МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский  
Нижегородский государственный университет им. Н.И. Лобачевского»

(ННГУ)

Институт информационных технологий, математики и механики

ЛАБОРАТОРНАЯ РАБОТА

«Трансляция арифметических выражений»

Выполнил: студент группы 381603-4

Леонтьев Вадим Евгеньевич

Подпись \_\_\_\_\_\_\_\_\_\_\_\_

Руководитель:

Барышева Ирина Викторовна

Подпись \_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2017

Оглавление

[Введение. 3](#_Toc501885471)

[Польская инверсная запись 3](#_Toc501885472)

[Стек 3](#_Toc501885473)

[Таблица 3](#_Toc501885474)

[Постановка задачи 5](#_Toc501885475)

[Задача на лабораторную работу 5](#_Toc501885476)

[Возможные решения поставленной задачи 5](#_Toc501885477)

[Структура проекта 6](#_Toc501885478)

[Инструкция для пользователя 8](#_Toc501885479)

[Контрольный пример 10](#_Toc501885480)

[Заключение 11](#_Toc501885481)

[Список литературы 12](#_Toc501885482)

[Приложение 13](#_Toc501885483)

[Windows Form 13](#_Toc501885484)

[Программа с main 18](#_Toc501885485)

[Класс Zapis 18](#_Toc501885486)

[Класс MyStack 19](#_Toc501885487)

[Класс MyTablice 20](#_Toc501885488)

[Класс Pol 22](#_Toc501885489)

[Класс ArithmExp 24](#_Toc501885490)

[Метод strToMass 26](#_Toc501885491)

Введение.

Польская инверсная запись

Польская инверсная запись - форма записи [математических](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) и логических выражений, в которой [операнды](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4) расположены перед знаками [операций](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Эта запись очень удобна для вычисления арифметических выражений на машинах со стеками. Но для вычисления обратной польской записи в лабораторной работе также требуется таблица приоритетов операций, о которой я расскажу позже.

Главный недостаток обратной польской записи также проистекает из метода вычисления выражений в ней: поскольку используется стек, то для работы с ним всегда доступна только верхушка стека, а это делает крайне затруднительной оптимизацию выражений в форме обратной польской записи. Практически выражения в форме обратной польской записи почти не поддаются оптимизации.

Но для моей лабораторной работы не имеет большого значения оптимизация вычисления выражений, и в ней обратная польская запись оказывается очень удобным методом внутреннего представления программы.

Лабораторная работа выполнена с использованием визуального программирования.

Стек

Стек – это динамическая структура данных, представляющий собой [список элементов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), в которой запись «взять» и «положить» организованна по правилу: «last in - first out» («последним пришёл — первым вышел»).

Программный вид стека используется для обхода структур данных, например, дерево или граф. При использовании рекурсивных функций также будет применяться стек, но аппаратный его вид. Кроме этих назначений, стек используется для организации стековой машины, реализующей вычисления в обратной инверсной записи, что и необходимо в моей лабораторной работе.

Для отслеживания точек возврата из подпрограмм используется стек вызовов.

Арифметические сопроцессоры, программируемые микрокалькуляторы используют стековую модель вычислений.

Какой же структурой данных описывается стек? Я опишу 2 способа реализации программного стека:

1. Массив

Для структуры «Стек» можно использовать массив данных, необходимого, для конкретной задачи, типа. Который будет иметь размер – size, и поле – last, которое указывает на последующий (свободный для записи) элемент массива.

1. Линейный список

Для структуры «Стек» так же можно использовать структуру линейного списка. Структура линейного списка состоит из двух полей: информации и указатель на следующий элемент этого же списка.

Таблица

Таблица – это динамическая структура данных, которое представляет собой поля индификаторов (ключей), по которым осуществляется поиск по таблице, и связной с этим ключом информации.

С физической точки зрения таблица представляет собой вектор, элементами которого являются записи.

Характерной логической особенностью таблиц, которая определяет их отдельное рассмотрение, является то, что доступ к элементам таблицы производится не по номеру (индексу), а по ключу – по значению одного из свойств объекта, описываемого элементом таблицы.

Ключ – это свойство, идентифицирующее данную запись во множестве однотипных записей. Как правило, к ключу предъявляется требование уникальности в данной таблице. Ключ может включаться в состав записи и быть одним из ее полей, но может и не включаться в запись, а вычисляться по положению записи. Таблица может иметь один или несколько ключей.

Основной операцией при работе с таблицами является операция доступа к записи по ключу. Она реализуется процедурой поиска. Поскольку поиск может быть значительно более эффективным в таблицах, упорядоченных по значениям ключей, довольно часто над таблицами необходимо выполнять операции сортировки.

Постановка задачи

Задача на лабораторную работу

Целью данной лабораторной работы является демонстрация преобразования арифметического выражения в форму обратной польской записи, с последующим его вычислением, с использованием стека (программного стека).

Данная лабораторная работа должна быть выполнена с использование визуального программирования (в среде Windows Form - SLR). На которой необходимо присутствие поля для ввода арифметического выражения, поле с преобразованной из этого выражения обратной польской записи, таблицы переменных этого выражения, с возможностью изменения буквенных переменных и поле, в котором будет вычисленный ответ.

Исходные данные: строка с арифметическим выражением, таблица приоритетов операций.

В процессе преобразования арифметического выражения в обратную польскую запись должны получится данные типа: строки с обратной польской записью данного арифметического выражения, таблица переменных, содержащихся в этом выражении, благодаря которой будет высчитано это арифметическое выражение и записано в строку.

Описание алгоритма:

– Пока строка с арифметическим выражением не пуста, и если встречается:

1. Операнд, то записываю его в строку обратной польской записи
2. Операция, то если:
   1. Сравниваю текущую операцию с верхушкой стека (если есть), и пока у текущей операции приоритет «Хуже или такой же», как и на верхушке стека – то «вытесняю» операцию в строку с обратной польской записью
   2. Помещаю текущую операцию в стек
3. « ( », то записываю её в стек, с «наихудшим» приоритетом
4. « ) », то вытесняю всё до « ( » в строку с обратной польской записью, а её саму в никуда

– Выталкиваем из стека оставшиеся операции в строку с обратной польской записью

Возможные решения поставленной задачи

Структуру хранения стека можно представить как массив элементов, и как линейный список. Из практики в программировании, в большинстве случаев, стек не требует больше 20 элементов. Структура линейного списка представлена полем информации и указателем на следующий элемент массива, что требует дополнительной памяти. Поэтому в своей лабораторной работе я использую структуру хранения стека как «массив элементов».

