Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

­­­­­

Основы алгоритмизации и программирования

Отчет по лабораторной работе №13

««Обратная польская запись»

Выполнил: Усов А.М.

Студент группы 310901

Преподаватель: Кабариха В. А.

Минск 2023

Цель: сформировать знания и умения по работе с подпрограммами, приобрести навыки написания программ с использованием обратной польской записи (ОПЗ).

# Задание 1. Постфиксной формой записи (ОПЗ) выражения a◦b называется запись, в которой знак операции размещен за операндами ab◦.

# Например:

# Обычная запись Обратная польская запись

# a-b a b -

# a\*b+c a b \* c +

# a\*(b+c) a b c + \*

# (a+c)/(c\*a-d) a c + c a \* d - /

# Описать функции, которая вычисляет значение заданного выражения. Входные данные. В первой строке содержит обратную польскую запись арифметического выражения. Все операнды целые положительные числа. Выходные данные. Вывести результат вычисления ОПЗ. Технические требования. Используются знаки операций: +, -, \*, /. Примеры

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

template <typename T>

class Stack {

class Node {

T data;

Node\* prev;

public:

Node() : prev(nullptr) {}

Node(T data, Node\* node) : data(data), prev(node) {}

T getData() { return data; }

Node\* getPrev() { return prev; }

void setPrev(Node\* prev) { this->prev = prev; }

void setData(T data) { this->data = data; }

};

Node\* tail;

int size;

public:

Stack() : tail(nullptr), size(0) {}

~Stack() {

while (tail != nullptr) {

Node\* temp = tail;

tail = tail->getPrev();

delete temp;

}

}

void push(T data) {

if (tail == nullptr) {

tail = new Node(data, nullptr);

}

else {

Node\* newNode = new Node(data, tail);

tail = newNode;

}

size++;

}

T pop() {

if (size == 0) {

throw "Stack is empty";

}

T data = tail->getData();

tail = tail->getPrev();

size--;

return data;

}

T peek() {

if (size == 0) {

throw "Stack is empty";

}

return tail->getData();

}

bool isEmpty() {

return size == 0;

}

};

float solveOPZ(string expression) {

Stack<float> stack;

for (int i = 0; i < expression.size(); i++) {

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

continue;

}

int x = expression[i] - '0';

if (expression[i] >= '0' && expression[i] <= '9') {

float number = 0;

while (expression[i] != ' ' && i < expression.size()) {

number = number \* 10 + (expression[i] - '0');

i++;

}

stack.push(number);

}

else {

float a = stack.pop();

float b = stack.pop();

switch (expression[i]) {

case '+': {

stack.push(a + b);

break;

}

case '-': {

stack.push(b - a);

break;

}

case '\*': {

stack.push(a \* b);

break;

}

case '/': {

if (a == 0) {

cout << "error devision to zero";

return -1;

}

stack.push(b / a);

break;

}

}

}

}

if (stack.isEmpty()) {

cout << "Stack is empty" << endl;

return -1;

}

else {

float result = stack.pop();

stack.~Stack();

return result;

}

}

string\* getExpressionFromFile() {

ifstream file("input.txt");

if (file.is\_open()) {

string\* expression = new string[10];

string temp;

int i = 0;

while (getline(file, temp)) {

expression[i++] = temp;

}

return expression;

file.close();

}

else {

throw "File not found";

}

}

int main() {

string\* arrExp = getExpressionFromFile();

//print arrExp

for (int i = 0; i < 10; i++) {

cout << arrExp[i] << endl;

cout << solveOPZ(arrExp[i]) << endl << endl;

}

ofstream file("output.txt");

if (file.is\_open()) {

for (int i = 0; i < 10; i++) {

if (solveOPZ(arrExp[i]) == -1) continue;

file << solveOPZ(arrExp[i]) << endl;

}

file.close();

}

string expression1 = "3 1 +";

string expression2 = "12 5 \* 10 -";

string expression3 = "1 2 30 + \*";

string expression4 = "2 10 + 2 4 + 6 - 2 /";

if (expression4 == arrExp[3]) {

cout << "123213" << endl;

}

cout << solveOPZ(expression1) << endl;

cout << solveOPZ(expression2) << endl;

cout << solveOPZ(expression3) << endl;

cout << solveOPZ(expression4) << endl;

if (solveOPZ(expression1) == 4) {

cout << "Test 1 passed" << endl;

}

else {

cout << "Test 1 failed" << endl;

}

if (solveOPZ(expression2) == 50) {

cout << "Test 2 passed" << endl;

}

else {

cout << "Test 2 failed" << endl;

}

if (solveOPZ(expression3) == 32) {

cout << "Test 3 passed" << endl;

}

else {

cout << "Test 3 failed" << endl;

}

if (solveOPZ(expression4) == 6) {

cout << "Test 4 passed" << solveOPZ(expression4)<< endl;

}

else {

cout << "Test 4 failed" << endl;

}

string expression;

cout << "Enter expression in reverse polish notation: " << endl;

cout << " use this format: 3 1 + " << endl;

getline(cin, expression);

cout << "Result : " << solveOPZ(expression) << endl;

return 0;

}

# Результат работы программы представлен на рисунке 1.

# 

Рисунок 1 – Результат выполнения программы

Входные данные и выходные данные показаны на рисункая 2 – 3.

# 

Рисунок 2 – входные данные



Рисунок 3 – выходные данные