Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение  
высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

Институт Информационных технологий, математики и механики

Отчёт по учебной практике

**Вычисление арифметических выражений**

**Выполнил:**

студент гр. 381806-1

Касьянычев Михаил Петрович

**Проверил:**

кандидат технических наук, доцент кафедры МОСТ ИИТММ

Кустикова Валентина Дмитриевна

Нижний Новгород

2019 г.

Оглавление

[Оглавление 2](#_Toc28207562)

[Введение 3](#_Toc28207563)

[Постановка задачи 4](#_Toc28207564)

[Руководство пользователя 5](#_Toc28207565)

[Руководство программиста 6](#_Toc28207566)

[Структура программы 6](#_Toc28207567)

[Описание алгоритма 6](#_Toc28207568)

[Класс Stack 6](#_Toc28207569)

[Преобразование выражения в постфиксную форму 6](#_Toc28207570)

[Значения операндов и подсчет итогового значения 7](#_Toc28207571)

[Описание функций 8](#_Toc28207572)

[Функции класса Stack 8](#_Toc28207573)

[Функции класса Calculator 9](#_Toc28207574)

[Заключение 9](#_Toc28207575)

[Литература 10](#_Toc28207576)

[Приложения 11](#_Toc28207577)

[Приложение 1. Основная функция 11](#_Toc28207578)

[Приложение 2. Класс Stack 11](#_Toc28207579)

[Приложение 3. Класс Calculator 13](#_Toc28207580)

Введение

Целью лабораторной работы является изучение и практическое применение структуры Стек. Темой данной работы является разработка программы, позволяющей вычислять значения произвольного арифметического выражения при помощи структуры Стек.

Постановка задачи

Разработать программу для вычисления значения арифметического выражения при введенном выражении и значениях переменных.

Входные данные:

1. Арифметическое выражение.
2. Значения переменных.

Выходные данные:

1. Постфиксная форма выражения.
2. Значение выражения.

Руководство пользователя

После запуска программы откроется интерфейс с предложением ввести арифметическое выражение (рис.1).



Рис.1. Запрос на ввод выражения

После введения выражения нажмите Enter. Если введенное выражение корректно, программа выведет постфиксную форму выражения и предложит ввести значения указанных переменных (рис.2).

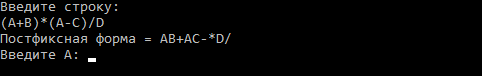


Рис.2. Программа вывела постфиксную форму и ожидает ввода переменной А

После ввода значений всех переменных программа посчитает значение выражения и выведет результат (рис.3).

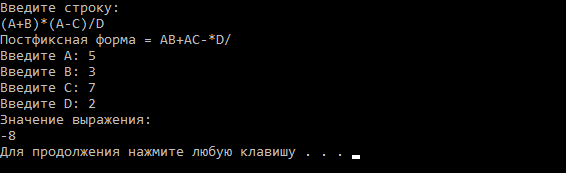


Рис.3. Программа вывела значение выражения

Если вы введете некорректное выражение или значение переменной, при которой значение выражения не вычисляется, программа выдаст ошибку и завершит работу (рис.4).

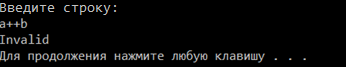


Рис.4. Введено неверное выражение

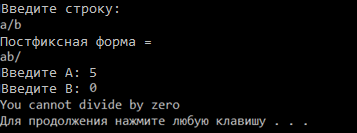


Рис.5. Попытка деления на ноль

Руководство программиста

Структура программы

Программа состоит из 4-х файлов:

1. Main.cpp, который содержит функцию main.
2. Stack.h, который содержит класс Stack.
3. Calculator.h, который содержит класс Calculator.
4. Calculator.cpp, который содержит реализацию функций класса Calculator.

Описание алгоритма

Класс Stack

Стек ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) stack — стопка; читается стэк) — [абстрактный тип данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), 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)), организованных по принципу [LIFO](https://ru.wikipedia.org/wiki/LIFO) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) lastin — firstout, «последним пришёл — первым вышел»).

