Федеральное государственное автономное учреждение

высшего образования

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

Институт информационных технологий математики и механики

Отчет по лабораторной работе

**Вычисление арифметических выражений**

Выполнил:

Студент ИИТММ гр. 0823-1

Кузнецов В.В.

Проверила:

Преподаватель кафедры МОСТ ИИТММ

Пирова А. Ю.

Нижний Новгород

2016г.

**Содержание**

1.[Введение 3](#_Toc469333591)

2.[Постановка учебно-практической задачи 4](#_Toc469333592)

3.[Руководство пользователя 5](#_Toc469333593)

4.[Руководство программиста 9](#_Toc469333594)

4.1.[*Описание структуры программы* 9](#_Toc469333595)

4.2.[*Описание структур данных* 11](#_Toc469333596)

4.3.[*Описание алгоритмов* 13](#_Toc469333597)

5.[Заключение 17](#_Toc469333598)

6.[Список литературы 18](#_Toc469333599)

7.[Приложение 19](#_Toc469333600)

7.1.[*Приложение 1* 19](#_Toc469333601)

7.2.[*Приложение 2* 24](#_Toc469333602)

7.3.[*Приложение 3* 25](#_Toc469333603)

7.4.[*Приложение 4* 34](#_Toc469333604)

## **Введение**

Задача вычислений является неотъемлемой частью жизни как человека, так и общества в целом. Ни один проект, ни одна техническая задача не может быть решена без использования математических вычислений. Ежедневно миллионы людей используют калькуляторы, компьютеры и другую вычислительную технику с целью решить некоторую задачу.

Также в ходе данной работы использована структура хранения данных – стек, которая является одной из основных структур в области программирования. Стек - [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)). Стек используется для организации [стековой машины](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C), реализующей вычисления в обратной инверсной записи.

Для отслеживания точек возврата из подпрограмм используется стек вызовов.

Арифметические [сопроцессоры](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80), программируемые микрокалькуляторы и язык [Forth](https://ru.wikipedia.org/wiki/Forth" \o "Forth) используют стековую модель вычислений.

Идея стека используется в стековой машине среди [стековых языков программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA).

В нашем случае стек используется для организации вычисления арифметических выражений.

В отчете приводится постановка задачи разработки и описания алгоритмов шаблонного класса стеков на основе шаблонного класса списков, а также набора функций, позволяющих разбивать исходное выражение на лексемы, проверять корректность выражения, переводить выражение в обратную польскую запись, вычислять выражение по польской записи, консольный интерфейс пользователя. Также в отчете представлено описание программы, правила её использования и код самой программы.

## **Постановка учебно-практической задачи**

Требуется разработать программу, выполняющую вычисление арифметического выражения с вещественными числами. Выражение в качестве операндов может содержать переменные и вещественные числа. Допустимые операции известны:

+, -, /, \*

Допускается наличие знака "-" (унарный минус) в начале выражения или после открывающей скобки.

Программа должна выполнять предварительную проверку корректности выражения и сообщать пользователю вид ошибки и номера символов строки, в которых были найдены ошибки.

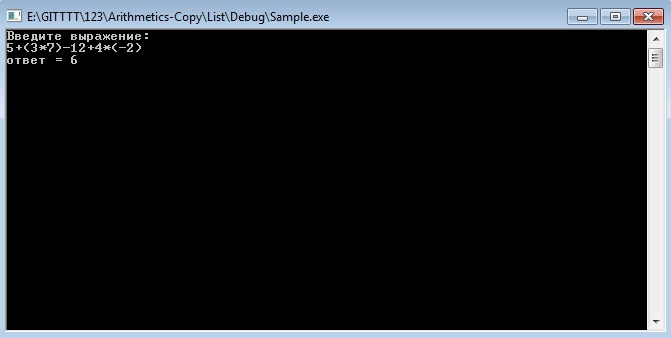
*Исходные данные:*

s – строка, содержащая исходное выражение;

*Требуемый рез-тат:*

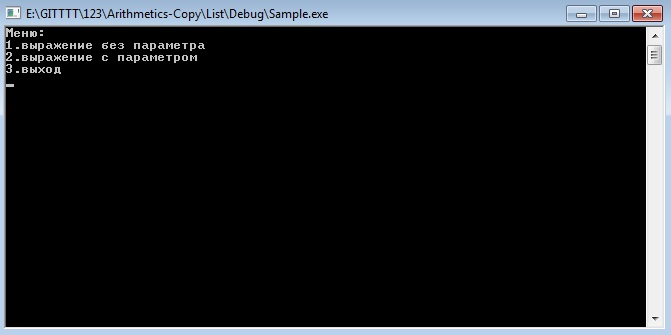
res – результат, полученный в ходе вычисления значения данного выражения

Контрольный пример:

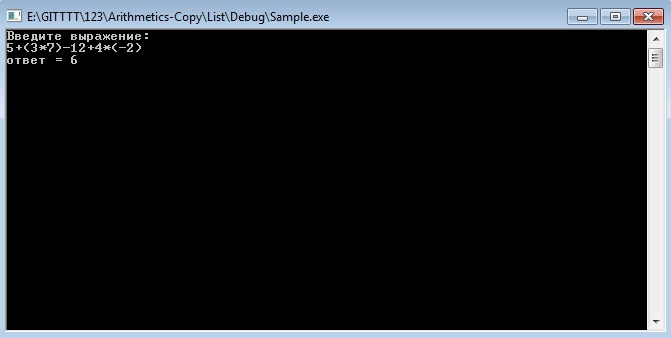


# **Руководство пользователя**

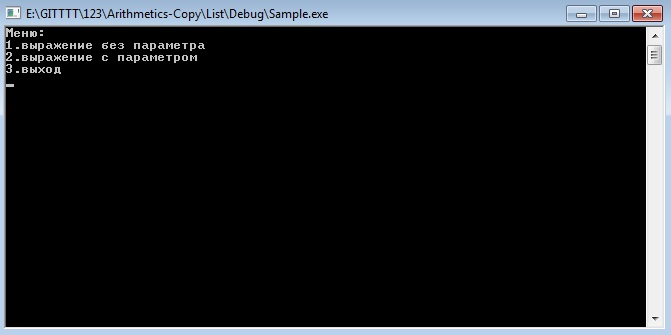
* Для работы с программой пользователю необходимо запустить файл Sample.exe;
* При запуске программы пользователю будет предложено одно из следующий действий:



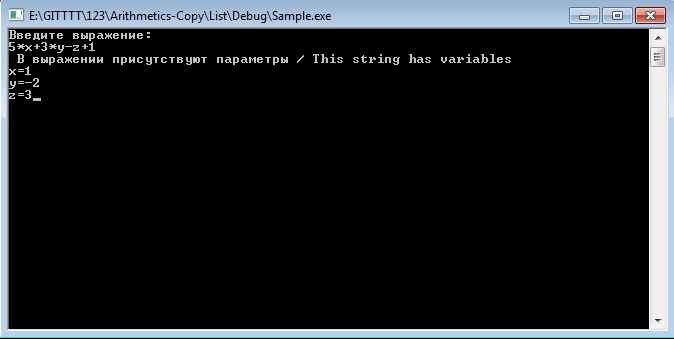
* При выборе первого пункта «выражение без параметра» пользователю будет предложено ввести выражение без параметра, и после нажатия клавиши «Enter» будет выведен результат вычислений:



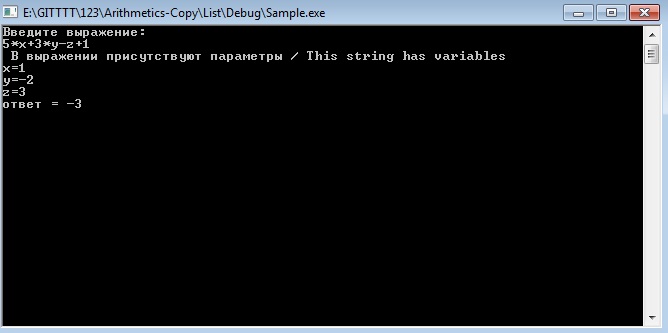
* Далее, при нажатии произвольной клавиши пользователь вернется в исходное меню:



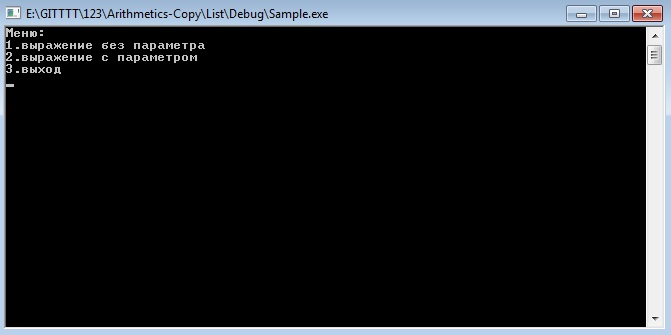
* При выборе второго пункта «выражение с параметром» пользователю будет предложено ввести соответствующее выражение и значения параметров выражения:



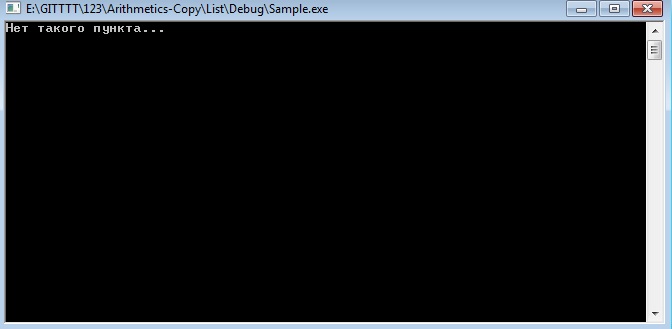
* После этого на экран будет выведен результат вычисления:



* Далее, при нажатии произвольной клавиши пользователь вернется в исходное меню:



* Чтобы выйти из программы пользователю необходимо выбрать 3-ий пункт;
* Если пользователь введет некорректное значение пункта меню, то на экране будет выведено сообщение об ошибке и пользователь будет возвращен в главное меню:



# **Руководство программиста**

# ***Описание структуры программы***

Разработана многофайловая система, состоящая из следующий составных частей:

Проекты:

1. gtest.lib – статическая библиотека, позволяющая использовать систему Google-test для тестирования работоспособности программы. Содержит файлы gtest.h и gtest-all.cc;
2. List.lib – статическая библиотека, содержащая интерфейс и реализацию списка, стека (на основе списка), а также набор функций, позволяющих вычислять арифметические выражения и их реализацию. Содержит файлы tlist.h, arithmetic.h и arithmetic.cpp;
3. Sample.exe – консольное приложение, позволяющее пользователю вычислять арифметические выражения. Содержит файл Sample.cpp;
4. test\_list.exe – консольное приложение, тестирующее работоспособность списка, стека и арифметических выражений. Содержит файлы test\_main.cpp, test\_list.cpp и test\_arithmetic.cpp.

Файлы:

gtest.h – заголовочный файл Google-test для тестирования работоспособности программы;

gtest-all.cc – реализация системы Google-test для тестирования работоспособности программы;

tlist.h - заголовочный файл, содержащий структуры данных: односвязный список и стек, с соответствующими интерфейсами шаблонных классов и реализациями;

arithmetic.h - заголовочный файл, содержащий набор функций, необходимый для вычисления арифметических выражений.

