БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

ОТЧЕТ

по лабораторной работе

**«Обратная польская запись»**

|  |  |  |
| --- | --- | --- |
| Выполнила |  | Семенова В. С. |
| Проверила |  | Шатилова О. О. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск 2023

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

**Общая постановка задачи**

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

**Вариант 11**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выражение | a | b | c | d | e | Результат |
| a – (b / c ∙ (d + e)) | 5.6 | 3.2 | 0.9 | 1.7 | 4.8 | – 17.51 |

**Листинг**

#include <iostream>

#include <cmath>

#include <string.h>

#include <string>

#include <windows.h>

using namespace std;

struct Numbers {

double u;

Numbers\* next;

};

struct Symbols{

char s;

Symbols\* next;

};

void InStack\_N(Numbers\*\*, double);

void InStack\_S(Symbols\*\*, char);

Symbols\* pull\_S(Symbols\*\*);

Numbers\* pull\_N(Numbers\*\*);

int prio(char);

void PEDRO(Numbers\*\*, char, double\*);

bool proverrka(string);

bool proverka(string);

void InStack\_N(Numbers\*\* p, double s) {

Numbers\* t = new Numbers;

t->u = s;

t->next = \*p;

\*p = t;

}

void InStack\_S(Symbols\*\* p, char s) {

Symbols\* t = new Symbols;

t->s = s;

t->next = \*p;

\*p = t;

}

Symbols\* pull\_S(Symbols\*\* b) {

Symbols\* t = new Symbols;

t = \*b;

\*b = (\*b)->next;

return t;

}

Numbers\* pull\_N(Numbers\*\* b) {

Numbers\* t = new Numbers;

t = \*b;

\*b = (\*b)->next;

return t;

}

int prio(char c) {

switch (c) {

case '/': return 2;

case '\*': return 2;

case '+': return 1;

case '-': return 1;

case '(': return 0;

case ')': return 3;

default:return -1;

}

}

void PEDRO(Numbers\*\* N, char s, double\* res) {

double n1, n2;

if ((\*N)->next != NULL) {

n2 = pull\_N(N)->u;

n1 = pull\_N(N)->u;

switch (s) {

case '+': (\*res) = n1 + n2; break;

case '-': (\*res) = n1 - n2; break;

case '\*': (\*res) = n1 \* n2; break;

case '/': (\*res) = n1 / n2; break;

default: cout << "Ошибка !\n";

}

InStack\_N(N, \*res);

}

}

bool proverrka(string nac)

{

int len=0, len1=0, len2=0, len3=0, f=0,i=0;

while(nac[f]!='\0')

{

if ((nac[0] == '+') or (nac[0] == '\*') or (nac[0] == '-') or (nac[0] == '/'))

{

cout << "Выражение не может начинаться со знака операции!" << endl;

return true;

}

if (nac[f] == '(')

{

len++;

}

if (nac[f] == ')')

{

len1++;

}

if ((nac[f] == '+') || (nac[f] == '\*') || (nac[f] == '-') || (nac[f] == '/'))

{

len2++;

}

if (nac[f] >= 'a' && nac[f] <= 'z')

{

len3++;

}

f++;

}

if (f == 1)

{

cout << "Выражение не может состоять из одного символа!" << endl;

return true;

}

if (len != len1)

{

cout << "Количество открывающих и закрывающих скобок должно быть равно!" << endl;

return true;

}

if (len2 == len3)

{

cout << "Неправильный ввод1" << endl;

return true;

}

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

{

if (((nac[i] == '+')|| (nac[i] == '\*')|| (nac[i] == '-')|| (nac[i] == '/')) && ((nac[i + 1] == '+') || (nac[i + 1] == '\*') || (nac[i + 1] == '-') || (nac[i + 1] == '/')))

{

cout << "Знаки операции не могут стоять рядом!" << endl;

return true;

}

else

{

if ((nac[i] >= 'a' && nac[i] <= 'z') && (nac[i + 1] >= 'a' && nac[i + 1] <= 'z'))

{

cout << "Переменные должны разделяться знаками операций!" << endl;

return true;

