Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования (ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**Программное средство «Chem helper»**

БГУИР КП 1-40 01 01 29 ПЗ

|  |  |  |
| --- | --- | --- |
|  |  | Студент: гр. 751001 А.А. Кимаев |
|  |  | Руководитель: асс. Шостак Е.В. |

Минск 2018

СОДЕРЖАНИЕ

[Введение 5](#_Toc513665878)

[1 Анализ предметной области 7](#_Toc513665879)

[1.1 Обзор аналогов 7](#_Toc513665880)

[1.2 Постановка задачи 9](#_Toc513665881)

[2 Разработка программного средства 10](#_Toc513665882)

[2.1 Структура программы 10](#_Toc513665883)

[2.2 Интерфейс программного средства 10](#_Toc513665884)

[2.3 Логика приложения 15](#_Toc513665888)

[3 Тестирование программного средства](#_Toc513665890) 21

[4 Руководство пользователя 22](#_Toc513665891)

[Заключение 24](#_Toc513665894)

[Список использованных источников 25](#_Toc513665895)

[Приложение А. Исходный код программы 26](#_Toc513665896)

ВВЕДЕНИЕ

Технический прогресс в последние несколько десятков лет сильно изменил жизнь обычного человека. Цифровые технологии ускорили развитие различных областей науки, позволили повысить качество жизни. Немалое влияние развитие информационных технологий оказало и на научную сферу: большое количество сложной вычислительной техники, всевозможные программы и приборы, определяющие температуру, плотность и многие другие показатели веществ, в том числе и химический состав.

Для формулирования понятия про химическое соединение важное значение имеет закон постоянства состава, открытый Жозефом Пустом в 1799 году. Этот закон утверждает, что независимо от того, каким образом было получено определённое соединение, оно всегда состоит из одних и тех же химических элементов, причём отношение масс этих элементов выражается малыми целыми числами. Через несколько лет, в 1803 году Джон Дальтон сформулировал закон кратных отношений, согласно которому в случае, когда из определённых элементов могут образовываться два соединения, с определённой массой первого из них сочетаются такие массы другого, что отношение их тоже даёт целое число. Эти два утверждения подготовили путь к пониманию атомного строения вещества.

Состав химического соединения записывается в виде химических формул, а строение часто изображается структурными формулами. Систематическое название также указывает состав соединения. Для установления качественного и количественного состава химического соединения, используются различные методы химического анализа (например, колориметрия, хроматография). Эти методы являются предметом изучения аналитической химии.

Необходимо понимать разницу между соединением и смесью. Различие физических и химических свойств – один из главных критериев отличия соединения от смесей простых или сложных веществ, так как свойства смеси обычно тесным образом связаны со свойствами компонентов. Другим критерием отличия является то, что смесь обычно может быть разделена на составляющие нехимическими процессами, такими, как просеивание, фильтрация, выпаривание, использование магнитов, тогда как компоненты химического соединения могут быть разделены только при помощи химической реакции. И наоборот, смеси могут быть созданы без использования химической реакции, а соединения – нет.

Некоторые смеси так тесно связаны, что многие их свойства сходны со свойствами химических соединений, и их легко спутать. Наиболее частым примером таких смесей являются сплавы. Сплавы изготавливаются при помощи физических процессов, обычно – путём расплавления и смешивания компонентов с последующим остыванием.

Химические соединения получают в результате химических реакций. Сложные вещества могут разлагаться с образованием нескольких других веществ. Образование химических соединений сопровождается выделением (экзотермическая реакция) или поглощением (эндотермическая реакция) энергии. Физические и химические свойства химических соединений отличаются от свойств веществ, из которых они получены. Химические соединения разделяются на неорганические и органические. Известно более ста тысяч неорганических и более трёх миллионов органических соединений. Каждое химическое соединение, которое описано в литературе, имеет уникальный идентификатор.

В настоящее время затруднительно быстро найти информацию об интересующем соединении, особенно, если оно достаточно сложное.

Целью данного проекта является разработка приложения, позволяющего выполнять поиск ссылок на интернет-ресурсы, посвящённые запрашиваемому соединению – «Chem Helper».

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
   1. Обзор аналогов

На данный момент существует множество различных поисковых систем, таких как Google, Яндекс, Yahoo и другие, менее известные аналоги. Они позволяют находить информацию по абсолютно любым предметам, однако это и является их главным минусом, поскольку не существует какого-либо способа отсеивания сайтов, исходя из тематики запроса пользователя. Иными словами, подобные поисковые системы не могут понять, что пользователя интересуют вещи, связанные с физикой, химией или другими узкими направлениями.

Есть несколько примеров такого поведения. Посылая запрос «Na» на сайт www.google.com, заметно, что поисковая система выдаёт результаты не только связанные с натрием, но и различные сторонние страницы, где просто было использовано сочетание букв «Na». Данная ситуация представлена на рисунке 1.1.

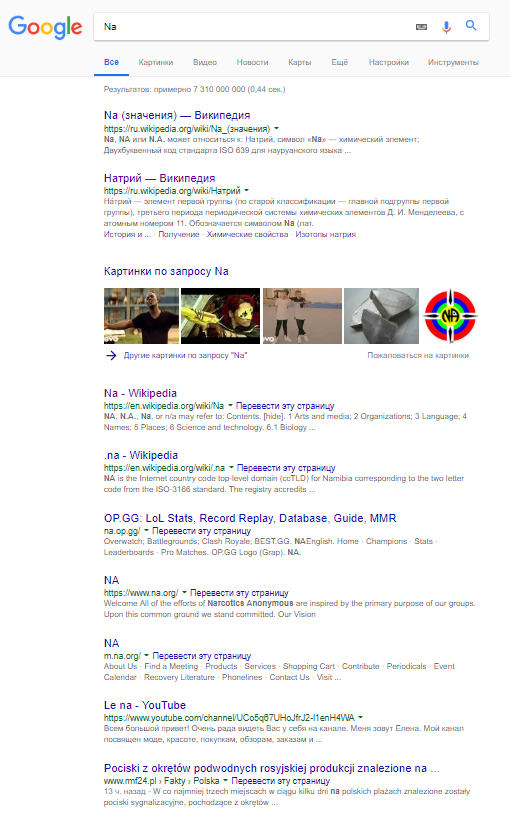


Рисунок 1.1 – Запрос «Na» на сайте google.com

Идентичную ситуацию можно наблюдать и на сайте yandex.ru, что продемонстрировано на рисунке 1.2.

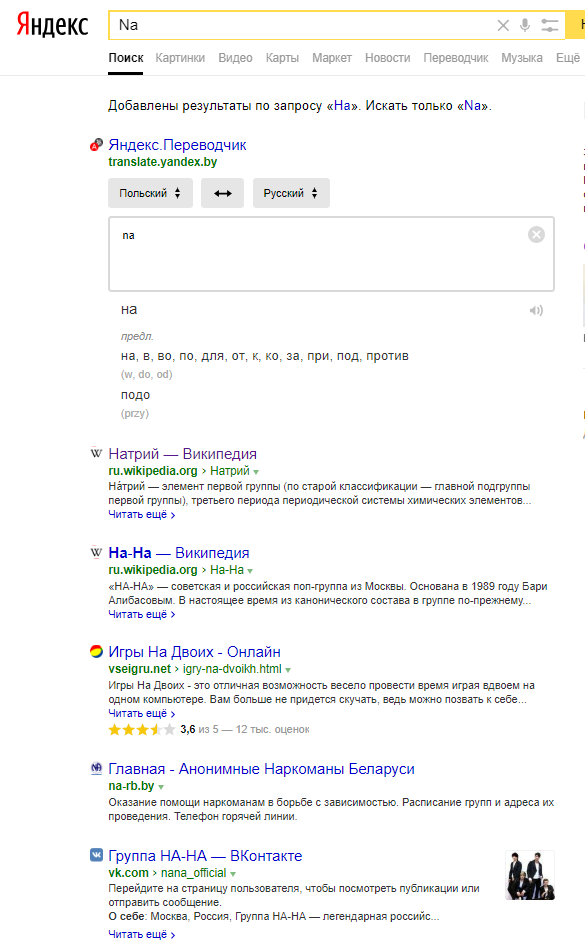
­­­­

Рисунок 1.2 – Запрос «Na» на сайте yandex.ru

Такое поведение наблюдается абсолютно на всех глобальных системах поиска: появляется много бесполезной для пользователя информации, что усложняет поиск необходимых данных.

* 1. Постановка задачи

В рамках данного курсового проекта планируется разработка программного средства «Chem Helper», адаптированного под запуск на современных операционных системах, в частности, версиях ОС Windows 10 и старше.

В процессе реализации будет разработан алгоритм определения соответствия химического соединения законам физики и химии, составлены протоколы работы с сетью интернет, позволяющие находить требуемую информацию, сформированы списки необходимых химических соединений.

В программном средстве планируется реализовать следующие функции:

* проверка правильности введённой формулы;
* нахождение и вывод на экран списка ссылок на интернет-ресурсы;
* запуск браузера при выборе ссылки.

Для разработки программного средства будет использоваться язык программирования C++ и среда разработки Qt Creator.

1. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА
2. 1. Структура программы

В данном приложении необходимо использовать четыре структурных блока:

– chemhelper (h/cpp) – модуль, отображающий главное окно программы и управляющий всеми текущими процессами;

– ch\_graphics (h/cpp) – модуль, отвечающий за отображение и анимацию элементов приложения;

– ch\_model (h/cpp) – модуль, отвечающий за все вычисления и изменения текущего состояния программы;

– ch\_variables\_and\_types (h/cpp) – модуль, в котором хрянятся все глобальные переменные и типы, используемые другими модулями.

* 1. Интерфейс программного средства

Внешний вид и удобность в использовании являются одними из главных критериев качества программного средства. Поэтому взаимодействие приложения с пользователем необходимо организовать максимально интуитивно и просто.

При нажатии посреди экрана на поле ввода вызывается процедура launchGEditAnimation, которая подсвечивает выбранное поле. Окно нажатия на данное поле отображено на рисунке 2.2.

