضایت

تکمیل پرسشنامه طرحی پژوهش، روش پژوهش علمی و مبانی علمی موضوع پژوهش تأیید می‌کنم. (زمان صرف شده: ساعت)

تاریخ:

• نوبت طرح در شورا: اول ☐ دوم ☐ تاریخ طرح در شورا: •

• نتیجه بررسی:

برای انجام طرح پژوهش قرارداد ☐ عقد شود ☐ نشود

نام و نام خانوادگی ناظر برای انجام طرح (در صورت تغییر یا تعیین):

توجه: لطفاً در صورت امکان، فرم را با پست و در غیر این صورت با خط خنثی و رنگ مشکی تکمیل فرمایید.

لطفاً در این قسمت چیزی ننویسید

در این پژوهش مجموعه اطلاعات علمی شیمی و مهندسی شیمی که در سمینارهای تخصصی شیمی و مهندسی شیمی ایران ارائه شده اند و در بانک اطلاعات جامع مرکز موجودند، مورد باز پردازش تخصصی قرار خواهند گرفت. همچنین کاربرگه های ثبت اطلاعات شیمی در انواع مدارک موجود در مرکز طراحی و پیشنهاد خواهند شد و نیز برنامه نویسی به منظور تجهیز نرم افزار CDS جهت بازیابی اطلاعات شیمی با استفاده از نمایه نامه های تخصصی و همچنین طراحی نرم افزار بانک اطلاعاتی سازگار با CDS (با قابلیت انتقال داده ها از/ به CDS) صورت خواهد گرفت و پایگاه تخصصی اطلاعات شیمی و مهندسی شیمی ایران ایجاد خواهد شد.

اطلاعات پردازش شده، فهرستهای تخصصی و واژه های یکدست شده در این طرح می توانند بصورت وسیعی توسط دانشجویان، اساتید، محققان، رساله نویسان، نمایه سازان، مترجمان و فرهنگ نویسان مورد استفاده قرار گیرند و نیازهای اطلاعاتی محققان را در بخش صنعت و دانشگاه حداقل در بخش پژوهشهای داخل کشور (در حوزه های مختلف شیمی و مهندسی شیمی) مرتفع ساخته و از دوباره کاری جلوگیری نمایند. همچنین واژگان یکدست شده در ساخت اصطلاحنامه شیمی مورد استفاده قرار خواهد گرفت. کاربرگه های طراحی شده، هم به نمایه سازان و هم به محققان در ثبت استاندارد اطلاعات کمک شایانی خواهند نمود و نیز پردازش تخصصی اطلاعات را بصورت سهولتری عملی خواهند کرد و در واقع گامی در جهت عملیاتی شدن فرآیند پردازش اطلاعات شیمی و مهندسی شیمی خواهند بود. نرم افزار طراحی شده به نمایه ساز و ماشین نویس امکان ورود اطلاعات به شکل صحیح را می دهد و همچنین کاربر را قادر می سازد با استفاده از نمایه نامه های مورد نظر خود به جستجوی اطلاعات بپردازد. پایگاه تخصصی اطلاعات شیمی و مهندسی شیمی ضمن حفظ و اشاعه اطلاعات پردازش شده شیمی، به کاربران موضوعی امکان جستجوی اطلاعات شیمی با استفاده از نمایه نامه های تخصصی و نیز امکان مشاهده علائم و صور مورد استفاده در متون شیمی را می دهد. در این پایگاه، اطلاعات پردازش شده در طرح پیشین یعنی پایان نامه های شیمی و مهندسی شیمی در کنار اطلاعاتی که در این طرح پردازش خواهند شد قرار خواهند گرفت و نیز امکان روز آمد سازی و توسعه برای سایر انواع مدارک شیمی فراهم خواهد آمد. الگوهای بر آمده از این طرح در جهت عملیاتی نمودن فرآیند پردازش اطلاعات شیمی و تغییر و اصلاح در بخش مدیریت اسناد و اطلاعات مورد استفاده قرار خواهند گرفت، نیز مجموعه اطلاعات پردازش شده، آماده تحلیل جهت ارائه وضعیت موجود و مقایسه آن با وضعیت مطلوب در تحقیقات شیمی کشور خواهند بود.

۶) شرح مساله یا موضوع پژوهش

ایجاد نشریات و پایگاههای موضوعی شیمی از گامهای بسیار مؤثر در اطلاع رسانی علوم پایه است. تولید نمایه نامه های تخصصی، یکدست نمودن واژگان و نیز استاندارد کردن ذخیره و بازیابی و شیوه های ثبت اطلاعات شیمی از ضروریات اینکار است. پس از تدوین کتاب اطلاعات علمی شیمی و مهندسی شیمی ایران جلد اول پایان نامه ها، هم اکنون ضروری به نظر می رسد مجموعه اطلاعات شیمی از جمله سمینارهای شیمی نیز مورد پردازش و باز تولید تخصصی قرار گیرند تا نیل به اهداف اطلاع رسانی شیمی عملی تر گردد. تجربیات طرح پیشین و توفیقات بدست آمده، در این طرح مورد استفاده قرار خواهد گرفت. در این پژوهش مجموعه اطلاعات علمی شیمی و مهندسی شیمی که در سمینارهای تخصصی شیمی و مهندسی شیمی ایران ارائه شده اند و در بانک اطلاعات جامع مرکز موجودند مورد باز پردازش تخصصی قرار خواهند گرفت. همچنین کاربرگه ثبت اطلاعات شیمی در انواع مدارک موجود در مرکز طراحی و پیشنهاد خواهد شد و نیز برنامه نویسی به منظور تجهیز نرم افزار CDS جهت قابلیت بازیابی اطلاعات شیمی با نمایه نامه های تخصصی صورت خواهد گرفت. و پایگاه تخصصی اطلاعات شیمی و مهندسی شیمی ایران ایجاد خواهد شد. همچنین نرم افزار بانک اطلاعاتی سازگار با CDS دارای قابلیت ورود، نمایش و بازیابی اطلاعات شیمی طراحی خواهد شد.