}

else

{

if ((nac[i] >= 'a' && nac[i] <= 'z') && (nac[i + 1] == '('))

{

cout << "После переменной не может стоять открывающая скобка!" << endl;

return true;

}

else

{

if ((nac[i] == '(') && (nac[i + 1] == ')'))

{

cout << "Внутри скобок должно быть выражение!" << endl;

return true;

}

}

}

}

i++;

}

return false;

}

bool proverka(string D) {

int p = 0, u = 0, i=1;

if (D[0] == '-') {

p++;

}

else {

if (D[0] == '.') {

p++; u++;

}

else {

for (char ii = '0'; ii <= '9'; ii++) {

if (D[0] == ii) {

p++;

}

}

}

}

if (p == 1) {

while(D[i]!='\0'){

if (D[i] == ' ') {

return false;

}

for (char ii = '0'; ii <= '9'; ii++) {

if (D[i] == ii) {

p++;

}

}

if (u == 0) {

if (D[i] == '.') {

p++; u++;

}

}

i++;

}

}

else {

return false;

}

if (p != i) {

return false;

}

if (i+1 > 1 && D[0] == '0' && D[1] != '.') {

return false;

}

if (i+1 > 1 && D[0] == '-' && D[1] == '0' && D[2] != '.') {

return false;

}

return true;

}

void main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

int i = 0, st = 0, m = 0, o = 0, f, j = 0; string nac, OPZ, L; Symbols\* S = NULL, \* k; Numbers\* N = NULL; bool skoba = FALSE, prov = TRUE; double\* a, res=NULL; char buf[10];

while (true) {

cout << "Введите выражение" << endl;

cin >> nac;

if (proverrka(nac)) return;

//cout << "a = "; cin >> a[0]; cout << "b = "; cin >> a[1]; cout << "c = "; cin >> a[2]; cout << "d = "; cin >> a[3]; cout << "e = "; cin >> a[4];

OPZ = "";

while (nac[i] != '\0') {

if ((nac[i] >= 'A' && nac[i] <= 'Z') || (nac[i] >= 'a' && nac[i] <= 'z')) { OPZ += nac[i]; buf[st] = nac[i]; st++; }

if (nac[i] == '/' || nac[i] == '\*' || nac[i] == '+' || nac[i] == '-') {

if (S == NULL || prio(S->s) < prio(nac[i])) InStack\_S(&S, nac[i]);

else {

while (S != NULL && prio(S->s) >= prio(nac[i])) {

k = pull\_S(&S); OPZ += k->s; delete k;

}

InStack\_S(&S, nac[i]);

}

}

if (nac[i] == '(') { InStack\_S(&S, nac[i]); skoba = TRUE; }

if (nac[i] == ')') {

if (!skoba) { cout << "Нет открывающей скобки!" << endl; prov = FALSE; break; }

else {

while (S->s != '(') { k = pull\_S(&S); OPZ += k->s; delete k; }

if (S->s == '(') { k = S; S = S->next; delete k; }

}

}

i++;

}

while (S != NULL) {

if (S->s == '(') { cout << "Нет закрывающей скобки!" << endl; prov = FALSE; break; }

else {

k = pull\_S(&S); OPZ += k->s; delete k;

}

}

if (!prov) cout << "Ошибка ввода!";

else

cout << OPZ << endl;

a = new double[st]; bool p;

for (i = 0; i < st; i++) {

p = false;

if (i == 0) {

cout << "\nВведите значение переменой: ";

cin >> a[i];

}

else {

for (j = i - 1; j >= 0; j--) {

if (buf[j] == buf[i]) {

p = true; break;

}

}

if (p) a[i] = a[j];

else {

cout << "\nВведите значение переменой: ";

cin >> a[i];

}

}

}

i = 0;

while (OPZ[m] != '\0') {

if (OPZ[m]== '/' || OPZ[m] == '\*' || OPZ[m] == '+' || OPZ[m] == '-') PEDRO(&N, OPZ[m], &res);

else {InStack\_N(&N, a[i]); i++; }

m++;

}

if (N->next == NULL) {

cout << "Ответ: " << res << endl; return;

}

}

}

Проверка на примере индивидуального задания:

