

МІНІСТЕРСТВО ОСВІТИ І НАУКИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ “ХПІ”

Кафедра “Обчислювальна техніка та програмування”

Розрахункове завдання з дисципліни  
«Основи програмування ч.2»

Пояснювальна записка  
ЄСПД ГОСТ 19.404–79(СТЗВО – ХПІ – 30.05-2021 ССОНП)  
КІТ.120А.13-01 90 01-1 -ЛЗ

Виконав:  
студент групи КІТ-120А  
Зозуля Ігор Дмитрович

Перевірив:  
Давидов В'ячеслав Вадимович

Харків 2021

# **Розрахункове завдання**

***Тема:** Розробка інформаційно-довідкової системи*

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

## **1. Призначення та галузь застосування**

Розроблена інформаційно-довідкова система, являє собою колекцію годинників та методи роботи з нею. Згідно заданого завдання колекція має методи: пошуку класичних годинників, пошуку годинника з ціною менше за 400 доларів, пошуку швейцарського годиннику зі скелетомом та сортування списку за вказаним користувачем критерієм і напрямком. Також є методи роботи зі списком, які дають змогу: видалити заданий користувачем годинник зі списку, або весь список, додати годинник до списку, замінити чи отримати годинник по індексу.

Створену інформаційно-довідкову систему можна застосовувати в галузі роботи з годинниками, наприклад інтернет-магазин, або веб-каталог.

## **2. Постановка завдання до розробки**

### **1.1 Загальне завдання**

1) З розділу "Розрахункове завдання / Індивідуальні завдання", відповідно до варіанта завдання, обрати прикладну галузь;

2) Для прикладної галузі розробити розгалужену ієрархію класів, що описана у завданні та складається з одного базового класу та двох спадкоємців. Класи повинні мати перевантажені оператори введення-виведення даних та порівняння;

3) 3. Розробити клас-список `List.[h/cpp]`, що буде включати до себе масив (STL-колекцію) вказівників до базового класу. А також базові методи роботи з списком: а) очистка списку б) відображення списку в) додання/видалення/отримання/оновлення елементу;

4) Розробити клас-контролер `Controller.[h/cpp]`, що буде включати колекцію розроблених класів, та наступні методи роботи з цією колекцією: а) читання даних з файлу та їх запис у контейнер (STL-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 вказаними критеріями (три критерія, щоприсутні у кожному варіанті);

5) Розробити клас `Menu.[h/cpp]`, який має відображати діалогове меню для демонстрації реалізованих функцій класу контролера;

6) Оформити схеми алгоритмів функцій класів контролера (за необхідністю), тесту-контролера та діалогового меню;

7) Оформити документацію: пояснювальну записку.

#### **Додаткові вимоги на оцінку «відмінно»:**

- виконати перевірку вхідних даних за допомогою регулярних виразів.
- критерій для пошуку та сортування задавати у вигляді функтора;
- розробити клас-тестер контролеру `ControllerTest.cpp`, основною метою якого буде перевірка коректності роботи класу-контролера.

### **1.2 Індивідуальне завдання**

– Варіант 13. "Годинник"

- Поля базового класу:
  - Чи є водонепроникним (наприклад: так, ні)

