|  |  |
| --- | --- |
| Пермский национальный исследовательский политехнический университет  Кафедра информационные технологии и автоматизированные системы | |
|  |  |
|  |  |
| Лабораторная работа №8  "Блоковый ввод-вывод"  Вариант №9 | |
|  |  |
|  | Выполнил:  Студент группы ИВТ-24-2б  А.С. Ваулин  Проверил:  Доцент кафедры ИТАС  О.А. Полякова |
|  |  |
| Пермь 2025 | |

# Постановка задачи

Сформировать двоичный файл из элементов, заданной в варианте структуры, распечатать его содержимое, выполнить удаление и добавление элементов в соответствии со своим вариантом, используя для поиска удаляемых или добавляемых элементов функцию. Формирование, печать, добавление и удаление элементов оформить в виде функций. Предусмотреть сообщения об ошибках при открытии файла и выполнении операций ввода/вывода.

1. Структура "Пациент":

* фамилия, имя, отчество;
* домашний адрес;
* номер медицинской карты;
* номер страхового полиса.

Удалить элемент с заданным номером медицинской карты, добавить 2 элемента в начало файла.

# Блок-схема

начало

struct Patient

#ifndef PATIENT\_H

#define PATIENT\_H

#include <iostream>

#include <fstream>

#include <windows.h>

// Константы для максимальной длины полей

const int MAX\_NAME\_LENGTH = 50;

const int MAX\_ADDRESS\_LENGTH = 100;

char lastName[MAX\_NAME\_LENGTH];

char firstName[MAX\_NAME\_LENGTH];

char patronymic[MAX\_NAME\_LENGTH];

char address[MAX\_ADDRESS\_LENGTH];

int medicalCardNumber;

long long insuranceNumber;

Patient\* InputPatient();

void WritePatient(std::ofstream& file, const Patient& patient);

bool ReadPatient(std::ifstream& file, Patient& patient);

α

{

}

;

α

bool CreateInitialFile(const char\* filename);

void PrintFile(const char\* filename);

bool DeletePatient(const char\* filename, int medicalCardNumber);

bool AddTwoPatientsToStart(const char\* filename);

#endif

Patient\* InputPatient()

Patient\* patient = new Patient;

#include "Patient.h"

using namespace std;

cout << "Введите фамилию: ";

cin.getline(patient->lastName, MAX\_NAME\_LENGTH);

β

{

β

cout << "Введите имя: ";

cin.getline(patient->firstName, MAX\_NAME\_LENGTH);

cout << "Введите отчество: ";

cin.getline(patient->patronymic, MAX\_NAME\_LENGTH);

cout << "Введите домашний адрес: ";

cin.getline(patient->address, MAX\_ADDRESS\_LENGTH);

cout << "Введите номер медицинской карты: ";

cin >> patient->medicalCardNumber;

cin.ignore(10000, '\n');

cout << "Введите номер страхового полиса: ";

cin >> patient->insuranceNumber;

γ

γ

cin >> patient->insuranceNumber;

cin.ignore(10000, '\n');

return patient;

}

void WritePatient(ofstream& file, const Patient& patient)

file.write(patient.lastName, MAX\_NAME\_LENGTH);

file.write(patient.firstName, MAX\_NAME\_LENGTH);

file.write(patient.patronymic, MAX\_NAME\_LENGTH);

file.write(patient.address, MAX\_ADDRESS\_LENGTH);

file.write(reinterpret\_cast<const char\*>(&patient.medicalCardNumber), sizeof(patient.medicalCardNumber));

file.write(reinterpret\_cast<const char\*>(&patient.insuranceNumber), sizeof(patient.insuranceNumber));

return;

{

}

bool ReadPatient(ifstream& file, Patient& patient)

if (!file.read(patient.lastName, MAX\_NAME\_LENGTH))

return false;

if (!file.read(patient.firstName, MAX\_NAME\_LENGTH))

return false;

if (!file.read(patient.patronymic, MAX\_NAME\_LENGTH))

return false;

if (!file.read(patient.address, MAX\_ADDRESS\_LENGTH))

return false;

δ

1

0

1

0

1

0

1

0

{

δ

if (!file.read(reinterpret\_cast<char\*>(&patient.medicalCardNumber),

sizeof(patient.medicalCardNumber)))

return false;

if (!file.read(reinterpret\_cast<char\*>(&patient.insuranceNumber),

sizeof(patient.insuranceNumber)))

return false;

return true;

1

0

1

0

}

bool CreateInitialFile(const char\* filename)

ofstream file(filename, ios::binary);

if (!file.is\_open())

cerr << "Ошибка: Не удалось создать файл " << filename << endl;

{

return false;

cout << "Введите данные для 3 пациентов:\n";

for (int i = 1; i <= 3; ++i)

cout << "\nПациент " << i << ":\n";

Patient\* patient = InputPatient();

ε

ζ

file.close();

return true;

}

1

0

{

}

{

1

0

ε

if (patient)

WritePatient(file, \*patient);

delete patient;

ζ

}

1

0

{

}

void PrintFile(const char\* filename)

ifstream file(filename, ios::binary);

if (!file.is\_open())

cerr << "Ошибка: Не удалось открыть файл " << filename << " для чтения" << endl;

return;

cout << "\nСодержимое файла " << filename << ":\n";

η

{

1

0

{

}

η

cout << "----------------------------------------\n";

Patient patient;

int patientCount = 0;

while (ReadPatient(file, patient))

patientCount++;

cout << "Пациент " << patientCount << ":\n";

cout << "ФИО: " << patient.lastName << " "

<< patient.firstName << " "

<< patient.patronymic << endl;

cout << "Адрес: " << patient.address << endl;

cout << "Номер мед. карты: " <<

patient.medicalCardNumber << endl;

cout << "Номер страховки: " << patient.insuranceNumber << endl;

cout << "----------------------------------------\n";

θ

{

}

1

0

θ

if (patientCount == 0)

cout << "Файл пуст\n";

file.close();

bool DeletePatient(const char\* filename, int medicalCardNumber)

const char\* tempFilename = "temp.dat";

ifstream inFile(filename, ios::binary);

ofstream outFile(tempFilename, ios::binary);

if (!inFile.is\_open() || !outFile.is\_open())

cerr << "Ошибка: Не удалось открыть файлы для

операции удаления" << endl;

return false;

ί

}

return;

1

0

{

1

0

{

}

ί

bool found = false;

Patient patient;

while (ReadPatient(inFile, patient))

if (patient.medicalCardNumber != medicalCardNumber)

WritePatient(outFile, patient);

found = true;

inFile.close();

outFile.close();

κ

1

{

}

0

1

0

κ

if (remove(filename) != 0)

cerr << "Ошибка при удалении исходного файла " << filename << endl;

return false;

if (rename(tempFilename, filename) != 0)

cerr << "Ошибка при переименовании временного файла в " << filename << endl;

return false;

return found;

}

1

0

{

}

1

0

{

}

bool AddTwoPatientsToStart(const char\* filename)

const char\* tempFilename = "temp.dat";

ofstream tempFile(tempFilename, ios::binary);

if (!tempFile.is\_open())

cerr << "Ошибка: Не удалось создать временный файл" << endl;

return false;

cout << "\nДобавление первого пациента:\n";

Patient\* patient1 = InputPatient();

if (patient1)

WritePatient(tempFile, \*patient1);

delete patient1;

λ

{

1

0

{

}

1

0

{

}

λ

cout << "\nДобавление второго пациента:\n";

Patient\* patient2 = InputPatient();

if (patient2)

WritePatient(tempFile, \*patient2);

delete patient2;

ifstream originalFile(filename, ios::binary);

if (originalFile.is\_open())

Patient patient;

while (ReadPatient(originalFile, patient))

WritePatient(tempFile, patient);

originalFile.close();

ξ

1

0

{

}

1

0

{

}

1

0

ξ

tempFile.close();

if (remove(filename) != 0)

cerr << "Ошибка при удалении исходного файла " << filename << endl;

return false;

if (rename(tempFilename, filename) != 0)

cerr << "Ошибка при переименовании временного файла в " << filename << endl;

return false;

return true;

}

1

0

{

}

1

0

{

}

int main()

#include "Patient.h"

SetConsoleOutputCP(CP\_UTF8);

SetConsoleCP(CP\_UTF8);

const char\* filename = "patients.dat";

if (!CreateInitialFile(filename))

return 1;

PrintFile(filename);

int cardToDelete;

cout << "\nВведите номер медицинской карты для удаления: ";

cin >> cardToDelete;

*o*

{

1

0

*o*

if (DeletePatient(filename, cardToDelete))

cout << "Пациент успешно удален\n";

cout << "Пациент с таким номером карты не найден\n";

PrintFile(filename);

if (AddTwoPatientsToStart(filename))

cout << "Два пациента успешно добавлены в начало файла\n";

PrintFile(filename);

return 0;

1

0

1

0

}

# Код программы

**Patient.h**

#ifndef PATIENT\_H

#define PATIENT\_H

#include <iostream>

#include <fstream>

#include <windows.h>

// Константы для максимальной длины полей

const int MAX\_NAME\_LENGTH = 50;

const int MAX\_ADDRESS\_LENGTH = 100;

// Структура для хранения данных пациента

struct Patient

{

char lastName[MAX\_NAME\_LENGTH]; // Фамилия

char firstName[MAX\_NAME\_LENGTH]; // Имя

char patronymic[MAX\_NAME\_LENGTH]; // Отчество

char address[MAX\_ADDRESS\_LENGTH]; // Домашний адрес

int medicalCardNumber; // Номер медицинской карты

long long insuranceNumber; // Номер страхового полиса

};

// Прототипы функций

Patient\* InputPatient();

void WritePatient(std::ofstream& file, const Patient& patient);

bool ReadPatient(std::ifstream& file, Patient& patient);

bool CreateInitialFile(const char\* filename);

void PrintFile(const char\* filename);

bool DeletePatient(const char\* filename, int medicalCardNumber);

bool AddTwoPatientsToStart(const char\* filename);

#endif // PATIENT\_H

**Patient.cpp**

#include "Patient.h"

using namespace std;

// Функция ввода данных одного пациента с клавиатуры

Patient\* InputPatient()

{

Patient\* patient = new Patient;

cout << "Введите фамилию: ";

cin.getline(patient->lastName, MAX\_NAME\_LENGTH);

cout << "Введите имя: ";

cin.getline(patient->firstName, MAX\_NAME\_LENGTH);

cout << "Введите отчество: ";

cin.getline(patient->patronymic, MAX\_NAME\_LENGTH);

cout << "Введите домашний адрес: ";

cin.getline(patient->address, MAX\_ADDRESS\_LENGTH);

cout << "Введите номер медицинской карты: ";

cin >> patient->medicalCardNumber;

cin.ignore(10000, '\n');

cout << "Введите номер страхового полиса: ";

cin >> patient->insuranceNumber;

cin.ignore(10000, '\n');

return patient;

}

// Функция записи данных пациента в бинарный файл

void WritePatient(ofstream& file, const Patient& patient)

{

file.write(patient.lastName, MAX\_NAME\_LENGTH);

file.write(patient.firstName, MAX\_NAME\_LENGTH);

file.write(patient.patronymic, MAX\_NAME\_LENGTH);

file.write(patient.address, MAX\_ADDRESS\_LENGTH);

file.write(reinterpret\_cast<const char\*>(&patient.medicalCardNumber), sizeof(patient.medicalCardNumber));

file.write(reinterpret\_cast<const char\*>(&patient.insuranceNumber), sizeof(patient.insuranceNumber));

}

// Функция чтения данных пациента из бинарного файла

bool ReadPatient(ifstream& file, Patient& patient)

{

if (!file.read(patient.lastName, MAX\_NAME\_LENGTH)) return false;

if (!file.read(patient.firstName, MAX\_NAME\_LENGTH)) return false;

if (!file.read(patient.patronymic, MAX\_NAME\_LENGTH)) return false;

if (!file.read(patient.address, MAX\_ADDRESS\_LENGTH)) return false;

if (!file.read(reinterpret\_cast<char\*>(&patient.medicalCardNumber), sizeof(patient.medicalCardNumber))) return false;

if (!file.read(reinterpret\_cast<char\*>(&patient.insuranceNumber), sizeof(patient.insuranceNumber))) return false;

return true;

}

// Функция создания файла с начальными данными (3 пациента)

bool CreateInitialFile(const char\* filename)

{

ofstream file(filename, ios::binary);

if (!file.is\_open())

{

cerr << "Ошибка: Не удалось создать файл " << filename << endl;

return false;

}

cout << "Введите данные для 3 пациентов:\n";

for (int i = 1; i <= 3; ++i)

{

cout << "\nПациент " << i << ":\n";

Patient\* patient = InputPatient();

if (patient)

{

WritePatient(file, \*patient);

delete patient;

}

}

file.close();

return true;

}

// Функция вывода содержимого файла на экран

void PrintFile(const char\* filename)

{

ifstream file(filename, ios::binary);

if (!file.is\_open())

{

cerr << "Ошибка: Не удалось открыть файл " << filename << " для чтения" << endl;

return;

}

cout << "\nСодержимое файла " << filename << ":\n";

cout << "----------------------------------------\n";

Patient patient;

int patientCount = 0;

while (ReadPatient(file, patient))

{

patientCount++;

cout << "Пациент " << patientCount << ":\n";

cout << "ФИО: " << patient.lastName << " "

<< patient.firstName << " "

<< patient.patronymic << endl;

cout << "Адрес: " << patient.address << endl;

cout << "Номер мед. карты: " << patient.medicalCardNumber << endl;

cout << "Номер страховки: " << patient.insuranceNumber << endl;

cout << "----------------------------------------\n";

}

if (patientCount == 0)

{

cout << "Файл пуст\n";

}

file.close();

}

// Функция удаления пациента по номеру медицинской карты

bool DeletePatient(const char\* filename, int medicalCardNumber)

{

const char\* tempFilename = "temp.dat";

ifstream inFile(filename, ios::binary);

ofstream outFile(tempFilename, ios::binary);

if (!inFile.is\_open() || !outFile.is\_open())

{

cerr << "Ошибка: Не удалось открыть файлы для операции удаления" << endl;

return false;

}

bool found = false;

Patient patient;

while (ReadPatient(inFile, patient))

{

if (patient.medicalCardNumber != medicalCardNumber)

{

WritePatient(outFile, patient);

}

else

{

found = true;

}

}

inFile.close();

outFile.close();

if (remove(filename) != 0)

{

cerr << "Ошибка при удалении исходного файла " << filename << endl;

return false;

}

if (rename(tempFilename, filename) != 0)

{

cerr << "Ошибка при переименовании временного файла в " << filename << endl;

return false;

}

return found;

}

// Функция добавления двух пациентов в начало файла

bool AddTwoPatientsToStart(const char\* filename)

{

const char\* tempFilename = "temp.dat";

ofstream tempFile(tempFilename, ios::binary);

if (!tempFile.is\_open())

{

cerr << "Ошибка: Не удалось создать временный файл" << endl;

return false;

}

// Ввод и запись двух новых пациентов

cout << "\nДобавление первого пациента:\n";

Patient\* patient1 = InputPatient();

if (patient1)

{

WritePatient(tempFile, \*patient1);

delete patient1;

}

cout << "\nДобавление второго пациента:\n";

Patient\* patient2 = InputPatient();

if (patient2)

{

WritePatient(tempFile, \*patient2);

delete patient2;

}

// Копирование существующих данных из исходного файла

ifstream originalFile(filename, ios::binary);

if (originalFile.is\_open())

{

Patient patient;

while (ReadPatient(originalFile, patient))

{

WritePatient(tempFile, patient);

}

originalFile.close();

}

tempFile.close();

if (remove(filename) != 0)

{

cerr << "Ошибка при удалении исходного файла " << filename << endl;

return false;

}

if (rename(tempFilename, filename) != 0)

{

cerr << "Ошибка при переименовании временного файла в " << filename << endl;

return false;

}

return true;

}

**Main.cpp**

#include "Patient.h"

using namespace std;

int main()

{

SetConsoleOutputCP(CP\_UTF8);

