Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине «Основы алгоритмизации и программирования»

За 2 семестр

Тема: «Бинарные файлы. Файл из структур»

Выполнили:

Студенты 1 курса

Группы ИИ-21(1)

Кирилович А. А.

Литвинюк Т. В.

Проверил:

Гирель Т. Н.

Брест 2022

**Цель работы:** изучить принципы программирования с использованием бинарных файлов.

**Ход работы:**

**Вариант 4**

**Задание 1**

Реализовать меню, разработанное с применением массива указателей на функции.

Для каждого из вариантов задания необходимо разработать следующие функции:

1. Создание файла.

2. Добавление элемента в конец файла.

3. Вывод содержимого файла на экран.

4. Удаление элемента из файла:

5. Упорядочивание элементов в файле по выбранному признаку.

#include <iostream>

#include <windows.h>

#include <string>

#include <iomanip>

#include <fstream>

#include <vector>

using namespace std;

struct Date {

    int year = 0;

    int month = 0;

    int day = 0;

};

struct Address {

    int zipCode = 0;

    string country;

    string region;

    string district;

    string city;

    string street;

    int house = 0;

    int apartment = 0;

};

struct Patient {

    string surname;

    string name;

    string lastName;

    string sex;

    string nationality;

    float growth = 0;

    float weight = 0;

    struct Date dateOfBirth;

    string phoneNumber;

    struct Address homeAddress;

    int hospitalNumber = 0;

    string branch;

    int medicalCardNumber = 0;

    string diagnosis;

    string bloodType;

};

void create();

void viewing();

void add();

void remove();

void sort();

int menu();

void(\*masf[])() = {create, viewing, add, remove, sort};

Patient\* AddStruct();

void setData();

void recording();

void head();

int amount = 0;

int I = 0;

Patient\* OurPatients = new Patient[0];

string PATH = "file1.txt";

ofstream file\_out;

ifstream file\_in;

int main() {

    head();

    while(1) (\*masf[menu()])();

    try {

        delete[] OurPatients;

    }

    catch(const std::exception& e) {

        std::cerr << e.what() << '\n';

    }

}

int menu() {

    char n;

    do {

        cout << "\t   Menu\n";

        cout << "\t1. Create.\n";

        cout << "\t2. Viewing.\n";

        cout << "\t3. Add.\n";

        cout << "\t4. Delete.\n";

        cout << "\t5. Sort.\n";

        cout << "\t6. Exit.\n";

        printf ("Make a choice: ");

        n = getchar();

        system("cls");

    }

    while (strchr("123456", n) == NULL);

    if (n == '6') exit(0);

    return n - 49;

}

void create() {

    int YesOrNot = 0;

    do {

        OurPatients = AddStruct();

        setData();

        amount++;

        cout << "Continue entering data (1 - yes, 0 - no)?";

        cin >> YesOrNot;

        cin.get();

        recording();

        delete[] OurPatients;

        amount--;

    } while (YesOrNot != 0);

}

void viewing() {

    file\_in.open(PATH, ios\_base::in);

    string s, sum\_s = "";

    while (getline(file\_in, s)) {

        sum\_s += s + "\n";

    }

    file\_in.close();

    cout << sum\_s;

    system("pause");

}

void recording() {

    file\_out.open(PATH, ios\_base::app);

    for(int i = 1; i <= 300; i++) {

        file\_out << "=";

    }

    file\_out << endl;

    file\_out << left << setw(3) << I + 1 << "|" << setw(15) << OurPatients[0].surname << "|"

            << setw(10) << OurPatients[0].name << "|" << setw(15) << OurPatients[0].lastName << "|"

            << setw(6) << OurPatients[0].sex << "|" << setw(15) << OurPatients[0].nationality << "|"

            << setw(6) << OurPatients[0].growth << "|" << setw(6) << OurPatients[0].weight << "|"

            << setw(2) << OurPatients[0].dateOfBirth.day << "."

