



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего
образования

«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий
Кафедра математического обеспечения и стандартизации
информационных технологий

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2
«Управление файлом»
по дисциплине
«Структуры и алгоритмы обработки данных»

Выполнил студент группы *ИКБО-03-22*

Хохлинов Д.И.

Принял

Сорокин А.В.

Практическая
работа выполнена

«__»_____2023 г.

«Зачтено»

«__»_____2023 г.

Москва 2023

СОДЕРЖАНИЕ

ЗАДАНИЕ 1	3
1.1 Формулировка и требования.....	3
1.2 Тестовый пример.....	4
1.3 Реализация приложения	5
1.3.1 Функционал приложения	5
1.3.2 Реализация приложения	6
1.4 Результаты тестирования	11
ЗАДАНИЕ 2	15
2.1 Условие задания	15
2.1.1 Формулировка	15
2.1.2 Требования.....	15
2.1.3 Формулировка задания варианта	16
2.2 Тестовый пример.....	17
2.3 Реализация приложения	19
2.3.1 Структура используемой записи	19
2.3.2 Структура двоичного файла	19
2.2.3 Функционал приложения	20
2.4 Код программы.....	24
2.5 Тестирование	40
ЗАКЛЮЧЕНИЕ	45
ИНФОРМАЦИОННЫЙ ИСТОЧНИК	46

ЗАДАНИЕ 1

1.1 Формулировка и требования

Разработать программу, управления текстовым файлом.

Требования:

- Имя физического файла вводится пользователем и передается в функции обработки через параметр.
- При открытии файла выполнять контроль его существования и открытия. Примечание. При отладке программы можете имя физического файла определить через константу.
- Разработать функции для выполнения операций над текстовым файлом:
 - 1) создание текстового файла средствами текстового редактора кодировки ASCII, содержащего десятичные числа по несколько чисел на строке;
 - 2) вывод содержимого текстового файла;
 - 3) добавление новой записи в конец файла;
 - 4) прочесть значение числа, указав его порядковый номер в файле, и вернуть его значение;
 - 5) определить количество чисел в файле.
- Разработать приложение и выполнить тестирование всех функций. Приложение должно содержать диалоговый интерфейс на основе текстового меню.
- Контроль открытия и существования файла выполнить в основной программе перед вызовом функции. Перед закрытием файла, проверить отсутствие ошибок ввода и вывода (метод good)

- Создать модуль и перенести в него все отлаженные функции. Исключить функции из приложения. Отладить приложение, подключив к нему модуль с функциями.
- Разработать функции для реализации дополнительных операций, определенных вариантом и сохранить их в модуле с остальными функциями.
- Выполнить тестирование приложения в полном объеме.

В соответствии с вариантом (25) дополнительная операция имеет следующий алгоритм: создать новый файл из значений исходного, переписав в него только простые числа, располагая каждое на отдельной строке текстового файла.

1.2 Тестовый пример

Для тестирования программы использовался следующий файл (листинг 1):

Листинг 1. Содержание тестового файла

```
300 456 164 985
133 657
111 445 664 404 152 322
25 22 455
976
130
133 134 135
1024
-1023
307
311 313 317
```

1.3 Реализация приложения

1.3.1 Функционал приложения

Для выполнения задания использовались следующие методы модуля:

- 1) Метод открытия файла `openFile`. Принимает на вход строку `path`. Затем открывает файл, расположенный по пути `path`, в режиме чтения и записи. В случае неудачного открытия вызывает метод `closeFile` и возвращает «ложь», иначе возвращает «истину»;
- 2) Метод закрытия файла `closeFile`. Если ошибки во время работы с файлом не были обнаружены, то выводит соответствующее сообщение. Затем закрывает файл;
- 3) Метод повторного открытия файла `reopenFile`. Сначала закрывает файл, затем заново его открывает с помощью метода `openFile`;
- 4) Метод инициализации модуля `init`. Запрашивает у пользователя путь к рабочему файлу, затем пытается его открыть. Если инициализация успешна, возвращает «истину», иначе «ложь»;
- 5) Метод чтения содержимого `showContent`. Вызывает метод `reopenFile`, затем в случае успешного открытия считывает его содержимое и выводит его на экран;
- 6) Метод добавления записи `append`. Запрашивает у пользователя строку, которую необходимо записать в текущий файл, затем записывает ее. Если запись успешна, выводит соответствующее сообщение;
- 7) Метод поиска числа по индексу `readByIndex`. Принимает в качестве аргумента индекс `index`. Вызывает метод `reopenFile`; если файл не был успешно открыт, то возвращает соответствующий код ошибки. Если `index`-ое число существует в файле, то возвращает его; если нет, то возвращает код ошибки, сигнализирующий о том, что числа с таким индексом нет в файле;

- 8) Метод подсчета чисел в файле `numberOfNumbers`. Вызывает метод `reopenFile`; если файл не был успешно открыт, то возвращает соответствующий код ошибки. Возвращает количество чисел в файле;
- 9) Метод определения простоты числа `isPrimal`. Принимает в качестве аргумента целое число `n`. Если оно положительно и имеет только 2 делителя: 1 и `n`, то возвращает «истину», иначе – «ложь»;
- 10) Метод создания файла с простыми числами `createFileWithPrimals`. Вызывает метод `reopenFile`; если файл не был успешно открыт, то возвращает «ложь». Запрашивает у пользователя путь к файлу, куда будут сохранены простые числа; затем проходит по всем числам в исходном файле: если очередное число простое, то записывает его в новый файл с новой строки. В случае успешной записи выводит соответствующее сообщение и количество записанных чисел, а также возвращает «истину»; иначе возвращает «ложь».
- 11) Меню модуля `process`. Вызывает метод `init`; если он возвращает «ложь», то завершает работу. Затем в цикле предлагает пользователю одно из доступных действий. После выбора пользователем действия «выход» закрывает файл с помощью `closeFile`.

1.3.2 Реализация приложения

Запишем программу на языке C++ (листинги 2-4):

Листинг 2 – Файл `FileEditor.h`

```
#ifndef FileEditorH
#define FileEditorH

#include <iostream>
#include <fstream>
#include <string>
#include <conio.h>
#include <cmath>
```

Продолжение листинга 2

```
using namespace std;

namespace FileEditor {

    bool openFile(string path); //открыть файл
    void closeFile();
    bool reopenFile();

    bool showContent(); //содержимое
    void append(); //добавить запись в конец файла
    int readByIndex(int index); //прочитать число по индексу
    int numberOfNumbers(); //количество чисел
    bool createFileWithPrimals();

    void process(); //интерфейс для взаимодействия с файлом, который
    запускается из основной программы

    bool init(); //инициализация
    bool isPrimal(int n); //проверка простоты
}

#endif
```

Листинг 3 – Файл FileEditor.cpp

```
#ifndef FileEditorCpp
#define FileEditorCpp
#include "FileEditor.h"

int NO_SUCH_INDEX_CODE = 2147483647;
int FILE_READING_ERROR_CODE = 2147483646;
fstream file;
string _path = "";

bool FileEditor::openFile(string path)
{
    file = fstream(path.c_str(), ios::in | ios::out);
    if (!file)
    {
        closeFile();
        return false;
    }
    return true;
}

bool FileEditor::reopenFile()
{
    file.close();
    return openFile(_path);
}

void FileEditor::closeFile()
{
    if (file.good())
    {
```

Продолжение листинга 3

```
        cout << "Ошибка ввода/вывода в файл не обнаружено." << endl;
    }
    file.close();
}

bool FileEditor::init()
{
    string path;
    cout << "Введите путь к файлу: ";
    getline(cin, path);
    _path = path;
    return openFile(path);
}

bool FileEditor::showContent()
{
    if (!reopenFile())
    {
        cout << "Ошибка при чтении файла" << endl;
        return false;
    }
    cout << "Содержимое файла:" << endl;
    while (!file.eof())
    {
        string temp;
        getline(file, temp);
        cout << temp << endl;
    }
    return true;
}

void FileEditor::append()
{
    file.seekg(0, ios::end);
    cout << "Введите новую строку файла: ";
    string temp;
    getline(cin, temp);
    file << "\n" << temp;
    if (!file.fail())
        cout << "Запись произведена успешно." << endl;
}

int FileEditor::readByIndex(int index)
{
    if (index > numberOfNumbers())
    {
        cout << "Числа с индексом " << index << " не существует" <<
endl;
        return NO_SUCH_INDEX_CODE;
    }
    if (!reopenFile())
    {
        cout << "Ошибка при чтении файла" << endl;
        return FILE_READING_ERROR_CODE;
    }
}
```


