Технически университет – Варна

Факултет по изчислителна техника и автоматизация

Катедра: ,,Софтуерни и интернет технологии”

Специалност: ,,Софтуерни и интернет технологии”

Тема: ,,Турнир по тенис”

Студент: Сезгин Бейханов Шабанов

Факултетен номер: 20621521

Условие на проекта

Турнир по тенис Да се напише компютърна програма, реализираща информационна система за данни турнир по тенис, с играчи (номер в турнира, номер в световната ранглистата, име, фамилия, държава, текущи точки, брой спечелени купи(1ви места)). Максималният брой състезатели е 100.

Базова задача – Сложност ниска

A. Меню с избор на функциите в програмата (7 седмица)

B. Добавяне на състезатели в турнира. (7-8 седмица)

a. Добавяне на нов състезател.

b. Добавяне на списък със състезатели. Въвежда се цяло число n и след него n на брой състезатели

C. Извеждане на всички състезатели на екрана (8-9 седмица)

a. Извеждане на състезатели с най-малко спечелени купи

b. Извеждане на състезатели от определена държава Допълнение Първо – Сложност средна (+ Базова задача

D. Коригиране на данни за състезател. (9-10 седмица)

a. Ако липсват данни за състезателя, да се изведе подходящо съобщение.

b. Ако състезателя го има, се искат нови данни за корекция.

c. Да се предвиди, че ако се коригира номер, то не може да има двама играчи с еднакви номера.

Допълнение Второ – Сложност висока (+ Базова задача + Допълнение Първо)

E. Съставяне на двубои за турнира. (10 -11 седмица)

a. Съставяне на турнирни двубои по схема до дадени финали (примерно се въвеждат твърдо двубои за класиране за 8мина финали)

b. Добавяне на резултати за двубоите.

c. Ръчно съставяне на елиминационните двубои за 8мина,4върт,полуфаналите и финала, като се има предвид, кои са победителите.

Допълнение Трето – Сложност висока (+ Базова задача + Допълнение Първо + Допълнение Второ)

F. Одит на спортистите в подменю (11-12 седмица)

a. Извеждане на всички състезатели, сортирани по номер във Световната ранглиста

b. Извеждане на всички състезатели от дадена държава, сортирани в азбучен ред.

c. Извеждане на всички състезателите от дадена държава, сортирани по брой спечелени купи в намаляващ ред.

G. Данните в програмата да могат да се запазват във файл между две стартирания на програмата

Допълнение Четвърто – (за допълнителни точки)

H. Допълнителни условия:

a. Възможност за корекция на двубой в случай на преиграване. Допуска се еднократна корекция.

b. Автоматично съставяне на 8мина/4върт -> полу и финалния двубои, след като са въведени резултатите съответно за 16тина/8мина/4върт и полуфиналите (автоматично програмата съставя двубоите от 8мина финалите, на база резултатите, след което за 4върт и т.н. до финалния включително)

c. Данните записани в двоичен файл да могат да се записват в текстов файл и обратно.

Анализ на решението

```
#include <iostream>
#include<string>
#include<list>
#include <sstream>
#include<conio.h>
#include<fstream>

using namespace std;
```

Структура Player това е структурата от която ще си изградим нашите играчи в тенис турнира. Тя в себе си съдържа следните полета:

```
struct Player
{
    // Номер в турнира.
    int number_in_tournament = 0;
    // Номер в световната ранглиста.
    int number_in_world_rankings = 0;
    //Име.
    string first_name = "";
    //Фамилия.
    string second_name = "";
    //Държава.
    string country = "";
    // Текущи точки.
    int current_points = 0;
    //Брой спечелени първи места(купи).
    int first_places_cnt = 0;

public:
```

// Конструктор на структурата Player с всички параметри, по този начин може да си направим играч(променлива) от тип Player като извикаме конструктора и подадем удовлетворяващи типове данни за да формираме нашият играч.

```
    Player(int, int, string, string, string, int, int);
```

// Празен конструктор, във някои случаи искаме да си направим променлива на която по-нататък в кода ще присвоим данни но в момента на формиране просто ги нямаме по този може да имаме променлива без да сме длъжни да подаваме някакви параметри.

```
    Player();
```

// Функция get_num_in_tournament я използваме, когато искаме да разберем за даден играч какъв му е номерът в турнира.

```
    int get_num_in_tournament() {
        return number_in_tournament;
    }
```

