

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ “ХПІ”

Кафедра “Обчислювальна техніка та програмування”

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка
1КІТ.102.8А. 18046-01 81 01-1 –АЗ

Розробник

Виконав:

студент групи 1.КІТ-102.8А

_____/ Соколенко Д.Г./

Перевірив:

_____/Старший викладач. Молчанов Г.І./

Харків 2019

ЗАТВЕРДЖЕНО

1.KIT102.8A.18046-01 81 01-1 –АЗ

Розрахункове завдання з дисципліни
«Алгоритми та структури даних»

Пояснювальна записка
1KIT.102.8A.18046-01 81 01-1 -АЗ

Листів 20

Харків 2019

РОЗРАХУНКОВОГО ЗАВДАННЯ З ДИСЦИПЛІНИ «ПРОГРАМУВАННЯ»

Тема роботи. Розробка інформаційно-довідкової системи.

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

1 ВИМОГИ

1.1 Розробник

- Соколенко Дмитро Григорович;

- Студент групи КІТ 102.8(а);

- 07-06-2019р..

1.2 Загальне завдання

Завдання до роботи:

Кожний студент отримує індивідуальне завдання. Варіант завдання обирається за номером прізвища студента у журналі групи. При виконанні завдання з розробки інформаційно-довідкової системи необхідно виконати наступне:

- 1) з табл. 1, відповідно до варіанта завдання, обрати прикладну галузь;
- 2) дослідити літературу стосовно прикладної галузі. За результатами аналізу літератури оформити перший, аналітичний розділ пояснювальної записки обсягом 2–3 сторінки;
- 3) для прикладної галузі розробити розгалужену ієрархію класів, яка складається з не менш ніж трьох класів, один з яких є «батьком» для інших (класів-спадкоємців). Класи повинні мати перевантажені оператори введення-виведення даних та порівняння;
- 4) розробити клас-контролер, що буде включати колекцію розроблених класів, та наступні методи роботи з цією колекцією:
 - а) читання даних з файлу та їх запис у контейнер;
 - б) запис даних з контейнера у файл;
 - в) сортування елементів у контейнері за вказаними критеріями: поле та напрям сортування, які задаються користувачем з клавіатури;
 - г) пошук елементів за вказаним критерієм (див. «Завдання для обходу колекції» в табл. 1);

- 5) розробити клас, який має відображати діалогове меню для демонстрації реалізованих функцій класу контролера;
- 6) оформити схеми алгоритмів функцій класів контролера та діалогового меню;
- 7) оформити документацію: пояснювальну записку (див. розділ 2 даних методичних вказівок).

Увага. Текст програми та результати роботи програми мають бути подані в додатках.

Вимоги:

- усі класи повинні мати конструктори та деструктори;
- якщо функція не змінює поля класу, вона має бути декларована як константна;
- рядки повинні бути типу string;
- при перевантаженні функції треба використовувати ключове слово `override`;
- програмний код усіх класів має бути 100 % doxygenдокументований;
- у звіті текст програми слід оформляти стилем Courier new 8 пт, інтервал – одиничний; довжина рядка не повинна перевищувати 80 символів.

Додаткові вимоги на оцінку «добре»:

- виконання основного завдання та додаткових наступних вимог:
- додати обробку помилок; при цьому функція, що генерує виключення, при її декларуванні повинна мати ключове слово `throw`;
- виконати перевірку вхідних даних за допомогою регулярних виразів.

Додаткові вимоги на оцінку «відмінно»:

- виконати завдання відповідно до вимог на оцінку «добре» та додаткові наступні вимоги:
- критерій для пошуку та сортування задавати у вигляді функтора;
- розробити клас-тестер, основною метою якого буде перевірка коректності роботи класу-контролера.

Індивідуальне завдання:

Варіант: 17

Прикладна галузь: Мобільні пристрої

Базовий клас: Телефон (наприклад Nokia 1100)

Завдання для обходу колекції: Визначити телефон з найменшою щільністю пікселів.

2 ОПИС ПРОГРАМИ

2.1 Функціональне призначення

Програма призначена для виконання комплексних задач з курсу програмування

2.2 Опис логічної структури

Нижче продемонстрована діаграми класів (див. рис. 2.2.1)