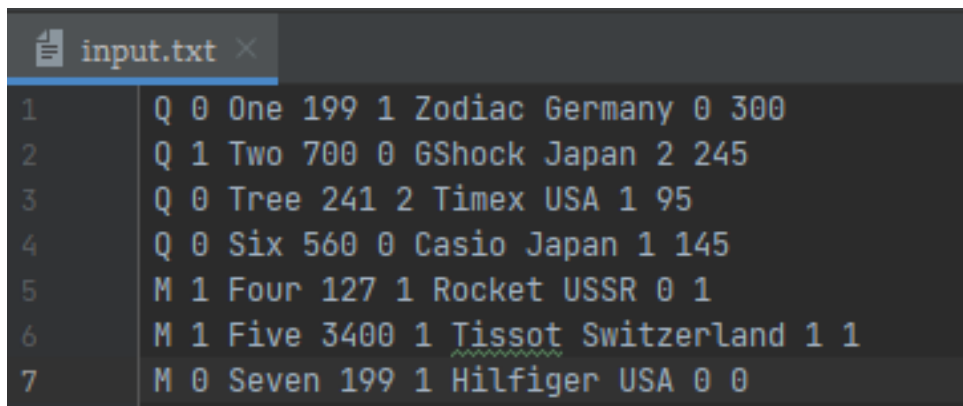
- Назва моделі (наприклад: EFR-526L-1AVUEF, CS 55)
- Ціна, USD (наприклад: 300, 1200)
- Виробник (структура, що містить назву фірми та країну її місцезнаходження)
- Стиль (один з переліку: спорт, класика, захищений)
- Спадкоємець 1 – Механічний годинник. Додаткові поля:
  - Наявність автопідзаводу (наприклад: так, ні)
  - Наявність скелетону (наприклад: так, ні)
- Спадкоємець 2 – Кварцовий годинник. Додаткові поля:
  - Тип батареї (один з переліку: сонячна, звичайна)
  - Ємність батареї, mAh (наприклад: 250, 330)
- Методи роботи з колекцією:
  1. Знайти годинники з ціною менше 400\$
  2. Знайти всі швейцарські годинники зі скелетоном
  3. Знайти всі годинники стилю «Класика»

### **3. Опис вхідних та вихідних даних**

#### **3.1 Опис вхідних даних**

Під час запуску програма зчитує данні з файлу за шляхом «../assets/input.txt». В файл повинен містити наступні параметри: перший, це символ ('М' (Mechanical) чи 'Q' (Quartz)), який позначає тип вхідного об'єкту, наступний цифра 1 чи 0, що позначає чи є годинник вологозахищеним (1 – так, 0 – ні), потім модель годиннику, його ціна та стиль (0 – захищений, 1 – класичний, 2 – спортивний), назва фірми та країну її місце знаходження, потім в залежності від типу, якщо обраний 'М' (механічний годинник): цифра 1 чи 0, що позначає чи є в годинника автопідзавод (1 – так, 0 – ні), потім цифра 1 чи 0, що позначає чи є в годинника скелетон(1 – так, 0 – ні); якщо обраний 'Q' (кварцовий

годинник): тип акумулятору (0 – графеновий, 1 – Li-іонний, 2 – сонячний) та його ємність в mAh. Приклад файлу з вхідними даними продемонстровані на рисунку 1.

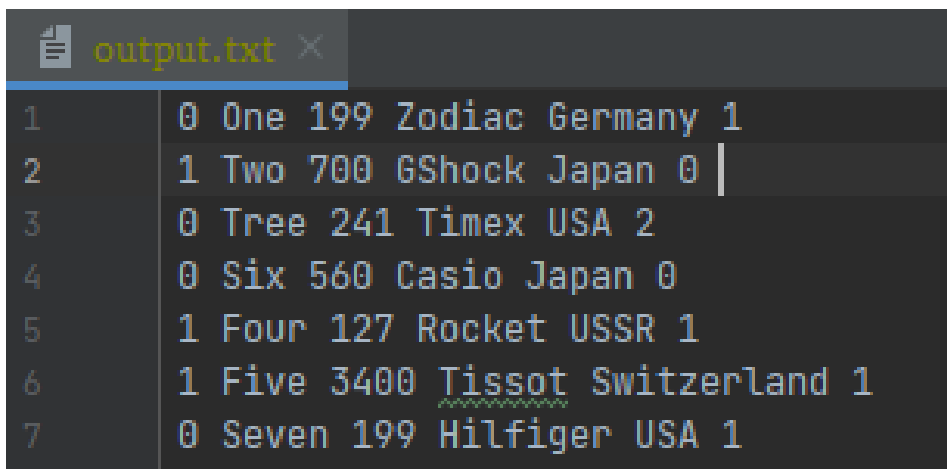


```
input.txt x
1 Q 0 One 199 1 Zodiac Germany 0 300
2 Q 1 Two 700 0 GShock Japan 2 245
3 Q 0 Tree 241 2 Timex USA 1 95
4 Q 0 Six 560 0 Casio Japan 1 145
5 M 1 Four 127 1 Rocket USSR 0 1
6 M 1 Five 3400 1 Tissot Switzerland 1 1
7 M 0 Seven 199 1 Hilfiger USA 0 0
```