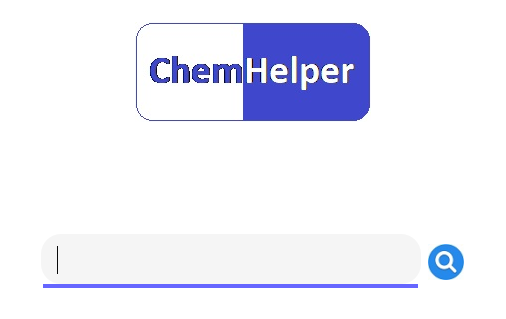


Рисунок 2.2 – Нажатие на поле ввода

Ниже приведён код обработчика данного события и процедуры launchGEditAnimation.

void gEditFocusIn(QTextEdit &edtInputFormula)

{

launchGEditAnimation();

if (edtInputFormula.toPlainText().toStdString() == "Ваша формула")

edtInputFormula.setText("");

}

void launchGEditAnimation()

{

SYSTEMTIME UTC;

if (!WasGEditAnimation)

{

GetSystemTime(&UTC);

GEditAnimationRunning = true;

TimeGEditAnimationStart = sysTimeToInt(UTC);

WasGEditAnimation = true;

}

}

При выходе из поля ввода, вызывается процедура launchGEditInverseAnimation, которая приводит поле ввода в изначальное состояние. Код обработчика данного события и процедуры launchGEditInverseAnimation имеет вид:

void gEditFocusOut(QTextEdit &edtInputFormula)

{

launchGEditInverseAnimation();

if(edtInputFormula.toPlainText().toStdString() == "")

edtInputFormula.setText("Ваша формула");

}

void launchGEditInverseAnimation()

{

SYSTEMTIME UTC;

if (WasGEditAnimation)

{

GetSystemTime(&UTC);

GEditInverseAnimationRunning = true;

TimeGEditInverseAnimationStart = sysTimeToInt(UTC);

WasGEditAnimation = false;

}

}

При нажатии в поле ввода на Enter или на значок поиска рядом с ним после прохождения проверок вызываются процедуры sendRequest, которая вызывает процедуру launchAllAnimation, ответственную за анимацию запроса, и появляется список результатов поиска. В результате анимации плавно изменится расположения ряда компонентов. В таком случае окно принимает вид, показанный на рисунке 2.4. Если же проверка на правильность введённой формулы не была пройдена, поле ввода подсветится красным цветом, что показано на рисунке 2.3.

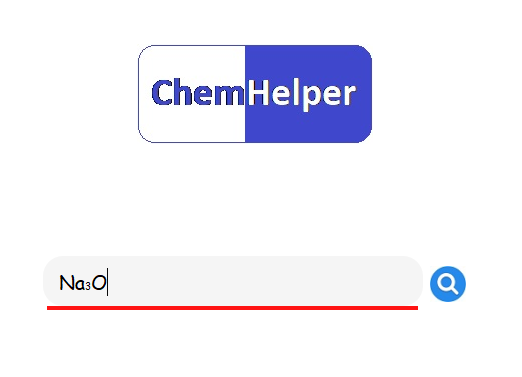


Рисунок 2.3 – Окно при неправильно введённой формуле

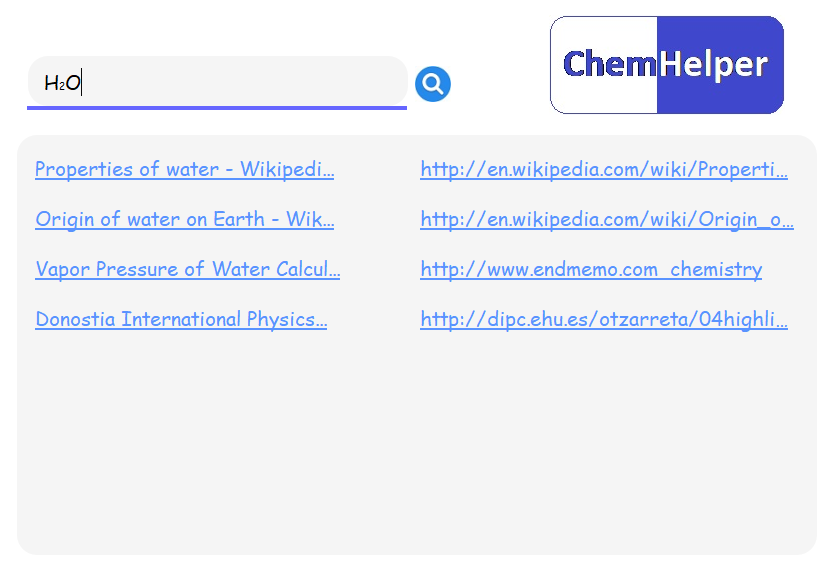


Рисунок 2.4 – Окно после отправки запроса

Код процедур sendRequest и launchAllAnimation приведён ниже.

void ChemHelper::sendRequest(const string &inputFormula)

{

ui->twSearchResults->clear();

ui->twSearchResults->setRowCount(0);

for(int i = 0; i < CiteArray.size(); i++)

{

reply = nam->get (QNetworkRequest (QUrl (QString::fromStdString ("https://www.google.com/search?q=" + CiteArray[i] + inputFormula))));

}

QObject::connect(reply, SIGNAL(finished()), this, SLOT(searchFinished()));

launchAllAnimation();

}

void launchAllAnimation()

{

SYSTEMTIME UTC;

if (!WasAllAnimation)

{

GetSystemTime(&UTC);

AllAnimationRunning = true;

TimeAllAnimationStart = sysTimeToInt(UTC);

WasAllAnimation = true;

}

}

Все рассмотренные анимации происходят по таймеру, который вызывает процедуру redraw, основанную на текущем времени и времени начала конкретной анимации. Она изменяет положение компонентов формы в зависимости от этой анимации. Ниже представлен код обработки события таймера.

void ChemHelper::update()

{

SYSTEMTIME UTC;

GetSystemTime(&UTC);

long long now = sysTimeToInt(UTC);

sysTimeToInt(UTC);

if (now - TimeGEditAnimationStart > GEditAnimationTime && GEditAnimationRunning)

gEditAnimationFinished(\*(ui->gviewUnderscore));

if (now - TimeGEditInverseAnimationStart > GEditAnimationTime && GEditInverseAnimationRunning)

gEditInverseAnimationFinished(\*(ui->gviewUnderscore));

if (now - TimeAllAnimationStart > AllAnimationTime && AllAnimationRunning)

allAnimationFinished(\*(ui->GEdit), \*(ui->lblLogo), \*(ui->twSearchResults));

if (now - TimeAllInverseAnimationStart > AllAnimationTime && AllInverseAnimationRunning)

allInverseAnimationFinished(\*(ui->GEdit), \*(ui->lblLogo), \*(ui->twSearchResults));

if(GEditAnimationRunning || GEditInverseAnimationRunning || AllAnimationRunning || AllInverseAnimationRunning)

needRedraw(\*(ui->GEdit), \*(ui->gviewUnderscore), \*(ui->lblLogo), \*(ui->twSearchResults));

}

Ещё одной важной частью интерфейса является обработчик событий приложения, код которого представлен ниже.

bool ChemHelper::eventFilter(QObject \*watched, QEvent \*event)

{

if(watched == ui->edtInputFormula)

{

switch (event->type())

{

case QEvent::KeyPress :

{

int key = (static\_cast<QKeyEvent\*>(event))->key();

//cout<<key<<endl;

if(!isValidKey(key))

return true;

if(key == 16777220) // enter

{

emit(on\_btnSearch\_clicked());

return true;

}

if(key == 16777216) //esc

QCoreApplication::postEvent(watched, new QEvent(QEvent::FocusOut));

else

{

ui->gviewUnderscore->setBackgroundBrush(QBrush(UnderscoreCorrectColor, Qt::SolidPattern));

needRedraw(\*(ui->GEdit), \*(ui->gviewUnderscore), \*(ui->lblLogo), \*(ui->twSearchResults));

}

if(isKNumber(key))

ui->edtInputFormula->setCurrentFont(QFont("Comic Sans MS", 6));

if(isKLetter(key) || (key >= 40 && key <= 43) || key == 61 || key == 32 || key == 91 || key == 93)

ui->edtInputFormula->setCurrentFont(QFont("Comic Sans MS", 12));

break;

}

case QEvent::FocusIn:

{

gEditFocusIn(\*(ui->edtInputFormula));

break;

}

case QEvent::FocusOut:

{

gEditFocusOut(\*(ui->edtInputFormula));

ui->edtInputFormula->clearFocus();

break;

}

}

}

return QObject::eventFilter(watched, event);

}

* 1. Логика приложения

Работу данной части программы обеспечивают модули ch\_model(h/cpp) и chemhelper(h/cpp).

Главными задачами данного приложения являются определение правильности введённой химической формулы, поиск сайтов, посвящённых этой формуле в интернете, и вывод их на экран с возможноcтью переходить через них в окно браузера. Для ввода формулы наиболее подходящим является стандартный компонент QTextEdit.

Для проверки правильности введённой химической формулы используется внешний загружаемый файл Elements, в котором хранится информация о каждом элементе: формула, степени окисления от -8 до +8 с указанием свойств оксидов данной степени окисления, и название на русском языке.

Для реализации подобного подхода описывается класс chemElement, код которого представлен ниже.

class chemElement

{

public:

std::string formula;

std::vector<int> degreeOfOxidation;

std::string name;

int quantity;

chemElement () : formula("") {}

chemElement (const chemElement &x) :

formula (x.formula), degreeOfOxidation (x.degreeOfOxidation), name (x.name), quantity (x.quantity) {}

chemElement (std::string \_formula, std::string \_degreeOfOxidation, std::string \_name, int \_quantity) :

formula(\_formula), name(\_name), quantity(\_quantity)

for(size\_t i = 0; i < \_degreeOfOxidation.size(); i++)

degreeOfOxidation.push\_back(\_degreeOfOxidation[i] == '-'? -1 : \_degreeOfOxidation[i] - '0');

};

На основе файла Elements создаётся объект класса vector<chemElement> elementList.

Формула разбивается на отдельные элементы функцией CutToIons, код которой имеет вид:

vector<chemElement> cutToIons(const string &inputFormula)

{

vector<chemElement> result;

int i = inputFormula.size() - 1, factor = 1;

chemElement currentElement;

bool valid = true;

stack<int> allFactors;

while ((i>=0) && valid)

{

currentElement = getLatestElement(inputFormula, i);

if (currentElement.formula == "")

valid = false;

else

{

if (currentElement.formula == ")")

{

factor \*= currentElement.quantity;

allFactors.push(currentElement.quantity);

}

if (currentElement.formula == "(")

{

if (!allFactors.empty())

{

factor /= allFactors.top();

allFactors.pop();

}

else

valid = false;

}

if (currentElement.name != "")

{

currentElement.quantity \*= factor;

result.push\_back(currentElement);

}

}

}

if (!valid)

result.clear();

return result;

}

Код функции, возвращающей последний элемент, представлен ниже.

chemElement getLatestElement(const string &inputFormula, int &i)

{

chemElement result;

int someQuantity = 0;

someQuantity = getIntegerBeforePosition( inputFormula, i );

if (i > 0)

result = findElement(inputFormula.substr(i - 1, 2));

if (result.formula == "")

result = findElement(inputFormula.substr(i, 1));

if (inputFormula[i] == '(' || inputFormula[i] == ')')

{

result.formula = inputFormula[i];

result.name = "";

}

if (result.formula != "")

{

result.quantity = someQuantity;

i -= result.formula.size();

}

else

i = -1;

return result;

}

После получения массива элементов, из которых состоит формула, выполняется рекурсивная функция sortThroughOxidationDegreeses, возвращающая значение типа bool: можно ли составить формулу из исходных элементов. Она перебирает все возможные для всех элементов степени окисления, и если хоть в одном из наборов суммарная степень окисления равна нулю, то формула существует, иначе нет.

