МИНОБРНАУКИ РОСИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Восточно-Сибирский государственный университет технологий и управления»

(ФГБОУ ВО ВСГУТУ)

Факультет компьютерных наук и технологий

Кафедра программной инженерии и искусственного интеллекта

**ОТЧЕТ**по лабораторной работе №3  
на тему «Моделирование стека и очереди»

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент гр. Б763-2 Угнивенко Д.А.

Проверил: Ассистент кафедры ПИиИИ

Николаева Т.В.

Улан-Удэ

2024

## Словесная постановка задачи:

Реализация простой очереди двумя способами: последовательным (хранение в массиве) и связным (хранение в списке).

## Алгоритмы решения задачи:

### 1. Начало

* Программа начинается.

### 2. Отображение меню

* Отображается главное меню:
  + 1. Выбрать реализацию (массив или список)
  + 2. Добавить элемент в очередь
  + 3. Удалить элемент из очереди
  + 4. Показать очередь
  + 5. Выйти

### 3. Пользовательский выбор (реализация очереди)

* Ввод: пользователь выбирает, какую структуру данных использовать:
  + 1. Массив — выбор очереди на массиве.
  + 2. Связный список — выбор очереди на связанном списке.

### 4. Меню операций с очередью

После того, как выбрана реализация очереди, отображаются следующие операции:

* Добавить элемент в очередь (enqueue)
  + Запрашиваем у пользователя значение элемента.
  + В зависимости от выбранной структуры данных:
    - Если очередь на массиве — элемент добавляется в массив.
    - Если очередь на списке — создаётся новый узел, который добавляется в конец списка.
* Удалить элемент из очереди (dequeue)
  + Если очередь не пуста, удаляется элемент из начала (массив или список).
* Показать очередь (display)
  + Если очередь не пуста, показывается её содержимое (массив или список).

### 5. Проверки и решения

* Очередь пуста?
  + Если очередь пуста, выводится сообщение об ошибке.
* Очередь переполнена (для массива)?
  + Если массив переполнен, выводится сообщение о переполнении.

### 6. Конец

* После того как пользователь выбирает выйти, программа завершает выполнение.

## Листинг программы:

#include <iostream>

using namespace std;

// --- Очередь на массиве ---

const int SIZE = 100; // Максимальный размер массива

int queueArray[SIZE]; // Массив для хранения элементов очереди

int frontArray = -1; // Индекс начала очереди

int rearArray = -1; // Индекс конца очереди

void enqueueArray(int value) {

if (rearArray == SIZE - 1) { // Проверка на переполнение

cout << "Очередь (массив) переполнена!" << endl;

return;

}

if (frontArray == -1) { // Если очередь пуста

frontArray = 0; // Устанавливаем начало

}

rearArray++; // Увеличиваем конец

queueArray[rearArray] = value; // Добавляем элемент

}

int dequeueArray() {

if (frontArray == -1 || frontArray > rearArray) { // Проверка на пустоту

cout << "Очередь (массив) пуста!" << endl;

return -1;

}

int removedValue = queueArray[frontArray]; // Сохраняем значение

frontArray++; // Сдвигаем начало

return removedValue;

}

void displayArray() {

if (frontArray == -1 || frontArray > rearArray) { // Проверка на пустоту

cout << "Очередь (массив) пуста!" << endl;

return;

}

cout << "Очередь (массив): ";

for (int i = frontArray; i <= rearArray; i++) { // Печать всех элементов

cout << queueArray[i] << " ";

}

cout << endl;

}

// --- Очередь на связном списке ---

struct Node {

int value; // Значение узла

Node\* next; // Указатель на следующий узел

};

Node\* frontList = nullptr; // Указатель на начало очереди

Node\* rearList = nullptr; // Указатель на конец очереди

void enqueueList(int value) {

Node\* newNode = new Node(); // Создаём новый узел

newNode->value = value; // Присваиваем значение

newNode->next = nullptr; // Последний узел, поэтому next = nullptr

if (rearList == nullptr) { // Если очередь пуста

frontList = rearList = newNode; // Начало и конец совпадают

}

else {

rearList->next = newNode; // Привязываем новый узел

rearList = newNode; // Обновляем указатель конца

}

}

int dequeueList() {

if (frontList == nullptr) { // Если очередь пуста

cout << "Очередь (список) пуста!" << endl;

return -1;