Рисунок 1 – Приклад вхідного файлу

### 3.2 Опис вихідних даних

Вихідні данні записуються у файл розташований за шляхом «../dist/output.txt», в тому ж порядку, в якому задані данні у списку. Приклад файлу з вихідними даними продемонстровані на рисунку 2.



```
output.txt x
1 0 One 199 Zodiac Germany 1
2 1 Two 700 GShock Japan 0
3 0 Tree 241 Timex USA 2
4 0 Six 560 Casio Japan 0
5 1 Four 127 Rocket USSR 1
6 1 Five 3400 Tissot Switzerland 1
7 0 Seven 199 Hilfiger USA 1
```

Рисунок 2 – Приклад вихідного файлу

## 4. Опис складу технічних та програмних засобів

### 4.1 Функціональне призначення

Програма виводить меню можливих дій с колекцією, та в залежності від отриманих від користувача даних виконує методи із загального та індивідуально завдань.

### 4.2 Опис логічної структури програми

*Головна функція*(`main()`). Виводить меню та викликає функції `gettingPoint()`, який отримує від користувача номер дії, яку необхідно виконати з колекцією. І в залежності від отриманого результату функція `main()` викликає необхідну функцію.

*Метод демонстрації вмісту контейнеру* (`showListMenu()`). Виводить вміст усього контейнеру на екран.

*Метод зчитування даних з файлу* (`readFromFile()`). Зчитує данні з файлу та записує їх у контейнер. Схема алгоритму методу подана на рисунку 3.

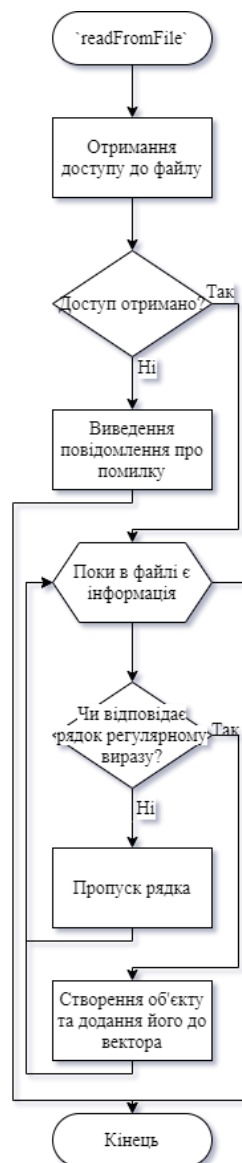


Рисунок 3 – Схема алгоритму методу `readFromFile`

Метод запису даних до файлу (`writeToFile()`). Записує данні з контейнеру до файлу. Схема алгоритму методу подана на рисунку 4.