Поиск формулы в интернете выполняется с помощью класса QNetworkAcessManager являющегося инициатором запроса, и QNetworkReply получающего ответ. Работа этих компонентов была представлена в процедуре sendRequest.

Как компонент для поля с результатами поиска был выбран QTableWidget. Его заполнение было реализовано с помощью процедуры fillTWSearchResults. Ниже приведён код этой процедуры.

void ChemHelper::fillTWSearchResults(const std::wstring &HTMLCode)

{

int i = 0;

bool divgOpened = false, h3Opened = false, aOpened = false, citeOpened = false;

int countDivOpened = 0;

vector<wstring> titles, URLs;

while (i < HTMLCode.size())

{

if (divgOpened && forwardNSymbols(i, 6, HTMLCode) == L"</div>" && countDivOpened == 1)

{

divgOpened = false;

countDivOpened--;

}

if (countDivOpened > 0 && forwardNSymbols(i, 6, HTMLCode) == L"</div>")

countDivOpened--;

if (citeOpened && HTMLCode[i] == '<')

{

if (forwardNSymbols(i, 7, HTMLCode) == L"</cite>")

citeOpened = false;

else

{

shiftUntilX(i, '>', HTMLCode);

i++;

}

}

if (citeOpened)

{

URLs.push\_back(shiftUntilX(i, '<', HTMLCode));

i--;

}

if (divgOpened && backNSymbols(i, 5, HTMLCode) == L"<cite")

{

shiftUntilX(i, '>', HTMLCode);

citeOpened = true;

}

if (aOpened && forwardNSymbols(i, 4, HTMLCode) == L"</a>")

aOpened = false;

if (h3Opened && HTMLCode[i] == '<')

{

if (forwardNSymbols(i, 5, HTMLCode) == L"</h3>")

h3Opened = false;

else

{

shiftUntilX(i, '>', HTMLCode);

i++;

}

}

if (h3Opened)

{

titles.push\_back(shiftUntilX(i, '<', HTMLCode));

if ( titles.back().size() > 30 )

{

titles.back().erase(30, titles.size() - 30);

titles.back() += L"...";

aOpened = false;

}

i--;

}

if (aOpened && backNSymbols(i, 3, HTMLCode) == L"<h3")

{

shiftUntilX(i, '>', HTMLCode);

h3Opened = true;

}

if (divgOpened && backNSymbols(i, 2, HTMLCode) == L"<a")

{

shiftUntilX(i, '>', HTMLCode);

aOpened = true;

}

if (countDivOpened > 0 && backNSymbols(i, 4, HTMLCode) == L"<div")

countDivOpened++;

if (backNSymbols(i, 14, HTMLCode) == L"<div class=\"g\"")

{

//cout<<"kek"<<endl;

shiftUntilX( i, '>', HTMLCode );

divgOpened = true;

countDivOpened = 1;

}

i++;

}

/\*for(int i = 0; i < titles.size(); i++)

{

wcout << QString::fromStdWString (titles[i]).toStdWString() << L" " << URLs[i] << endl;

}\*/

for (int i = 0; i < titles.size(); i++)

{

ui->twSearchResults->insertRow(ui->twSearchResults->rowCount());

ui->twSearchResults->setRowHeight(ui->twSearchResults->rowCount() - 1, 50);

//titles[i] = L"ывамыфывфыdfsdfvsdfваьидла";

//titles[i] = QString::fromStdWString(titles[i]).toStdWString();

QTableWidgetItem \*x = new QTableWidgetItem(QString::fromStdWString(titles[i]));

//wcout << x->text().toStdWString() << endl;

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 0, x);

if (forwardNSymbols(0, 4, URLs[i]) == L"http")

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 1, new QTableWidgetItem(QString::fromStdWString(URLs[i])));

else

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 1, new QTableWidgetItem(QString::fromStdWString(L"http://" + URLs[i])));

}

}

Таким образом, мы получили все названия и ссылки на сайты, полученные с текущей страницы.

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе тестирования приложения были выявлены некоторые недостатки интерфейса и работы программного средства.

Была обнаружена проблема при запуске программы. Окно автоматически выделяло всю область поля ввода и устанавливало курсор в начало. Решением этой проблемы стала функция clearFocus поля ввода. При её вызове будет снято выделение с edtInputFormula.

ui->edtInputFormula->clearFocus();

Второй сложностью стала невозможность вызова сайтов, которые не имеют в обычной адресной строке протокола http или https. Для решения проблемы оказалось достаточным добавить в название сайта протокол http://.

if (forwardNSymbols(0, 4, URLs[i]) == L"http")

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 1, new QTableWidgetItem(QString::fromStdWString(URLs[i])));

else

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 1, new QTableWidgetItem(QString::fromStdWString(L"http://" + URLs[i])));

Также проблемой оказалось взаимодействие между классами программы, поскольку в методах одного класса мы никак не можем взаимодействовать с полями и методами другого класса. Решением данной проблемы оказалась предоставленная средой разработки механика сигналов и слотов, реализуемая с помощью ключевого слова connect.

connect(lblSearchButton, SIGNAL(clicked()), this, SLOT(on\_btnSearch\_clicked()));

Остальные проблемы были связаны с недочётами на стадии проектирования и были устранены на стадии разработки.

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для поиска формулы необходимо ввести её в поле по центру экрана и затем нажать клавишу Enter или значок рядом с полем ввода, как показано на рисунке 4.1.

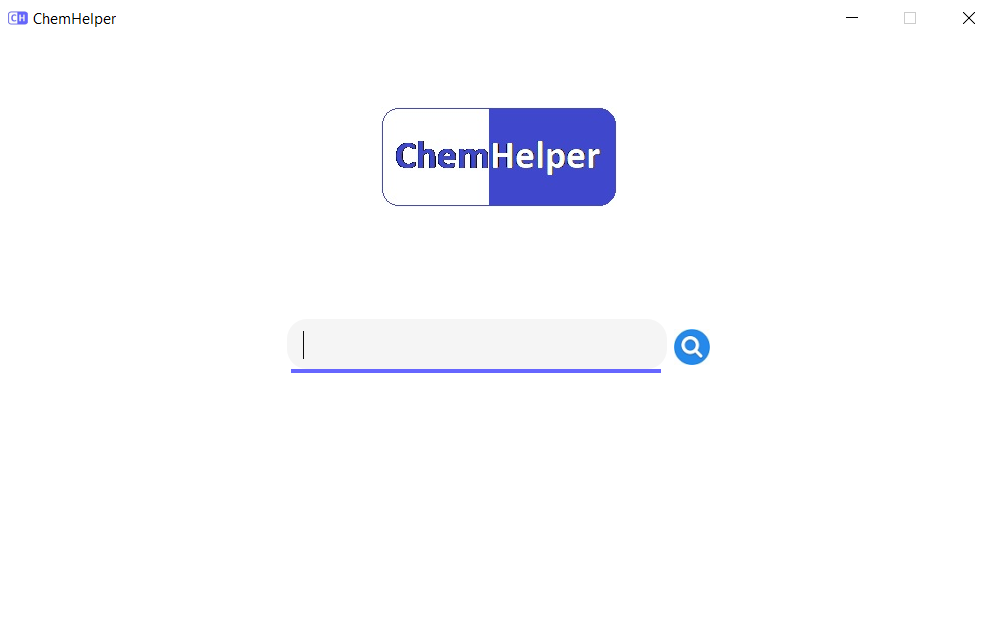


Рисунок 4.1 – Внешний вид окна программы

Если введена корректная, согласно степеням окисления, формула, то после небольшой задержки появится таблица с заголовками страниц и ссылок к ним, что проиллюстрировано на рисунке 4.2.

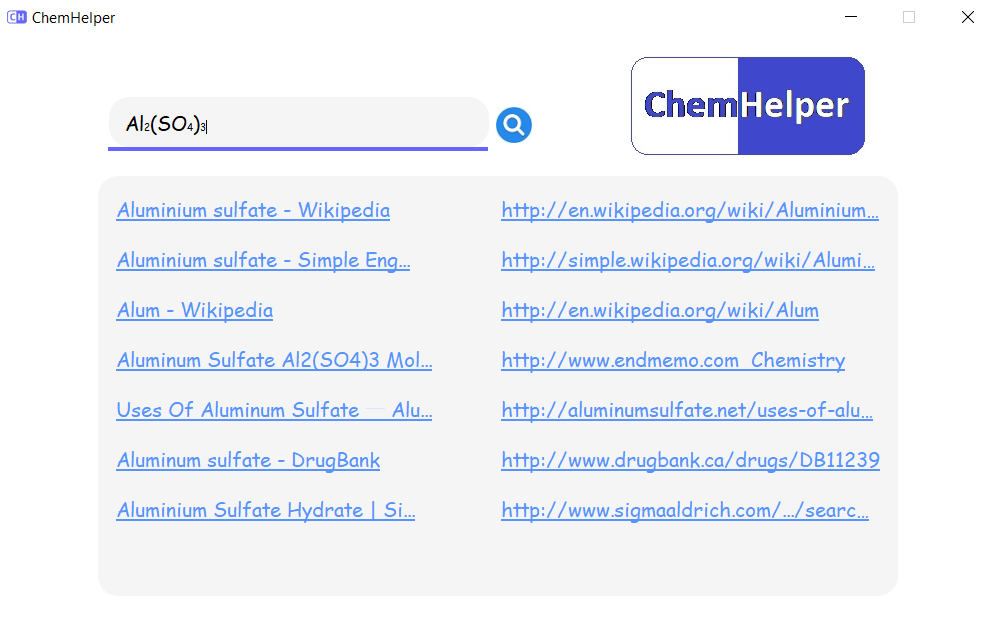


Рисунок 4.2 – Внешний вид окна программы после выполнения поиска

Для того, чтобы перейти по ссылке необходимо лишь нажать на интересующую вас ссылку и автоматически будет открыта страница браузера.

