Московский государственный технический

университет им. Н.Э. Баумана

Проверил:

Преподаватель каф. ИУ5

Подпись и дата:

Факультет «Информатика и система управления»

Кафедра ИУ5 «Система обработки информации и управления»

Выполнил:

Студент группы ИУ5-22Б

Яншин Т. И.

Подпись и дата:

Курс «Программирование на основе классов и шаблонов»

Отчёт по лабораторной работе №2

Переход от структур к классам

**Цель работы:**

Цель данной лабораторной работы заключается в освоении навыков:

* создания программных объектов пользовательских типов с использованием классов;
* создания статических библиотек для описания классов;
* ввода-вывода объектов пользовательских типов в файл;
* перегрузки операций потокового ввода-вывода (<<, >>), операций отношения (<, ==) для пользовательских типов;
* перегрузки конструктора копирования и операции присваивания;
* сортировки массивов объектов пользовательских типов, хранящихся в файле, с использованием перегруженных операций.

## **Задание:**

***Первая часть***

Ознакомьтесь с приведенным материалом в (Введении и Пример 1) о переходе от структуры к классу.

Разработайте класс "Планета" для планет солнечной системы (4 характеристики планет разного типа приведены в Приложении 1).

Память для строк должна выделяться динамически в зависимости от данных (размер выделяемой памяти для поля типа char\* не должен задаваться константной). Размер выделяемой памяти для поля char\* необходимо определять в зависимости от поступающих данных. Для промежуточного хранения строк с целью считать из файла или с потока cin можно воспользоваться массивом char с заданной длинной через константу (буфер). Далее на основе определения длинны строки, попавшей в буфер, выделить динамически память для поля char\*.

Не забудьте воспользоваться деконструктором, если вы будете использовать динамический массив для строк. Использование деконструктора показано в примере 3.

Создайте статическую библиотеку для класса "Планета" и отладить программу, которая создает один объект класса "Планета" и выводит значения его полей на экран. Пример программы приведен в Примере 4.

Реализуйте функции работы с множеством объектов класса "Планета":

чтение БД из файла;

запись БД в файл;

сортировка БД;

добавление нового объекта в БД;

удаление объекта из БД;

редактирование БД;

вывод БД на экран.

Для реализации вышеперечисленных функций, необходимо:

Создать текстовый файл (в блокноте) с данными о планетах солнечной системы и сохранить его в папке проекта. Первоначально, для отладки, введите две записи.

***Вторая часть***

Ознакомьтесь с примером 2 перегрузки операции <<.

Перегрузите конструктор копирования, деструктор и оператор присваивания.

Вставьте в конструкторы и деструктор печать типа "Создание (Удаление) ID n", где n - номер объекта, для которого они вызываются (при реализации данного пункта может быть полезен пример 3).

Перегрузите операцию >> для класса "Планета" и ifstream и прочитайте данные о планетах из файла в массив "Солнечная система" из объектов класса "Планета".

Перегрузите операцию << для классов "Планета" и ofstream и выведите на экран данные из массива.

***Третья часть***

Перегрузите операции сравнения < и == для класса "Планета", использовав для этого значение одного из полей.

Отсортируйте массив планет солнечной системы, хранящийся в файле, с использованием перегруженных операций.

Четвертая часть

На основе разработанного класса "Планета" выполните задание по варианту (не менее 4 характеристик в классе разного типа). Варианты представлены в приложении 2.

sunsys::Planets\* planets = nullptr; // Нулевой указатель на массив объектов БД

const int N = 100; // Макс. Число объектов в БД (для проверки)

int n\_planet = 0; // Фактическое число объектов

const char\* file\_name = "sunsys.txt"; // Имя файла с данными

int current\_index = 0; // Индекс для обращения к объекту

char find\_name[20]; // Имя объекта для поиска по БД

Комментарии к функциям – в определениях функций

## **Текст Программы**

main.cpp

#include "planets/planets.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

sunsys::Planets\* planets = nullptr;

const int N = 100;

int n\_planet = 0;

const char\* file\_name = "sunsys.txt";

int current\_index = 0;

char find\_name[20];

while (true)

{

switch (sunsys::menu())