Файл arithmetic.h содержит следующий набор функций:

1. Определение приоритета операций:

int Priority(char);

1. Контроль корректности скобок:

bool Brackets(char \*);

1. Запись унарного минуса:

void UnarMinus(char \*,char \*);

1. Определение операции:

bool Operation(char );

1. Разбиение на лексемы:

int TypeChar(char );

1. Перевод в обратную польскую запись:

void ConvertInPostfix (const char\* ,char \*);

1. Перевод точки в запятую:

void PointToComma(char \*);

1. Поиск унарного минуса в выражении:

bool FindUnarMinus(char \*);

1. Проверка корректности ввода операций в начале, конце строки и со скобками:

bool CheckOperationsInBegEndWithBrack(char\* );

1. Проверка корректности ввода операций, вводимых пользователем подряд:

bool CheckOperationsInSuccession(char \*);

1. Проверка корректности записи символов:

bool CheckCorrectSymbols(char \*);

1. Проверка корректности расстановки точек и запятых:

bool CheckPositionOfDotAndComma(char \*);

1. Проверка наличия переменных:

bool CheckVariable(char \*);

1. Проверка корректности записи нескольких переменных подряд:

bool CheckManyVariablesInSuccession(char \*);

1. Все проверки корректности вместе взятые:

bool CheckStrAtallChecks(char \*);

1. Поиск и определение позиции параметра:

void FindWritePositVariable (const char \*, int \* );

1. Проверка корректности ввода числа:

bool CorrectOfNumb(char \*);

1. Перевод строки в число:

double Numb(char \*);

1. Ввод параметра:

void InputValuable(char \*,char \*);

1. Подсчет выражение, вывод результата:

double ResultsCount(char \*);

arithmetic.cpp – файл, содержащий реализацию функций, необходимых для вычисления арифметических выражений;

sample.cpp – файл, содержащий реализацию консольного интерфейса пользователя в виде меню;

test\_main.cpp – файл, позволяющий подключить систему Google-test в данный проект;

test\_list.cpp – файл, содержащий тесты для проверки работоспособности списка и стека;

test\_arithmetic.cpp - файл, содержащий тесты для проверки работоспособности арифметических выражений;

# ***Описание структур данных***

1. Class List, реализующего структуру данных: односвязный список, содержащий в качестве своих полей размер списка и указатель на первое звено. Список содержит следующие методы: конструктор без параметров, конструктор с параметрами, деструктор, получение размера списка, поиск и возвращение звена с заданным значением, проверка на пустоту, вставки звеньев в начало, конец и на заданную позицию, а также удаление звена с начала, конца и с заданной позиции списка, потоковый ввод, вывод, перегрузка индексации. Предварительно была создана структура: List\_node – звено списка, содержащее 2 поля: данное, хранящееся в звене и указатель на следующее звено.

В этом же файле присутствует реализация этих методов:

1. Конструктор без параметров:

List();

2. Конструктор с параметрами:

List (int);

3. Деструктор:

~List();

4. Получение размера списка:

int GetSize()

5. Поиск и возвращение звена с заданным значением:

List\_node<ValType>\* Search (ValType);

6. Перегрузка индексации:

ValType& operator[](int);

7. Проверка на пустоту:

bool Empty\_list();

8. Вставка звена с заданной позиции:

void Add(ValType ,List\_node<ValType> \*);

9. Вставка звена с начала списка:

void AddWithBegin(ValType);

10. Вставка звена в конец списка:

void AddWithEnd(ValType);

11. Удаления звена из начала списка:

ValType DeleteNodeWithBegin();

12. Удаление звена с конца списка:

void DeleteNodeWithEnd();

13. Удаление заданного звена:

void DeleteNodeWithPos(List\_node<ValType> \*);

14. Перегрузка потокового ввода:

friend istream & operator >> (istream &in, List &p);

15. Перегрузка потокового вывода:

friend ostream & operator << (ostream &out, const List &p);

1. Class Stack, реализующего структуру данных: стек, содержащий в качестве поля список. Стек обладает следующими методами: конструктор без параметров, проверка на пустоту, функция, которая «кладет» элемент в стек, удаление элемента из стека, очистка стека и возвращение «верхнего» элемента, лежащего в стеке.

Шаблонный класс стек содержит следующие методы:

1. Конструктор без параметров:

Stack();

1. Проверка на пустоту:

bool EmptyStack();

1. Функция, которая «кладет» элемент в стек:

void PushElem(const ValType);

1. Функция, которая удаляет элемент из стека:

ValType DelElem();

1. Очистка стека:

void ClearStack();

1. Функция, возвращающая «верхний» элемент стека:

ValType RetTopElem();

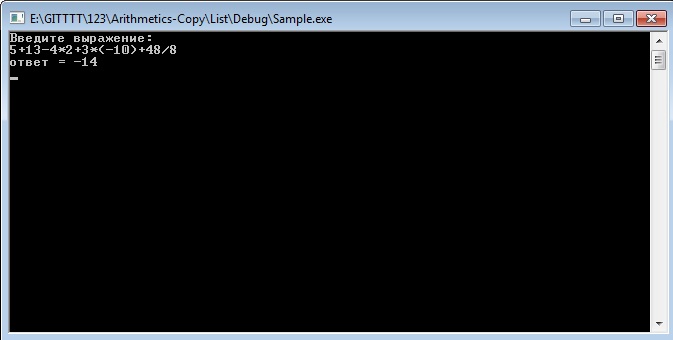
# ***Описание алгоритмов***

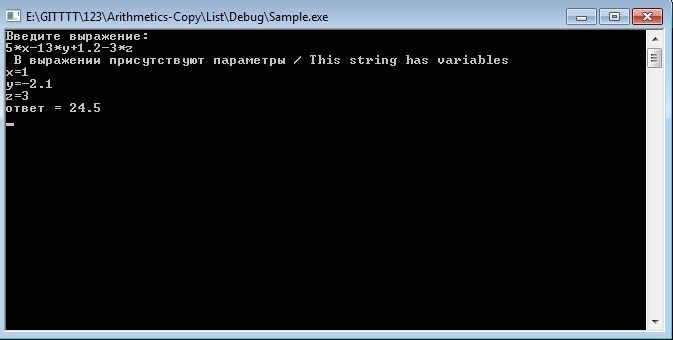
Описание наиболее важных алгоритмов необходимых для подсчета арифметических выражений:

1. Алгоритм определение приоритета операции:
   1. На вход функция получает символ исходной строки (причем этим символом будет являться оператор);
   2. Создается переменная prior, в которой будет хранится число, обозначающее приоритет операции;
   3. При помощи оператора switch происходит определение приоритета:
2. Если оператор – скобка, то приоритет равен 0;
3. Если оператор – плюс или минус, то приоритет равен 1;
4. Если оператор – умножить или разделить, то приоритет равен 2;
   1. На выходе возвращаем значение prior.
5. Алгоритм разбиения выражения на лексемы:
   1. На вход функция получает символ исходной строки;
   2. В зависимости от того какая лексема, функция будет возвращать различное значение типа int;
   3. В начале проверяем, является ли данный символ оператором. Для этого отдельно создана функция определения оператора, типа bool. Если это действительно оператор, то функция возвращает 1;
   4. Далее следует проверка на то, что символ является числом. Если это так, то будет возвращена 2;
   5. Далее – проверка скобок. Если символ – скобка, то возвращается значение, равное 3;
   6. Если символ – точка или запятая, то вернется значение, равное 4;
   7. Если символ – параметр, то будет возвращено значение, равное 5;
   8. В остальных случаях функция будет возвращать 0 (данные лексемы в нашей задаче являются лишними и впоследствии будет реализована функция, говорящая о некорректном вводе строки в случае обнаружения подобных символов).
6. Алгоритм перевода выражения в обратную польскую запись:
   1. На вход функция получает строку (выражение, которое было введено (унарный минус к этому моменту уже учтен)) и строку для записи результата;
   2. Создается стек для хранения операций;
   3. Далее начинаем идти по циклу до тех пор, пока не закончится наша строка (исходное выражение);
   4. Если встречается открывающаяся скобка, то кладем её в стек;
   5. Если встречается закрывающаяся скобка, то начинаем переносить операторы из стека в результирующую строку до тех пор, пока не встретим открывающуюся скобку;
   6. Если встретился оператор, то в начале проверяем пуст ли стек? Если он пустой, то просто кладем оператора в стек. Если в нем уже лежат какие-то операторы, то начинается сравнение их приоритетов. Если оператор имеет «больший» приоритет, он записывается в итоговую строку. Как только приоритет оператора становится меньше тех, что уже лежат в стеке, то данный оператор также заносится в стек;
   7. Если встретилась цифра, то заносим её в результирующую строку. Учитывается, есть ли за цифрой или точка, или запятая, или другая цифра. Если есть, то просто записываем цифру, если нет, то сделаем дополнительный пробел в результирующей строке;
   8. Если встретился параметр, то он вносится в результирующую строку и за ним следует пробел;
   9. Точку или запятую просто переносим в результирующую строку;
   10. После прохождения всей строки необходимо выполнить проверку на пустоту стека. Пока он не станет пустым, последовательно переносим все операции из него в результирующую строку.
7. Алгоритм записи унарного минуса:
   1. На вход функция получает строку (выражение, которое было введено) и строку для записи результата;
   2. Если минус попался в самом начале строки, то заносим в результирующую строку символы «0» и «-». Если же первый символ не минус, то переписываем его в результирующую строку.
   3. Если минус встретился где-то внутри строки, то проверяем несколько условий. Унарный минус должен идти после открывающейся скобки и перед числом или параметром. Если данное условие выполнилось, то заносим в результирующую строку символы «0» и «-». В ином случае это бинарный минус, и он просто переписывается в результирующую строку;
   4. Остальные символы просто переписываются в результирующую строку.
8. Алгоритм ввода параметра:
   1. На вход функция получает строку (выражение, которое было введено (уже в обратной польской записи)) и строку для записи результата;
   2. Создается вспомогательный массив, который заполняется значениями, равными: -1. В элементы этого массива вносятся позиции, на которых располагаются параметры исходного выражения;
   3. Далее следует проверка, если у нас первый элемент массива не равен -1, значит можно сделать вывод, что параметр в строке точно есть;
   4. Далее двигаемся по строке и если счетчик цикла не совпал со значением элемента вспомогательного массива, значит данный элемент строки не является параметром, и мы просто переписываем его в результирующую строку. Если эти 2 значения совпали, то это означает, что мы наткнулись на параметр. Далее происходит ввод значения параметра и проверка корректности введенного значения. Если параметр введен некорректно, на экране будет выведено сообщение об ошибке. В ином случае происходит проверка: имеется ли в записи значения параметра унарный минус. Если он есть, то в результирующую строку значение параметра будет записано в виде «0, пробел, число, заданное пользователем, пробел, -». Если унарного минуса нет, то просто переписываем значение параметра в результирующую строку;
   5. В конце не забываем удалить память, выделенную под вспомогательный массив.
9. Алгоритм подсчета результата вычисления арифметического выражения:
   1. На вход функция получает строку (выражение, которое было введено (к этому моменту учтены все проверки корректности ввода, унарный минус, значение параметра, а само выражение представлено в обратной польской записи));
   2. Для решения данной задачи создан стек, в котором будут храниться операнды для вычисления выражения;
   3. Если попадается цифра, то создается вспомогательный массив, в который вносится число целиком (его формат в общем случае char\*);
   4. Переводим число из символьного формата в вещественный и кладем в стек;
   5. Если попадается оператор, берем два верхних элемента стека, левый и правый операнды соответственно, и в зависимости от оператора выполняем соответствующую действие (сложение, вычитание, деление, умножение). Это осуществлено при помощи оператора switch;
   6. Полученный результат кладем в стек и повторяем все те же операции циклично, пока не закончится строка;
   7. В конце извлекаем из стека полученный итоговый результат и возвращаем его. Это и будет наш ответ.

# **Заключение**

В ходе лабораторной работы была разработана программа, реализующая вычисление арифметического выражения с вещественными числами. Выражение в качестве операндов может содержать переменные и вещественные числа. Допустимые операции: +, -, /, \*. Допускается наличие знака "-" в начале выражения или после открывающей скобки. Результаты работы программы представлены ниже.





Полученные результаты полностью удовлетворяют поставленным учебно-практическим задачам.