Продолжение листинга 3

```
int res = 0;
for (int i = 0; i < index; i++)
{
    file >> res;
}
return res;
}

int FileEditor::numberOfNumbers()
{
    if (!reopenFile())
    {
        cout << "Ошибка при чтении файла" << endl;
        return FILE_READING_ERROR_CODE;
    }
    int temp;
    int amount = 0;
    while (!file.eof())
    {
        file >> temp;
        amount++;
    }
    return amount;
}

bool FileEditor::isPrimal(int n)
{
    if (n <= 0) return false;
    for (int i = 2; i*i <= n; i++)
    {
        if ((n % i) == 0) return false;
    }
    return true;
}

bool FileEditor::createFileWithPrimals()
{
    if (!reopenFile())
    {
        cout << "Ошибка при чтении файла" << endl;
        return false;
    }
    int temp = 0;
    int amount = 0;
    string path;
    cout << "Введите название файла, куда необходимо сохранить
данные: ";
    getline(cin, path);
    fstream newfile = fstream(path.c_str(), ios::out);
    if (!newfile)
    {
        cout << "Не удалось создать файл" << endl;
    }
    else
    {

```

Продолжение листинга 3

```
        while (!file.eof())
        {
            file >> temp;
            if (isPrimal(temp))
            {
                if (amount) newfile << "\n";
                newfile << temp;
                amount++;
            }
        }
        cout << "Создан новый файл; записано простых чисел: " <<
amount << endl;
    }
    return true;
}

void FileEditor::process()
{
    if (!init())
    {
        cout << "Не удалось открыть файл" << endl;
        return;
    }
    char choose = ' ';
    bool running = true;
    int ind = 0;
    int res;
    while (running)
    {
        cout << "Выберите действие: " << endl <<
            "[R] - вывести содержимое файла" << endl <<
            "[A] - вывести количество чисел в файле" << endl <<
            "[F] - найти число с заданным номером" << endl <<
            "[W] - добавить запись в конец файла" << endl <<
            "[N] - создать новый файл, записав в него только
простые числа из исходного" << endl <<
            "[Q] - выход" << endl;
        choose = _getch();
        switch (tolower(choose))
        {
            case 'r':
                if (!showContent()) running = false;
                break;
            case 'a':
                if (!numberOfNumbers()) running = false;
                else { cout << "Количество чисел в файле: " <<
numberOfNumbers() << endl; }
                break;
            case 'f':
                cout << "Введите номер числа для поиска: ";
                cin >> ind;
                cin.ignore(32768, '\n');
                res = readByIndex(ind);
                if (res == NO_SUCH_INDEX_CODE)
                {
```

Продолжение листинга 3

```
    }
    else if (res == FILE_READING_ERROR_CODE)
        running = false;
    else
    {
        cout << "Число с номером " << ind << ": " << res
<< endl;
    }
    break;
case 'w':
    append();
    break;
case 'n':
    createFileWithPrimals();
    break;
case 'q':
    running = false;
    break;
}
if (running)
{
    system("pause");
    system("cls");
}
}
closeFile();
}
#endif
```

Листинг 4 – Файл main.cpp

```
#include <iostream>
#include <locale.h>
#include "FileEditor.h"

int main()
{
    setlocale(LC_ALL, "ru");
    FileEditor::process();
}
```

1.4 Результаты тестирования

Результаты тестирования приведены на рисунках 1-7:

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_1\Debug\siaod_p2_2_1.exe
Введите путь к файлу: file.txt
Выберите действие:
[R] - вывести содержимое файла
[A] - вывести количество чисел в файле
[F] - найти число с заданным номером
[W] - добавить запись в конец файла
[N] - создать новый файл, записав в него только простые числа из исходного
[Q] - выход
Содержимое файла:
300 456 164 985
133 657
111 445 664 404 152 322
25 22 455
976
130
133 134 135
1024
-1023
307
311 313 317
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1 – Вывод содержимого файла

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_1\Debug\siaod_p2_2_1.exe
Выберите действие:
[R] - вывести содержимое файла
[A] - вывести количество чисел в файле
[F] - найти число с заданным номером
[W] - добавить запись в конец файла
[N] - создать новый файл, записав в него только простые числа из исходного
[Q] - выход
Количество чисел в файле: 26
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2 – Количество чисел в файле