В случае ошибки в формуле, полоска под полем ввода станет красной и поиск не выполнится. Эта ситуация показана на рисунках 4.3 и 4.4.

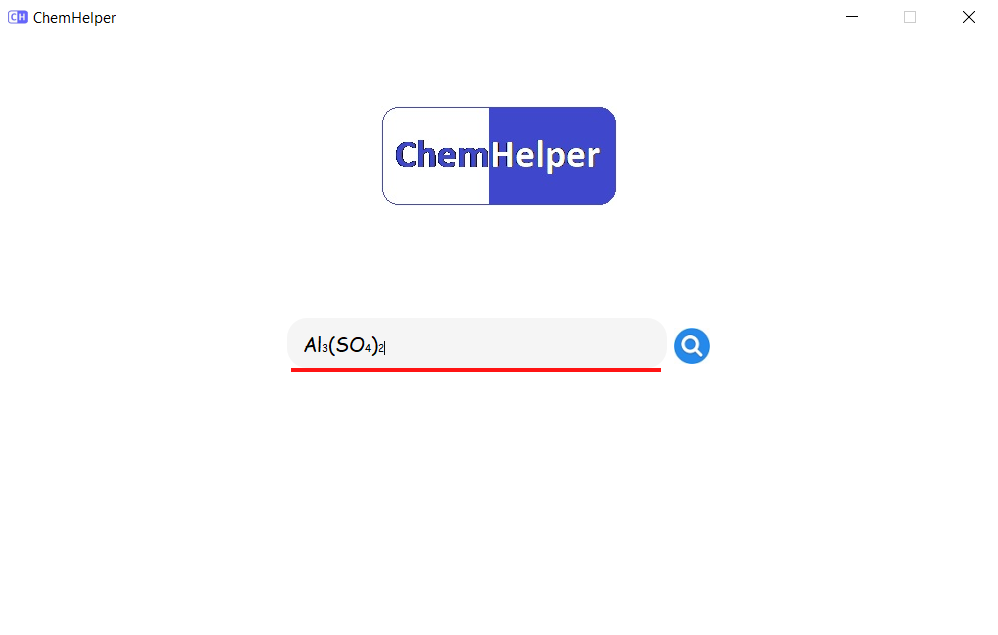


Рисунок 4.3 – Внешний вид окна программы при попытке поиска неправильной формулы

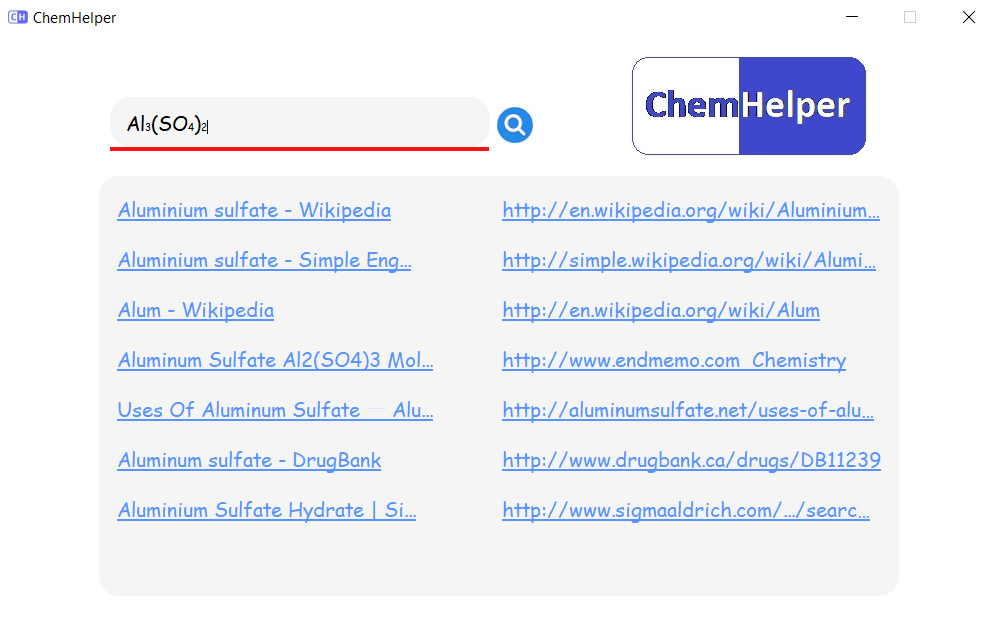


Рисунок 4.4 – Внешний вид окна программы при попытке поиска без интернет-соединения **ЗАКЛЮЧЕНИЕ**

Несмотря на то, что существует множество различных поисковых систем, ни одна из них не предоставляет готовой возможности искать информацию на конкретную узкую тему.

В рамках данного курсового проекта было разработано программное средство «Chem Helper», предназначенное для поиска информации по химическому соединению. Данное программное средство обладает всеми качествами, необходимыми для того, чтобы быть наиболее понятным и удобным для тех, кто использует программу такого типа впервые.

Были реализованы простые и красивые анимации для комфортной работы с приложением.

Тема данного курсового проекта потребовала изучения различных свойств химических формул, строения сайтов поисковых систем, методов работы с интернетом и удалёнными серверами по различным протоколам.

В процессе разработки были приобретены навыки работы с файловыми базами данных, множеством функций WinApi и Indy10, разработки своего формата интернет-запросов, что является необходимой частью данного приложения.

Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

* проверка правильности введённой формулы;
* нахождение и вывод на экран списка ссылок на интернет-ресурсы;
* запуск браузера при выборе ссылки.

Архитектура программы является модульной и гибкой, что обеспечивает потенциал для развития. Так, в дальнейшем в программу могут быть добавлены дополнительные функции, такие как поиск не только по формуле, но и просто по набору элементов, из которых может быть составлена формула, а также калькулятор химических уравнений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Глухова, Л. А. Основы алгоритмизации и программирования: лаб. практикум для студ. спец. I–40 01 01 «Программное обеспечение информационных технологий» дневной формы обуч. В 4 ч. / Л. А. Глухова, Е. Е. Фадеева, Е. П. Фадеева. – Минск : БГУИР, 2007. – Ч. 3. – 51 с.

[2] Глухова, Л. А. учеб. пособие по дисциплине: Основы алгоритмизации и программирования для студ. спец. I–40 01 01 «Программное обеспечение информационных технологий» дневной формы обучения. В 2 ч. Ч.2. – Минск: БГУИР, 2006 – 177 с.

[3] Программирование на языке Delphi / Д. А. Сурков [и др.]. – учеб. пособие. – Режим доступа: http://www.rsdn.ru/?summary/3165.xml, – 2005.

[4] Орлов, С. А. Технологии разработки программного обеспечения: учеб. Пособие. – СПб. : Питер, 2003. – 480 с.

[5] Уилсон, С. Принципы проектирования и разработки программного

обеспечения, yчеб. курс. – СПб. : Русская Редакция, 2003. – 570 с.

[6] Фараонов, В. В. Delphi 6. Учебный курс: / Фараонов В. В. – СПб. :

изд. С. В. Малгачёва, 2001. – 231 с.

[7] Шупрута, В. В. Delphi 2005. Учимся программировать: / В. В.

Шупрута. – Москва : изд. «НТ Пресс», 2001. – 140 с.

ПРИЛОЖЕНИЕ А

Исходный код программы

Модуль контроллера:

chemhelper.h:

#ifndef CHEMHELPER\_H

#define CHEMHELPER\_H

#include <QMainWindow>

#include <QTimer>

#include <QNetworkAccessManager>

#include <QNetworkReply>

namespace Ui

{

class ChemHelper;

}

class ChemHelper : public QMainWindow

{

Q\_OBJECT

public:

explicit ChemHelper(QWidget \*parent = nullptr);

~ChemHelper();

void sendRequest(const std::string &inputFormula);

void fillTWSearchResults(const std::wstring &HTML);

protected:

bool eventFilter(QObject \*watched, QEvent \*event); //override;

private slots:

void on\_btnSearch\_clicked();

void searchFinished();

void update();

void on\_twSearchResults\_cellClicked(int row, int column);

private:

Ui::ChemHelper \*ui;

QTimer \*tmrProcessTimig;

QNetworkAccessManager \*nam;

QNetworkReply \*reply;

};

#endif // CHEMHELPER\_H#include <QDesktopWidget>

chemhelper.cpp:

#include <iostream>

#include <QKeyEvent>

#include <QGraphicsView>

#include <QGraphicsScene>

#include <cmath>

#include <windows.h>

#include "chemhelper.h"

#include "ui\_chemhelper.h"

#include "ch\_variables\_and\_types.h"

#include "ch\_model.h"

using namespace std;

ChemHelper::ChemHelper(QWidget \*parent) : QMainWindow(parent), ui(new Ui::ChemHelper), tmrProcessTimig(new QTimer), nam(new QNetworkAccessManager)

{

MonitorWidth = QApplication::desktop()->width();

MonitorHeight = QApplication::desktop()->height();

btnSearchWidth = min(MonitorWidth, MonitorHeight) / 30;

ui->setupUi(this);

tmrProcessTimig->setInterval(15);

connect(tmrProcessTimig, SIGNAL(timeout()), this, SLOT(update()));

tmrProcessTimig->start();

ui->gviewUnderscore->setScene(new QGraphicsScene);

ui->gviewUnderscore->setBackgroundBrush(QBrush(UnderscoreCorrectColor, Qt::SolidPattern));

ui->ltGEdit->setAlignment(ui->gviewUnderscore, Qt::AlignHCenter);

QString filename = "..\\data\\images\\SearchButton.jpg";

QClickableLabel \*lblSearchButton = new QClickableLabel(this);

lblSearchButton->setAlignment(Qt::AlignCenter);

lblSearchButton->setSizePolicy(QSizePolicy::Preferred, QSizePolicy::Fixed);

lblSearchButton->setFixedSize(btnSearchWidth, btnSearchWidth);

lblSearchButton->setScaledContents(true);

lblSearchButton->setCursor(Qt::PointingHandCursor);

//lblSearchButton->setGeometry(100, 200, 0, 0);

connect(lblSearchButton, SIGNAL(clicked()), this, SLOT(on\_btnSearch\_clicked()));

QPixmap pix;

if(pix.load(filename))

lblSearchButton->setPixmap(pix);

ui->ltGEB->addWidget(lblSearchButton);

ui->twSearchResults->setEditTriggers(QAbstractItemView::NoEditTriggers);

qApp->installEventFilter(this);

initializeVariables(\*(ui->GEdit), \*(ui->gviewUnderscore), \*(ui->lblLogo), \*(ui->twSearchResults));

ui->edtInputFormula->clearFocus();

}

ChemHelper::~ChemHelper()

