Отчёт по лабораторной работе № 11 **Вариант 2**

Дисциплина «Программирование и информатика», **2 семестр**

«Многомодульные проекты»

Выполнил

студент группы ДИПРб11 _____Тагиров Р.Р. «____»_____ 2020

Проверила

ст. преп. кафедры АСОИУ _____ Толасова В.В. «____»_____ 2020

Цель:

Получить практические навыки работы с многомодульными проектами

Постановка задачи:

Дана структура для хранения данных о планшетных сканерах:

```
struct scan_info
{
	char model[25]; / / наименование модели
	int price: / / цена
	double x_size: / / горизонтальный размер области сканирования
	double y_size: / / вертикальный размер области сканирования
	int optr: / / оптическое разрешение
	int grey: / / число градаций серого
};
```

Написать функцию, которая записывает в бинарный файл данные о сканере из приведенной структуры.

Структура файла: в первых двух байтах размещается значение тина short int, определяющее количество сделанных в файл записей; далее без пропусков размещаются записи о сканерах.

Написать функцию, которая сортирует записи в описанном выше бинарном файле по одной из следующих характеристик: цена либо число градаций серого. Обязательный параметр – признак, задающий критерий сортировки.

Привести пример программы, создающей файл с данными о сканерах (данные вводятся с клавиатуры) из не менее восьми записей и осуществляющий его сортировку.

Описание функций

В таблице 2 приведены внешние функции для работы со сложными структурами данных.

Таблица 2 – Функции, обеспечивающие работу со сложными структурами данных.

Прототип	Назначение
void append(std::string filename, std::vector<scanner> v)	Добавляет данный массив scanner в данный бинарный файл и обновляет первый элемент, содержащий их количество
void sort_by(std::string filename, scanner_traits tr)	Сортирует данный бинарный файл scanner по цене или по числу градаций серого
void print(std::string filename)	Выводит содержимое данного бинарного файла scanner

Во всех программах присутствуют данные функции, но них отличаются сигнатуры в зависимости от задания.

Программа и методика испытаний

Scanners.bin содержит:
Canon 4400 2400x2400 1000 300
Canon 5530 4800x4800 400 500
Epson 9959 4800x4800 700 1000
HP 12990 2400x2400 5590 600
Canon 22350 600x600 1200 2400
Epson 24790 600x600 1630 720
Microtek 18911 1200x1200 800 3300
Panasonic 12200 600x600 1200 810

В таблице 3 приведены тестовые данные для проверки работоспособности программы

Таблица 3 – Данные для проверки работоспособности

№	Тестовые данные	Результат
1	Отсортировать по цене	Epson 24790 600x600 1630 720 Canon 22350 600x600 1200 2400 Microtek 18911 1200x1200 800 3300 HP 12990 2400x2400 5590 600 Panasonic 12200 600x600 1200 810 Epson 9959 4800x4800 700 1000 Canon 5530 4800x4800 400 500 Canon 4400 2400x2400 1000 300
2	Отсортировать по числу градаций серого	Microtek 18911 1200x1200 800 3300 Canon 22350 600x600 1200 2400 Epson 9959 4800x4800 700 1000 Panasonic 12200 600x600 1200 810 Epson 24790 600x600 1630 720 HP 12990 2400x2400 5590 600 Canon 5530 4800x4800 400 500 Canon 4400 2400x2400 1000 300

