**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра** **ВТ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: «Калькулятор»**

**Вариант 12**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0322 |  | Шаронин А.Д. |
| Преподаватель |  | Пестерев Д. О. |

Санкт-Петербург

2022

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Шаронин Алексей Денисович | | |
| Группа 0322 | | |
| Тема работы: Калькулятор. | | |
| Исходные данные: Реализовать вычисление арифметических выражений с бинарными операциями {+, -, \*, /, ^} и числовыми операндами, записанных в префиксной/инфиксной/постфиксной записи. | | |
| Содержание пояснительной записки: «Содержание», «Описание программы», «Оценка сложности каждого метода», «Примеры работы программы», «Заключение», « Листинг программного кода», «Список использованных источников». | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц | | |
| **Дата выдачи задания: 15.11.2022** | | |
| Дата сдачи курсовой работы: 05.12.2022 | | |
| Дата защиты курсовой работы: 12.12.2022 | | |
| Студент |  | Шаронин А.Д. |
| Преподаватель |  | Пестерев Д. О. |

**Содержание**

Оглавление

[Описание программы 4](#_Toc121252397)

[Оценка сложности каждой функции 5](#_Toc121252398)

[Примеры работы программы 6](#_Toc121252399)

[Заключение](#_Toc121252400) 8

[Приложение 1. Листинг программного кода 9](#_Toc121252401)

[Приложение 2. Список использованных источников.](#_Toc121252402) 14

# Описание программы

Данная программа предлагает пользователю выбор записи, в которой потом вводится выражение: префиксная/инфиксная/постфиксная. Далее программа переводит, при необходимости запись в постфиксную и вычисляет результат. Результат выводится на экран.

Функции, реализованные в программе:

* is\_operation — проверяет, является ли символ операцией (+, -, \*, /, ^)
* inf\_to\_post — переводит запись из инфиксной в постфиксную
* pre\_to\_post — переводит запись из префиксной в постфиксную
* post\_to\_post — переводит запись из постфиксной в постфиксную (возвращает принимаемое значение)
* calc — производит операцию над двумя операндами и возвращает значение
* count\_post — вычисляет значение постфиксной записи

# Оценка сложности каждого метода.

* is\_operation — O(1)
* inf\_to\_post — O(n)
* pre\_to\_post — O(n)
* post\_to\_post — O(1)
* calc — O(1)
* count\_post — O(n)

n — длина вводимого выражения

# Примеры работы программы

Мы будем проверять работу программы на примере выражения (((9,8\*65,4)/5) / (7^ (1,5-2))). В качестве проверки возьмём результат посчитанный в гугле (Рисунок 1).

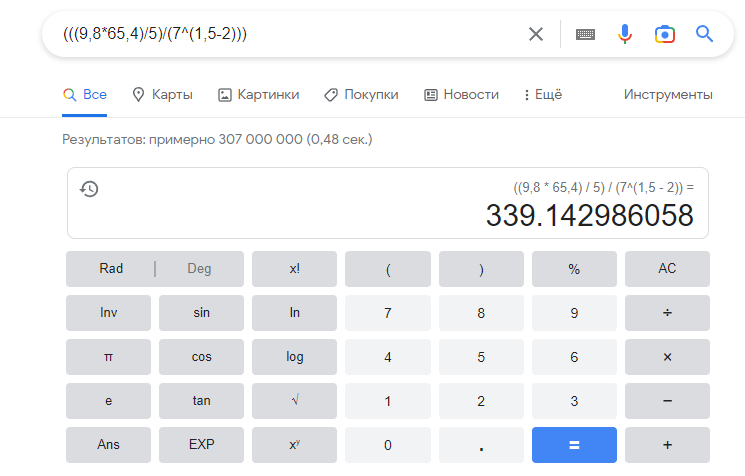
****

Рисунок —Контрольное значение

Пример 1.

Входящее выражение является префиксным и выглядит как:

/ / \* 9,8 65,4 5 ^ 7 - 1,5 2

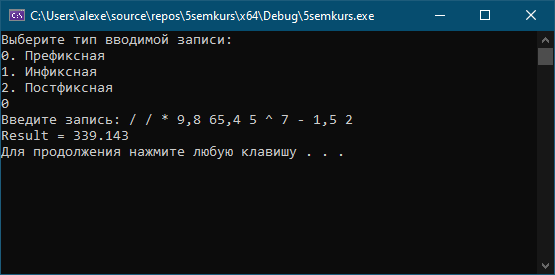


Рисунок —префиксная запись

Пример 2.

Входящее выражение является инфиксным и выглядит как:

(((9,8\*65,4)/5)/ (7^ (1,5-2)))

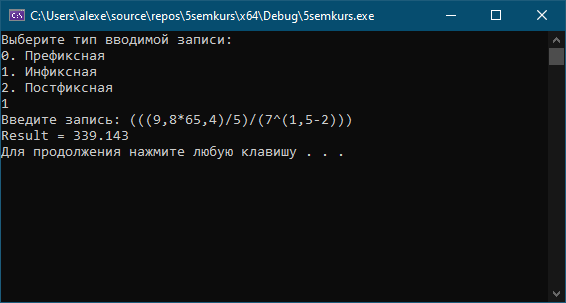


Рисунок — инфиксная запись

Пример 3.