Таблица переменных выглядит как: операция – приоритет (рисунок 1)

|  |  |
| --- | --- |
| + | 1 |
| - | 1 |
| \* | 2 |
| / | 2 |
| ^ | 3 |

Рисунок 1. Вид таблицы приоритетов операций

Благодаря выбору конкретных структур хранения данных, моя лабораторная работа не будет занимать много времени в подсчёте результата, и максимально эффективно будет использовать память.

Структура проекта

Для решения поставленной задачи разработан проект со структурой построения классов UML (Рисунок 2).

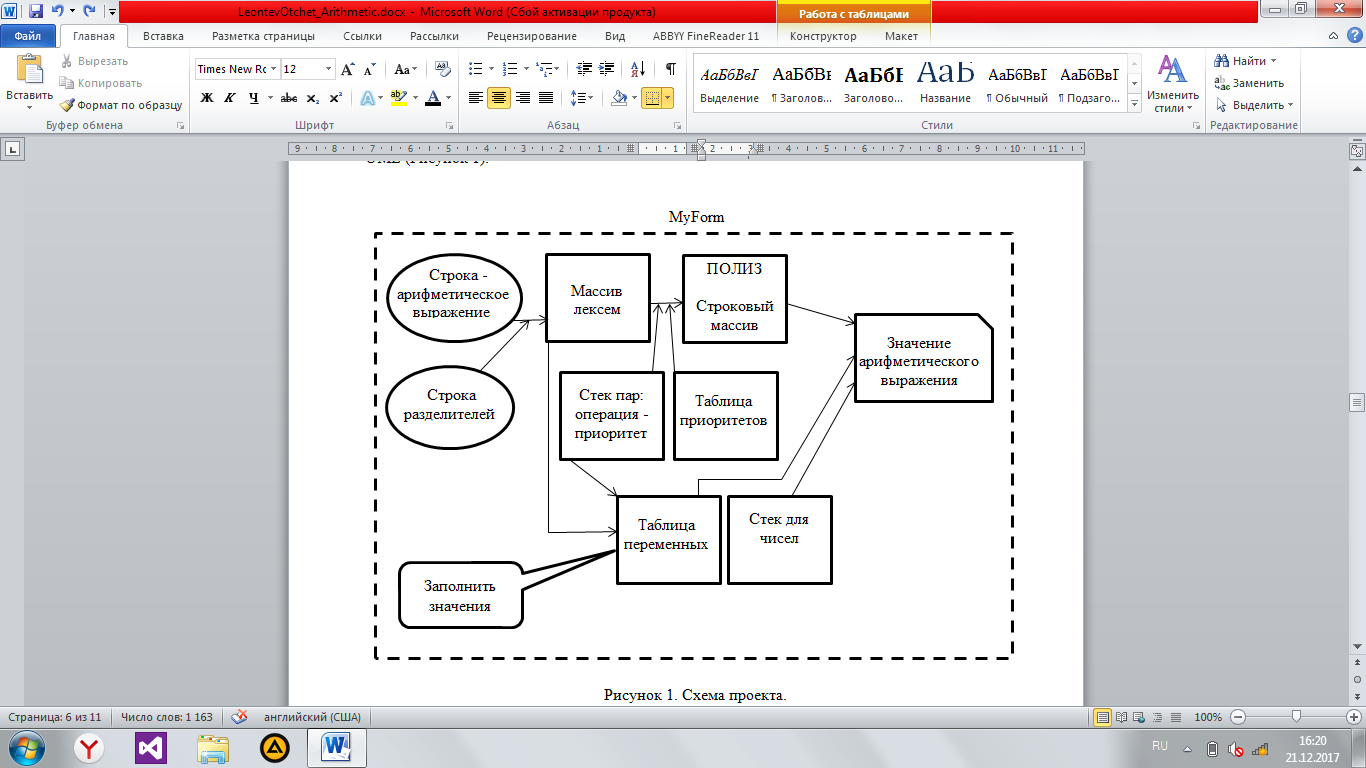


Рисунок 2. Схема проекта.

Проект располагается в форме (Windows Form) описание которой находится в Приложении «Windows Form». Моя лабораторная работа строиться на классах:

1. Zapis – в этом классе создаётся структура данных с полями: «Имя» и «Значения». Этот класс используется для структур: «стек» и «таблица». Класс Zapis описан в приложении «Класс Zapis».
2. MyStack – в этом классе создаётся динамическая структура данных, основанная на массиве элементов. Этот класс является классом на шаблоне, и в моей лабораторной работе это необходимо для вычисления обратной польской записи, с элементами класса Zapis. Класс MyStack подробно описывается в приложении «Класс MyStack».
3. MyTablice – в этом классе создаётся динамическая структура данных, которая содержит массив элементов класса Zapis. В лабораторной работе MyTablice используется для представления приоритета операций (вида: «операция» - «приоритет») и переменных арифметического выражения (вида: «переменная» - «значение»). Класс MyTablice описан в приложении «Класс MyTablice».
4. Pol – этот класс необходим для преобразования арифметического выражения в обратную польскую запись. В классе Pol описывается алгоритм создания обратной польской записи из стоки с арифметическим уравнением. Преобразование арифметического уравнения в обратную польскую запись требует использование класса MyStack и класса MyTablice с приоритетами операций. Класс Pol подробно описан в приложении «Класс Pol».
5. ArithmExp – в этом класс производятся основные (глобальные) методы над получаемой строкой арифметического выражения. Подробное описание этого класса приведено в приложении «Класс ArithmExp».