            << setw(2) << OurPatients[0].dateOfBirth.month << "." << setw(4) << OurPatients[0].dateOfBirth.year

            << "|" << setw(13) << OurPatients[0].phoneNumber << "|" << setw(15) << OurPatients[0].hospitalNumber << "|"

            << setw(6) << OurPatients[0].branch << "|" << setw(19) << OurPatients[0].medicalCardNumber << "|"

            << setw(15) << OurPatients[0].diagnosis << "|" << setw(10) << OurPatients[0].bloodType << "|"

            << setw(8) << OurPatients[0].homeAddress.zipCode << "|" << setw(10) << OurPatients[0].homeAddress.country << "|"

            << setw(10) << OurPatients[0].homeAddress.region << "|" << setw(10) << OurPatients[0].homeAddress.district << "|"

            << setw(10) << OurPatients[0].homeAddress.city << "|" << setw(20) << OurPatients[0].homeAddress.street << "|"

            << setw(5) << OurPatients[0].homeAddress.house << "|" << setw(9) << OurPatients[0].homeAddress.apartment << endl;;

    I++;

    file\_out.close();

}

void head() {

    file\_out.open(PATH, ios\_base::out);

    file\_out << left << setw(3) << "N" << "|" << setw(15) << "Surname" << "|" << setw(10) << "Name" << "|" << setw(15) << "Last name" << "|"

                << setw(6) << "Gender" << "|" << setw(15) << "Nationality" << "|" << setw(6) << "Growth" << "|"

                << setw(6) << "Weight" << "|" << setw(10) << "Date" << "|" << setw(13) << "Phone number" << "|"

                << setw(15) << "Hospital number" << "|" << setw(6) << "Branch" << "|" << setw(19) << "Medical card number" << "|"

                << setw(15) << "Diagnosis" << "|" << setw(10) << "Blood type" << "|"

                << setw(8) << "Zip code" << "|" << setw(10) << "Country" << "|" << setw(10) << "Region" << "|"

                << setw(10) << "District" << "|" << setw(10) << "City" << "|" << setw(20) << "Street" << "|"

                << setw(5) << "House" << "|" << setw(9) << "Apartment" << endl;

    file\_out.close();

}

void add() {

    create();

}

void remove() {

    cout <<"Enter the line number to delete: "<< endl;

    int lineNumberToDelete;

    cin >> lineNumberToDelete;

    file\_in.open(PATH, ios\_base::in);

    string s, sum\_s = "";

    int i = 0;

    while (getline(file\_in, s)) {

        i++;

        if (i == 1 || i == 2) continue;

        if (i - 2 == lineNumberToDelete) continue;

        sum\_s += s + "\n";

    }

    file\_in.close();

    head();

    file\_out.open(PATH, ios\_base::app);

    file\_out << sum\_s;

    file\_out.close();

}

void sort() {

}

Patient\* AddStruct() {

    if (amount == 0) {

        OurPatients = new Patient[amount + 1];

    }

    else {

        Patient\* tempOurPatients = new Patient[amount + 1];

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

            tempOurPatients[i] = OurPatients[i];

        }

        delete [] OurPatients;

        OurPatients = tempOurPatients;

    }

    return OurPatients;

}

void setData() {

    cout << "Surname: ";

    cin >> OurPatients[amount].surname;

    cout << "Name: ";

    cin >> OurPatients[amount].name;

    cout << "Last name: ";

    cin >> OurPatients[amount].lastName;

    cout << "Gender (m/f): ";

    cin >> OurPatients[amount].sex;

    cout << "Nationality: ";

    cin >> OurPatients[amount].nationality;

    cout << "Growth: ";

    cin >> OurPatients[amount].growth;

    cin.get();

    cout << "Weight: ";

    cin >> OurPatients[amount].weight;

    cin.get();

    cout << "Date of birth (year, month, day): ";

    year:

    cin >> OurPatients[amount].dateOfBirth.year;

    if(OurPatients[amount].dateOfBirth.year < 1900) {

        cout << "Enter the normal year! >>";

        goto year;

    }

    month:

    cin >> OurPatients[amount].dateOfBirth.month;

    if(OurPatients[amount].dateOfBirth.month > 12 || OurPatients[amount].dateOfBirth.month < 1) {

        cout << "Enter the normal month! >>";

        goto month;

