МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

Высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

оставь надежды всяк сие читающий (карин т.а.)

**Отчет по учебной практике**

**«Преобразование арифметических выражений в обратную польскую запись»**

**Выполнил:**

студент группы 381706­1

Карин Тимофей Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Проверил:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc1346413)

[3. Руководство пользователя 5](#_Toc1346414)

[4. Руководство программиста 6](#_Toc1346415)

[4.1. Описание структуры программы 6](#_Toc1346416)

[4.2 Описание структур данных 6](#_Toc1346417)

[4.3 Описание алгоритмов 7](#_Toc1346418)

[5. Заключение 9](#_Toc1346419)

[6. Список литературы 10](#_Toc1346420)

1. Введение

Для начала давайте разберёмся, чему будет равно выражение 2+2\*2? Если операции выполняются по порядку (как это делает обычный калькулятор), то результат будет равен 8. А если так, как учили когда-то в школе, то 6. В чём разница? Разница в приоритете операций. В школе учили, что сначала нужно умножать и делить, а потом складывать и вычитать. Теперь возникает вопрос: как сообщить компьютеру, что мы планируем выполнять действия не в прямом порядке, а в том, в котором этого требует приоритет операций?

Самым лучшим (производительным) является представление выражения в обратной польской записи (Reverse Polish notation, RPN). Это такая форма записи математических и логический выражений, в которой операнды расположены перед знаком операции. Так ОПЗ приведённого выше выражения будет выглядеть так: 222\*+.

Но числа могут быть и больше 10, а лабораторная работа должна предусматривать работу с многозначными числами. Поэтому числа нужно как-то разграничивать, иначе 22 2 нельзя отличить от 2 22. В связи с этим, числа будут располагаться между квадратными скобками. И выражение [2] [22] уже нельзя ни с чем спутать. Так в нашем примере выражение будет выглядеть, как [2][2][2]\*+.

Таким образом, в ходе данной лабораторной работы необходимо создать библиотеку для работы с ОПЗ.

2. Постановка задачи

Целью данной лабораторной работы является создание библиотеки для перевода арифметический выражений в обратную польскую нотацию и вычисления результата выражений. Для этого необходимо:

* Создать функцию перевода строки в обратную польскую запись;
* Создать функцию, высчитывающую результат выражения, записанного в обратной польской записи;
* Обработать ситуации, когда строка представляет собой некорректное выражение;
* Проверить работоспособность функций.

3. Руководство пользователя

Для работы пользователей была написана небольшая программа, демонстрирующая возможности библиотеки. Данная программа расположена в файле *RPN.cpp* модуля *RPN.*

В ходе работы пользователю будет предложено ввести какое-либо арифметическое выражение. Основные требования к вводу выражения:

* Ввод производится без пробелов;
* Числа могут быть многозначными, но обязательно целыми и положительными;
* Допустимыми операциями являются только сложение (+), вычитание (-), умножение (\*) и деление (/);
* Выражение может начинаться со знака (-), но все остальные числа должны быть положительными.

В случае, если данные требования нарушены, а также, если выражение не соответствует требованиям построения арифметических выражений, программа прекратит своё выполнение, а пользователь увидит сообщение об ошибке.

4. Руководство программиста

4.1. Описание структуры программы

Программа состоит из трёх проектов:

* *RPNLib* – библиотека для работы с очередью;
* *RPN* – пример использования библиотеки *RPNLib*. Состоит из файла Queue;
* *RPNTest* – набор тестов для проверки работоспособности библиотеки *RPNLib.*

Также в программе используется класс исключений *TExсeption*, содержащийся в файле *Exсeption.h* проекта *Exсeption*.

4.2 Описание структур данных

Рассмотрим поля и методы класса *TString* расположенный в файле *MyString.h:*

Со спецификатором доступа private:

* *char\* str –* массив символов;
* *int len –* длина строки (включая нулевой символ ‘\0’);

Со спецификатором доступа public:

* *TString()* – конструктор по умолчанию;
* *TString(TString& \_str) –* конструктор копирования;
* *TString(char\* ch)* – конструктор преобразования типа;
* *~TString()* – деструктор;
* *int GetLength()* – возвращает размер (длину) строки;
* *TString& operator=(const TString& \_str)* – оператор присваивания;
* *TString operator+(TString& \_str)* – оператор сложения строк (конкатенация);
* *char& operator[] (int i)* – оператор индексирования;
* *friend std::istream& operator>>(std::istream &is, TString& str)* – функция ввода строки;
* *friend std::ostream& operator<<(std::ostream &os, const TString& str)* – функция вывода строки.