{

delete ui;

delete tmrProcessTimig;

}

void ChemHelper::update()

{

SYSTEMTIME UTC;

GetSystemTime(&UTC);

long long now = sysTimeToInt(UTC);

sysTimeToInt(UTC);

if (now - TimeGEditAnimationStart > GEditAnimationTime && GEditAnimationRunning)

gEditAnimationFinished(\*(ui->gviewUnderscore));

if (now - TimeGEditInverseAnimationStart > GEditAnimationTime && GEditInverseAnimationRunning)

gEditInverseAnimationFinished(\*(ui->gviewUnderscore));

if (now - TimeAllAnimationStart > AllAnimationTime && AllAnimationRunning)

allAnimationFinished(\*(ui->GEdit), \*(ui->lblLogo), \*(ui->twSearchResults));

if (now - TimeAllInverseAnimationStart > AllAnimationTime && AllInverseAnimationRunning)

allInverseAnimationFinished(\*(ui->GEdit), \*(ui->lblLogo), \*(ui->twSearchResults));

if(GEditAnimationRunning || GEditInverseAnimationRunning || AllAnimationRunning || AllInverseAnimationRunning)

needRedraw(\*(ui->GEdit), \*(ui->gviewUnderscore), \*(ui->lblLogo), \*(ui->twSearchResults));

}

wstring vectorWStringToWString(const vector<wstring> &src)

{

wstring result = L"";

for(int i = 0; i < src.size(); i++)

result.append(src[i]);

return result;

}

void ChemHelper::fillTWSearchResults(const std::wstring &HTMLCode)

{

int i = 0;

bool divgOpened = false, h3Opened = false, aOpened = false, citeOpened = false;

int countDivOpened = 0;

vector<wstring> titles, URLs;

while (i < HTMLCode.size())

{

//wcout<<HTMLCode[i]<<endl;

//ShowMessage(IntToStr(i) + backNSymbols(i, 6, HTMLCode));

if (divgOpened && forwardNSymbols(i, 6, HTMLCode) == L"</div>" && countDivOpened == 1)

{

divgOpened = false;

countDivOpened--;

}

if (countDivOpened > 0 && forwardNSymbols(i, 6, HTMLCode) == L"</div>")

countDivOpened--;

if (citeOpened && HTMLCode[i] == '<')

{

if (forwardNSymbols(i, 7, HTMLCode) == L"</cite>")

citeOpened = false;

else

{

shiftUntilX(i, '>', HTMLCode);

i++;

}

}

if (citeOpened)

{

URLs.push\_back(shiftUntilX(i, '<', HTMLCode));

i--;

}

if (divgOpened && backNSymbols(i, 5, HTMLCode) == L"<cite")

{

shiftUntilX(i, '>', HTMLCode);

citeOpened = true;

}

if (aOpened && forwardNSymbols(i, 4, HTMLCode) == L"</a>")

aOpened = false;

if (h3Opened && HTMLCode[i] == '<')

{

if (forwardNSymbols(i, 5, HTMLCode) == L"</h3>")

h3Opened = false;

else

{

shiftUntilX(i, '>', HTMLCode);

i++;

}

}

if (h3Opened)

{

titles.push\_back(shiftUntilX(i, '<', HTMLCode));

if ( titles.back().size() > 30 )

{

titles.back().erase(30, titles.size() - 30);

titles.back() += L"...";

aOpened = false;

}

i--;

}

if (aOpened && backNSymbols(i, 3, HTMLCode) == L"<h3")

{

shiftUntilX(i, '>', HTMLCode);

h3Opened = true;

}

if (divgOpened && backNSymbols(i, 2, HTMLCode) == L"<a")

{

shiftUntilX(i, '>', HTMLCode);

aOpened = true;

}

if (countDivOpened > 0 && backNSymbols(i, 4, HTMLCode) == L"<div")

countDivOpened++;

if (backNSymbols(i, 14, HTMLCode) == L"<div class=\"g\"")

{

//cout<<"kek"<<endl;

shiftUntilX( i, '>', HTMLCode );

divgOpened = true;

countDivOpened = 1;

}

i++;

}

/\*for(int i = 0; i < titles.size(); i++)

{

wcout << QString::fromStdWString(titles[i]).toStdWString() << L" " << URLs[i] << endl;

}\*/

for (int i = 0; i < titles.size(); i++)

{

ui->twSearchResults->insertRow(ui->twSearchResults->rowCount());

ui->twSearchResults->setRowHeight(ui->twSearchResults->rowCount() - 1, 50);

//titles[i] = L"ывамыфывфыdfsdfvsdfваьидла";

//titles[i] = QString::fromStdWString(titles[i]).toStdWString();

QTableWidgetItem \*x = new QTableWidgetItem(QString::fromStdWString(titles[i]));

//wcout << x->text().toStdWString() << endl;

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 0, x);

if (forwardNSymbols(0, 4, URLs[i]) == L"http")

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 1, new QTableWidgetItem(QString::fromStdWString(URLs[i])));

else

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 1, new QTableWidgetItem(QString::fromStdWString(L"http://" + URLs[i])));

}

}

void ChemHelper::searchFinished()

{

wstring HTML(reply->readAll().toStdString().begin(), reply->readAll().toStdString().end());

wcout<<HTML<<endl;

fillTWSearchResults(HTML);

}

void ChemHelper::sendRequest(const string &inputFormula)

{

ui->twSearchResults->clear();

ui->twSearchResults->setRowCount(0);

/\*for(int i = 0; i < CiteArray.size(); i++)

{ CHANGE TO I

reply = nam->get(QNetworkRequest(QUrl(QString::fromStdString("https://www.google.com/search?q=" + CiteArray[0] + inputFormula))));

}

QObject::connect(reply, SIGNAL(finished()), this, SLOT(searchFinished()));\*/

//cout<<inputFormula<<endl;

if(inputFormula == "H2O")

{

//cout<<"kek"<<endl;

//wcout<<vectorWStringToWString(HTMLH2O)<<endl;

fillTWSearchResults(vectorWStringToWString(HTMLH2O));

}

if(inputFormula == "HCl")

{

fillTWSearchResults(vectorWStringToWString(HTMLHCl));

}

if(inputFormula == "Al2(SO4)3")

{

fillTWSearchResults(vectorWStringToWString(HTMLAl2SO43));

}

//SrchResFinishHeight = min(SrchResFinishHeight, ui->twSearchResults->rowCount() \* 50 + 20);

launchAllAnimation();

}

void ChemHelper::on\_btnSearch\_clicked()

{

if(isValidChemical(ui->edtInputFormula->toPlainText().toStdString()))

{

ui->edtInputFormula->clearFocus();

//cout<<ui->edtInputFormula->toPlainText().toStdString()<<endl;

sendRequest(normallize(ui->edtInputFormula->toPlainText().toStdString()));

}

else

{

ui->gviewUnderscore->setBackgroundBrush(QBrush(UnderscoreWrongColor, Qt::SolidPattern));

WasGEditAnimation = false;

launchGEditAnimation();

WasChangedInputFormula = false;

}

}

bool isValidKey(int key)

{

/\*

\* 16777220 - enter

\* 16777216 - esc

\* 16777219 - backspase

\* 16777223 - del

\* 16777(234, 235, 236, 237) - arrows (l, u, r, d)

\* 40 .. 43 - '(', ')', '\*', '+'

\* 61 - '='

\* 32 - ' '

\* 91 - '['

\* 93 - ']'

\*/

return (isKNumber(key) || isKLetter(key) ||

key == 16777220 || key == 16777216 || key == 16777219 || key == 16777223 ||

key == 16777234 || key == 16777235 || key == 16777236 || key == 16777237 ||

(key >= 40 && key <= 43) || key == 61 || key == 32 || key == 91 || key == 93);

}

bool ChemHelper::eventFilter(QObject \*watched, QEvent \*event)

{

if(watched == ui->edtInputFormula)

{

switch (event->type())

{

case QEvent::KeyPress :

{

int key = (static\_cast<QKeyEvent\*>(event))->key();

//cout<<key<<endl;

if(!isValidKey(key))

return true;

if(key == 16777220) // enter

{

emit(on\_btnSearch\_clicked());

return true;

}

if(key == 16777216) //esc

QCoreApplication::postEvent(watched, new QEvent(QEvent::FocusOut));

else

{

ui->gviewUnderscore->setBackgroundBrush(QBrush(UnderscoreCorrectColor, Qt::SolidPattern));

needRedraw(\*(ui->GEdit), \*(ui->gviewUnderscore), \*(ui->lblLogo), \*(ui->twSearchResults));

}

if(isKNumber(key))

ui->edtInputFormula->setCurrentFont(QFont("Comic Sans MS", 6));

if(isKLetter(key) || (key >= 40 && key <= 43) || key == 61 || key == 32 || key == 91 || key == 93)

ui->edtInputFormula->setCurrentFont(QFont("Comic Sans MS", 12));

break;

}

case QEvent::FocusIn:

{

gEditFocusIn(\*(ui->edtInputFormula));

break;

}

case QEvent::FocusOut:

{

gEditFocusOut(\*(ui->edtInputFormula));

ui->edtInputFormula->clearFocus();

break;

}

}

}

return QObject::eventFilter(watched, event);

}

void ChemHelper::on\_twSearchResults\_cellClicked(int row, int column)

{

const char \*url = ui->twSearchResults->item(row, 1)->text().toUtf8();

const char \*open = "open";

ShellExecuteA(NULL, open, url , NULL, NULL, SW\_HIDE);

}

Графический модуль:

ch\_graphics.h:

#ifndef CHGRAPHICS\_H

#define CHGRAPHICS\_H

#include <QGraphicsView>

#include <QLabel>

#include <QLayout>

#include <QTableWidget>

void setGEditFinishPosition(QGraphicsView &Underscore);

void setGEditStartPosition(QGraphicsView &Underscore);

void setAllFinishPosition(QWidget &GEdit, QLabel &Logo, QTableWidget &SearchResults);

void setAllStartPosition(QWidget &GEdit, QLabel &Logo, QTableWidget &SearchResults);

void redraw(QWidget &GEdit, QGraphicsView &Underscore, QLabel &Logo, QTableWidget &SearchResults);

//void showMessageInvalidInputFormula();

#endif // CHGRAPHICS\_H

ch\_graphics.cpp:

#include <windows.h>

#include "ch\_graphics.h"

#include "ch\_variables\_and\_types.h"

void setGEditFinishPosition(QGraphicsView &Underscore)

{

Underscore.setGeometry(10, EdtInputFormulaHeight + 11, EdtInputFormulaWidth, 4);

}

void setGEditStartPosition(QGraphicsView &Underscore)