Метод strToMass – этот метод делит исходную строку арифметического выражения на массив лексем со знаками (в разных элементах этого массива) и без пробелов. Этот метод необходим для создания обратной польской записи.

Инструкция для пользователя

После запуска появиться окно программы (рисунок 3). В нём нужно ввести арифметическое выражение и нажать на кнопку «ОК».

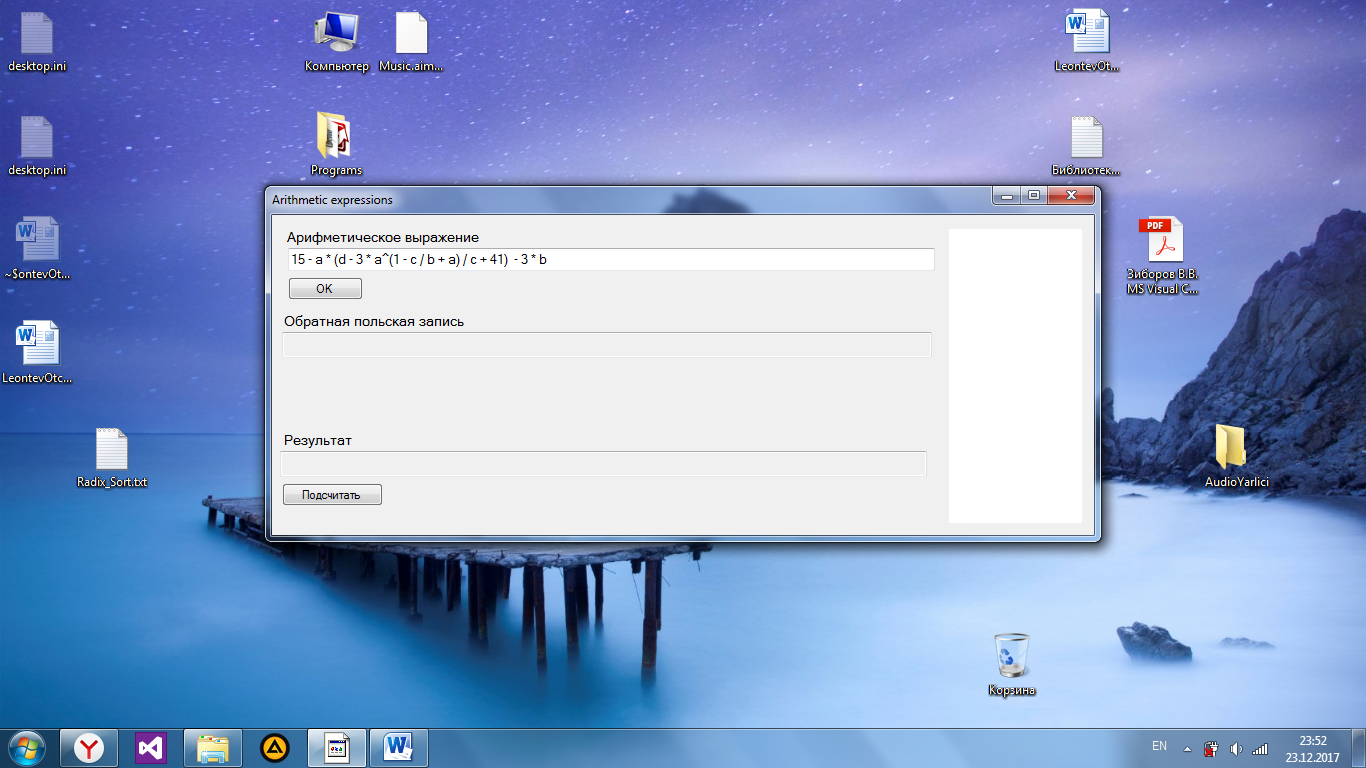


Рисунок 3. Окно программы с арифметическим выражением.

После чего арифметическое выражение будет преобразовано в обратную польскую запись и таблицу переменных, содержащихся во введённом выражении (рисунок 4).