SetConsoleCP(CP\_UTF8);

const char\* filename = "patients.dat";

// Создание файла с начальными данными

if (!CreateInitialFile(filename))

{

return 1;

}

PrintFile(filename);

// Удаление пациента по номеру медицинской карты

int cardToDelete;

cout << "\nВведите номер медицинской карты для удаления: ";

cin >> cardToDelete;

if (DeletePatient(filename, cardToDelete))

{

cout << "Пациент успешно удален\n";

}

else

{

cout << "Пациент с таким номером карты не найден\n";

}

PrintFile(filename);

// Добавление двух новых пациентов в начало

if (AddTwoPatientsToStart(filename))

{

cout << "Два пациента успешно добавлены в начало файла\n";

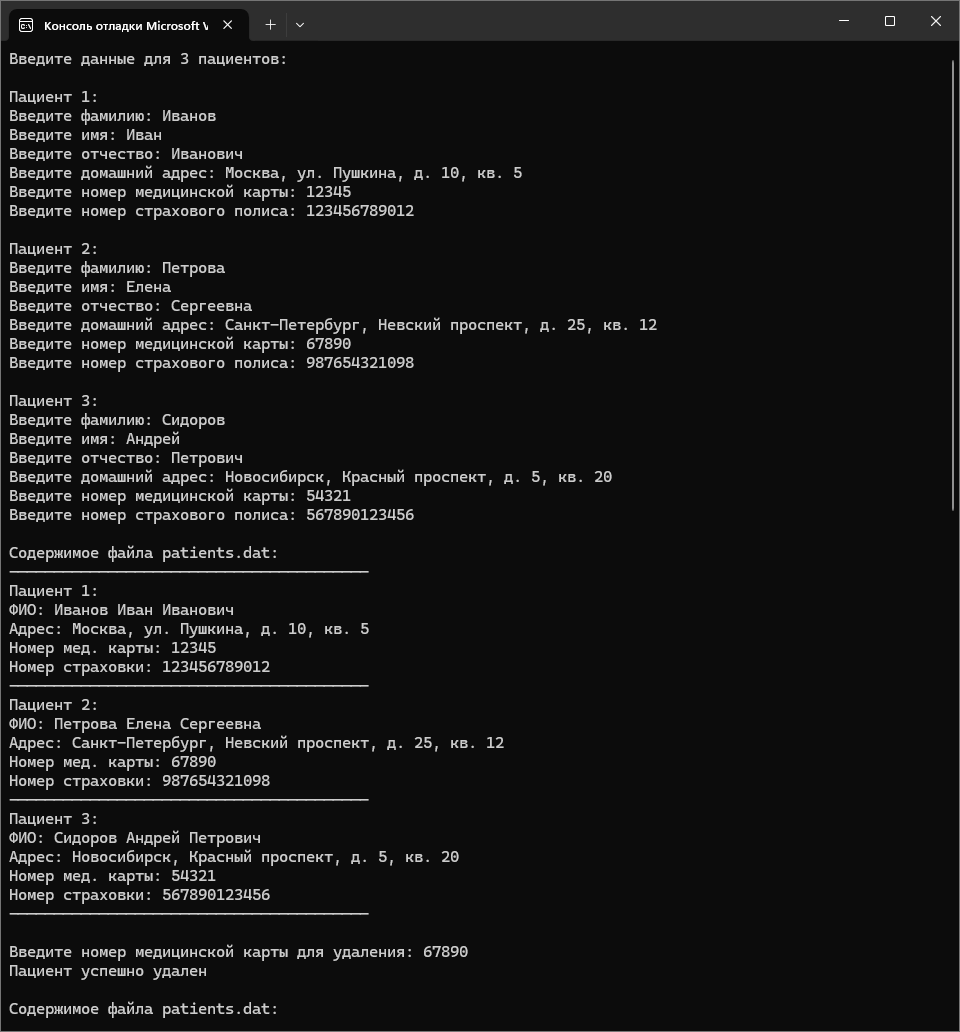
}

PrintFile(filename);

return 0;

}

# Результат



Git: <https://github.com/Artivaa/PatientBin>