

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.

должность, уч. степень, звание

подпись, дата

К. А. Курицын

инициалы, фамилия

ОТЧЁТ О КУРСОВОЙ РАБОТЕ
«ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ»

по дисциплине: «Технологии программирования»

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

1742

подпись, дата

А. Д. Пивоваров

инициалы, фамилия

Санкт-Петербург 2020

Содержание

1. Постановка задачи.....	2
1.1. Функциональные требования.....	2
1.2. Спецификация.....	2
2. Основная часть.....	3
2.1. Условия проведения.....	3
2.2. Программа и методика испытаний.....	3
3. Листинг программы.....	4
4. Список использованной литературы.....	14
Приложение 1.....	15

1. Постановка задачи

Создать класс «Олимпийские игры». Клиент выбирает, за какую страну-участницу болеть. Участники из России, Канады, Франции, Финляндии и Японии участвуют во всех зимних видах олимпийского спорта: биатлон, скелетон, конькобежный спорт, фигурное катание. Из внешнего файла загружаются данные по каждому спортсмену: ФИО, страна, сила, ловкость, удача, выносливость. Во время соревнования по каждому виду спорта вычисляется, какой спортсмен какое место занял. Во время игр клиент наблюдает за соревнованиями и по результатам узнает, какая страна какое место заняла; выводится сообщение, выиграла ли текущее соревнование выбранная страна. По суммарным результатам всех соревнований сообщается – выиграл клиент или нет. Использовать паттерн «Наблюдатель» для реализации клиента. Результаты всех игр записываются в выходной файл.

1.1. Функциональные требования

- 1.1.1. Создать класс «Олимпийские игры»;
- 1.1.2. Реализовать пользовательское меню: выбор страны, за которую хочет болеть клиент; вывод результатов соревнований, вывод суммарных результатов;
- 1.1.3. Входные данные (данные и характеристики каждого спортсмена) загружаются из внешнего файла;
- 1.1.4. Результаты всех соревнований записываются в выходной файл;

1.2. Спецификация

- 1.2.1. Реализация клиента с использованием паттерна «Наблюдатель»;
- 1.2.2. При запуске программы клиент видит: общее количество спортсменов, участвующих в соревновании; данные и характеристики каждого спортсмена (ФИО, страна, сила, ловкость, удача, выносливость); после вывода информации клиент выбирает страну, за которую он будет болеть;
- 1.2.3. После выбора страны пользователь видит результаты соревнований по каждому виду спорта и суммарные результаты (выиграла выбранная страна или нет; в случае, если страна проиграла, на экран выводится информация о стране-победителе);
- 1.2.4. Загрузка данных о каждом спортсмене происходит из файла «ath.json». Json – текстовый формат обмена данными, основанный на JavaScript. Несмотря на происхождение от JavaScript, json является независимым от языка форматом и может использоваться практически с любым языком программирования. Формат используется для кроссплатформенной разработки.
- 1.2.5. Запись результатов соревнований по каждому виду спорта происходит в файл «results.txt»
- 1.2.6. Файл «ath.json» содержит информацию о каждом спортсмене, участвующем в играх: ФИО, страна, сила, ловкость, удача, выносливость;
- 1.2.7. Класс «People» содержит поля, в которых находится информация о спортсмене: ФИО, страна, сила, ловкость, удача, выносливость;
- 1.2.8. Если клиент вводит число, которого не содержится в списке стран, программа выводит ошибку и предлагает выбрать страну еще раз.

2. Основная часть

2.1. Условия проведения

Проведение на тестовом пользовательском компьютере.

2.2. Программа и методика испытания

№	Сценарий проверки	Ожидаемый результат	Результат	№ спецификации
1	Запуск программы	Чтение из файла данных о спортсменах и вывод их на экран; вывод сообщения на экран	Рисунок 1	1.2.2, 1.2.4, 1.2.6
2	Ввод номера, отсутствующего в списке	Вывод сообщения об ошибке; повторный вывод сообщения на экран	Рисунок 2	1.2.8
3	Выбор страны, присутствующей в списке	Вывод на экран результатов каждого соревнования; вывод на экран суммарного результата;	Рисунок 3	1.2.1, 1.2.3, 1.2.5, 1.2.7
4	Запуск программы	Чтение из файла данных о спортсменах и вывод их на экран; вывод сообщения на экран	Рисунок 1	1.2.2, 1.2.4, 1.2.6
5	Выбор страны, присутствующей в списке	Вывод на экран результатов каждого соревнования; вывод на экран суммарного результата и сообщения о том, какая страна оказалась победителем (проверка случая, когда клиент выбрал проигравшую страну)	Рисунок 4	1.2.1, 1.2.3, 1.2.5, 1.2.7