# **Список литературы**

1. *Павловская Т.А.* C/C++ Программирование на языке высокого уровня. – СПб.: Питер, 2003. – 461 с.: ил.
2. *Страуструп Б.* Язык программирования C++: Пер. с англ. – 3-е спец. изд. – М.: Бином, 2003. – 1104 с.

# **Приложение**

# ***Приложение 1***

Файл tlist.h – файл, содержащий интерфейс и реализацию структур данных: односвязный список и стек:

#ifndef \_\_LIST\_STRUCT

#define \_\_LIST\_STRUCT

#include <conio.h>

#include <iostream>

using namespace std;

template <class ValType>

struct List\_node {

ValType data;

List\_node \*next;

};

template <class ValType>

class List {

List\_node<ValType> \*first;

int size\_list;

public:

List(); //конструктор без параметров

List (int); //конструктор с задан. числом звеньев

~List(); //деструктор

int GetSize() { return size\_list; } //получение размера

List\_node<ValType>\* Search (ValType); //поиск звена с задан. знач.

ValType& operator[](int); //перегрузка индексации

bool Empty\_list(); //проверка списка на пустоту

void Add(ValType ,List\_node<ValType> \*); //вставка с зад. позиц.

void AddWithBegin(ValType); //вставка в начало

void AddWithEnd(ValType); //вставка в конец

void DeleteNodeWithPos(List\_node<ValType> \*);

ValType DeleteNodeWithBegin();

void DeleteNodeWithEnd();

friend ostream & operator << (ostream &out, const List &p);

friend istream & operator >> (istream &in, List &p);

};/\*---------------------------------------------------------------------\*/

template <class ValType>

ostream & operator << (ostream &out, const List<ValType> &p)

{

List\_node <ValType>\* list = p.first;

while (list!=0)

{

out << p->data << ' ';

list=list->next;

}

out<<endl;

return out;

};/\*---------------------------------------------------------------------\*/

template <class ValType>

istream & operator >> (istream &in, List<ValType> &p)

{

in >> p.size\_list;

ValType temp;

for (int i = 0; i < p.size\_list; ++i)

{

in >> temp;

p.AddWithEnd(temp);

}

return in;

};/\*---------------------------------------------------------------------\*/

template <class ValType>

List<ValType>::List()

{

first=0;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

List<ValType>::List(int s)

{ if (s==0)

{

first=0;

size\_list=0;

}

if (s>=0)

{

size\_list=s;

first=new List\_node<ValType>;

List\_node<ValType> \*temp=first;

for (int i=1; i<size\_list; ++i)

{

temp->next=new List\_node<ValType>;

temp=temp->next;

}

temp->next=0;

}

else

throw 1;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

List<ValType>::~List()

{ if (first!=0)

{

List\_node<ValType>\* node=first;

List\_node<ValType> \*list=first->next;

while (list!=0)

{

delete node;

node=list;

list=list->next;

}

delete node;

first=0;

}

}/\*---------------------------------------------------------------------\*/

template <class ValType>

bool List<ValType>::Empty\_list()

{

if (first!=0)

return 0;

else

return 1;

}/\*---------------------------------------------------------------------\*/

template<class ValType>

ValType& List<ValType>::operator[](int ind)

{

if ((ind>=0) && (ind<size\_list))

{

List\_node<ValType>\* list=first;

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

list = list->next;

return list->data;

}

else

throw 1;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

void List<ValType>::Add(ValType data\_node,List\_node<ValType> \*p)

{

if (p)

{

List\_node<ValType>\* node=new List\_node<ValType>;

node->data=data\_node;

node->next=p->next;

p->next=node;

++size\_list;

}

else

throw 1;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

void List<ValType>::AddWithBegin(ValType data\_node)

{

List\_node<ValType>\*node=new List\_node<ValType>;

node->data=data\_node;

node->next=first;

first=node;

++size\_list;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

void List<ValType>::AddWithEnd(ValType data\_node)

{

List\_node<ValType>\*node=new List\_node<ValType>;

node->data=data\_node;

node->next=0;

List\_node<ValType>\* list=first;

while ((list->next)!=0)

list=list->next;

list->next=node;

++size\_list;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

List\_node<ValType>\* List<ValType>::Search(ValType data\_node)

{

if (first!=0)

{

List\_node<ValType>\* list=first;

while ((list->data!=data\_node) && (list!=0))

list=list->next;

return list;

}

else

return 0;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

void List<ValType>::DeleteNodeWithPos(List\_node<ValType> \*p)

{

if ((p) && (first!=0))

{

List\_node<ValType>\* node=p->next->next;

delete p->next;

(p->next)=node;

--size\_list;

}

else

throw 1;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

ValType List<ValType>::DeleteNodeWithBegin()

{

if (first!=0)

{

List\_node<ValType>\* node=first->next;

ValType Res=first->data;

delete first;

first=node;

--size\_list;

return Res;

}

else

throw 1;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

void List<ValType>::DeleteNodeWithEnd()

{

if (first!=0)

{

List\_node<ValType>\* list=first;

while ((list->next->next)!=0)

list=list->next;

delete list->next;

(list->next)=0;

--size\_list;

}

else

throw 1;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

class Stack {

List <ValType> Mem;

public:

Stack(); //конструктор без пар-ров

bool EmptyStack(); //проверка на пустоту

void PushElem(const ValType); //положить в стек

ValType DelElem(); //вытащить из стека

ValType RetTopElem(); //вернуть верхнее значение

void ClearStack(); //очистка стека

};

template <class ValType>

Stack <ValType>::Stack()

{

}/\*---------------------------------------------------------------------\*/

template <class ValType>

bool Stack<ValType>::EmptyStack()

{

return Mem.Empty\_list();

}/\*---------------------------------------------------------------------\*/

template <class ValType>

void Stack <ValType>::PushElem(const ValType elem)

{

Mem.AddWithBegin(elem);

}/\*---------------------------------------------------------------------\*/

template <class ValType>

ValType Stack <ValType>::DelElem()

{

return Mem.DeleteNodeWithBegin();

