МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 14

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Доц., к.т.н.		К. А. Курицын
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ О КУРСОВОЙ РАБОТЕ
«ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ»

по дисциплине: «Технологии программирования»

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 1742		И.С.Панфилов
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2020

Содержание

1. Постановка задачи

Создать класс «Военная техника». В результате использования класса мы можем получить: боевой танк, бронетранспортер, боевой робот, гаубица, истребитель, рассматривается техника для двух стран – Россия и США. Различный набор характеристик (мощность, дальность, тип оружия, калибр, метод передвижения (воздух, земля, вода), год производства, количество, боевая масса, экипаж (число человек)) пользователь задает через входной файл (или набор входных файлов по типу создаваемой военной техники) или через консоль. Использовать паттерн «Абстрактная фабрика». Результат создания техники и все их характеристики записываются в выходной файл.

1.1. Функциональные требования

1.1.1. Создать класс «Военная техника»;
1.1.2. Реализовать пользовательское меню: Работа через файл, работа через консоль или выход.
1.1.3. Входные данные (информация о техники) загружаются из внешнего файла;
1.1.4. Программа должна быть на русском языке для удобства пользованием.

1.2. Спецификация

1.2.1. Реализация с использованием паттерна «Абстрактная фабрика»;
1.2.2. При запуске программы клиент видит: Меню, предлагающее выбрать режим работы
1.2.3. После выбора режима работы пользователь попадает в меню создание техники (Выбран пункт через консоль) или в меню выбора файла(Выбран пункт через файл)
1.2.4. Загрузка данных происходит из файла Файл «out.txt» содержит информацию о технике: мощность, дальность, тип оружия, калибр, метод передвижения (воздух, земля, вода), год производства, количество, боевая масса, экипаж (число человек)
1.2.5. Класс «Charectiristics» содержит поля, в которых находится информация о технике
1.2.6. Если клиент вводит число, которого нет в меню, программа просит ввести корректный номер.
1.2.7. Для создания вооружения моно выбрать две страны: США и Россия
1.2.8. После выбора страны пользователь попадает в меню выбора типа транспортного средства
1.2.9. После выбора типа передвижения пользователь попадает в меню заполнения данных ТС
1.2.10. Вывод данных происходит в консоль в режиме реального времени

2. Основная часть

2.1. Условия проведения

Проведение на тестовом компьютере.

2.2. Программа и методика испытания

№	Сценарий проверки	Ожидаемый результат	Результат	№ спецификации
1	Запуск программы	Попадание в начальное меню	Рисунок 1	1.2.2, 1.2.6
2	Ввод номера, отсутствующего в списке	Вывод сообщения об ошибке с просьбой выбрать корректный номер	Рисунок 2	1.2.6
3	Выбор пункта 1(Работа через консоль)	Вывод на экран меню выбора страны. Переход на меню выбора нации	Рисунок 3	1.2.3 1.2.7 1.2.2
4	Выбор Нации: США(1)	Переход на меню выбора типа техники	Рисунок 4	1.2.1,1.2.2, 1.2.7
5	Выбор типа техники:Наземная(1)	Вывод на экран данных, переход в меню заполнения характеристик	Рисунок 5 Рисунок 6	1.2.1, 1.2.5, 1.2.7
6	Заполнение Характеристик	Опишется тип техники	Рисунок 7	1.2.9
7	Вывод данных на экран	Данные о технике выведутся на экран	Рисунок 8	1.2.10
8	Выбор режима чтения данных из файла	Данные считаются и выведутся на экран	Рисунок 9 Рисунок 10	1.2.10,1.2.4