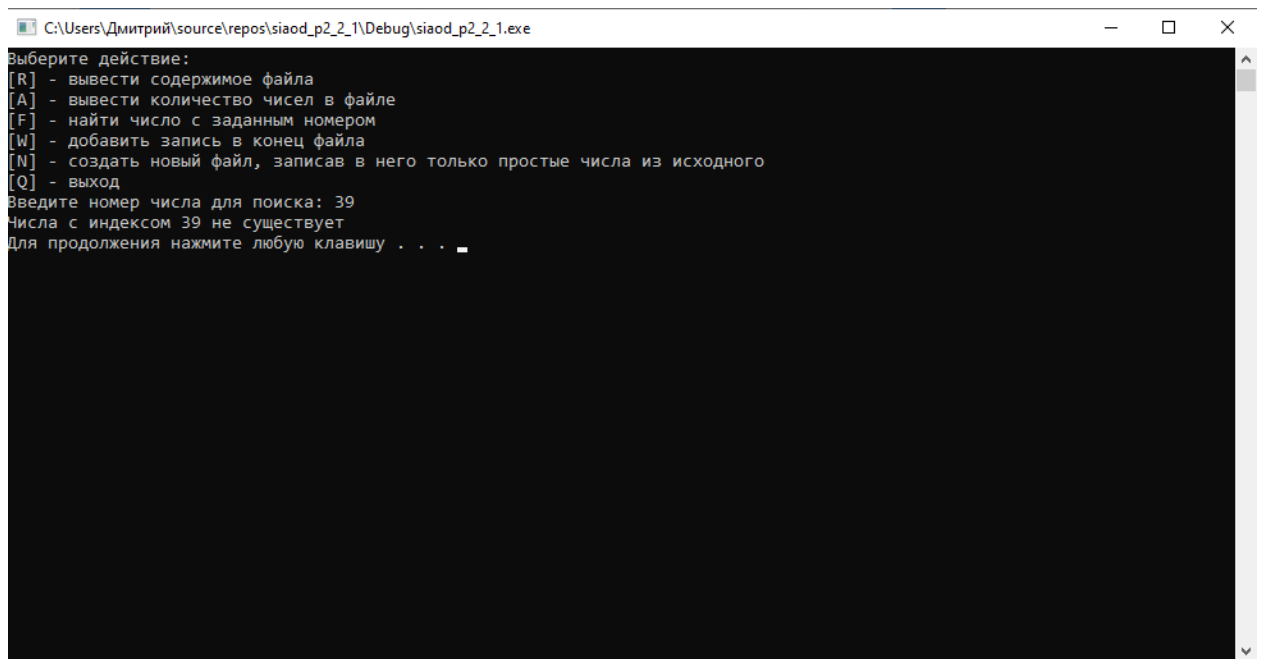


Рисунок 3 – Попытка поиска числа с несуществующим индексом

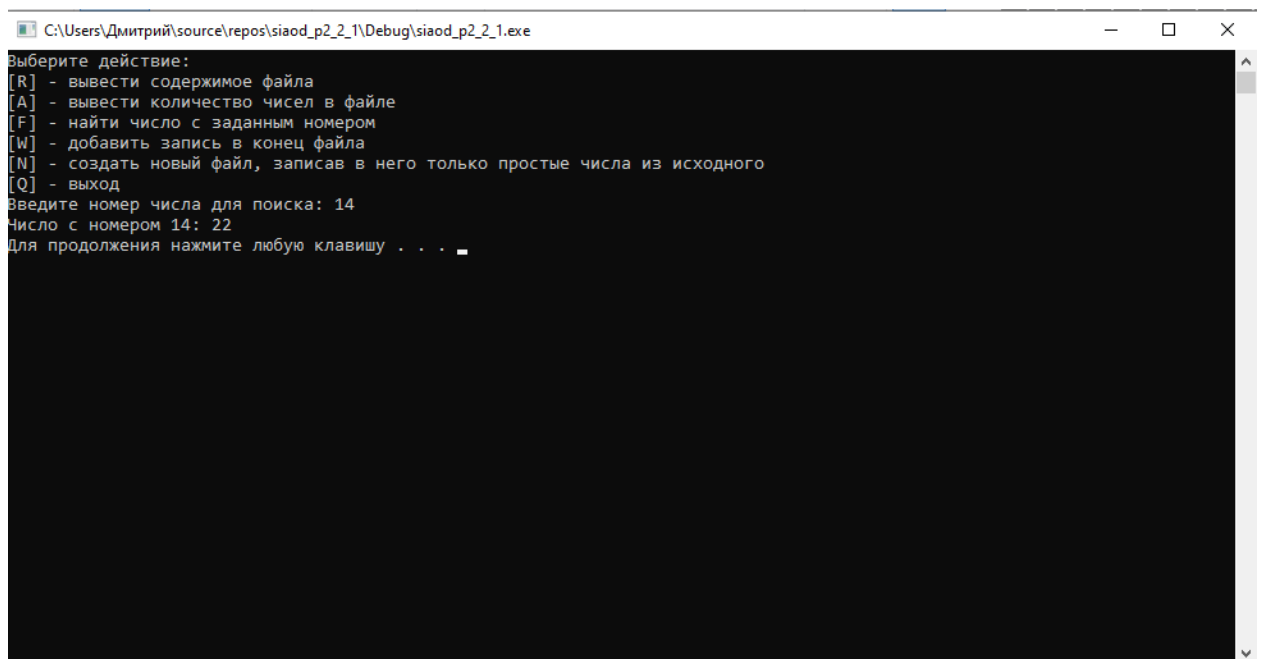


Рисунок 4 – Поиск числа с номером 14

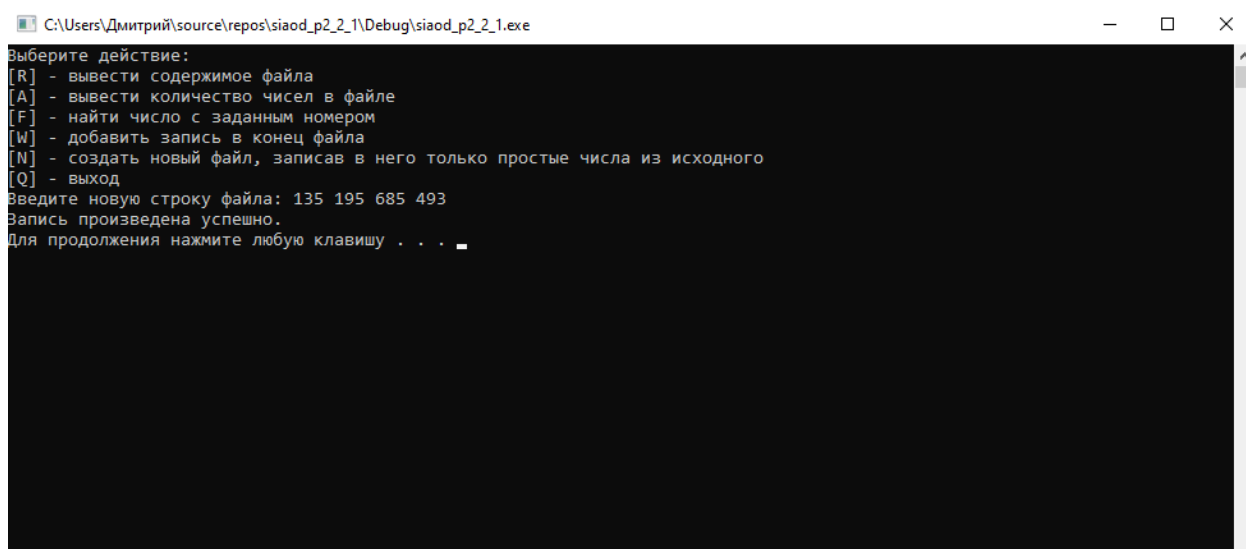


Рисунок 5 – Запись новой строки в файл

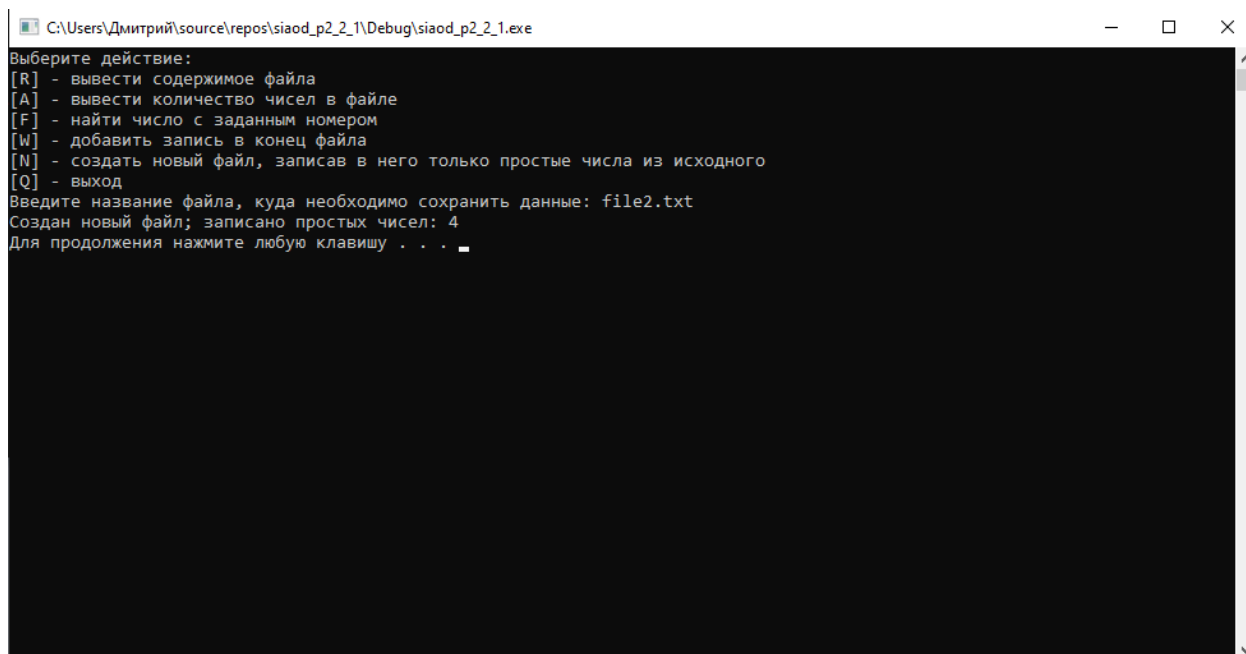


Рисунок 6 – Создание нового файла с простыми числами



Рисунок 7 – Содержимое нового файла

ЗАДАНИЕ 2

2.1 Условие задания

2.1.1 Формулировка

Разработать программу управление двоичными файлами с записями фиксированной длины. Файл состоит из записей определенной структуры, согласно варианту. Записи имеют ключ, уникальный в пределах файла.

2.1.2 Требования

1) Разработать структуру записи двоичного файла согласно варианту задания.

2) Подготовить тестовые данные в текстовом файле с кодировкой ASCII, в соответствии со структурой записи варианта. При открытии файла выполнить контроль его существования и открытия. Примечание. Реализация операций по чтению данных из файла будет проще, если значение для каждого поля записи размещать на отдельной строке текстового редактора.

3) Имя файла вводит пользователь.

4) При открытии файла обеспечить контроль существования и открытия файла.

5) При применении механизма прямого доступа к записи файла выполнить контроль присутствия записи с заданным номером в файле.

6) Разработать функции для выполнения операций:

– преобразование тестовых данных из текстового файла в двоичный файл;

– сохранение данных двоичного файла в текстовом, так, чтобы используя их можно было восстановить двоичный файл;

– вывод всех записей двоичного файла;

- доступ к записи по ее порядковому номеру в файле, используя механизм прямого доступа к записи в двоичном файле;

- удаление записи с заданным значением ключа, выполнить путем замены на последнюю запись.

- манипулирование записями в двоичном файле согласно дополнительным операциям, определенным в варианте 25, а именно:

- сформировать список пациентов, которым поставлен диагноз с заданным кодом заболевания в новом двоичном файле;

- удалить сведения о пациенте с заданным ключом, сохраняя порядок следования остальных записей.

7) Сохраните функции в новом модуле.