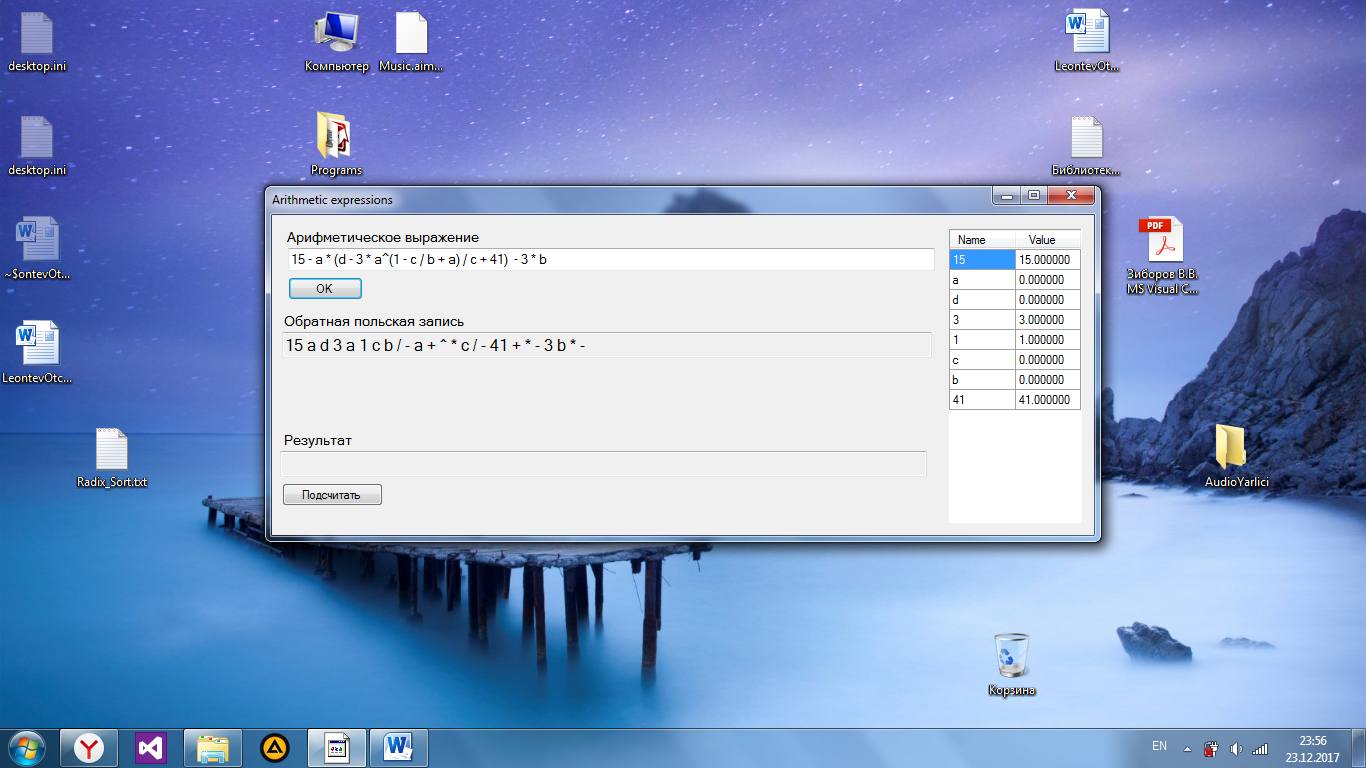


Рисунок 4. Окно программы с преобразованной польской записью и таблицей переменных

В таблице переменных, по умолчанию, все переменные равны нулю. Для подсчёта арифметического выражения надо заполнить таблицу переменных необходимыми значениями и нажать на кнопку «Подсчитать». После чего в поле «Результат» появится значение арифметического выражения с заданными значениями из таблице переменных (рисунок 5).

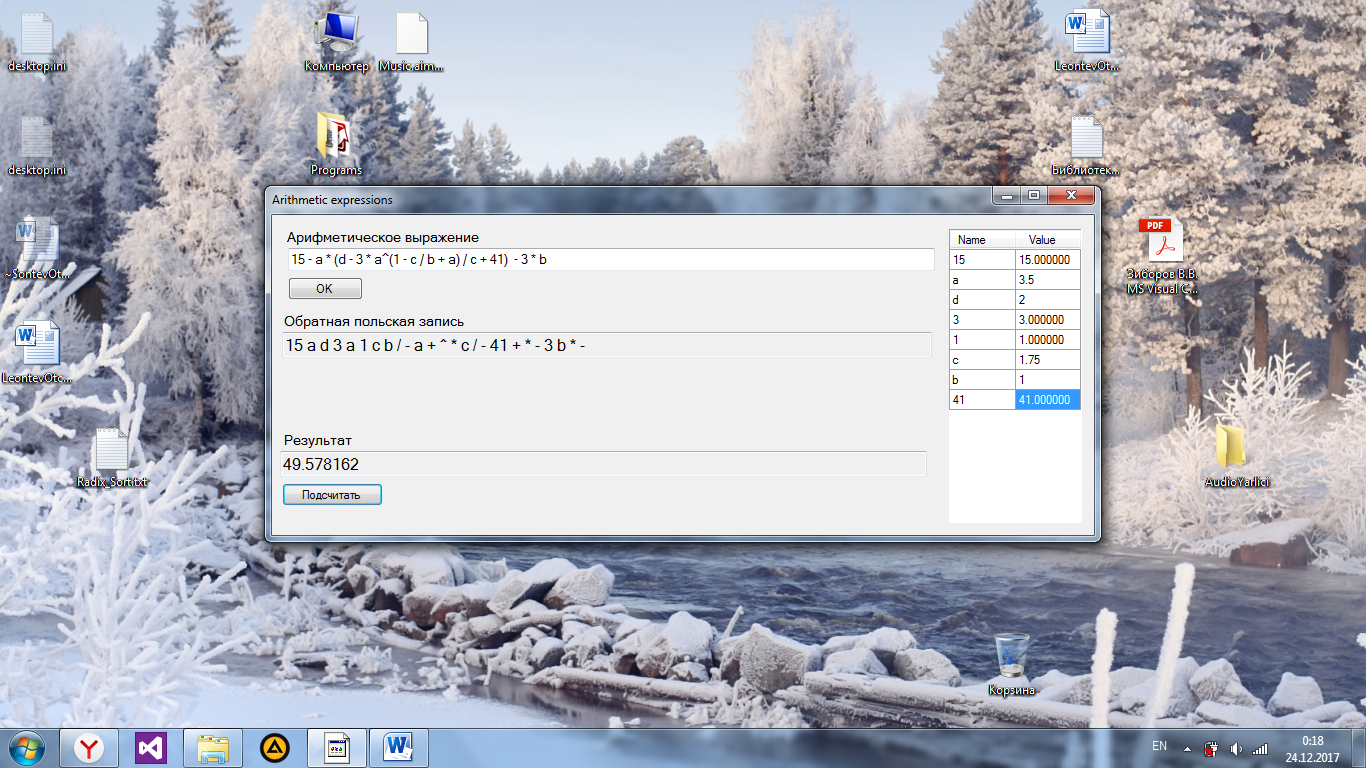


Рисунок 5. Окно программы с заполненной таблицей переменных и высчитанном результатом

Программа завершена, но так же можно ввести другое арифметическое выражение и выполнить все перечисленные действия уже для другого арифметического выражения. В конечном итоге будет высчитан результат для последнего введённого выражения.

Контрольный пример

Контрольный пример я провожу на:

Арифметическом выражении – « 15 – a \* ( d – 3 \* a^( 1 – c / b + a ) / c + 41 ) – 3 \* b »

Обратная польская запись этого выражения будет –

« 15 a d 3 a 1 c b / - a + ^ \* c / - 41 + \* - 3 b \* - », которая появляется в поле «Обратная польская запись». Так же с этим появляется таблица переменных этого выражения. После чего я присваиваю переменным значения (рисунок 6).

|  |  |
| --- | --- |
| Name | Value |
| 15 | 15.0000 |
| a | 3.5 |
| d | 2 |
| 3 | 3.0000 |
| 1 | 1.0000 |
| c | 1.75 |
| b | 1 |
| 41 | 41.0000 |

Рисунок 6. Таблица переменных.