3. Листинг программы

GitHub: <https://github.com/Arem322/Curs>

main.cpp

```
#include <iostream>

#include "OlympicGames.h"
#include "Client.h"

string readCountry();

int main() {

    string country;
    Client *client = new Client();

    OlympicGames *games = new OlympicGames();
    games->subscribe(client);

    while (country == "") {
        cout << "Choose country:" << endl;
        cout << "1 - Russia, 2 - Canada, 3 - Finland, 4 - France, 5 - Japan; 0 - Exit" << endl;
        country = readCountry();
        if (country == "exit") {
            cout << "Exit from programm" << endl;
            return 0;
        }
    }

    client->setChoice(country);
    cout << "Your choose: " << country << endl << endl;

    games->getResults();
    free(games);

    return 0;
}

string readCountry() {
    int num = 0;
    cin >> num;

    switch (num) {
        case 1:
            return "Russia";
            break;
        case 2:
            return "Canada";
            break;
        case 3:
            return "Finland";
            break;
        case 4:
            return "France";
            break;
        case 5:
            return "Japan";
    }
}
```

```

        break;
    case 0:
        return "exit";
    default:
        cout << "Country error" << endl;
        return "";
    }
}

```

People.h

```

#ifndef PEOPLE_H
#define PEOPLE_H

#include <string>
#include <iostream>

#include "json.hpp"

using json = nlohmann::json;

using namespace std;

class People {
    string fio;
    string country;
    string sport;
public:
    const string &getSport() const;

    void setSport(const string &sport);

private:
    int force;
    int agility;
    int luck;
    int stamina;

public:
    const string &getFio() const;

    void setFio(const string &fio);

    const string &getCountry() const;

    void setCountry(const string &country);

    int getForce() const;

    void setForce(int force);

    int getAgility() const;

    void setAgility(int agility);

    int getLuck() const;

    void setLuck(int luck);

```

```

    int getStamina() const;

    void setStamina(int stamina);

    void parseJson2People(json j);

    friend ostream &operator<<(ostream &os, const People &people);
};

#endif //PEOPLE_H

```

People.cpp

```

#include "People.h"

const string &People::getFio() const {
    return fio;
}

void People::setFio(const string &fio) {
    People::fio = fio;
}

const string &People::getCountry() const {
    return country;
}

void People::setCountry(const string &country) {
    People::country = country;
}

int People::getForce() const {
    return force;
}

void People::setForce(int force) {
    People::force = force;
}

int People::getAgility() const {
    return agility;
}

void People::setAgility(int agility) {
    People::agility = agility;
}

int People::getLuck() const {
    return luck;
}

void People::setLuck(int luck) {
    People::luck = luck;
}

int People::getStamina() const {
    return stamina;
}

```

```

void People::setStamina(int stamina) {
    People::stamina = stamina;
}

void People::parseJson2People(json j) {
    this->fio = j["fio"];
    this->agility = j["a"];
    this->force = j["f"];
    this->luck = j["l"];
    this->stamina = j["s"];
    this->country = j["country"];
    this->sport = j["sport"];
}

ostream &operator<<(ostream &os, const People &people) {
    os << "fio: " << people.fio << " Sport:" << people.sport << " country: " <<
    people.country << " force: " << people.force << " agility: "
    << people.agility << " luck: " << people.luck << " stamina: " <<
    people.stamina;
    return os;
}

const string &People::getSport() const {
    return sport;
}

void People::setSport(const string &sport) {
    People::sport = sport;
}

```

OlympicGames.h

```

#ifndef OLYMPICGAMES_H
#define OLYMPICGAMES_H

#include "People.h"
#include "Client.h"

class OlympicGames {

    People **peoples;
    int count;

    Client **clients;
    int peopleCount;

    enum {RUSSIA, CANADA, FINLAND, FRANCE, JAPAN};
    int results[5];

    string readFromFile(string filename);

    const string sports[4] = {
        "Skeleton",
        "Biatlon",
        "Skating",
        "IceSkating"
    };

public:
    void notify();

```



```

void subscribe(Client *client);
void unsubscribe(int index);

void addPeople(People &p);

OlympicGames();

int getCount() const;
string sort(People **peoples, int len, string &sport);
int filterSportParam(People *current, string sport);

void addCountryResult(string &country);
string getWinner();

void getResults();
void writeResult(string &line);

virtual ~OlympicGames();

};

#endif //OLYMPICGAMES_H

```

OlympicGames.cpp

```

#include "OlympicGames.h"

#include <fstream>
#include "json.hpp"

using json = nlohmann::json;

void OlympicGames::notify() {
    string winner = this->getWinner();

    for (int i = 0; i < count; ++i) {
        clients[i]->update(winner);
    }
}

int OlympicGames::getCount() const {
    return count;
}

void OlympicGames::unsubscribe(int index) {
    if (index < 0 || index >= this->count) {
        cout << "Error: Invalid array index" << endl;
    } else {
        int k = 0;
        Client **copy = new Client*[this->count-1];
        for (int i = 0; i < this->count; ++i) {
            if (i != index) {
                copy[k] = this->clients[i];
                k++;
            }
        }
        delete[] clients;
        clients = copy;
    }
}

```

```

        } else {
            Client *removed = copy[k];
            free(removed);
        }
    }
    this->count--;
    this->clients = new Client*[this->count];
    for (int j = 0; j < this->count; ++j) {
        this->clients[j] = copy[j];
    }
}

void OlympicGames::addPeople(People &p) {
    this->peopleCount++;
    People **copy = new People*[this->peopleCount - 1];
    for (int i = 0; i < this->peopleCount - 1; ++i) {
        copy[i] = this->peoples[i];
    }
    this->peoples = new People*[this->peopleCount];
    for (int j = 0; j < this->peopleCount - 1; ++j) {
        this->peoples[j] = copy[j];
    }
    this->peoples[this->peopleCount - 1] = &p;
}

void OlympicGames::subscribe(Client *client) {
    this->count++;
    Client **copy = new Client*[this->count - 1];
    for (int i = 0; i < this->count - 1; ++i) {
        copy[i] = this->clients[i];
    }
    this->clients = new Client*[this->count];
    for (int j = 0; j < this->count - 1; ++j) {
        this->clients[j] = copy[j];
    }
    this->clients[this->count - 1] = client;
}

OlympicGames::OlympicGames() {
    for (int i = 0; i < 5; ++i) {
        results[i] = 0;
    }
    this->peopleCount = 0;
    this->count = 0;
    this->clients = new Client*[this->count];
    this->peoples = new People*[this->peopleCount];

    string textJson = this->readFromFile("ath.json");
    json j = json::parse(textJson);
    int size = j["size"];
    j = j["peoples"];

    cout << size << " athletes read;" << endl;
    for (int i = 0; i < size; ++i) {
        json current = j[i];
        People *p = new People();
        p->parseJson2People(current);
        this->addPeople(*p);
    }
}

```

```

        for (int k = 0; k < peopleCount; ++k) {
            cout << *peoples[k] << endl;
        }
    }

string OlympicGames::readFromFile(string filename) {
    ifstream file;
    file.open("../" + filename);
    string text = "";
    string line;

    if (file.fail()) {
        cerr << "File reading failed" << endl;
        return "";
    }

    while (getline(file, line))
    {
        text += line;
    }

    file.close();
    return text;
}

void OlympicGames::getResults() {
    string results = "";
    for (int j = 0; j < 4; ++j) {
        int len = 0;
        string sport = this->sports[j];
        string winnerCountry = "Nobody";
        int max = -1;
        for (int i = 0; i < peopleCount; ++i) {
            People *current = this->peoples[i];
            int result = 0;

            if (current->getSport() != sport) {
                continue;
            } else {
                len++;
            }

            if (sport == "Skeleton") {
                result = current->getForce();
            } else if (sport == "Biatlon") {
                result = current->getAgility();
            } else if (sport == "Skating") {
                result = current->getLuck();
            } else if (sport == "IceSkating") {
                result = current->getStamina();
            }

            if (result > max) {
                max = result;
                winnerCountry = current->getCountry();
            }
        }
    }
}

