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**

[Введение 3](#_Toc35880130)

[Постановка задачи 4](#_Toc35880131)

[Руководство пользователя 5](#_Toc35880132)

[Руководство программиста 6](#_Toc35880133)

[Структура программы 6](#_Toc35880134)

[Описание структур данных 6](#_Toc35880135)

[Класс Stack 6](#_Toc35880136)

[Класс Calculator 7](#_Toc35880137)

[Описание алгоритма 8](#_Toc35880138)

[Класс Stack 8](#_Toc35880139)

[Преобразование выражения в постфиксную форму 8](#_Toc35880140)

[Подсчет итогового значения 9](#_Toc35880141)

[Заключение 10](#_Toc35880142)

[Литература 11](#_Toc35880143)

[Приложения 12](#_Toc35880144)

[Приложение 1. Основная функция 12](#_Toc35880145)

[Приложение 2. Класс Stack 12](#_Toc35880146)

[Приложение 3. Класс Calculator 14](#_Toc35880147)

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. Строка с арифметическим выражением.
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. . 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).

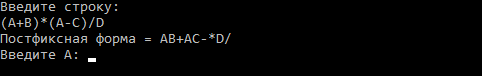


Рис. . Программа вывела постфиксную форму и ожидает ввода переменной А

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

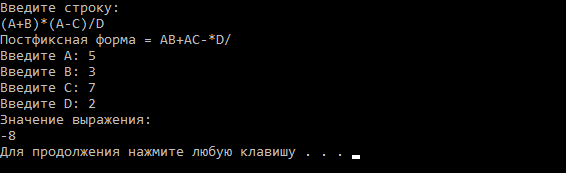


Рис. . Программа вывела значение выражения

Если вы введете некорректное выражение или значение переменной, при которой значение выражения не вычисляется, программа выдаст ошибку и завершит работу (Рис. 4), (Рис. 5).

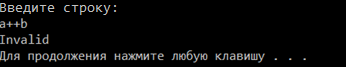


Рис. . Введено неверное выражение

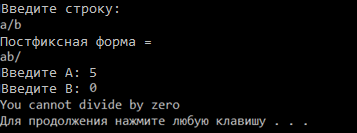


Рис. . Попытка деления на ноль

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 структур данных

Класс Stack

Stack – класс, реализующий функционал структуры данных стек.

**Объявление:**

template<class ValueType>

class TStack {

public:

ValueType\* arr;

int size;

int top;

public:

TStack(int size = 10);

TStack(const TStack&);

~TStack();

void Push(ValueType);

void Pop();

bool Is Empty() const;

bool Is Full() const;

ValueType Top() const;

};

**Описание методов класса:**

TStack(int size = 10)

Назначение: конструктор, инициализация объекта.

Входные параметры: размер стека.

Возвращаемое значение: нет.

TStack(const TStack&)

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)

Назначение: добавление значения в стек.

Входные параметры: добавляемое значение.

Возвращаемое значение: нет.

void Pop()

Назначение: извлечение элемента из стека.

Входные параметры: нет.

Возвращаемое значение: извлекаемое значение.

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.

ValueType Top() const

Назначение: просмотр значения на вершине стека.

Входные параметры: нет.

Возвращаемое значение: значение на вершине стека.

Класс Calculator

Calculator – класс, реализующий методы для работы с арифметическими выражениями.

**Объявление**

static class Calculator {

private:

static bool Prior(char, char);

static double Calc(double, double, char);

public:

static string CreatePostForm(const string&);

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

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

};

**Описание методов класса**

static string CreatePostForm(const string&)

Назначение: создание постфиксной формы.

Входные параметры: исходное выражение.

Возвращаемое значение: преобразованное в польскую нотацию исходное выражение.

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

Назначение: вычисление выражения при помощи польской нотации.

Входные параметры: исходное выражение, записанное в польской нотации, массив значений переменных, количество переменных.

Возвращаемое значение: результат вычисления выражения.

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

Назначение: получение значений переменных.

Входные параметры: исходное выражение, записанное в польской нотации, 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.

Описание алгоритма

Преобразование выражения в постфиксную форму

На вход приходит строка, содержащая выражение, на выходе получается строка, содержащая постфиксную форму.

Перед выполнением алгоритма создаем два стека, в (1) стек будем записывать операнды, во (2) стек будем записывать операции

**Алгоритм:**

1. Если пришел операнд, то записываем в стек (1).
2. Если пришла левая скобка, то записываем в стек (2).
3. Если пришла правая скобка, то перезаписываем все значения из стека (2) в стек (1), пока не встретится левая скобка, которую тоже удаляем.
4. Если пришла операция, то сравниваем ее приоритет с приоритетом операции на вершине стека.
5. Если пришла операция с более высоким приоритетом чем на вершине стека, то записываем ее в стек (2).
6. Если приоритет пришедшей операции ниже или равен приоритету операции на вершине стека, то перезаписываем все операции с большим или равным приоритетом из стека (2) в стек (1), пока не встретим левую скобку или не дойдем до конца стека или не встретим операцию с меньшим приоритетом.
7. После того, как дошли до конца строки перекладываем все значения из стека (2) в стек (1), а затем при помощи конкатенации строк формируем постфиксную форму.