    }

    cin >> OurPatients[amount].dateOfBirth.day;

    cin.get();

    cout << "Phone number: ";

    cin >> OurPatients[amount].phoneNumber;

    cout << "Home address (zip code, country, region, district, city, street, house, apartment): \n";

    cin >> OurPatients[amount].homeAddress.zipCode;

    cin >> OurPatients[amount].homeAddress.country;

    cin >> OurPatients[amount].homeAddress.region;

    cin >> OurPatients[amount].homeAddress.district;

    cin >> OurPatients[amount].homeAddress.city;

    cin >> OurPatients[amount].homeAddress.street;

    cin >> OurPatients[amount].homeAddress.house;

    cin >> OurPatients[amount].homeAddress.apartment;

    cin.get();

    cout << "Hospital number: ";

    cin >> OurPatients[amount].hospitalNumber;

    cin.get();

    cout << "Branch: ";

    cin >> OurPatients[amount].branch;

    cout << "Medical card number: ";

    cin >> OurPatients[amount].medicalCardNumber;

    cin.get();

    cout << "Diagnosis: ";

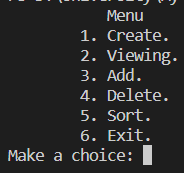
    cin >> OurPatients[amount].diagnosis;

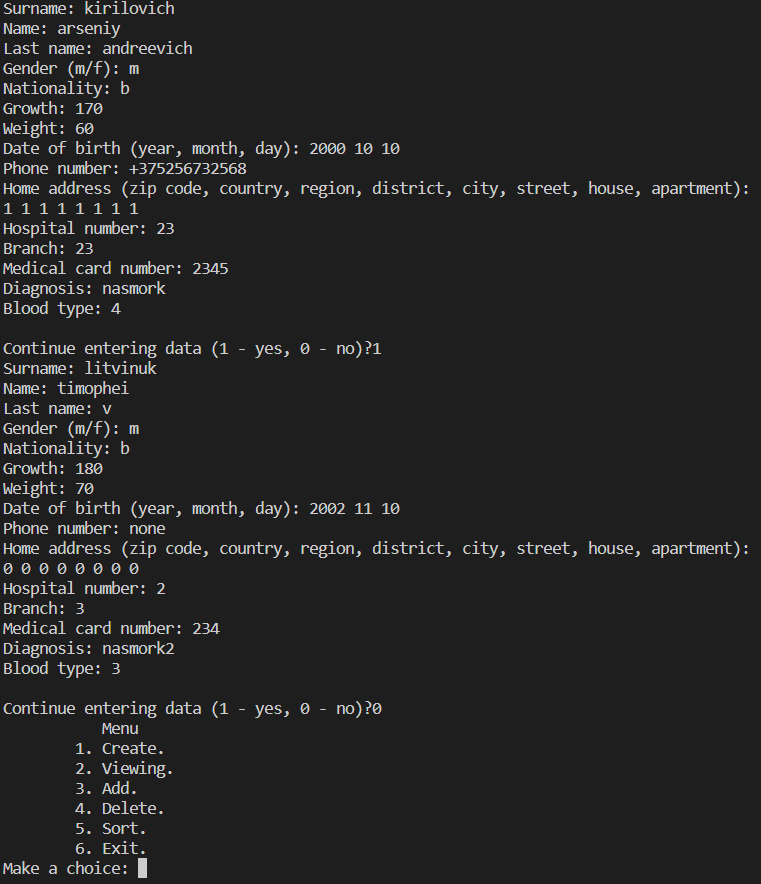
    cout << "Blood type: ";

