ГУО «БГУИР»

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

Отчет по

Лабораторной работе №4

Обратная польская запись

Подготовил:

Студент гр.222401

Рабинчук И.М.

Проверила:

Семижон Е.А.

Минск 2023

Цель: изучить правила формирования постфиксной записи арифметических выражений с использованием стека.

Написать программу формирования ОПЗ и расчета полученного выражения. Разработать удобный интерфейс ввода исходных данных и вывода результатов. Работу программы проверить на конкретном примере

Вариант: 9

Код:

#include <iostream>

using namespace std;

struct Stack

{

char element;

Stack \*next;

};

int Prior(char);

Stack\* InStack(Stack\*, char);

Stack\* OutStack(Stack\*, char&);

double Result(char\*);

void Confus(char[], int);

int main()

{

setlocale(LC\_ALL, "rus");

Stack \*t, \*operations = NULL;

char a, input\_string[81], output\_string[81];

int k=0, l = 0;

cout << "Введите формулу:" << endl;

cin >> input\_string;

Confus(input\_string, k);

for (k = 0; input\_string[k] != NULL; ++k)

{

//Если элемент-буква

if (input\_string[k] >= 'A' and input\_string[k] <= 'z')

output\_string[l++] = input\_string[k];

else

{

//Если элемент-открывающая скобка

if (input\_string[k] == '(')

operations = InStack(operations, input\_string[k]);

else

{

//Если элемент-закрывающая скобка

if (input\_string[k] == ')')

{

while ((operations->element) != '(')

{

operations = OutStack(operations, a);

if (!operations)

a = NULL;

output\_string[l++] = a;

}

operations = OutStack(operations, a);

}

else

{

//Если элемент-знак операции

if (input\_string[k] == '+' or input\_string[k] == '-' or input\_string[k] == '\*' or input\_string[k] == '/')

{

while (operations != NULL and Prior(operations->element) >= Prior(input\_string[k]))

{

operations = OutStack(operations, a);

output\_string[l++] = a;

}

operations = InStack(operations, input\_string[k]);

}

}

}

}

}

//Если операций не пуст, извлекаем все из него в выходную строку

while (operations != NULL)

{

operations = OutStack(operations, a);

output\_string[l++] = a;

}

output\_string[l] = NULL;

cout << "Польская запись: " << output\_string << endl;

cout << "Результат выражения: " << Result(output\_string) << endl;

system("pause");

}

int Prior(char a)

{

switch (a)

{

case'\*': case'/': return 3;

case'+': case'-': return 2;

case'(': return 1;

}

return 0;

}

Stack\* InStack(Stack \*p, char s)

{

Stack \*t = new Stack;

t->element = s;

t->next = p;

return t;

}

Stack\* OutStack(Stack \*p, char &s)

{

Stack \*t = p;

s = p->element;

p = p->next;

delete t;

return p;

}

double Result(char \*str)

{

int i; Stack \*begin = NULL;

char ss, ss\_1, ss\_2, ss\_Result = 'z' + 1;

double operation\_1, operation\_2, result, mas[50];

cout << "Введите переменные" << endl;

for (i = 0; str[i] != NULL; ++i)

{

ss = str[i];

if (ss >= 'A' and ss <= 'z')

{

cout << ss << "=";

cin >> mas[int(ss - 'a')];

}

}

for (i = 0; str[i] != NULL; ++i)

{

ss = str[i];

if (!(ss == '+' or ss == '-' or ss == '\*' or ss == '/'))

begin = InStack(begin, ss);

else

{

begin = OutStack(begin, ss\_2);

begin = OutStack(begin, ss\_1);

operation\_2 = mas[int(ss\_2 - 'a')];

operation\_1 = mas[int(ss\_1 - 'a')];

switch (ss)

{

case'+': result = operation\_1 + operation\_2; break;

case'-': result = operation\_1 - operation\_2; break;

case'\*': result = operation\_1 \* operation\_2; break;

case'/': result = operation\_1 / operation\_2;

if (operation\_2 == 0)

{

cout << "Невозможно выполнить деление на 0." << endl;

return 0;

}

break;

