Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

Высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий математики механики

Польская запись

Отчет по лабораторной работе

Выполнил:

студент ИИТММ гр. 381706-2

Антипин А.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

ассистент каф. МОСТ, ИИТММ

Лебедев И.Г\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Нижний Новгород

2018 г.

**Содержание**

[Введение 3](#_Toc820902)

[Постановка целей и задач 4](#_Toc820903)

[Руководство пользователя 5](#_Toc820904)

[Руководство программиста 6](#_Toc820905)

[Описание структуры программы 6](#_Toc820906)

[Описание структур данных 7](#_Toc820907)

[Описание алгоритмов 8](#_Toc820908)

[Заключение 10](#_Toc820909)

[Литература 11](#_Toc820910)

# 1.Введение

Для визуального представления математических выражений мы привыкли использовать прямой порядок запись чисел и арифметических знаков, т.е. выражение a + b мы привыкли понимать так: необходимо к числу a прибавить число b. Но существует и другая форма записи числа, которая подразумевает под собой изначальную запись аргументов, а уже потом арифметических знаков. Такую запись принято называть обратной польской записью и выглядит она следующим образом: выражение a + b преобразуется в ab +. Более сложный пример приведен на рис. 1. По данным Википедии, данная форма записи используется в «организации байт-кода конфигураций прикладных решений системы 1С:Предприятие. Официального подтверждения компания 1С не дает, но пользователи данной системы на специализированных форумах приводят доказательства и алгоритмы, позволяющие декомпилировать исходные тексты.».

(1 + 2) \* 4 + 3 1 2 + 4 \* 3 +

рис. 1 (Инфиксное выражение – слева и аналогичное ей постфиксная запись справа)

Для   
{\displaystyle (1+2)\times 4+3}преобразование из обычной записи в польскую обычно используют один стек, в нашем случае для преобразования и подсчета выражения будет использоваться два стека: один – целочисленных чисел, в котором будем хранить преобразованные из char числа (в нем окажется и результат вычислений), и стек символов – арифметических знаков.

# 2.Постановка целей и задач

Основной целью лабораторной работы является создание структуры данных «Polish string» и реализация такого метода как: сосчитать выражение:

Для реализации алгоритмов будет использоваться классы Polish string, Calculator и структура Number.

Для проверки правильности работы этого класса будут написаны тесты с использованием фреймворка Google Test, а также тестовый образец программы, которая использует данный класс.

# 3.Руководство пользователя

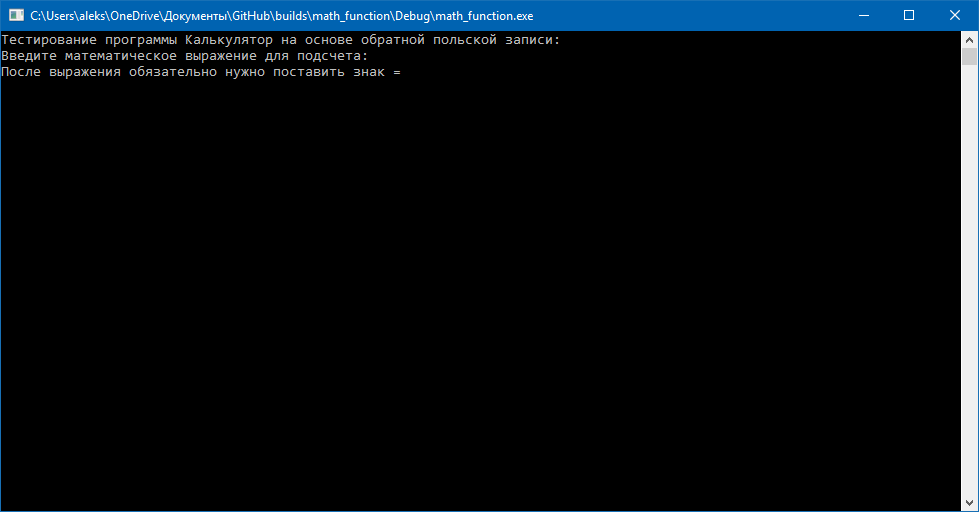
После запуска программы пользователя встречает консольное окно (рис. 2):