{

Underscore.setGeometry(10, EdtInputFormulaHeight + 10, 0, 4);

}

void setAllFinishPosition(QWidget &GEdit, QLabel &Logo, QTableWidget &SearchResults)

{

GEdit.move(GEditFinishPosition.left, GEditFinishPosition.top);

Logo.move(LogoFinishPosition.left, LogoFinishPosition.top);

SearchResults.setGeometry(SrchResStartPosition.left, SrchResStartPosition.top, SrchResWidth, SrchResFinishHeight);

}

void setAllStartPosition(QWidget &GEdit, QLabel &Logo, QTableWidget &SearchResults)

{

GEdit.move(GEditStartPosition.left, GEditStartPosition.top);

Logo.move(LogoStartPosition.left, LogoStartPosition.top);

SearchResults.setGeometry(SrchResStartPosition.left, SrchResStartPosition.top, SrchResWidth, SrchResStartHeight);

}

void redraw(QWidget &GEdit, QGraphicsView &Underscore, QLabel &Logo, QTableWidget &SearchResults)

{

SYSTEMTIME UTC;

int now = 0;

GetSystemTime(&UTC);

now = sysTimeToInt(UTC);

if (GEditAnimationRunning)

{

int underscoreWidth;

if (now - TimeGEditAnimationStart <= GEditAnimationTime / 2)

underscoreWidth = GEditUnderscoreAcceleration \* sqr(now - TimeGEditAnimationStart );

else

underscoreWidth = EdtInputFormulaWidth - GEditUnderscoreAcceleration \* sqr(TimeGEditAnimationStart + GEditAnimationTime - now);

Underscore.setGeometry(10 + EdtInputFormulaWidth/2 - underscoreWidth/2, EdtInputFormulaHeight + 11, underscoreWidth, 4);

}

if (GEditInverseAnimationRunning)

{

int underscoreWidth;

if (now - TimeGEditInverseAnimationStart < GEditAnimationTime / 2)

underscoreWidth = EdtInputFormulaWidth - GEditUnderscoreAcceleration \* sqr(now - TimeGEditInverseAnimationStart );

else

underscoreWidth = GEditUnderscoreAcceleration \* sqr(TimeGEditInverseAnimationStart + GEditAnimationTime - now );

Underscore.setGeometry(10 + EdtInputFormulaWidth/2 - underscoreWidth/2, EdtInputFormulaHeight + 11, underscoreWidth, 4);

}

if (AllAnimationRunning)

if (now - TimeAllAnimationStart <= AllAnimationTime / 2)

{

GEdit.move(GEditStartPosition.left + GEditLeftAcceleration \* sqr(now - TimeAllAnimationStart) / 2,

GEditStartPosition.top + GEditTopAcceleration \* sqr(now - TimeAllAnimationStart) / 2);

Logo.move(LogoStartPosition.left + LogoLeftAcceleration \* sqr(now - TimeAllAnimationStart) / 2,

LogoStartPosition.top + LogoTopAcceleration \* sqr(now - TimeAllAnimationStart) / 2);

SearchResults.setGeometry(SrchResStartPosition.left, SrchResStartPosition.top,

SrchResWidth, SGAcceleration \* sqr(now - TimeAllAnimationStart) / 2);

}

else

{

GEdit.move(GEditFinishPosition.left - GEditLeftAcceleration \* sqr(TimeAllAnimationStart + AllAnimationTime - now) / 2,

GEditFinishPosition.top - GEditTopAcceleration \* sqr(TimeAllAnimationStart + AllAnimationTime - now) / 2);

Logo.move(LogoFinishPosition.left - LogoLeftAcceleration \* sqr(TimeAllAnimationStart + AllAnimationTime - now) / 2,

LogoFinishPosition.top - LogoTopAcceleration \* sqr(TimeAllAnimationStart + AllAnimationTime - now) / 2);

SearchResults.setGeometry(SrchResStartPosition.left, SrchResStartPosition.top,

SrchResWidth, SrchResFinishHeight - SGAcceleration \* sqr(TimeAllAnimationStart + AllAnimationTime - now) / 2);

}

if (AllInverseAnimationRunning)

if (now - TimeAllInverseAnimationStart <= AllAnimationTime / 2)

{

GEdit.move(GEditFinishPosition.left + GEditLeftAcceleration \* sqr(now - TimeAllInverseAnimationStart) / 2,

GEditFinishPosition.top + GEditTopAcceleration \* sqr(now - TimeAllInverseAnimationStart) / 2);

Logo.move(LogoStartPosition.left + LogoLeftAcceleration \* sqr(now - TimeAllInverseAnimationStart) / 2,

LogoStartPosition.top + LogoTopAcceleration \* sqr(now - TimeAllInverseAnimationStart) / 2);

SearchResults.setGeometry(SrchResStartPosition.left, SrchResStartPosition.top,

SrchResWidth, SrchResFinishHeight - SGAcceleration \* sqr(now - TimeAllInverseAnimationStart) / 2);

}

else

{

GEdit.move(GEditFinishPosition.left - GEditLeftAcceleration \* sqr(TimeAllInverseAnimationStart + AllAnimationTime - now) / 2,

GEditFinishPosition.top - GEditTopAcceleration \* sqr(TimeAllInverseAnimationStart + AllAnimationTime - now) / 2);

//strngrdSearchResults.Height = - SGAcceleration \* Sqr( TimeAllInverseAnimationStart + AllAnimationTime - Now ) / 2;

Logo.move(LogoFinishPosition.left - LogoLeftAcceleration \* sqr(TimeAllInverseAnimationStart + AllAnimationTime - now) / 2,

LogoFinishPosition.top - LogoTopAcceleration \* sqr(TimeAllInverseAnimationStart + AllAnimationTime - now) / 2);

SearchResults.setGeometry(SrchResStartPosition.left, SrchResStartPosition.top,

SrchResWidth, SGAcceleration \* sqr(TimeAllInverseAnimationStart + AllAnimationTime - now) / 2);

}

}

Модуль модели:

ch\_model.h:

#ifndef CH\_MODEL\_H

#define CH\_MODEL\_H

#include <string>

#include <QTextEdit>

#include <QGraphicsView>

#include <QLabel>

#include <QTableWidget>

std::string normallize(const std::string &inputFormula);

bool isValidChemical(const std::string &someFormula );

void gEditFocusIn(QTextEdit &edtInputFormula);

void gEditFocusOut(QTextEdit &edtInputFormula);

void launchGEditAnimation();

void launchGEditInverseAnimation();

void launchAllAnimation();

void launchAllInverseAnimation();

void gEditAnimationFinished(QGraphicsView &Underscore);

void gEditInverseAnimationFinished(QGraphicsView &Underscore);

void allAnimationFinished(QWidget &GEdit, QLabel &Logo, QTableWidget &SearchResults);

void allInverseAnimationFinished(QWidget &GEdit, QLabel &Logo, QTableWidget &SearchResults);

void needRedraw(QWidget &GEdit, QGraphicsView &Underscore, QLabel &Logo, QTableWidget &SearchResults);

void initializeVariables(QWidget &GEdit, QGraphicsView &Underscore, QLabel &Logo, QTableWidget &SearchResults);

#endif // A\_H

ch\_model.cpp:

#include <sstream>

#include <stack>

#include <QNetworkAccessManager>

#include <QNetworkReply>

#include <QEventLoop>

#include "ch\_model.h"

#include "ch\_variables\_and\_types.h"

#include "ch\_graphics.h"

using namespace std;

string intToStr(int x)

{

stringstream ss;

string str;

ss << x;

ss >> str;

return str;

}

int strToInt(string str)

{

stringstream ss;

int x;

ss << str;

ss >> x;

return x;

}

string normallize(const string &inputFormula)

{

string result;

for (size\_t i = 0; i < inputFormula.size(); i++)

{

switch (inputFormula[i])

{

case ' ':

result.push\_back('%');

result.push\_back('2');

result.push\_back('0');

break;

case '[':

result.push\_back('(');

break;

case ']':

result.push\_back(')');

break;

default:

result.push\_back(inputFormula[i]);

}

// Add case if ((SO4))2;

}

return result;

}

// ToDo

/\*function CheckForGydrate(var InputFormula: string): Boolean;

var

i: Integer;

begin

for i := 1 to Length(InputFormula) do

begin

end;

end;\*/

bool isNumber(char x)

{

return (x >= '0' && x <= '9');

}

bool isLetter(char x)

{

return (x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z');

}

int getIntegerBeforePosition(const string &str, int &i)

{

int result = 0;

int numberStartPosition = 0, numberFinishPosition = 0;

if(isNumber(str[i]))

{

numberFinishPosition = i;

while ((i >= 0) && isNumber(str[i]))

i--;

i++;

numberStartPosition = i;

while (i <= numberFinishPosition)

{

result \*= 10;

result += str[i] - '0';

i++;

}

i = numberStartPosition - 1;

}

if ( result == 0 )

result++;

return result;

}

chemElement findElement(const string &someElement)

{

int i = 0;

for (size\_t i = 0; i < ElementList.size(); i++)

if (ElementList[i].formula == someElement)

return ElementList[i];

return chemElement();

}

chemElement getLatestElement(const string &inputFormula, int &i)

{

chemElement result;

int someQuantity = 0;

someQuantity = getIntegerBeforePosition( inputFormula, i );

if (i > 0)

result = findElement(inputFormula.substr(i - 1, 2));

if (result.formula == "")

result = findElement(inputFormula.substr(i, 1));

if (inputFormula[i] == '(' || inputFormula[i] == ')')

{

result.formula = inputFormula[i];

result.name = "";

}

if (result.formula != "")

{

result.quantity = someQuantity;

i -= result.formula.size();

}

else

i = -1;

return result;

}

vector<chemElement> cutToIons(const string &inputFormula)

{

vector<chemElement> result;

int i = inputFormula.size() - 1, factor = 1;

chemElement currentElement;

bool valid = true;

stack<int> allFactors;

while ((i>=0) && valid)

{

currentElement = getLatestElement(inputFormula, i);

if (currentElement.formula == "")

valid = false;

else

{

if (currentElement.formula == ")")

{

factor \*= currentElement.quantity;

allFactors.push(currentElement.quantity);

}

if (currentElement.formula == "(")

{

if (!allFactors.empty())

{

factor /= allFactors.top();

allFactors.pop();

}

else

valid = false;

}

if (currentElement.name != "")

{

currentElement.quantity \*= factor;

result.push\_back(currentElement);

}

}

}

if (!valid)

result.clear();

return result;

}

bool checkOxidationDegreeses(const vector<chemElement> &ionArray, int currentOxidationDegree, size\_t currentPosition)