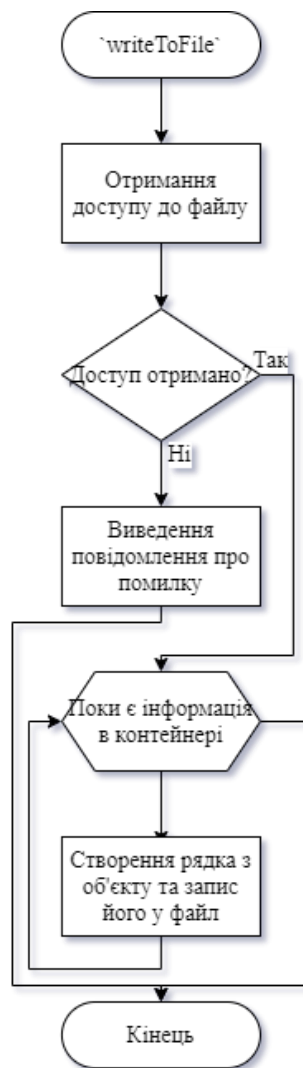


Рисунок 4 — Схема алгоритму методу writeToFile

*Метод сортування вмісту контейнера* (sorting(char way, int criterion)). Сортує данні відповідно критерію (1 – сортування за вологозахистом; 2 – сортування за ціною; 3 – сортування за моделлю; 4 – сортування за фірмою; 5 – сортування за країною; 6 – сортування за стилем) та напрямку сортування ('<' – від меншого до більшого; '>' – від більшого до меншого). Схема алгоритму методу подана на рисунку 5. (Реалізація методу знаходиться в додатку А та реалізація функторів знаходиться в додатку Б)



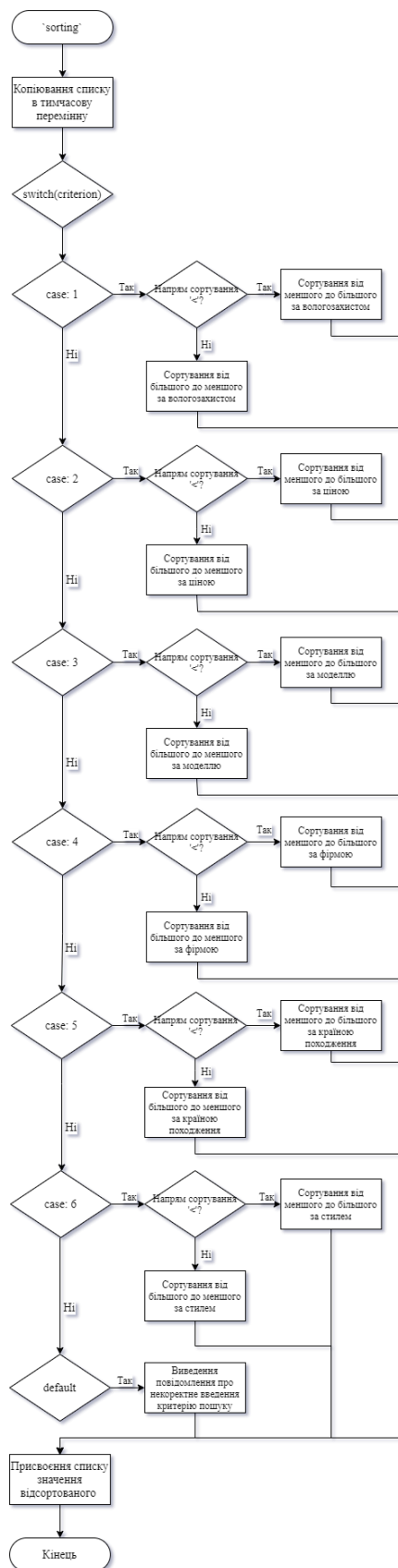


Рисунок 5 — Схема алгоритму методу sorting

*Метод додавання ланки до контейнера* (`addLink(watch *watchLink)`). Додає ланку до контейнеру, використовуючи вже існуючий об'єкт. Схема алгоритму методу подана на рисунку 6.

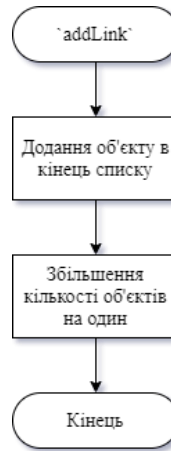


Рисунок 6 — Схема алгоритму методу `addLink`

*Метод видалення ланки за індексом* (`deleteLink(int index)`). Видаляє ланку з контейнеру за індексом. Схема алгоритму методу подана на рисунку 7.