8) Разработать приложение, демонстрирующее выполнение всех операций, подключив к нему модуль с функциями.

9) Выполнить тестирование приложения, продемонстрировав выполнение всех операций.

2.1.3 Формулировка задания варианта

Согласно варианту 25, используемая структура данных имеет следующие поля: номер полиса (ключ), фамилия, имя, отчество, код заболевания, дата установки диагноза, код врача.

Дополнительные операции:

- сформировать список пациентов, которым поставлен диагноз с заданным кодом заболевания в новом двоичном файле;

- удалить сведения о пациенте с заданным ключом, сохраняя порядок следования остальных записей.

2.2 Тестовый пример

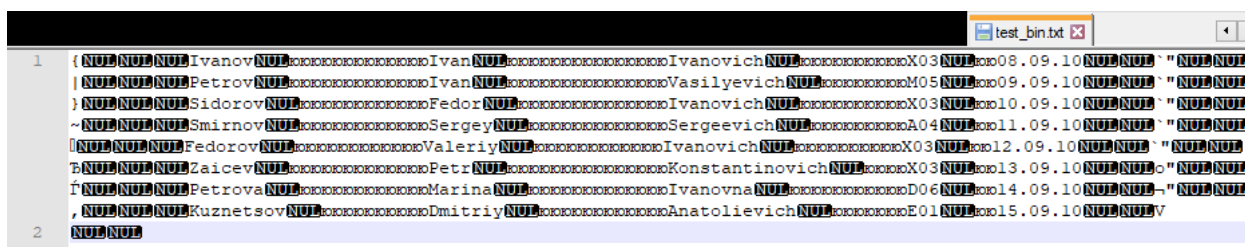
Для тестирования приложения использовался следующий текстовый файл (листинг 5) и полученный на его основе двоичный файл (рисунок 8):

Листинг 5 – Файл test.txt

```
123
Ivanov
Ivan
Ivanovich
X03
08.09.10
8800
124
Petrov
Ivan
Vasilyevich
M05
09.09.10
8800
125
Sidorov
Fedor
Ivanovich
X03
10.09.10
8800
126
Smirnov
Sergey
Sergeevich
A04
11.09.10
8800
127
Fedorov
```

Продолжение листинга 5

Valeriy
Ivanovich
X03
12.09.10
8800
128
Zaicev
Petr
Konstantinovich
X03
13.09.10
8815
129
Petrova
Marina
Ivanovna
D06
14.09.10
8876
130
Kuznetsov
Dmitriy
Anatolievich
E01
15.09.10
2646



```
1 { "name": "Ivanov Ivanovich", "date": "08.09.10", "other": "Ivanov Ivanovich" }
2 { "name": "Kuznetsov Dmitriy Anatolievich", "date": "15.09.10", "other": "Kuznetsov Dmitriy Anatolievich" }
```

Рисунок 8 – Содержимое файла test_bin.txt

2.3 Реализация приложения

2.3.1 Структура используемой записи

Для выполнения задания использовалась следующая структура:

```
struct record {  
    int oms_code; //ключ  
    char surname[20]; //фамилия  
    char name[20]; //имя  
    char patronymic[20]; //отчество  
    char disease_code[6]; //код болезни  
    char date[9]; //дата ДД.ММ.ГГ  
    int doctor_id; //код врача  
}
```

Количество памяти, занимаемое полями этой структуры:

- oms_code – 4 байта;
- surname, name, patronymic – по $1 * 20 = 20$ байт;
- disease_code – 6 байт;
- date – 9 байт;
- doctor_id – 4 байта.

Итого один экземпляр занимает 83 байта. Проверим это системно (рисунок 9):

Размер структуры в байтах: 84

Рисунок 9 – Системное определение количества байт, занятых структурой

Расхождение в результатах обусловлено особенностями системы: количество выделяемых байт должно делиться на 4.

2.3.2 Структура двоичного файла

Структура двоичного файла, используемого для хранения записей, изображена на рисунке 10.

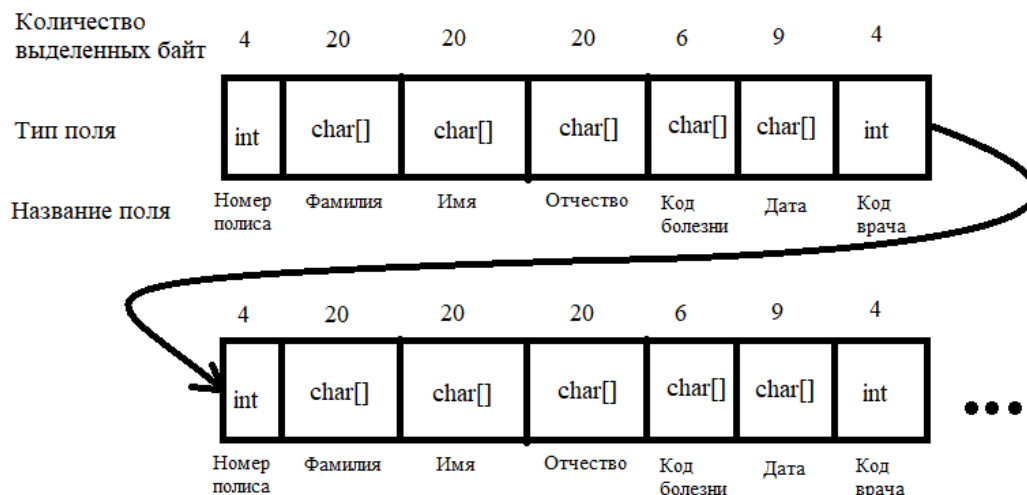


Рисунок 10 – Структура используемого двоичного файла

2.2.3 Функционал приложения

Для выполнения задания использовались следующие функции и модули:

1) Структура record:

- a. Конструктор по умолчанию: заполняет структуру данными по умолчанию;
- b. Конструктор копии: создает новую структуру – копию исходной;
- c. Конструктор с параметрами: заполняет структуру данными из аргументов;
- d. Оператор ==: если все поля структур равны друг другу, то возвращает «истину», иначе – «ложь»;
- e. Оператор !=: возвращает инверсию результата сравнения структур.

2) Структура двунаправленного списка:

- a. Поля: указатели на первый и последний элементы `first` и `last` соответственно, длина списка;
- b. Конструктор: устанавливает указатели `first`, `last` равными `nullptr`, длину списка —равной 0;
- c. Деструктор: пока список содержит элементы, вызывает метод `delete_last`;
- d. Оператор доступа к элементу по индексу: проверяет, есть ли в списке элемент с индексом, переданным в аргументе; если да, возвращает указатель на него, если нет, возвращает `nullptr`;
- e. Метод `empty`: если указатель `last` — `nullptr`, возвращает «истину»;
- f. Метод `append_to_end`: добавляет новый элемент из аргумента в список;
- g. Метод `delete_last`: удаляет последний элемент списка и обновляет указатель `last`;
- h. Метод `delete_element`: удаляет элемент списка с индексом `index`, обновляет указатели предыдущего и следующего элемента, если такие есть;
- i. Метод `clear`: пока список содержит элементы, вызывает метод `delete_last`;
- j. Метод `show`: выводит на экран содержимое списка;
- k. Метод `is_in`: проверяет, есть ли в списке элемент, равный переданному в аргументе элементу;

- l. Метод `get_element_index`: проверяет, есть ли в списке элемент, равный переданному в аргументе элементу; если да, возвращает его индекс в списке, если нет, возвращает -1;
- m. Метод `get_element`: проверяет, есть ли в списке элемент, равный переданному в аргументе элементу; если да, возвращает указатель на него, если нет, возвращает `nullptr`;
- n. Метод `get_element_by_index`: действие аналогично действию оператора доступа к элементу по индексу;
- o. Метод `get_element_by_key`: проверяет, есть ли в списке элемент с ключом, равным переданному в аргументе; если да, возвращает указатель на него, если нет, возвращает `nullptr`;
- p. Метод `get_first`: возвращает указатель `first`;
- q. Метод `get_last`: возвращает указатель `last`;
- r. Метод `size`: возвращает длину списка;

3) Модуль `BinFileEditor`:

- a. Метод `openFile`. Принимает в качестве аргумента путь к файлу. Открывает файл, расположенный по заданному пути, в текстовом или двоичном формате. Если файл открыт успешно, возвращает «истину», иначе «ложь»;
- b. Метод `reopenFile`. Закрывает файл с помощью метода `close`, затем вызывает метод `openFile`;
- c. Метод закрытия файла `closeFile`. Если ошибки во время работы с файлом не были обнаружены, то выводит соответствующее сообщение. Затем закрывает файл;
- d. Метод `init`. Запрашивает у пользователя тип файла и путь к нему. Затем открывает его; если файл открыт успешно, то

считывает данные из него в структуру списка и возвращает «истину», иначе «ложь»;

- e. Метод `saveToBin`. Запрашивает у пользователя путь к создаваемому двоичному файлу. Затем создает файл с таким путем и, в случае успешного открытия, записывает в него данные списка и возвращает «истину», иначе «ложь»;
- f. Метод `saveToText`. Запрашивает у пользователя путь к создаваемому текстовому файлу. Затем создает файл с таким путем и, в случае успешного открытия, записывает в него данные списка и возвращает «истину», иначе «ложь»;
- g. Метод `saveBin`. Перезаписывает текущий файл текущими данными из списка и закрывает его, затем снова открывает файл;
- h. Метод `saveText`. Перезаписывает текущий файл текущими данными из списка и закрывает его, затем снова открывает файл;
- i. Метод `deleteRecordByKey`. Принимает в качестве аргумента номер полиса. Если запись с таким номером полиса найдена, то удаляет ее в списке и перезаписывает файл с помощью `saveBin` или `saveText` (в зависимости от формата открытого файла), после чего возвращает «истину», иначе «ложь»;
- j. Метод `searchByDisease`. Принимает в качестве аргумента код болезни. Запрашивает у пользователя путь к файлу, куда необходимо сохранить данные. Затем открывает файл по этому пути; если открытие успешно, записывает в файл все записи с переданным кодом болезни и возвращает «истину», иначе «ложь»;

- k. Метод `deleteRecord`. Принимает в качестве аргумента номер полиса. Если запись с таким номером полиса найдена, то заменяет ее последней записью в списке и перезаписывает файл с помощью `saveBin` или `saveText` (в зависимости от формата открытого файла), после чего возвращает «истину», иначе «ложь»;
- l. Метод `readByIndex`. Принимает в качестве аргументов индекс и ссылку на переменную типа `record`. Если переданный индекс меньше 0 или больше числа элементов в списке, то печатает соответствующее сообщение и возвращает «ложь»; в противном случае вызывает метод `reopenFile`; если он возвращает значение «ложь», то возвращает «ложь»; иначе считывает записи из файла в переменную, ссылка на которую была передана в аргументе, пока не будет достигнута запись с индексом, переданным в аргументе, после чего возвращает «истину»;
- m. Метод `showContent`. Вызывает метод `reopenFile`; если он возвращает «ложь», то возвращает «ложь»; если нет, то вызывает метод `clear` списка и считывает записи из файла в список. Затем вызывает метод `show` списка и возвращает «истину».
- n. Меню модуля `process`. Вызывает метод `init`; если он возвращает «ложь», то завершает работу. Затем в цикле предлагает пользователю одно из доступных действий. После выбора пользователем действия «выход» закрывает файл с помощью `closeFile`.

2.4 Код программы

Запишем код программы на языке C++ (листинги 6-10):

Листинг 6 – list_2way.h

```
#pragma once
#ifndef __list2way_h__
#define __list2way_h__
#include <iostream>
#include <string>
using namespace std;

struct record {
    int oms_code; //ключ
    char surname[20]; //фамилия
    char name[20]; //имя
    char patronymic[20]; //отчество
    char disease_code[6]; //код болезни
    char date[9]; //дата ДД.ММ.ГГ
    int doctor_id; //код врача

    record()
    {
        oms_code = -1;
        strcpy_s(this->surname, "generic");
        strcpy_s(this->name, "generic");
        strcpy_s(this->patronymic, "generic");
        strcpy_s(this->disease_code, "XXXXX");
        strcpy_s(this->date, "--.---");
        doctor_id = -1;
    }

    record(const record& copy)
    {
        this->oms_code = copy.oms_code;
        strcpy_s(this->surname, copy.surname);
        strcpy_s(this->name, copy.name);
        strcpy_s(this->patronymic, copy.patronymic);
        strcpy_s(this->disease_code, copy.disease_code);
        strcpy_s(this->date, copy.date);
        this->doctor_id = copy.doctor_id;
    }

    record(int oms_code, char* surname, char* name, char* patr, char*
disease_code, char* date, int doctor_id) :
        oms_code(oms_code), doctor_id(doctor_id)
    {
        strcpy_s(this->surname, surname);
        strcpy_s(this->name, name);
        strcpy_s(this->patronymic, patronymic);
        strcpy_s(this->disease_code, disease_code);
        strcpy_s(this->date, date);
    }

    bool operator == (record right)
    {
        if (this->oms_code != right.oms_code) return false;
        if (strcmp(this->surname, right.surname)) return false;
        if (strcmp(this->name, right.name)) return false;
    }
};
```

Продолжение листинга 6

```
        if (strcmp(this->patronymic, right.patronymic)) return
false;
        if (strcmp(this->disease_code, right.disease_code)) return
false;
        if (strcmp(this->date, right.date)) return false;
        if (this->doctor_id != right.doctor_id) return false;
        return true;
    }

    bool operator != (record right)
    {
        return !(*this == right);
    }
};

using std::cout; using std::endl;
struct list_chain_2way
{
    void show();

    record data;
    list_chain_2way* prev = nullptr;
    list_chain_2way* next = nullptr;
};

struct list_2way
{
private:
    list_chain_2way* first;
    list_chain_2way* last;
    int length = 0;
public:
    list_2way();

    ~list_2way();

    list_chain_2way* operator [] (int index);

    bool empty();

    void append_to_end(record new_record);

    void delete_last();
    void delete_element(int index);
    void clear();

    void show();

    bool is_in(record data);

    int get_element_index(record data);

    list_chain_2way* get_element(record data);
    list_chain_2way* get_element_by_key(int key);
    list_chain_2way* get_element_by_index(int index);
```

Продолжение листинга 6

```
list_chain_2way* get_first();
list_chain_2way* get_last();
int size();
};
#endif
```

Листинг 7 – list_2way.cpp

```
#include "list_2way.h"

void list_chain_2way::show()
{
    cout << "{ Номер полиса: " << this->data.oms_code << ", ФИО: " <<
this->data.surname << " " << this->data.name <<
    " " << this->data.patronymic << ", код болезни: " << this-
>data.disease_code << ", дата обращения - " << this->data.date
    << ", ИД врача: " << this->data.doctor_id << "}" << endl;
}

list_2way::list_2way()
{
    first = nullptr;
    last = nullptr;
}

list_2way::~~list_2way()
{
    while (!empty())
        delete_last();
}

void list_2way::clear()
{
    while (!empty())
        delete_last();
}

list_chain_2way* list_2way::operator [] (int index)
{
    if (empty()) return nullptr;
    list_chain_2way* cur = first;
    for (int i = 0; i < index; i++)
    {
        cur = cur->next;
        if (!cur)
            return nullptr;
    }
    return cur;
}

bool list_2way::empty()
{
    return (last == nullptr);
}
```

Продолжение листинга 7

```
}

void list_2way::append_to_end(record new_record)
{
    list_chain_2way* temp = new list_chain_2way;
    temp->data = new_record;
    length++;
    if (empty())
    {
        first = temp;
        last = temp;
        return;
    }
    temp->prev = last;
    last->next = temp;
    last = temp;
}

void list_2way::delete_last()
{
    if (empty()) return;
    list_chain_2way* temp = last;
    last = temp->prev;
    if (last)
        last->next = nullptr;
    delete temp;
    length--;
}

void list_2way::delete_element(int index)
{
    if (empty()) return;
    if (index >= this->size()) return;
    list_chain_2way* temp = this->get_element_by_index(index);
    list_chain_2way* old_prev = temp->prev;
    list_chain_2way* old_next = temp->next;
    if ((old_prev == nullptr) && (old_next == nullptr))
    {
        first = nullptr;
        last = nullptr;
    }
    else if (old_prev == nullptr)
    {
        first = old_next;
        old_next->prev = old_prev;
    }
    else if (old_next == nullptr)
    {
        last = old_prev;
        old_prev->next = old_next;
    }
    else
    {
        old_prev->next = old_next;
```