{

case 1: // Чтение из файла

planets = sunsys::CreateDynArray(file\_name, n\_planet);

sunsys::read\_dbase(file\_name, planets, n\_planet, N);

break;

case 2: // Запись в файл

sunsys::write\_dbase(file\_name, planets, n\_planet);

break;

case 3: // Редактирование (поиск по имени планеты)

cout << "Введите имя необходимой для планеты: ";

cin >> find\_name;

current\_index = sunsys::find(planets, n\_planet, find\_name);

sunsys::edit\_object(planets, current\_index);

break;

case 4: // Вывод на экран

sunsys::print\_dbase(planets, n\_planet);

break;

case 5: // Удаление планеты

cout << "Введите имя объекта для УДАЛЕНИЯ: ";

cin >> find\_name;

current\_index = sunsys::find(planets, n\_planet, find\_name);

sunsys::remove\_planet(planets, n\_planet, current\_index);

break;

case 6: // Сортировка

int key, order;

sunsys::prepare\_sort(key, order);

sunsys::sort\_planets(planets, n\_planet, key, order);

break;

case 7: // Добавить новую планету

planets = sunsys::add\_planet(planets, n\_planet, N);

break;

case 8: // Перегрузка сортировка

sunsys::sort\_planets\_overload(planets, n\_planet);

break;

case 0: // Выход

delete[] planets;

return 0;

default:

cout << " Неправильный ввод" << endl;

break;

}

}

delete[] planets;

return 0;

}

Planets.cpp

#include "planets.h"

using namespace std;

// Меню

int sunsys::menu()

{

std::cout << std::endl;

std::cout << " ======================== ГЛАВНОЕ МЕНЮ ========================\n";

std::cout << "1 - Чтение БД из файла\t\t 4 - Вывод БД на экран" << std::endl;

std::cout << "2 - Запись БД в файл\t\t 5 - Удаление из БД" << std::endl;

std::cout << "3 - Редактирование\t\t 6 - Сортировка" << std::endl;

std::cout << "7 - Добавить новый объект\t 8 - Cортировка перегрузкой" << std::endl;

std::cout << "0 - ВЫХОД \t\t 0 - ВЫХОД" << std::endl;

std::cout << "Для выбора операции введите цифру от 0 до 8" << std::endl;

int resp;

std::cin >> resp;

std::cin.clear();

std::cin.ignore(10, '\n');

return resp;

}

// Создание динамического массива

sunsys::Planets\* sunsys::CreateDynArray(const char\* filename, int& n)

{

ifstream fin(filename, ios::in);

if (!fin)

{

cout << "Heт файла " << filename << endl;

exit(1);

}

else

{

fin >> n;

sunsys::Planets\* planets = new sunsys::Planets[n];

return planets;

}

}

// Чтение БД из файла

void sunsys::read\_dbase(const char\* filename, sunsys::Planets\* arr, int n, const int N)

{

int num;

ifstream fin(filename, ios::in);

if (!fin)

{

cout << "Heт файла " << filename << endl;

}

fin >> num;

if (n > N)

{

cout << "Переполнение БД. n= " << n << endl;

}

for (int i = 0; i < n; i++)

{

fin >> arr[i];

}

fin.close();

cout << "Чтение завершено" << endl;

}

// Вывод базы в файл

int sunsys::write\_dbase(const char\* filename, sunsys::Planets const\* arr, int n)

{

ofstream fout(filename, ios::out);

if (!fout)

{

cout << "Ошибка открытия файла" << endl;

return 1;

}

fout << n << '\n';

for (int i = 0; i < n; i++)

{

fout << arr[i];

}

cout << "Успешно записано в файл" << endl;

fout.close();

return 0;

}

// Вывод базы на экран

void sunsys::print\_dbase(sunsys::Planets const\* arr, int n)

{

cout << " База Данных " << endl;

for (int i = 0; i < n; i++)

{

cout << setw(3) << i + 1 << ". " << arr[i];

}

}

// Поиск планеты по имени

int sunsys::find(sunsys::Planets\* arr, int n, const char\* name)

{

int ind = -1;

for (int i = 0; i < n; i++)

{

if (!strcmp(arr[i].GetName(), name)) // Когда имя совпало

{

cout << name << " найдено в БД" << endl;

ind = i;

}

}

return ind;

}

// Удаление объекта по индексу

void sunsys::remove\_planet(sunsys::Planets\* arr, int& n, int index)

{

if (index < 0 or index >= n)

{

cout << "Ошибка: Объект не найден. Убедитесь, что БД считана " << endl;

return;

}

// Перемещаем все элементы после удаляемого на одну позицию влево

for (int i = index; i < n - 1; i++)

{

arr[i] = arr[i + 1];

}

n--; // Уменьшаем размер массива

cout << "Объект удален из БД" << endl;

