Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Нижегородский государственный университет

им. Н.И. Лобачевского»

**Институт информационных технологий математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Название работы**

**Транслятор арифметических выражений**

**Выполнил**:студент группы 0823-1

Косоруков Дмитрий Олегович

Подпись

**Научный руководитель**:

Подпись

Нижний Новгород

2016

Оглавление

[Оглавление 2](#_Toc451370556)

[**Введение** 3](#_Toc451370557)

[**Постановка задачи** 4](#_Toc451370558)

[**Исходные данные** 4](#_Toc451370559)

[**Результат** 4](#_Toc451370560)

[**Руководство пользователя** 5](#_Toc451370561)

[**Руководство программиста** 6](#_Toc451370562)

[**Заключение** 8](#_Toc451370564)

[**Список литературы** 8](#_Toc451370565)

[**Приложение** 9](#_Toc451370566)

**Введение**

Для вычисления значения выражения в определённых точках необходимо преобразовать строку в понятной машине структуру, которая будет вычислена.

Наиболее простым способом вычисления выражения является вычисление по польской записи. Следовательно, основной задачей является разбор на лексемы и преобразование в польскую запись.

**Постановка задачи**

Требуется разработать транслятор арифметических выражений

**Исходные данные**

На вход программе поступает строка задающее арифметическое выражение

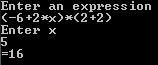
G:\Лобач\АиСД\3 Горячкина неболшой реврайт\1.jpg

Программа запрашивает значение переменной.

G:\Лобач\АиСД\3 Горячкина неболшой реврайт\2.jpg

**Результат**

После вычисления выражения выводит его результат.



**Руководство пользователя**

Для выполнения работы необходимо запустить файл Arihm.exe. Далее ввести строку представляющую нужное выражение. Эта трока может содержать целые числа, операции +, -, \*, /, скобки и имена переменных состоящих из прописной буквы латинского алфавита (a-z). Если в выражении присутствует хотя бы одна переменная, то затем нужно будет ввести значение этой(-их) переменной(-ых). После этого на экране будет выведен ответ. В случае некорректности выражения возникнет ошибка.

**Руководство программиста**

В программе реализован класс арифметических выражений со следующим интерфейсом.

class Arithm

{

protected:

string Str;

TList<Lexeme> Lex, Pol;

Variable var[27];

public:

Arithm(); // конструктор

~Arithm(); // деструктор

void Enter(string inps); // ввод строки

bool Translate(); // расчленение на лексемы

double Compute(); // вычисление выражение

};

Поля класса:

Str – строка с арифметическим выражением

Lex – прямая очередь лексем

Pol – очередь лексем в польской записи

var – массив структур соответствующих переменным

Для того чтобы разбить строку на лексемы необходима структура Lexseme:

struct Lexeme

{

char type; // тип лексеммы

double data; // значение, если число

};

type - определяет :

'('=1 , ')'=2 , '+'=3 , '-'=4 , '\*'=5 , '/'=6

число=0 , переменная='имя переменной', где a<'имя переменной'<z

data – значение, если лексема является числом

Для присутствующих переменных необходим массив структур var:

variable var[27];

struct Variable

{

bool name; // флаг есть ли переменная с именем var[i]

double data; // для ввода ее значения

};

Arithm();

В конструкторе происходит инициализация массива переменных. По умолчанию переменных нет.

void Enter(string inps)

Функция ввода копирует введенную строку inps в поле класса Str

bool Translate();

Осуществляет проверку выражения на корректность и разбор на лексемы, возвращает true если выражение корректно и false в противном случае. В процессе разбора заполняет очередь Lex

double Compute();

Функция осуществляет перевод лексем в польскую запись, в результате заполняется очередь Pol, вычисляет результат и возвращает его.

**Заключение**

В рамках лабораторной работы разработан транслятор арифметических выражений, в котором реализована возможность вводы целых чисел, переменных и операций +, -,\*, /.

**Список литературы**

* Тетрадь

[https://basegroup.ru/community/glossary/mult-vector](https://ru.wikipedia.org/wiki/Ряд_Тейлора)

**Приложение**

#ifndef \_\_ARITHM\_H\_\_

#define \_\_ARITHM\_H\_\_

#include "tlist.h"

#include <iostream>

#include <string>

using namespace std;

// type:

// '('=1 , ')'=2 , '+'=3 , '-'=4 , '\*'=5 , '/'=6

// число=0 , переменная='имя переменной', где a<'имя переменной'<z

struct Lexeme

{

char type; // тип лексеммы

double data; // значение, если число

};

struct Variable

{

bool name; // флаг есть ли переменная с именем var[i]

double data; // для ввода ее значения

};

class Arithm

{

protected:

string Str;

TList<Lexeme> Lex, Pol;

Variable var[27];

public:

Arithm();

~Arithm();

void Enter(string strin); // ввод строки

bool Translate(); // расчленение на лексемы

double Compute(); // вычисление выражение

};

Arithm::Arithm()