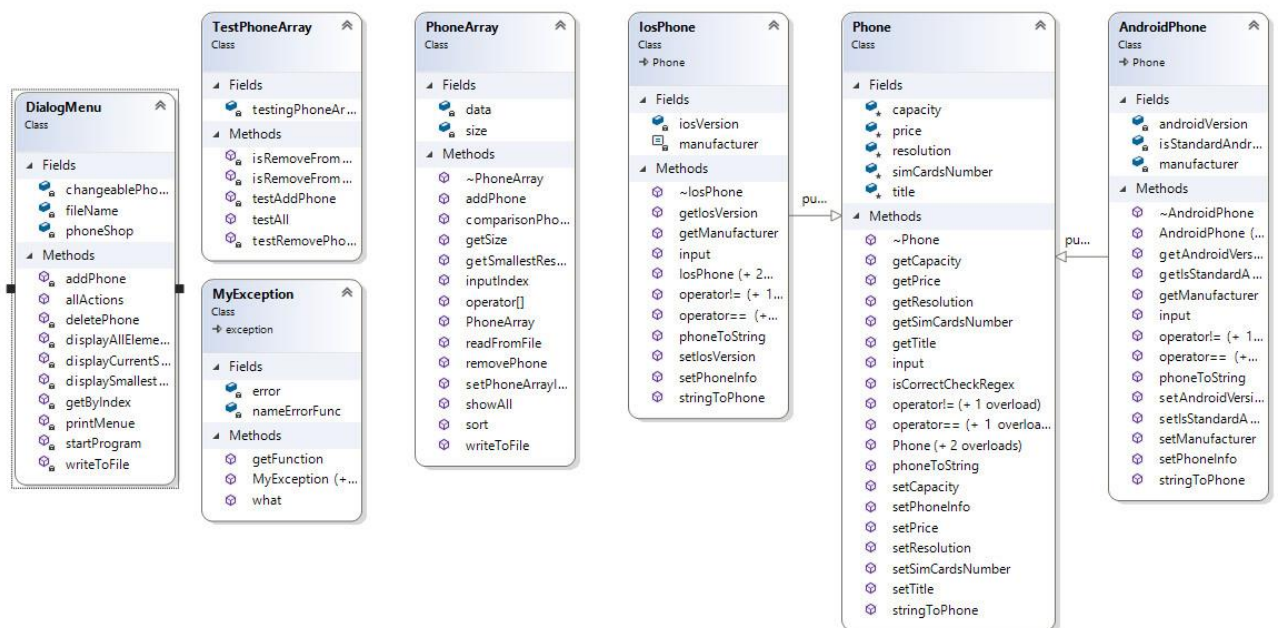
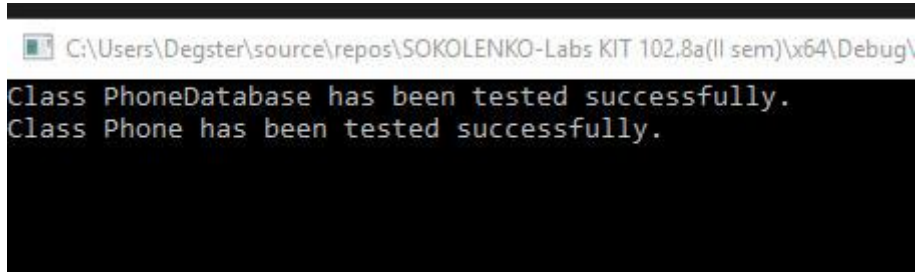


Рис.2.2.1 - Діаграма класів

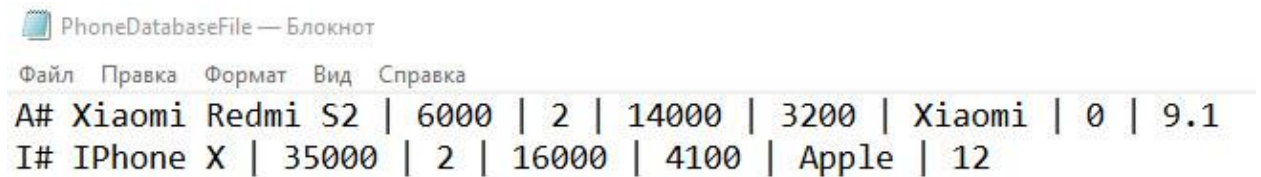
Усі пояснення див. у документації.

3 ВАРІАНТИ ВИКОРИСТАННЯ



```
C:\Users\Degster\source\repos\SOKOLENKO-Labs KIT 102.8a(II sem)\x64\Debug\
Class PhoneDatabase has been tested successfully.
Class Phone has been tested successfully.
```