}

Node\* temp = frontList; // Временный указатель

int removedValue = frontList->value; // Сохраняем значение

frontList = frontList->next; // Сдвигаем начало

if (frontList == nullptr) { // Если очередь опустела

rearList = nullptr; // Обнуляем конец

}

delete temp; // Удаляем старый узел

return removedValue;

}

void displayList() {

if (frontList == nullptr) { // Если очередь пуста

cout << "Очередь (список) пуста!" << endl;

return;

}

cout << "Очередь (список): ";

Node\* current = frontList; // Начинаем с начала

while (current != nullptr) { // Проходим по всем узлам

cout << current->value << " ";

current = current->next;

}

cout << endl;

}

// --- Главная программа ---

int main() {

setlocale(LC\_ALL, "Russian");

int implementationChoice;

cout << "Выберите реализацию очереди:" << endl;

cout << "1. Последовательная (массив)" << endl;

cout << "2. Связная (список)" << endl;

cin >> implementationChoice;

while (true) {

cout << "\nМеню:" << endl;

cout << "1. Добавить элемент (enqueue)" << endl;

cout << "2. Удалить элемент (dequeue)" << endl;

cout << "3. Показать очередь (display)" << endl;

cout << "4. Выйти" << endl;

cout << "Ваш выбор: ";

int choice, value;

cin >> choice;

if (choice == 1) {

cout << "Введите значение: ";

cin >> value;

if (implementationChoice == 1) {

enqueueArray(value); // Добавление в очередь на массиве

}

else {

enqueueList(value); // Добавление в очередь на списке

}

}

else if (choice == 2) {

int removedValue;

if (implementationChoice == 1) {

removedValue = dequeueArray(); // Удаление из массива

}

else {

removedValue = dequeueList(); // Удаление из списка

}

if (removedValue != -1) {

cout << "Удалённый элемент: " << removedValue << endl;

}

}

else if (choice == 3) {

if (implementationChoice == 1) {

displayArray(); // Показ очереди на массиве

}

else {

displayList(); // Показ очереди на списке

}

}

else if (choice == 4) {

cout << "Выход из программы" << endl;

break;

}

else {

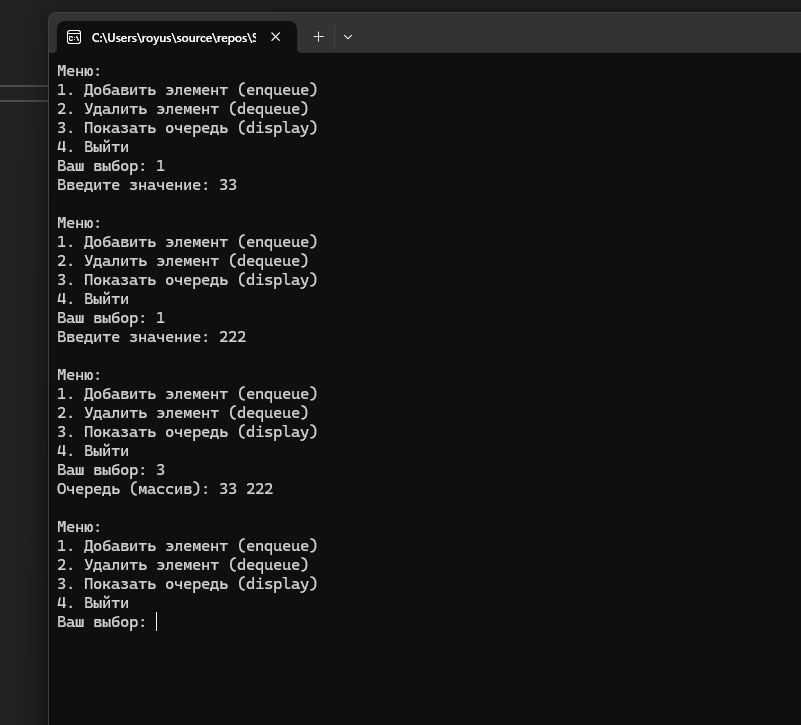
cout << "Неверный выбор. Попробуйте снова!" << endl;

}

}

return 0;

}

**Тестирование:**

