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

Отчет по лабораторной работе «Вычисление арифметических выражений»

Выполнил:
студент группы 3821б1пм2
Головин Роман Михайлович
Проверил:
Преподаватель каф. МОСТ,
Волокитин В.Д.

Нижний Новгород
2022

Содержание

1. Введение

Обратная польская запись — форма записи математических и логических выражений, в которой операнды расположены перед знаками операций. Также именуется как обратная бесскобочная запись, постфиксная нотация, бесскобочная символика Лукасевича, польская инверсная запись.

Реализация вычисления арифметического выражения с использованием ОПЗ чрезвычайно проста и может быть очень эффективной. Обратная польская запись совершенно унифицирована — она принципиально одинаково записывает унарные, бинарные, тернарные и любые другие операции, а также обращения к функциям, что позволяет не усложнять конструкцию вычислительных устройств при расширении набора поддерживаемых операций. Это и послужило причиной использования обратной польской записи в некоторых научных и программируемых микрокалькуляторах.

2. Постановка задачи

Цель данной работы — разработка структуры данных Стек и ее использование для расчета арифметических выражений с использованием обратной польской записи (постфиксной формы).

Выполнение работы предполагает решение следующих задач:

1. Разработка интерфейса шаблонного класса TStack.
2. Реализация методов шаблонного класса TStack.
3. Разработка интерфейса класса TPostfix для работы с постфиксной формой.
4. Реализация методов класса TPostfix.
5. Разработка и реализация тестов для классов TStack и TPostfix на базе Google Test.
6. Публикация исходных кодов в личном репозитории на GitHub.

3. Руководство пользователя

При запуске программы на экране высвечивается краткая информация о доступных арифметических операциях,формате числа ,доступных математических функциях, формате переменных.

Программа ожидает от пользователя ввод арифметического выражения со стандарным обще принятым синтаксисом. Если пользователь допустил ошибку при вводе арифметического выражения ,программа укажет на нее , подскажет позицию ошибки во входной строке.

При спользовании переменных в арифметическом выражении программа запросит их значение.

4. Описание программной реализации

4.1. Описание структуры файлов

1. Папка arithmetic\Header Files

   - stack.h - файл содержит объявление и реализацию класса TStack;
   - arithmetic.h - файл содержит объявление классов TPostfix, реализацию класса Lexem, Operation, Operand, Constant, Variable и TPostfixException;

2. Папка arithmetic\Source Files

   - arithmetic.cpp - файл содержит реализацию класса TPostfix;

3. Папка tests\Source Files

   - test_arithmetic.cpp - содержит тесты для методов класса TPostfix,;
   - test_stack.h - содержит тесты для методов класса TStack.

4. Папка sample\Source Files

   - main_arithmetic.cpp - содержит реализацию пользовательского приложения;

4.2. Описание классов

1. Класс TStack

   TStack является шаблонным классом и представляет из себя обычный стек.

   private:

   - int data_size - количество блоков памяти в T* data;
   - int size=0 - количество заполненных элементов в T* data;
   - int top - индекс верхнего элемента T* data;
   - T* data - указатель на начало блока памяти с данными типа T.

   public:

   - Stack(int size = 1) - конструктор позволяющий задать стек заданного размера;
   - void push(T item) - метод добавляет элемент типа T на вершину стека;
   - T pop() - метод извлекает элемент типа T с вершины стека, если стек пуст будет брошено исключение;
   - T head() const - метод позволяет просматривать элемент типа T ,находящийся на вершине стека, если стек пуст будет брошено исключение;
   - bool isEmpty() const - метод позволяет узнать есть ли элементы в стеке (пуст стек или нет);= Если стек пуст будет выдано значение true . иначе - false;
   - int getSize() const noexcept - метод позволяет получить количество элементов в стеке;

- void clear() - метод очищает стек , устанавливая его размер на 0.

2. Класс TPostfix

   Класс хранит исходное выражение,переводит и хранит его представление в обратной польской записи,позволяет вычислить значение выражения.

   private:

   - string infix - строка, содержащая арифметическое выражение в инфиксной форме.Задается пользователем;
   - map<string, std::function<double(double x)» funcs - мап хранящий пару из имени функции и лямбда функии;
   - map<string, std::function<double (double x, double y)» ops- мап хранящий пару из имени операции и лямбда функии;
   - vector<Lexeme*> postfix - вектор хранящий постфиксную запись выражения;
   - map<string, double> variable_map - мап хранящий имя переменной и ее значение;
   - Lexeme* CreateLexeme(string str, int start, int stop) - меод принимает на вход строковое представление лексемы,индекс ее начала и конца в исходной строке. В зависимости от того,чем является входная строка, создается объект соответствующего класса (операция,число или переменная) и возвращается указатель на него, приведенный к классу предку Lexeme, для возможности записать все в один массив;
   - double GetNum(string num) - вспомогательная функция для функции StrToNum. На вход принимает строку,содержащую только цифры и переводит ее в число;
   - double StrToNum(string str) - метод принимает на вход строку, содержащую число, производит проверку на корректность записи числа и ,если возможно, преобразует входную строку в цисло типа double;
   - unsigned int GetPriority(Operation op) - метод принимает на вход операция и возвращает приоритет ее выполнения;
   - void Parse() - метод разбивает строку infix на части,которые преобразует в лексемы, используя методы GetNum и CreateLexeme. Так же метод проверяет на соответствие математическому синтаксису порядок следования лексем,если обнаруживается несоответствие ,то происходит выбрасывание исключения;
   - void ToPostfix() - метод переводит в постфиксное представление исходное выражение записанное в векторе лексем.