Рис.3.1 - Запуск програми, зчитання даних з файлу

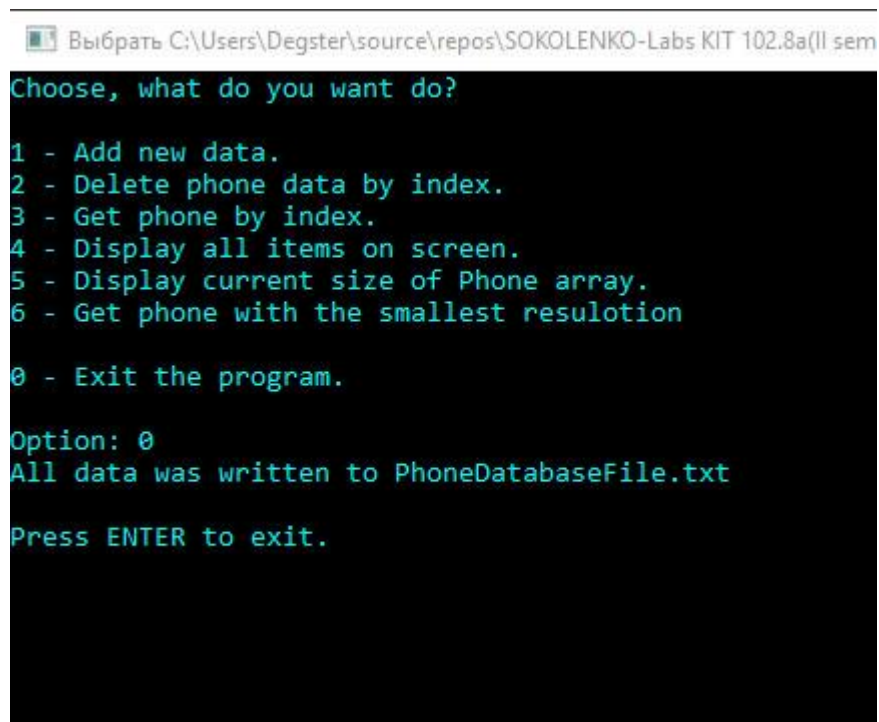


PhoneDatabaseFile — Блокнот

Файл Правка Формат Вид Справка

A#	Xiaomi	Redmi S2		6000		2		14000		3200		Xiaomi		0		9.1
I#	iPhone	X		35000		2		16000		4100		Apple		12		

Рис.3.2 - Вихідні дані у файлі



```
Выбрать C:\Users\Degster\source\repos\SOKOLENKO-Labs KIT 102.8a(II sem
Choose, what do you want do?

1 - Add new data.
2 - Delete phone data by index.
3 - Get phone by index.
4 - Display all items on screen.
5 - Display current size of Phone array.
6 - Get phone with the smallest resolution

0 - Exit the program.

Option: 0
All data was written to PhoneDatabaseFile.txt

Press ENTER to exit.
```

Рис.3.3 - Додавання нового елементу

```
C:\Users\Degster\source\repos\SOKOLENKO-Labs KIT 102.8a(II sem)\x64\Debug\sokolenko0
Choose, what do you want do?
1 - Add new data.
2 - Delete phone data by index.
3 - Get phone by index.
4 - Display all items on screen.
5 - Display current size of Phone array.
6 - Get phone with the smallest resolution
7 - Display all phones with two or more words in the title
0 - Exit the program.

Option: 7

Phones with two or more words in the title:

Lenovo A2010 | 32 | 2 | 4564 | 9400
Xiaomi RedmiS2 | 342 | 1 | 423 | 2900

Press ENTER to continue.
```

Рис.3.4 - Видалення елементу

```
C:\Users\Degster\source\repos\SOKOLENKO-Labs KIT 102.8a(II sem)\x64\Det
Choose, what do you want do?
1 - Add new data.
2 - Delete phone data by index.
3 - Get phone by index.
4 - Display all items on screen.
5 - Display current size of Phone array.
6 - Get phone with the smallest resolution
0 - Exit the program.

Option: 6

Phone with the smallest resolution:

Nokia 3210 | 0 | 1 | 400 | 1000

Press ENTER to continue.
```

Рис.3.5 - Отримання по індексу

```
C:\Users\Degster\source\repos\SOKOLENKO-Labs KIT 102.8a(II sem)\x64\Det
Choose, what do you want do?
1 - Add new data.
2 - Delete phone data by index.
3 - Get phone by index.
4 - Display all items on screen.
5 - Display current size of Phone array.
6 - Get phone with the smallest resolution
0 - Exit the program.

Option: 5
Current size of Phone array: 5

Press ENTER to continue.
```

Рис.3.6 - Вивід на екран усіх елементів

```
C:\Users\Degster\source\repos\SOKOLENKO-Labs KIT 102.8a(II sem)\x64\Debug\sokolenko0
Choose, what do you want do?
1 - Add new data.
2 - Delete phone data by index.
3 - Get phone by index.
4 - Display all items on screen.
5 - Display current size of Phone array.
6 - Get phone with the smallest resolution
7 - Display all phones with two or more words in the title
0 - Exit the program.

Option: 4

Phone with index: 0
Lenovo A2010 | 32 | 2 | 4564 | 9400

Phone with index: 1
Xiaomi RedmiS2 | 342 | 1 | 423 | 2900

Phone with index: 2
TeslaPhone | 5 | 131 | 42 | 93

Press ENTER to continue.
```

