

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего  
образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №14

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

доц., д.т.н.  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

Курицын К.А.  
инициалы, фамилия

## ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

На тему: “Создание класса Телефон”

по дисциплине: «Технология программирования»

РАБОТУ ВЫПОЛНИЛ  
СТУДЕНТ ГР. 1542

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург  
2018

## Оглавление

1. Постановка задачи .....	3
1. Функциональные и нефункциональные требования .....	3
2. Ограничения и иные требования к оборудованию: .....	3
3. Взаимодействие с внутренними системами: .....	3
4. Требования к стороннему ПО:.....	4
5. Взаимодействие с внешними системами: .....	4
6. Технические спецификации: .....	4
2. Основная часть .....	5
2.1. Программа и методика испытаний .....	5
3. Листинг .....	7
4. Литература .....	7
Приложение №1 .....	7
Приложение №2 .....	9
Приложение №3 .....	9

## Постановка задачи

Определить класс «Телефон».

Базовый класс определяет интерфейс для построения телефонного оборудования; он может быть специализирован для конкретного устройства (смартфон, домашний телефон, спутниковый телефон). Каждый тип телефонов имеет свои характеристики в зависимости от типа: год производства, цвет, наличие GPS, ГЛОНАСС, Wi-Fi, 4G, камеры (и количество МР), размер экрана, объем памяти, производительность, наличие сенсора, тип связи.

### 1. Функциональные и нефункциональные требования

- Необходимо использовать паттерн “Одиночка”.
- ПО должно обеспечивать выполнение следующих функций над телефонами: просмотр телефона.
- Загрузка и сохранение информации о телефонах в файл.
  - Каждый телефон обладает следующими характеристиками:
  - Год производства
  - Цвет
  - Тип связи
  - Телефон так же может обладает следующими характеристиками:
  - GPS
  - ГЛОНАСС
  - Wi-Fi
  - 4G
  - Диагональ экрана
  - Частота процессора
  - Объем памяти
  - Камера
  - Наличие сенсора

### 2. Ограничения и иные требования к оборудованию:

- Процессор с тактовой частотой не ниже 1,8 ГГц. Рекомендуется использовать как минимум двухъядерный процессор.
- Не менее 1 ГБ ОЗУ; рекомендуется 3 ГБ ОЗУ
- Место на жестком диске: до 130 МБ свободного места в зависимости от установленных компонентов, обычно для установки требуется от 20 до 50 МБ свободного места
- Видеоадаптер с минимальным разрешением 720p (1280 на 720 пикселей); для оптимальной работы Visual Studio рекомендуется разрешение WXGA (1366 на 768 пикселей) или более высокое.
- Командная строка - для ввода команд с клавиатуры
- Минимальная версия операционной системы Windows 8.

### 3. Взаимодействие с внутренними системами: есть.

4. Требования к стороннему ПО:

Любое приложение, способное скомпилировать исходный код. Например, Visual Studio. Для получения наилучших результатов используйте средства диагностики с самым последним обновлением для вашей версии Visual Studio.

5. Взаимодействие с внешними системами: отсутствует, но есть загрузка из файла и сохранение в файл.

6. Технические спецификации:

1.6.1. При запуске программы на экране высвечивается меню, в котором можно выбрать один из четырех пунктов меню: Загрузки данных из файла, сохранение данных в файл, вывод данных на экран, выход.

1.6.2. Загрузка данных из файла. Выбор элемента меню.

1.6.2.1. Если с файлом возникли проблемы, то вывести ошибку.

1.6.2.2. При успешной загрузке файла вывести сообщение.

1.6.3. Вывод данных на экран. Выбор элемента меню.

1.6.4. Сохранение в файл. Выбор элемента меню.

1.6.4.1. Если с файлом возникли проблемы, то вывести ошибку.

1.6.4.2. При успешном сохранении файла вывести сообщение.

1.6.5. Выход. Выбор элемента меню.

## 2. Основная часть

### 2.1. Программа и методика испытаний.

#### 2.1.1. Условия проведения

Проведение на тестовом пользовательском компьютере.

#### 2.1.2. Методика

N <sub>тп</sub>	Сценарий проверки	Ожидаемый результат	Результат № рисунка (см. в Приложении №3)	№ФТ Технические спецификации
1	Запустить ПО	Вывод на экран меню, содержащего пункты: Загрузки данных из файла, сохранение данных в файл, вывод данных на экран, выход.	Получен Рисунок 1	1.6.1
2	Загрузка из файла	<ul style="list-style-type: none"><li>Если с файлом возникли проблемы, то вывести ошибку</li><li>При успешной загрузке файла выводит сообщение</li></ul>	Получен рисунок 3  Получен рисунок 4	1.6.2.1  1.6.2.2
3	Просмотр всех телефонов.	1. На экран выводится информация по телефонам.	Получен рисунок 2	1.6.3
7	Сохранение в файл	<ul style="list-style-type: none"><li>Если с файлом возникли проблемы то вывести ошибку</li><li>При успешном сохранении в файл выводит сообщение</li></ul>	Получен рисунок 5  Получен рисунок 2	1.6.4.1  1.6.4.2
8	Выход	Закрытие приложения.		1.6.5

### 3. Листинг

Приведен в приложении №2.

<https://github.com/AkioKoneko/coursework>

#### 1. Литература

1. [https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%B8%D0%BD%D0%BE%D1%87%D0%BA%D0%B0\\_\(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD\\_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F\)](https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%B8%D0%BD%D0%BE%D1%87%D0%BA%D0%B0_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))
2. <http://www.sbp-program.ru/c/sbp-file-c.htm>
3. <http://cppstudio.com/post/439/>
4. <http://cppstudio.com/post/10103/>
5. <https://ravesli.com/urok-85-dinamicheskoe-vydelenie-pamyati-operatory-new-i-delete/>

## **Формат файлов для загрузки и выгрузки информации о телефонах**

Имя и расширение файла для считывания – input.txt.

Имя и расширение файла для сохранения – output.txt.

Файл должен содержать обычный текст без форматирования и без кодирования, состоящий из записей (строк) в кодировки ASCII. Все поля разделены символом переноса '\n'.

Структура файла:

- Телефоны

Структура телефона:

- Год производства
- Цвет
- Тип связи
- GPS
- ГЛОНАСС
- Wi-Fi
- 4G
- Диагональ экрана
- Частота процессора
- Объем памяти
- Камера
- Наличие сенсора

Пример:



input.txt — Блокнот

Файл Правка Формат Вид Справка

LandlinePhone

Год производства: 1999

Цвет: зеленый

Тип связи: проводная

Smartphone

Год производства: 2003

Цвет: синий

Тип связи: сотовая

GPS: Есть

ГЛОНАСС: Нет

Wi-Fi: Есть

4G: Нет

Экран: Есть

Диагональ экрана: 1016

Частота процессора: 1200

Объем памяти: 2048

Камера: Нет

SatellitePhone

Год производства: 1998

Цвет: красный

Тип связи: спутниковая

GPS: Есть

ГЛОНАСС: Нет

Wi-Fi: Есть

4G: Нет

Экран: Нет

Частота процессора: 1000

Объем памяти: 1024



## Листинг

### main.cpp

```
#include <list>
#include <memory>
#include <iostream>
#include "phone.hpp"

int main()
{
    char c;
    std::list<std::unique_ptr<Phone>> phones;

    do
    {
        std::cout << '\n';

        std::cout << "L: Загрузить данные из файла\n"
                    << "P: Вывести данные на экран\n"
                    << "S: Сохранить данные в файл\n"
                    << "Q: Выход\n\n";

        std::cin >> c;

        if (c == 'L')
        {
            for (auto &&p : PhoneLoader("input.txt"))
                phones.emplace_back(p);
        }
    }
```

```

        else if (c == 'P')
        {
            for (auto &p : phones)
            {
                p->Print();
                std::cout << '\n';
            }
        }

        else if (c == 'S')
        {
            for (auto &p : phones)
                p->Save("output.txt");
        }
    }
    while (c != 'Q');
}

```

### **phone.hpp**

```

#ifndef PHONE_HPP
#define PHONE_HPP

```

```

#include <map>
#include <string>
#include <fstream>
#include <iostream>
#include <functional>
#include <cctype>