рис. 2 (вывод работы калькулятора для пользователя)

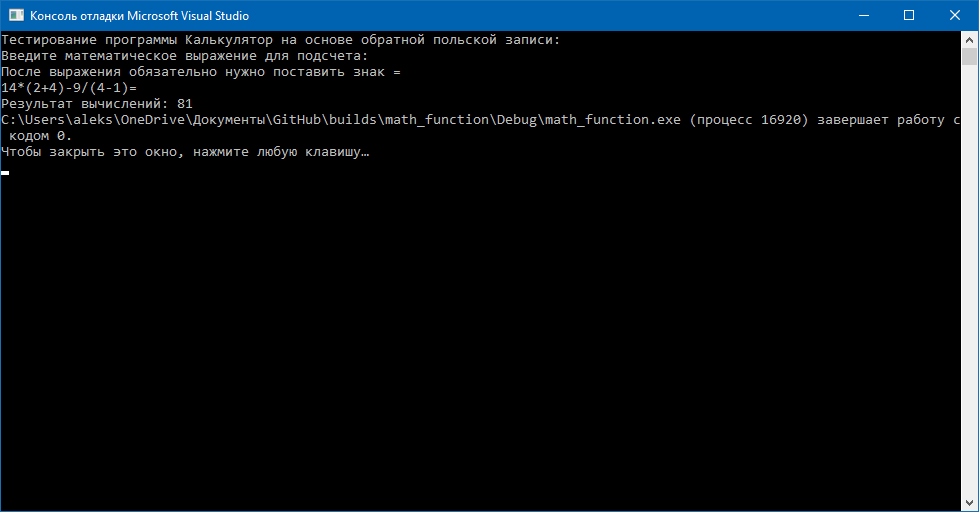
От пользователя требуется ввести какое-либо математическое выражение без пробелов и в конце поставить знак равно как признак его окончания. Затем нажать Enter, и программа сама подсчитает его и выведет результат на экран (рис. 3).

рис. 3 (вывод конечного результата вычислений программы)

# 4.Руководство программиста

## 4.1.Описание структуры программы

При создании объекта типа Calculator подразумевается, что максимальная длина строки, которую передают на вычисление не превышает 100 символов. Такого объема достаточно для вычисления несложных математических выражений, а при необходимости структуру можно легко изменить для более большого объема информации.

Для реализации алгоритмов будет создано два класса Calculator и Polish\_string, а также структура Number.

Проверку правильности работы алгоритмов будет осуществлять фреймворк Google Test.

**Структура Number**

**Класс Calculator**

**Polish\_record.cpp**

**Polish\_record.h**

**Polish\_string.cpp**

Древо классов

**Класс gtest**

**test\_main.cpp**

**Polish\_string\_test.cpp**

**Проект для пользователя**

**main.cpp**

**Polish\_string.h**

**Класс Polish\_string**

**Calculator\_test.cpp**

**Структура Number:**

Эта структура временно хранит либо текущее число, либо текущий арифметический знак, взятые из строки, переданной пользователем.

**Класс Polish\_string:**

Этот класс включает в себя методы обработки строки переданной пользователем для ее дальнейшего преобразования и вычисления результата.

**Класс Calculator:**

йКласс, в котором реализованы методы вычисления и обработки текущих символов, которые поступили из строки.

**Класс gtest:**

Класс gtest реализует тестирование всех классов, по средствам фреймворка Google Test. Тесты пишутся для каждого метода классов, каждого ветвления этих методов и для всех возможных исключений этих методов.

**Проект main:**

В данном проекте реализован примет использования калькулятора, в котором пользователь может ввести математическое выражение и получить результат вычислений.