Рис.3.7 - Відображення поточного розміру масиву


```
C:\Users\Degster\source\repos\SOKOLENKO-Labs KIT 102.8a(II sem)\x64\Debug\sokolenko03.exe
Choose, what do you want do?

1 - Add new data.
2 - Delete phone data by index.
3 - Get phone by index.
4 - Display all items on screen.
5 - Display current size of Phone array.
6 - Get phone with the smallest resolution

0 - Exit the program.

Option: 3

Enter index of element that you want to remove.You can choose from 0 to 4.
(If you choose larger value, the last index will be selected).

Index: 4

Phone with index: 4

TeslaPhone S | 0 | 3 | 3200000 | 15000

Press ENTER to continue.
```

Рис.3.8 - Отримання об'єкту Телефон з найменшою кількістю пікселів.

```
C:\Users\Degster\source\repos\SOKOLENKO-Labs KIT 102.8a(II sem)\x64\Debug\sokolenko03.exe
Choose, what do you want do?

1 - Add new data.
2 - Delete phone data by index.
3 - Get phone by index.
4 - Display all items on screen.
5 - Display current size of Phone array.
6 - Get phone with the smallest resolution

0 - Exit the program.

Option: 2

Enter index of element that you want to remove.You can choose from 0 to 5.
(If you choose larger value, the last index will be selected).

Index: 5
Done! Element with index 5 was removed!

Press ENTER to continue.
```

Рис.3.9 - Виведення на екран

```
A# Xiaomi Redmi S2 | 6000 | 2 | 14000 | 3200 | Xiaomi | 0 | 9.1
I# iPhone X | 35000 | 2 | 16000 | 4100 | Apple | 12
```

Рис.3.10 - Остаточні дані у файлі

ВИСНОВОК

В ході виконання поставленої задачі були закріплені отримані знання з дисципліни «Програмування» шляхом виконання типового комплексного завдання.

Приклад тексту програми

Phone.h

```
/**
 * @file Phone.h
 * Declaration Phone class.
 * @author Sokolenko Dmitro
 * @version 0.3
 * @date 2019.06.06
 */

#pragma once

#include <string>
#include <sstream>
#include <iostream>
#include <regex>

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::stringstream;
using std::getline;
using std::ostream;
using std::istream;
using std::regex;
using std::regex_match;
using std::regex_search;

/**
 * Declaration Phone class with fields and methods.
 * Used Javadoc style comments to handle Doxygen.
 */
class Phone {
protected:
    /** Phone title. */
    string title;
    /** Phone price. */
    unsigned int price;
    /** Phone sim-cards number. */
    unsigned int simCardsNumber;
    /** Phone screen resolution. */
    unsigned int resolution;
    /** Phone battery capacity. */
    unsigned int capacity;
public:
    /**
     * Default constructor
     * Used initialization lists.
     */
    Phone();

    /**
     * Constructor with parameters.
     * Used initialization lists.
     * @param newTitle initializes Phone::title.
     * @param newPrice initializes Phone::price.
     * @param newSimNum initializes Phone::simCardsNumber.
     * @param newResolution initializes Phone::resolution.
     * @param newCapacity initializes Phone::capacity.
     */
    Phone(string newTitle,
          unsigned int newPrice,
          unsigned int newSimNum,
          unsigned int newResolution,
          unsigned int newCapacity);

    /**
     * Copie-constructor.
     * Used initialization lists.
     * @param copiedPhone: its fields initialize fields current object.
     */
    Phone(const Phone& copiedPhone);

    /**
     * Destructor.
     */
};
```

```

*/
virtual ~Phone();

/**
 * Overloaded comparison operator.
 * @param otherPhone: its fields compare with fields current object.
 * @return result of comparison.
 */
bool operator==(const Phone& otherPhone) const;

/**
 * Overloaded comparison operator.
 * @param otherPhone: its fields compare with fields current object.
 * @return result of comparison.
 */
bool operator!=(const Phone& otherPhone) const;

/**
 * Overloaded comparison operator.
 * @param otherPhone: its fields compare with fields current object.
 * @return result of comparison.
 */
bool operator==(const Phone* otherPhone) const;

/**
 * Overloaded comparison operator.
 * @param otherPhone: its fields compare with fields current object.
 * @return result of comparison.
 */
bool operator!=(const Phone* otherPhone) const;

/**
 * Overloaded input operator.
 * @param in - reference to input stream.
 * @param inputPhone - reference to Phone object.
 * @return reference to input stream.
 */
friend istream& operator>>(istream& in, Phone& inputPhone);

/**
 * Overloaded input operator.
 * @param in - reference to input stream.
 * @param inputPhone - reference to Phone object.
 * @return reference to input stream.
 */
friend istream& operator>>(istream& in, Phone* inputPhone);

/**
 * Overloaded output operator.
 * @param out - reference to output stream.
 * @param outputPhone - reference to Phone object.
 * @return reference to output stream.
 */
friend ostream& operator<<(ostream& out, const Phone& outputPhone);

/**
 * Overloaded output operator.
 * @param out - reference to output stream.
 * @param outputPhone - reference to Phone object.
 * @return reference to output stream.
 */
friend ostream& operator<<(ostream& out, const Phone* outputPhone);

/**
 * Assignment to object new field value.
 * @param newTitle initializes Phone::title.
 * @param newPrice initializes Phone::price.
 * @param newSimNum initializes Phone::simCardsNumber.
 * @param newResolution initializes Phone::resolution.
 * @param newCapacity initializes Phone::capacity.
 */
void setPhoneInfo(string newTitle,
    unsigned int newPrice,
    unsigned int newSimNum,
    unsigned int newResolution,
    unsigned int newCapacity);

/**
 * A method that checks the string for correctness using regular expressions.
 * @param checkString - the checked string.

