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

**Отчет по лабораторной работе**

**«Вычисление арифметических выражений»**

**Выполнил**:

студент группы 382003-1

Медведев М. С.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2021

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 8](#_Toc26962566)

[Заключение 10](#_Toc26962568)

[Приложение 11](#_Toc26962569)

# Постановка задачи

Передо мной стояла задача разработать программу, которая выполняла бы вычисление арифметического выражения с вещественными числами. Выражение в качестве операндов может содержать переменные и вещественные числа. Допустимые операции известны: +, -, /, \*. Допускается наличие знака “-” в начале выражения или после открывающейся скобки. Программа должна выполнять предварительную проверку корректности выражения и сообщать пользователю вид ошибки и номера символов строки, в которых были найдены ошибки.

# Метод решения

Для того, чтобы реализовать программу, нужно знать алгоритм перевода выражения в обратную польскую запись. Инфиксная запись: a+b. Обратная польская запись: ab+.

Для начала нужно у каждой операции определить приоритет:

|  |  |
| --- | --- |
| Операция | Приоритет |
| ( | 0 |
| +, - | 1 |
| \*, / | 2 |
| ^ | 3 |

На вход алгоритма поступает строка выражения в инфиксной записи. Каждый символ рассматривается следующим образом:

1. Числа переписываются в выходную строку.
2. ‘(‘ помещается в стек.
3. Если символ ‘)’, то из стека извлекаются все символы до следующей ‘(‘ и переписываются в выходную строку.
4. Если символ – это операция, то нужно посмотреть на стек:

А) Стек пуст, то знак операции из входной строки записывается в стек.

Б) Стек не пуст:

а) Пока приоритет операции на вершине стека не меньше, чем у той, которая поступает, выталкиваем верхний элемент стека в выходную строку.

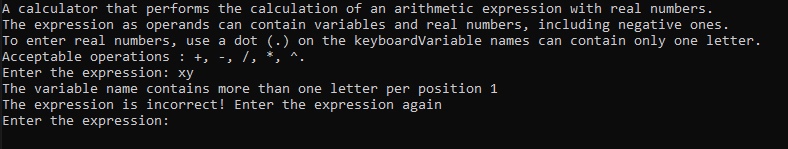
б) Помещаем операцию в стек.

5. Когда входная строка закончилась, выталкиваем все символы из стека в выходную строку.

# Руководство пользователя

При запуске программы пользователю будет предложено ввести выражение, которое он хочет посчитать. Программа покажет, какие операции доступны. В данном случае – это (+, -, \*, /). Также пользователь может ввести несколько параметров (любые латинские буквы) и задать их значения. После того, как выражение посчитается, пользователь получит численный ответ или сообщение об ошибке, если что-то введено неверно (вид ошибки и индекс ошибочного символа).

Пример неверного ввода:



В данном случае выражение имеет вид xy. Ошибка – имя переменной содержит более одной буквы на позиции 1.

После вывода типа ошибки будет предложено ввести выражение заново.

# Описание программной реализации

Программа состоит из четырёх проектов: gtest, sample, tests, arithmetic.

gtest содержит реализацию Google Tests.

sample содержит реализацию пользовательского приложения.

tests содержит тесты для стека и проверки всех функций.

arithmetic содержит реализацию алгоритма.

**arithmetic.cpp**

int DeterminationType(char s) – функция, определяющая тип элемента;

void ProcessingUnaryMinus(char\* s, char\* res) – обработка унарного минуса;

bool ThereIsUnaryMinus(char\* s) – проверка на наличие унарного минуса;

void ReplacementCommaToPointInDouble(char\* s) – функция замены запятой на точку;

bool CheckBrackets(char\* s) – проверка расставления скобок;

bool CheckAmountOperands(char\* s) – проверка на недостаток операндов;

bool CheckOperationsInRow(char\* s) – проверка на количество операций подряд;

bool CheckPlaceDotsOrCommas(char\* s) – проверка на правильную расстановку точек и запятых у double;