3. Листинг программы

Ссылка на GitHub https://github.com/Panfinator/Kursuch

main.cpp

```
#include "MilitaryVehicle.hpp"
#include "USFabrica.hpp"
#include "RFFabrica.hpp"

#include <fstream>

using namespace std;

int main() {
	system("color F0");
	setlocale(LC_ALL, "");
	cout << "\tВас приветствует программа выбора вооружения:\n\n";
	cout << "-----------------------------------------------"<<endl;
	cout << "\tВыберите Режим работы:\n";
	cout << "1--- Через консоль" << endl;
	cout << "2--- Через файл" << endl;
	cout << "3--- Выход" << endl;
	cout << "-----------------------------------------------" << endl;
	cout << "Ваш выбор: ";

	MilitaryVehicle* mv = nullptr;
	int a;
	cin >> a;

	while (a != 1 && a != 2 && a != 3)
	{
		cout << "\t ВНИМАНИЕ! Выбирите действие 1,2 или 3  " << endl;
		cout << "Ваш выбор: ";
		cin >> a;
	}


	if (a == 1)
	{
		system("cls");
		cout << "\tВыберите Страну:\n";
		cout << "1--- США" << endl;
		cout << "2--- Россия" << endl;
		cout << "-----------------------------------------------" << endl;
		cout << "Ваш выбор: ";
		cin >> a;
		system("cls");
		cout << "\tВыберите тип передвижения:\n";
		cout << "1--- Ездит(Наземная)" << endl;
		cout << "2--- Летает(Воздушная)" << endl;
		cout << "3--- Плавает(Водная)" << endl;
```

```
		cout << "-------------------------------------------------" << endl;
		cout << "Ваш выбор: ";
		int b;
		cin >> b;
		system("cls");


		cout << "\t\t Заполните характеристики техники" <<endl;
		cout << "1)Кол-во 2)расстояние 3)экипаж 4)мощность 5)тип_оружия 6)масса
7)год выпуска 8)тип техники:\n";
		if (a == 1) {

			mv = new MilitaryVehicle(new USFabrica(), cin, (b == 0 ?
MovementType::Fly : (b == 1 ? MovementType::Ride : MovementType::Swim)));
		}
		else {

			mv = new MilitaryVehicle(new RFFabrica(), cin, (b == 0 ?
MovementType::Fly : (b == 1 ? MovementType::Ride : MovementType::Swim)));
		}
	}
	if (a == 2) {
		//string fileName;
		//cout << "File name: ";
		//cin >> fileName;
		ifstream in("out.txt");
		in >> a;
		int b;
		in >> b;
		system("cls");
		cout << "\tИнформация о технике:\n";
		cout << "1)Кол-во 2)расстояние 3)экипаж 4)мощность 5)тип_оружия 6)масса
7)год выпуска 8)тип техники:\n";
		cout << "-------------------------------------------------" << endl;
		if (a == 1) {
			mv = new MilitaryVehicle(new USFabrica(), in, (b == 0 ?
MovementType::Fly : (b == 1 ? MovementType::Ride : MovementType::Swim)));
		}
		else {
			mv = new MilitaryVehicle(new RFFabrica(), in, (b == 0 ?
MovementType::Fly : (b == 1 ? MovementType::Ride : MovementType::Swim)));
		}
	}


	cout << '\n' << (*mv);
	ofstream out("out.txt");
	out << (*mv);
	system("pause>>void");

	if (a == 3)
		return 0;

}
```

FlyMovement.hpp

```
#pragma once
#include "Movement.hpp"
#include <string>


class FlyMovement : public Movement {
```

```
public:
	const char* Name() { return "Fly"; }
};
```

MilitaryVehicle.hpp

```
#pragma once
#include "MilitaryVehicleFactory.hpp"

class MilitaryVehicle {
private:
	Characteristic* characteristic;
	const char* Country;
	Movement* movement;
public:
	MilitaryVehicle(MilitaryVehicleFactory* militaryVehicleFactory, std::istream& in,
MovementType type) {
		characteristic = militaryVehicleFactory->CreateCharacteristic(in);
		Country = militaryVehicleFactory->Country();
		movement = militaryVehicleFactory->CreateMovement(type);
	}

	friend std::ostream& operator <<(std::ostream& out, const MilitaryVehicle&
militaryVehicle) {
		out << militaryVehicle.Country << '\n' << *(militaryVehicle.characteristic)
<< '\n' << (militaryVehicle.movement != nullptr ? militaryVehicle.movement->Name() : "");
		return out;
	}
	~MilitaryVehicle();
};
```