```

```

* @return the status of the correct line.
*/
bool isCorrectCheckRegex(string& checkString) const;

/**
* Reading from the console and filling in the fields of the Phone object.
*/
virtual void input() = 0;

/**
* Converting information about the current object to a string.
* @return a line with information about the current object.
*/
virtual string phoneToString() const = 0;

/**
* Fill the fields of the current object with
* information contained in the line.
* @param phoneString - the source line with information about the object.
*/
virtual void stringToPhone(const string phoneString) = 0;

/**
* Set the value of the variable Phone::title.
* @param newTitle is assigned the Phone::title field.
*/
void setTitle(string newTitle);

/**
* Get copy of field Phone::title.
* @return current value Phone::title.
*/
string getTitle() const;

/**
* Set the value of the variable Phone::price.
* @param newPrice is assigned the Phone::title field.
*/
void setPrice(unsigned int newPrice);

/**
* Get copy of field Phone::price.
* @return current value Phone::price.
*/
unsigned int getPrice() const;

/**
* Set the value of the variable Phone::simCardsNumber.
* @param newSimCardsNumber is assigned the Phone::simCardsNumber field.
*/
void setSimCardsNumber(unsigned int newSimCardsNumber);

/**
* Get copy of field Phone::simCardsNumber.
* @return current value Phone::simCardsNumber.
*/
unsigned int getSimCardsNumber() const;

/**
* Set the value of the variable Phone::resolution.
* @param newCapacity is assigned the Phone::resolution field.
*/
void setResolution(unsigned int newCapacity);

/**
* Get copy of field Phone::resolution.
* @return current value Phone::resolution.
*/
unsigned int getResolution() const;

/**
* Set the value of the variable Phone::capacity.
* @param newCapacity is assigned the Phone::capacity field.
*/
void setCapacity(unsigned int newCapacity);

```

```

    /**
    * Get copy of field Phone::capacity.
    * @return current value Phone::capacity.
    */
    unsigned int getCapacity() const;
};

```

Phone.cpp

```

#include "Phone.h"

Phone::Phone() :
    title(""),
    price(0),
    simCardsNumber(0),
    resolution(0),
    capacity(0) {}

Phone::Phone(string newTitle,
    unsigned int newPrice,
    unsigned int newSimNum,
    unsigned int newResolution,
    unsigned int newCapacity) :
    title(newTitle),
    price(newPrice),
    simCardsNumber(newSimNum),
    resolution(newResolution),
    capacity(newCapacity) {}

Phone::Phone(const Phone & copiedPhone) :
    title(copiedPhone.title),
    price(copiedPhone.price),
    simCardsNumber(copiedPhone.simCardsNumber),
    resolution(copiedPhone.resolution),
    capacity(copiedPhone.capacity) {}

Phone::~~Phone() {}

bool Phone::operator==(const Phone& otherPhone) const
{
    bool isEqualTitle = this->title == otherPhone.title;
    bool isEqualPrice = this->price == otherPhone.price;
    bool isEqualSimNumb = this->simCardsNumber == otherPhone.simCardsNumber;
    bool isEqualResol = this->resolution == otherPhone.resolution;
    bool isEqualCapac = this->capacity == otherPhone.capacity;

    if (isEqualTitle &&
        isEqualPrice &&
        isEqualSimNumb &&
        isEqualResol &&
        isEqualCapac)
        return true;
    else
        return false;
}

bool Phone::operator!=(const Phone& otherPhone) const
{
    return !(*this == otherPhone);
}

bool Phone::operator==(const Phone* otherPhone) const
{
    bool isEqualTitle = this->title == otherPhone->title;
    bool isEqualPrice = this->price == otherPhone->price;