Характеристики стека:

1. Массив данных
2. Индекс последнего занятого
3. Размер стека

Методы стека:

1. **Добавление элемента в стек:** если стек не полный, то увеличить значение индекса последнего не занятого на единицу и положить по этому индексу элемент в массив данных
2. **Получение данных на вершине стека:** если стек не пустой, то возвращается значение по индексу последнего занятого
3. **Удаление данных:** если стек не пустой, то возвращается значение по индексу последнего занятого и индекс последнего занятого уменьшается на единицу
4. **Проверка на пустоту:** если значение последнего занятого равно -1, то стек пуст
5. **Проверка на полноту:** если значение последнего занятого равно размеру стека – 1, то стек полон

Преобразование выражения в постфиксную форму

1. Создаются 2 стека: постфиксная форма и стек операторов
2. Поочередно просматривается каждый символ арифметического выражения
3. Если символ является арифметическим знаком («+», «-», «\*», «/») или знак открывающейся скобки «(», он отправляется в стек операторов
4. Если символ является закрывающейся скобкой «)», достаются все элементы из стека операторов до «(» и добавляются в стек постфиксной формы
5. Если символ операнд, он добавляется в стек постфиксной формы
6. После завершения цикла все элементы стека операторов перекладываются в стек постфиксной формы
7. Все элементы стека постфиксной формы добавляются в итоговую строку, разделяясь пробелами [[1]](#Таблица1)

Таблица 1. Пример преобразования в постфиксную форму

Исходная строка: A+(B-С)\*A

|  |  |  |
| --- | --- | --- |
| Элемент строки | Постфиксная форма | Стек операторов |
| A | A |  |
| + | A | + |
| ( | A | +( |
| B | AB | +( |
| - | AB | +(- |
| С | ABС | +(- |
| ) | ABС- | + |
| \* | ABС- | +\* |
| A | ABС-A | +\* |
|  | ABС-A\*+ |  |

Итоговая строка: ABС–A\*+

Значения операндов и подсчет итогового значения

1. На вход алгоритма подаётся выражение, записанное в польской нотации и значения переменных. На выходе имеем значение выражения.
2. Перед началом работы создаётся стек, в который будут подаваться промежуточные результаты, и на вершине которого в результате работе алгоритма окажется значение выражения.
3. В ходе алгоритма выражение просматривается слева направо, в зависимости от того, поступил оператор или операнд проводятся действия:
4. Если поступил операнд, то кладём в стек.
5. Если поступил оператор, то изымаем два значения из стека и проводим над ними операцию в порядке второй-первый.
6. В результате работы на вершине стека лежит значение выражения.

Рассмотрим в качестве примера SYN+G/\*

Таблица 2. Помещаем операнд в стек

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | S | Y | N | + | G | / | \* |
| Стек | S |  |  |  |  |  |  |

Таблица 3. Помещаем операнд в стек

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | S | Y | N | + | G | / | \* |
| Стек | S | Y |  |  |  |  |  |

Таблица 4. Помещаем операнд в стек

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | S | Y | N | + | G | / | \* |
| Стек | S | Y | N |  |  |  |  |

Таблица 5. Складываем 2 верхних операнда и результат помещаем в стек

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | S | Y | N | + | G | / | \* |
| Стек | S | Y+N |  |  |  |  |  |

Таблица 6. Помещаем операнд в стек

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | S | Y | N | + | G | / | \* |
| Стек | S | Y+N | G |  |  |  |  |

Таблица 7. Выполняем деление

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | S | Y | N | + | G | / | \* |
| Стек | S | (Y+N)/G |  |  |  |  |  |

Таблица 8. Выполняем умножение

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | S | Y | N | + | G | / | \* |
| Стек | S\*(Y+N)/G |  |  |  |  |  |  |

Описание функций

Функции класса Stack

TStack(int \_size = 10)

Назначение: конструктор, инициализация объекта

Входные параметры: размер стека

Возвращаемое значение: нет

TStack(const TStack<ValueType>& tmp)

Назначение: конструктор копирования, создание копии текущего стека

Входные параметры: константная ссылка на копируемый объект

Возвращаемое значение: нет

~TStack()

Назначение: деструктор, высвобождение памяти и удаление объекта класс

Входные параметры: нет

Возвращаемое значение: нет

void Push(ValueType c)

Назначение: добавление значения в стек

Входные параметры: добавляемое значение

Возвращаемое значение: нет

ValueType Pop()

Назначение: извлечение элемента из стека

Входные параметры: нет

Возвращаемое значение: извлекаемое значение

ValueType Top() const

Назначение: просмотр значения на вершине стека

Входные параметры: нет

Возвращаемое значение: значение на вершине стека

bool IsEmpty() const

Назначение: проверка стека на пустоту

Входные параметры: нет

Возвращаемое значение: логическая переменная

bool IsFull() const

Назначение: проверка стека на полноту

Входные параметры: нет

Возвращаемое значение: логическая переменная

Функции класса Calculator

Calculator - класс, реализующий методы для работы с арифметическими выражениями

static string CreatePostForm(const string&)