```

```

        int k = 0;
        People **copy = new People*[len];

        for (int i = 0; i < peopleCount; ++i) {
            People *current = this->peoples[i];
            if (current->getSport() != sport) {
                continue;
            }
            copy[k] = current;
            k++;
        }

        results += winnerCountry + " won the " + sport + "\n";
        cout << winnerCountry << " won the " << sport << endl;
        results += this->sort(copy, len, sport);

        addCountryResult(winnerCountry);
    }
    this->writeResult(results);
    this->notify();
}

void OlympicGames::writeResult(string &line) {
    ofstream file;
    file.open("../results.txt");
    file << line;
    file.close();
}

void OlympicGames::addCountryResult(string &country) {
    if (country == "Russia") {
        this->results[RUSSIA]++;
    } else if (country == "Canada") {
        this->results[CANADA]++;
    } else if (country == "Finland") {
        this->results[FINLAND]++;
    } else if (country == "France") {
        this->results[FRANCE]++;
    } else if (country == "Japan") {
        this->results[JAPAN]++;
    }
}

string OlympicGames::getWinner() {
    int max = 0;
    int index = 0;
    for (int i = 0; i < 4; ++i) {
        if (results[i] > max) {
            max = results[i];
            index = i;
        }
    }
    string winner;
    switch (index) {
        case RUSSIA:
            winner = "Russia";
            break;
        case CANADA:
            winner = "Canada";

```

```

        break;
    case FINLAND:
        winner = "Finlan";
        break;
    case FRANCE:
        winner = "France";
        break;
    case JAPAN:
        winner = "Japan";
        break;
    }

    return winner;
}

OlympicGames::~~OlympicGames() {
    for (int i = 0; i < this->count; ++i) {
        free(clients[i]);
    }
    for (int i = 0; i < this->peopleCount; ++i) {
        free(peoples[i]);
    }
}

string OlympicGames::sort(People **peoples, int len, string &sport) {
    int result;
    string line;
    for (int i = 0; i < len; ++i) {
        People *current = peoples[i];
        result = this->filterSportParam(current, sport);
        for (int j = i; j < len; ++j) {
            People *second = peoples[j];
            int secParam = this->filterSportParam(second, sport);
            if (secParam > result) {
                People *tmp = peoples[i];
                peoples[i] = peoples[j];
                peoples[j] = tmp;
            }
        }
    }

    for (int k = 0; k < len; ++k) {
        cout << k + 1 << " " << peoples[k]->getFio() << " " << peoples[k]-
>getCountry() << " " << this->filterSportParam(peoples[k], sport) << endl;
        line += to_string(k + 1) + " " + peoples[k]->getFio() + " " + peoples[k]-
>getCountry() + " " + to_string(this->filterSportParam(peoples[k], sport)) + "\n";
    }

    return line;
}

int OlympicGames::filterSportParam(People *current, string sport) {
    int result = 0;
    if (sport == "Skeleton") {
        result = current->getForce();
    } else if (sport == "Biatlon") {
        result = current->getAgility();
    } else if (sport == "Skating") {
        result = current->getLuck();
    }
}

```

```

    } else if (sport == "IceSkating") {
        result = current->getStamina();
    }
    return result;
}

```

Client.h

```

#ifndef CLIENT_H
#define CLIENT_H

#include <iostream>
#include <string>

using namespace std;

class Client {
    string choice;
public:
    Client(const string &choice);

public:
    Client();

public:
    const string &getChoice() const;

    void setChoice(const string &choice);

    void update (string winner);
};

#endif //CLIENT_H

```

Client.cpp

```

#include "Client.h"

void Client::update(string winner) {
    if (winner == this->choice) {
        cout << "Congratulations! The country you supported (" << this->choice <<
") won!"<< endl;
    } else {
        cout << "The country you supported (" << this->choice << ") lost! Country
that won is " << winner << "." << endl;
    }
}

const string &Client::getChoice() const {
    return choice;
}

void Client::setChoice(const string &choice) {
    Client::choice = choice;
}