```

```

    bool isEqualSimNumb = this->simCardsNumber == otherPhone->simCardsNumber;
    bool isEqualResol = this->resolution == otherPhone->resolution;
    bool isEqualCapac = this->capacity == otherPhone->capacity;

    if (isEqualTitle &&
        isEqualPrice &&
        isEqualSimNumb &&
        isEqualResol &&
        isEqualCapac)
        return true;
    else
        return false;
}

bool Phone::operator!=(const Phone* otherPhone) const
{
    return !(this == otherPhone);
}

istream& operator>>(istream& in, Phone& inputPhone)
{
    getline(in, inputPhone.title, '|');
    inputPhone.title.erase(inputPhone.title.end() - 1);
    in.ignore(1);

    in >> inputPhone.price;
    in.ignore(3);

    in >> inputPhone.simCardsNumber;
    in.ignore(3);

    in >> inputPhone.resolution;
    in.ignore(3);

    in >> inputPhone.capacity;

    return in;
}

istream& operator>>(istream& in, Phone* inputPhone)
{
    getline(in, inputPhone->title, '|');
    inputPhone->title.erase(inputPhone->title.end() - 1);
    in.ignore(1);

    in >> inputPhone->price;
    in.ignore(3);

    in >> inputPhone->simCardsNumber;
    in.ignore(3);

    in >> inputPhone->resolution;
    in.ignore(3);

    in >> inputPhone->capacity;

    return in;
}

ostream& operator<<(ostream& out, const Phone& outputPhone)
{
    out << outputPhone.title << " | " << outputPhone.price << " | " <<
outputPhone.simCardsNumber <<
    " | " << outputPhone.resolution << " | " << outputPhone.capacity;
}

```

```

        return out;
    }

ostream& operator<<(ostream& out, const Phone* outputPhone)
{
    out << outputPhone->title << " | " << outputPhone->price << " | " << outputPhone->simCardsNumber <<
        " | " << outputPhone->resolution << " | " << outputPhone->capacity;

    return out;
}

void Phone::setPhoneInfo(string newTitle,
    unsigned int newPrice,
    unsigned int newSimNum,
    unsigned int newResolution,
    unsigned int newCapacity)
{
    title = newTitle;
    price = newPrice;
    simCardsNumber = newSimNum;
    resolution = newResolution;
    capacity = newCapacity;
}

bool Phone::isCorrectCheckRegex(string& checkString) const
{
    regex incorrectSymbols("[^\\w\\s.,;:~!-\\\"()*]*");
    regex repeatSymbols("(?<=([\\s.,;:~!-\\\"])\")1+");
    regex lowerCase("[a-z]");

    if (!regex_search(checkString, incorrectSymbols) || !regex_search(checkString,
        repeatSymbols) || !regex_search(checkString, lowerCase))
        return true;
    else
        return false;
}

void Phone::input()
{
    cout << "Enter information about phone." << endl;

    rewind(stdin);
    cout << "Phone title: ";
    getline(cin, title);
    if (!isCorrectCheckRegex(title))
        cout << "Incorrect string" << endl;

    rewind(stdin);
    cout << "Cost, UAN: ";
    cin >> price;

    rewind(stdin);
    cout << "Number of SIM-cards: ";
    cin >> simCardsNumber;

    rewind(stdin);
    cout << "Screen resolution, pixels: ";
    cin >> resolution;

    rewind(stdin);
    cout << "Battery capacity, mAh: ";
    cin >> capacity;
}

```



```

void Phone::setTitle(string newTitle) { title = newTitle; }
string Phone::getTitle() const { return title; }

void Phone::setPrice(unsigned int newPrice) { price = newPrice; }
unsigned int Phone::getPrice() const { return price; }

void Phone::setSimCardsNumber(unsigned int newsimCardsNumber) { simCardsNumber =
newsimCardsNumber; }
unsigned int Phone::getSimCardsNumber() const { return simCardsNumber; }

void Phone::setResolution(unsigned int newResolution) { resolution = newResolution; }
unsigned int Phone::getResolution() const { return resolution; }

void Phone::setCapacity(unsigned int newCapacity) { capacity = newCapacity; }
unsigned int Phone::getCapacity() const { return capacity; }

```

IosPhone.h

```

/**
 * @file IosPhone.h
 * Declaration IosPhone class.
 * @author Sokolenko Dmitro
 * @version 0.3
 * @date 2019.06.06
 */

#pragma once

#include "Phone.h"

/**
 * Declaration IosPhone class with fields and methods.
 * Used Javadoc style comments to handle Doxygen.
 */
class IosPhone : public Phone {
private:
    /** Company-manufacturer name. */
    const string manufacturer = "Apple";
    /** Ios version. */
    float iosVersion;
public:
    /**
     * Default constructor
     * Used initialization lists.
     */
    IosPhone();

    /**
     * Constructor with parameters.
     * Used initialization lists.
     * @param newTitle initializes IosPhone::title.
     * @param newPrice initializes IosPhone::price.
     * @param newSimNum initializes IosPhone::simCardsNumber.
     * @param newResolution initializes IosPhone::resolution.
     * @param newCapacity initializes IosPhone::capacity.
     * @param newIosVersion initializes IosPhone::iosVersion.
     */
    IosPhone(string newTitle,
              unsigned int newPrice,
              unsigned int newSimNum,
              unsigned int newResolution,
              unsigned int newCapacity,
              float newIosVersion);

    /**
     * Copie-constructor.