}

// Редактирование

void sunsys::edit\_object(Planets\* arr, int index)

{

if (index < 0)

{

cout << "Ошибка: Объект не найден. Убедитесь, что БД считана " << endl;

return;

}

int choice;

char input[50];

do

{

cout << "========================================================================" << endl;

cout << setw(3) << index + 1 << ". " << arr[index].GetName() << setw(10 - strlen(arr[index].GetName()) + 6)

<< arr[index].GetDiam() << setw(10) << arr[index].GetLife() << setw(10) << arr[index].GetSate() << endl;

std::cout << " ======================== МЕНЮ РЕДАКТИРОВАНИЯ ========================\n";

std::cout << "1 - Редактировать имя" << std::endl;

std::cout << "2 - Редактировать диаметр" << std::endl;

std::cout << "3 - Редактировать наличие жизни" << std::endl;

std::cout << "4 - Редактировать кол-во спутников" << std::endl;

std::cout << "0 - ЗАКОНЧИТЬ редактирвоание" << std::endl;

std::cout << "Для выбора операции введите цифру от 1 до 5" << std::endl;

std::cin >> choice;

switch (choice)

{

case 1:

std::cout << "Enter new name: ";

std::cin >> input;

arr[index].SetName(input);

break;

case 2:

std::cout << "Enter new diameter: ";

std::cin >> input;

arr[index].SetDiam(atoi(input));

break;

case 3:

std::cout << "Enter new life status (0 for false, 1 for true): ";

std::cin >> input;

arr[index].SetLife(atoi(input));

break;

case 4:

std::cout << "Enter new number of satellites: ";

std::cin >> input;

arr[index].SetSate(atoi(input));

break;

case 0:

std::cout << "Выход из меню редактирования" << endl;

break;

default:

cout << " Неправильный ввод" << endl;

break;

}

} while (choice != 0);

}

// Сортировка пузырьком

void sunsys::sort\_planets(Planets\* arr, int size, int key, int order)

{

sunsys::Planets temp;

bool stop = 0;

for (int i = 0; i < size - 1; i++)

{

for (int j = 0; j < size - i - 1; j++)

{

switch (key)

{

case 1:

if (order == 0)

{

if (strcmp(arr[j].GetName(), arr[j + 1].GetName()) > 0)

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else

{

if (strcmp(arr[j].GetName(), arr[j + 1].GetName()) < 0)

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

break;

case 2:

if (order == 0)

{

if (arr[j].GetDiam() > arr[j + 1].GetDiam())

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else

{

if (arr[j].GetDiam() < arr[j + 1].GetDiam())

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

break;

case 3:

if (order == 0)

{

if (int(arr[j].GetLife()) > int(arr[j + 1].GetLife()))

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else

{

if (int(arr[j].GetLife()) < int(arr[j + 1].GetLife()))

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

break;

case 4:

if (order == 0)

{

if (arr[j].GetSate() > arr[j + 1].GetSate())

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else

{

if (arr[j].GetSate() < arr[j + 1].GetSate())

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

break;

case 0:

stop = 1;

cout << "Выход в ГЛАВНОЕ МЕНЮ, сортировка ОТМЕНЕНА" << endl;

break;

}

if (stop)

{

break;

}

}

if (stop)

{

break;

}

}

if (key != 0)

{

cout << "Отсортировано по выбранным условиям, выход в ГЛАВНОЕ МЕНЮ" << endl;

}

}

// Меню сортировки

void sunsys::prepare\_sort(int& key, int& order)

{

bool ready = 0;

do

{

std::cout << std::endl;

std::cout << " ======================== МЕНЮ СОРТИРОВКИ ========================\n";

std::cout << "Выберите ключ сортировки" << std::endl;

std::cout << "1 - Имя" << std::endl;

std::cout << "2 - Диаметр" << std::endl;

std::cout << "3 - Наличие жизни" << std::endl;

std::cout << "4 - Кол-во спутников" << std::endl;

std::cout << "0 - ВЫЙТИ из меню сортировки" << std::endl;

std::cout << "Для выбора операции введите цифру от 0 до 4" << std::endl;

std::cin >> key;

if (key == 0)

{

break;

}

if (key > 4 or key < 0)

{

std::cout << "Некорректный ввод, возвращение в меню сортировки: повторите попытку" << std::endl;

continue;

}

std::cout << "Выберите порядок сортировки" << std::endl;

std::cout << "0 - по возрастанию, 1 - по убыванию" << std::endl;

std::cin >> order;

if (order > 1 or order < 0)