}/\*---------------------------------------------------------------------\*/

template <class ValType>

ValType Stack <ValType>::RetTopElem()

{

ValType temp=Mem.DeleteNodeWithBegin();

Mem.AddWithBegin(temp);

return temp;

}/\*---------------------------------------------------------------------\*/

template <class ValType>

void Stack <ValType>::ClearStack()

{

delete Mem;

}/\*---------------------------------------------------------------------\*/

#endif

# ***Приложение 2***

Файл arithmetic.h – файл, содержащий набор функций, необходимых для корректной работы арифметических выражений:

#ifndef \_\_ARITHMETIC\_STRUCT

#define \_\_ARITHMETIC\_STRUCT

#include <iostream>

#include <string>

#include "cstdlib"

#include "tlist.h"

using namespace std;

int Priority(char); //done приоритет

bool Brackets(char \*); //done контроль скобок

void UnarMinus(char \*,char \*); //done запись унарного минуса

bool Operation(char ); //done определение операций

int TypeChar(char ); //done разбиение на лекс.

void ConvertInPostfix (const char\* ,char \*); //done обр. польская запись

void PointToComma(char \*); //done перев. точки в запятую

bool FindUnarMinus(char \*); //done определение унар. минуса

bool CheckOperationsInBegEndWithBrack(char\* ); //done проверка кор. операций в начале конце выражения и после откр. и перед закрыв. скобками

bool CheckOperationsInSuccession(char \*); //done проверка кор. операций на запись их подряд

bool CheckCorrectSymbols(char \*); //done проверка кор. записи символов

bool CheckPositionOfDotAndComma(char \*); //done кор. расстановки точек и запятых

bool CheckVariable(char \*); //done кор. наличия переменных

bool CheckManyVariablesInSuccession(char \*); //done кор. записи нескольких переменных подряд

bool CheckStrAtallChecks(char \*); //done все проверки корректности вместе взятые

void FindWritePositVariable (const char \*, int \* ); //done поиск переменных в выражении

bool CorrectOfNumb(char \*); //done кор. ввода числа

double Numb(char \*); //done перевод строки в число (atof)

void InputValuable(char \*,char \*); //done входные параметры переменных

double ResultsCount(char \*); //done подсчет выражения

#endif

# ***Приложение 3***

Файл arithmetic.cpp – файл, содержащий реализацию функций, необходимых для корректной работы арифметических выражений:

#include "tlist.h"

#include "arithmetic.h"

int Priority(char s) //done приоритет

{

int prior;

switch (s) //определение приоритета операций

{

case '(':

{ prior=0; break;}

case ')':

{ prior=0; break;}

case '+':

{ prior=1; break;}

case '-':

{ prior=1; break;}

case '\*':

{ prior=2; break;}

case '/':

{ prior=2; break;}

}

return prior;

}/\*---------------------------------------------------------------------\*/

bool Operation(char s) //done определение операций

{

bool f=false;

char signum[]="-+\*/";

for(int i=0;i<6;i++)

if (s==signum[i]) //данный символ является ли знаком операции?

{

f=true;

break;

}

return f;

}/\*---------------------------------------------------------------------\*/

int TypeChar(char s) //done разбиение на лекс.

{

if ( Operation(s) ) //операция

return 1;

else if(isdigit(s)) //число

return 2;

else if((s=='(')||(s==')')) //скобка

return 3;

else if((s=='.')||(s==',')) //тчк. или зап.

return 4;

else if(isalpha(s)) //буква (параметр)

return 5;

else

return 0; //остальное

}/\*---------------------------------------------------------------------\*/

bool Brackets(char \*s) //done контроль скобок

{

Stack<char> Stack\_1;

int i=0;

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

{

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

Stack\_1.PushElem('(');

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

{

if (Stack\_1.EmptyStack())

{

return false; //т.е. в стеке не оказ. откр. скобки

}

else

Stack\_1.DelElem();

}

i++;

}

if (Stack\_1.EmptyStack()) //в конце концов стек освободился

return true;

else

{

return false;

}

}/\*---------------------------------------------------------------------\*/

void UnarMinus(char \*s, char \*res) //done запись унарного минуса

{

int len = strlen(s);

int j=0;

if (s[0]=='-') //первый символ оказался -

{

res[j++]='0';

res[j++]='-';

}

else

res[j++]=s[0];

for (int i=1;i<len;i++)

{

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

{

if((s[i-1]=='(')&&((TypeChar(s[i+1])==2)||(TypeChar(s[i+1])==5))) //проверка является ли унарным минус (после откр. скобкой перед симв.)

{

res[j++]='0';

res[j++]='-';

}

else

res[j++]='-'; //иначе бинарный

}

else

{

res[j]=s[i];

j++;

}

}

res[j]='\0';

}/\*---------------------------------------------------------------------\*/

void ConvertInPostfix(const char\* s,char \*res) //done обр. польская запись

{

Stack<char> Stack\_2;

int len = strlen(s);

int j=0;

int m=0;

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

{

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

Stack\_2.PushElem('('); //кладем если скобка

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

{

char a =Stack\_2.DelElem(); //если закрывается удаляем эл-т из ст.

while (a !='(') //пока не дошли до откр. скобки заносим в массив поэлементно символы

{

res[j]=a;

j++;

a=Stack\_2.DelElem();

}

}

if(Operation(s[i])) //если попалась операция

{

if (Stack\_2.EmptyStack())

Stack\_2.PushElem(s[i]); //кладем ее в пустой ст.

else

{

char oper = Stack\_2.RetTopElem();

while(Priority(s[i])<=Priority(oper)) //если не пустой смотрим на приоритет операций

{

res[j]=Stack\_2.DelElem(); //более "сильную" операцию в ответ

j++;

if (Stack\_2.EmptyStack()!=true)

oper=Stack\_2.RetTopElem();

else

oper='(';

}

Stack\_2.PushElem(s[i]); //в итоге клад операц.

