**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИС**

ОТЧЁТ

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и Структуры Данных»**

Тема: Списочные структуры

|  |  |  |
| --- | --- | --- |
| Студент гр. 1375 |  | Подберёзский А.Д. |
| Преподаватель |  | Пелевин М.С. |

Санкт-Петербург

2022

**Цель работы:**

Знакомство со списочными структурами, их реализацией и использованием

**Задание:**

1. Реализовать следующие структуры: односвязный список, динамический массив и стек. Стек можно реализовать как на базе списка, так и отдельно.
2. Использовать стек для реализации [алгоритма сортировочной станции](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D0%BE%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D1%86%D0%B8%D0%B8). Разрешённые символы в исходном выражении: +, -, \*, /, ^, sin, cos, (, ), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Для упрощения разбиения входной строки на [токены](https://en.wikipedia.org/wiki/Lexical_analysis" \l "Tokenization) разрешается отделять каждый символ пробелом.

**Код программы:**

**Файл DynamicArray.h:**

#include <string>

#include "Stack.h"

char priority(char op) {

switch (op) {

case '^':

return 3;

break;

case '\*':

case '/':

return 2;

break;

case '+':

case '-':

return 1;

break;

default:

return op;

break;

}

}

bool isOperator(char op) {

if (op == '-' || op == '+' || op == '\*' || op == '/' || op == '^')

{

return true;

}

else

{

return false;

}

}

class DynamicArray

{

public:

int\* head = nullptr;

int size = 0;

DynamicArray(int arraySize = 0)

{

if (arraySize > 0)

{

head = new int[arraySize];

size = arraySize;

}

}

void resize(int arraySize)

{

if (arraySize > 0)

{

int\* tmp = new int[arraySize];

for (int i = 0; i < arraySize; i++)

{

if (i < size)

{

tmp[i] = head[i];

}

}

delete[] head;

head = tmp;

size = arraySize;

}

}

~DynamicArray()

{

delete[] head;

}

int& operator[](int i)

{

return head[i];

}

};

**Файл List.h:**

#pragma once

#include <iostream>

class List

{

struct Node

{

int value;

Node\* next;

Node(int val)

:value(val)

,next(nullptr)

{}

};

Node\* head;

Node\* tail;

int listSize;

void clear()//function to clear the list

{

Node\* current = head;

while (current)

{

Node\* temp = current;

current = current->next;

delete temp;

}

head = nullptr;

tail = nullptr;

}

public:

List()

:head(nullptr)

,tail(nullptr)

,listSize(0)

{}

List(size\_t size)

:head(nullptr)

,tail(nullptr)

,listSize(size)

{

if (size)

{

head = new Node{0};

Node\* current = head;

for (size\_t i = 0; i < size - 1; i++)

{

current->next = new Node{0};

current = current->next;

}

}

}

bool empty(){//if list empty - true

return head == nullptr;

}

void append(int val){//pushing new element in list

Node\* temp = new Node(val);

if (empty())

{

head = temp;

tail = temp;

return;

}

head->next = temp;

head = temp;

listSize++;

}

void remove(){//remove from list

Node\* temp = head->next;

delete head;

head = temp;

listSize--;

}

void setAt(size\_t index, int value){//function for changing element at "index" pos

size\_t currentIndex = 0;

Node\* current = head;

while (current)

{

if (index == currentIndex)

{

current->value = value;

break;

}

++currentIndex;

current = current->next;

}

}

int size(){//return listSize

return listSize;

}

void del(int pos){//delete Node at pos

Node\* temp1 = head->next;

Node\* temp2 = nullptr;

for (int i = 0; i < pos; ++i)

{

temp2 = temp1;

temp1 = temp1->next;

}

if (temp2 != nullptr) {

temp2 = temp1->next;

}

delete temp1;

}

void output(){//output of List

Node\* out = head;

while (out)

{

std::cout << out->value << ' ';

}

std::cout << std::endl;

}

~List()

{

clear();

}

};

**Файл Stack.h:**

#pragma once

class Stack {

private:

struct Node {

char value;

Node\* next = nullptr;

};

Node\* head;

unsigned stackSize = 0;

public:

Stack() {

head = nullptr;

};

~Stack() {

while (!empty()) {

pop();

}

};

void push(char val){

Node\* temp = new Node;

temp->value = val;

temp->next = head;

head = temp;

stackSize++;

}

int size(){

return stackSize;

}

bool empty(){

return head == nullptr;

}

char top(){

if (!empty())

{

return head->value;

}

}

char peek(int pos){

if (!empty() && (pos < stackSize))

{

Node\* temp = head;

for (int i = 0; i < pos; ++i)

{

temp = temp->next;

}

return temp->value;

}

}

void pop(){

if (!empty())

{

Node\* temp = head;

head = head->next;

delete temp;

stackSize--;

}

}

bool find(char val){

if (!empty()) {

Node\* temp = head;

for (int i = 0; i < size(); ++i)

{

if (temp->value == val) {

return true;

}

temp = temp->next;

}

return false;

}

}

char& operator[](int pos){

Node\* temp = head;

for (int i = size() - 1; i >= pos; i--)

{

temp = temp->next;

}

return temp->value;

}

};

**Файл main.cpp:**

#include <iostream>

#include <string>

#include "Stack.h"

#include "functions.h"

void SortStationAlgorithm(std::string input, std::string &output, int &error) {

int i = 0;

Stack op;

while (i < input.size() && !error)

{

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

{

output += input[i];

}

else if (input[i] == 's' || input[i] == 'c')

{

op.push(input[i]);

i += 2;

}

else if (isOperator(input[i]))

{

while (isOperator(op.top()) && (priority(input[i]) <= priority(op.top())) && !op.empty())

{

output += " ";

output += op.top();

op.pop();

}

output += " ";

op.push(input[i]);

}

else if (input[i] == '(')

{

op.push(input[i]);

}

else if (input[i] == ')')

{

while (op.top() != '(')

{

output += " ";

output += op.top();

op.pop();

if (op.empty())

{

error = 1;

break;

}

}

op.pop();

if (op.top() == 's')

{

output += " sin";

op.pop();

}

else if (op.top() == 'c')

{

output += " cos";

op.pop();

}

}

++i;

}

while (!op.empty())

{

if (op.top() == '(')

{

error = 1;

}

output += ' ';

output += op.top();

op.pop();

}

}

int main(int argc, char\* argv[]) {

std::string inputExpression, outputExpression;

int errorNum = 0;

std::cout << "Hi! Please insert your expression:" << std::endl;

std::cin >> inputExpression;

SortStationAlgorithm(inputExpression, outputExpression, errorNum);

std::cout << "Here is your remade expression:" << std::endl;

if (errorNum == 0)

{

std::cout << outputExpression << std::endl;

}

else if (errorNum == 1)

{

std::cout << "Oh shit! You've fucked up with brackets! GTFO from here, you morron!" << std::endl;

}

system("pause");

return 0;

}

**Пример работы программы:**