Рассмотрим функции, расположенные в файле *RPN.h* модуля *RPNLib:*

* *int GetPrior(const char c) –* функция, определяющая и возвращающая приоритет операций;
* *bool IsOperation(char c) –* проверяет, является ли символ символом операции;
* *TQueue<char> StrToRPN(TString str) –* функция перевода строки в обратную польскую нотацию;
* *double Calculate(TQueue<char> queue) –* функция, считающее значение выражения, записанного в обратной польской записи;
* *double Result(TString& A) –* функция, которая сразу выводит значение выражения (то есть программисту не нужно создавать очередь и вызывать функции перевода и подсчёта).

4.3 Описание алгоритмов

**Перевод выражения в обратную польскую запись**

Создаём очередь (которая в конечном итоге будет представлять из себя ОПЗ) и стек, в который будем помещать операции. Ещё нам понадобится переменная, которая будет отдельно считать количество открывающихся и количество закрывающихся скобок.

В качестве первого символа выражение может иметь либо цифру, либо открывающуюся скобку, либо унарный минус. Проверяем, выполняется ли это условие. Если нет, то выбрасываем исключение. Если первый символ является минусом, то добавляем в очередь число, равное нулю, и минус превращается в бинарный, при этом значение выражения не изменилось.

Если символ не первый, то определяем, с чем имеем дело: с числом или со знаком операции.

Если попалось число, то записываем его в очередь. Для этого добавляем в очередь открывающуюся квадратную скобку, затем до тех пор, пока символы являются цифрами их тоже добавляем в очередь. Напоследок ставим закрывающуюся квадратную скобку. Квадратные скобки нужны для разграничения чисел, т.к. если в ОПЗ будут рядом стоять два многозначных числа, то будет непонятно, в каком месте заканчивается одно, и начинается другое.

Если попался символ операции, то нужно рассматривать два случая. Если стек пустой, то кладём число в стек. При этом, если попался знак закрывающейся скобки, то выбрасываем исключение, т.к. в стеке должна быть как минимум открывающаяся скобка. Если стек не пустой, и добавляемый символ является закрывающейся скобкой, то нужно переложить из стека в очередь все знаки операций, пока из стека не будет взята открывающаяся скобка. Сами скобки в очередь записывать не нужно. Если перед нами не скобки, то проверяем приоритет операции. Если на вершине стека хранится знак, с меньшим приоритетом, то добавляем этот знак в стек. Если приоритет вершины стека равен или больше добавляемого знака, то нужно переписать элементы из стека в очередь до тех пор, пока условие не нарушится или пока не опустеет стек.

Если мы имеем дело ни с числом, ни со строкой, то проверяем, не равен ли символ символу окончания строки. Если равен, то переписываем всё, что осталось в стеке в очередь и возвращаем её. Если это не символ окончания строки, то это непонятно что, и нужно выбросить исключение.

Такие операции проделываются до тех пор, пока не дойдём до конца строки. Также в конце нужно проверить количество открывающихся и закрывающихся скобок. Если они не совпадают, то выбрасываем исключение.

**Получение результата выражения с помощью его ОПЗ**

Создаём стек, в который будем хранить операнды. Проверяем, не является ли первый символ очереди знаком операции. Если является, то выбрасываем исключение, т.к. в ОПЗ знаки операций идут только после операндов.

Затем поочерёдно получаем из очереди символы.

Если полученный символ оказался открывающейся квадратной скобкой, то забираем из очереди символы, пока не дойдём до закрывающейся квадратной скобки. Из того, что находится между ними собираем число и помещаем его в стек.

Если символ оказался знаком операции, то забираем из стека два элемента и производим над ними эту операцию. Результат кладём обратно в стек. Следует помнить, что первый взятый из стека элемент является правым операндом, а второй – левым.

Когда очередь будет пуста, в стеке должен оказаться один элемент, являющийся результатом всего выражения.

5. Заключение

В ходе данной лабораторной работы были получены следующие результаты:

* Написаны функции перевода выражения в обратную польскую запись и подсчёта значения этого выражения;
* Написан пример использования библиотеки, позволяющий пользователю вводить различные арифметические выражения и получать ОПЗ и результат этого выражения;
* Обработаны основные исключительные ситуации, связанные с некорректностью выражения как с математической, так и с технической стороны.
* С помощью фреймворка GoogleTest написаны тесты, проверяющие работоспособность библиотеки.

6. Список литературы

1. Ахо Альфред В, Хопкрофт Джон Э и Ульман Джеффри Д Структуры данных и алгоритмы [Книга]. - [б.м.] : Вильямс, 2003.
2. Лафоре Роберт Структуры данных и алгоритмы в Java [Книга]. - СПб : Питер, 2013. - 2 : стр. 704.
3. Павловская Т. А. C/C++ Программирование на языке высокого уровня [Книга]. - СПб : Питер, 2003.
4. Страуструп Бьерн Язык программирования C++ Бином, 2004.