Для вычисления выражения с заданными значениями я нажимаю кнопку «Подсчитать». В поле «Результат» появляется значение – 49.578162.

Это значение и есть результат вычисления данного арифметического выражения с заданными значениями.

Заключение

В данной лабораторной работе демонстрировалась работа стековой машины. Для этого был реализован, программным способом, класс MyStack, возможная структура хранения которого была рассмотрена во главе «Введение» и выбрана при постановке задачи. Описание программного стека расположено в приложении «Класс MyStack». Демонстрация машинного стека выполнена на вычислении арифметического выражения через обратную польскую запись.

Для работы с введённым арифметическим выражением в моей лабораторной работе потребовался класс Pol, который, в свою очередь, основан на классе MyTablice (с применением программного стека). Для вычисления обратной польской записи в классе Pol организованна таблица приоритетов (рисунок 1).

Структура «Таблицы» была выбрана не случайно, так как в этой структуре легко организовать поиск по «ключу», и использовать значение у ключа, для сравнения приоритетов операций. Также из-за надобности присваивания неизвестным (на момент ввода) переменным значения, очень удобно оказывается использовать структуру «Таблица». Подробно об этой структуре описывалось во главе «Введение».

Для удобства взаимодействия всех классов использовался метод strToMass, который разбивал введённое выражение в строковый массив лексем, что позволяет назначать имя переменным более одной буквы.

Для обеспечения инкапсуляции реализации лабораторной работы «выступил» класс ArithmExp, собравший во едино все классы и скрыв их реализацию. Этот класс оказался очень удобным для работы со средой Windows Form.

Слаженную работу и предсказуемость результата вычисления было проверено на примере « 15 – a \* ( d – 3 \* a^( 1 – c / b + a ) / c + 41 ) – 3 \* b » (значения переменных продемонстрированы на рисунке 6), с заранее вычисленным результатом (49.578162). В результате вычисления все промежуточные и конечный результаты были правильно подсчитанными.

Таким образом, в лабораторной работе продемонстрирована работа стековой машины. А благодаря среде Windows Form лабораторная работа, помимо правильной работы совокупности классов (что является приоритетной задачей данного проекта), имеет красивое оформление (рисунок 5).

Список литературы

1. Павловская Т. А. C/C++. Программирование на языке высокого уровня. — СПб.:Питер, 2003. – 461
2. Буч Г. Объектно-ориентированный анализ и проектирование с применением приложений. – М : «Вильямс», 2008 – 720 стр.
3. Шилдт Г.С. С++. Базовый курс. – М.: «Вильямс», 2010. – 624
4. Сабуров С.В. Языки программирования C и C++. \_ М.: Бук\_пресс, 2006. – 647 с.
5. Окулов С. М. Программирование в алгоритмах – М.: БИНОМ, 2002. – 341
6. Зиборов В.В. MS Visual C++ 2010 в среде .NET.Библиотека програмиста. –СПБ.: Питер, 2012. – 320 с.

Интернет-ресурсы:

1. <https://ru.wikipedia.org/wiki/Стек>
2. <https://habrahabr.ru/post/100869/>
3. <https://ru.wikipedia.org/wiki/Обратная_польская_запись#Пример_вычисления_ выра>жений
4. https://www.intuit.ru/studies/courses/40/40/lecture/1200

Приложение

Windows Form

#pragma once

#include <msclr\marshal\_cppstd.h>

#include <string>

#include "MyClasses.h"

#define \_NColumn 2 //Количество столбцов в dataGridView1

#define NameCells 0 //Номер столбца с именем в dataGridView1

#define ValueCells 1 //Номер столбца со значением переменной в dataGridView1

ArithmExp RES;

namespace ArithmeticExprwssions {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для FormToSteak

/// </summary>

public ref class FormToSteak : public System::Windows::Forms::Form{

public:

FormToSteak(void){

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~FormToSteak(){

if (components){

delete components;}}

private: System::Windows::Forms::Label^ label1;

protected:

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::DataGridView^ dataGridView1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::TextBox^ textBox3;

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void){

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->button1 = (gcnew System::Windows::Forms::Button());

this->button2 = (gcnew System::Windows::Forms::Button());

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Pixel,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(12, 13);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(46, 17);

this->label1->TabIndex = 0;

this->label1->Text = L"label1";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Pixel,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(9, 97);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(46, 17);

this->label2->TabIndex = 1;

this->label2->Text = L"label2";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Pixel,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(9, 216);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(46, 17);

this->label3->TabIndex = 2;

this->label3->Text = L"label3";

//

// button1

//

this->button1->Location = System::Drawing::Point(16, 62);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(75, 23);

this->button1->TabIndex = 3;

this->button1->Text = L"button1";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &FormToSteak::button1\_Click);

//

// button2

//

this->button2->Location = System::Drawing::Point(10, 268);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(101, 23);

this->button2->TabIndex = 4;

this->button2->Text = L"button2";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &FormToSteak::button2\_Click);

//

// dataGridView1

//

this->dataGridView1->AllowUserToAddRows = false;

this->dataGridView1->AllowUserToDeleteRows = false;

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Location = System::Drawing::Point(677, 14);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->Size = System::Drawing::Size(133, 294);

this->dataGridView1->TabIndex = 5;

//

// textBox1

//

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->Location = System::Drawing::Point(16, 33);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(647, 23);

this->textBox1->TabIndex = 6;

//

// textBox2

//

this->textBox2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox2->Location = System::Drawing::Point(10, 117);

this->textBox2->Name = L"textBox2";

this->textBox2->RightToLeft = System::Windows::Forms::RightToLeft::No;

this->textBox2->Size = System::Drawing::Size(650, 26);

this->textBox2->TabIndex = 7;

//

// textBox3

//

this->textBox3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox3->Location = System::Drawing::Point(8, 236);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(647, 26);

this->textBox3->TabIndex = 8;