Рассмотрим в качестве примера выражение A\*(B+C)/D.

Таблица 1. Помещаем операнд в стек 2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 |  |  |  |  |  |  |  |  |  |
| Стек 2 | A |  |  |  |  |  |  |  |  |

Таблица 2. Помещаем операцию в стек 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 | \* |  |  |  |  |  |  |  |  |
| Стек 2 | A |  |  |  |  |  |  |  |  |

Таблица 3. Помещаем открывающую скобку в стек 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 | \* | ( |  |  |  |  |  |  |  |
| Стек 2 | A |  |  |  |  |  |  |  |  |

Таблица 4. Помещаем операнд в стек 2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 | \* | ( |  |  |  |  |  |  |  |
| Стек 2 | A | B |  |  |  |  |  |  |  |

Таблица 5. Помещаем операцию в стек 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 | \* | ( | + |  |  |  |  |  |  |
| Стек 2 | A | B |  |  |  |  |  |  |  |

Таблица 6. Помещаем операнд в стек 2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 | \* | ( | + |  |  |  |  |  |  |
| Стек 2 | A | B | C |  |  |  |  |  |  |

Таблица 7. Перемещаем все операции из стека 1 в стек 2 пока не встречаем открывающую скобку.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 | \* |  |  |  |  |  |  |  |  |
| Стек 2 | A | B | C | + |  |  |  |  |  |

Таблица 8. Помещаем операцию в стек 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 | \* | / |  |  |  |  |  |  |  |
| Стек 2 | A | B | C | + |  |  |  |  |  |

Таблица 9. Помещаем операнд в стек 2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 | \* | / |  |  |  |  |  |  |  |
| Стек 2 | A | B | C | + | D |  |  |  |  |

Таблица 10. Перемещаем все операции из стека 1 в стек 2(записано в обратном порядке).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выражение | A | \* | ( | B | + | C | ) | / | D |
| Стек 1 |  |  |  |  |  |  |  |  |  |
| Стек 2 | A | B | C | + | D | / | \* |  |  |

Подсчет итогового значения

1. На вход алгоритма подаётся выражение, записанное в постфиксной форме 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 ABC+D/\*.

Таблица 11. Помещаем операнд в стек.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | A | B | C | + | D | / | \* |
| Стек | A |  |  |  |  |  |  |

Таблица 12. Помещаем операнд в стек.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | A | B | C | + | D | / | \* |
| Стек | A | B |  |  |  |  |  |

Таблица 13. Помещаем операнд в стек.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | A | B | C | + | D | / | \* |
| Стек | A | B | C |  |  |  |  |

Таблица 14. Складываем 2 верхних операнда и результат помещаем в стек.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | A | B | C | + | D | / | \* |
| Стек | A | B+C |  |  |  |  |  |

Таблица 15. Помещаем операнд в стек.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | A | B | C | + | D | / | \* |
| Стек | A | B+C | G |  |  |  |  |

Таблица 16. Выполняем деление.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | A | B | C | + | D | / | \* |
| Стек | A | (B+C)/D |  |  |  |  |  |

Таблица 17. Выполняем умножение.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Об. п. н. | A | B | C | + | D | / | \* |
| Стек | A\*(B+C)/D |  |  |  |  |  |  |

Заключение

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

Литература

1. Обратная польская запись. Автор: Горьков А. 2010г.
2. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гегреля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.

Приложения

Приложение 1. Основная функция

void main()

{

setlocale(LC\_ALL, "Russian");

string s, rez;

int h = 1;

double PFV = 0;

cout << "Введите строку:" << endl;

cin >> s;

try

{

rez = Calculator :: CreatePostForm(s);

}

catch(const char \*osh)

{

cout << osh << endl;

system("pause");

return;

}

cout << "Постфиксная форма = ";

cout << rez << endl;

double\* ex;

char\* oper;

try

{

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

}

catch (const char\* osh)

{

cout << osh << endl;

system("pause");

return;

}

try

{

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

}

catch (const char\* osh)

{

cout << osh << endl;

system("pause");

return;

}

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

cout << PFV << endl;

system("pause");

}

Приложение 2. Класс Stack

template<classValueType>

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);

boolIsFull()const;

boolIsEmpty()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 (inti = 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>

void TStack<ValueType>::Pop()

{

if (this -> IsEmpty())

throw "Empty";

top--;

}

template<class ValueType>

ValueType TStack<ValueType>::Top() const

{

return arr[top];

}

template<class ValueType>

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

{

size = tmp.size;

top = tmp.top;

delete[] arr;

arr = newValueType[size];

for (inti = 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.Top());

operators.Pop();

}

operators.Pop();

continue;

}

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

{

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

{

operands.Push(operators.Top());

operators.Pop();

}

operators.Push(s[i]);

continue;

}

operators.Push(s[i]);

}

}

while (!operators.IsEmpty())

{

operands.Push(operators.Top());

operators.Pop();

}

while (!operands.IsEmpty())

{

operators.Push(operands.Top());

operands.Pop();

}

while (!operators.IsEmpty())

{

PFForm += operators.Top();

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.Top();

L.Pop();

double second = L.Top();

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.Top();

}

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

{

h = 0;

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++;

}

}

}

}