```

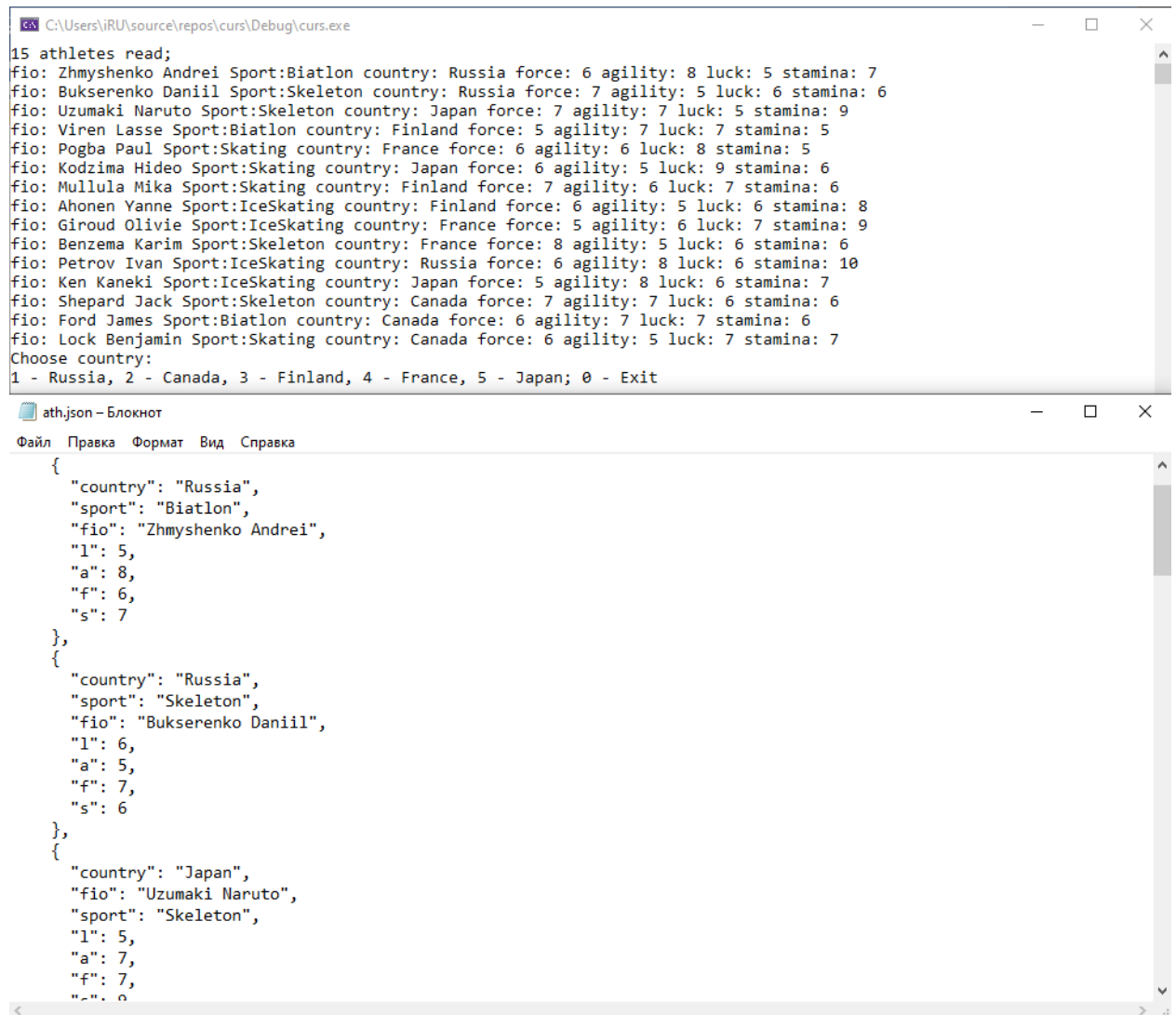
```
Client::Client() {}
```

```
Client::Client(const string &choice) : choice(choice) {}
```

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) Стивен Прата. Язык программирования C++. Учебное пособие 2012 год.
- 3) <https://habr.com/ru/post/210288/> - Шпаргалка по шаблонам проектирования
- 4) <https://refactoring.guru/ru/design-patterns/observer> - Наблюдатель
- 5) [https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B1%D0%BB%D1%8E%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_\(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F\)](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B1%D0%BB%D1%8E%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%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)) – Наблюдатель (шаблон проектирования)
- 6) <http://cpp-reference.ru/patterns/behavioral-patterns/observer/> - Паттерн (шаблон) проектирования Observer (наблюдатель)