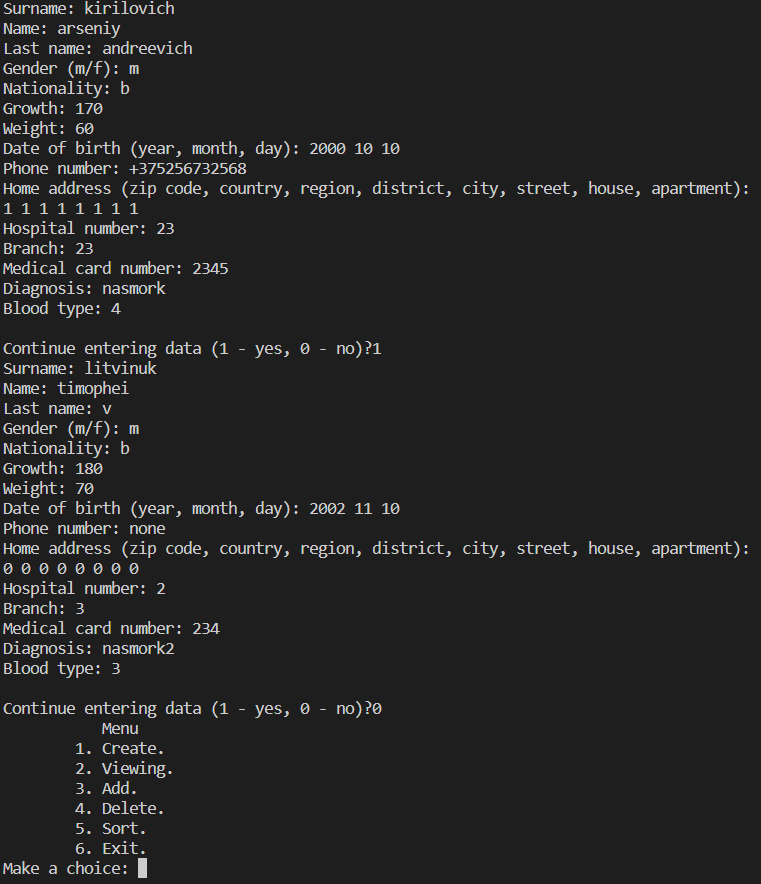
    cin >> OurPatients[amount].bloodType;

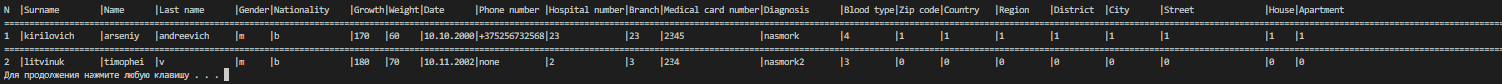
    cout << endl;

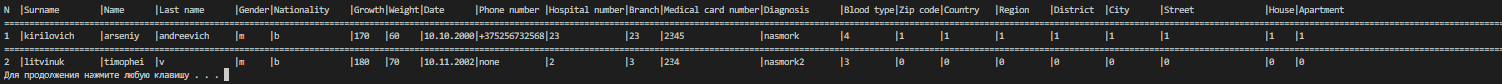
}

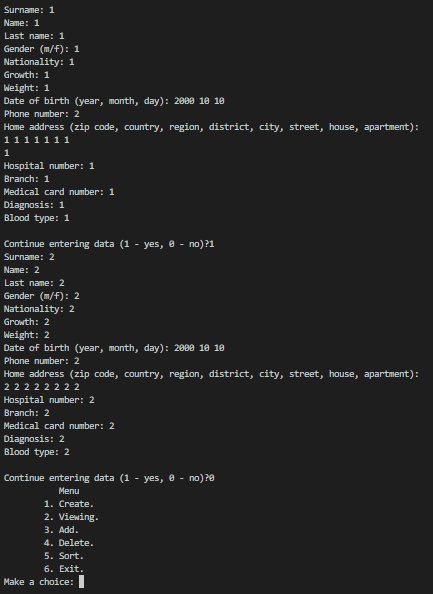
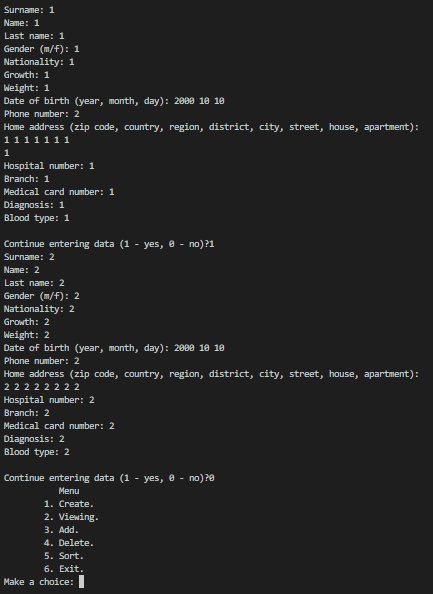