{

std::cout << "Некорректный ввод, возвращение в меню сортировки: повторите попытку" << std::endl;

continue;

}

ready = 1;

} while (ready != 1);

}

// Добавить элемент

sunsys::Planets\* sunsys::add\_planet(sunsys::Planets\* &arr, int& n, const int N)

{

if (n < N)

{

char name[100];

int diam{};

bool life{};

int satellite{};

std::cout << "Введите имя планеты:";

std::cin >> name;

while (std::cout << "Введите диаметр планеты: " and !(std::cin >> diam) and !diam)

{

std::cin.clear(); // Удалить некорректный ввод

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Отменить ввод

std::cout << "Неверный ввод\n";

}

while (std::cout << "Наличие жизни (1 - да, 0 - нет): " and !(std::cin >> life))

{

std::cin.clear(); // Удалить некорректный ввод

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Отменить ввод

std::cout << "Неверный ввод\n";

}

while (std::cout << "Количество спутников: " and !(std::cin >> satellite))

{

std::cin.clear(); // Удалить некорректный ввод

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Отменить ввод

std::cout << "Неверный ввод\n";

}

sunsys::Planets\* newPlanets = new sunsys::Planets[n + 1];

for (int i = 0; i < n; ++i)

{

newPlanets[i] = arr[i];

}

delete[] arr;

arr = newPlanets;

arr[n].SetName(name);

arr[n].SetDiam(diam);

arr[n].SetLife(life);

arr[n].SetSate(satellite);

n += 1;

return arr;

}

else

{

std::cout << "База данных переполнена" << std::endl;

return arr;

}

}

// Сортировка с использованием перегрузки

void sunsys::sort\_planets\_overload(Planets\* arr, int n)

{

sunsys::Planets temp;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (arr[j] < arr[j + 1])

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

Planets.h

#pragma once

#include <iostream>

#include <fstream>

#include <iomanip>

#include <cstring>

namespace sunsys

{

class Planets

{

private:

char\* name; // Имя

int diam; // Диаметр

bool life; // Наличие жизни

int satellite; // Спутники

static int total;

int id;

public:

Planets()

{

name = nullptr;

diam = life = satellite = 0;

total++; // увеличить счетчик объектов

id = total; // id равен текущему значению total

std::cout << "Создание ID " << id << std::endl;

}

Planets(char\* planetName, int diameter, bool hasLife, int numSatellites, int idf) :

name(planetName), diam(diameter), life(hasLife), satellite(numSatellites), id(idf) {}

~Planets()

{

delete[] name;

total--;

std::cout << "Удаление ID " << id << std::endl;

}

Planets& operator=(const Planets& other)

{

if (this != &other)

{

delete[] name; // Освобождаем память для текущей строки name

name = new char[strlen(other.name) + 1]; // Выделяем память для новой строки name

strcpy(name, other.name); // Копируем значение из строки other.name в строку name

// Копирование других полей класса

diam = other.diam;

life = other.life;

satellite = other.satellite;

id = other.id;

}

return \*this;

}

friend std::ostream& operator<<(std::ostream& out, const Planets& obj)

{

out << obj.name << std::setw(20 - strlen(obj.name) + 6) << obj.diam << std::setw(10) << obj.life << std::setw(10) << obj.satellite << '\n';

return out;

}

friend std::istream& operator>>(std::istream& in, Planets& planet)

{

const int initialSize = 5; // Начальный размер для имени планеты

char\* temp\_name = new char[initialSize];

int size = initialSize;

int length = 0;

char c;

in.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

while (in.get(c) and c != ' ' and c != '\t')

{

temp\_name[length] = c;

length++;

if (length == size)

{

size \*= 2;

char\* temp = new char[size];

for (int j = 0; j < length; j++)

{

temp[j] = temp\_name[j];

}

delete[] temp\_name;

temp\_name = temp;

}

}

temp\_name[length] = '\0'; // Конец строки

delete[] planet.name;

planet.name = temp\_name;

in >> planet.diam >> planet.life >> planet.satellite;

return in;

}

// Перегрузки для сортировки

bool operator<(const Planets& other) const

{

return diam < other.diam;

}

bool operator==(const Planets& other) const

{

return diam == other.diam;

}

char\* GetName() { return name; }

int GetDiam() const { return diam; }

bool GetLife() const { return life; }

int GetSate() const { return satellite; }

void SetName(const char\* name)