```

```

using mm = short;
using MHz = short;
using MB = short;

```

```
constexpr const char * btos(bool arg) { return arg ? "Есть" : "Нет"; }
```

```
class Phone
```

```
{
```

```
    short prod_year;
```

```
    std::string color;
```

```
    std::string comm_type;
```

```
public:
```

```
    Phone(std::ifstream &in);
```

```
        inline auto ProductionYear() const { return prod_year; }
```

```
    const inline auto & CommType()    const { return comm_type; }
```

```
    const inline auto & Color()        const { return color;   }
```

```
    virtual const char * ClassName() const { return ""; }
```

```
    virtual void Print(std::ostream &out = std::cout) const;
```

```
    void Save(const std::string filename) const;
```

```
};
```

```
class LandlinePhone : public Phone
```

```
{
```

```
    virtual const char * ClassName() const { return "LandlinePhone"; }
```

```
public:
```

```
    LandlinePhone(std::ifstream &in) : Phone(in) {}
```

```
};
```

```
class MobilePhone : public Phone
```

```
{
```

```
    bool has_gps, has_glonass, has_wifi, has_4g;
```

```
    mm screen_diagonal;
```

```

    MHz cpu_frequency;

    MB memory_size;

public:
    MobilePhone(std::ifstream &in);

    inline bool HasGPS()      const { return has_gps;      }
    inline bool HasGLONASS()  const { return has_glonass;    }
    inline bool HasWiFi()     const { return has_wifi;      }
    inline bool Has4G()       const { return has_4g;        }
    inline bool HasScreen()   const { return screen_diagonal != 0; }
    inline mm  ScreenDiagonal() const { return screen_diagonal; }
    inline MHz  CPUFrequency() const { return cpu_frequency; }
    inline MB   MemorySize()  const { return memory_size;   }

    virtual void Print(std::ostream &out = std::cout) const;

};

class Smartphone : public MobilePhone
{
    float camera_mp;

public:
    Smartphone(std::ifstream &in);

    inline bool HasCamera() const { return camera_mp != 0; }
    inline auto CameraMP()  const { return camera_mp;      }

    virtual const char * ClassName() { return "Smartphone"; }
    virtual void Print(std::ostream &out = std::cout) const;

};

class SatellitePhone : public MobilePhone

```

```

{
    virtual const char * ClassName() const { return "SatellitePhone"; }

public:
    SatellitePhone(std::ifstream &in) : MobilePhone(in) {}
};

template <typename T>
inline Phone * PhoneFactory(std::ifstream &in) { return new T(in); }

static std::map<std::string, std::function<Phone*(std::ifstream &)>> NewPhone =
{
    { "Smartphone" , PhoneFactory<Smartphone> },
    { "LandlinePhone" , PhoneFactory<LandlinePhone> },
    { "SatellitePhone", PhoneFactory<SatellitePhone> }
};

bool nws_left(std::istream &in);

inline void skip_colon(std::ifstream &in)
{
    do in.get(); while (in.peek() != ':');
}

class PhoneIterator
{
    std::ifstream &file;
    std::string type;
    bool is_end;

public:
    inline PhoneIterator(std::ifstream &f, bool e = false)
        : file(f), is_end(e) {}

```

```

using value_type = Phone *;
using difference_type = int;
using pointer = Phone **;
using reference = Phone *&;
using iterator_category = std::input_iterator_tag;

inline bool operator!=(const PhoneIterator &other) const
{
    return (is_end ^ other.is_end);
}

inline bool operator==(const PhoneIterator &other) const
{
    return !(*this != other);
}

inline PhoneIterator operator++()
{
    is_end = !nws_left(file);
    if (!is_end)
        file >> type;
    return *this;
}

inline PhoneIterator operator++(int)
{
    PhoneIterator old = *this;
    ++(*this);
    return old;
}

inline value_type operator*() { return NewPhone[type](file); }

```

```

};

class PhoneLoader
{
    std::ifstream file;

public:
    inline PhoneLoader(const std::string filename) : file(filename) { }
    inline auto begin() { return PhoneIterator(file); };
    inline auto end() { return PhoneIterator(file, true); };
};

#endif