MilitaryVehicleFactory.hpp

```
#pragma once
#include "FlyMovement.hpp"
#include "SwimMovement.hpp"
#include "RideMovement.hpp"

#include "Characteristic.hpp"
#include "Movement.hpp"

enum MovementType {
	Fly = 0,
	Ride = 1,
	Swim = 2
};


class MilitaryVehicleFactory {
public:
	virtual const char* Country() = 0;
	virtual Movement* CreateMovement(MovementType type) = 0;
	virtual Characteristic* CreateCharacteristic(std::istream& in) = 0;
};
```

Movement.hpp

```
#pragma once
#include <string>
```

```
class Movement {
public:
	virtual const char* Name() = 0;
};
```

RFFabrica.hpp

```
#pragma once
#include "MilitaryVehicleFactory.hpp"

class RFFabrica : public MilitaryVehicleFactory {
public:
	virtual const char* Country() { return "Russia"; };

	virtual Movement* CreateMovement(MovementType type) {
		if (type == MovementType::Fly) {
			return new FlyMovement();
		}
		else if (type == MovementType::Ride) {
			return new RideMovement();
		}
		else if (type == MovementType::Swim) {
			return new SwimMovement();
		}
		else {
			return nullptr;
		}
	}
	virtual Characteristic* CreateCharacteristic(std::istream& in) {
		Characteristic* c = new Characteristic();
		in >> *c;
		return c;
	}
};
```

RideMovement.hpp

```
#pragma once
#include "Movement.hpp"
#include <string>


class RideMovement : public Movement {
public:
	const char* Name() { return "Ride"; }
};
```

SwimMovement.hpp

```cpp
#pragma once
#include "Movement.hpp"
#include <string>


class SwimMovement : public Movement {
public:
	const char* Name() { return "Swim"; }
};
```

USFabrica.hpp

```cpp
#pragma once
#include "MilitaryVehicleFactory.hpp"

class USFabrica : public MilitaryVehicleFactory {
public:
	virtual const char* Country() { return "USA"; };

	virtual Movement* CreateMovement(MovementType type) {
		if (type == MovementType::Fly) {
			return new FlyMovement();
		}
		else if (type == MovementType::Ride) {
			return new RideMovement();
		}
		else if (type == MovementType::Swim) {
			return new SwimMovement();
		}
		else {
			return nullptr;
		}
	}
	virtual Characteristic* CreateCharacteristic(std::istream& in) {
		Characteristic* c = new Characteristic();
		in >> *c;
		return c;
	}
};
```

Charecteristics.hpp

```cpp
#pragma once
#include <string>
#include <iostream>

class Characteristic {
	std::string power;
	double distance;
	std::string weaponType;
	std::string vehicleType;
	double weigt;
	int year;
	unsigned peopleCount;
	unsigned count;
public:
	std::string& Power() { return power; }
	double& Distance() { return distance; }
	std::string& WeaponType() { return weaponType; }
	std::string& VehicleType() { return vehicleType; }
	double& Weight() { return weight; }
```

```
        int& Year() { return year; }
        unsigned& PeopleCount() { return peopleCount; }
        unsigned& Count() { return count; }

        const std::string& Power() const { return power; }
        const double& Distance() const { return distance; }
        const std::string& WeaponType() const { return weaponType; }
        const std::string& VehicleType() const { return vehicleType; }
        const double& Weight() const { return weight; }
        const int& Year() const { return year;
        const unsigned& PeopleCount() const { return peopleCount; }
        const unsigned& Count() const { return count; }

        friend std::istream& operator >>(std::istream& in, Characteristic& characteristic)
{
                in >> characteristic.count >> characteristic.distance >>
characteristic.peopleCount
                        >> characteristic.power >> characteristic.weaponType >>
characteristic.weight
                        >> characteristic.year >> characteristic.vehicleType;
                return in;
        }

        friend std::ostream& operator <<(std::ostream& out, const Characteristic&
characteristic) {
                out << characteristic.count << ' ' << characteristic.distance << ' ' <<
characteristic.peopleCount << ' '
                        << characteristic.power << ' ' << characteristic.weaponType << ' ' <<
characteristic.weight
                        << ' ' << characteristic.year << ' ' << characteristic.vehicleType;
                return out;
        }
};
```