{

delete[] this->name;

this->name = new char[strlen(name) + 1];

strncpy(this->name, name, strlen(name) + 1);

}

void SetDiam(int diam) { this->diam = diam; }

void SetLife(bool life) { this->life = life; }

void SetSate(int satellite) { this->satellite = satellite; }

};

int menu();

void read\_dbase(const char\* filename, sunsys::Planets\* arr, int n, const int N);

int write\_dbase(const char\* filename, sunsys::Planets const\* arr, int n);

void print\_dbase(sunsys::Planets const\* arr, int n);

int find(sunsys::Planets\* arr, int n, const char\* name);

void remove\_planet(sunsys::Planets\* arr, int& n, int index);

void edit\_object(Planets\* arr, int index);

void sort\_planets(Planets\* arr, int size, int key, int order);

void prepare\_sort(int& key, int& order);

void sort\_planets\_overload(Planets\* arr, int n);

Planets\* add\_planet(sunsys::Planets\* &arr, int& n, const int N);

Planets\* CreateDynArray(const char\* filename, int& n);

}

CMakeLists

# CMakeList.txt: файл проекта CMake верхнего уровня;

cmake\_minimum\_required (VERSION 3.8)

set(project "lab2")

project(**${project}**)

set(CMAKE\_CXX\_STANDARD 23)

set(**${project}**\_SOURCES

main.cpp)

add\_subdirectory(planets)

set(**${project}**\_SOURCE\_LIST

${**${project}**\_SOURCES})

add\_executable(**${project}**

${**${project}**\_SOURCE\_LIST} )

configure\_file(**${CMAKE\_CURRENT\_SOURCE\_DIR}**/sunsys.txt

**${CMAKE\_CURRENT\_BINARY\_DIR}** COPYONLY)

target\_link\_libraries(**${project}** planets)

cmake\_minimum\_required(VERSION 3.8)

set(project "planets")

project(**${project}**)

set(CMAKE\_CXX\_STANDARD 23)

set(**${project}**\_SOURCES

planets.cpp)

set(**${project}**\_HEADERS

planets.h)

set(**${project}**\_SOURCE\_LIST

${**${project}**\_SOURCES}

${**${project}**\_HEADERS})

add\_library(**${project}**

STATIC

${**${project}**\_SOURCE\_LIST})

***Задание по вариантам “Автобус”***

Добавлено в Cmakelists

<...>

# CMakeList.txt: файл проекта CMake верхнего уровня;

configure\_file(**${CMAKE\_CURRENT\_SOURCE\_DIR}**/busses.txt

**${CMAKE\_CURRENT\_BINARY\_DIR}** COPYONLY)

target\_link\_libraries(**${project}** busses)

<...>

set(**${project}**\_SOURCES

busses.cpp)

set(**${project}**\_HEADERS

busses.h)

<...>

Busses.h

#pragma once

#include <iostream>

#include <fstream>

#include <iomanip>

#include <cstring>

namespace depo

{

class Bus

{

private:

char\* name; // Марка производителя

int passangers; // Кол-во пассажиров

bool tour; // Туристический / рейсовый

int year; // Год производства

static int total;

int id;

public:

Bus()

{

name = nullptr;

passangers = tour = year = 0;

total++; // увеличить счетчик объектов

id = total; // id равен текущему значению total

std::cout << "Создание ID " << id << std::endl;

}

Bus(char\* busName, int pass, bool tourist, int Year, int idf) :

name(busName), passangers(pass), tour(tourist), year(Year), id(idf) {}

~Bus()

{

delete[] name;

total--;

std::cout << "Удаление ID " << id << std::endl;

}

Bus& operator=(const Bus& other)

{

if (this != &other)

{

delete[] name; // Освобождаем память для текущей строки name

name = new char[strlen(other.name) + 1]; // Выделяем память для новой строки name

strcpy(name, other.name); // Копируем значение из строки other.name в строку name

// Копирование других полей класса

passangers = other.passangers;

tour = other.tour;

year = other.year;

id = other.id;

}

return \*this;

}

friend std::ostream& operator<<(std::ostream& out, const Bus& obj)

{

out << obj.name << std::setw(20 - strlen(obj.name) + 6) << obj.passangers << std::setw(10) << obj.tour << std::setw(10) << obj.year << '\n';

return out;

}

friend std::istream& operator>>(std::istream& in, Bus& planet)