```

```

* Used initialization lists.
* @param copiedPhone: its fields initialize fields current object.
*/
IosPhone(const IosPhone& copiedPhone);

/**
* Destructor.
*/
virtual ~IosPhone() override;

/**
* Overloaded comparison operator.
* @param otherIosPhone: its fields compare with fields current object.
* @return result of comparison.
*/
bool operator==(const IosPhone& otherPhone) const;

/**
* Overloaded comparison operator.
* @param otherIosPhone: its fields compare with fields current object.
* @return result of comparison.
*/
bool operator!=(const IosPhone& otherPhone) const;

/**
* Overloaded comparison operator.
* @param otherIosPhone: its fields compare with fields current object.
* @return result of comparison.
*/
bool operator==(const IosPhone* otherPhone) const;

/**
* Overloaded comparison operator.
* @param otherIosPhone: its fields compare with fields current object.
* @return result of comparison.
*/
bool operator!=(const IosPhone* otherPhone) const;

/**
* Overloaded input operator.
* @param in - reference to input stream.
* @param inputPhone - reference to Phone object.
* @return reference to input stream.
*/
friend istream& operator>>(istream& in, IosPhone& inputPhone);

/**
* Overloaded input operator.
* @param in - reference to input stream.
* @param inputPhone - reference to Phone object.
* @return reference to input stream.
*/
friend istream& operator>>(istream& in, IosPhone* inputPhone);

/**
* Overloaded output operator.
* @param out - reference to output stream.
* @param outputPhone - reference to Phone object.
* @return reference to output stream.
*/
friend ostream& operator<<(ostream& out, const IosPhone& outputPhone);

/**
* Overloaded output operator.
* @param out - reference to output stream.

```

```

* @param outputPhone - reference to Phone object.
* @return reference to output stream.
*/
friend ostream& operator<<(ostream& out, const IosPhone* outputPhone);

/**
* Assignment to object new field value.
* @param newTitle initializes Phone::title.
* @param newPrice initializes Phone::price.
* @param newSimNum initializes Phone::simCardsNumber.
* @param newResolution initializes Phone::resolution.
* @param newCapacity initializes Phone::capacity.
* @param newIosVersion initializes IosPhone::iosVersion.
*/
void setPhoneInfo(string newTitle,
    unsigned int newPrice,
    unsigned int newSimNum,
    unsigned int newResolution,
    unsigned int newCapacity,
    float newIosVersion);

/**
* Reading from the console and filling in the fields of the Phone object.
*/
virtual void input() override;

/**
* Converting information about the current object to a string.
* @return a line with information about the current object.
*/
virtual string phoneToString() const override;

/**
* Fill the fields of the current object with
* information contained in the line.
* @param phoneString - the source line with information about the object.
*/
virtual void stringToPhone(const string phoneString) override;

/**
* Get copy of field IosPhone::manufacturer.
* @return current value IosPhone::manufacturer.
*/
string getManufacturer() const;

/**
* Set the value of the variable IosPhone::iosVersion.
* @param newIosVersion is assigned the IosPhone::iosVersion field.
*/
void setIosVersion(float newIosVersion);

/**
* Get copy of field IosPhone::iosVersion.
* @return current value IosPhone::iosVersion.
*/
float getIosVersion() const;
};

```

IosPhone.cpp

```

#include "IosPhone.h"

IosPhone::IosPhone() :
    iosVersion(10.0) {}

```

```

IosPhone::IosPhone(string newTitle,
    unsigned int newPrice,
    unsigned int newSimNum,
    unsigned int newResolution,
    unsigned int newCapacity,
    float newIosVersion) :

    Phone(newTitle,
        newPrice,
        newSimNum,
        newResolution,
        newCapacity),
    iosVersion(newIosVersion) {}

IosPhone::IosPhone(const IosPhone& copiedPhone) :
    Phone(copiedPhone.title,
        copiedPhone.price,
        copiedPhone.simCardsNumber,
        copiedPhone.resolution,
        copiedPhone.capacity),
    iosVersion(copiedPhone.iosVersion) {}

IosPhone::~IosPhone() {}

void IosPhone::setPhoneInfo(string newTitle,
    unsigned int newPrice,
    unsigned int newSimNum,
    unsigned int newResolution,
    unsigned int newCapacity,
    float newIosVersion)
{
    Phone::setPhoneInfo(newTitle, newPrice, newSimNum, newResolution, newCapacity);
    iosVersion = newIosVersion;
}

bool IosPhone::operator==(const IosPhone& otherPhone) const
{
    bool isEqualTitle = this->title == otherPhone.title;
    bool isEqualPrice = this->price == otherPhone.price;
    bool isEqualSimNumb = this->simCardsNumber == otherPhone.simCardsNumber;
    bool isEqualResol = this->resolution == otherPhone.resolution;
    bool isEqualCapac = this->capacity == otherPhone.capacity;
    bool isEqualVersion = this->iosVersion == otherPhone.iosVersion;

    if (isEqualTitle &&
        isEqualPrice &&
        isEqualSimNumb &&
        isEqualResol &&
        isEqualCapac &&
        isEqualVersion)
        return true;
    else
        return false;
}

bool IosPhone::operator!=(const IosPhone& otherPhone) const
{
    return !(*this == otherPhone);
}

bool IosPhone::operator==(const IosPhone* otherPhone) const
{
    bool isEqualTitle = this->title == otherPhone->title;
    bool isEqualPrice = this->price == otherPhone->price;