Назначение: создание постфиксной формы

Входные параметры: исходное выражение

Возвращаемое значение: преобразованное в постфиксную форму исходное выражение

static double Calculate(const string&, const double\*, const char\*, int)

Назначение: вычисление выражения при помощи постфиксной формы

Входные параметры: исходное выражение, записанное в постфиксной форме, массив значений переменных, массив символов, количество переменных

Возвращаемое значение: результат вычисления выражения

static void GetOper(const string&, double\*&, char\*&, int\*)

Назначение: получение значений переменных

Входные параметры: исходное выражение, записанное в постфиксной форме, ссылка на переменную которая содержит количество переменных

Возвращаемое значение: массив значений переменных

Static bool Prior(char, char)

Назначение: сравнение приоритета пришедшей операции и операции на вершине стека

Входные параметры: 2 значения операций

Возвращаемое значение: логическая переменная

Static double Calc(double, double, char)

Назначение: калькулятор

Входные параметры: значения переменных и арифметическое действие в типе char

Возвращаемое значение: результат вычисления

Заключение

Разработана программа для вычисления значения арифметического выражения при введенном выражении и значениях переменных с выводом постфиксной формы выражения.

Литература

Горьков Алексей Обратная польская запись: [<https://habr.com/ru/post/100869/>] 03.08.2010.

Приложения

Приложение 1. Основная функция

void main()

{

setlocale(LC\_ALL, "Russian");

string s, rez;

int size = 1;

int\* h = &size;

double PFV = 0;

cout << "Введите строку:" << endl;

cin >> s;

try

{

rez = Calculator::CreatePostForm(s);

}

catch(char \*osh)

{

cout << osh << endl;

}

cout << "Постфиксная форма = ";

cout << rez << endl;

double\* ex = new double[rez.length()];

char\* oper = new char[rez.length()];

try

{

Calculator::GetOper(rez, ex, oper, \*h);

}

catch (char\* osh)

{

cout << osh << endl;

}

try

{

PFV = Calculator::Calculate(rez, ex, oper, \*h);

}

catch (char\* osh)

{

cout << osh << endl;

}

cout << "Значение выражения: " << endl;

cout << PFV << endl;;

system("pause");

}

Приложение 2. Класс Stack

template<class ValueType>

class TStack

{

private:

int size;

ValueType \*arr;

int top;

public:

TStack(int \_size = 10);

TStack(const TStack<ValueType>& tmp);

~TStack();

void Push(ValueType c);

ValueType Pop();

ValueType Top() const;

TStack& operator=(const TStack& tmp);

bool IsFull()const;

bool IsEmpty()const;

};

template<class ValueType>

TStack<ValueType>::TStack(int \_size)

{

size = \_size;

top = -1;

arr = new ValueType[size];

}

template<class ValueType>

TStack<ValueType>::TStack(const TStack<ValueType>& tmp)

{

size = tmp.size;

top = tmp.top;

arr = new ValueType[size];

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

arr[i] = tmp.arr[i];

}

template<class ValueType>

TStack<ValueType>::~TStack()

{

size = 0;

top = -1;

delete[] arr;

}

template<class ValueType>

void TStack<ValueType>::Push(ValueType c)

{

if (this->IsFull())

throw "Full";

arr[++top] = c;

}

template<class ValueType>

ValueType TStack<ValueType>::Pop()

{

if (this->IsEmpty())

throw "Empty";

return arr[top--];

}

template<class ValueType>

ValueType TStack<ValueType>::Top() const

{

return arr[top];

}

template<class ValueType>

TStack<ValueType>& TStack<ValueType>::operator=(const TStack<ValueType>& tmp)

{

size = tmp.size;

top = tmp.top;

delete[] arr;

arr = new ValueType[size];

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

{

arr[i] = tmp.arr[i];

}

return \*this;

}

template<class ValueType>

bool TStack<ValueType>::IsFull() const

{

return (top == size);

}

template<class ValueType>

bool TStack<ValueType>::IsEmpty() const

{

return (top == -1);

}

Приложение 3. Класс Calculator

bool Calculator::Prior(char a, char b)