{

const int initialSize = 5; // Начальный размер для имени

char\* temp\_name = new char[initialSize];

int size = initialSize;

int length = 0;

char c;

in.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

while (in.get(c) and c != ' ' and c != '\t')

{

temp\_name[length] = c;

length++;

if (length == size)

{

size \*= 2;

char\* temp = new char[size];

for (int j = 0; j < length; j++)

{

temp[j] = temp\_name[j];

}

delete[] temp\_name;

temp\_name = temp;

}

}

temp\_name[length] = '\0'; // Конец строки

delete[] planet.name;

planet.name = temp\_name;

in >> planet.passangers >> planet.tour >> planet.year;

return in;

}

// Перегрузки для сортировки

bool operator<(const Bus& other) const

{

return passangers < other.passangers;

}

bool operator==(const Bus& other) const

{

return passangers == other.passangers;

}

char\* GetName() { return name; }

int GetDiam() const { return passangers; }

bool GetLife() const { return tour; }

int GetSate() const { return year; }

void SetName(const char\* name)

{

delete[] this->name;

this->name = new char[strlen(name) + 1];

strncpy(this->name, name, strlen(name) + 1);

}

void SetDiam(int pas) { this->passangers = pas; }

void SetLife(bool tour) { this->tour = tour; }

void SetSate(int yo) { this->year = yo; }

};

int menu();

void read\_dbase(const char\* filename, depo::Bus\* arr, int n, const int N);

int write\_dbase(const char\* filename, depo::Bus const\* arr, int n);

void print\_dbase(depo::Bus const\* arr, int n);

int find(depo::Bus\* arr, int n, const char\* name);

void remove\_obj(depo::Bus\* arr, int& n, int index);

void edit\_object(Bus\* arr, int index);

void sort\_objects(Bus\* arr, int size, int key, int order);

void prepare\_sort(int& key, int& order);

void sort\_overload(Bus\* arr, int n);

Bus\* add\_obj(depo::Bus\* &arr, int& n, const int N);

Bus\* CreateDynArray(const char\* filename, int& n);

}

Main.cpp

#include "busses/busses.h"

using namespace std;

int depo::Bus::total = 0;

int main()

{

setlocale(LC\_ALL, "Russian");

depo::Bus\* busses = nullptr;

const int N = 100;

int n\_bus = 0;

const char\* file\_name = "busses.txt";

int current\_index = 0;

char find\_name[20];

while (true)

{

switch (depo::menu())

{

case 1: // Чтение из файла

busses = depo::CreateDynArray(file\_name, n\_bus);

depo::read\_dbase(file\_name, busses, n\_bus, N);

break;

case 2: // Запись в файл

depo::write\_dbase(file\_name, busses, n\_bus);

break;

case 3: // Редактирование (поиск по имени планеты)

cout << "Введите название марки автобуса: ";

cin >> find\_name;

current\_index = depo::find(busses, n\_bus, find\_name);

depo::edit\_object(busses, current\_index);

break;

case 4: // Вывод на экран

depo::print\_dbase(busses, n\_bus);

break;

case 5: // Удаление планеты

cout << "Введите имя объекта для УДАЛЕНИЯ: ";

cin >> find\_name;

current\_index = depo::find(busses, n\_bus, find\_name);

depo::remove\_obj(busses, n\_bus, current\_index);

break;

case 6: // Сортировка

int key, order;

depo::prepare\_sort(key, order);

depo::sort\_objects(busses, n\_bus, key, order);

break;

case 7: // Добавить

busses = depo::add\_obj(busses, n\_bus, N);

break;

case 8: // Перегрузка сортировка

depo::sort\_overload(busses, n\_bus);

break;

case 0: // Выход

delete[] busses;

return 0;

default:

cout << " Неправильный ввод" << endl;

break;

}

}

delete[] busses;

return 0;

}

Busses.cpp

#include "busses.h"

using namespace std;

// Меню

int depo::menu()

{

std::cout << std::endl;

std::cout << " ======================== ГЛАВНОЕ МЕНЮ ========================\n";

std::cout << "1 - Чтение БД из файла\t\t 4 - Вывод БД на экран" << std::endl;

std::cout << "2 - Запись БД в файл\t\t 5 - Удаление из БД" << std::endl;

std::cout << "3 - Редактирование\t\t 6 - Сортировка" << std::endl;

std::cout << "7 - Добавить новый объект\t 8 - Cортировка перегрузкой" << std::endl;

std::cout << "0 - ВЫХОД \t\t 0 - ВЫХОД" << std::endl;

std::cout << "Для выбора операции введите цифру от 0 до 8" << std::endl;

int resp;

std::cin >> resp;

std::cin.clear();

std::cin.ignore(10, '\n');

return resp;

