МИНПРОСВЕЩЕНИЯ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Нижегородский государственный педагогический университет

имени Козьмы Минина»

Факультет информационных технологий

Кафедра информационных систем и цифровых сервисов в управлении

**ОТЧЕТ**

## по лабораторной работе №1 на тему «Работа со стеком и очередью»

**по дисциплине «Объектно-ориентированное программирование»**

Направление подготовки: **Информационные системы и технологии**

Профиль подготовки: **Информационные системы и технологии**

**Выполнила:** Гусева Екатерина Александровна

**Группа:** ИСТ-23-2

г. Нижний Новгород

2025 год

**Введение**

Динамические структуры данных — это структуры, размер и форма которых могут изменяться во время выполнения программы. В отличие от статических структур (например, массивов фиксированного размера), они позволяют гибко управлять памятью: выделять её по мере необходимости и освобождать, когда данные больше не нужны. Работа с такими структурами обычно осуществляется через указатели или ссылки, которые связывают элементы между собой.

Стек — это линейная структура данных, которая позволяет добавлять и удалять элементы только из начала. Она работает по принципу LIFO — Last In, First Out (англ. «последним пришёл — первым ушёл»). Последний добавленный в стек элемент должен будет покинуть его раньше остальных.

Очередь — это тип структуры данных, в которой элементы хранятся в определённой последовательности. В отличие от стеков, операции с очередью выполняются по принципу FIFO (первым пришёл — первым обслужен). Элемент, который вставляется первым, первым и удаляется.

**Ход работы:**

Задание:

Разработать программу для работы с информационно-логическими структурами – стеком и очередь. Программа должна обеспечивать следующие возможности работы со структурами:

* добавление элемента согласно правилам работы со структурой;
* удаление элемента согласно правилам работы со структурой;
* удаление всех элементов структуры;
* отображение элементов структуры в прямом порядке.

При всех операциях соблюдать правила работы с информационно-логическими структурами – доступ осуществляется только к первому и/или последнему элементу структуры. Для сохранения, отображения элементов и удаления перекладывать элементы структуры в другую аналогичную структуру.

Создать стек и очередь, содержащие целые числа в диапазоне от -20 до +60. Из созданных стека и очереди удалить все элементы, кратные 7, используя аналогичную структуру для перекладывания элементов.

**Код:**

#include <iostream>

struct Node {

int data;

Node\* next;

};

// Класс для реализации стека

class Stack {

private:

Node\* top;

public:

Stack() : top(nullptr) {}

// Добавление элемента в стек

void push(int value) {

Node\* newNode = new Node{ value, top };

top = newNode;

}

// Удаление элемента из стека

void pop() {

if (top == nullptr) {

std::cout << "Stack is empty." << std::endl;

return;

}

Node\* temp = top;

top = top->next;

delete temp;

}

// Очистка стека

void clear() {

while (top != nullptr) {

pop();

}

}

// Отображение элементов стека в прямом порядке

void display() {

Node\* current = top;

while (current != nullptr) {

std::cout << current->data << " ";

current = current->next;

}

std::cout << std::endl;

}

// Удаление элементов, кратных 7, с использованием вспомогательного стека

void removeMultiplesOf7() {

Stack tempStack;

while (top != nullptr) {

if (top->data % 7 != 0) {

tempStack.push(top->data);

}

pop();

}

// Перекладываем обратно в оригинальный стек

while (tempStack.top != nullptr) {

push(tempStack.top->data);

tempStack.pop();

}

}

~Stack() {

clear();

}

};

// Класс для реализации очереди

class Queue {

private:

Node\* front;

Node\* rear;

public:

Queue() : front(nullptr), rear(nullptr) {}

// Добавление элемента в очередь

void enqueue(int value) {

Node\* newNode = new Node{ value, nullptr };

if (rear != nullptr) {

rear->next = newNode;

}

rear = newNode;

if (front == nullptr) {

front = newNode;

}

}

// Удаление элемента из очереди

void dequeue() {

if (front == nullptr) {

std::cout << "Queue is empty." << std::endl;

return;

}

Node\* temp = front;

front = front->next;

if (front == nullptr) {

rear = nullptr;