}

}

if (TypeChar(s[i])==2) //число

{

if ((TypeChar(s[i+1])==2)||(TypeChar(s[i+1])==4))

{

res[j]=s[i]; //если число целое или с дес. частью запишем в ответ

j++;

}

else

{

res[j]=s[i]; //в ином сл. если дальше уже не число, то сделаем пробел

j++;

res[j]=' ';

j++;

}

}

if (TypeChar(s[i])==5) //параметр

{

res[j]=s[i]; //занесли + пробел

j++;

res[j]=' ';

j++;

}

if (TypeChar(s[i])==4) //точку или зап. просто заносим

{

res[j]=s[i];

j++;

}

}

while(Stack\_2.EmptyStack()!=true) //в конце если стек не пустой выгружаем из него всё оставшееся

{

res[j]=Stack\_2.DelElem();

j++;

}

res[j]='\0';

}/\*---------------------------------------------------------------------\*/

void PointToComma(char \*s) //done перев. точки в запятую

{

int len = strlen(s);

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

{

if (TypeChar(s[i])==4) //перевод из . в ,

{

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

s[i]=',';

}

}

}/\*---------------------------------------------------------------------\*/

bool FindUnarMinus(char \*s) //done определение унар. минуса

{

int i=1;

int f=0;

int len = strlen(s);

if (s[0]=='-') //попался унарный минус в самом начале выражения

f=1;

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

{

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

{

if((s[i-1]=='(')&&((TypeChar(s[i+1])==2)||(TypeChar(s[i+1])==5))) //ловим унарный минус после откр. скобки

{

f=1;

break;

}

}

i++;

}

if (f==1) //попался

return true;

else //не попался

return false;

}/\*---------------------------------------------------------------------\*/

bool CheckOperationsInBegEndWithBrack(char\* s) //done проверка кор. операций в начале конце выражения и после откр. и перед закрыв. скобками

{

int len=strlen(s);

for (int i=1;i<len-1;i++)

{

if(Operation(s[i]))

if((s[i-1]=='(')||(s[i+1]==')'))

{

cout<< " операция после открыв. скобки (не унарный -) или перед закрыв. " << i+1 << endl;

return false;

}

}

if (Operation(s[0]))

{

cout<< " операция в начале выражения (не унарный -) " << endl;

return false;

}

else if (Operation(s[len-1]))

{

cout<< " операция в конце выражения " << len << endl;

return false;

}

return true;

}/\*---------------------------------------------------------------------\*/

bool CheckOperationsInSuccession(char \*s) //done проверка кор. операций на запись их подряд

{

int len=strlen(s);

for (int i=0;i<len-1;i++)

{

if (Operation(s[i]))

{

if (Operation(s[i+1]))

{

cout<< " 2 операции подряд " << i+1 << " и " << i+2 << endl;

return false;

}

}

}

return true;

}/\*---------------------------------------------------------------------\*/

bool CheckCorrectSymbols(char \*s) //done проверка кор. записи символов

{

int len = strlen(s);

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

if(TypeChar(s[i])==0)

{

cout << " Посторонний символ " << i+1 <<endl;

return false;

}

return true;

}/\*---------------------------------------------------------------------\*/

bool CheckPositionOfDotAndComma(char \*s) //done кор. расстановки точек и запятых

{

int len = strlen(s);

if ((TypeChar(s[0])==4)||(TypeChar(s[len-1])==4))

{

cout << " Точка или запятая в начале или конце выражения " <<endl;

return false;

}

else

{

for (int i=1;i<len-1;i++)

if(TypeChar(s[i])==4)

{

if((TypeChar(s[i-1])!=2)||(TypeChar(s[i+1])!=2))

{

cout << " Точка или запятая используется не с числом " <<endl;

return false;

}

}

}

return true;

}/\*---------------------------------------------------------------------\*/

bool CheckVariable(char \*s) //done кор. наличия переменных

{

int len = strlen(s);

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

{

if(TypeChar(s[i])==5) //проверяем налич. переменных

return true;

}

return false;

}/\*---------------------------------------------------------------------\*/

bool CheckManyVariablesInSuccession(char \*s) //done кор. записи нескольких переменных подряд

{

int len = strlen(s);

for (int i=0;i<len-1;i++)

if((TypeChar(s[i])==5)&&(TypeChar(s[i+1])==5))

{

cout << " несколько перемменных подряд " <<endl;

return false;

}

return true;

}/\*---------------------------------------------------------------------\*/

bool CheckStrAtallChecks(char\* s) //done все проверки корректности вместе взятые

{

if((Brackets(s))&&(CheckOperationsInBegEndWithBrack(s))&&(CheckOperationsInSuccession(s))&&(CheckCorrectSymbols(s))&&(CheckPositionOfDotAndComma(s))&&(CheckManyVariablesInSuccession(s)))

return true;

else

return false;

}/\*---------------------------------------------------------------------\*/

void FindWritePositVariable (const char \*s, int \* res) //done поиск переменных в выражении

{

int j=0;

for (int i=0;i<strlen(s);i++) //поиск и запись переменных в выражениии (запись в массив)

if(TypeChar(s[i])==5)

{

res[j]=i;

j++;

}

}/\*---------------------------------------------------------------------\*/

bool CorrectOfNumb(char \*s) //done кор. ввода числа

{

int len = strlen(s);

if (CheckPositionOfDotAndComma(s)!=true) //нарушен ввод точек и запятых

return false;

if ((Operation(s[0]))&&(s[0]!='-')) //нарушение: ввод бинарных операций

return false;

for (int i=0;i<len;i++) //нарушение: ввод постронних эл-тов

{

if((TypeChar(s[i])==0)||(TypeChar(s[i])==5)||(TypeChar(s[i])==3))

{

return false;

}

}

return true;

}/\*---------------------------------------------------------------------\*/

double Numb(char \*s) //done перевод строки в число (atof)

{

double result=atof(s);

return result;

}/\*---------------------------------------------------------------------\*/

void InputValuable(char \*s, char \*res) //done входные параметры переменных

{

int \*numb;

int Size\_numb=1024;

int j=0;

int k=0;

int len=strlen(s);

numb=new int[Size\_numb];

for (int i=0;i<Size\_numb;i++) //создание спец. мас.