4. Список использованной литературы

1) Керниган Б., Ритчи Д. – Язык программирования Си. Учебное пособие. – 3-е изд., испр. — СПб.: «Невский Диалект», 2001г. – 188 с.
2) https://habr.com/ru/post/210288/ - Информация о шаблонах проектирования
3) https://refactoring.guru/ru/design-patterns/abstract-factory- Абстрактная фабрика
4) https://habr.com/ru/post/465835/ Абстрактная фабрика паттерн проектирования
5) Страуструп Бьерн-Программирование. Принципы и практика с использованием C++ -«Вильямс»,2016г-201с
6) https://ru.wikipedia.org/wiki/Абстрактная_фабрика_(шаблон_проектирования)- Абстрактная фабрика

Приложение 1

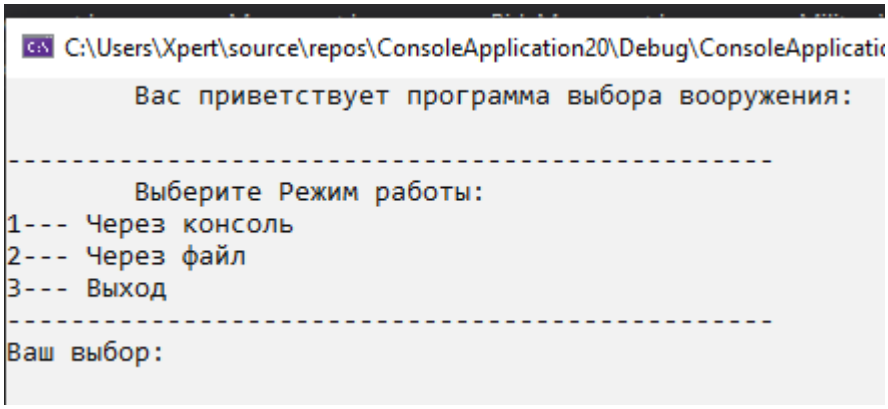

Рисунок 1 – Запуск программы

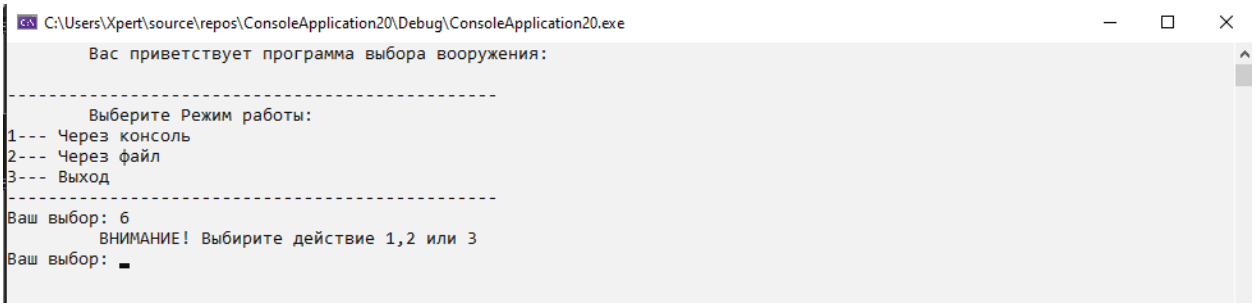

Рисунок 2 – Ввод некорректного значения

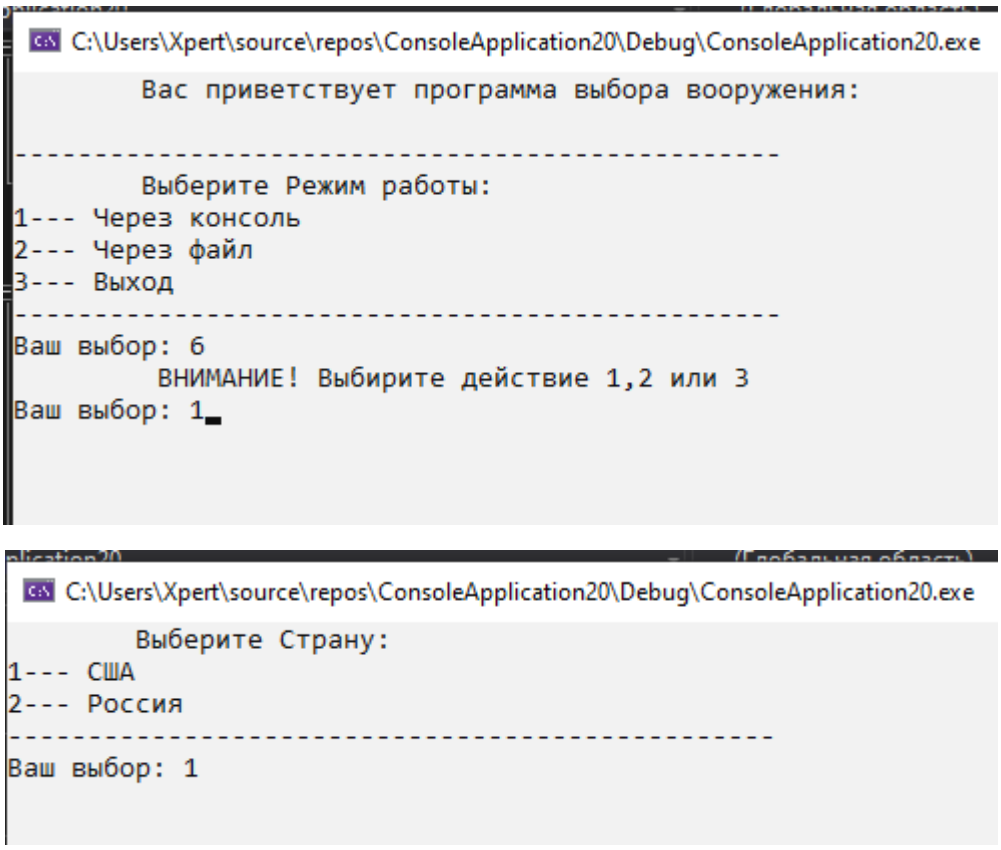

Рисунок 3 – Выбор режима работы через консоль

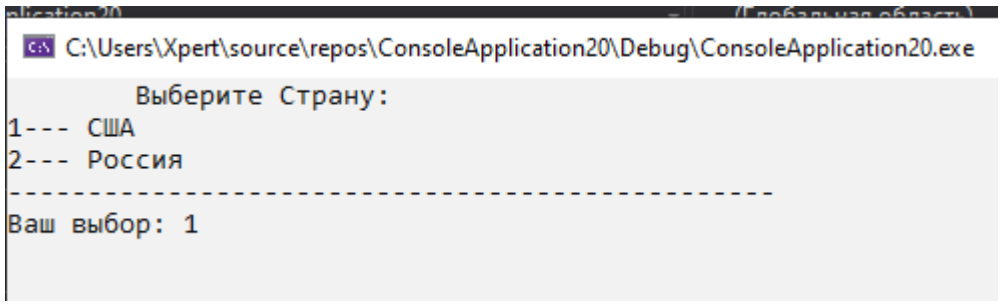