}

mas[int(ss\_Result - 'a')] = result;

begin = InStack(begin, ss\_Result);

ss\_Result++;

}

}

return result;

}

void Confus(char input\_string[], int k)

{

while (input\_string[k] != NULL)

{

if ((input\_string[k] >= 'A' and input\_string[k] <= 'z') == true and (input\_string[k + 1] >= 'A' and input\_string[k + 1] <= 'z') == true)

{

cout << "Две переменные не могут стоять подряд." << endl;

exit(0);

}

if ((input\_string[k] == '+' or input\_string[k] == '-' or input\_string[k] == '\*' or input\_string[k] == '/') == true and (input\_string[k + 1] == '+' or input\_string[k + 1] == '-' or input\_string[k + 1] == '\*' or input\_string[k + 1] == '/') == true)

{

cout << "Два оператора не могут стоять подряд." << endl;

exit(0);

}

if ((input\_string[k] == '(') == true and (input\_string[k+1] == '+' or input\_string[k+1] == '-' or input\_string[k+1] == '\*' or input\_string[k+1] == '/') == true)

{

cout << "После открывающей скобки должна идти переменная, а не оператор." << endl;

exit(0);

}

if ((input\_string[k + 1] == ')') == true and (input\_string[k] == '+' or input\_string[k] == '-' or input\_string[k] == '\*' or input\_string[k] == '/') == true)

{

cout << "Перед закрытием скобки должна идти переменная, а не оператор." << endl;

exit(0);

}

if ((input\_string[k + 1] == '(') == true and (input\_string[k] >= 'A' and input\_string[k] <= 'z') == true)

{

cout << "Перед открытием скобки должен идти оператор, а не переменная." << endl;

exit(0);

}

if ((input\_string[0] == '+' or input\_string[0] == '-' or input\_string[0] == '\*' or input\_string[0] == '/') == true)

{

cout << "Выражение должно начинаться с переменной или скобки." << endl;

exit(0);

}

if ((input\_string[k + 1] == '\0') == true and (input\_string[k] == '+' or input\_string[k] == '-' or input\_string[k] == '\*' or input\_string[k] == '/') == true)

{

cout << "Выражение должно оканчиваться переменной или скобкой" << endl;

exit(0);

}

k++;

}

}

// (a+(b/c– d))\*e

// a= 5.6

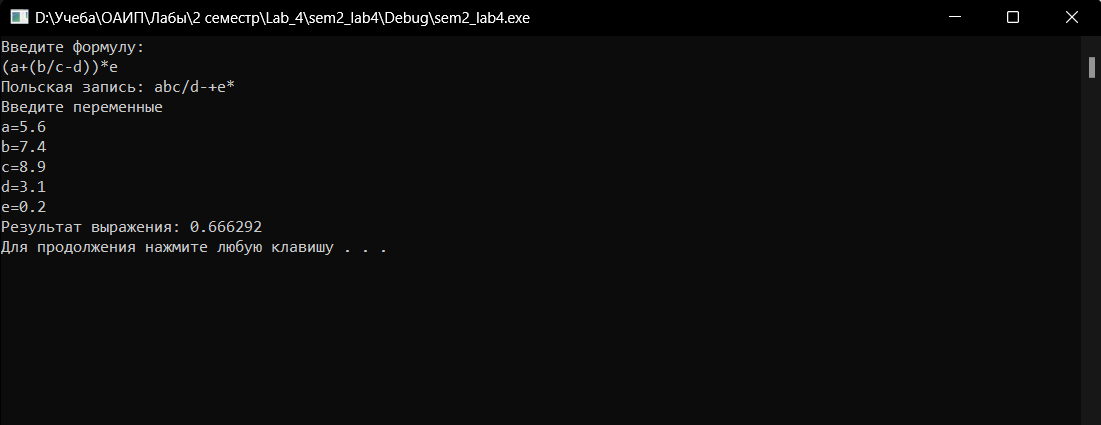
// b= 7.4

// c= 8.9

// d= 3.1

// e= 0.2

// res= 0.666



Вывод: изучили правила формирования постфиксной записи арифметических выражений с использованием стека.