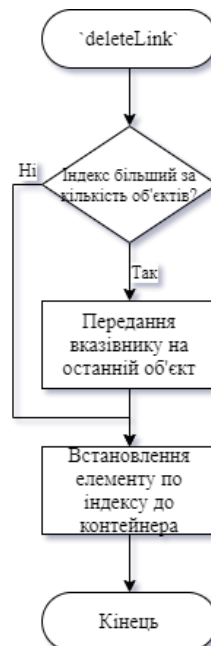


Рисунок 7 — Схема алгоритму методу `deleteLink`



*Метод очищення всього списку (clearList()). Очищає інформацію з усього списку.*

### 4.3 Структура проекту

```
.
├─ CMakeLists.txt
├─ Doxyfile
├─ Makefile
├─ assets
│   └─ input.txt
├─ dist
│   └─ main.bin
├─ doc
│   └─ RGZ.docx
│   └─ assets
│       └─ RGZ.drawio
│       └─ addLink.png
│       └─ afterSort.png
│       └─ befoureSort.png
│       └─ classicWatches.png
│       └─ deleteLink.png
│       └─ find.png
│       └─ output samFile.png
│       └─ readFromFile.png
│       └─ sample file.png
│       └─ sorting.png
│       └─ swissWithSkeleton.png
│       └─ watchUnder400.png
│       └─ writeToFile.png
│   └─ ~$RGZ.docx
├─ src
│   └─ controller.cpp
│   └─ controller.h
│   └─ list.cpp
│   └─ list.h
│   └─ main.cpp
│   └─ menu.cpp
│   └─ menu.h
│   └─ watch.cpp
│   └─ watch.h
└─ test
    └─ test.cpp
```

## 4.4 Варіанти використання

Для демонстрації результатів використовується IDE Clion. Нижче наводиться послідовність дій запуску програми.

*Крок 1* (рисунок 11). Виконаємо методи пошуку.

<Watch_under_400USD>								
Waterproof	Model	Firm	Country	Cost	Style	Battery/Self wilding	Capacity/	Skeleton
No	One	Zodiac	Germany	199USD	Classic	Graphene	300mAh	
No	Tree	Timex	USA	241USD	Sport	Li-Ion	95mAh	
Have	Four	Rocket	USSR	127USD	Classic	No		Have
No	Seven	Hilfiger	USA	199USD	Classic	No		No

а) Пошук годинників з ціною менше 400 доларів

<Classic_watches>								
Waterproof	Model	Firm	Country	Cost	Style	Battery/Self wilding	Capacity/	Skeleton
No	One	Zodiac	Germany	199USD	Classic	Graphene	300mAh	
Have	Four	Rocket	USSR	127USD	Classic	No		Have
Have	Five	Tissot	Switzerland	3400USD	Classic	Have		Have
No	Seven	Hilfiger	USA	199USD	Classic	No		No

б) Пошук класичних годинників

<Switzerland_watches_with_skeleton>								
Waterproof	Model	Firm	Country	Cost	Style	Battery/Self wilding	Capacity/	Skeleton
Have	Five	Tissot	Switzerland	3400USD	Classic	Have		Have

в) Пошук швейцарських годинників зі скелетоном

*Рисунок 11* — результати роботи методів пошуку

*Крок 2* (рисунок 12). Виконаємо метод сортування.

<List>								
Waterproof	Model	Firm	Country	Cost	Style	Battery/Self wilding	Capacity/	Skeleton
No	One	Zodiac	Germany	199USD	Classic	Graphene	300mAh	
Have	Two	GShock	Japan	700USD	Armoured	Solar	245mAh	
No	Tree	Timex	USA	241USD	Sport	Li-Ion	95mAh	
No	Six	Casio	Japan	560USD	Armoured	Li-Ion	145mAh	
Have	Four	Rocket	USSR	127USD	Classic	No		Have
Have	Five	Tissot	Switzerland	3400USD	Classic	Have		Have
No	Seven	Hilfiger	USA	199USD	Classic	No		No