bool CheckCorrectnessOfSymbols(char\* s) – проверка на наличие некорректных символов;

bool CheckAmountLettersaInVars(char\* s) – проверка на количество букв в переменных;

bool CheckVariableToDigit(char\* s) - проверка на наличие знака умножения между операндами;

bool CheckStr(char\* s) - проверка на выполнение всех условий;

int Prioritet(char\* s) – приоритет операций;

bool IsSign(char\* s) – определение знак операции или нет

bool IsOperation(char\* s) – определение знак операции или нет без скобок;

void FindVars(const char\* s, int\* res) – нахождение переменных в формуле;

bool CheckCorrecnessOfValuesOfVars(string s) – проверка на корректность введённых значений;

void InputValues(char\* s, char\* res) – функция для ввода значений переменных;

double GetNumber(char\* s) – получение числа из char;

void ConvertInPostfixNotation(const char\* s, char\* res) – перевод в постфиксную запись;

double EvaluationOfExpression(char\* s) – вычисление выражения;

**stack.h**

Class TStack:

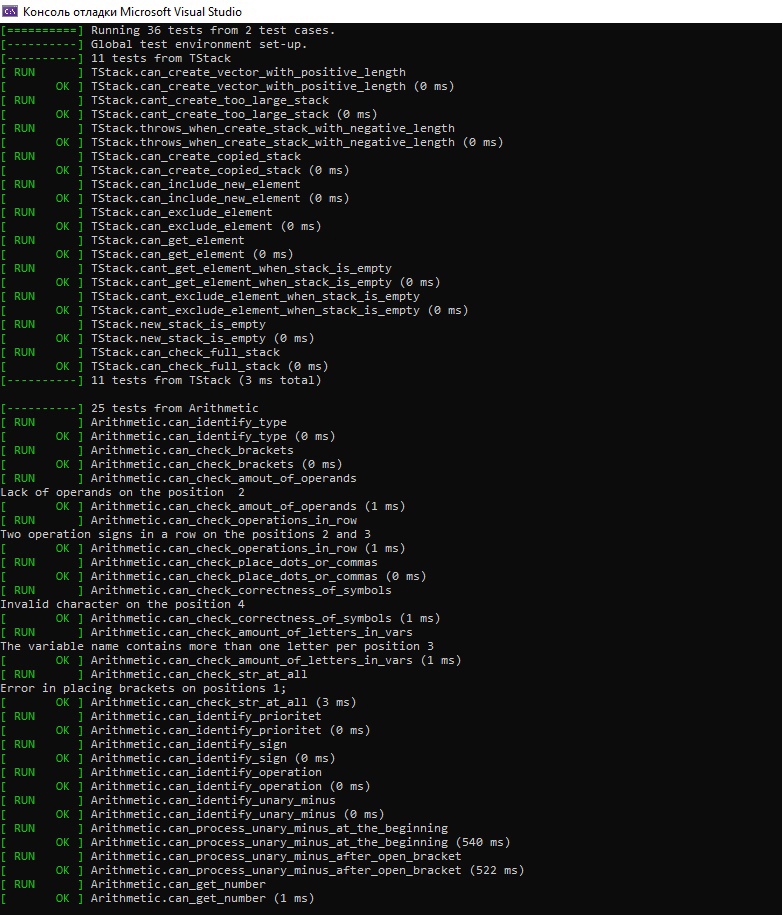
* TStack(int n = MAXSIZE) – инициализация;
* TStack(const TStack& v) – конструктор копирования;
* ~TStack() - деструктор;
* ValType Pop() – извлечение элемента;
* ValType Get() – извлечение элемента без удаления;
* void Push(double elem) – вставка элемента;
* bool IsEmpty() – проверка на пустоту;
* ValType View() – просмотр верхнего элемента;
* int End() – получение индекса последнего элемента;
* int Size() – получение количества элементов в стеке;
* void Clear() – очистка стека;

