МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ “ХПІ”

Кафедра “Обчислювальна техніка та програмування”

Розрахункове завдання з дисципліни «Програмування»

Пояснювальна записка
ЄСПД ГОСТ 19.404-79(СТЗВО-ХПІ-2.01-2018 ССОНП)

Розробники

Виконав:
студент групи КІТ-120Д

Дєнін Б. С.

Перевірив:

Пасько Д. А.

Харків 2021

ЗАТВЕРДЖЕНО
ЄСПД ГОСТ 19.404-79(СТЗВО-ХПІ-2.01-2018 ССОНП)

Розрахункове завдання з дисципліни
«Алгоритми та структури даних»

Пояснювальна записка

Листів 12
ЄСПД ГОСТ 19.404-79(СТЗВО-ХПІ-2.01-2018 ССОНП)

Харків 2021

Вступ

Тема роботи:
Розробка інформаційно-довідкової системи.

Мета роботи:
Закріпити отримані знання з дисципліни «Програмування» шляхом використання типового комплексного завдання.

1. Призначення та галузь застосування;
2. Постановка завдання до розробки;
3. Опис вхідних та вихідних даних;
4. Опис складу технічних та програмних засобів;
5. Список джерел інформації;
6. Додаток, який складається з розробленого коду.

Призначення та галузь застосування

Об'єктно-орієнтоване програмування — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція. Однією з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-х роках, вона не мала широкого застосування до 1990-х, коли розвиток комп'ютерів та комп'ютерних мереж дав змогу писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм.

Постановка завдання до розробки

При виконанні завдання з розробки інформаційно-довідкової системи необхідно виконати наступне:

1. З розділу «Розрахункове завдання/Індивідуальне завдання», відповідно до варіанта завдання, обрати прикладну галузь(варіант 3);
2. Для прикладної галузі розробити розгалужену ієрархію класів, що описана у завданні та складається з одного базового класу та двох спадкоємців. Клас повинен мати перевантажені оператори;
3. Розробити клас-список, що буде включати до себе масив вказівників до базового класу. А також базові методи роботи з списком: а) очистка списку б) відображення списку в) додавання/видалення/оновлення;
4. Розробити клас-контролер, що буде включати колекцію розроблених

класів, та наступні методи роботи з колекцією: а) читання даних з файлу б) запис даних у файл;
5. Розробити клас меню, який має відображати діалогове меню для демонстрації реалізованих функцій класу контролера;
6. Виконати перевірку вхідних даних за допомогою регулярних виразів;
7. Оформити документацію: пояснювальну записку.

Опис вхідних та вихідних даних

Вхідні дані:
1. Чи акустичний інструмент(char acoustic);
2. Фірма(char firm);
3. Рік створення(int age);
4. Смичок(char fiddlestick);
5. Розмір(int size).

Вихідні дані:
1. Список даних про інструмент (Базовий клас);
2. Список даних про скрипку (Спадкоємець 1);
3. Список даних про контрабас (Спадкоємець 2).

Опис складу технічних та програмних засобів

1. Створив базовий клас та двох спадкоємців відповідно до свого індивідуального завдання.
2. Створив функції читання даних з файлу та запис даних у файл (контролер).
3. Створив функції додавання та видалення елементу за вибором користувача.
4. Створив функцію меню для зручного користування програмою.
5. Створив функцію очищення списку.
6. Виконав перевірку вхідних даних за допомогою регулярних виразів.
7. Виконав перевантаження операторів.
8. Створив функцію main з головним меню програми.
9. Код програми показано в додатку А.