numb[i]=-1;

FindWritePositVariable(s,numb);

if (numb[0]!=-1) //проверяем, есть ли вообще параметры в нашей задаче

cout << " В выражении присутствуют параметры / This string has variables " <<endl;

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

{

if (numb[j]!=t) //позиция пар-ра не совпала с текущей, зн. просто число

{

res[k]=s[t];

k++;

}

else //параметр

{

int f=1;

char str[1024];

int i=0;

while (f==1)

{

cout << s[numb[j]]<< "=";

cin>>str; //ввод значения параметра

if (CorrectOfNumb(str)!=true) //проверка корректности

{

cout << " Некорректный ввод параметра! / Incorrect input! " <<endl;

}

else

{

if(str[0]=='-') //унарный минус

{

res[k]='0';

k++;

res[k]=' ';

k++;

i=1;

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

{

res[k]=str[i];

k++;

i++;

}

res[k]=' ';

k++;

res[k]='-';

k++;

t++;

}

else //без унарного минуса

{

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

{

res[k]=str[i];

k++;

i++;

}

}

f=0;

}

}

j++;

}

}

res[k]='\0';

delete []numb;

}/\*---------------------------------------------------------------------\*/

double ResultsCount(char \*s) //done подсчет выражения

{

Stack<double> Stack\_1;

int len=strlen(s);

int p=0;

int t=0;

int j=0;

double res;

while(s[j]!='\0') //пошли по строке

{

if(TypeChar(s[j])==2) //если число

{

char str[1024];

int k=0;

double numb;

t=j;

while(s[t]!=' ') //занесли число в строку

{

str[k]=s[t];

t++;

k++;

}

str[k]='\0';

p=t-j+1; //переход (на сколько)

numb=Numb(str); //переводим стр. в число

Stack\_1.PushElem(numb); //клад в стек

j=j+p-1; //сам переход

}

if (Operation(s[j])) //попалась операция

{

double r,l; //2 верхних эл-та стека

r=Stack\_1.DelElem();

l=Stack\_1.DelElem();

switch(s[j]) //выполняем операцию

{

case '+':

{

res=l+r;

break;

}

case '-':

{

res=l-r;

break;

}

case '\*':

{

res=l\*r;

break;

}

case '/':

{

res=l/r;

break;

}

}

Stack\_1.PushElem(res); //положили рез-тат в стек

}

j++;

}

return Stack\_1.DelElem();

}/\*---------------------------------------------------------------------\*/

# ***Приложение 4***

Файл Sample.cpp – файл, содержащий реализацию консольного интерфейса пользователя:

#include "arithmetic.h"

#include "tlist.h"

void main(void)

{

setlocale(LC\_ALL, "Russian");

char String[1024];

int Size=1024;

char k;

cout<<"Меню:"<<endl;

cout<<"1.выражение без параметра"<<endl;

cout<<"2.выражение с параметром"<<endl;

cout<<"3.выход"<<endl;

cin>>k;

bool temp=true;

while (temp==true)

{

switch (k)

{

case '1':

{

system("cls");

cout<<"Введите выражение:"<<endl;

cin>>String;

char \* working\_str;

working\_str = new char [Size];

if(FindUnarMinus(String))

{

UnarMinus(String,working\_str);

}

else

{

int i=0;

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

{

working\_str[i]=String[i];

i++;

}

working\_str[i]='\0';

}

if (CheckStrAtallChecks(working\_str)==false)

cout<<"Исходное выражение введено некорректно"<<endl;

else

{

double res;

char \* working\_str\_postf\_1 ;

char \* working\_str\_postf\_2 ;

working\_str\_postf\_1 = new char [Size];

working\_str\_postf\_2 = new char [Size];

ConvertInPostfix(working\_str,working\_str\_postf\_1);

res=ResultsCount(working\_str\_postf\_1);

cout << "ответ = " << res <<endl;

\_getch();

delete []working\_str\_postf\_1;

delete []working\_str\_postf\_2;

system("cls");

cout<<"Меню:"<<endl;

cout<<"1.выражение без параметра"<<endl;

cout<<"2.выражение с параметром"<<endl;

cout<<"3.выход"<<endl;

cin>>k;

}

break;

}

case '2':

{

system("cls");

cout<<"Введите выражение:"<<endl;

cin>>String;

char \* working\_str;

working\_str = new char [Size];

if(FindUnarMinus(String))

{

UnarMinus(String,working\_str);

}

else

{

int i=0;

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

{

working\_str[i]=String[i];

i++;

}

working\_str[i]='\0';

}

if (CheckStrAtallChecks(working\_str)==false)

cout<<"Исходное выражение введено некорректно"<<endl;

else

{

double res;

char \* working\_str\_postf\_1 ;

char \* working\_str\_postf\_2 ;

working\_str\_postf\_1 = new char [Size];

working\_str\_postf\_2 = new char [Size];

ConvertInPostfix(working\_str,working\_str\_postf\_1);

InputValuable(working\_str\_postf\_1,working\_str\_postf\_2);

PointToComma(working\_str\_postf\_2);

res=ResultsCount(working\_str\_postf\_2);

cout << "ответ = " << res <<endl;

\_getch();

delete []working\_str\_postf\_1;

delete []working\_str\_postf\_2;

system("cls");

cout<<"Меню:"<<endl;

cout<<"1.выражение без параметра"<<endl;

cout<<"2.выражение с параметром"<<endl;

cout<<"3.выход"<<endl;

cin>>k;

}

break;

}

case '3':

{

system("cls");

exit(0);

}

default:

{

system("cls");

cout<<"Нет такого пункта..."<<endl;

\_getch();

system("cls");

cout<<"Меню:"<<endl;

cout<<"1.выражение без параметра"<<endl;

cout<<"2.выражение с параметром"<<endl;

cout<<"3.выход"<<endl;

cin>>k;

}

break;

}

}

\_getch();

}