```

```

bool isEqualSimNumb = this->simCardsNumber == otherPhone->simCardsNumber;
bool isEqualResol = this->resolution == otherPhone->resolution;
bool isEqualCapac = this->capacity == otherPhone->capacity;
bool isEqualVersion = this->iosVersion == otherPhone->iosVersion;

    if (isEqualTitle &&
        isEqualPrice &&
        isEqualSimNumb &&
        isEqualResol &&
        isEqualCapac &&
        isEqualVersion)
        return true;
    else
        return false;
}

bool IosPhone::operator!=(const IosPhone* otherPhone) const
{
    return !(this == otherPhone);
}

istream& operator>>(istream& in, IosPhone& inputPhone)
{
    in.ignore(3);

    getline(in, inputPhone.title, '|');
    inputPhone.title.erase(inputPhone.title.end() - 1);
    in.ignore(1);

    in >> inputPhone.price;
    in.ignore(3);

    in >> inputPhone.simCardsNumber;
    in.ignore(3);

    in >> inputPhone.resolution;
    in.ignore(3);

    in >> inputPhone.capacity;
    in.ignore(11);

    in >> inputPhone.iosVersion;

    return in;
}

istream& operator>>(istream& in, IosPhone* inputPhone)
{
    in.ignore(3);

    getline(in, inputPhone->title, '|');
    inputPhone->title.erase(inputPhone->title.end() - 1);
    in.ignore(1);

    in >> inputPhone->price;
    in.ignore(3);

    in >> inputPhone->simCardsNumber;
    in.ignore(3);

    in >> inputPhone->resolution;
    in.ignore(3);

    in >> inputPhone->capacity;
    in.ignore(11);
}

```

```

        in >> inputPhone->iosVersion;

        return in;
    }

    ostream& operator<<(ostream& out, const IosPhone& outputPhone)
    {
        out << "I# " << outputPhone.title << " | " << outputPhone.price << " | " <<
        outputPhone.simCardsNumber <<
            " | " << outputPhone.resolution << " | " << outputPhone.capacity << " | "
        << outputPhone.manufacturer <<
            " | " << outputPhone.iosVersion;

        return out;
    }

    ostream& operator<<(ostream& out, const IosPhone* outputPhone)
    {
        out << "I# " << outputPhone->title << " | " << outputPhone->price << " | " <<
        outputPhone->simCardsNumber <<
            " | " << outputPhone->resolution << " | " << outputPhone->capacity << " | "
        << outputPhone->manufacturer <<
            " | " << outputPhone->iosVersion;

        return out;
    }

    void IosPhone::input()
    {
        Phone::input();

        cout << "Manufacturer: Apple" << endl;

        rewind(stdin);
        cout << "Ios version: ";
        cin >> iosVersion;

        cout << endl << endl;
    }

    string IosPhone::phoneToString() const
    {
        stringstream phoneStream;
        string phoneString;

        phoneStream << "I# " << title << " | " << price << " | " << simCardsNumber <<
            " | " << resolution << " | " << capacity << " | " << manufacturer << " | "
        <<
            iosVersion;

        getline(phoneStream, phoneString);

        return phoneString;
    }

    void IosPhone::stringToPhone(const string phoneString)
    {
        stringstream phoneStream;
        phoneStream << phoneString;

        phoneStream.ignore(3);

        getline(phoneStream, title, '|');
        title.erase(title.end() - 1);
    }

```

```
    phoneStream.ignore(1);

    phoneStream >> price;
    phoneStream.ignore(3);

    phoneStream >> simCardsNumber;
    phoneStream.ignore(3);

    phoneStream >> resolution;
    phoneStream.ignore(3);

    phoneStream >> capacity;
    phoneStream.ignore(11);

    phoneStream >> iosVersion;
}

string IosPhone::getManufacturer() const { return manufacturer; }

void IosPhone::setIosVersion(float newIosVersion) { iosVersion = newIosVersion; }
float IosPhone::getIosVersion() const { return iosVersion; }
```