```

### **phone.cpp**

```

#include "phone.hpp"

Phone::Phone(std::ifstream &in)
{
    skip_colon(in);
    in >> prod_year;
    skip_colon(in);
    std::getline(in, color);
    skip_colon(in);
    std::getline(in, comm_type);
}

MobilePhone::MobilePhone(std::ifstream &in) : Phone(in)
{
    std::string buff;
    skip_colon(in);
    in >> buff;

```

```
has_gps = (buff == "Есть");
```

```
skip_colon(in);
```

```
in >> buff;
```

```
has_glonass = (buff == "Есть");
```

```
skip_colon(in);
```

```
in >> buff;
```

```
has_wifi = (buff == "Есть");
```

```
skip_colon(in);
```

```
in >> buff;
```

```
has_4g = (buff == "Есть");
```

```
skip_colon(in);
```

```
in >> buff;
```

```
if (buff == "Есть")
```

```
{
```

```
    skip_colon(in);
```

```
    in >> screen_diagonal;
```

```
}
```

```
skip_colon(in);
```

```
in >> cpu_frequency;
```

```
skip_colon(in);
```

```
in >> memory_size;
```

```
}
```

```
Smartphone::Smartphone(std::ifstream &in) : MobilePhone(in)
```

```
{
```

```
    std::string has_camera;
```

```
    skip_colon(in);
```



```

in >> has_camera;

if (has_camera == "Есть")
{
    skip_colon(in);
    in >> camera_mp;
}
}

void Phone::Print(std::ostream &out) const
{
    out << "Год производства: " << prod_year << "\n";
    out << "Цвет: " << color << "\n";
    out << "Тип связи: " << comm_type << "\n";
}

void MobilePhone::Print(std::ostream &out) const
{
    Phone::Print(out);

    out << "GPS: " << btos(has_gps) << "\n";
    out << "ГЛОНАСС: " << btos(has_glonass) << "\n";
    out << "Wi-Fi: " << btos(has_wifi) << "\n";
    out << "4G: " << btos(has_4g) << "\n";
    out << "Экран: " << btos(HasScreen()) << "\n";
    if (HasScreen())
        out << "Диагональ экрана: " << screen_diagonal << " mm\n";
    out << "Частота процессора: " << cpu_frequency << " MHz\n";
    out << "Объем памяти: " << memory_size << " MB\n";
}

void Smartphone::Print(std::ostream &out) const
{

```

```

MobilePhone::Print(out);

out << "Камера: " << btos(HasCamera()) << '\n';
if (HasCamera())
    out << "Разрешение матрицы: " << camera_mp << " MP\n";
}

void Phone::Save(const std::string filename) const
{
    std::ofstream file(filename, std::ios_base::app);
    file << ClassName() << '\n';
    Print(file);
    file << '\n';
}

bool nws_left(std::istream &in)
{
    for (char c; (c = in.peek()) != '\n' && c != EOF; in.get())
        if (!std::isspace(c))
            return true;
    return false;
}

```

## Скриншоты, результатов работы программы

```
Текущая кодовая страница: 1251

L: Загрузить данные из файла
P: Вывести данные на экран
S: Сохранить данные в файл
Q: Выход
```

Рисунок 1 – Главное меню

```
LandlinePhone
Год производства : 1999
Цвет : зеленый
Тип связи : проводная

Smartphone
Год производства : 2003
Цвет : синий
Тип связи : сотовая
GPS : Есть
ГЛОНАСС : Нет
Wi - Fi : Есть
4G : Нет
Экран : Есть
Диагональ экрана : 1016
Частота процессора : 1200
Объем памяти : 2048
Камера : Нет
```

Рисунок 2 – Просмотр телефонов

```
Error file
```

Рисунок 3 – Отсутствие загружаемого файла

```
Данные успешно загружены
```

Рисунок 4 – Успешная загрузка из файла

```
Error file
```

Рисунок 5 – Отсутствие сохраняемого файла

```
Данные сохранены в файл
```

Рисунок 6 – Успешное сохранение