Приложение 1



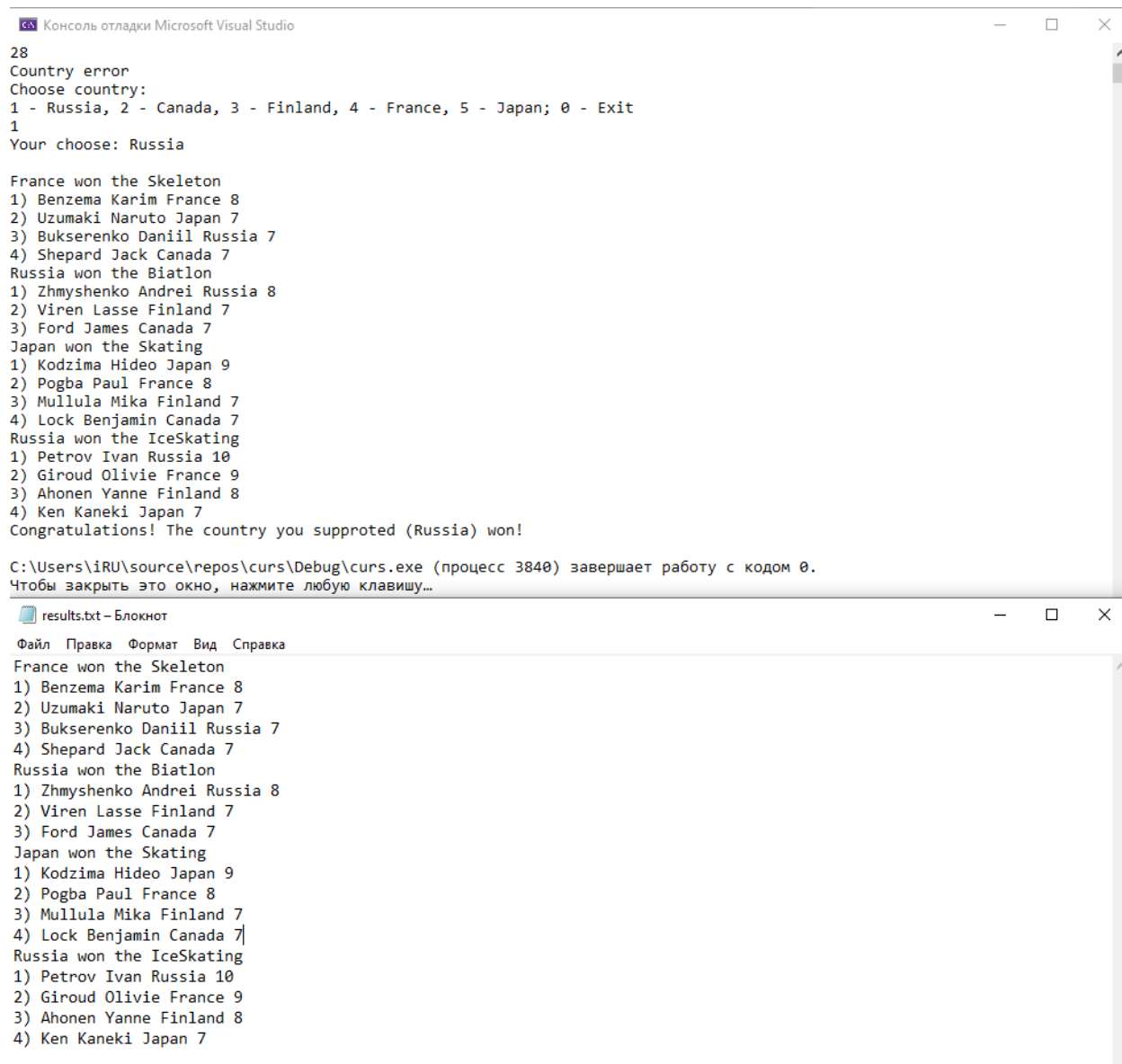
```
C:\Users\iRU\source\repos\curs\Debug\curs.exe
15 athletes read;
fio: Zhmyshenko Andrei Sport:Biatlon country: Russia force: 6 agility: 8 luck: 5 stamina: 7
fio: Bukserenko Daniil Sport:Skeleton country: Russia force: 7 agility: 5 luck: 6 stamina: 6
fio: Uzumaki Naruto Sport:Skeleton country: Japan force: 7 agility: 7 luck: 5 stamina: 9
fio: Viren Lasse Sport:Biatlon country: Finland force: 5 agility: 7 luck: 7 stamina: 5
fio: Pogba Paul Sport:Skating country: France force: 6 agility: 6 luck: 8 stamina: 5
fio: Kodzima Hideo Sport:Skating country: Japan force: 6 agility: 5 luck: 9 stamina: 6
fio: Mullula Mika Sport:Skating country: Finland force: 7 agility: 6 luck: 7 stamina: 6
fio: Ahonen Yanne Sport:IceSkating country: Finland force: 6 agility: 5 luck: 6 stamina: 8
fio: Giroud Olivier Sport:IceSkating country: France force: 5 agility: 6 luck: 7 stamina: 9
fio: Benzema Karim Sport:Skeleton country: France force: 8 agility: 5 luck: 6 stamina: 6
fio: Petrov Ivan Sport:IceSkating country: Russia force: 6 agility: 8 luck: 6 stamina: 10
fio: Ken Kaneki Sport:IceSkating country: Japan force: 5 agility: 8 luck: 6 stamina: 7
fio: Shepard Jack Sport:Skeleton country: Canada force: 7 agility: 7 luck: 6 stamina: 6
fio: Ford James Sport:Biatlon country: Canada force: 6 agility: 7 luck: 7 stamina: 6
fio: Lock Benjamin Sport:Skating country: Canada force: 6 agility: 5 luck: 7 stamina: 7
Choose country:
1 - Russia, 2 - Canada, 3 - Finland, 4 - France, 5 - Japan; 0 - Exit
```

```
ath.json - Блокнот
Файл  Правка  Формат  Вид  Справка
{
  "country": "Russia",
  "sport": "Biatlon",
  "fio": "Zhmyshenko Andrei",
  "l": 5,
  "a": 8,
  "f": 6,
  "s": 7
},
{
  "country": "Russia",
  "sport": "Skeleton",
  "fio": "Bukserenko Daniil",
  "l": 6,
  "a": 5,
  "f": 7,
  "s": 6
},
{
  "country": "Japan",
  "fio": "Uzumaki Naruto",
  "sport": "Skeleton",
  "l": 5,
  "a": 7,
  "f": 7,
  "s": 9
}
```