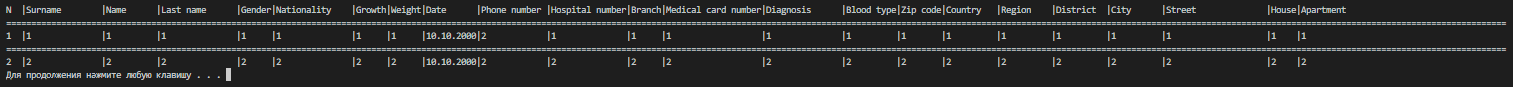


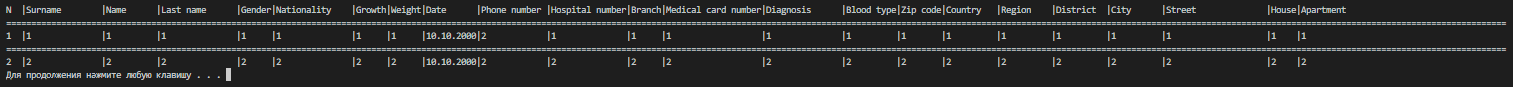


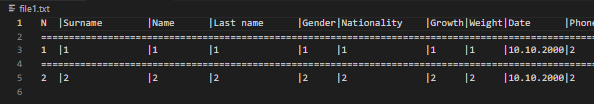












**Задание 2**

В первом файле хранится k матриц, во втором l матриц размерности n×m. Добавить во второй файл те матрицы из первого, которых нет во втором. Вывести на экран содер­жимое первого и второго файлов.

#define TRUE 1

#include <stdio.h>

#include <conio.h>

#include <process.h>

#include <stdlib.h>

void BlockWriteFile(char\* ,char\* , unsigned , unsigned , unsigned );

int\* InitMatrix(unsigned ,unsigned , unsigned );

int\* FreeMemory(unsigned , unsigned );

void DisplayFile( char\* , char\* , unsigned , unsigned );

void CompareFile(char\* , char\* , char\* , unsigned , unsigned );

void DisplayMatrix (int\* , unsigned , unsigned );

int main(void) {

    unsigned k, l, n, m ;

    char FileName1 [20], FileName2 [20];

    while(TRUE) {

        printf("\nEnter k number of matrixs:\n");

        scanf("%u",&k);

        printf("\nEnter n, m dimentions of matrixs:\n");

        scanf("%u%u",&n,&m);

        if ( (k > 0) && (n > 0) && (m > 0) ) break;

        printf("\nNumber is incorrect!!! Try again!!!\n");

    }

    printf("\nEnter the name of file: \n");

    scanf("%s",FileName1);

    BlockWriteFile(FileName1, "wb", k, n, m);

    printf("\nThe contents of file <<%s>>:\n",FileName1);

    DisplayFile(FileName1, "rb", n, m);

    printf("\nEnter l number of matrixs:\n");

    scanf("%u",&l);

    printf("\nEnter the name of file: \n");

    scanf("%s",FileName2);

    BlockWriteFile(FileName2, "wb", l, n, m);

    printf("\nThe contents of file <<%s>>:\n",FileName2);

    DisplayFile(FileName2, "rb", n, m);

    CompareFile(FileName1, FileName2, "a+b", n, m);

    printf("\nThe contents of file <<%s>>:\n",FileName1);

    DisplayFile(FileName1, "rb", n, m);

    printf("\nThe contents of file <<%s>>:\n",FileName2);

    DisplayFile(FileName2, "rb", n, m);

    printf("\n Press any key to exit...");

    getch();

    return 0;

}

int\* InitMatrix(unsigned l, unsigned n, unsigned m) {

    unsigned i;

    int\* Pointer = (int\*)malloc(n\*m\*sizeof(int));

    for( i = 0; i < n \* m; i++) {

        Pointer[i] = l + 1;//rand()%10;

    }

    return Pointer;

}

int\* FreeMemory(unsigned n, unsigned m) {

    int\* Pointer = (int\*)malloc(n\*m\*sizeof(int));

    return Pointer;

}

void BlockWriteFile(char\* String, char\* Mode, unsigned k, unsigned n, unsigned m) {

    int BufSize = sizeof(int) \* n \* m;

    int\* Pointer = (int\*)malloc(BufSize);

    unsigned i;