а) Список до сортування

<List>									
Waterproof	Model	Firm	Country	Cost	Style	Battery/Self wilding	Capacity/	Skeleton	
Have	Five	Tissot	Switzerland	3400USD	Classic				
Have	Two	GShock	Japan	700USD	Armoured	Solar	245mAh		
No	Six	Casio	Japan	560USD	Armoured	Li-Ion	145mAh		
No	Tree	Timex	USA	241USD	Sport	Li-Ion	95mAh		
No	One	Zodiac	Germany	199USD	Classic	Graphene	300mAh		
No	Seven	Hilfiger	USA	199USD	Classic			No	No
Have	Four	Rocket	USSR	127USD	Classic			No	Have

б) Список після сортування

Рисунок 12 — результат роботи методу сортування

## Висновки

Виконуючи розрахункове завдання я закріпив отримані знання з дисципліни «Програмування» та отримав практичні навички шляхом виконання типового комплексного завдання.

## Додаток А. Реалізація методу `sorting(char way, int criterion)`

```
void controller::sorting(char way, int criterion) {
    auto tmp = watchList.getList();
    switch (criterion) {
        case 1:
            if (way == '<') {
                sort(tmp.begin(), tmp.end(), functorMoreWaterproof);
            } else {
                sort(tmp.begin(), tmp.end(), functorLessWaterproof);
            }
            break;
        case 2:
            if (way == '<') {
                sort(tmp.begin(), tmp.end(), functorMoreCost);
            } else {
                sort(tmp.begin(), tmp.end(), functorLessCost);
            }
            break;
        case 3:
            if (way == '<') {
                sort(tmp.begin(), tmp.end(), functorMoreModel);
            } else {
                sort(tmp.begin(), tmp.end(), functorLessModel);
            }
            break;
        case 4:
            if (way == '<') {
                sort(tmp.begin(), tmp.end(), functorMoreFirm);
            } else {
                sort(tmp.begin(), tmp.end(), functorLessFirm);
            }
            break;
        case 5:
            if (way == '<') {
                sort(tmp.begin(), tmp.end(), functorMoreCountry);
            } else {
                sort(tmp.begin(), tmp.end(), functorLessCountry);
            }
            break;
        case 6:
            if (way == '<') {
                sort(tmp.begin(), tmp.end(), functorMoreStyle);
            } else {
                sort(tmp.begin(), tmp.end(), functorLessStyle);
            }
            break;
        default:
            cout << ("| Incorrect variant!!!") << endl;
            break;
    }
    this->watchList.setLinks(tmp);
}
```