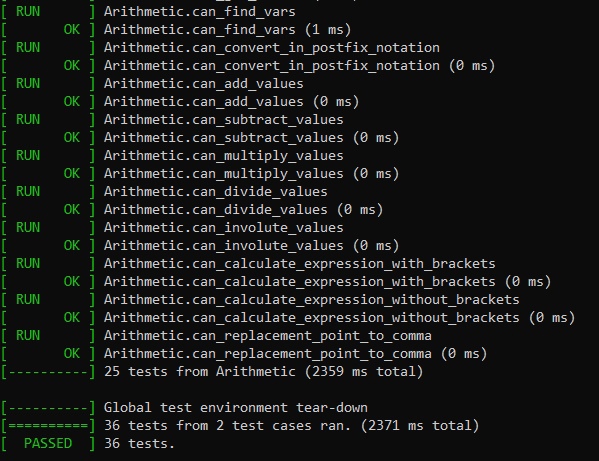
**arithmetic.h**

В данном файле объявлены функции, которые используются в одноимённом .cpp файле.

# Подтверждение корректности

Для подтверждения корректности работы моей программы реализованы тесты, проверяющие наличие каких-либо ошибок:





# Заключение

В ходе данной лабораторной работы мной была разработана программа, которая выполняет вычисление выражения с вещественными числами, и шаблонный класс TStack. Были реализованы различные проверки на корректность введённых данных. Все функции тестируются с помощью Google test-ов.

# Приложение

void ConvertInPostfixNotation(const char\* s, char\* res)//перевод в постфиксную запись

{

TStack<char> sg(256);

int len = strlen(s);

int type[256];

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

type[i] = DeterminationType(s[i]);

int j = 0; int m = 0;

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

{

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

sg.Push('(');

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

{

char a = sg.Pop();

while (a != '(')

{

res[j] = a;

j++;

a = sg.Pop();

}

}

if (IsOperation(s[i]))

{

if (sg.IsEmpty())

sg.Push(s[i]);

else

{

char op = sg.Get();

while (Prioritet(s[i]) <= Prioritet(op))

{

res[j] = sg.Pop();

j++;

if (sg.IsEmpty() != true)

op = sg.Get();

else

op = '(';

}

sg.Push(s[i]);

}

}

if (type[i] == 1)

{

if ((type[i + 1] == 1) || (type[i + 1] == 4))

{

res[j] = s[i];

j++;

}

else

{

res[j] = s[i];

j++;

res[j] = ' ';

j++;

}

}

if (type[i] == 2)

{

res[j] = s[i];

j++;

res[j] = ' ';

j++;

}

if (type[i] == 4)

{

res[j] = s[i];

j++;

}

}

while (sg.IsEmpty() != true)

{

res[j] = sg.Pop();

j++;

}

res[j] = '\0';

}

double EvaluationOfExpression(char\* s)//вычисление выражения

{

TStack <double> num(256);

int len = strlen(s);

int l = 0; int i = 0; int m = 0;

double res;

int type[256];

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

type[j] = DeterminationType(s[j]);

while (s[i] != '\0')

{

if (type[i] == 1)

{

char str[256];

int k = 0;

double number;

m = i;

while (s[m] != ' ')

{

str[k] = s[m];

m++;

k++;

}

str[k] = '\0';

l = m - i + 1;

number = GetNumber(str);

num.Push(number);

i = i + l - 1;

}

if (IsOperation(s[i]))

{

double opright, opleft;

opright = num.Pop();

opleft = num.Pop();

switch (s[i])

{

case '+':

{

res = opleft + opright;

break;

}

case '-':

{

res = opleft - opright;

break;

}

case '\*':

{

res = opleft \* opright;

break;

}

case '/':

{

res = opleft / opright;

break;

}

case '^':

{

int op;

op = (int)opright;

int count = 1;

res = opleft;

do

{

if (op == 0)

res = 1;

else if ((op != 1) && (op > 0))

res = res \* opleft;

count++;

} while (op > count);

break;

}

}

num.Push(res);

}

i++;

}

return num.Pop();

}