**МИНОБРНАУКИ РОССИИ**

**ФЕДЕРАЛЬАНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра высшей математики**

**Отчёт по лабораторной работе №9**

по курсу «Алгоритмизация и программирование»

на тему: **«Работа с очередями»**

Выполнил:

студент гр. КТбо1-1

**Бураков А. В.**

Проверил: ст.преподаватель каф. ИАСБ

**Барковский С. А.**

Таганрог 2024

**Цель**: Целью данной лабораторной работы является выработка практических навыков по работе с такими данными, как очереди, а также закрепление практических навыков по работе с данными типа, структура.

**Задачи**: Написать программы, реализующие линейную и циклическую очереди соответственно. В программах предусмотреть операции добавления, удаления, редактирование элементов очереди, а также вывод на экран всей очереди.

**Описание варианта:** Студент использует в качестве элементов очереди структуру из своего варианта задания, полученного к лабораторной работе No10.

**Реализация .**

#include <iostream>

#define MAXSIZE 3

#include <string>

class Patient {

std::string surname;

std::string name;

std::string patronymic;

int birthYear;

std::string phone;

std::string address;

int medicalCardNumber;

std::string bloodType;

char rhFactor;

std::string diagnosis;

public:

Patient(std::string \_surname, std::string \_name, std::string \_patronymic, int \_birthYear, std::string \_phone, std::string \_address, int \_medicalCardNumber, std::string \_bloodType, char \_rhFactor, std::string \_diagnosis) : surname(\_surname), name(\_name), patronymic(\_patronymic), birthYear(\_birthYear), phone(\_phone), address(\_address), medicalCardNumber(\_medicalCardNumber), bloodType(\_bloodType), rhFactor(\_rhFactor), diagnosis(\_diagnosis) {}

void cprint() {

std::cout << surname << " " << name << " " << patronymic << " " << birthYear << " " << phone << " " << address << " " << medicalCardNumber << " " << bloodType << " " << rhFactor << " " << diagnosis << "\n";

}

};

class Queue {

Patient \*array[MAXSIZE];

int front = 0;

int end = 0;

public:

void qstore(Patient\* data) {

if (end == MAXSIZE) {

std::cout << "Очередь полна\n";

return;

}

array[end] = data;

end++;

}

Patient\* qretrieve() {

if (end == front) {

std::cout << "Очередь пуста\n";

return NULL;

}

front++;

return array[front - 1];

}

void qcahge(int index, Patient\* data) {

if (front == end || index < 1 || index > end - front) {

std::cout << "Некорректный индекс\n";

return;

}

array[index + front -1] = data;

}

void qdisplay() {

if (front == end) {

std::cout << "Очередь пуста\n";

return;

}

std::cout << "Элементы очереди:\n";

for (int i = front; i < end; i++)

array[i]->cprint();

std::cout << "\n";

}

};

class CyclicQueue {

Patient\* array[MAXSIZE];

int front = 0;

int end = 0;

bool passing = false;

public:

void qstore(Patient\* q) {

if (passing && end == front) {

std::cout << "Очередь полна\n";

return;

}

array[end] = q;

end++;

if (end == MAXSIZE) {

end = 0;

passing = true;

}

}

Patient\* qretrieve(void) {

if (front == MAXSIZE - 1) {

front = 0;

passing = false;

return array[MAXSIZE - 1];

}

if (!passing && front == end) {

std::cout << "Очередь пуста\n";

return NULL;

}

front++;

return array[front - 1];

}

void qcahge(int index, Patient\* data) {

if (front == end && !passing) std::cout << "Очередь пуста\n";

if (!passing) {

if (index < 1 || index >= end - front) std::cout << "Некорректный индекс\n";

else array[index + front - 1] = data;

}

else {

if (index < 1 && index > end + MAXSIZE - front) std::cout << "Некорректный индекс\n";

else {

if (index + front - 1 > MAXSIZE - 1) array[index + front - MAXSIZE - 1] = data;

else array[index + front - 1] = data;

}

}

}

void qdisplay() {

if (front == end) {

std::cout << "Очередь пуста\n";

return;

}

std::cout << "Элементы очереди:\n";

if (passing) {

for (int i = front; i < MAXSIZE; ++i)

array[i]->cprint();

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

array[i]->cprint();

}

else {

for (int i = front; i < end; i++)

array[i]->cprint();

}

}

};

int main(){ setlocale(LC\_ALL, "Ru"); }

**Вывод:** В данной лабораторной работе я выработал практические навыки по работе с такими данными, как очереди, а также закрепил практические навыки по работе с данными типа, структура. Написал программы, реализующие линейную и циклическую очереди соответственно.