{

if ((a == '\*' || a == '/') && (b == '\*' || b == '/')) return true;

if ((a == '+' || a == '-') && (b == '+' || b == '-')) return true;

if ((a == '+' || a == '-') && (b == '\*' || b == '/')) return true;

return false;

}

double Calculator::Calc(double a, double b, char c)

{

switch (c)

{

case '+':

return a + b;

case '-':

return a - b;

case '\*':

return a \* b;

case '/':

if (b == 0) throw "You cannot divide by zero";

return a / b;

}

}

string Calculator::CreatePostForm(const string& s)

{

int left = 0;

int right = 0;

for (int i = 0; i < s.length(); i++) {

if (s[i] == '(') left++;

if (s[i] == ')') right++;

}

if (left != right) throw "Invalid";

if ((s[0] == '+') || (s[0] == '-') || (s[0] == '\*') || (s[0] == '/')) throw "Invalid";

if (s[s.length() - 1] == '+' || s[s.length() - 1] == '-' || s[s.length() - 1] == '\*' || s[s.length() - 1] == '/') throw "Invalid";

for (int i = 0; i < s.length(); i++)

if ((s[i] == '1') || (s[i] == '2') || (s[i] == '3') || (s[i] == '4') || (s[i] == '5') || (s[i] == '6') || (s[i] == '7') || (s[i] == '8') || (s[i] == '9') || (s[i] == '0'))

throw "Invalid";

for (int j = 1; j < s.length(); j++)

if ((s[j] != '+') && (s[j] != '-') && (s[j] != '\*') && (s[j] != '/') && (s[j] != '(') && (s[j] != ')') && (s[j - 1] != '+') && (s[j - 1] != '-') && (s[j - 1] != '\*') && (s[j - 1] != '/') && (s[j - 1] != '(') && (s[j - 1] != ')'))

throw "Invalid";

for (int i = 1; i < s.length(); i++)

if (((s[i] == '+') || (s[i] == '-') || (s[i] == '\*') || (s[i] == '/')) && ((s[i - 1] == '+') || (s[i - 1] == '-') || (s[i - 1] == '\*') || (s[i - 1] == '/')))

throw "Invalid";

TStack<double> operands(s.length());

TStack<double> operators(s.length());

string PFForm = "";

for (int i = 0; i < s.length(); i++)

{

bool f = (s[i] == '+') || (s[i] == '-') || (s[i] == '\*') || (s[i] == '/') || (s[i] == '(') || (s[i] == ')');

if (!f)

{

operands.Push(s[i]);

continue;

}

if (f)

{

if (s[i] == ')')

{

while (operators.Top() != '(')

operands.Push(operators.Pop());

operators.Pop();

continue;

}

if (Prior(s[i], operators.Top()))

{

while ((Prior(s[i], operators.Top())) && !(operators.IsEmpty()) && (operators.Top() != '('))

operands.Push(operators.Pop());

operators.Push(s[i]);

continue;

}

operators.Push(s[i]);

}

}

while (!operators.IsEmpty())

operands.Push(operators.Pop());

while (!operands.IsEmpty())

operators.Push(operands.Pop());

while (!operators.IsEmpty())

PFForm += operators.Pop();

return PFForm;

}

double Calculator::Calculate(const string& s, const double\* a, const char\* c, int h)

{

TStack<double> L(h);

for (int i = 0; i < s.length(); i++)

{

if ((s[i] == '+') || (s[i] == '-') || (s[i] == '\*') || (s[i] == '/'))

{

double first = L.Pop();

double second = L.Pop();

L.Push(Calc(second, first, s[i]));

}

else

{

for (int j = 0; j < h; j++)

if (c[j] == s[i])

L.Push(a[j]);

}

}

return L.Pop();

}

void Calculator::GetOper(const string& s, double\*& a, char\*& c, int& h)

{

h--;

for (int i = 0; i < s.length(); i++)

if ((s[i] != '\*') && (s[i] != '/') && (s[i] != '+') && (s[i] != '-'))

h++;

a = new double[h];

c = new char[h];

int P = 0;

int W;

for (int i = 0; i < s.length(); i++)

{

W = 0;

if (s[i] != '\*' && s[i] != '/' && s[i] != '+' && s[i] != '-')

{

for (int j = 0; j < P; j++) if (c[j] == s[i])

{

W = 1;

break;

}

if (W == 0)

{

c[P] = s[i];

cout << "Введите " << s[i] << ": ";

cin >> a[P];

P++;

}

}

}

}