Рисунок 4 – Выбор страны и переход на след.меню

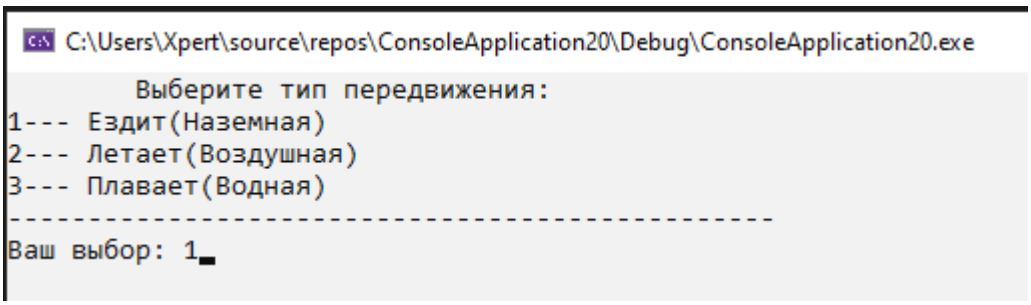

Рисунок 5 – Выбор наземной техники и переход в след.меню

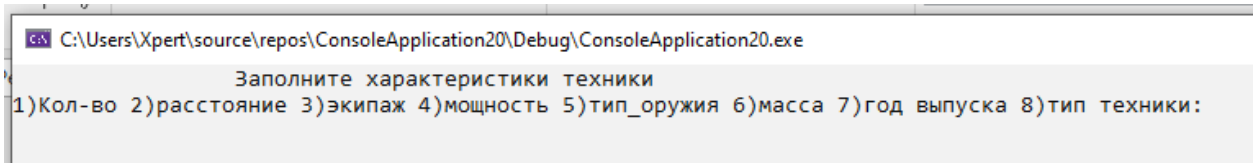

Рисунок 6-Переход в меню заполнения данных

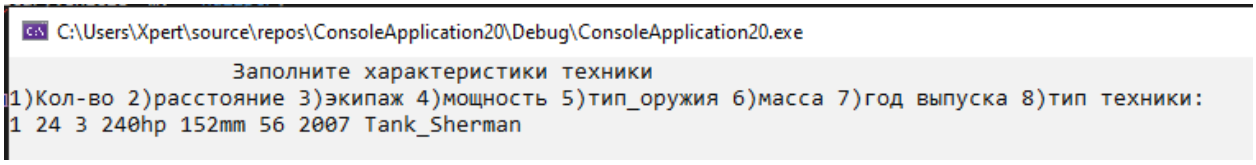

Рисунок 7-Заполнение данных

Рисунок 8-Вывод данных на экран

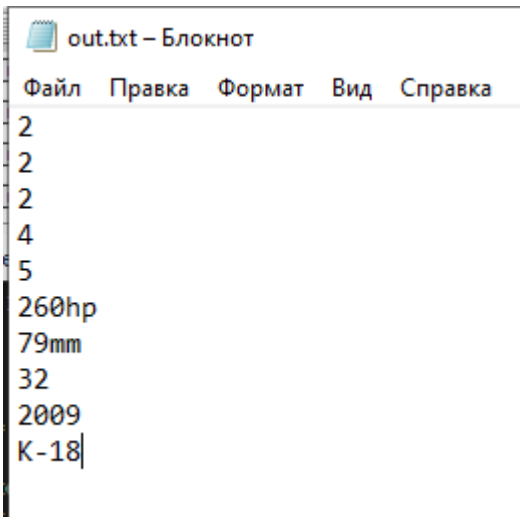

Рисунок 9-Содержание файла

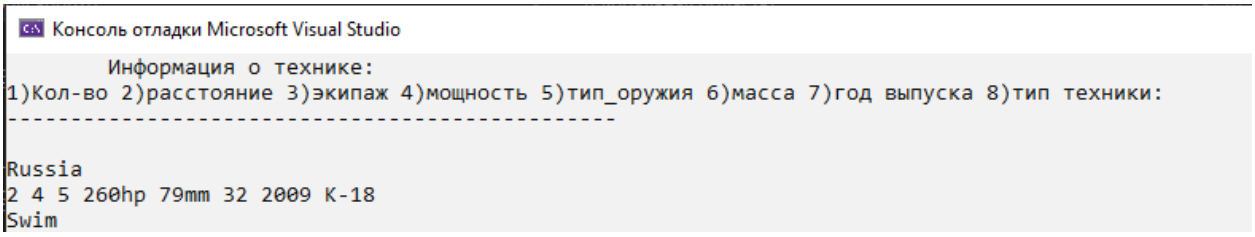

Рисунок 10-Вывод данных из файла