Продолжение листинга 7

```
        old_next->prev = old_prev;
    }
    delete temp;
    length--;
}

void list_2way::show()
{
    if (empty()) { cout << "Список пуст" << endl; return; }
    list_chain_2way* temp = first;
    cout << "[" << endl;
    while (temp != nullptr)
    {
        temp->show();
        temp = temp->next;
    }
    cout << "]" (длина: " << length << ")" << endl;
}

bool list_2way::is_in(record data)
{
    if (empty()) return false;
    list_chain_2way* temp = first;
    while ((temp != nullptr) && (temp->data != data))
    {
        temp = temp->next;
    }
    if (temp == nullptr)
        return false;
    else
        return true;
}

int list_2way::get_element_index(record data)
{
    if (empty()) return -2;
    list_chain_2way* temp = first;
    int index = 0;
    while ((temp != nullptr) && (temp->data != data))
    {
        temp = temp->next;
        index++;
    }
    if (temp == nullptr)
        return -1;
    else
        return index;
}

list_chain_2way* list_2way::get_element(record data)
{
    if (empty()) return nullptr;
    list_chain_2way* temp = first;
    while ((temp != nullptr) && (temp->data != data))
    {
```

Продолжение листинга 7

```
        temp = temp->next;
    }
    if (temp == nullptr)
        return nullptr;
    else
        return temp;
}

list_chain_2way* list_2way::get_element_by_key(int key)
{
    if (empty()) return nullptr;
    list_chain_2way* temp = first;
    while ((temp != nullptr) && (temp->data.oms_code != key))
    {
        temp = temp->next;
    }
    if (temp == nullptr)
        return nullptr;
    else
        return temp;
}

list_chain_2way* list_2way::get_element_by_index(int index)
{
    if (empty()) return nullptr;
    list_chain_2way* temp = first;
    int i = 0;
    while ((temp != nullptr) && (i < index))
    {
        temp = temp->next;
        i++;
    }
    if (temp == nullptr)
        return nullptr;
    else
        return temp;
}

list_chain_2way* list_2way::get_first()
{
    return this->first;
}

list_chain_2way* list_2way::get_last()
{
    return this->last;
}

int list_2way::size()
{
    return this->length;
}
```

Листинг 8 – BinFileEditor.h

```
#pragma once
#ifndef __BinFileEditor_H__
#define __BinFileEditor_H__
#include <iostream>
#include <string>
#include <conio.h>
#include <fstream>
#include "list_2way.h"
using namespace std;

namespace BinFileEditor {
    bool openFile(string path);
    bool reopenFile();
    void closeFile();
    void saveBin(); //служебное сохранение в двоичный файл
    void saveText(); //служебное сохранение в текстовый файл

    bool deleteRecordByKey(int key); //удалить пациента, сохраняя
    порядок остальных записей
    bool searchByDisease(string disease_code); //сформировать новый
    двоичный файл из людей с заболеванием
    bool deleteRecord(int key); //удалить запись по индексу,
    последнюю запись переместить на место удаленной
    bool readByIndex(int index, record &res); //прочитать запись по
    индексу
    bool showContent(); //содержимое
    bool saveToText(); //создание текстового файла на основе текущих
    записей
    bool saveToBin(); //создание двоичного файла на основе текущих
    записей

    void process(); //интерфейс для взаимодействия с файлом, который
    запускается из основной программы

    bool init(); //инициализация
}

#endif
```

Листинг 9 – BinFileEditor.cpp

```
#include "list_2way.h"
#include "BinFileEditor.h"

list_2way entries; //нужно для записи в файл
fstream file;
bool isBinary;
string _path;

bool BinFileEditor::openFile(string path)
{
    file = fstream(path.c_str(), ios::in | ios::out | (isBinary *
    ios::binary));
```

Продолжение листинга 9

```
        if (!file)
        {
            closeFile();
            return false;
        }
        return true;
    }

bool BinFileEditor::reopenFile()
{
    file.close();
    return openFile(_path);
}

void BinFileEditor::closeFile()
{
    if (file.good())
    {
        cout << "Ошибки ввода-вывода не обнаружены." << endl;
    }
    file.close();
}

bool BinFileEditor::init()
{
    char filetype = ' ';
    cout << "Выберите тип файла, который необходимо прочесть ([T] -  
текстовый, [B] - двоичный):";
    while (!((tolower(filetype) == 'b') || (tolower(filetype) ==  
't'))))
    {
        filetype = _getch();
    }
    string path;
    cout << endl << "Введите путь к файлу: ";
    getline(cin, path);
    _path = path;
    if (tolower(filetype) == 'b')
    {
        file = fstream(_path, ios::in | ios::binary);
        isBinary = true;
    }
    else
    {
        file = fstream(_path, ios::in);
        isBinary = false;
    }
    if (!file)
    {
        cout << "Ошибка при открытии файла" << endl;
        return false;
    }
    else
    {
        while (!file.eof())
```


Продолжение листинга 9

```
        {
            int temp_i;
            string temp_s;
            record temp_r = record();
            if (isBinary)
            {
                file.read((char*)&temp_r, sizeof(record));
            }
            else
            {
                getline(file, temp_s); temp_r.oms_code =
strtol(temp_s.c_str(), nullptr, 0);
                file.getline(temp_r.surname, 20);
                file.getline(temp_r.name, 20);
                file.getline(temp_r.patronymic, 20);
                file.getline(temp_r.disease_code, 6);
                file.getline(temp_r.date, 9);
                getline(file, temp_s); temp_r.doctor_id =
strtol(temp_s.c_str(), nullptr, 0);
            }
            entries.append_to_end(temp_r);
        }
        if (isBinary)
            entries.delete_last();
    }
    file.close();
    return true;
}

bool BinFileEditor::saveToBin()
{
    string path;
    cout << "Введите путь к создаваемому двоичному файлу: ";
    getline(cin, path);
    fstream file2(path, ios::out | ios::binary);
    if (!file2)
    {
        cout << "Произошла ошибка при записи в файл." << endl;
        return false;
    }
    for (int i = 0; i < entries.size(); i++)
    {
        file2.write((char*)(entries.get_element_by_index(i)),
sizeof(record));
    }
    file2.close();
    return true;
}

bool BinFileEditor::saveToText()
{
    string path;
    cout << "Введите путь к создаваемому текстовому файлу: ";
    getline(cin, path);
    fstream file2(path, ios::out);
```

Продолжение листинга 9

```
        if (!file2)
        {
            cout << "Произошла ошибка при записи в файл." << endl;
            return false;
        }
        for (int i = 0; i < entries.size(); i++)
        {
            file2 << entries[i]->data.oms_code << "\n";
            file2 << entries[i]->data.surname << "\n";
            file2 << entries[i]->data.name << "\n";
            file2 << entries[i]->data.patronymic << "\n";
            file2 << entries[i]->data.disease_code << "\n";
            file2 << entries[i]->data.date << "\n";
            file2 << entries[i]->data.doctor_id;
            if (i < entries.size() - 1)
                file2 << "\n";
        }
        file2.close();
        return true;
    }

    void BinFileEditor::saveBin()
    {
        file.close();
        file = fstream(_path.c_str(), ios::out | ios::binary);
        if (!file)
        {
            cout << "Произошла ошибка при записи в файл." << endl;
            return;
        }
        for (int i = 0; i < entries.size(); i++)
        {
            file.write((char*)(entries.get_element_by_index(i)),
sizeof(record));
        }
        file.close();
        file = fstream(_path.c_str(), ios::in | ios::out | ios::binary);
    }

    void BinFileEditor::saveText()
    {
        file.close();
        file = fstream(_path, ios::out);
        if (!file)
        {
            cout << "Произошла ошибка при записи в файл." << endl;
            return;
        }
        for (int i = 0; i < entries.size(); i++)
        {
            file << entries[i]->data.oms_code << "\n";
            file << entries[i]->data.surname << "\n";
            file << entries[i]->data.name << "\n";
            file << entries[i]->data.patronymic << "\n";
            file << entries[i]->data.disease_code << "\n";
```