## Додаток Б. Реалізація функторів

```
bool functorLessCost(watch *one, watch *two) {
    bool result = false;
    auto *tempOne = (watch *) one->copy();
    auto *tempTwo = (watch *) two->copy();
    if (tempOne->getCost() > tempTwo->getCost()) {
        result = true;
        delete tempOne;
        delete tempTwo;
    }
    return result;
}

bool functorMoreCost(watch *one, watch *two) {
    bool result = false;
    auto *tempOne = (watch *) one->copy();
    auto *tempTwo = (watch *) two->copy();
    if (tempOne->getCost() < tempTwo->getCost()) {
        result = true;
        delete tempOne;
        delete tempTwo;
    }
    return result;
}

bool functorLessWaterproof(watch *one, watch *two) {
    bool result = false;
    auto *tempOne = (watch *) one->copy();
    auto *tempTwo = (watch *) two->copy();
    if (tempOne->getWaterproof() > tempTwo->getWaterproof()) {
        result = true;
        delete tempOne;
        delete tempTwo;
    }
    return result;
}

bool functorMoreWaterproof(watch *one, watch *two) {
    bool result = false;
    auto *tempOne = (watch *) one->copy();
    auto *tempTwo = (watch *) two->copy();
    if (tempOne->getWaterproof() < tempTwo->getWaterproof()) {
        result = true;
        delete tempOne;
        delete tempTwo;
    }
    return result;
}

bool functorLessModel(watch *one, watch *two) {
    bool result = false;
    auto *tempOne = (watch *) one->copy();
    auto *tempTwo = (watch *) two->copy();
    if (tempOne->getModel() > tempTwo->getModel()) {
        result = true;
        delete tempOne;
        delete tempTwo;
    }
    return result;
}

bool functorMoreModel(watch *one, watch *two) {
    bool result = false;
    auto *tempOne = (watch *) one->copy();
    auto *tempTwo = (watch *) two->copy();
    if (tempOne->getModel() < tempTwo->getModel()) {
        result = true;
        delete tempOne;
        delete tempTwo;
    }
    return result;
}

bool functorLessFirm(watch *one, watch *two) {
    bool result = false;
```



```

        auto *tempOne = (watch *) one->copy();
        auto *tempTwo = (watch *) two->copy();
        if (tempOne->getManufacturer().getFirm() > tempTwo->getManufacturer().getFirm()) {
            result = true;
            delete tempOne;
            delete tempTwo;
        }
        return result;
    }
    bool functorMoreFirm(watch *one, watch *two) {
        bool result = false;
        auto *tempOne = (watch *) one->copy();
        auto *tempTwo = (watch *) two->copy();
        if (tempOne->getManufacturer().getFirm() < tempTwo->getManufacturer().getFirm()) {
            result = true;
            delete tempOne;
            delete tempTwo;
        }
        return result;
    }
    bool functorLessCountry(watch *one, watch *two) {
        bool result = false;
        auto *tempOne = (watch *) one->copy();
        auto *tempTwo = (watch *) two->copy();
        if (tempOne->getManufacturer().getCountry() > tempTwo->getManufacturer().getCountry()) {
            result = true;
            delete tempOne;
            delete tempTwo;
        }
        return result;
    }
    bool functorMoreCountry(watch *one, watch *two) {
        bool result = false;
        auto *tempOne = (watch *) one->copy();
        auto *tempTwo = (watch *) two->copy();
        if (tempOne->getManufacturer().getCountry() < tempTwo->getManufacturer().getCountry()) {
            result = true;
            delete tempOne;
            delete tempTwo;
        }
        return result;
    }
    bool functorLessStyle(watch *one, watch *two) {
        bool result = false;
        auto *tempOne = (watch *) one->copy();
        auto *tempTwo = (watch *) two->copy();
        if (tempOne->getStyle() > tempTwo->getStyle()) {
            result = true;
            delete tempOne;
            delete tempTwo;
        }
        return result;
    }
    bool functorMoreStyle(watch *one, watch *two) {
        bool result = false;
        auto *tempOne = (watch *) one->copy();
        auto *tempTwo = (watch *) two->copy();
        if (tempOne->getStyle() < tempTwo->getStyle()) {
            result = true;
            delete tempOne;
            delete tempTwo;
        }
        return result;
    }
}

```

## Додаток В. Реалізація методу find(int criterion)

```
vector<watch *> controller::find(int criterion) const {
    vector<watch *> result;
    vector<watch *> tmp = this->watchList.getList();
    bool flag = true;
    auto iter = tmp.begin();
    while (true) {
        switch (criterion) {
            case 1:
                iter = find_if(iter, tmp.end(), findPriceHelp);
                break;
            case 2:
                iter = find_if(iter, tmp.end(), findClassicHelp);
                break;
            case 3:
                iter = find_if(iter, tmp.end(), findSwitzerlandWithSkeletonHelp);
                break;
            default:
                cout << ("| Criterion entered incorrectly!!!") << endl;
                return result;
        }
        if (iter == tmp.end()) {
            break;
        }
        flag = false;
        result.push_back((mechanicalWatches *) *iter);
        ((mechanicalWatches *) *iter)->show();
        cout << endl;
        iter++;
    }
    if (flag) {
        cout << "|There are no suitable watches!" << endl;
    }
    while (!tmp.empty()) {
        tmp.pop_back();
    }
    tmp.clear();
    tmp.shrink_to_fit();
    return result;
}
```