**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. Вариант 3.

**Реализация .**

#include <iostream>

#define MAX1 10

#define MAX2 10

#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 Print() {

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

}

};

class Queue {

Patient\* p[MAX1];

int rpos = 0;

int spos = 0;

public:

void qstore(Patient\* data) {

if (spos == MAX1) {

std::cout << "The queue is full\n";

return;

}

p[spos] = data;

spos++;

}

Patient\* qretrieve() {

if (spos == rpos) {

std::cout << "The queue is empty\n";

return NULL;

}

rpos++;

return p[rpos - 1];

}

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

if (rpos == spos || i < 1 || i > spos - rpos) {

std::cout << "Invalid index\n";

return;

}

p[i + rpos - 1] = data;

}

void qdisplay() {

if (rpos == spos) {

std::cout << "The queue is empty\n";

return;

}

for (int h = rpos; h < spos; h++)

p[h]->Print();

std::cout << "\n";

}

};

class CyclicQueue {

Patient\* array[MAX2];

int rpos = 0;

int spos = 0;

bool pass = false;

public:

void qstore(Patient\* q) {

if (pass && spos == rpos) {

std::cout << "The queue is full\n";

return;

}

array[spos] = q;

spos++;

if (spos == MAX2) {

spos = 0;

pass = true;

}

}

Patient\* qretrieve(void) {

if (rpos == MAX2 - 1) {

rpos = 0;

pass = false;

return array[MAX2 - 1];

}

if (!pass && rpos == spos) {

std::cout << "The queue is empty\n";

return NULL;}

rpos++;

return array[rpos - 1];

}

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

if (rpos == spos && pass) std::cout << "The queue is empty\n";

if (!pass) {

if (i - rpos < 1 || i >= spos - rpos) std::cout << "Invalid index\n";

else array[i + rpos - 1] = data;

}

else {

if (i < 1 && i > spos + MAX2 - rpos) std::cout << "Invalid index\n";

else {

if (i + rpos - 1 > MAX2 - 1) array[i + rpos - MAX2 - 1] = data;

else array[i + rpos - 1] = data;

}

}

}

void qdisplay() {

if (rpos == spos) {

std::cout << "The queue is empty\n";

return;

}

if (pass) {

for (int h = rpos; h < MAX2; ++h)

array[h]->Print();

for (int h = 0; h < spos; ++h)

array[h]->Print();

}

else {

for (int h = rpos; h < spos; h++)

array[h]->Print();

}

}

};

int main() {}

**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.