{

bool result = false;

if (currentPosition >= ionArray.size())

return (currentOxidationDegree == 0);

for (int i = 0; i < 17; i++)

{

if (ionArray[currentPosition].degreeOfOxidation[i] != -1)

result |= checkOxidationDegreeses(ionArray, currentOxidationDegree + (i - 8)\*ionArray[currentPosition].quantity, currentPosition + 1);

}

return result;

}

bool isValidChemical(const string &someFormula )

{

int i = 0;

vector<chemElement> ionArray;

ionArray = cutToIons(someFormula);

return (ionArray.size() >= 1? (ionArray.size() > 1? checkOxidationDegreeses(ionArray, 0, 0) : true) : false);

}

/\*function Interpreter(InputFormula: string): string;

var

InputKation, InputAnion: TIonsArray;

begin

InputAnion := CutToIons(GetAnionCore, InputFormula);

InputKation := CutToIons(GetKationCore, InputFormula);

//GetHtmlPage()

end;\*/

void launchGEditAnimation()

{

SYSTEMTIME UTC;

if (!WasGEditAnimation)

{

GetSystemTime(&UTC);

GEditAnimationRunning = true;

TimeGEditAnimationStart = sysTimeToInt(UTC);

WasGEditAnimation = true;

}

}

void launchGEditInverseAnimation()

{

SYSTEMTIME UTC;

if (WasGEditAnimation)

{

GetSystemTime(&UTC);

GEditInverseAnimationRunning = true;

TimeGEditInverseAnimationStart = sysTimeToInt(UTC);

WasGEditAnimation = false;

//Form1.FocusControl(NULL);

}

}

void launchAllAnimation()

{

SYSTEMTIME UTC;

if (!WasAllAnimation)

{

GetSystemTime(&UTC);

AllAnimationRunning = true;

TimeAllAnimationStart = sysTimeToInt(UTC);

WasAllAnimation = true;

}

}

void launchAllInverseAnimation()

{

SYSTEMTIME UTC;

if (WasAllAnimation)

{

GetSystemTime(&UTC);

AllInverseAnimationRunning = true;

TimeAllInverseAnimationStart = sysTimeToInt(UTC);

WasAllAnimation = false;

}

}

void gEditFocusIn(QTextEdit &edtInputFormula)

{

launchGEditAnimation();

if(edtInputFormula.toPlainText().toStdString() == "Ваша формула")

edtInputFormula.setText("");

}

void gEditFocusOut(QTextEdit &edtInputFormula)

{

launchGEditInverseAnimation();

if(edtInputFormula.toPlainText().toStdString() == "")

edtInputFormula.setText("Ваша формула");

}

void gEditAnimationFinished(QGraphicsView &Underscore)

{

GEditAnimationRunning = false;

setGEditFinishPosition(Underscore);

}

void gEditInverseAnimationFinished(QGraphicsView &Underscore)

{

GEditInverseAnimationRunning = false;

setGEditStartPosition(Underscore);

}

void allAnimationFinished(QWidget &GEdit, QLabel &Logo, QTableWidget &SearchResults)

{

AllAnimationRunning = false;

setAllFinishPosition(GEdit, Logo, SearchResults);

}

void allInverseAnimationFinished(QWidget &GEdit, QLabel &Logo, QTableWidget &SearchResults)

{

AllInverseAnimationRunning = false;

setAllStartPosition(GEdit, Logo, SearchResults);

}

void needRedraw(QWidget &GEdit, QGraphicsView &Underscore, QLabel &Logo, QTableWidget &SearchResults)

{

redraw(GEdit, Underscore, Logo, SearchResults);

}

void initializeVariables(QWidget &GEdit, QGraphicsView &Underscore, QLabel &Logo, QTableWidget &SearchResults)

{

GEdit.move(GEditStartPosition.left, GEditStartPosition.top);

Logo.move(LogoStartPosition.left, LogoStartPosition.top);

SearchResults.setGeometry(SrchResStartPosition.left, SrchResStartPosition.top, SrchResWidth, SrchResStartHeight);

SearchResults.setColumnWidth(0, 385);

SearchResults.setColumnWidth(1, 385);

ElementList = getChemElementArrayFromFile( "data\\files\\Elements.txt" );

ExceptionList = getStringFromFile( "data\\files\\ExceptionList.txt" );

PathToBrowser = getStringFromFile( "data\\files\\PathToBrowser.txt" );

CiteArray = getStringArrayFromFile( "data\\files\\Cites.txt" );

//string htmlh2o = getStringFromFile("data\\files\\HTMLH2O.txt");

//string htmlhcl = getStringFromFile("data\\files\\HTMLHCl.txt");

//string htmlal2so43 = getStringFromFile("data\\files\\HTMLAl2SO43.txt");

HTMLH2O = getWStringArrayFromFile("data\\files\\HTMLH2O.txt"); //wstring(htmlh2o.begin(), htmlh2o.end());

HTMLHCl = getWStringArrayFromFile("data\\files\\HTMLHCl.txt"); //wstring(htmlhcl.begin(), htmlhcl.end());

HTMLAl2SO43 = getWStringArrayFromFile("data\\files\\HTMLAl2SO43.txt"); //wstring(htmlal2so43.begin(), htmlal2so43.end());

}

Модуль переменных и типов:

ch\_variables\_and\_types.h:

#ifndef CHVARIABLESANDTYPES\_H

#define CHVARIABLESANDTYPES\_H

#include <string>

#include <vector>

#include <QLabel>

#include <windows.h>

#include <iostream>

class chemElement

{

public:

std::string formula;

std::vector<int> degreeOfOxidation;

std::string name;

int quantity;

chemElement () : formula("") {}

chemElement (const chemElement &x) :

formula(x.formula), degreeOfOxidation(x.degreeOfOxidation), name(x.name), quantity(x.quantity) {}

chemElement (std::string \_formula, std::string \_degreeOfOxidation, std::string \_name, int \_quantity) :

formula(\_formula), name(\_name), quantity(\_quantity)

{

for(size\_t i = 0; i < \_degreeOfOxidation.size(); i++)

degreeOfOxidation.push\_back(\_degreeOfOxidation[i] == '-'? -1 : \_degreeOfOxidation[i] - '0');

}

};

struct position

{

int left;

int top;

position () {}

position (const position &x) : left(x.left), top(x.top) {}

position (const int \_left, const int \_top) : left(\_left), top(\_top) {}

};

class QClickableLabel : public QLabel

{

Q\_OBJECT

public:

QClickableLabel(QWidget \*parent = nullptr) : QLabel(parent) {}

QClickableLabel(QString text, QWidget \*parent = nullptr) : QLabel(text, parent) {}

~QClickableLabel() {}

signals:

void clicked();

protected:

void mousePressEvent(QMouseEvent \*event) { emit clicked(); }

};

std::vector<chemElement> getChemElementArrayFromFile(std::string path);

std::vector<std::string> getStringArrayFromFile(std::string path);

std::vector<std::wstring> getWStringArrayFromFile(std::string path);

std::string getStringFromFile(std::string path);

std::wstring getWStringFromFile(std::string path);

std::wstring stringToWString(std::string str);

int sysTimeToInt(SYSTEMTIME x);

int sqr(int a);

bool isKNumber(int code);

bool isKLetter(int code);

std::wstring backNSymbols( int i, int n, const std::wstring &someStr);

std::wstring forwardNSymbols(int i, int n, const std::wstring &someStr);

std::wstring shiftUntilX(int& i, char x, const std::wstring &someStr);

extern const position GEditStartPosition, GEditFinishPosition;

extern const position LogoStartPosition, LogoFinishPosition;

extern const position SrchResStartPosition;

extern int MonitorHeight;

extern int MonitorWidth;

extern const int EdtInputFormulaWidth, EdtInputFormulaHeight;

extern const int SrchResWidth, SrchResStartHeight;

extern int SrchResFinishHeight;

extern int btnSearchWidth;

extern bool GEditAnimationRunning;

extern bool GEditInverseAnimationRunning;

extern bool AllAnimationRunning;

extern bool AllInverseAnimationRunning;

extern bool WasGEditAnimation;

extern bool WasAllAnimation;

extern bool WasChangedInputFormula;

extern bool WasJustCreated;

extern std::vector<chemElement> ElementList;

extern std::string ExceptionList;

extern std::string PathToBrowser;

extern std::vector<std::string> CiteArray;

extern std::vector<std::wstring> HTMLH2O;

extern std::vector<std::wstring> HTMLHCl;

extern std::vector<std::wstring> HTMLAl2SO43;

extern int PrevTime;

extern int TimeGEditAnimationStart;

extern int TimeGEditInverseAnimationStart;

extern int TimeAllAnimationStart;

extern int TimeAllInverseAnimationStart;

extern const int GEditAnimationTime;

extern const int AllAnimationTime;

extern const double StartAcceleration;

extern const int StartSpeed;

extern const double CurrentAccelerationCoefficient, CurrentSpeed;

extern const double GEditLeftAcceleration, GEditTopAcceleration;

extern const double SGAcceleration;

extern const double GEditUnderscoreAcceleration;

extern const double LogoLeftAcceleration, LogoTopAcceleration;

extern const QColor UnderscoreCorrectColor;

extern const QColor UnderscoreWrongColor;

#endif // CHVARIABLESANDTYPES\_H

ch\_variables\_and\_types.cpp:

#include <iostream>

#include <QDesktopWidget>

#include <QKeyEvent>

#include <QGraphicsView>

#include <QGraphicsScene>

#include <cmath>

#include <windows.h>

#include "chemhelper.h"

#include "ui\_chemhelper.h"

#include "ch\_variables\_and\_types.h"

#include "ch\_model.h"

using namespace std;

ChemHelper::ChemHelper(QWidget \*parent) : QMainWindow(parent), ui(new Ui::ChemHelper), tmrProcessTimig(new QTimer), nam(new QNetworkAccessManager)

{

MonitorWidth = QApplication::desktop()->width();

MonitorHeight = QApplication::desktop()->height();

btnSearchWidth = min(MonitorWidth, MonitorHeight) / 30;

ui->setupUi(this);

tmrProcessTimig->setInterval(15);

connect(tmrProcessTimig, SIGNAL(timeout()), this, SLOT(update()));

tmrProcessTimig->start();

ui->gviewUnderscore->setScene(new QGraphicsScene);

ui->gviewUnderscore->setBackgroundBrush(QBrush(UnderscoreCorrectColor, Qt::SolidPattern));

ui->ltGEdit->setAlignment(ui->gviewUnderscore, Qt::AlignHCenter);

QString filename = "..\\data\\images\\SearchButton.jpg";