    FILE\* FilePointer= fopen(String, Mode);

    if (FilePointer== NULL) {

        printf("Can't open file to write.");

        getch();

        abort();

    }

    for ( i = 0; i < k; i++ ) {

        Pointer = InitMatrix(i, n, m);

        fwrite(Pointer, BufSize,1, FilePointer);

    }

    fclose(FilePointer);

    free(Pointer);

}

void DisplayFile(char\* String, char\* Mode, unsigned n, unsigned m) {

    int BufSize = sizeof(int)\*n\*m, Sector = 0;

    int\* Pointer = FreeMemory(n, m);

    FILE\* FilePointer= fopen(String, Mode);

    if ( FilePointer== NULL) {

        printf("\nCan't open file to read.");

        getch();

        abort();

    }

    while(fread(Pointer,BufSize,1, FilePointer) != 0) {

        printf("\n %d's matrix \n",(Sector + 1));

        DisplayMatrix(Pointer, n, m);

        Sector++;

    }

    fclose(FilePointer);

    free(Pointer);

}

void CompareFile(char\* String1, char\* String2, char\* Mode, unsigned n, unsigned m) {

    int BufSize = sizeof(int)\*n\*m;

    int\* Pointer1 = FreeMemory(n, m);

    int\* Pointer2 = FreeMemory(n, m);

    FILE\* FilePointer1= fopen(String1, Mode);

    if ( FilePointer1== NULL) {

        printf("\nCan't open file to read.");

        getch();

        abort();

    }

    FILE\* FilePointer2= fopen(String2, Mode);

    if ( FilePointer2== NULL) {

        printf("\nCan't open file to read.");

        getch();

        abort();

    }

    int flagSmall = 0, flagBig;

    while(fread(Pointer1,BufSize,1, FilePointer1) != 0 ) {

        fseek(FilePointer2,0,SEEK\_SET);

        while(fread(Pointer2,BufSize,1, FilePointer2) != 0 ) {

            unsigned i, j;

            for( i = 0; i < n; i++) {

                for( j = 0; j < m; j++) {

                    if (\*(Pointer2 + i \* m + j) != \*(Pointer1 + i \* m + j)) {

                        flagSmall += 1;

                    }

                }

            }

            if (flagSmall > 0) {

                flagBig = 0;

            }

            else {

                flagBig = 1;

            }

            flagSmall = 0;

        }

        if (flagBig == 0) {

            fwrite(Pointer1, BufSize,1, FilePointer2);

        }

    }

    fclose(FilePointer1);

    fclose(FilePointer2);

    free(Pointer1);

    free(Pointer2);

}

void DisplayMatrix(int\* Pointer, unsigned n, unsigned m) {

    unsigned i, j;

    for( i = 0; i < n; i++) {

        for( j = 0; j < m; j++) {

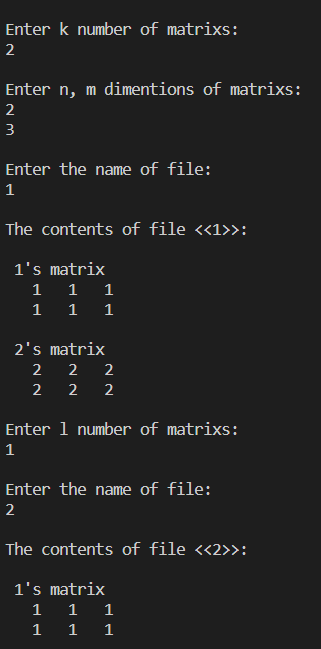
            printf("%4d",\*(Pointer + i \* m + j));

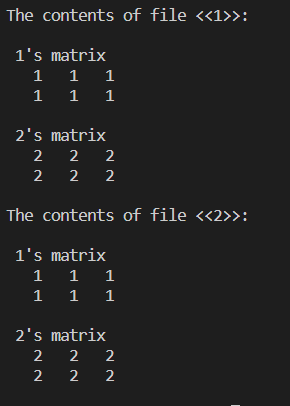
        }

        printf("\n");

    }

}

****

****

**Вывод:** изучил принципы программирования с использованием бинарных файлов.