Рисунок 1 – Запуск программы, чтение информации из файла


```
C:\Users\iRU\source\repos\curs\Debug\curs.exe
15 athletes read;
fio: Zhmyshenko Andrei Sport:Biatlon country: Russia force: 6 agility: 8 luck: 5 stamina: 7
fio: Bukserenko Daniil Sport:Skeleton country: Russia force: 7 agility: 5 luck: 6 stamina: 6
fio: Uzumaki Naruto Sport:Skeleton country: Japan force: 7 agility: 7 luck: 5 stamina: 9
fio: Viren Lasse Sport:Biatlon country: Finland force: 5 agility: 7 luck: 7 stamina: 5
fio: Pogba Paul Sport:Skating country: France force: 6 agility: 6 luck: 8 stamina: 5
fio: Kodzima Hideo Sport:Skating country: Japan force: 6 agility: 5 luck: 9 stamina: 6
fio: Mullula Mika Sport:Skating country: Finland force: 7 agility: 6 luck: 7 stamina: 6
fio: Ahonen Yanne Sport:IceSkating country: Finland force: 6 agility: 5 luck: 6 stamina: 8
fio: Giroud Olivie Sport:IceSkating country: France force: 5 agility: 6 luck: 7 stamina: 9
fio: Benzema Karim Sport:Skeleton country: France force: 8 agility: 5 luck: 6 stamina: 6
fio: Petrov Ivan Sport:IceSkating country: Russia force: 6 agility: 8 luck: 6 stamina: 10
fio: Ken Kaneki Sport:IceSkating country: Japan force: 5 agility: 8 luck: 6 stamina: 7
fio: Shepard Jack Sport:Skeleton country: Canada force: 7 agility: 7 luck: 6 stamina: 6
fio: Ford James Sport:Biatlon country: Canada force: 6 agility: 7 luck: 7 stamina: 6
fio: Lock Benjamin Sport:Skating country: Canada force: 6 agility: 5 luck: 7 stamina: 7
Choose country:
1 - Russia, 2 - Canada, 3 - Finland, 4 - France, 5 - Japan; 0 - Exit
28
Country error
Choose country:
1 - Russia, 2 - Canada, 3 - Finland, 4 - France, 5 - Japan; 0 - Exit
```