## 4.2.Описание структур данных

**Структура Number:**

char\* number – указатель на выделяемую память под текущий символ;

int len – длина текущего символа;

**Класс Polish\_string:**

char\* str – указатель на память выделяемую под строку, переданную пользователем;

int index – индекс текущего элемента в строке;

int char\_count – количество символов в строке;

char\* servicechar – массив математических выражений, которые могут использоваться;

**Класс Calculator:**

Polich\_string str – переменная типа Polish\_string;

Stack<int>\* numbers – стек под числа в строке;

Stack<char>\* s\_char – стек под математические выражения;

Exceptions\_from\_polinom\_and\_calculator exception – переменная для вызова исключений;

**Описание методов:**

|  |  |
| --- | --- |
| Метод: | Описание: |
| Number::Number() | Конструктор по умолчанию для Number. |
| Polich\_string::Polich\_string() | Конструктор по умолчанию для класса Polich\_string. |
| int Polich\_string::GetIndex() | Метод, который позволяет вернуть индекс рассматриваемой строки. |
| void Polich\_string::WrightString(char\* string) | Метод, который принимает строку. |
| Number Polich\_string::GetNextChar() | Метод, который возвращает следующий символ в формате Number для класса Calculator. |
| bool Polich\_string::IsServiceChar() | Метод, который определяет является ли текущий символ математических знаков. |
| Calculator::Calculator(char\* string) | Конструктор с параметром, который принимает строку из символов. |
| int Calculator::Calculate() | Основной метод, который производит вычисления. |
| int Calculator::AddToStacks(Number n) | Метод, который обеспечивает заполнение стеков новыми символами. |
| int Calculator::CharToInt(Number st) | Метод, который по принятой строке возвращает числовое значение. |
| int Calculator::Priority() | Метод, который возвращает приоритет поступившей операции. |
| bool Calculator::WhatsMore() | Метод, который дает разрешение на выполнение операции по ее приоритету. |

## 4.3.Описание алгоритмов

**Подробное описание некоторых методов**

Превращение строки в целочисленное значение:

* Заводятся две целочисленные переменные, одна для результата, вторая для исключения, если оно возникнет.
* В цикле от 0 до длины данного символа, которая передается в Number, из каждого элемента строки вычитается символ «0», тем самым можно узнать какое это число. Затем оно умножается на 10 в степени текущей итерации цикла. После чего это число складывается с предыдущим своим значением
* Если какое-то число оказывается большим 9, или меньше 0 на какой-то итерации цикла до умножения на 10 в степени, то вызывается исключение, т.к. этот символ не является числом, а кокай-либо другой символ.

Подсчет результата:

* Создаются две целочисленные переменны, которые будут отвечать за аргументы в текущей операции, а также создается булева переменная для разрешения или запрета выполнения операции на определенной итерации цикла;
* Вводится бесконечный цикл, который закончится, когда в целочисленном стеке останется только одно число, а в знаковом все закончатся.
* Получение следующего символа, распределение этого символа в нужный стек и определение приоритета математических выражений.
* Если можно выполнить математическое действие, то:
  + Если встречается открывающая скобка, то создается еще одна строка с элементами, расположенными в этой скобке, создается дополнительная переменная типа «Calculator», от которой рекурсивно вызывается метод подсчета текущего выражения, затем общий цикл одет на следующий круг.
  + Если же встречается какое-либо другое математическое выражение, то из числового стека берутся два верхних числа, над ними выполняется действие и результат кладется в числовой стек.
* После того, как в числовом стеке останется только одно число, оно возвращается.

# 5.Заключение

Подводя итоги можно сказать, что все поставленные цели работы были выполнены, а именно созданы методы представления строки в математическое выражение и его подсчет. Все тесты к данной работе бы успешно пройдены, что подтверждает правильность ее работы и возможность применения в реальных задачах.

# 6.Литература

* Учебные материалы к учебному курсу «Методы программирования» - Гергель В.П.
* https://ru.wikipedia.org/wiki/Обратная\_польская\_запись