**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 №5**

по курсу «Алгоритмизация и программирование»

на тему: **«Работа со структурами»**

Выполнил:

студент гр. КТбо1-1

**Бураков А.В.**

Проверил:

**Барковский С. А.**

Таганрог 2024

**Цель**: Научиться работать со структурами

**Задачи**: Написать программу, которая записывает с клавиатуры в файл структуру согласно выданному варианту задания. В качестве разделителя полей структуры использовать символ табуляции. В программе реализовать:

а) дополнение существующего массива структур новыми структурами. Массив структур записывается в файл после добавления.

б) при повторном запуске программы в массив структур записываются ранее сохраненные структуры из файла.

в) поиск структуры с заданным значением выбранного элемента.

г) вывод на экран содержимого массива структур.

д) сортировка массива структур по заданному полю (элементу), например государство по численности.

е) интерфейс программы должен предоставлять пользователю выбор желаемой функции (добавление новой структуры, вывод на экран массива структур, поиск в массиве структур, сортировка массива структур).

**Описания варианта:** «Пациент»: фамилия, имя, отчество, год рождения, телефон, адрес, номер медицинской карты, группа крови, резус-фактор, диагноз.

**Реализация .**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <algorithm>

#include <Windows.h>

struct Patient {

std::string lastName;

std::string firstName;

std::string middleName;

std::string birthYear;

std::string phoneNumber;

std::string address;

std::string medicalCardNumber;

std::string bloodGroup;

std::string rhFactor;

std::string diagnosis;

};

//Вывод в консоль всего массива

void writeToConsole(std::vector<Patient>& patients) {

for (const Patient patient : patients) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

//Запись в файл

void writeToFile(std::vector<Patient>& patients) {

std::ofstream file("patients.txt");

for (Patient patient : patients) {

file << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

file.close();

}

//Чтение из консоли

void readFromConsole(std::vector<Patient>& patients) {

Patient newPatient;

std::cout << "Заполните информацию о пациенте:\n";

std::cout << "Фамилия: ";

std::cin >> newPatient.lastName;

std::cout << "Имя: ";

std::cin >> newPatient.firstName;

std::cout << "Отчество: ";

std::cin >> newPatient.middleName;

std::cout << "Год рождения: ";

std::cin >> newPatient.birthYear;

std::cout << "Номер телефона: ";

std::cin >> newPatient.phoneNumber;

std::cout << "Адрес: ";

std::cin >> newPatient.address;

std::cout << "Номер мед-карты: ";

std::cin >> newPatient.medicalCardNumber;

std::cout << "Группа крови: ";

std::cin >> newPatient.bloodGroup;

std::cout << "Резус фактор: ";

std::cin >> newPatient.rhFactor;

std::cout << "Диагноз: ";

std::cin >> newPatient.diagnosis;

patients.push\_back(newPatient);

writeToFile(patients);

}

//Чтение из файла

std::vector<Patient> readFromFile() {

std::ifstream file("patients.txt");

std::vector<Patient> patients;

if (file.is\_open()) {

Patient patient;

while (file >> patient.lastName >> patient.firstName >> patient.middleName >> patient.birthYear >> patient.phoneNumber >> patient.address >> patient.medicalCardNumber >> patient.bloodGroup >> patient.rhFactor >> patient.diagnosis) {

patients.push\_back(patient);

}

file.close();

}

else {

std::cout << "Не удалось открыть файл" << std::endl;

}

return patients;

}

//Компараторы для sort

bool compare1(const Patient& a, const Patient& b) {

return a.lastName > b.lastName;

}

bool compare2(const Patient& a, const Patient& b) {

return a.firstName > b.firstName;

}

bool compare3(const Patient& a, const Patient& b) {

return a.middleName > b.middleName;

}

bool compare4(const Patient& a, const Patient& b) {

return a.birthYear > b.birthYear;

}

bool compare5(const Patient& a, const Patient& b) {

return a.phoneNumber > b.phoneNumber;

}

bool compare6(const Patient& a, const Patient& b) {

return a.address > b.address;

}

bool compare7(const Patient& a, const Patient& b) {

return a.medicalCardNumber > b.medicalCardNumber;

}

bool compare8(const Patient& a, const Patient& b) {

return a.bloodGroup > b.bloodGroup;

}

bool compare9(const Patient& a, const Patient& b) {

return a.rhFactor > b.rhFactor;

}

bool compare10(const Patient& a, const Patient& b) {

return a.diagnosis > b.diagnosis;

}

//Сортировка

void sorting(std::vector<Patient>& patients) {

std::cout << "Выберите элемент сортировки:\n";

std::cout << "1 - Фамилия\n2 - Имя\n3 - Отчество\n4 - Год рождения\n5 - Номер телефона\n6 - Адресс\n7 - Номер медицинской карты\n8 - Группа крови\n9 - Резус фактор\n10 - Диагноз\n";

std::cout << "Напишите цифру: ";

int sr;

bool prnt = true;

std::cin >> sr;

system("cls");

std::cout << "Результаты сортировки:\n";

switch (sr) {

case 1:

std::sort(patients.begin(), patients.end(), compare1);

break;

case 2:

std::sort(patients.begin(), patients.end(), compare2);

break;

case 3:

std::sort(patients.begin(), patients.end(), compare3);

break;

case 4:

std::sort(patients.begin(), patients.end(), compare4);

break;

case 5:

std::sort(patients.begin(), patients.end(), compare5);

break;

case 6:

std::sort(patients.begin(), patients.end(), compare6);

break;

case 7:

std::sort(patients.begin(), patients.end(), compare7);

break;

case 8:

std::sort(patients.begin(), patients.end(), compare8);

break;

case 9:

std::sort(patients.begin(), patients.end(), compare9);

break;

case 10:

std::sort(patients.begin(), patients.end(), compare10);

break;

default:

std::cout << "Неизвестный код";

prnt = false;

break;

}

if (prnt) {

writeToConsole(patients);

}

}

//Поиск

void finding(std::vector<Patient>& patients) {

std::cout << "Выберите элемент поиска:\n";

std::cout << "1 - Фамилия\n2 - Имя\n3 - Отчество\n4 - Год рождения\n5 - Номер телефона\n6 - Адресс\n7 - Номер медицинской карты\n8 - Группа крови\n9 - Резус фактор\n10 - Диагноз\n";

std::cout << "Напишите цифру: ";

int fn;

std::string src;

std::cin >> fn;

std::cout << "Напишите слово поиска: ";

std::cin >> src;

system("cls");

std::cout << "Результаты поиска:\n";

switch (fn) {

case 1:

for (Patient patient : patients) {

if (patient.lastName == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 2:

for (Patient patient : patients) {

if (patient.firstName == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 3:

for (Patient patient : patients) {

if (patient.middleName == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 4:

for (Patient patient : patients) {

if (patient.birthYear == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 5:

for (Patient patient : patients) {

if (patient.phoneNumber == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 6:

for (Patient patient : patients) {

if (patient.address == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 7:

for (Patient patient : patients) {

if (patient.medicalCardNumber == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 8:

for (Patient patient : patients) {

if (patient.bloodGroup == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 9:

for (Patient patient : patients) {

if (patient.rhFactor == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

case 10:

for (Patient patient : patients) {

if (patient.diagnosis == src) {

std::cout << patient.lastName << "\t" << patient.firstName << "\t" << patient.middleName << "\t" << patient.birthYear << "\t" << patient.phoneNumber << "\t" << patient.address << "\t" << patient.medicalCardNumber << "\t" << patient.bloodGroup << "\t" << patient.rhFactor << "\t" << patient.diagnosis << "\n";

}

}

break;

default:

std::cout << "Неизвестный код";

break;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "RUS");

std::vector<Patient> Patients = readFromFile();

char ch;

std::cout << "а) Добавление пациентов.\n";

std::cout << "б) Поиск структуры с заданным значением выбранного элемента.\n";

std::cout << "в) Вывод на экран содержимого массива структур.\n";

std::cout << "г) Сортировка массива структур по заданному полю.\n";

std::cout << "Напишите букву : ";

ch = fgetc(stdin);

if (ch == 'а') {

int ps;

system("cls");

std::cout << "Добавление пациентов.\nСколько пациентов хотите добавить: ";

std::cin >> ps;

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

readFromConsole(Patients);

std::cout << "Добавлен " << i + 1 << " пациент\n";

}

}

else if (ch == 'б') {

system("cls");

finding(Patients);

}

else if (ch == 'в') {

system("cls");

std::cout << "Массив структур:\n";

writeToConsole(Patients);

}

else if (ch == 'г') {

system("cls");

sorting(Patients);

}

else {

std::cout << "Неизвестный символ\n";

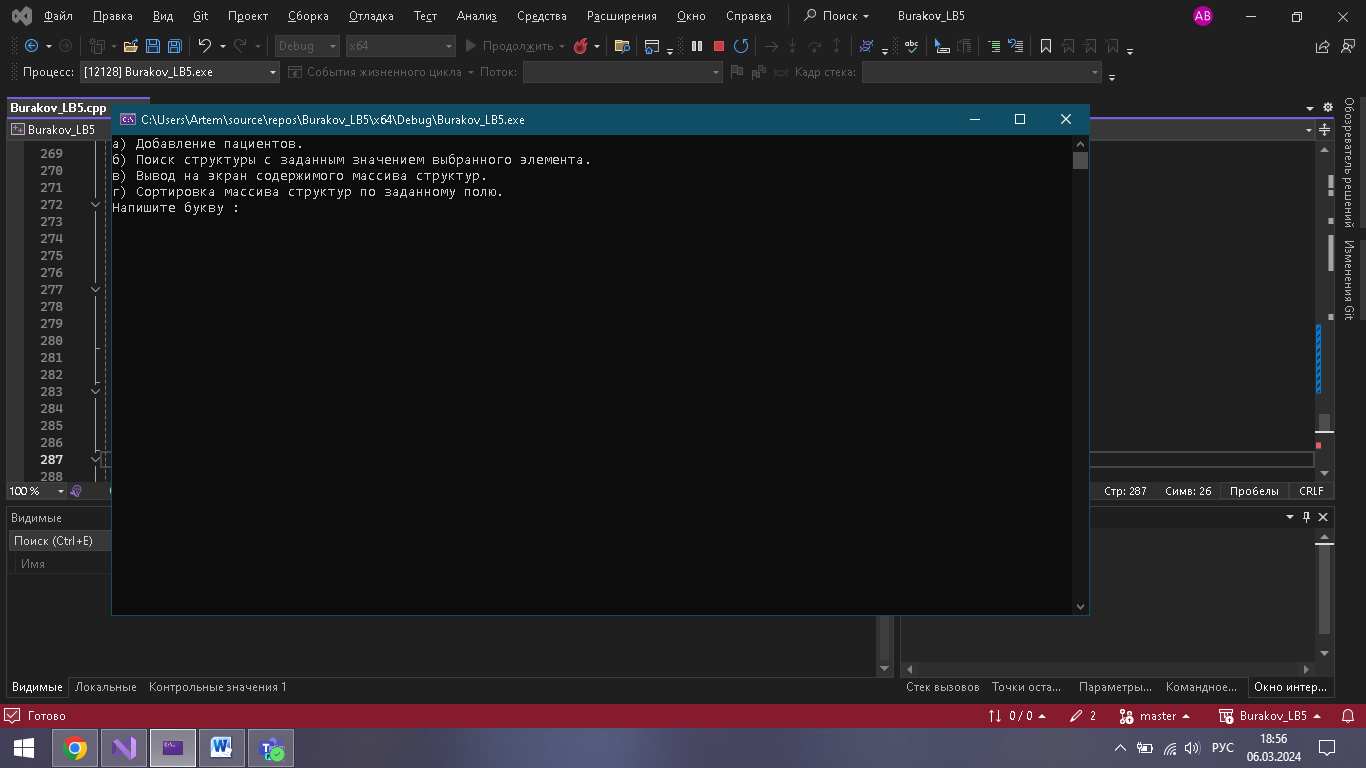
}

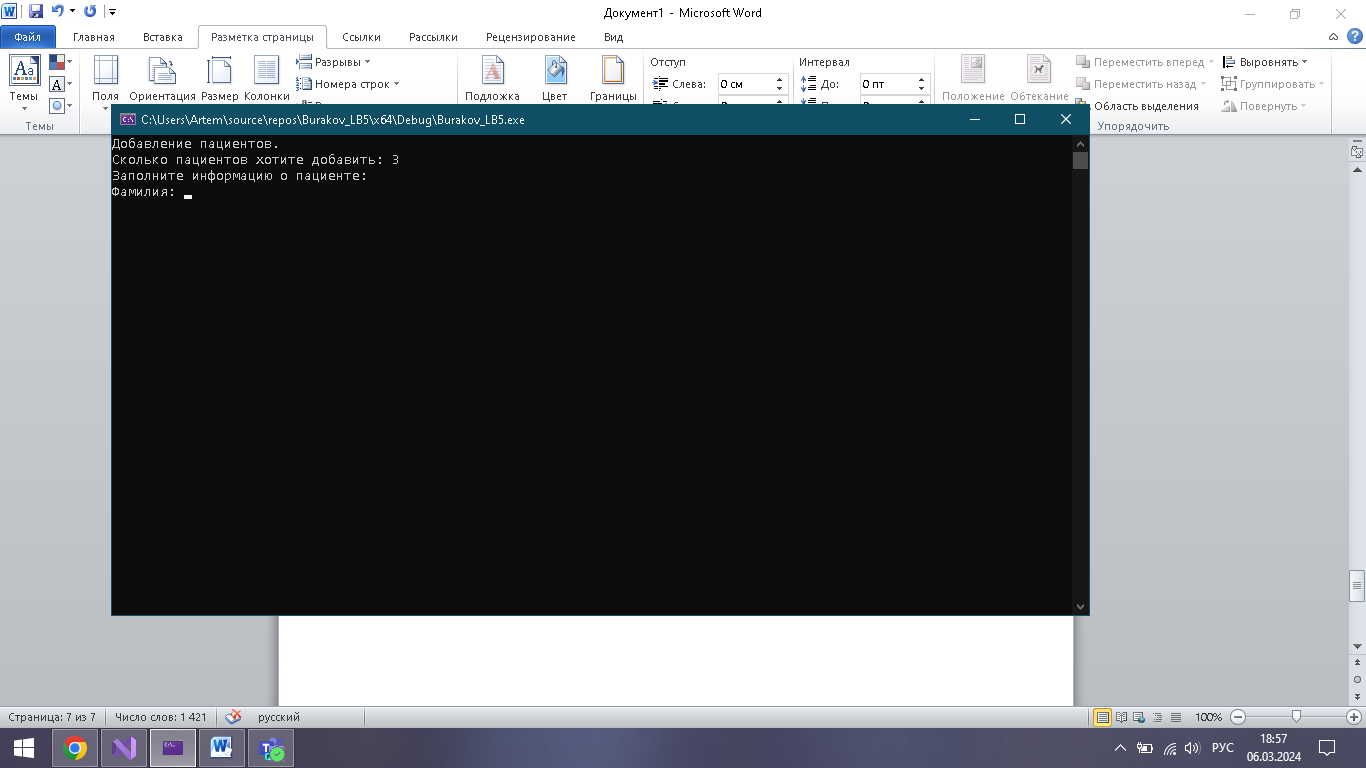
system("pause");

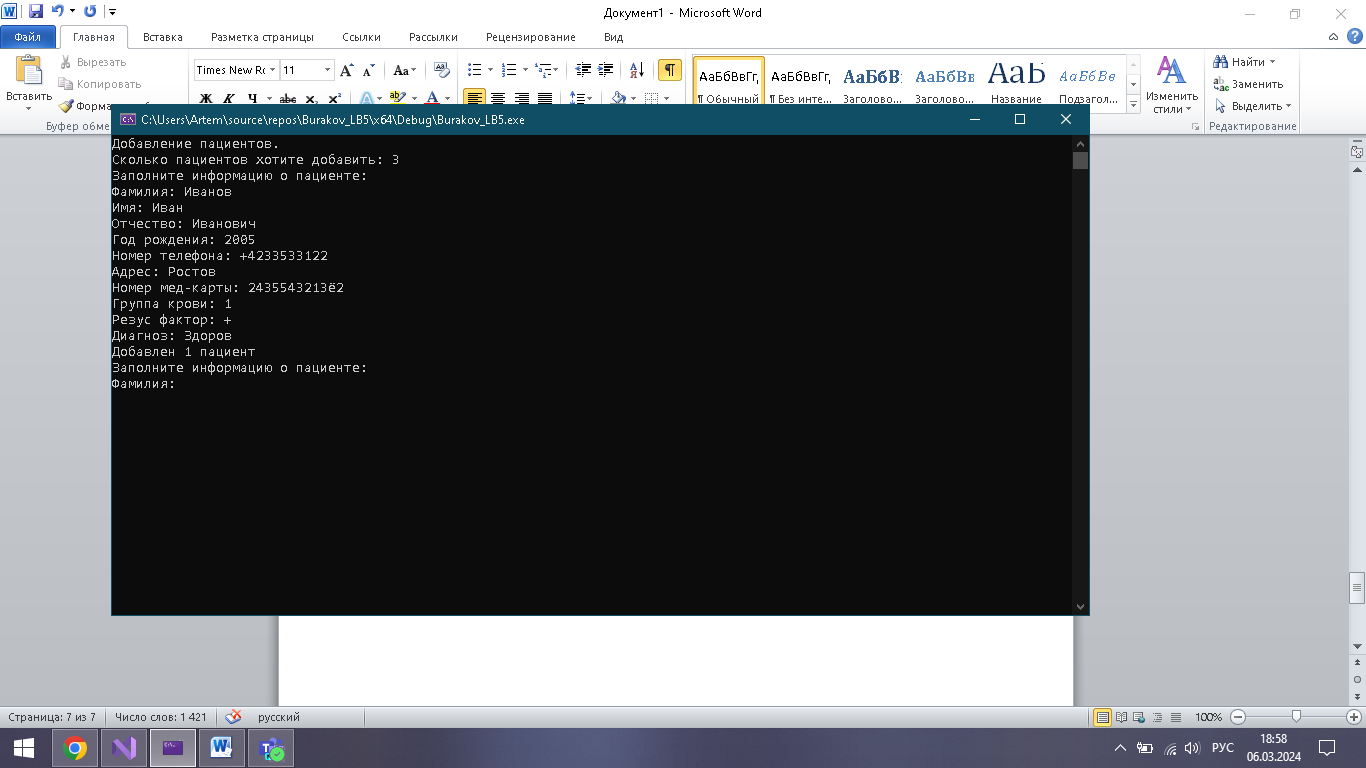
return 0;

}

**Результат работы программы.**







**Вывод:** Научился работать соструктурами, закрепил умения работы с файлами, алгоритмами поиска и сортировки.