Рисунок 2 – Ввод некорректного номера страны



The image shows two windows from a Windows operating system. The top window is the 'Консоль отладки Microsoft Visual Studio' (Microsoft Visual Studio Debug Console). It contains the following text:

```
28
Country error
Choose country:
1 - Russia, 2 - Canada, 3 - Finland, 4 - France, 5 - Japan; 0 - Exit
1
Your choose: Russia

France won the Skeleton
1) Benzema Karim France 8
2) Uzumaki Naruto Japan 7
3) Bukserenko Daniil Russia 7
4) Shepard Jack Canada 7
Russia won the Biatlon
1) Zhmyshenko Andrei Russia 8
2) Viren Lasse Finland 7
3) Ford James Canada 7
Japan won the Skating
1) Kodzima Hideo Japan 9
2) Pogba Paul France 8
3) Mullula Mika Finland 7
4) Lock Benjamin Canada 7
Russia won the IceSkating
1) Petrov Ivan Russia 10
2) Giroud Olivie France 9
3) Ahonen Yanne Finland 8
4) Ken Kaneki Japan 7
Congratulations! The country you supproted (Russia) won!
```

Below the console output, a message indicates the program has finished: 'C:\Users\iRU\source\repos\curs\Debug\curs.exe (процесс 3840) завершает работу с кодом 0. Чтобы закрыть это окно, нажмите любую клавишу...'

The bottom window is a Notepad application titled 'results.txt - Блокнот'. It contains the same text as the console window, but without the initial 'Country error' and 'Choose country:' prompts, and without the final congratulatory message. The text in the Notepad window is:

```
France won the Skeleton
1) Benzema Karim France 8
2) Uzumaki Naruto Japan 7
3) Bukserenko Daniil Russia 7
4) Shepard Jack Canada 7
Russia won the Biatlon
1) Zhmyshenko Andrei Russia 8
2) Viren Lasse Finland 7
3) Ford James Canada 7
Japan won the Skating
1) Kodzima Hideo Japan 9
2) Pogba Paul France 8
3) Mullula Mika Finland 7
4) Lock Benjamin Canada 7
Russia won the IceSkating
1) Petrov Ivan Russia 10
2) Giroud Olivie France 9
3) Ahonen Yanne Finland 8
4) Ken Kaneki Japan 7
```

Рисунок 3 – Вывод результатов и запись в файл

Консоль отладки Microsoft Visual Studio

```
fio: Ford James Sport:Biatlon country: Canada force: 6 agility: 7 luck: 7 stamina: 6
fio: Lock Benjamin Sport:Skating country: Canada force: 6 agility: 5 luck: 7 stamina: 7
Choose country:
1 - Russia, 2 - Canada, 3 - Finland, 4 - France, 5 - Japan; 0 - Exit
3
Your choose: Finland

France won the Skeleton
1) Benzema Karim France 8
2) Uzumaki Naruto Japan 7
3) Bukserenko Daniil Russia 7
4) Shepard Jack Canada 7
Russia won the Biatlon
1) Zhmyshenko Andrei Russia 8
2) Viren Lasse Finland 7
3) Ford James Canada 7
Japan won the Skating
1) Kodzima Hideo Japan 9
2) Pogba Paul France 8
3) Mullula Mika Finland 7
4) Lock Benjamin Canada 7
Russia won the IceSkating
1) Petrov Ivan Russia 10
2) Giroud Olivie France 9
3) Ahonen Yanne Finland 8
4) Ken Kaneki Japan 7
The country you supported (Finland) lost! Country that won is Russia.

C:\Users\iRU\source\repos\curs\Debug\curs.exe (процесс 19876) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 4 – Вывод результатов и сообщения о поражении