{

for(int i=0;i<27;i++) var[i].name=false;

}

Arithm::~Arithm() {}

void Arithm::Enter(string inps)

{

Str=inps;

}

bool Arithm::Translate()

{

int i=0; // индекс считываемого символа

int state=0; // состояние ожидания или ошибка

int k=0; // кол-во левых скобок - кол-во правых скобок

Lexeme num;

char c=Str[0];

while((i<Str.size())&&(state!=2))

{

if(state==0) // ожидание откр скобки или чтение числа

{

if(c=='(')

{

num.type=1;

Lex.Put(num);

k++;

if(i<Str.size()-1) c=Str[++i]; // считывание следующего элемента

}

else

if(c=='-') // чтение отрицательного числа

{

i++;

if(i<Str.size()) c=Str[i];

if((c>='0')&&(c<='9'))

{

num.type=0;

num.data=-(c-48);

i++;

while (i<Str.size())

{

c=Str[i];

if((Str[i]>='0')&&(Str[i]<='9')) // наматываем число

{

num.data\*=10;

num.data+=c-48;

i++;

}

else break;

}

Lex.Put(num);

state=1;

}

else

if(c=='(')

{

num.type=0;

num.data=0;

Lex.Put(num);

num.type=4;

Lex.Put(num);

}

else state=2;

} // посчитали отрицательное число

else

if(c=='+') // считаем положительное число с +

{

i++;

if(i<Str.size()) c=Str[i];

if((c>='0')&&(c<='9'))

{

num.type=0;

num.data=c-48;

i++;

while (i<Str.size())

{

c=Str[i];

if((Str[i]>='0')&&(Str[i]<='9')) // наматываем число

{

num.data\*=10;

num.data+=c-48;

i++;

}

else break;

}

Lex.Put(num);

state=1;

}

else state=2;

} // посчитали положительное число с +

else

if((c>='0')&&(c<='9')) // считаем положительное число

{

num.type=0;

num.data=c-48;

i++;

while(i<Str.size())

{

c=Str[i];

if((Str[i]>='0')&&(Str[i]<='9')) // наматываем число

{

num.data\*=10;

num.data+=c-48;

i++;

}

else break;

}

Lex.Put(num);

state=1;

} // посчитали положительное число

else

if((c>='a')||(c<='z')) // записываем переменную

{

num.type=c;

Lex.Put(num);

i++;

if(i<Str.size()) c=Str[i];

state=1;

}

else state=2;

}

else

if(state==1) // ожидание закр скобки или операции

{

if(c==')')

{

if(k>0)

{

k--;

num.type=2;

Lex.Put(num);

i++;

if(i<Str.size()) c=Str[i];

}

else state=2;

}

else

if((c=='+')||(c=='-')||(c=='\*')||(c=='/'))

{

switch(c){

case '+':

num.type=3; break;

case '-':

num.type=4; break;

case '\*':

num.type=5; break;

case '/':

num.type=6; break;

}

state=0;

i++;

Lex.Put(num);

if(i<Str.size()) c=Str[i];

}

else state=2;

}

}

return (k==0 && state!=2); // возвращаем нет ли ошибок

}

double Arithm::Compute()

{

Lexeme temp;

if(Translate())

{

TStack<Lexeme> op;

while(!Lex.IsEmpty())

{

temp=Lex.Get();

if(temp.type==0) Pol.Put(temp); // если число

else

if((temp.type>='a')&&(temp.type<='z')) // если переменная

{

var[temp.type-97].name=true;

Pol.Put(temp);

}

else

if(temp.type==1) op.Push(temp); // если открывающаяся скобка

else

if(temp.type==2) // если закрывающаяся скобка

{

while(op.Top().type!=1) Pol.Put(op.Pop());

op.Pop();

}

else // если операция

{

while(!op.IsEmpty() &&(op.Top().type>=temp.type))

{

Pol.Put(op.Pop());

}

op.Push(temp);

}

}

while(!op.IsEmpty()) // извлечение операций

{

Pol.Put(op.Pop());

}

}

else throw "not correct";

for(int i=0;i<27;i++) // инициализируем переменные

if(var[i].name)

{

char c='a'+i;

cout<<"Enter "<<c<<'\n';

cin>>var[i].data;

}

TStack<double> exe;

while(!Pol.IsEmpty()) // вычисление

{

double a,b;

temp=Pol.Get();

if(temp.type==0) exe.Push(temp.data); // если число

else

if((temp.type>='a')&&(temp.type<='z')) exe.Push(var[temp.type-97].data); // если переменная

else // если операция

{

b=exe.Pop();

a=exe.Pop();

switch(temp.type)

{

case 3:

{

exe.Push(a+b);

}

break;

case 4:

{

exe.Push(a-b);

}

break;

case 5:

{

exe.Push(a\*b);

}

break;

case 6:

{

exe.Push(a/b);

}

break;

}

}

}

return exe.Pop();

}

double ArithmCompute(string str)

{

Arithm A;

A.Enter(str);

return A.Compute();

}

#endif