۷) هدف (های) پژوهش

۱ - کتاب اطلاعات علمی شیمی و مهندسی شیمی (سمینارها) شامل : الف - متن اصلی حاوی : عنوان ، کلید واژه و اطلاعات مدرک شناختی : هر یک ویرایش شده ، معادل گذاری شده ، تصحیح شده ، استاندارد شده ، یکسان سازی شده و نمایه سازی شده . ب - فهرستهای تخصصی : موضوعی ، اسامی ترکیبات شیمیایی ، فرمولهای بسته شیمیایی ، کلید واژه های اصلاح شده و معادل یافته (استاندارد کردن ، ارجاع دهی ، حرف نگاری ، کورته نوشت ها ، فارسی سازی و ...) و واژگان یکدست شده هر یک با فهرست الفبایی انگلیسی به فارسی و فارسی به انگلیسی ۲ - نرم افزار تقریب شده CDS با قابلیت بازیابی اطلاعات با استفاده از نمایه نامه های تخصصی شیمی (فرمولهای بسته شیمیایی و اسامی ترکیبات شیمیایی) ۳ - کاربرد ویژه جهت ثبت اطلاعات شیمی بصورت استاندارد در انواع مدارک ۴ - ایجاد پایگاه تخصصی اطلاعات علمی شیمی و مهندسی شیمی ایران ۵ - نرم افزار بانک اطلاعاتی ویژه اطلاعات شیمی و سازگار با CDS (دارای قابلیت ورود ، ذخیره ، نمایش و بازیابی اطلاعات موجود در مدارک شیمی مانند فرمولهای بسته شیمیایی ، ساختارهای گسترده شیمیایی ، اسامی ترکیبات شیمیایی ، واکنشهای شیمیایی و علائم ریاضی مورد استفاده در متون علمی شیمی) علاوه بر سایر ویژگیهای

۸) قلمرو پژوهش

وسعت های وابسته

اطلاعات علمی شیمی و مهندسی شیمی ارائه شده (در سمینارهای شیمی و مهندسی شیمی برگزار شده در ایران از آغاز تا کنون) موجود در بانک اطلاعات جامع مرکز .

تعداد رکوردها : در حدود ۲۵۰۰ رکورد

زمان : از سال ۶۳ تا سال ۷۴

مکان : کشور ایران

۹) ضرورت انجام پژوهش / اهمیت پژوهش

۱- اطلاعات موجود در بانک اطلاعات جامع مرکز در حوزه شیمی و مهندسی شیمی توسط متخصصان غیر موضوعی و با روشها و ابزار ناکافی و دارای اشکال نمایه شده اند. در گزارش طرح پژوهشی پیشین ، بصورت مفصل و مشروح به مشکلات عدیده موجود در بازیابی این اطلاعات اشاره کرده ایم و ویژگیهای خاص اصطلاحات شیمی و اسامی ترکیبات شیمیایی و نحوه رفتار با آنها را در ایجاد پایگاه تخصصی شیمی و نمایه نامه های تخصصی شیمی بر شمرده ایم. هم اکنون که روشهای صحیح نمایه سازی و چکیده نویسی متون شیمی در بخش مدیریت اسناد و اطلاعات مرکز عملیاتی نشده است ، ادامه این روش حداقل تا تدوین جلد سوم یعنی طرحهای پژوهشی شیمی و مهندسی شیمی ضروری است تا پس از آن ، الگوهای بر آمده از این طرح ها در قالب طرحی مجزا با همکاری گروههای پژوهشی مدیریت سیستمهای اطلاعاتی ، تکنولوژی اطلاعات و کتابداری و اطلاع رسانی ، بصورت الگویی عملیاتی در آید و در انتظار بودجه ، فضا ، نیروی انسانی ، آموزش و پشتیبانی بماند تا روزی به اجرا در آید.

۲- فهرستهای تخصصی و واژه های یکدست شده در این طرح می توانند بصورت وسیعی توسط دانشجویان ، اساتید ، محققان ، رساله نویسان ، نمایه سازان ، مترجمان و فرهنگ نویسان مورد استفاده قرار گیرند و نیازهای اطلاعاتی محققان را در بخش صنعت و دانشگاه حداقل در بخش پژوهشهای داخل کشور (در حوزه های مختلف شیمی و مهندسی شیمی) مرتفع ساخته واز دوباره کاری جلوگیری نمایند.

۳- اگر قرار است ارتباط صنعت و دانشگاه عملی شود ، نقش پایگاه های اطلاعاتی موضوعی بسیار حائز اهمیت است.

۴- مدارک مورد نمایه سازی در سطح تحصیلات تکمیلی هستند بنابراین واژه های یکدست شده و استخراج شده از این مجموعه سنگ بنای تدوین اصطلاحنامه شیمی فارسی است. طرحی که هم اکنون در گروه پژوهشی اصطلاحشناسی در حال اجرا است.

۵- پس از ایجاد بانکهای موضوعی شیمی، میتوان به تحلیل اطلاعات آنها پرداخت و سمت و سوی فعالیتهای پژوهشی شیمی در کشور را مشخص و جهت دهی کرد و نیز به کنترل کیفیت و ارزیابی فرآیند پردازش اطلاعات پرداخت.

حرف نگاری (Transliteration): نوشتن حروف زبان مبدأ با حروف زبان مقصد بر اساس تلفظ حروف در زبان مبدأ.
 آوانگاری (Transcription): نوشتن آواهای زبان با حروف بین‌المللی آوانگاری.

ایوپاک (IUPAC): International Union Of Pure And Applied Chemistry

نمایه نامه فرمولی: Formula Index..