//

// FormToSteak

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(822, 320);

this->Controls->Add(this->textBox3);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->dataGridView1);

this->Controls->Add(this->button2);

this->Controls->Add(this->button1);

this->Controls->Add(this->label3);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;

this->Name = L"FormToSteak";

this->Text = L"FormToSteak";

this->Load += gcnew System::EventHandler(this, &FormToSteak::FormToSteak\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->dataGridView1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();}

#pragma endregion

private: System::Void FormToSteak\_Load(System::Object^ sender, System::EventArgs^ e)

{

dataGridView1->BackgroundColor = Color::White;

this->CenterToScreen();

this->Text = "Arithmetic expressions";

label1->Text = "Арифметическое выражение";

label2->Text = "Обратная польская запись";

label3->Text = "Результат";

button1->Text = "OK";

button2->Text = "Подсчитать";

textBox2->ReadOnly = true;

textBox3->ReadOnly = true;

//Выравнивание столбцов по содержимому

dataGridView1->AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode::AllCells;

//Выравнивание строк по содержимому

dataGridView1->AutoSizeRowsMode = DataGridViewAutoSizeRowsMode::AllCells;

//Задаёт количество столбцов

dataGridView1->ColumnCount = \_NColumn;

//Удаляет самый левый столбец(Заголовки)

dataGridView1->RowHeadersVisible = false;

//Цвет фона dataGridView1

dataGridView1->BackgroundColor = BackColor.White;

//Задаёт стиль границы dataGridView1

dataGridView1->BorderStyle = BorderStyle::None;

//Скрывает верхнюю строку

dataGridView1->ColumnHeadersVisible = false;

textBox1->Focus();

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e)

{

dataGridView1->Columns->Clear();//Отчистка dataGridView1

//Задаёт верхнюю строку

dataGridView1->ColumnHeadersVisible = true;

//Задаёт количество столбцов в dataGridView1

dataGridView1->ColumnCount = \_NColumn;

//Устанавливаю названия столюцов (Name, Value)

dataGridView1->Columns[0]->HeaderText = "Name ";

dataGridView1->Columns[1]->HeaderText = " Value";

dataGridView1->Columns[0]->ReadOnly = true;//

// Обработка textBox1

std::string Arithm = (msclr::interop::marshal\_as<std::string>(textBox1->Text));//String^ to string

RES = ArithmExp(Arithm);

textBox2->Text = msclr::interop::marshal\_as<System::String^>(RES.getPoliz());//string to String^

//добавим в dataGridView1 строки ( = количеству переменных в арифметическом выражении )

dataGridView1->RowCount = RES.getVarNumber();

for(int i = 0; i < RES.getVarNumber(); i++)

{

//Запись имени переменной в dataGridView1

dataGridView1->Rows[i]->Cells[NameCells]->Value =

msclr::interop::marshal\_as<System::String^>(RES[i].GetName());

//Запись значения в dataGridView1

dataGridView1->Rows[i]->Cells[ValueCells]->Value =

msclr::interop::marshal\_as<System::String^>(std::to\_string(RES[i].GetValue()));

System::String ^strc = Convert::ToString(dataGridView1->Rows[i]->Cells[NameCells]->Value);

if(Char::IsDigit(strc[0]))

{

dataGridView1->Rows[i]->Cells[ValueCells]->ReadOnly = true;

}

}

textBox1->Text = msclr::interop::marshal\_as<System::String^>(/\*\*/Arithm);//string to String^

//textBox1->Select(textBox1->Text->Length, textBox1->Text->Length); //Курсор в конец строки "ArrFirst->Text" C#

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e)

{

int L = RES.getVarNumber();

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

{

System::String ^ NameD = dataGridView1->Rows[i]->Cells[NameCells]->Value->ToString();

System::String ^ ValueD = dataGridView1->Rows[i]->Cells[ValueCells]->Value->ToString();

std::string stNam = msclr::interop::marshal\_as<std::string>(NameD);

double kp = stod(msclr::interop::marshal\_as<std::string>(ValueD));

//Изменение значения переменной Name-Value

RES.setValue(stNam, kp);

}

textBox3->Text= msclr::interop::marshal\_as<System::String^>(std::to\_string(RES.calculate()));

}}; }

Программа с main

#include "FormToSteak.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main(array<System::String ^> ^args){

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

ArithmeticExprwssions::FormToSteak Form;

Application::Run(%Form);

return 0;}

Класс Zapis

#include <string>

#include "Sort.h"

#ifndef \_My\_Zapis\_For\_Tablice\_\_

#define \_My\_Zapis\_For\_Tablice\_\_

template <class T>

class Zapis{

std::string Name;

T Value;

public:

//Конструктор (+ по-умолчанию)

Zapis<T>(std::string str = "", T \_Value = (T)0);

//Конструктор копирования

Zapis<T>(const Zapis<T> &tmp);

//Деструктор

~Zapis<T>(void);

//Операция присвоения

Zapis<T>& operator =(const Zapis<T> &tmp);

//Получить имя в записи

std::string GetName(void) const;

//Получить значение

T GetValue(void) const;

//Установить значение

void SetValue(const T k);};

//Конструктор (+ по-умолчанию)

template <class T>

Zapis<T>::Zapis(std::string str, T \_Value): Name(str), Value(\_Value){}

//Конструктор копирования

template <class T>

Zapis<T>::Zapis(const Zapis<T> &tmp): Name(tmp.Name), Value(tmp.Value){}

//Деструктор

template <class T>

Zapis<T>::~Zapis(void) {}

//Операция присвоения

template <class T>

Zapis<T>& Zapis<T>::operator =(const Zapis<T> &tmp){

if(this != &tmp){

Name = tmp.Name;

Value = tmp.Value;}

return \*this;}

//Получить имя в записи

template <class T>

std::string Zapis<T>::GetName(void) const{ return Name; }

//Получить значение

template <class T>

T Zapis<T>::GetValue(void) const{ return Value; }

//Установить значение

template <class T>

void Zapis<T>::SetValue(const T k) { Value = k; };

#endif //\_My\_Zapis\_For\_Tablice\_\_

Класс MyStack

#ifndef \_\_My\_\_Steak\_\_

#define \_\_My\_\_Steak\_\_

template <class Typ>

class MySteak{

Typ \*mem;

int size, count;

public:

//Констрктор (-|- по-умолчанию)

MySteak<Typ>(int \_size = 20): size(\_size), count(0){

mem = new Typ[size];}

//Констрктор копирования

MySteak<Typ>(const MySteak<Typ> &tmp): size(tmp.size), count(tmp.count){

mem = new Typ[size];

/\*\*///Некоторые классы не поддерживают операцию присваивания или присваивание по индексу "[]"

for(int i = 0; i < size; i++){

mem[i] = tmp.mem[i];}}

//Деструктор

~MySteak<Typ>(void){

delete []mem;

mem = nullptr;}

//Операция присваивания

MySteak<Typ>& operator =(const MySteak<Typ> &tmp){

if(this != &tmp){

if(size != tmp.size){

if(size != NULL) { delete []mem; }

size = tmp.size;

mem = new Typ[size];}

count = tmp.count;

/\*\*///Некоторые классы не поддерживают операцию присваивания или присваивание по индексу "[]"

for(int i = 0; i < size; i++){

mem[i] = tmp.mem[i];}}

return \*this;}

//Взять элемент

Typ top(void){

return mem[--count];}

//Положить элемент

void push(const Typ tmp){

mem[count++] = tmp;}

//Посмотреть элемент

Typ pop(void) const{

if(count <= size && count > 0) { return mem[count - 1]; }

else { throw -1; }}

//Провека на пустоту

//Если == 0, то стек пуст

int isEmpty(void) const{

return count;}

//Провека на полноту

//Если == 0, то стек полный

int isFull(void) const{

return size - count;}};

#endif //\_\_My\_\_Steak\_\_

Класс MyTablice

#ifndef \_\_My\_Tablice\_\_\_\_

#define \_\_My\_Tablice\_\_\_\_

template <class T>

class MTablice{

Zapis<T> \*mem;

int size, count;

public:

//Констрктор (-|- по-умолчанию)

MTablice<T>(int \_size = 0): size(\_size), count(0){

mem = new Zapis<T>[size];

Zapis<T> nol;

/\*\*///Класс запись не должен менять имя посредством "=", что будет после заполнения таблицей записью типа: ("", 0.0)

for(int i = 0; i < size; i++){

mem[i] = nol;}}

//Констрктор копирования

MTablice<T>(const MTablice<T> &tmp): size(tmp.size), count(tmp.count){

mem = new Zapis<T>[size];

/\*\*///Класс запись не должен менять имя посредством "=", что будет после заполнения таблицей записью типа: ("", 0.0)

for(int i = 0; i < size; i++){

mem[i] = tmp.mem[i];}}

//Деструктор

~MTablice<T>(void){

delete []mem;

mem = nullptr;}

//Операция присваивания

MTablice<T>& operator =(const MTablice<T> &tmp){

if(this != &tmp){

if(size != tmp.size){

if(size != NULL) { delete []mem; }

size = tmp.size;

mem = new Zapis<T>[size];}

count = tmp.count;

/\*\*///Класс запись не должен менять имя посредством "=", что будет после заполнения таблицей записью типа: ("", 0.0)

for(int i = 0; i < size; i++){

mem[i] = tmp.mem[i];}}

return \*this;}

//Взятие по индексу

Zapis<T> operator[](const int index) const{

Zapis<T> rez;

if((index > -1) && (index < count)){

rez = mem[index];}

return rez;}

//Добавить в таблицу запись

void add(std::string Name, T Value){

int L = -1;

for(int i = 0; i < count && -1 == L; i++){

if(Name == mem[i].GetName()){ L = i; }}

if(-1 == L){

mem[count++] = Zapis<T>(Name, Value);}

else{

mem[L].SetValue(Value);}}

//Удалить запись

void del(std::string Name){

int L = -1;

for(int i = 0; ((i < count) && (-1 == L)); i++){

if(Name == mem[i].GetName()){

L = i;

mem[i] = mem[--count];}}}

//Поиск по имени

Zapis<T> search(std::string Name) const{

Zapis<T> res;

int L = -1;

for(int i = 0; i < count && -1 == L; i++){

if(Name == mem[i].GetName()){

L = i;

res = mem[i];}}

return res;}

//Проверка на пустоту

//Если 0 - то пуст

int isEmpty(void) const{

return count;}

//Проверка на переполнение

//Если 0 - то места нет(Полный)

int isFull(void) const{

return (size - count);}

//Посмотреть количество записей

int getCount(void) const{

return count;}};

#endif //\_\_My\_Tablice\_\_\_\_

Класс Pol

#ifndef \_\_POLIZ\_across\_Steak\_\_

#define \_\_POLIZ\_across\_Steak\_\_

#include "MyClasses.h"

class Pol{

std::string arifm, \*poliz, \*w;

int size, count;

MySteak<Zapis<int> > operacia;

MTablice<int> tab\_operc;

MTablice<double> tab\_var;

//Обработка левой скобки "("

void work\_left(void){

operacia.push(Zapis<int>("(", 0));}

//Обработка правой скобки ")"

void work\_right(void){

while((operacia.isEmpty()) && (operacia.pop().GetName() != "(")){

poliz[count++] = operacia.top().GetName();}

if("(" == operacia.pop().GetName()) { operacia.top(); }}

//Обработка переменной

//(если строка это переменная или константа)

void work\_var(std::string st){

poliz[count++] = st;

char SerNum = st[0];

if((SerNum >= '0') && (SerNum <= '9')){//Если цифра

tab\_var.add(st, stod(st));}

else{//если не цифра

tab\_var.add(st, 0);}}

//Обработка оператора

//(если строка - это какой-либо знак из "+ - \* / ^" )

void work\_operat(Zapis<int> tmp){

while((operacia.isEmpty()) && (operacia.pop().GetValue() >= tmp.GetValue())){

poliz[count++] = operacia.top().GetName(); }

operacia.push(tmp);}

//Создани польско-инвёстной записи

//ОПЗ - обратная польская запись

void make\_poliz(void){

const std::string razdelitel = " +-\*/^()";