Листинги использованных программ

main.cpp:

```
#include <iostream>

#include <limits>

#include <cmath>

#include <fstream>

#include <array>

#include <vector>

#include <algorithm>

#include <sstream>

#include <string>

#include <scanner.hh>

#include <cstring>

using namespace std;



template <typename T>

T read(bool (*predicate)(T, int), int arg = 0)

{

    T x;

    cin >> x;

    while ((cin.fail()) || !predicate(x, arg))

    {

        cin.clear();

        cin.ignore(std::numeric_limits<streamsize>::max(), '\n');

        cin >> x;

    }

    return x;

}
```

```
template <typename T>
bool answer(T a, int b)
{
    return a >= 0 && a <= 3;
}
template <typename T>
bool all(T a, int b)
{
    return true;
}
template <typename T>
bool greather(T a, int arg)
{
    return a > arg;
}
template <typename T>
bool lower(T a, T arg)
{
    return a < arg;
}

template <typename T>
bool shorter(T a,int arg)
{
    return a.size() < arg;
}
```

```
int main()

{

    system("chcp 65001");

    int choise;

    do

    {

         cout << "Введите номер операции:\n1.Вывести\n2.сортировать по\n3.Добавить\n0.Выйти"
<< endl;

        choise = read<int>(answer);

        switch (choise)

        {

        case 1:

            print("Scanners.bin");

            system("pause");

            break;

        case 2:

            {

              cout << "Введите Y если хотите отсортировать по стоимости или что-нибудь еще,
чтобы отсортировать по градации серого" << endl;

            char grad = read<char>(all);

                    sort_by("Scanners.bin",tolower(grad) == 'y' ? scanner_traits::price :
scanner_traits::grey);

            system("pause");

            }

            break;

        case 3:

            {

                vector<scanner> v;

                char should_continue = 'n';
```

```
                do
                {
                    scanner s;
                    memset(&s,0,sizeof(scanner));
                    cout << "Введите название модели" << endl;
                    string model = read<string>(shorter,25);
                    memcpy(s.model,model.c_str(),model.size());
                    cout << "Введите цену" << endl;
                    s.price = read<int>(greather,0);
                    cout << "Введите длину" << endl;
                    s.x_size = read<double>(greather,0);
                    cout << "Введите ширину" << endl;
                    s.y_size = read<double>(greather,0);
                    cout << "Введите разрешение" << endl;
                    s.optr = read<int>(greather,0);
                    cout << "Введите число градаций серого" << endl;
                    s.grey = read<int>(greather,0);
                    v.push_back(s);
                     cout << "Введите Y если хотите внести еще одну запись, или что либо еще
чтобы сохранить изменения в файле" << endl;
                    should_continue = read<char>(all);
                }while(tolower(should_continue) == 'y');
                append("Scanners.bin",v);
                system("pause");
            }
            break;
        default:
```

```
            break;
        }
        system("cls");
    } while (choise);

    return 0;
}
```

scanner.cpp:

```
#include <scanner.hh>
#include <fstream>
#include <algorithm>
#include <iostream>

using namespace std;

void append(std::string filename, std::vector<scanner> v)
{
    fstream file; file.open(filename, ios::binary | ios::ate | ios::in | ios::out);
    if(!file)
    {
        file.open(filename, ios::binary | ios::out);
        short sz = v.size();
        file.write((char *)&sz, sizeof(short));
        file.write((char *)v.data(), sizeof(scanner) * v.size());
        return;
    }
    file.write((char *)v.data(), sizeof(scanner) * v.size());
```

```
    size_t sz = file.tellg();
    sz -= sizeof(short);
    sz /= sizeof(scanner);
    file.seekg(0);
    file.write((char *)&sz, sizeof(short));
}

void sort_by(std::string filename, scanner_traits tr)
{
    fstream file; file.open(filename, ios::binary | ios::in | ios::out);
    if(file)
    {
        short count;
        file.read((char *)&count, sizeof(short));
        vector<scanner> v(count);
        file.read((char *)v.data(), count * sizeof(scanner));
        if (tr == scanner_traits::grey)
        {
            sort(v.begin(), v.end(), [](const scanner &a, const scanner &b) -> bool {
                return a.grey > b.grey;
            });
        }
        else
        {
            sort(v.begin(), v.end(), [](const scanner &a, const scanner &b) -> bool {
                return a.price > b.price;
            });
```

```
        }
        file.seekg(2);
        file.write((char *)v.data(), sizeof(scanner) * v.size());
    }
}

void print(std::string filename)
{
    fstream file = fstream(filename, ios::binary | ios::in | ios::out);
    if(file)
    {
        short count;
        file.read((char *)&count, sizeof(short));
        vector<scanner> v(count);
        file.read((char *)v.data(), count * sizeof(scanner));
        cout << "Записи:" << endl;
        for(auto a : v)
        {
             cout << a.model << " " << a.price << " " << a.x_size << "x" << a.y_size << " "
<< a.optr << " " << a.grey << endl;
        }
    }
}
```

scanner.hh:

```
#if !defined(SCANNER_H)
#define SCANNER_H
```

```
#include <vector>
#include <string>

struct scanner
{
    char model[25];
    int price;
    double x_size;
    double y_size;
    int optr;
    int grey;
};

enum scanner_traits
{
    price,
    grey
};

void append(std::string filename, std::vector<scanner> v);
void sort_by(std::string filename, scanner_traits tr);
void print(std::string filename);

#endif // SCANNER_H
```

Ст. преп. Толасова В.В. _____