Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

**Лабораторна робота № 5**

**«Структури: стек, черга, дерева»**

Виконав

ст. гр. ІП-22-1

Микитій Ю.М.

Перевірив

доцент Процюк В.Р.

Івано-Франківськ

2023

**Мета:** сформувати практичні навички організації таких поширених структур даних як стеки, черги і дерева, а також їх використання при вирішенні завдань

**Умова: Варіант 15.**

2,4,6

2) Розташувати елементи цілочисельного масиву розміром N в зворотньому порялку з використанням стеку.

4) Нехай є файл дійсних чисел і деяке число С. Використовуючи чергу, надрукувати спочатку всі елементи, що менші за числа С, а потім всі інші числа.

6) Знайти максимальний елемент бінарного дерева і кількість повторень максимального елемента в даному дереві.

**Розв’язок**

#include <iostream>

#include <vector>

#include <fstream>

using namespace std;

// Структура для реалізації стеку

struct Stack {

vector<int> data;

// Функція додавання елемента в стек

void push(int value) {

data.push\_back(value);

}

// Функція видалення останнього елемента зі стеку

void pop() {

if (!data.empty()) {

data.pop\_back();

}

}

// Функція отримання останнього елемента стеку

int top() {

if (!data.empty()) {

return data.back();

}

else {

return -1; // Якщо стек порожній, повертаємо -1 (можна обрану константу помилки)

}

}

// Функція перевірки, чи стек порожній

bool empty() {

return data.empty();

}

};

// Структура для реалізації черги

struct Queue {

vector<int> data;

// Функція додавання елемента в чергу

void push(int value) {

data.push\_back(value);

}

// Функція видалення першого елемента з черги

void pop() {

if (!data.empty()) {

data.erase(data.begin());

}

}

// Функція отримання першого елемента черги

int front() {

if (!data.empty()) {

return data.front();

}

else {

return -1; // Якщо черга порожня, повертаємо -1 (можна обрану константу помилки)

}

}

// Функція перевірки, чи черга порожня

bool empty() {

return data.empty();

}

};

// Структура для представлення вузла бінарного дерева

struct TreeNode {

int data;

TreeNode\* left;

TreeNode\* right;

TreeNode(int value) : data(value), left(nullptr), right(nullptr) {}

};

// Функція для знаходження максимального елемента в бінарному дереві

int findMax(TreeNode\* root) {

if (root == nullptr) {

return INT\_MIN; // Повертаємо мінімальне можливе значення як помилку

}

int max = root->data;

int leftMax = findMax(root->left);

int rightMax = findMax(root->right);

if (leftMax > max) {

max = leftMax;

}

if (rightMax > max) {

max = rightMax;

}

return max;

}

// Функція для підрахунку кількості повторень максимального елемента в бінарному дереві

int countMax(TreeNode\* root, int max) {

if (root == nullptr) {

return 0;

}

int count = 0;

if (root->data == max) {

count = 1;

}

int leftCount = countMax(root->left, max);

int rightCount = countMax(root->right, max);

return count + leftCount + rightCount;

}

int main() {

Stack muStack;

const int N = 5; // Розмір масиву

int array[N] = { 1, 2, 3, 4, 5 };

cout << "Array:";

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

cout << array[i] << " ";

}

cout << "\nArray in reverse order using a stack: ";

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

muStack.push(array[i]);

}

while (!muStack.empty()) {

cout << muStack.top() << " ";

muStack.pop();

}

cout << std::endl;

cout << "\n------------------------------------------------------------------\n";

double C = 26.0;

std::ifstream inputFile("input.txt");

if (inputFile.is\_open()) {

double number;

Queue lowerQueue, greaterQueue;

while (inputFile >> number) {

if (number < C) {

lowerQueue.push(number);

}

else {

greaterQueue.push(number);

}

}

std::cout << "Numbers less than " << C << ": ";

while (!lowerQueue.empty()) {

std::cout << lowerQueue.front() << " ";

lowerQueue.pop();

}

std::cout << "\nNumbers greater than or equal to " << C << ": ";

while (!greaterQueue.empty()) {

std::cout << greaterQueue.front() << " ";

greaterQueue.pop();

}

std::cout << std::endl;

inputFile.close();

}

else {

std::cerr << "Error opening the file." << std::endl;

}

cout << "\n------------------------------------------------------------------\n";

TreeNode\* root = new TreeNode(10);

root->left = new TreeNode(5);

root->right = new TreeNode(15);

root->left->left = new TreeNode(3);

root->left->right = new TreeNode(7);

root->right->right = new TreeNode(15);

// Знайти максимальний елемент та кількість повторень максимального елемента.

int max = findMax(root);

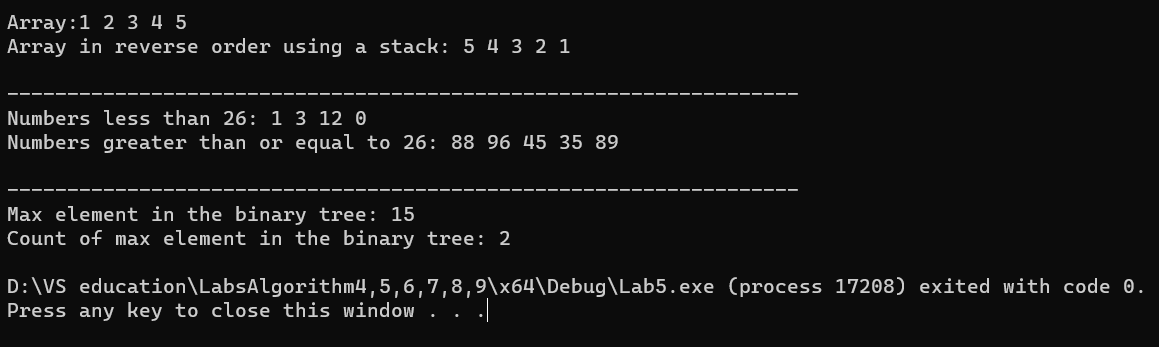
int count = countMax(root, max);

std::cout << "Max element in the binary tree: " << max << std::endl;

std::cout << "Count of max element in the binary tree: " << count << std::endl;

return 0;

}

**Результат:  
**

**Висновок:** на цій лабораторній роботі я сформував практичні навички організації таких поширених структур даних як стеки, черги і дерева, а також їх використав при вирішенні завдань