شماره های ثبت شیمیایی (Chemical Registry Numbers): بالغ بر ۲۴/۰۰۰/۰۰۰ ماده شیمیایی ثبت شده وجود دارند که هر یک از آنها دارای شماره ثبت منحصر به فردی می باشند که این شماره ها دارای فرمت خاص و معنی و مفهوم ویژه ای در ارتباط با ساختار آن ماده شیمیایی هستند. مانند: [71-43-2] که شماره ثبت شیمیایی «بنزن» است.

فرمول بسته (Close Formula): نمایشی از یک ماده شیمیایی که نشان دهنده تعداد و نوع عناصر به کار رفته در آن

ترکیب می باشد. مانند: $C_6H_{12}O_2$ و $K[PtCl_3(C_2H_4)]$

ساختار گسترده (Expand Structure): نمایشی از یک ترکیب شیمیایی که نشان دهنده نحوه قرار گرفتن و نوع پیوند

شیمیایی بین اتمها در آن ترکیب است. مانند:



واکنش شیمیایی (Chemical Reaction): فرآیندی که طی آن از ترکیب مواد اولیه تحت شرایط ویژه ای محصولات

واکنش بدست می آید. مانند:



در این طرح پنج بخش اصلی پیش بینی شده است:

- ۱ - تدوین کتاب اطلاعات علمی شیمی و مهندسی شیمی ایران (سمینارها).
- ۲ - تهیه کاربرگه استاندارد برای ثبت اطلاعات علمی شیمی در انواع مدارک.
- ۳ - تقویت نرم افزار CDS (WIN/ISIS) به منظور توانایی در بازیابی اطلاعات شیمیایی با استفاده از نمایه نامه های خاص شیمی. (فرمولهای بسته شیمیایی و اسامی ترکیبات شیمیایی)
- ۴ - ایجاد پایگاه اطلاعات علمی شیمی و مهندسی شیمی.
- ۵ - طراحی نرم افزار بانک اطلاعاتی با قابلیت ذخیره، نمایش و بازیابی ساختارها و علائم خاص مورد استفاده در متون شیمی.

در مورد اول مباحث اصطلاح شناسی (ترمینولوژی) موضوعی و فنون و روشهای نمایه سازی و چکیده نویسی بویژه در بخش شیمی که حاصل مطالعات و تجربیات سازمان چکیده نویسی شیمی است و در بخش دوم علاوه بر این مطالعات و تجربیات، استانداردهای مارک آمریکا و امکان ویژه سازی آنها و در بخش سوم قابلیتهای برنامه نویسی نرم افزار CDS (WIN/ISIS) و نیز زبان برنامه نویسی C و SQL میتواند بعنوان پشتوانه های نظری طرح در نظر گرفته شوند. همچنین قواعد و دستورالعملهای سازمان ایوپاک برای نامگذاری ترکیبات شیمیایی آلی و معدنی از دیگر پشتوانه های نظری کار می باشد. در خصوص نمایه نامه فرمولی قاعده Hill به منظور فهرست کردن فرمولهای بسته ترکیبات شیمیایی اساس برنامه نویسی کامپیوتری قرار می گیرد.

مجموعه های چکیده نامه های شیمی (chemical abstract) از سال ۱۹۰۷ انتشار یافته اند. هم اکنون فعالیتهای این سازمان بسیار گسترده است و شرح و بسط آن در اینجا مقدور نیست. علاقه مندان می توانند در مراجع زیر به تفصیل با فعالیتهای این سازمان از آغاز تا کنون آشنا شوند.

۱ - آشنایی با chemical abstract و چگونگی استفاده از آن، مصطفی سید صادق، نشریه فنی مرکز مدارک علمی، دوره چهارم، شماره ۱ و ۲ تهران ۱۳۵۵.

۲ - تاثیر زبان طبیعی در نمایه سازی و بازیابی علوم، بها تا چاپاریا، ترجمه و تلخیص مصطفی سید صادق، نشریه فنی مرکز مدارک علمی، دوره چهارم، شماره ۱ و ۲، تهران ۱۳۵۵

۳ - راهنمای استفاده از چکیده نامه های شیمی، انجمن شیمی آمریکا (ویرایش ۱۹۸۴)، ترجمه مسعود حسن پور، مرکز نشر دانشگاهی، چاپ اول ۱۳۷۵

4 - Chemical Abstract Service: vista for the 1990s, Robert J. Massie, Chemical Abstracts Service, Columbus, ohio, USA, farther advances in chemical information, The Royal Society Of Chemistry 1994, H. collier

همچنین سرویسهای اطلاع رسانی بسیاری در دنیا به پردازش و تولید پایگاه ها و مجموعه های چاپی و نرم افزاری در زمینه های اصلی و فرعی دانش شیمی پرداخته اند که شرح مبسوط آنها را در مراجع زیر می توان دید.

1 - Information Sources in Chemistry, Fourth edition, R.T. Bottle & J.F.B. Rowland, Bowker, 1993, chapter three

2 - Chemical Information, A Practical Guide To Utilization, Yechezkel Wolman, JOHAN Wiley & Sons, 1988, Bulestian, Chemical Abstract

نیز مرور دو سایت زیر در آشنایی با تولیدات Chemical Abstract و Bulestian بسیار مفید است:

1) www.cas.org

2) www.chemweb.com

همچنین در زمینه نرم افزاری، نرم افزارهای بسیاری تولید شده اند که به کار بر امکان می دهند به راحتی به رسم انواع و اقسام ساختارهای شیمیایی دو بعدی و سه بعدی پردازد، اسامی ترکیبات شیمیایی و علائم خاص در نامگذاری را تایپ نماید، واکنشهای شیمیایی را ترسیم کند، با دادن نام ترکیبات شیمیایی (ایریاک) ساختار گسترده مونوکولی را بدست آورد و بالعکس و بسیاری از قابلیتهای جالب دیگر. چند نمونه از معروفترین این نرم افزارها عبارتند از:

1) isis-draw 2) chemoffice 3) chem - x 4) CAS CD's 5) hyperchem 6) chemSketch

اما در زبان فارسی نگارنده این سطور تا کنون نتوانسته است اطلاعات علمی شیمی پردازش شده ای را بیابد که در آن الگوهای نمایه سازی تخصصی شیمی مورد استفاده واقع شده باشد. همچنین حتی هیچ الگوی نظری در این مورد یافت نشده است. این در حالی است که فعالیتهای CAS از سال ۱۹۰۷ آغاز شده بود. شاید این عقب افتادگی به این دلیل بوده است که تولید اطلاعات داخلی در حدی نبوده است که چنین نیازی را ایجاد کند ولی حتی نبودن یک الگوی نظری برای پردازش تخصصی اطلاعات علمی شیمی و نمایه نامه های مربوطه و ثبت و ذخیره و بازیابی اطلاعات شیمی، تعجب بر انگیز است، و سوالی است که باید آن را از اطلاع رسانان و کتابداران و نیز شیمی دانان این دیار پرسید. تنها کار انجام شده در این زمینه کتاب اطلاعات علمی شیمی و مهندسی شیمی (جلد اول پایان نامه ها) است که از طرف مرکز اطلاعات و مدارک علمی ایران به چاپ رسیده است.

(۱۳) فرضیه‌ها / سوالهای پژوهش

- (۱) چرا باز یافت اطلاعات علمی شیمی در بانک جامع اطلاعات مرکز به نحو مطلوب ممکن نیست؟
 - ۱-۱ - آیا کاربرگه های موجود ثبت اطلاعات شیمی در انواع مدارک مناسب و استاندارد است؟
 - ۱-۲ - آیا نرم افزار CDS تحت داس یا ویندوز به شکل فعلی برای نمایه سازی و ذخیره سازی و بازیافت اطلاعات علمی شیمی در انواع مدارک مناسب و استاندارد است؟
 - ۱-۳ - آیا نمایه سازان غیر موضوعی توانایی انجام نمایه سازی مدارک شیمی را دارند؟
 - ۱-۴ - آیا منابع موجود مانند اصطلاحنامه ها و فرهنگهای شیمی برای نمایه سازی مدارک شیمی که در سطح تحصیلات تکمیلی هستند کافی است؟
 - ۱-۵ - آیا ایجاد نکردن پایگاه اطلاعات علمی شیمی و مهندسی شیمی در یک مرکز اطلاعات و مدارک علمی و فنی در سطح ملی، روشی علمی است؟
- (۲) روال موجود نمایه سازی و ذخیره سازی و بازیافت اطلاعات علمی شیمی در مرکز چگونه است؟
- (۳) روال مطلوب در نمایه سازی و ذخیره سازی و بازیافت اطلاعات علمی شیمی در مرکز چگونه است؟
- (۴) چگونه می توان کاربرگه استاندارد ثبت اطلاعات شیمی را برای انواع مدارک تهیه کرد؟
- (۵) چگونه می توان نرم افزار CDS را تقویت کرد و یا حتی نرم افزار دیگری را انتخاب و آنرا تجویز نمود به نحوی که بتواند ساختارهای شیمیایی را نمایش دهد و در بازیابی با نمایه نامه های تخصصی مانند نمایه نامه فرمولی و اسامی ترکیبات شیمیایی و یا ساختارهای حلقوی و یا شماره های ثبت شیمیایی موفق باشد؟
- (۶) آیا می توان کتابی تهیه نمود که در آن مجموعه سمینارهای شیمی و مهندسی شیمی ایران بصورت تخصصی پردازش شده باشد؟
- (۷) آیا می توان پایگاه اطلاعات علمی شیمی و مهندسی شیمی ایجاد کرد به نحوی که قابلیت بازیابی با نمایه نامه های تخصصی را داشته باشد؟
- (۸) آیا می توان با استفاده از شماره های ثبت شیمیایی (Chemical Registry numbers) بین اطلاعات علمی شیمی داخلی و خارجی مانند CD های CAS اتصال و ارتباط برقرار نمود؟

(۱۴) محدودیتها

- (۱) عدم حمایت و پشتیبانی لازم، از اجرای چنین طرحهایی که دقیقاً در راستای اهداف و سیاستهای اصلی مرکز اطلاعات و مدارک ایران قرار دارد.
- (۲) عدم همخوانی کامل بین نرم افزار CDS و زرنگار، توضیح اینکه در آخرین مرحله از کار هنگامیکه اطلاعات از محیط CDS به محیط زرنگار تبدیل می شود (به منظور صفحه آرایی و چاپ) حجم نامحاشیه ها و بهم ریختگیها بسیار زیاد است بخصوص در مورد اعداد و علائم و ترکیبات شیمیایی که نیازمند ویرایش مجدد می شوند و دقت و ارزش اطلاعات را به خصوص در مورد گزارشهای کمی بسیار پایین می آورد.
- (۳) کامل نبودن مجموعه های گرد آوری شده در مرکز، از ارزش اطلاعاتی مجموعه های نهایی ارائه شده می کاهد.
- (۴) عدم سهولت دسترسی به اصل مدارک در کیفیت کار تاثیر منفی خواهد گذاشت.
- (۵) گران بودن مجموعه های CD های CAS و عدم خریداری آنها از طرف مرکز در بخش هایی از طرح به خصوص در مورد بررسی اتصال بین پایگاه های اطلاعات تاثیر منفی خواهد داشت.
- (۶) عدم تبلیغ مناسب از طرف مرکز برای تولیدات طرح، که به لحاظ بازخوردی در روند اجرای طرح تاثیر منفی خواهد گذاشت.
- (۷) عدم تخصیص بودجه مناسب به طرح در بکار گرفتن نیروی انسانی متخصص در روند اجرای طرح تاثیر منفی خواهد گذاشت.
- (۸) وجود ویرسهای کامپیوتری در شبکه مرکز و همچنین قطع مکرر شبکه اینترنت در روند اجرای طرح تاثیر منفی خواهد گذاشت.
- (۹) عدم دسترسی مجری طرح به شبکه مرکز و خط اینترنت از منزل، در روند اجرای طرح تاثیر منفی خواهد گذاشت.

- ۱-۱۵) از نظر زمان: تاریخی ☒ پیمایشی ☒ تجربی ☒
 ۲-۱۵) از نظر هدف: توصیفی ☒ تطبیقی ☒ ارزشیابی ☒
 ۳-۱۵) از نظر روش گردآوری اطلاعات: مشاهده ☐ آزمایش ☒ پرسشنامه ☐ مصاحبه ☐ مطالعه ☐
 سایر (نقشه بید):

۴-۱۵) از نظر جامعه و نمونه:

- اطلاعات کلیه آحاد جامعه به روش سرشماری گردآوری می شود ☒ نمونه گیری می شود ☐
 در صورت نمونه گیری شیوه آن چیست؟
 تصادفی ساده ☐ تصادفی با طبقه بندی ☐ خوشه ای ☐ تصادفی سیستماتیک ☐ انتخابی ☐
 سایر (نقشه بید):

۵-۱۵) روش تحلیل اطلاعات و تکنیکهایی که استفاده می شود:

روش تحلیل اطلاعات کیفی است ، این روش کلی بوده و فعلا در این طرح در سطح پردازش اطلاعات ، ارائه قالبهای استاندارد و بهبود روشهای قبلی محدود می شود. مجری طرح با استفاده از تجربیات پیشین و به مدد دانش اطلاع رسانی ، اصطلاح شناسی و شیمی و با استفاده از قابلیت های نرم افزار CDS و به کمک منابع و مراجع معتبر در حوزه دانش شیمی و مهندسی شیمی و نیز با کمک متخصص برنامه نویسی کامپیوتر به بهبود روشها و ارائه قالبهای استاندارد و بازپردازش تخصصی اطلاعات اقدام می نماید.

۶-۱۵) هرگونه توضیح دیگری که در باره روش پژوهش لازم می دانید، اضافه کنید:

این پژوهش چند منظوره است . بعنوان مثال در یک مرحله از کار که به تولید نمایه نامه فرمونی اقدام می شود چنین فلوجارتی مورد عمل قرار می گیرد:

۱-۱۶) از لحاظ محتوا

xx راهنماهای ایجاد فهرست های تخصصی شیمی مانند:

- واژگان یکدست شده و معادل یافته
- نمایه نامه اسامی ترکیبات شیمیایی به فارسی و لاتین
- نمایه نامه فرمولی
- نمایه نامه موضوعی یکدست شده
- و

xx راهنماهای ایجاد کاربرگه های استاندارد

xx راهنماهای ایجاد و یا تقویت نرم افزارهایی مانند CDS جهت قابلیت های مورد نظر در پایگاه های اطلاعات شیمی

۲-۱۶) از لحاظ قالب

- کتاب اطلاعات علمی شیمی و مهندسی شیمی ایران (سمینارها)
- گزارش نهایی طرح
- بروشور به زبان فارسی
- پوستر تبلیغاتی طرح
- نرم افزار: کد اصلی، فایل اجرایی، راهنمای کاربران، راهنمای مستندات نرم افزار

کاربرگه ثبت اطلاعات شیمی در انواع مدارک زیر:

- الف - پایان نامه ها
- ب - طرحهای پژوهشی
- ج - سمینارها
- د - مقاله در مجله
- ه - گزارش دولتی

شماره مرحله	عنوان کلی مرحله	فعالیت‌های اصلی هر مرحله
۱	استخراج اطلاعات	استخراج کلیه اطلاعات سمینارهای شیمی و مهندسی شیمی از بانک جامع با فرمت مایکرو سافت iso
۲	تهیه و تدوین کتاب اطلاعات علمی شیمی و مهندسی شیمی ایران (سمینارها)	الف - ویرایش و نمایه سازی مجدد (تصحیح، حذف و افزایش کلید واژه‌ها) ب - یکسان سازی و استاندارد سازی (عناوین، کلید واژه‌ها و اسامی ترکیبات شیمیایی) ج - معادل یابی فارسی و لاتین د - ارجاع دهی و استخراج کوتاه نوشته‌ها هـ - استخراج اسامی ترکیبات شیمیایی و - تصحیح اسامی ترکیبات شیمیایی و ورود اطلاعات ز - فرمول دهی به اسامی ترکیبات شیمیایی و ورود اطلاعات ح - ایجاد نمایه نامه فرمولی ط - ایجاد نمایه نامه اسامی ترکیبات شیمیایی ی - تبدیل فرمت کلیه اطلاعات پردازش شده از cds به زرنگار و یا Word ک - صفحه آرایی و ویرایش نهایی ل - طراحی روی جلد م - چاپ
۳	طراحی کاربرگه	الف - یافتن فیلدهای مورد نیاز در کاربرگه‌های مربوط به انواع مدارک ب - افزایش این فیلدها به کاربرگه‌ها و تهیه فرمت نهایی بصورت فایلی
۴	تجهیز نرم افزار Win / isis جهت بازیابی اطلاعات با استفاده از نمایه نامه‌های فرمولی و اسامی ترکیبات شیمیایی	الف - برنامه نویسی به منظور افزایش فیلدهای اسامی ترکیبات شیمیایی و فرمولهای بسته شیمیایی به عنوان فیلدهای قابل بازیابی در نرم افزار win / isis ب - آزمایش قابلیت بازیابی و رفع مشکلات احتمالی
۵	ایجاد پایگاه اطلاعات علمی شیمی و مهندسی شیمی	الف - ایجاد پایگاه اطلاعات مورد نظر در پایگاه اطلاعات جامع مرکز به لحاظ ساختاری به صورتی که قابلیت جستجوی تخصصی با استفاده از نمایه نامه‌های فرمولی و اسامی ترکیبات شیمیایی در آن موجود باشد. ب - بارگذاری اطلاعات پردازش شده در طرح پیشین (۱۳۰۰ رکورد پایان نامه) و اطلاعات پردازش شده در این طرح (حدود ۲۵۰۰ رکورد سمینارها) در این پایگاه.

شماره مرحله	عنوان کلی مرحله	فعالیت‌های اصلی هر مرحله
۶	طراحی نرم افزار بانک اطلاعاتی ویژه اطلاعات علمی شیمی و مهندسی شیمی با قابلیت‌های ورود، ذخیره، نمایش و بازیابی اطلاعات شیمی مانند ساختارها و صور و علائم خاص مورد استفاده در متون شیمی، سازگار با cds	الف - انتخاب نرم افزار مناسب بانک اطلاعاتی ب - انتخاب نرم افزار مناسب شیمی ج - برنامه نویسی به منظور تجهیز نرم افزار بانک اطلاعاتی به قابلیت‌های نرم افزارهای شیمی د - آزمایش نهایی بر روی نرم افزار مورد نظر و رفع مشکلات احتمالی
۷	تهیه گزارش نهایی	الف - گزارش نهایی ب - بروشور ج - پوستر د - مستندات و راهنمای کاربران نرم افزار

کل مدت زمان لازم برای اجرای طرح (به ماه):

شماره مرحله	عنوان فعالیتهای اصلی در هر مرحله	زمان لازم به ماه																							
		۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰	۲۱	۲۲	۲۳	۲۴
۱	استخراج کلیه اطلاعات...	-																							
	ویرایش و نمایه سازی مجدد																								
	- یکسان سازی و استاندارد سازی																								
	- معادل یابی فارسی و لاتین																								
	- ارجاع دهی و استخراج کوتاه نوشته ها																								
۲	- استخراج اسامی ترکیبات شیمیایی																								
	- تصحیح اسامی ترکیبات شیمیایی																								
	- فرمول دهی به اسامی ترکیبات شیمیایی																								
	- ایجاد نمایه نامه فرمولی																								
	- ایجاد نمایه نامه اسامی ترکیبات شیمیایی																								
۳	- تبدیل فرمت از CDS به زرنگار و یا Word																								
	- صفحه آرایی و ویرایش نهایی																								
	- طراحی روی جلد																								
	- چاپ																								
	- یافتن فیله های مورد نیاز در کاربرگه...																								
۴	- افزایش این فیله ها به کاربرگه ها و تهیه...																								
	- برنامه نویسی به منظور افزایش فیله های																								
	اسامی ترکیبات شیمیایی به عنوان فیله های...																								
	- آزمایش قابلیت بازیابی و رفع مشکلات...																								
	- ایجاد پایگاه مورد نظر در پایگاه اطلاعات																								
۵	جامع مرکز به لحاظ ساختاری به صورتی که..																								
	- بارگذاری اطلاعات پردازش شده در طرح...																								

کل مدت زمان لازم برای اجرای طرح (به ماه):

شماره مرحله	عنوان فعالیتهای اصلی در هر مرحله	زمان لازم به ماه																							
		۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰	۲۱	۲۲	۲۳	۲۴
۶	- انتخاب نرم افزار مناسب بانک اطلاعاتی																								
	- انتخاب نرم افزار مناسب شیمی																								
	- برنامه نویسی به منظور تجهیز نرم افزار بانک																								
	اطلاعاتی به قابلیت های نرم افزارهای شیمی																								
	- آزمایش نهایی نرم افزار مورد نظر و رفع																								
	اشکالات احتمالی																								
۷	گزارش نهایی، بروشور، پوستر، مستندات و																								
	راهنمای کاربران																								

۱۹) هزینه‌های انجام طرح (ریال)

۱-۱۹) نیروی انسانی: فعالیت موظف اعضای هیات علمی مرکز اطلاعات و مدارک علمی ایران

میزان ساعت کار به تفکیک مراحل											شماره مرحله نام و نام خانوادگی هیات علمی
جمع	۱۰	۹	۸	۷	۶	۵	۴	۳	۲	۱	
۶۰۰				۴۰	۲۰	۱۰	۱۰	۲۰	۵۰۰		تقی رجیبی
۶۰۰											جمع کل (ساعت)

۲-۱۹) نیروی انسانی: فعالیت غیر موظف اعضای هیات علمی مرکز اطلاعات و مدارک علمی ایران

میزان ساعت کار به تفکیک مراحل													شماره مرحله نام و نام خانوادگی هیات علمی
جمع پرومانه (ریال)	پرومانه هر ساعت	جمع ساعات کار	۱۰	۹	۸	۷	۶	۵	۴	۳	۲	۱	
۲۸۸۰۰۰۰۰	۲۰۰۰۰	۱۴۴۰				۱۲۰	۵۰	۱۰	۱۰	۵۰	۱۲۰۰		تقی رجیبی
													-
۲۸۸۰۰۰۰۰۰	جمع کل (ریال)	۱۴۴۰											جمع کل (ساعت)

۳-۱۹) نیروی انسانی: سایر (افراد غیر عضو هیات علمی مرکز اطلاعات و مدارک علمی ایران)

میزان ساعت کار به تفکیک مراحل												شماره مرحله	
جمع پروژه / حق الزحمه (ریال)	پژوهانه / حق الزحمه میانگین هر ساعت	جمع ساعات کار	۱۰	۹	۸	۷	۶	۵	۴	۳	۲		۱
													مجری غ همکاری
۲۵,۹۰۰,۰۰۰	۱۷۵۰	۱۵۳۰				۸۵	۶۸۵	۱۸۰	۱۸۰	۷۵	۳۱۵	۱۰	همکاران اصلی
													مشاوره
													کارشناسی
۶,۰۰۰,۰۰۰	۶۰۰	۱۰۰									۱۰۰		اجرایی
۲۶,۵۰۰,۰۰۰	جمع کل (ریال)	۱۶۳۰				۸۵	۶۸۵	۱۸۰	۱۸۰	۷۵	۴۱۵	۱۰	جمع کل (ساعت)

۴-۱۹) تهیه اطلاعات

۷-۱۹ هزینه مسافرت

مقصد	در چه مرحله‌ای باید انجام شود (x)											هدف از مسافرت	تعداد افراد	هزینه (ریال)
	۱۰	۹	۸	۷	۶	۵	۴	۳	۲	۱				
جمع هزینه														
(۱ یا ۰)														

۸-۱۹ هزینه تهیه گزارشها

جمع (ریال)	هزینه تهیه گزارشها به تفکیک مراحل (ریال)									
	۱۰	۹	۸	۷	۶	۵	۴	۳	۲	۱
۷۰۰,۰۰۰				۷۰۰,۰۰۰						

۹-۱۹ سایر هزینه‌ها (طراحی و چاپ جلد)

جمع (ریال)	سایر هزینه‌ها به تفکیک مراحل (ریال)									
	۱۰	۹	۸	۷	۶	۵	۴	۳	۲	۱
۴,۰۰۰,۰۰۰									۴,۰۰۰,۰۰۰	

۱۰-۱۹ جمع هزینه‌ها

شماره مرحله	هزینه نیروی انسانی (۲-۱۹) و (۳-۱۹)	تهیه اطلاعات (۴-۱۹)	وسایل و تجهیزات (۶-۱۹)	مسافرت (۷-۱۹)	تهیه گزارشها (۸-۱۹)	سایر (۹-۱۹)	جمع هزینه‌های هر مرحله (ریال)
۱	۲۰۰,۰۰۰						
۲	۲۹,۷۰۰,۰۰۰	۱۰۰,۰۰۰				۴,۰۰۰,۰۰۰	
۳	۲,۵۰۰,۰۰۰						
۴	۳,۸۰۰,۰۰۰						
۵	۳,۸۰۰,۰۰۰						
۶	۱۱,۴۵۰,۰۰۰	۱۰۰,۰۰۰					
۷	۳,۸۵۰,۰۰۰						
۸							
۹							
۱۰							
جمع کل (ریال)	۵۵,۳۰۰,۰۰۰	۲۰۰,۰۰۰			۷۰۰,۰۰۰	۴,۰۰۰,۰۰۰	۶۰,۲۰۰,۰۰۰

مراجع

- (۱) روش استفاده از چکیده‌نامه‌های شیمی، ترجمه مسعود حسن‌پور، مرکز نشر دانشگاهی، ۱۳۷۰
- (۲) آشنایی با نحوه حمایت و توسعه فعالیتهای تحقیقاتی و تکنولوژی در ژاپن، محمدعلی شریفی، سازمان پژوهشهای علمی و صنعتی ایران، ۱۳۶۴
- (۳) نامگذاری در شیمی آلی، اتحادیه بین‌المللی شیمی محض و کاربردی I.U.P.A.C، مرکز نشر دانشگاهی ۱۳۷۳
- (۴) نامگذاری در شیمی معدنی، اتحادیه بین‌المللی شیمی محض و کاربردی I.U.P.A.C، مرکز نشر دانشگاهی ۱۳۶۴
- 5) An Encyclopedic Dictionary of Language and Languages, David crystal, Blackwell, 1992
- (۶) شیوه‌نامه ضبط اعلام انگلیسی در فارسی، ماندانا صدیق‌بهزادی، مرکز نشر دانشگاهی، ۱۳۷۵
- 7) McGraw - Hill dictionary of chemistry, Sybil P.Parker, McGraw-Hill Book Company, 1994-1997)
- 8) The Organic Chemist's Desk Reference, P.H.Rhodes, Chapman & hall, 1995
- (۹) راهنمای استفاده از چکیده‌نامه‌های شیمی، ترجمه مسعود حسن‌پور، مرکز نشر دانشگاهی، ۱۳۷۵
- (۱۰) نمایه‌سازی و چکیده‌نویسی، جنیفراولی، ترجمه دکتر جعفر مهرداد، سازمان مدارک فرهنگی انقلاب اسلامی، ۱۳۷۴
- 11) Hackh's chemical dictionary, Julius Grant, Mc Graw-Hill Book company, 1972
- 12) The Penguin dictionary of chemistry, D.W.A. SHARP, penguin books, 1983
- 13) McGraw-Hill dictionary of chemistry, sybil P.Parker, MCGraw-Hill Book company, 1984
- 14) Gardner's Chemical Synonyms and Trade Names, tenth edition, Michael and Irene Ash, Gower Publishing limited, 1994
- 15) Glossary of Chemical Term, Second edition, clifford A.Hamped & Gessner G.Hawley, VNR, 1982
- 16) Lange's Handbook of Chemistry, JOHN A.DEAN thirteenth edition, McGraw-Hill Book cmpany, 1985
- (۱۷) فرهنگ مصور شیمی، سیدرضا آقاپور مقدم، انتشارات اطلس، ۱۳۶۳
- (۱۸) واژه‌نامه فیزیک، مرکز نشر دانشگاهی، ۱۳۷۰
- (۱۹) واژه‌نامه علوم و تکنولوژی، مرکز نشر دانشگاهی، ۱۳۷۳

- (۲۰) واژه‌نامه شیمی، مرکز نشر دانشگاهی، ۱۳۷۰
- (۲۱) واژگان شیمی و مهندسی شیمی، مرکز نشر دانشگاهی
- (۲۲) فرهنگ شیمی، دیوید ویلیام آرتور شارپ، ترجمه دکتر عیسی یآوری، انتشارات فاطمی، ۱۳۷۵
- (۲۳) واژگان متالورژی (مواد)، مرکز نشر دانشگاهی، ۱۳۷۴
- (۲۴) واژه‌نامه ریاضی، مرکز نشر دانشگاهی
- (۲۵) واژه‌نامه زیست‌شناسی، دکتر حسن ابراهیم‌زاده و همکاران، انتشارات علوی، ۱۳۷۱
- (۲۶) اصطلاحنامه نما، یونسکو، ترجمه ملوک‌السادات بهشتی، جلد اول و دوم، ویرایش دوم، مرکز اطلاعات و مدارک علمی ایران، ۱۳۷۷
- (۲۷) سرعنوانهای موضوعی پزشکی، فاطمه رهادوست، دانشگاه علوم پزشکی ایران، ۱۳۷۲
- 28) INIS: Thesaurus, International Atomic Energy Agency, Vienna, January, 1996
- 29) Thesaurus of Engineering and Scientific Terms, us Department of Defence, Office of Naval Research, 1967
- 30) Industrial Chemical Thesaurus, Second edition, Volum 1 & 2, Micheal and Irene Ash, VCH, 1992
- 31) Medical Subject Headings, NIH Publication, 1988
- 32) Information Sources in chemistry, Bottle, Robert Thomas; Rowland, J.F.B, Bowker - saur Ltd 1993
- 33) Chemical Information, Yecheskel wolman, A wiley - Interscience Publication 1988
- 34) Further advances In chemical Information The Royal Society of chemistry 1994
- (۳۵) نمایه‌سازی-همارآ، علی آقابخشی، مرکز اطلاعات و مدارک علمی ایران، ۱۳۷۲
- (۳۶) راهنمای نگارش و ویرایش، دکتر محمدجعفر یاحقی - دکتر محمد مهدی ناصح، مؤسسه چاپ و انتشارات آستان قدس رضوی، مشهد، ۱۳۷۱
- (۳۷) شیوه‌نامه مرکز نشر دانشگاهی، ویرایش دوم
- (۳۸) غلط ننویسیم، ابوالحسن نجفی، مرکز نشر دانشگاهی، چاپ چهارم، ۱۳۷۰
- (۳۹) آواشناسی، علی محمد حق‌شناس، انتشارات آگاه، ۱۳۷۴
- (۴۰) زبان‌شناسی و زبان، جولیا اس. فالک، ترجمه خسرو غلامعلی‌زاده، مؤسسه چاپ و انتشارات آستان قدس رضوی، مشهد، ۱۳۷۲
- (۴۱) روشهای پژوهش در کتابداری و اطلاع‌رسانی، کریشان کومار، ترجمه فاطمه رهادوست، کتابخانه ملی جمهوری اسلامی ایران، ۱۳۷۴
- (۴۲) مبانی نظامها و خدمات اطلاعاتی، پاولین آرتون، ترجمه علی آقابخشی و همکاران، مرکز اطلاعات و مدارک علمی ایران، ۱۳۷۳
- (۴۳) راهنمای مرجع کاربران سی. دی. سی. اس آیین تحت ویندوز، امیر پارسی اصفهانی، ۱۳۷۷، مرکز اطلاعات و مدارک علمی ایران
- (۴۴) ایران مارک، الگوی ملی پیشنهادی برای ذخیره و مبادله پیشینه‌های کتاب‌شناختی، امیر پارسی اصفهانی، اطلاع‌رسانی نشریه فنی مرکز اطلاعات و مدارک علمی ایران، دوره ۱۱، شماره ۳، بهار ۱۳۷۵، ص ۳ - ۲۰

بسمه تعالی

مرکز اطلاعات و مدارک علمی ایران

مشخصات همکاران طرحهای پژوهش - پیوست فرم پرسشنامه

نام و نام خانوادگی همکار: صفا علی عسگری						تلفن تماس: ۲۲۹۷۹۸۹					
● مرتبه علمی: ☐ استاد ☐ دانشیار ☐ استادیار ☒ مربی ☐ هیچکدام ● تحصیلات:											
مقطع	رشته و گرایش	دانشگاه	کشور	سال دریافت	معدل کل						
کارشناسی	شیمی محض	دانشگاه تهران	ایران	۱۳۷۰	۱۴/۷۵						
کارشناسی ارشد	شیمی آلی	دانشگاه شیراز	ایران	۱۳۷۳	۱۶/۷۳						
دکتر											
● سوابق شغلی همکار که در توانایی وی برای انجام طرح موثر هستند (شغل فعلی حتماً قید شود):											
ردیف	نام سازمان محل خدمت	پژوهشی	آموزشی	اجرائی	نوع فعالیت	ست	تاریخ شروع	تاریخ پایان			
۱	دانشگاه آزاد اسلامی واحد شاهرود	✓	✓			عضو هیأت علمی	۱۳۷۴	ادامه دارد			
● سوابق پژوهشی همکار: ○ تعداد طرحهای پژوهشی که مجری بوده اید: در وزارت فرهنگ و آموزش عالی: در سایر سازمانها: ○ تعداد طرحهای پژوهشی که همکار بوده اید: در وزارت فرهنگ و آموزش عالی: در سایر سازمانها: ○ مشخصات طرحهایی که به موضوع این پیشنهاد ارتباط دارند:											
ردیف	عنوان طرح پژوهش	ست	محل انجام	اعتبار (ریال)	تاریخ پایان						
۱	یکسان سازی شیوه رسم الحاظ اسامی ترکیبات شیمیایی	همکار	مرکز اطلاعات و مدارک علمی ایران	۳۴۱۵۰۰۰۰	۷۹/۴/۱						
● انتشارات همکار: ○ تعداد کتاب: تألیف: ترجمه: ○ تعداد مقاله های فارسی: در نشریات ادواری داخلی: در سمنارهای داخلی: ○ تعداد مقاله های غیر فارسی: در نشریات ادواری خارجی: در سمنارهای خارجی: ○ مشخصات آثاری که به موضوع این پیشنهاد ارتباط دارند:											
ردیف	عنوان اثر	تألیف	ترجمه	تصحیح	نقد	کتاب	مقاله	سایر (نام ببرید)			
● زبانهای خارجی که به آن آشنایی دارید و میزان آشنایی: انگلیسی - خوب											

۲۲۷۶۱