Продолжение листинга 9

```
        file << entries[i]->data.date << "\n";
        file << entries[i]->data.doctor_id;
        if (i < entries.size() - 1)
            file << "\n";
    }
    file.close();
    file = fstream(_path.c_str(), ios::in | ios::out);
}

bool BinFileEditor::deleteRecordByKey(int key)
{
    if (!entries.get_element_by_key(key))
    {
        cout << "Запись с таким номером полиса не найдена" << endl;
        return false;
    }
    else
    {
        int index =
entries.get_element_index(entries.get_element_by_key(key)->data);
        entries.delete_element(index);
        if (isBinary)
        {
            saveBin();
        }
        else
        {
            saveText();
        }
        cout << "Запись успешно удалена" << endl;
    }
    return true;
}

bool BinFileEditor::searchByDisease(string disease_code)
{
    string path;
    cout << "Введите путь, по которому будут сохранены данные: ";
    getline(cin, path);
    fstream file2 = fstream(path, ios::out | ios::binary);
    if (!file2)
    {
        cout << "Произошла ошибка при записи в файл." << endl;
        return false;
    }
    else
    {
        int total = 0;
        for (int i = 0; i < entries.size(); i++)
        {
            if (entries[i]->data.disease_code == disease_code)
            {
                file2.write((char*)(entries.get_element_by_index(i)),
sizeof(record));
            }
        }
    }
}
```

Продолжение листинга 9

```
        total++;
    }
}
cout << "Создан двоичный файл; перенесено " << total << "
записей" << endl;
}
return true;
}

bool BinFileEditor::deleteRecord(int key)
{
    if (!entries.get_element_by_key(key))
    {
        cout << "Запись с таким номером полиса не найдена" << endl;
        return false;
    }
    else
    {
        int index =
entries.get_element_index(entries.get_element_by_key(key)->data);
        record last = record(entries.get_last()->data);
        entries[index]->data = record(last);
        if (isBinary)
        {
            saveBin();
        }
        else
        {
            saveText();
        }
        cout << "Запись успешно заменена на последнюю запись" <<
endl;
    }
    return true;
}

bool BinFileEditor::readByIndex(int index, record &res)
{
    if ((entries.size() < index) || (index < 0))
    {
        cout << "Записи с индексом " << index << " не существует" <<
endl;
        return false;
    }
    if (!reopenFile())
    {
        cout << "Ошибка при чтении файла" << endl;
        return false;
    }
    string temp_s;
    res = record();
    for (int i = 0; i < index; i++)
    {
        if (isBinary)
        {
```

Продолжение листинга 9

```
        file.read((char*)&res, sizeof(record));
    }
    else
    {
        getline(file, temp_s); res.oms_code =
        strtol(temp_s.c_str(), nullptr, 0);
        file.getline(res.surname, 20);
        file.getline(res.name, 20);
        file.getline(res.patronymic, 20);
        file.getline(res.disease_code, 6);
        file.getline(res.date, 9);
        getline(file, temp_s); res.doctor_id =
        strtol(temp_s.c_str(), nullptr, 0);
    }
    return true;
}

bool BinFileEditor::showContent()
{
    if (!reopenFile())
    {
        cout << "Ошибка при чтении файла" << endl;
        return false;
    }
    cout << "Содержимое файла: " << endl;
    entries.clear();
    while (!file.eof())
    {
        int temp_i;
        string temp_s;
        record temp_r = record();
        if (isBinary)
        {
            file.read((char*)&temp_r, sizeof(record));
        }
        else
        {
            getline(file, temp_s); temp_r.oms_code =
            strtol(temp_s.c_str(), nullptr, 0);
            file.getline(temp_r.surname, 20);
            file.getline(temp_r.name, 20);
            file.getline(temp_r.patronymic, 20);
            file.getline(temp_r.disease_code, 6);
            file.getline(temp_r.date, 9);
            getline(file, temp_s); temp_r.doctor_id =
            strtol(temp_s.c_str(), nullptr, 0);
        }
        entries.append_to_end(temp_r);
    }
    if (isBinary)
        entries.delete_last();
    entries.show();
    return true;
}
```

Продолжение листинга 9

```
void BinFileEditor::process()
{
    if (!init()) return;
    char choose = ' ';
    int n;
    record t;
    string disease_code_search = "";
    bool running = true;
    while (running)
    {
        cout << "Выберите действие: " << endl
        << "[T] - сохранить текущие данные в текстовый файл"
        << "[B] - сохранить текущие данные в двоичный файл" <<
        << "[C] - вывести содержимое текущего файла" << endl
        << "[I] - получить запись по индексу" << endl
        << "[D] - удалить запись по ключу путем замены на
последнюю запись" << endl
        << "[K] - удалить запись по ключу, сохраняя порядок
остальных записей" << endl
        << "[S] - сформировать новый двоичный файл из записей
с определенным кодом болезни" << endl
        << "[Q] - выход" << endl;
        choose = _getch();
        switch (tolower(choose))
        {
            case 'q':
                running = false;
                break;
            case 't':
                saveToText();
                break;
            case 'b':
                saveToBin();
                break;
            case 'c':
                showContent();
                break;
            case 'i':
                cout << "Введите индекс:" << endl;
                cin >> n;
                cin.ignore(32768, '\n');
                if (readByIndex(n, t))
                {
                    cout << "Запись с индексом " << n << ": " << "{
Номер полиса: " << t.oms_code << ", ФИО: " << t.surname << " " <<
t.name <<
                    " " << t.patronymic << ", код болезни: " <<
t.disease_code << ", дата обращения - " << t.date
                    << ", ИД врача: " << t.doctor_id << "}" <<
endl;
                }
                break;
        }
    }
}
```

Продолжение листинга 9

```
        case 'd':
            cout << "Введите номер полиса:" << endl;
            cin >> n;
            cin.ignore(32768, '\n');
            if (entries.get_element_by_key(n))
            {
                t = entries.get_element_by_key(n)->data;
                deleteRecord(t.oms_code);
            }
            else
            {
                cout << "Запись с таким номером полиса не
найдена" << endl;
            }
            break;
        case 'k':
            cout << "Введите номер полиса:" << endl;
            cin >> n;
            cin.ignore(32768, '\n');
            if (entries.get_element_by_key(n))
            {
                t = entries.get_element_by_key(n)->data;
                deleteRecordByKey(t.oms_code);
            }
            else
            {
                cout << "Запись с таким номером полиса не
найдена" << endl;
            }
            break;
        case 's':
            cout << "Введите код болезни, который необходимо
найти: " << endl;
            cin >> disease_code_search;
            cin.ignore(32768, '\n');
            searchByDisease(disease_code_search);
            break;
        default:
            cout << "Неизвестное действие." << endl;
    }
    if (running)
    {
        system("pause");
        system("cls");
    }
}
closeFile();
}
```

Листинг 10 – main.cpp

```
#include <iostream>
#include <locale.h>
#include "list_2way.h"
#include "BinFileEditor.h"

int main()
{
    setlocale(LC_ALL, "ru");
    BinFileEditor::process();
}
```

2.5 Тестирование

Проведем тестирование написанной программы (рисунки 11-):