}

// Создание динамического массива

depo::Bus\* depo::CreateDynArray(const char\* filename, int& n)

{

ifstream fin(filename, ios::in);

if (!fin)

{

cout << "Heт файла " << filename << endl;

exit(1);

}

else

{

fin >> n;

depo::Bus\* planets = new depo::Bus[n];

return planets;

}

}

// Чтение БД из файла

void depo::read\_dbase(const char\* filename, depo::Bus\* arr, int n, const int N)

{

int num;

ifstream fin(filename, ios::in);

if (!fin)

{

cout << "Heт файла " << filename << endl;

}

fin >> num;

if (n > N)

{

cout << "Переполнение БД. n= " << n << endl;

}

for (int i = 0; i < n; i++)

{

fin >> arr[i];

}

fin.close();

cout << "Чтение завершено" << endl;

}

// Вывод базы в файл

int depo::write\_dbase(const char\* filename, depo::Bus const\* arr, int n)

{

ofstream fout(filename, ios::out);

if (!fout)

{

cout << "Ошибка открытия файла" << endl;

return 1;

}

fout << n << '\n';

for (int i = 0; i < n; i++)

{

fout << arr[i];

}

cout << "Успешно записано в файл" << endl;

fout.close();

return 0;

}

// Вывод базы на экран

void depo::print\_dbase(depo::Bus const\* arr, int n)

{

cout << " База Данных " << endl;

for (int i = 0; i < n; i++)

{

cout << setw(3) << i + 1 << ". " << arr[i];

}

}

// Поиск по имени

int depo::find(depo::Bus\* arr, int n, const char\* name)

{

int ind = -1;

for (int i = 0; i < n; i++)

{

if (!strcmp(arr[i].GetName(), name)) // Когда имя совпало

{

cout << name << " найдено в БД" << endl;

ind = i;

}

}

return ind;

}

// Удаление объекта по индексу

void depo::remove\_obj(depo::Bus\* arr, int& n, int index)

{

if (index < 0 or index >= n)

{

cout << "Ошибка: Объект не найден. Убедитесь, что БД считана " << endl;

return;

}

// Перемещаем все элементы после удаляемого на одну позицию влево

for (int i = index; i < n - 1; i++)

{

arr[i] = arr[i + 1];

}

n--; // Уменьшаем размер массива

cout << "Объект удален из БД" << endl;

}

// Редактирование

void depo::edit\_object(Bus\* arr, int index)

{

if (index < 0)

{

cout << "Ошибка: Объект не найден. Убедитесь, что БД считана " << endl;

return;

}

int choice;

char input[50];

do

{

cout << "========================================================================" << endl;

cout << setw(3) << index + 1 << ". " << arr[index].GetName() << setw(10 - strlen(arr[index].GetName()) + 6)

<< arr[index].GetDiam() << setw(10) << arr[index].GetLife() << setw(10) << arr[index].GetSate() << endl;

std::cout << " ======================== МЕНЮ РЕДАКТИРОВАНИЯ ========================\n";

std::cout << "1 - Редактировать имя" << std::endl;

std::cout << "2 - Редактировать кол-во мест пассажиров" << std::endl;

std::cout << "3 - Редактировать тип тур/рейс" << std::endl;

std::cout << "4 - Редактировать год производства" << std::endl;

std::cout << "0 - ЗАКОНЧИТЬ редактирвоание" << std::endl;

std::cout << "Для выбора операции введите цифру от 1 до 5" << std::endl;

std::cin >> choice;

switch (choice)

{

case 1:

std::cout << "Enter new name: ";

std::cin >> input;

arr[index].SetName(input);

break;

case 2:

std::cout << "Enter new num of passangers: ";

std::cin >> input;

arr[index].SetDiam(atoi(input));

break;

case 3:

std::cout << "Enter new tour status (0 for false, 1 for true): ";

std::cin >> input;

arr[index].SetLife(atoi(input));

break;

case 4:

std::cout << "Enter new year: ";

std::cin >> input;

arr[index].SetSate(atoi(input));

break;

case 0:

std::cout << "Выход из меню редактирования" << endl;

break;

default:

cout << " Неправильный ввод" << endl;

break;

}

} while (choice != 0);

}

// Сортировка пузырьком

void depo::sort\_objects(Bus\* arr, int size, int key, int order)