Входящее выражение является постфиксным и выглядит как:

9,8 65,4 \* 5 / 7 1,5 2 - ^ /

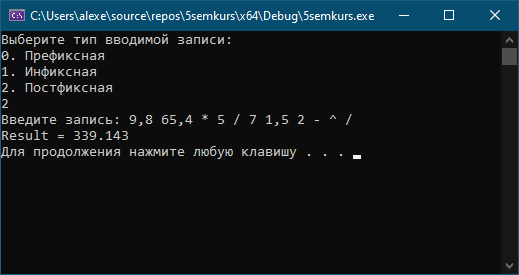


Рисунок — постфиксная запись

# Заключение

В ходе выполнения курсовой работы был сделан калькулятор для префиксной/инфиксной/постфиксной нотации. Так как все три раза вычисления сошлись с контрольным, то можно считать, что калькулятор сделан правильно. Также были получены навыки для подсчёта значения выражения в постфиксной нотации.**Приложение 1. Листинг программного кода**

Часть main

#include "Header.h"

int main()

{

//post 9,8 65,4 \* 5 / 7 1,5 2 - ^ /

//inf (((9,8\*65,4)/5)/(7^(1,5-2)))

//pre / / \* 9,8 65,4 5 ^ 7 - 1,5 2

setlocale(0, "");

string str;

cout << "Выберите тип вводимой записи:\n";

cout << "0. Префиксная\n";

cout << "1. Инфиксная\n";

cout << "2. Постфиксная\n";

int type;

cin >> type;

cout << "Введите запись: ";

cin.clear();

cin.ignore(32756, '\n');

getline(cin, str);

switch (type)

{

case 0:

str = pre\_to\_post(str);

break;

case 1:

str = inf\_to\_post(str);

break;

case 2:

str = post\_to\_post(str);

break;

default:

cout << "неверно введён тип записи\n";

break;

}

count\_post(str);

system("pause");

}

Часть Header.h

#include <string>

#include <iostream>

#include <queue>

#include <stack>

using namespace std;

char is\_operation(char c)

{

return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^');

}

string inf\_to\_post(string str)//inf to post

{

string res = "";

stack<char> stck;

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

{

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

{

stck.push(str[i]);

}

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

{

while (!stck.empty() && stck.top() != '(')

{

res.push\_back(stck.top());

res.push\_back(' ');

stck.pop();

}

stck.pop();

}

if (str[i] == '^')

{

while (!stck.empty() && stck.top() == '^')

{

res.push\_back(stck.top());

res.push\_back(' ');

stck.pop();

}

stck.push(str[i]);

}

if (str[i] == '\*' || str[i] == '/')

{

while (!stck.empty() && (stck.top() == '^' || stck.top() == '\*' || stck.top() == '/'))

{

res.push\_back(stck.top());

res.push\_back(' ');

stck.pop();

}

stck.push(str[i]);

}

if (str[i] == '+' || str[i] == '-')

{

stck.push(str[i]);

}

if (isdigit(str[i]) || str[i] == ',')

{

while (isdigit(str[i]) || str[i] == ',')

{

res.push\_back(str[i]);

i++;

}

res.push\_back(' ');

i--;

}

}

while (!stck.empty())

{

res.push\_back(stck.top());

res.push\_back(' ');

stck.pop();

}

return res;

}

string pre\_to\_post(string str)

{

stack<string> s;

stack<char> digit;

// length of expression

int length = str.length();

string temp1;

// reading from right to left

for (int i = length - 1; i >= 0; i--)

{

// check if symbol is operator

if (is\_operation(str[i]))

{

// pop two operands from stack

string op1 = s.top();

s.pop();

string op2 = s.top();

s.pop();

// concat the operands and operator

string temp = op1 + ' ' + op2 + ' ' + str[i];

// Push string temp back to stack

s.push(temp);

}

// if symbol is an operand

else if (isdigit(str[i]) || str[i] == ',')

{

while (isdigit(str[i]) || str[i] == ',')

{

digit.push(str[i]);

i--;

}

while (!digit.empty())

{

temp1 += digit.top();

digit.pop();

}

i++;

// push the operand to the stack

s.push(temp1);

temp1 = "";

}

}

// stack contains only the Postfix expression

return s.top();

}

string post\_to\_post(string str)

{

return str;

}

float calc(float op1, float op2, char oper)

{

switch (oper)

{

case '\*':

return op1 \* op2;

case '/':

return op1 / op2;

case '+':

return op1 + op2;

case '-':

return op1 - op2;

case '^':

return pow(op1,op2);

}

}

void count\_post(string str)

{

stack<float> stck;

string temp;

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

{

if (isdigit(str[i]) || str[i] == ',')

{

while (isdigit(str[i]) || str[i] == ',')

{

temp.push\_back(str[i]);

i++;

}

i--;

stck.push(stof(temp));

}

if (is\_operation(str[i]))

{

float op2 = stck.top();

stck.pop();

float op1 = stck.top();

stck.pop();

float res = calc(op1, op2, str[i]);

stck.push(res);

}

temp = "";

}

cout << "Result = " << stck.top() << endl;

}

# **Приложение 2. Список использованных источников**.

Обратная польская запись [Электронный ресурс]. URL: https://habr.com/ru/post/100869 (дата обращения: 25.12.2022)

Prefix to Postfix Conversion [Электронный ресурс]. URL: https://www.geeksforgeeks.org/prefix-postfix-conversion (дата обращения: 25.12.2022)