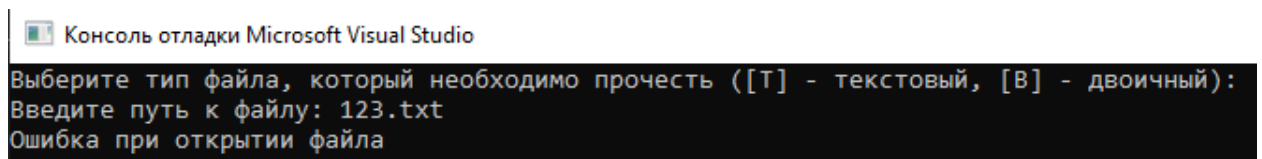


Рисунок 11 – Попытка открытия несуществующего файла

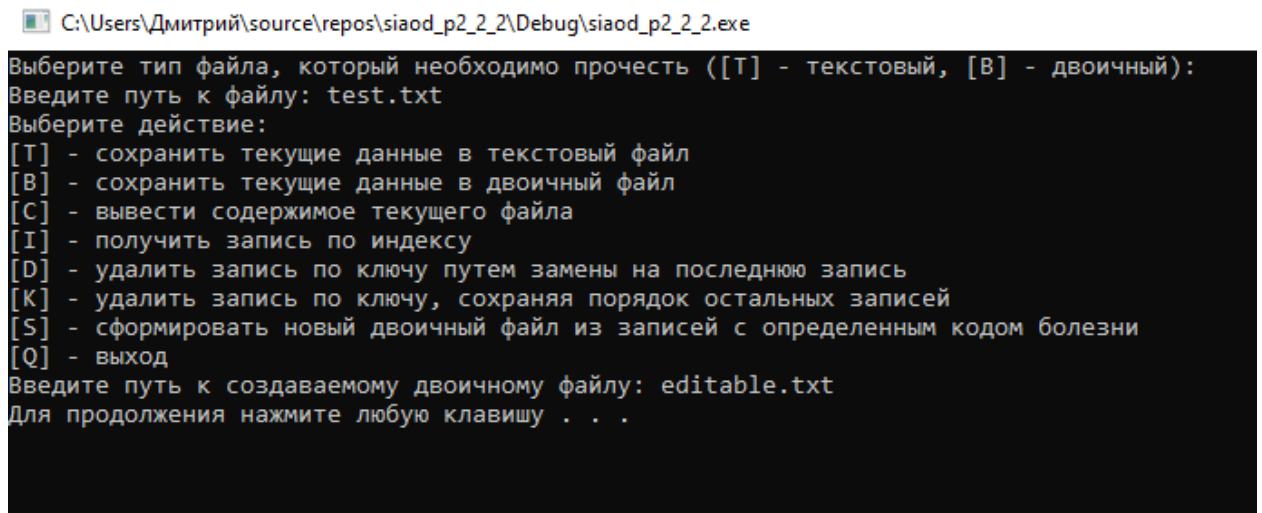


Рисунок 12 – Создание нового двоичного файла, с которым будет проводиться дальнейшая работа


```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe
Выберите тип файла, который необходимо прочесть ([T] - текстовый, [B] - двоичный):
Введите путь к файлу: editable.txt
Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Содержимое файла:
[
{ Номер полиса: 123, ФИО: Ivanov Ivan Ivanovich, код болезни: X03, дата обращения - 08.09.10, ИД врача: 8800}
{ Номер полиса: 124, ФИО: Petrov Ivan Vasilyevich, код болезни: M05, дата обращения - 09.09.10, ИД врача: 8800}
{ Номер полиса: 125, ФИО: Sidorov Fedor Ivanovich, код болезни: X03, дата обращения - 10.09.10, ИД врача: 8800}
{ Номер полиса: 126, ФИО: Smirnov Sergey Sergeevich, код болезни: A04, дата обращения - 11.09.10, ИД врача: 8800}
{ Номер полиса: 127, ФИО: Fedorov Valeriy Ivanovich, код болезни: X03, дата обращения - 12.09.10, ИД врача: 8800}
{ Номер полиса: 128, ФИО: Zaicev Petr Konstantinovich, код болезни: X03, дата обращения - 13.09.10, ИД врача: 8815}
{ Номер полиса: 129, ФИО: Petrova Marina Ivanovna, код болезни: D06, дата обращения - 14.09.10, ИД врача: 8876}
{ Номер полиса: 130, ФИО: Kuznetsov Dmitriy Anatolievich, код болезни: E01, дата обращения - 15.09.10, ИД врача: 2646}
] (длина: 8)
Для продолжения нажмите любую клавишу . . .
```

Рисунок 13 – Вывод содержимого двоичного файла

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe
Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Введите индекс:
76
Записи с индексом 76 не существует
Для продолжения нажмите любую клавишу . . .
```

Рисунок 14 – Попытка получения записи с несуществующим индексом

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe
Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Введите индекс:
4
Запись с индексом 4: { Номер полиса: 126, ФИО: Smirnov Sergey Sergeevich, код болезни: A04, дата обращения - 11.09.10, ИД врача: 8800}
Для продолжения нажмите любую клавишу . . .
```

Рисунок 15 – Получение записи с индексом 4

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe

Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Введите номер полиса:
103
Запись с таким номером полиса не найдена
Для продолжения нажмите любую клавишу . . .
```

Рисунок 16 – Попытка удаления записи с несуществующим номером полиса

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe

Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Введите номер полиса:
124
Запись успешно заменена на последнюю запись
Для продолжения нажмите любую клавишу . . .
```

Рисунок 17 – Замена записи с номером 124 на последнюю запись

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe

Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Содержимое файла:
[
{ Номер полиса: 123, ФИО: Ivanov Ivan Ivanovich, код болезни: X03, дата обращения - 08.09.10, ИД врача: 8800}
{ Номер полиса: 130, ФИО: Kuznetsov Dmitriy Anatolievich, код болезни: E01, дата обращения - 15.09.10, ИД врача: 2646}
{ Номер полиса: 125, ФИО: Sidorov Fedor Ivanovich, код болезни: X03, дата обращения - 10.09.10, ИД врача: 8800}
{ Номер полиса: 126, ФИО: Smirnov Sergey Sergeevich, код болезни: A04, дата обращения - 11.09.10, ИД врача: 8800}
{ Номер полиса: 127, ФИО: Fedorov Valeriy Ivanovich, код болезни: X03, дата обращения - 12.09.10, ИД врача: 8800}
{ Номер полиса: 128, ФИО: Zaicev Petr Konstantinovich, код болезни: X03, дата обращения - 13.09.10, ИД врача: 8815}
{ Номер полиса: 129, ФИО: Petrova Marina Ivanovna, код болезни: D06, дата обращения - 14.09.10, ИД врача: 8876}
{ Номер полиса: 130, ФИО: Kuznetsov Dmitriy Anatolievich, код болезни: E01, дата обращения - 15.09.10, ИД врача: 2646}
] (длина: 8)
Для продолжения нажмите любую клавишу . . .
```

Рисунок 18 – Содержимое файла editable.txt после замены, проведенной на рисунке 17

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe

Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Введите номер полиса:
125
Запись успешно удалена
Для продолжения нажмите любую клавишу . . .
```

Рисунок 19 – Удаление записи с номером 125

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe

Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Содержимое файла:
[
{ Номер полиса: 123, ФИО: Ivanov Ivan Ivanovich, код болезни: X03, дата обращения - 08.09.10, ИД врача: 8800}
{ Номер полиса: 130, ФИО: Kuznetsov Dmitriy Anatolievich, код болезни: E01, дата обращения - 15.09.10, ИД врача: 2646}
{ Номер полиса: 126, ФИО: Smirnov Sergey Sergeevich, код болезни: A04, дата обращения - 11.09.10, ИД врача: 8800}
{ Номер полиса: 127, ФИО: Fedorov Valeriy Ivanovich, код болезни: X03, дата обращения - 12.09.10, ИД врача: 8800}
{ Номер полиса: 128, ФИО: Zaicev Petr Konstantinovich, код болезни: X03, дата обращения - 13.09.10, ИД врача: 8815}
{ Номер полиса: 129, ФИО: Petrova Marina Ivanovna, код болезни: D06, дата обращения - 14.09.10, ИД врача: 8876}
{ Номер полиса: 130, ФИО: Kuznetsov Dmitriy Anatolievich, код болезни: E01, дата обращения - 15.09.10, ИД врача: 2646}
] (длина: 7)
Для продолжения нажмите любую клавишу . . .
```

Рисунок 20 - Содержимое файла editable.txt после удаления, проведенного на рисунке 19

```
C:\Users\Дмитрий\source\repos\siaod_p2_2_2\Debug\siaod_p2_2_2.exe

Выберите действие:
[T] - сохранить текущие данные в текстовый файл
[B] - сохранить текущие данные в двоичный файл
[C] - вывести содержимое текущего файла
[I] - получить запись по индексу
[D] - удалить запись по ключу путем замены на последнюю запись
[K] - удалить запись по ключу, сохраняя порядок остальных записей
[S] - сформировать новый двоичный файл из записей с определенным кодом болезни
[Q] - выход
Введите код болезни, который необходимо найти:
A04
Введите путь, по которому будут сохранены данные: found.txt
Создан двоичный файл; перенесено 1 записей
Для продолжения нажмите любую клавишу . . .
```

Рисунок 21 – Сохранение записей с кодом болезни A04 в новый файл



ЗАКЛЮЧЕНИЕ

В ходе работы были выполнены два задания по применению файловых потоков языка C++, по управлению текстовым и двоичным файлами, проведены тестирования программ на наборе тестовых данных. Все программы работают корректно, следовательно, можно говорить о правильности алгоритмов, реализованных с помощью программ.

ИНФОРМАЦИОННЫЙ ИСТОЧНИК

Задания для самостоятельной работы №2. URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F1137402%2Fmod_assign%2Fintroattachment%2F0%2FСиАОД%20Самостоятельная%20работа%202%20%28управление%20файлом%29.pdf&forcedownload=1, дата обращения: 23.09.23