{

depo::Bus temp;

bool stop = 0;

for (int i = 0; i < size - 1; i++)

{

for (int j = 0; j < size - i - 1; j++)

{

switch (key)

{

case 1:

if (order == 0)

{

if (strcmp(arr[j].GetName(), arr[j + 1].GetName()) > 0)

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else

{

if (strcmp(arr[j].GetName(), arr[j + 1].GetName()) < 0)

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

break;

case 2:

if (order == 0)

{

if (arr[j].GetDiam() > arr[j + 1].GetDiam())

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else

{

if (arr[j].GetDiam() < arr[j + 1].GetDiam())

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

break;

case 3:

if (order == 0)

{

if (int(arr[j].GetLife()) > int(arr[j + 1].GetLife()))

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else

{

if (int(arr[j].GetLife()) < int(arr[j + 1].GetLife()))

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

break;

case 4:

if (order == 0)

{

if (arr[j].GetSate() > arr[j + 1].GetSate())

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else

{

if (arr[j].GetSate() < arr[j + 1].GetSate())

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

break;

case 0:

stop = 1;

cout << "Выход в ГЛАВНОЕ МЕНЮ, сортировка ОТМЕНЕНА" << endl;

break;

}

if (stop)

{

break;

}

}

if (stop)

{

break;

}

}

if (key != 0)

{

cout << "Отсортировано по выбранным условиям, выход в ГЛАВНОЕ МЕНЮ" << endl;

}

}

// Меню сортировки

void depo::prepare\_sort(int& key, int& order)

{

bool ready = 0;

do

{

std::cout << std::endl;

std::cout << " ======================== МЕНЮ СОРТИРОВКИ ========================\n";

std::cout << "Выберите ключ сортировки" << std::endl;

std::cout << "1 - Имя" << std::endl;

std::cout << "2 - Кол-во пассажиров" << std::endl;

std::cout << "3 - Туристический/Рейсовый" << std::endl;

std::cout << "4 - Год производства" << std::endl;

std::cout << "0 - ВЫЙТИ из меню сортировки" << std::endl;

std::cout << "Для выбора операции введите цифру от 0 до 4" << std::endl;

std::cin >> key;

if (key == 0)

{

break;

}

if (key > 4 or key < 0)

{

std::cout << "Некорректный ввод, возвращение в меню сортировки: повторите попытку" << std::endl;

continue;

}

std::cout << "Выберите порядок сортировки" << std::endl;

std::cout << "0 - по возрастанию, 1 - по убыванию" << std::endl;

std::cin >> order;

if (order > 1 or order < 0)

{

std::cout << "Некорректный ввод, возвращение в меню сортировки: повторите попытку" << std::endl;

continue;

}

ready = 1;

} while (ready != 1);

}

// Добавить элемент

depo::Bus\* depo::add\_obj(depo::Bus\* &arr, int& n, const int N)

{

if (n < N)

{

char name[100];

int diam{};

bool life{};

int satellite{};

std::cout << "Введите имя (марку) автобуса:";

std::cin >> name;

while (std::cout << "Введите кол-во мест пассажиров: " and !(std::cin >> diam) and !diam)

{

std::cin.clear(); // Удалить некорректный ввод

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Отменить ввод

std::cout << "Неверный ввод\n";

}

while (std::cout << "Туристический (1 - да, 0 - нет): " and !(std::cin >> life))

{

std::cin.clear(); // Удалить некорректный ввод

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Отменить ввод

std::cout << "Неверный ввод\n";

}

while (std::cout << "Год производства: " and !(std::cin >> satellite))

{

std::cin.clear(); // Удалить некорректный ввод

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Отменить ввод

std::cout << "Неверный ввод\n";

}

depo::Bus\* newPlanets = new depo::Bus[n + 1];

for (int i = 0; i < n; ++i)

{

newPlanets[i] = arr[i];

}

delete[] arr;

arr = newPlanets;

arr[n].SetName(name);

arr[n].SetDiam(diam);

arr[n].SetLife(life);

arr[n].SetSate(satellite);

n += 1;

return arr;

}

else

{

std::cout << "База данных переполнена" << std::endl;

return arr;

}

}

// Сортировка с использованием перегрузки

void depo::sort\_overload(Bus\* arr, int n)

{

depo::Bus temp;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (arr[j] < arr[j + 1])

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

**Изображение выглядит как текст, снимок экрана, документ, меню

Автоматически созданное описаниеТестирование**

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание**