Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационных технологий, математики и механики

**Отчёт по лабораторной работе**

программный калькулятор для целых чисел на основе обратной польской записи

**Выполнил:**

Студент группы 0823-3

Юферев Кирилл Владимирович

**Проверил:**

Преподаватель группы 0823-3

Козинов Евгений Александрович

г. Нижний Новгород

2015 г.

Оглавление

[Содержание 2](#_Toc446450755)

[Введение 3](#_Toc446450756)

[Постановка задачи 4](#_Toc446450757)

[Руководство пользователя 5](#_Toc446450758)

[Руководство программиста 6](#_Toc446450759)

[Общий алгоритм работы программы: 7](#_Toc446450760)

[Заключение 8](#_Toc446450761)

[Список Используемой литературы: 9](#_Toc446450762)

[Приложения 10](#_Toc446450763)

# 

# Введение

В математике существует древняя традиция помещать оператор между операндами (x+y), а не после операндов (xy+). Форма с оператором между операндами называется инфиксной записью. Форма с оператором после операндов называется постфиксной, или обратной польской записью в честь польского логика Я. Лукасевича (1958), который изучал свойства этой записи.  
Обратная польская запись имеет ряд преимуществ перед инфиксной записью при выражении алгебраических формул. Во-первых, любая формула может быть выражена без скобок. Во-вторых, она удобна для вычисления формул в машинах со стеками. В-третьих, инфиксные операторы имеют приоритеты, которые произвольны и нежелательны. Например, мы знаем, что ab+c значит (ab)+c, а не a(b+c), поскольку произвольно было определено, что умножение имеет приоритет над сложением.

В общем виде ОПЗ выглядит следующим образом:

* Запись набора операций состоит из последовательности операндов и знаков операций. Операнды и операции в выражении при письменной записи разделяются пробелами.
* Выражение читается слева направо. Когда в выражении встречается знак операции, то выполняется соответствующая операция над двумя последними встретившимися перед ним операндами в порядке их записи. Полученный результат заменяет в выражении последовательность её операндов и её знак, после чего выражение вычисляется дальше по тому же правилу.
* Результатом вычисления становится результат последней вычисленной операции.

# Постановка задачи

На вход программы поступает выражение, состоящее из чисел и знаков арифметических действий. Требуется преобразовать это выражение в обратную польскую запись или же сообщить об ошибке.

В программе целью было реализовать операции над числами:

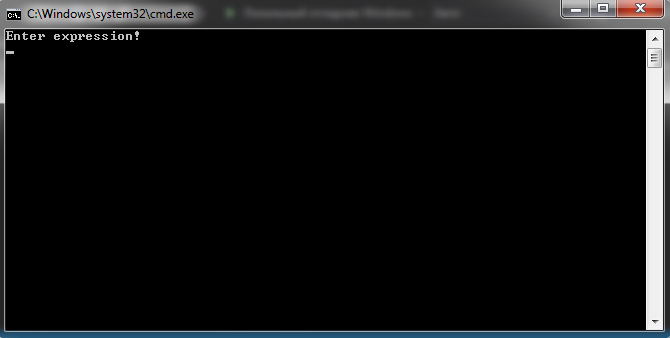
* Сложение
* Вычитание
* Умножение
* Деление
* Возведение в степень

А так же:

* Реализация шаблонного класса TStack
* Реализация проверок на правильность введенного выражения, таких как пустые скобки, не закрытые скобки, множественные операции, следующие друг за другом и т.п.
* Реализация возможности производить операции с действительными числами

# Руководство пользователя

После запуска программы Вы увидите на экране надпись ”Enter expression!”, что будет означать ,что программа готова к работе и ждет Вашего выражения. Для начала работы введите какое-нибудь выражение, например: (2+2)\*4 и нажмите Enter.



**Рис 1.** Ввод выражения.

Теперь Вы увидите несколько строк: Ваше исходное выражение, выражение в виде ОПЗ и конечный результат. Если вы введете выражение, которое невозможно посчитать или не закроете скобку, то программа выдаст Вам сообщение об ошибке.



**Рис 2.** Программа в действии. Конечный результат.

# Руководство программиста

**Функции:**

Для выполнения поставленной задачи мною было принято решение создать шаблонный класс **TStack**, чтобы избежать создания нескольких похожих классов.

**Для класса TStack реализованы следующие функции:**

**void Push(ValType x)** – функция добавления в стэк нового элемента

**bool isFull()** – функция проверки стэка на полноту

**ValType Pop()** – функция извлечения элемента на верхушке стэка

**ValType Peek()** – функция получения элемента на верхушке стэка

**bool IsEmpty()** – функция проверки стэка на пустоту

Для удобной работы, преобразования инфиксной записи в постфиксную, вычисления выражений были разработаны следующие функции:

* **bool SetSt()** – функция запроса выражения у пользователя.
* **void PrintOut()** – функция печати строки в постфиксной форме записи
* **bool Process()**  - функция преобразования в постфиксную форму записи
* **int Prior(char c)** – функция с расставленными приоритетами операций
* **double Calculate()** –функция вычисления результата
* **double Operation(double,double,int)** – вспомогательная вычислительная функция
* **void PrintStr()** – функция вывода исходной строки
* **double stepen(double,int)** - вспомогательная вычислительная функция(степень)

Для защиты программы от «плохих» выражений внутри программы были расставлены исключения и всевозможные выводы сообщений об ошибке.

В программе были реализованы все, задуманные мной операции, а именно:

* **Умножение**
* **Деление**
* **Вычитание**
* **Сложение**
* **Возведение в степень**

# Общий алгоритм работы программы:

1. Считываем строку, введенную пользователем, попутно выявляя нарушения ввода, правильность скобок и операций.
2. Введенная строка дублируется на экран
3. Производится преобразование инфиксной записи в постфиксную
4. Числа помещаются в стэк типа double, операции помещаются в стэк типа int
5. Производится вычисление, результат складывается в стэк типа double
6. Возвращается последнее значение стэка типа double, оно и является результатом

# Заключение

В заключении хочу сказать, что все цели лабораторной работы выполнены, калькулятор на основе Обратной Польской Записи (ОПЗ) работает нормально, заключения обрабатывает, операции сложения, деления, умножения, вычитания и возведения в степень выполняет.

# Список Используемой литературы:

Гергель В.П. Рабочие материалы к учебному курсу «Методы программирования», ННГУ, 2002. – 100 c.

**Список используемых интернет-ресурсов:**

<https://habrahabr.ru/post/100869/>

<https://ru.wikipedia.org/wiki/Обратная_польская_запись>

# Приложения

int OPZ::Prior(char c) //функция с заданием приоритетов

{

char p[7][2]=

{

{'(',0},{')',1},{'+',2},{'-',2},{'\*',3},{'/',3},{'^',4}

};

for (int i=0;i<7;i++)

{

if (c==p[i][0])

return p[i][1];

}

return -1;

} // конец примера

double OPZ::stepen(double Op1, int Op2) // функция возведения в степень числа

{

double result=Op1;

for (int i=2;i<=Op2;i++)

{

result=result\*Op1;

}

return result;

} //конец примера

double num=0; // Кусок функции calculate,которая преобразовывает и считает числа

double drob=0;

double ten=10;

while (out[i]!=' '&&out[i]!='.')

{

num=num\*ten+(out[i]-'0');

i++;

}

if(out[i]=='.')//есть дробная часть

{

i++;

while (out[i]!=' '&&out[i]!='.')

{

drob=drob+(out[i]-'0')/ten;

ten=ten\*10;

i++;

}

if(out[i]=='.')

{

cout<<"incorrect float number"<<endl;

return 0;

}

}

ds.Push(num+drob);

//конец примера