QClickableLabel \*lblSearchButton = new QClickableLabel(this);

lblSearchButton->setAlignment(Qt::AlignCenter);

lblSearchButton->setSizePolicy(QSizePolicy::Preferred, QSizePolicy::Fixed);

lblSearchButton->setFixedSize(btnSearchWidth, btnSearchWidth);

lblSearchButton->setScaledContents(true);

lblSearchButton->setCursor(Qt::PointingHandCursor);

//lblSearchButton->setGeometry(100, 200, 0, 0);

connect(lblSearchButton, SIGNAL(clicked()), this, SLOT(on\_btnSearch\_clicked()));

QPixmap pix;

if(pix.load(filename))

lblSearchButton->setPixmap(pix);

ui->ltGEB->addWidget(lblSearchButton);

ui->twSearchResults->setEditTriggers(QAbstractItemView::NoEditTriggers);

qApp->installEventFilter(this);

initializeVariables(\*(ui->GEdit), \*(ui->gviewUnderscore), \*(ui->lblLogo), \*(ui->twSearchResults));

ui->edtInputFormula->clearFocus();

}

ChemHelper::~ChemHelper()

{

delete ui;

delete tmrProcessTimig;

}

void ChemHelper::update()

{

SYSTEMTIME UTC;

GetSystemTime(&UTC);

long long now = sysTimeToInt(UTC);

sysTimeToInt(UTC);

if (now - TimeGEditAnimationStart > GEditAnimationTime && GEditAnimationRunning)

gEditAnimationFinished(\*(ui->gviewUnderscore));

if (now - TimeGEditInverseAnimationStart > GEditAnimationTime && GEditInverseAnimationRunning)

gEditInverseAnimationFinished(\*(ui->gviewUnderscore));

if (now - TimeAllAnimationStart > AllAnimationTime && AllAnimationRunning)

allAnimationFinished(\*(ui->GEdit), \*(ui->lblLogo), \*(ui->twSearchResults));

if (now - TimeAllInverseAnimationStart > AllAnimationTime && AllInverseAnimationRunning)

allInverseAnimationFinished(\*(ui->GEdit), \*(ui->lblLogo), \*(ui->twSearchResults));

if(GEditAnimationRunning || GEditInverseAnimationRunning || AllAnimationRunning || AllInverseAnimationRunning)

needRedraw(\*(ui->GEdit), \*(ui->gviewUnderscore), \*(ui->lblLogo), \*(ui->twSearchResults));

}

wstring vectorWStringToWString(const vector<wstring> &src)

{

wstring result = L"";

for(int i = 0; i < src.size(); i++)

result.append(src[i]);

return result;

}

void ChemHelper::fillTWSearchResults(const std::wstring &HTMLCode)

{

int i = 0;

bool divgOpened = false, h3Opened = false, aOpened = false, citeOpened = false;

int countDivOpened = 0;

vector<wstring> titles, URLs;

while (i < HTMLCode.size())

{

//wcout<<HTMLCode[i]<<endl;

//ShowMessage(IntToStr(i) + backNSymbols(i, 6, HTMLCode));

if (divgOpened && forwardNSymbols(i, 6, HTMLCode) == L"</div>" && countDivOpened == 1)

{

divgOpened = false;

countDivOpened--;

}

if (countDivOpened > 0 && forwardNSymbols(i, 6, HTMLCode) == L"</div>")

countDivOpened--;

if (citeOpened && HTMLCode[i] == '<')

{

if (forwardNSymbols(i, 7, HTMLCode) == L"</cite>")

citeOpened = false;

else

{

shiftUntilX(i, '>', HTMLCode);

i++;

}

}

if (citeOpened)

{

URLs.push\_back(shiftUntilX(i, '<', HTMLCode));

i--;

}

if (divgOpened && backNSymbols(i, 5, HTMLCode) == L"<cite")

{

shiftUntilX(i, '>', HTMLCode);

citeOpened = true;

}

if (aOpened && forwardNSymbols(i, 4, HTMLCode) == L"</a>")

aOpened = false;

if (h3Opened && HTMLCode[i] == '<')

{

if (forwardNSymbols(i, 5, HTMLCode) == L"</h3>")

h3Opened = false;

else

{

shiftUntilX(i, '>', HTMLCode);

i++;

}

}

if (h3Opened)

{

titles.push\_back(shiftUntilX(i, '<', HTMLCode));

if ( titles.back().size() > 30 )

{

titles.back().erase(30, titles.size() - 30);

titles.back() += L"...";

aOpened = false;

}

i--;

}

if (aOpened && backNSymbols(i, 3, HTMLCode) == L"<h3")

{

shiftUntilX(i, '>', HTMLCode);

h3Opened = true;

}

if (divgOpened && backNSymbols(i, 2, HTMLCode) == L"<a")

{

shiftUntilX(i, '>', HTMLCode);

aOpened = true;

}

if (countDivOpened > 0 && backNSymbols(i, 4, HTMLCode) == L"<div")

countDivOpened++;

if (backNSymbols(i, 14, HTMLCode) == L"<div class=\"g\"")

{

//cout<<"kek"<<endl;

shiftUntilX( i, '>', HTMLCode );

divgOpened = true;

countDivOpened = 1;

}

i++;

}

/\*for(int i = 0; i < titles.size(); i++)

{

wcout << QString::fromStdWString(titles[i]).toStdWString() << L" " << URLs[i] << endl;

}\*/

for (int i = 0; i < titles.size(); i++)

{

ui->twSearchResults->insertRow(ui->twSearchResults->rowCount());

ui->twSearchResults->setRowHeight(ui->twSearchResults->rowCount() - 1, 50);

//titles[i] = L"ывамыфывфыdfsdfvsdfваьидла";

//titles[i] = QString::fromStdWString(titles[i]).toStdWString();

QTableWidgetItem \*x = new QTableWidgetItem(QString::fromStdWString(titles[i]));

//wcout << x->text().toStdWString() << endl;

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 0, x);

if (forwardNSymbols(0, 4, URLs[i]) == L"http")

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 1, new QTableWidgetItem(QString::fromStdWString(URLs[i])));

else

ui->twSearchResults->setItem(ui->twSearchResults->rowCount() - 1, 1, new QTableWidgetItem(QString::fromStdWString(L"http://" + URLs[i])));

}

}

void ChemHelper::searchFinished()

{

wstring HTML(reply->readAll().toStdString().begin(), reply->readAll().toStdString().end());

wcout<<HTML<<endl;

fillTWSearchResults(HTML);

}

void ChemHelper::sendRequest(const string &inputFormula)

{

ui->twSearchResults->clear();

ui->twSearchResults->setRowCount(0);

/\*for(int i = 0; i < CiteArray.size(); i++)

{ CHANGE TO I

reply = nam->get(QNetworkRequest(QUrl(QString::fromStdString("https://www.google.com/search?q=" + CiteArray[0] + inputFormula))));

}

QObject::connect(reply, SIGNAL(finished()), this, SLOT(searchFinished()));\*/

//cout<<inputFormula<<endl;

if(inputFormula == "H2O")

{

//cout<<"kek"<<endl;

//wcout<<vectorWStringToWString(HTMLH2O)<<endl;

fillTWSearchResults(vectorWStringToWString(HTMLH2O));

}

if(inputFormula == "HCl")

{

fillTWSearchResults(vectorWStringToWString(HTMLHCl));

}

if(inputFormula == "Al2(SO4)3")

{

fillTWSearchResults(vectorWStringToWString(HTMLAl2SO43));

}

//SrchResFinishHeight = min(SrchResFinishHeight, ui->twSearchResults->rowCount() \* 50 + 20);

launchAllAnimation();

}

void ChemHelper::on\_btnSearch\_clicked()

{

if(isValidChemical(ui->edtInputFormula->toPlainText().toStdString()))

{

ui->edtInputFormula->clearFocus();

//cout<<ui->edtInputFormula->toPlainText().toStdString()<<endl;

sendRequest(normallize(ui->edtInputFormula->toPlainText().toStdString()));

}

else

{

ui->gviewUnderscore->setBackgroundBrush(QBrush(UnderscoreWrongColor, Qt::SolidPattern));

WasGEditAnimation = false;

launchGEditAnimation();

WasChangedInputFormula = false;

}

}

bool isValidKey(int key)

{

/\*

\* 16777220 - enter

\* 16777216 - esc

\* 16777219 - backspase

\* 16777223 - del

\* 16777(234, 235, 236, 237) - arrows (l, u, r, d)

\* 40 .. 43 - '(', ')', '\*', '+'

\* 61 - '='

\* 32 - ' '

\* 91 - '['

\* 93 - ']'

\*/

return (isKNumber(key) || isKLetter(key) ||

key == 16777220 || key == 16777216 || key == 16777219 || key == 16777223 ||

key == 16777234 || key == 16777235 || key == 16777236 || key == 16777237 ||

(key >= 40 && key <= 43) || key == 61 || key == 32 || key == 91 || key == 93);

}

bool ChemHelper::eventFilter(QObject \*watched, QEvent \*event)

{

if(watched == ui->edtInputFormula)

{

switch (event->type())

{

case QEvent::KeyPress :

{

int key = (static\_cast<QKeyEvent\*>(event))->key();

//cout<<key<<endl;

if(!isValidKey(key))

return true;

if(key == 16777220) // enter

{

emit(on\_btnSearch\_clicked());

return true;

}

if(key == 16777216) //esc

QCoreApplication::postEvent(watched, new QEvent(QEvent::FocusOut));

else

{

ui->gviewUnderscore->setBackgroundBrush(QBrush(UnderscoreCorrectColor, Qt::SolidPattern));

needRedraw(\*(ui->GEdit), \*(ui->gviewUnderscore), \*(ui->lblLogo), \*(ui->twSearchResults));

}

if(isKNumber(key))

ui->edtInputFormula->setCurrentFont(QFont("Comic Sans MS", 6));

if(isKLetter(key) || (key >= 40 && key <= 43) || key == 61 || key == 32 || key == 91 || key == 93)

ui->edtInputFormula->setCurrentFont(QFont("Comic Sans MS", 12));

break;

}

case QEvent::FocusIn:

{

gEditFocusIn(\*(ui->edtInputFormula));

break;

}

case QEvent::FocusOut:

{

gEditFocusOut(\*(ui->edtInputFormula));

ui->edtInputFormula->clearFocus();

break;

}

}

}

return QObject::eventFilter(watched, event);

}

void ChemHelper::on\_twSearchResults\_cellClicked(int row, int column)

{

const char \*url = ui->twSearchResults->item(row, 1)->text().toUtf8();

const char \*open = "open";

ShellExecuteA(NULL, open, url , NULL, NULL, SW\_HIDE);

}