const int maxMass = arifm.length() + 1;

w = new std::string[maxMass];

int k = 0;

strToMass(arifm, razdelitel, w, k);

size = k + 1;

const int sizeTabVar = (int)(k / 2) + 1;

poliz = new std::string[size];

tab\_var = MTablice<double>(sizeTabVar);

count = 0;

for(int i = 0; i < k; i++){

if("(" == w[i]) { work\_left(); }

else{

if(")" == w[i]) { work\_right(); }

else{

Zapis<int> tmp = tab\_operc.search(w[i]);

if("" == tmp.GetName()) { work\_var(w[i]); }

else { work\_operat(tmp); }}}}

end\_work();}

//Создание таблици операций с их приоритетами

void make\_tab\_operacia(void){

tab\_operc = MTablice<int>(6);

tab\_operc.add("+", 1);

tab\_operc.add("-", 1);

tab\_operc.add("\*", 2);

tab\_operc.add("/", 2);

tab\_operc.add("^", 3);}

//Выгрузка стека для создания ОПЗ

//когда в стеке остались только операции

void end\_work(void){

while(operacia.isEmpty()){

if(operacia.pop().GetName() != "(") { poliz[count++] = operacia.top().GetName(); }

else { operacia.top(); }}}

public:

//Констрктор (-|- по-умолчанию)

Pol(std::string st = ""): arifm(st){

make\_tab\_operacia();

make\_poliz();}

//Констрктор копирования

Pol(const Pol &tmp): arifm(tmp.arifm), operacia(tmp.operacia), tab\_var(tmp.tab\_var){

make\_tab\_operacia();

make\_poliz();}

//Деструктор

~Pol(void){

delete []poliz;

poliz = nullptr;

delete []w;

w = nullptr;}

//Операция присваивания

Pol& operator =(const Pol &tmp){

if(size != tmp.size){

if(size != NULL){

delete []poliz;}

size = tmp.size;

poliz = new std::string[size];}

arifm = tmp.arifm;

make\_tab\_operacia();

//for(int i = 0; i < size; i++) { poliz[i] = tmp.poliz[i]; }

make\_poliz();

//tab\_var = tmp.tab\_var; ///make\_poliz() - Сделает таблицу переменных

return \*this;}

//Взятие по индексу

std::string operator [](const int k){

if(k > -1 && k < count){

return poliz[k];}

else{

return "";}}

//Посмотреть количество элементов

int getCount(void) const{

return count;}

//Посмотреть ПОЛИЗ

std::string getPoliz(void) const{

std::string str = "";

for(int i = 0; i < count; i++){

str = str + poliz[i] + " ";}

return str;}

//Посмотреть таблицу переменных

MTablice<double>& getTablVar(void){

return tab\_var;}};

#endif //\_\_POLIZ\_across\_Steak\_\_

Класс ArithmExp

#ifndef \_\_Arithmetic\_\_Expressions\_\_\_\_

#define \_\_Arithmetic\_\_Expressions\_\_\_\_

class ArithmExp{

Pol p;

MySteak<double> s;

public:

//Констрктор (-|- по-умолчанию)

ArithmExp(std::string st = ""){

p = Pol(st);

s = MySteak<double>(20);}

//Констрктор копирования

ArithmExp(const ArithmExp &tmp): p(tmp.p), s(tmp.s){}

//Деструктор

~ArithmExp(void){}

//Операция присваивания

ArithmExp& operator =(const ArithmExp &tmp){

p = tmp.p;

s = tmp.s;

return \*this;}

//Взятие по индексу

Zapis<double> operator [](const int k){

return (p.getTablVar()[k]);}

//Посмотреть ПОЛИЗ

std::string getPoliz(void) const{

return p.getPoliz();}

//Показать количество переменных

int getVarNumber(void){

return p.getTablVar().getCount();}

//Посчитать выражение

double calculate(void){

const int sizePol = p.getCount();//Для цикла for(размер "Pol p")

double left, right;

for(int i = 0; i < sizePol; i++){

if("+" == p[i]){

right = s.top();

left = s.top();

s.push(left + right);}

else{

if("-" == p[i]){

right = s.top();

left = s.top();

s.push(left - right);}

else{

if("\*" == p[i]){

right = s.top();

left = s.top();

s.push(left \* right);}

else{

if("/" == p[i]){

right = s.top();

left = s.top();

s.push(left / right);}

else{

if("^" == p[i]){

right = s.top();

left = s.top();

s.push(pow(left,right));}

else{

if(" " != p[i]){

double num = ((p.getTablVar().search(p[i])).GetValue());

s.push(num);}}}}}}}

return s.top();}

//Установить значение по параметрам "Zapis<double> temp"

void setValue(Zapis<double> tmp){

//Zapis<double> Temp = (p.getTablVar().search(tmp.GetName()));

if((p.getTablVar().search(tmp.GetName())).GetName() != ""){

p.getTablVar().add(tmp.GetName(), tmp.GetValue());}}

//Установить значение по параметрам "Name\_" и "Value\_"

void setValue(std::string Name\_, double Value\_){

//Zapis<double> Temp = (p.getTablVar().search(tmp.GetName()));

if((p.getTablVar().search(Name\_)).GetName() != ""){

p.getTablVar().add(Name\_, Value\_);}}};

#endif //\_\_Arithmetic\_\_Expressions\_\_\_\_

Метод strToMass

#include "Sort.h"

void strToMass(std::string arif, std::string ZnakRazdel, std::string \*rezt, int &ki){

std::string Stok = arif + " ";

int Cn = Stok.length();

int m = ZnakRazdel.length();

ki = 0;

for (int i = 0; i < Cn; i++){

int L = ZnakRazdel.find(Stok[i]);

if ((L > m) || (L < 0)){ rezt[ki] = rezt[ki] + Stok[i]; }

else

if (Stok[i] == ZnakRazdel[0]){

if (rezt[ki] != ""){ rezt[++ki] = ""; }}

else{

if (rezt[ki] != "") { ki++; }

rezt[ki] = Stok[i];

rezt[++ki] = "";}}}