Результат роботи програми

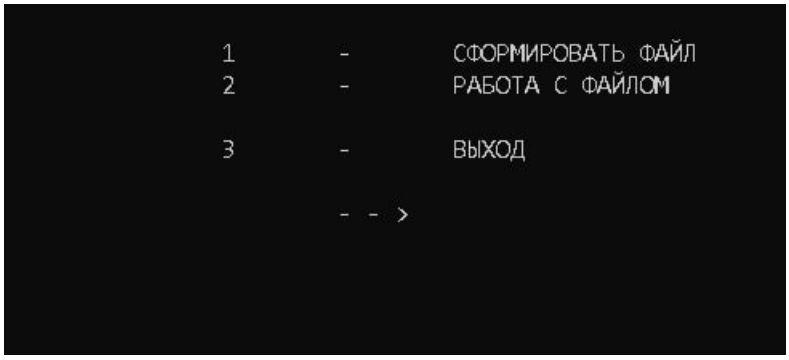

Рисунок 1 – Головне меню

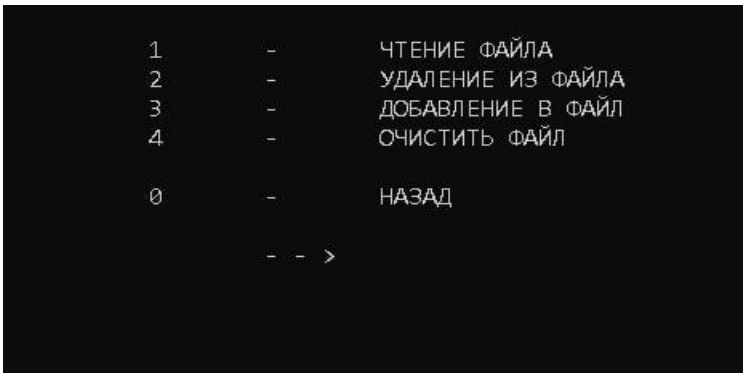

Рисунок 2 – Меню роботи з файлом

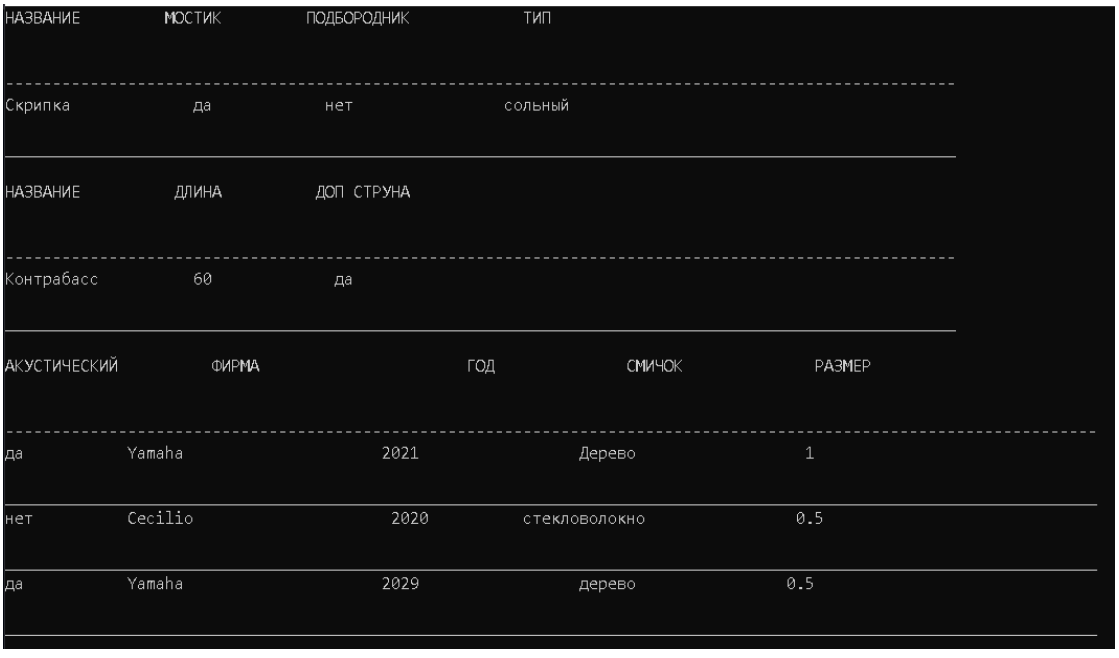

Рисунок 3 – Читання з файлу

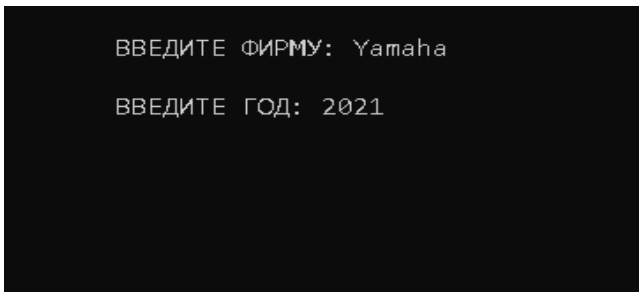

Рисунок 4 – Видалення з файлу

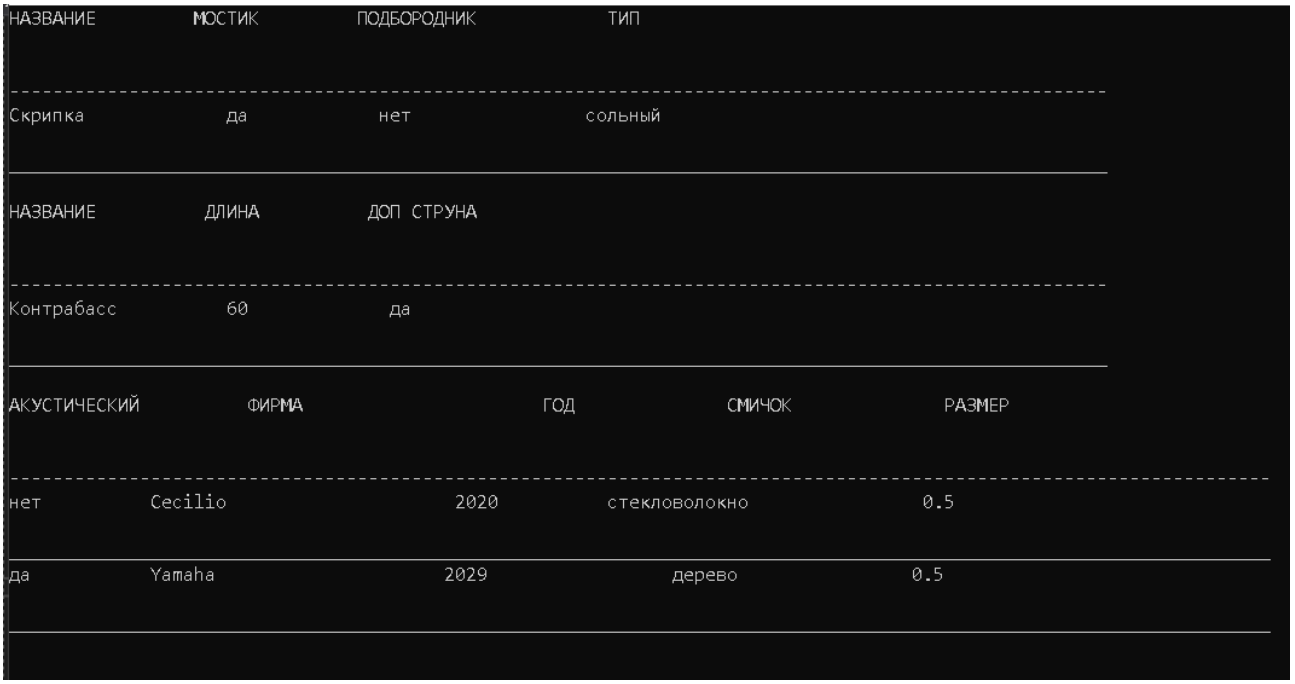

Рисунок 5 – Список після видалення елементу

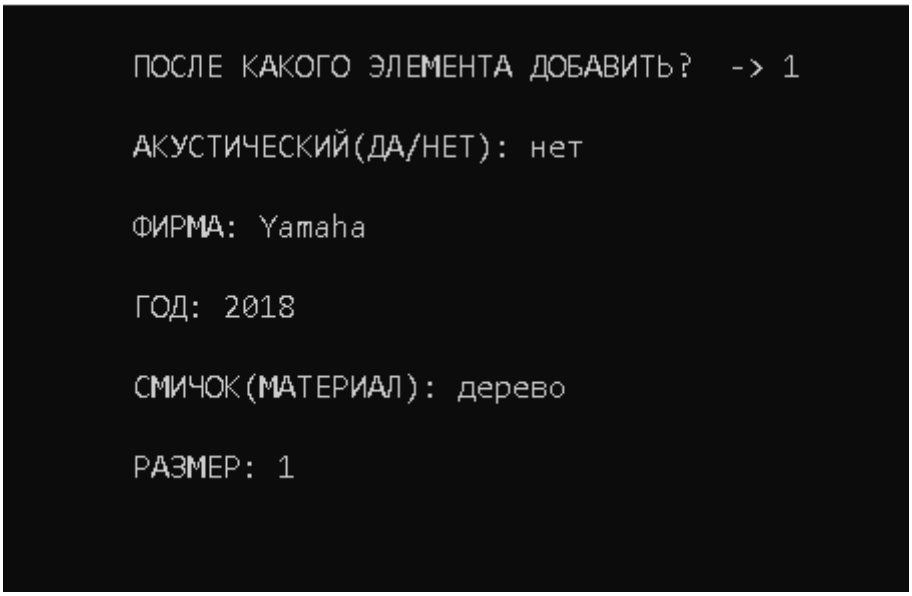

Рисунок 6 – Додавання елементу

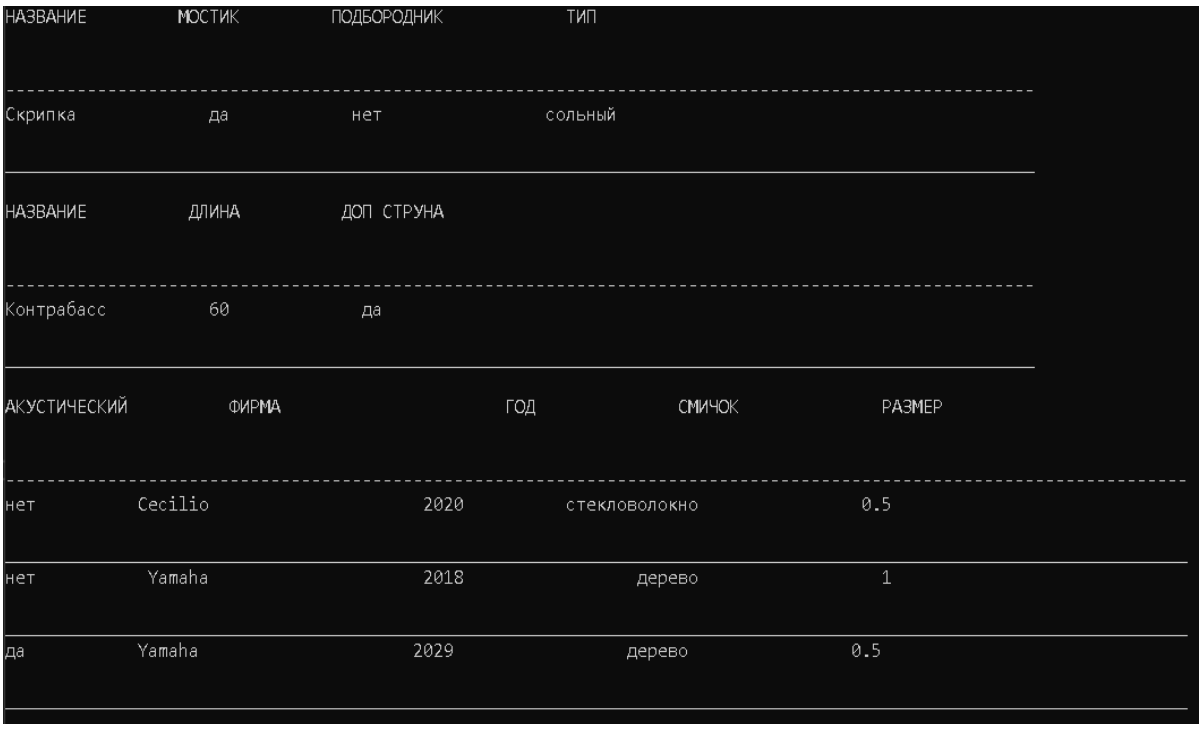

Рисунок 7 – Список після додавання елементу

Висновок

Закріпив отримані знання з дисципліни «Програмування» шляхом використання типового комплексного завдання.

Список джерел інформації

1. Ашарина, И.В. Объектно-ориентированное программирование в C++: лекции и упражнения: Учебное пособие для вузов / И.В. Ашарина. - М.: РиС, 2015. - 336 с.
2. Ашарина, И.В. Язык C++ и объектно-ориентированное программирование в C++. Лабораторный практикум: Учебное пособие для вузов / И.В. Ашарина, Ж.Ф. Крупская. - М.: ГЛТ , 2015. - 232 с.
3. Лафоре, Р. Объектно-ориентированное программирование в C++. Классика Computer Science / Р. Лафоре. - СПб.: Питер, 2013. - 928
4. Лафоре, Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. - СПб.: Питер, 2018. - 928 с.

Додаток А
Текст програми

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <iomanip>
#include <conio.h>
#include <Windows.h>
#include <regex>
#define FILENAME "file.dat"
using namespace std;
class List
{
private:
    char acoustic[50];
    float size;
    char fiddlestick[50];
    char continent[50];
    char firm[50];
    int age;
public:

    std::string nameC;
    std::string nameV;
    List(std::string nameV = "", std::string nameC = "")
        : nameV(nameV), nameC(nameC)
    {
    }
    std::string getName1() const { return nameV; }
    std::string getName2() const { return nameC; }

    static void console_clear();
    void form_file();
    void read_file();
    void work_file();
    void add_List(List t, int pos);
    void delete_from_file(int startAge, int endAge, char* acoustic);
    int clear_file(const char* filename);
    auto create();
    ~List() {}

    friend ostream& operator<< (ostream& out, const List& Zoo);
    friend istream& operator>> (std::istream& in, List& ls);
    List& operator= (const List& ls)
    {
        age = ls.age;
        size = ls.size;

        return *this;
    }

};
class Violin : public List
{
public:
    std::string bridge;
    std::string chin;
    std::string type;

    Violin(std::string bridge = "", std::string chin = "", std::string type = "")
        : bridge(bridge), chin(chin), type(type)
    {
    }
};
class Contrabass : public List
{
public:
    std::string extra_string;
    int length;

    Contrabass(std::string extra_string = "", int length = 0)
        : extra_string(extra_string), length(length)
    {
    }
};
istream& operator>> (istream& in, List& ls)
{
    in >> ls.age;
    in >> ls.size;
```

```
    return in;
}
ostream& operator<< (ostream& out, const List& ls)
{
    out << ls.age << ". " << ls.size << endl;

    return out;
}
auto List::create()
{
    List obj;
    const regex rx("[A-Za-z0-9А-Яа-я_  ,.;:-]");

    cout << "\n\tАКУСТИЧЕСКИЙ(ДА/НЕТ): ";
    char acoustic[50];
    cin >> acoustic;
    if (regex_search(acoustic, rx))
        cout << "";
    else
    {
        cout << "\n\tЗначение введено не правильно, повторите попытку...\n";
        obj.form_file();
    }
    cout << "\n\tФИРМА: ";
    char firm[50];
    cin >> firm;
    if (regex_search(firm, rx))
        cout << "";
    else
    {
        cout << "\n\tЗначение введено не правильно, повторите попытку...\n";
        obj.form_file();
    }
    cout << "\n\tГОД: ";
    int age;
    cin >> age;
    cout << "\n\tСМИЧОК(МАТЕРИАЛ): ";
    char fiddlestick[50];
    cin >> fiddlestick;
    if (regex_search(fiddlestick, rx))
        cout << "";
    else
    {
        cout << "\n\tЗначение введено не правильно, повторите попытку...\n";
        obj.form_file();
    }
    cout << "\n\tРАЗМЕР: ";
    float size;
    cin >> size;
    console_clear();
    List p;
    strcpy_s(p.firm, firm);
    strcpy_s(p.fiddlestick, fiddlestick);
    strcpy_s(p.acoustic, acoustic);
    p.size = size;
    p.age = age;
    return p;
}
void List::add_List(List t, int pos)
{
    if (pos < 1)
        cout << "\n\tНЕКОРРЕКТНЫЙ НОМЕР" << endl;
    else
    {
        FILE* file = fopen("file.dat", "rb");
        FILE* tempfile = fopen("temp.dat", "wb");
        List p;
        int index = 0;
        while (fread(&p, sizeof(List), 1, file))
        {
            fwrite(&p, sizeof(List), 1, tempfile);
            index++;
            if (index == pos)
                fwrite(&t, sizeof(List), 1, tempfile);
        }
        fclose(file);
        fclose(tempfile);
        remove("file.dat");
        rename("temp.dat", "file.dat");
        if (index < pos)
```

```
            cout << "\n\tНЕКОРРЕКТНЫЙ НОМЕР" << endl;
    }
}
void List::form_file()
{
    cout << "\n\tКОЛ-ВО ЭЛЕМЕНТА = ";
    int count;
    cin >> count;
    FILE* file = fopen("file.dat", "wb");
    if (file == NULL)
        exit(1);
    for (int i = 0; i < count; i++)
    {
        List p = create();
        fwrite(&p, sizeof(List), 1, file);
        if (ferror(file))
            exit(2);
    }
    fclose(file);
}
void List::read_file()
{
    FILE* file = fopen("file.dat", "rb");
    List p;

    Violin viol;
    viol.nameV = "Скрипка";
    viol.bridge = "да";
    viol.chin = "нет";
    viol.type = "сольный";

    Contrabass contrab;
    contrab.nameC = "Контрабасс";
    contrab.length = 60;
    contrab.extra_string = "да";

    cout << "НАЗВАНИЕ" << setw(15) << "МОСТИК" << setw(20) << "ПОДБОРОДНИК" << setw(15) << "ТИП"  <<
endl << endl;
    cout << "\n----------------------------------------------------------------------------------------
----------------\n";
    cout << viol.getName1() << setw(15) << viol.bridge << setw(15) << viol.chin << setw(23) <<
viol.type << endl;
    cout <<
"\n_____________________________________________________________________________________________
____\n\n";

    cout << "НАЗВАНИЕ" << setw(15) << "ДЛИНА" << setw(20) << "ДОП СТРУНА" << endl << endl;
    cout << "\n----------------------------------------------------------------------------------------
----------------\n";
    cout << contrab.getName2() << setw(12) << contrab.length << setw(15) << contrab.extra_string  <<
endl;
    cout <<
"\n_____________________________________________________________________________________________
____\n\n";

    cout << "АКУСТИЧЕСКИЙ" << setw(15) << "ФИРМА" << setw(25) << "ГОД" << setw(20) << "СМИЧОК" <<
setw(20) << "РАЗМЕР" <<  endl << endl;
    cout << "\n----------------------------------------------------------------------------------------
------------------------------\n";
    while (fread(&p, sizeof(List), 1, file))
    {
        cout << p.acoustic << setw(17) << p.firm << setw(25) << p.age << setw(23) << p.fiddlestick
<< setw(19) << p.size << endl;
        cout <<
"\n_____________________________________________________________________________________________
_____________________\n";
    }
    fclose(file);
}
int List::clear_file(const char* filename)
{
    FILE* f = NULL;
    if (fopen_s(&f, filename, "wb") != 0)
        return -1;
    fclose(f);
    return 0;
}
void List::delete_from_file(int startAge, int endAge, char* firm)
{
    FILE* file = fopen("file.dat", "rb");
    FILE* tempfile = fopen("temp.dat", "wb");
```