}

delete temp;

}

// Очистка очереди

void clear() {

while (front != nullptr) {

dequeue();

}

}

// Отображение элементов очереди в прямом порядке

void display() const {

Node\* current = front;

while (current != nullptr) {

std::cout << current->data << " ";

current = current->next;

}

std::cout << std::endl;

}

// Удаление элементов, кратных 7, с использованием вспомогательной очереди

void removeMultiplesOf7() {

Queue tempQueue;

while (front != nullptr) {

if (front->data % 7 != 0) {

tempQueue.enqueue(front->data);

}

dequeue();

}

// Перекладываем обратно в оригинальную очередь

while (tempQueue.front != nullptr) {

enqueue(tempQueue.front->data);

tempQueue.dequeue();

}

}

~Queue() {

clear();

}

};

int main() {

Stack stack;

Queue queue;

// Добавление элементов в стек и очередь

for (int i = -20; i <= 60; ++i) {

stack.push(i);

queue.enqueue(i);

}

std::cout << "Initial Stack: ";

stack.display();

std::cout << "Initial Queue: ";

queue.display();

// Удаление элементов, кратных 7

stack.removeMultiplesOf7();

queue.removeMultiplesOf7();

std::cout << "Stack after removing multiples of 7: ";

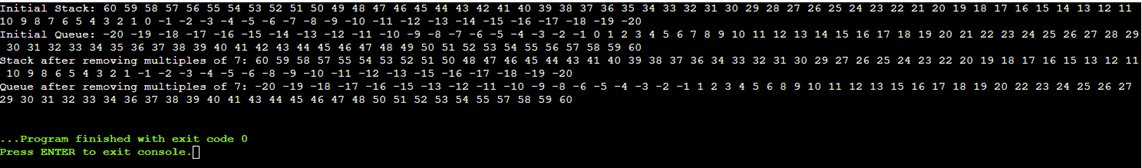
stack.display();

std::cout << "Queue after removing multiples of 7: ";

queue.display();

return 0;

}



**Задание 2:**

Реверс строки с использованием стека

Описание: Напишите программу, которая использует стек для реверса строки.

**Код:**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

string input;

cout << "Введите строку: ";

getline(cin, input); // Чтение строки с учётом пробелов

stack<char> charStack; // Стек для хранения символов

// Помещаем каждый символ строки в стек

for (char c : input) {

charStack.push(c);

}

// Извлекаем символы из стека для формирования перевёрнутой строки

string reversedString;

while (!charStack.empty()) {

reversedString += charStack.top(); // Добавляем верхний элемент

charStack.pop(); // Удаляем верхний элемент

}

cout << "Перевёрнутая строка: " << reversedString << endl;

return 0;

}



**Задание 3:**

Проверка симметричности последовательности с использованием стека

Описание: Напишите программу, которая использует стек для проверки, является ли последовательность чисел симметричной (палиндромом).

**Код:**

#include <iostream>

#include <stack>

#include <vector>

#include <sstream>

#include <string>

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

string input;

cout << "Введите последовательность чисел через пробел: ";

getline(cin, input); // Чтение всей строки

vector<int> numbers;

stringstream ss(input);

int num;

// Разбиваем строку на числа

while (ss >> num) {

numbers.push\_back(num);

}

if (numbers.empty()) {

cout << "Ошибка: последовательность пуста!" << endl;

return 1;

}

stack<int> stack;

size\_t middle = numbers.size() / 2;

// Помещаем первую половину в стек

for (size\_t i = 0; i < middle; ++i) {

stack.push(numbers[i]);

}

// Определяем стартовый индекс для второй половины

size\_t start = (numbers.size() % 2 == 0) ? middle : middle + 1;

bool isPalindrome = true;

// Сравниваем вторую половину с элементами стека

for (size\_t i = start; i < numbers.size(); ++i) {

if (stack.empty() || numbers[i] != stack.top()) {

isPalindrome = false;

break;

}

stack.pop();

}

if (isPalindrome) {

cout << "Последовательность симметрична!" << endl;

} else {

cout << "Последовательность НЕ симметрична!" << endl;

}

return 0;

}