   public:

   - TPostfix(const string & str) -
   - void SetInfix(const string & str) - метод позволяет перезадать исходное математическое выражение;
   - string GetInfixStr() const - метод возвращает строку ,содержащую исходное математическое выражение;

- string GetPostfixStr() const - метод вызвращает строку ,содержащую математическое выражение в постфиксной форме;
- double Calculate() - метод вычисляет математическое выражение и возвращает полученное значение. При возникновении ошибок метод бросает исключения. Если в выражении присутствуют переменные, метод запросит их значения через консоль;
- static void ShowInfo() - метод выводи информацию о доступных функциях программы;
- ~TPostfix() - деструктор.

3. Класс Lexem

   Представляет минимально возможную часть арифметического выражения, является базовым классом для класса операция (Operation), операнд (Operand).

   private:

   - const pair<int, int> pos - хранит позицию лексемы в строке.Первый индекс это начало лексемы, второй - конец;

   public:

   - Lexeme(pair<int, int> pos) - конструктор принимающий позицию начала и конца лексемы в строке;
   - pair<int, int> GetPos() - метод возвращающий позицию начала и конца лексемы в строке;
   - virtual const bool isOperation() const noexcept = 0 виртуальный метод позволяющий отлечить класс потомок Operation от класса потомка Operand;
   - virtual const string GetName() const = 0 - виртуальный метод возвращающий имя лексемы. Для функции это будет имя функции, для переменной - имя переменной, для константы - ее строковое представление;

4. Класс Operation

   Класс наследуется от класса Lexem. Является представлением математической операции.

   private:

   - const string name - хранит имя операции.Операциями являются: функция, математическая операция и круглые скобки;

   public:

   - Operation(string name, pair<int, int> pos) - конструктор класса, позволяет задать имя и позицию операции в строке;
   - const bool isOperation() const noexcept override - перегруженный метод из класса Lexem;
   - const string GetName() const - перегруженный метод из класса Lexem.

5. Класс Operand

   Класс наследуется от класса Lexem. Является представлением операнда. операнды это константы и переменные.

   public:

- Operand(pair<int, int> pos) - конструктор класса, позволяет задать имя и позицию операнда в строке;
- const bool isOperation() const noexcept override - перегруженный метод из класса Lexem;
- virtual double GetValue() const = 0 - виртуальный метод, позволяющий получить значение, которое хранит операнд(константа или переменная).

6. Класс Variable

   Класс наследуется от класса Operand. Является представлением переменной.

   private:

   - const string name - хранит имя переменной;
   - map<string, double> & var_map - хранит ссылку на мап со значениями переменных. Имена "e"и "pi"зарезервиврованы под математические константы и в случае их использование будет возвращено их значение;

   public:

   - Variable(string name ,map<string, double> & variables, pair<int, int> pos)- конструктор класса, позволяет задать имя, мап со значениями и позицию переменной в строке;
   - double GetValue() const override - перегруженный метод из класса Operand;
   - const string GetName() const override - перегруженный метод из класса Lexem.

7. Класс Constant

   Класс наследуется от класса Operand. Является представлением константы.

   private:

   - const double value - числовое значение константы;
   - const string name - числовое значение константы в виде строки.

   public:

   - Constant(double value,string name,pair<int, int> pos) - конструктор класса, позволяет задать имя,значение и позицию константы в строке;
   - double GetValue() const noexcept override - перегруженный метод из класса Operand;
   - const string GetName() const override - перегруженный метод из класса Lexem.

8. Класс TPostfixException

   Класс наследуется от класса exception. Класс хранит и выводит текст ошибки и позицию в строке,где эта ошибка произошла.

   private:

   - const pair<int, int> pos - хранит позицию в строке в которой произошла ошибка.Первый индекс это начало, второй - конец;

public:

- TPostfixException(const char* message,pair<int,int> pos) - конструктор класса. Первый параметр - текст сообщения об ошибке,второй - позиция начала и конца промежутка возникновения ошибки в строке;
- TPostfixException(char* message,int pos) - конструктор класса. Первый параметр - текст сообщения об ошибке,второй - позиция места возникновения ошибки в строке;
- string what() - возвращает строку ,содержащую текст ошибки и позицию ее возникновения;
- pair<int, int> GetPos() - возвращает пару, котрая хранти начало и конец промежутка возникновения ошибки в строке.

5. Подтверждение корректности работы

Для подтверждения корректности работы программы, были реализованы Google Tests. В тестах были реализованы проверка на корректность работы всех функций по отдельности, были написаны тесты проверяющие корректное вычисления для различных арифметических выражений и тесты проверяющие корректность отлова ошибок при вводе арифметического выражения.

Корректность работы программы в целом подтверждается правильнстью работы пользовательского приложения.

6. Заключение

В ходе выполнения лабораторной работы была разработана структуры данных стек и использована для расчета арифметических выражений с использованием обратной польской записи.

Был разработан интерфейс шаблонного класса TStack,реализованы методы шаблонного класса TStack,разработан интерфейс класса TPostfix для работы с постфиксной формой, реализованы методы класса TPostfix, разработаны и реализованы тесты для классов TStack и TPostfix на базе Google Test.

Исходные коды программы были опубликованы в моем личном репозитории на GitHub.