```
    List p;
    bool value = false;
    while (fread(&p, sizeof(List), 1, file))
    {
        if (!((p.age >= startAge && p.age <= endAge) && (strcmp(p.firm, firm) == 0)))
        {
            fwrite(&p, sizeof(List), 1, tempfile);
            value = true;
        }
    }
    fclose(file);
    fclose(tempfile);
    if (value)
    {
        remove("file.dat");
        rename("temp.dat", "file.dat");
    }
}
void List::work_file()
{
    int oper;
    do
    {
        cout << "\n\t\t1\t-\tЧТЕНИЕ ФАЙЛА" << endl;
        cout << "\t\t2\t-\tУДАЛЕНИЕ ИЗ ФАЙЛА" << endl;
        cout << "\t\t3\t-\tДОБАВЛЕНИЕ В ФАЙЛ" << endl;
        cout << "\t\t4\t-\tОЧИСТИТЬ ФАЙЛ" << endl;
        cout << "\n\t\t0\t-\tНАЗАД" << endl << endl;
        cout << "\t\t\t- - >\t";
        cin >> oper;
        console_clear();
        switch (oper)
        {
        case 1:
            read_file();
            break;
        case 2:
        {
            cout << "\n\tВВЕДИТЕ ФИРМУ: ";
            char str[30];
            cin >> str;
            int start;
            cout << "\n\tВВЕДИТЕ ГОД: ";
            cin >> start;
            int end = start;
            delete_from_file(start, end, str);
            console_clear();
            break;
        }
        case 3:
        {
            int pos;
            cout << "\n\tПОСЛЕ КАКОГО ЭЛЕМЕНТА ДОБАВИТЬ?\t -> ";
            cin >> pos;
            List p = create();
            add_List(p, pos);
            break;
        }
        case 4:
        {
            cout << "\n\tВЫ ТОЧНО ХОТИТЕ ОЧИСТИТЬ СПИСОК (+ / ANYKEY)\t";
            if (_getch() == '+')
                cout << "\n\n\tФАЙЛ ОЧИЩЕН\n", clear_file(FILENAME) == 0 ? "" : "НЕ";
            else
                cout << "\n\n\tОТМЕНА";
            cout << "\n\n\n\tANYKEY TO CONTINUE ";
            _getch();

            console_clear();
            break;
        }
        }
    } while (oper != 0);
}
int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    char oper;
    do
    {
```

```
        cout << "\n\t\t1\t-\tСФОРМИРОВАТЬ ФАЙЛ" << endl;
        cout << "\t\t2\t-\tРАБОТА С ФАЙЛОМ" << endl;
        cout << "\n\t\t3\t-\tВЫХОД" << endl << endl;
        cout << "\t\t\t- - >\t";
        cin >> oper;
        List::console_clear();
        List obj;
        switch (oper)
        {
        case '1':
            obj.form_file();
            break;
        case '2':
            obj.work_file();
            break;
        default:
            break;
        }
    } while (oper != 3);
}
void List::console_clear()
{
    COORD coordScreen = { 0, 0 };
    DWORD cCharsWritten;
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    DWORD dwConSize;
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    if (!GetConsoleScreenBufferInfo(hConsole, &csbi))
        return;
    dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
    if (!FillConsoleOutputCharacter(hConsole, (TCHAR)' ',
        dwConSize, coordScreen, &cCharsWritten))
        return;
    if (!GetConsoleScreenBufferInfo(hConsole, &csbi))
        return;
    if (!FillConsoleOutputAttribute(hConsole, csbi.wAttributes,
        dwConSize, coordScreen, &cCharsWritten))
        return;
    SetConsoleCursorPosition(hConsole, coordScreen);
}
```