// Функцията get_num_in_world_rankings я използваме, за да разберем за даден играч какъв му е номерът в световната ранглиста.

```
    int get_num_in_world_rankings() {
        return number_in_world_rankings;
    }
```

// Функцията get_first_name я използваме, когато искаме да разберем какво е името на даден играч .

```
    string get_first_name() {
```

```
        return first_name;
    }

    // Функцията get_second_name ни дава информация за това каква е фамилията на даден играч.
    string get_second_name() {
        return second_name;
    }

    // Функцията get_country я използваме, когато искаме да разберем от коя държава е дадения играч.
    string get_country() {
        return country;
    }

    // Функцията get_current_points я използваме, когато искаме да разберем колко текущи точки в турнира има даден играч.
    int get_current_points() {
        return current_points;
    }

    // Функцията get_first_places_cnt я използваме, за да получим информация за това колко са спечелените първи места(купи) на даден играч.



    int get_first_places_cnt() {
        return first_places_cnt;
    }



// Конструктор на структурата Player с всички параметри.
Player::Player(int num_in_tournament, int n_w_rankings, string f_name,
string s_name, string c, int c_points, int f_p_cnt)
{
    // Присвояване на стойности, които са били подадени от външния свят на нашият конструктор
    И така нашият играч ще бъде изграден със данни в зависимост от това какво е било подадено от потребителя.
    number_in_tournament = num_in_tournament;
    number_in_world_rankings = n_w_rankings;
    first_name = f_name;
    second_name = s_name;
    country = c;
    current_points = c_points;
    first_places_cnt = f_p_cnt;
}

// Празен конструктор на структурата Player.
Player::Player() {

}

// Структура Tournament това е структурата, която се явява тенис турнира и има само едно поле:
struct Tournament
{
    // Това е лист от плейъри, в който съхраняваме всеки един играч и информацията за него.
```

```
    list<Player> players;

    // Конструктора на структурата Tournament.
public:
    Tournament();

      // Функцията set_player приема един параметър и това е параметър от тип Player който параметър се явява нова играч, който потребителят е направил и го поставя в края на нашия лист с играчи.
    void set_player(Player p) {
// Поставянето на новия играч в турнира.
        players.push_back(p);
    }

    // Функцията set_n_players приема като параметър лист с играчи.
    void set_n_players(list<Player> player_list) {

        // Завъртаме while цикъла докато листът, който сме получили от външния свят не стане празен по този начин ще вземем всички играчи, които потребителя е подал и ще ги сложим в нашия турнир.
        while (!player_list.empty())
        {
            // Вземаме играч от началото на листа от външния свят.
            Player p = player_list.front();

            // Махаме играча от листа от който сме получили отвън.
            player_list.pop_front();

            //Поставяме играча в нашия лист players, който беше единственото поле на структурата Tournament, която отговаря за съхранението на играчите и техните данни.
            players.push_back(p);
        }
    }

    // Функция която при извикване връща като информация всички играчи които нашия лист "players" съдържа.
    list<Player> get_data() {
        return players;
    }

    //Функцията pop_player_of_tournament я използваме за да махнем един играч от "players".
    void pop_player_of_tournament() {
        players.pop_front();
    }

    // Функцията pop_player_of_index маха един определен играч като получава индекса на играча в турнира, който искаме да махнем.

    void pop_player_of_index(int m) {
        // Правим си нов list, който ще използваме, за да запишем в него всички играчи без този, който искаме да махнем.
        list<Player> new_player_list;
        int i = 1;
        // Въртим while цикъла докато нашите "players" не станат празни.

        while (!players.empty())
        {
            // Вземаме си играч от началото на players.
            Player p = players.front();
```

```
            // Махаме го от players.
            players.pop_front();

            // Ако i == m това значи, че сме намерили нашият играч и НЕ го
добавяме в  new_player_list, за да може  в new_player_list да са всички
играчи без този, който сме искали да махнем.

            if (i != m) {
                new_player_list.push_back(p);

            }
            i++;
        }
        // Присвояваме на нашите players всички играчи, които сме записали
в new_player_list.
        players = new_player_list;
    }

    // Функцията set_plyaer_index използваме, за да поставим играч на
определената позиция.
    Функцията приема два параметъра
    m - позицията на която искаме да поставим нашия играч
    new_p - играча, който ще поставим на тази позиция.

    void set_player_index(int m, Player new_p) {
        // Правим си нов лист с от Player.
        list <Player> new_player_list;

        // Тази променлива ще я използваме, за да проверяваме до кога да
въртим нашия for цикъл, който е по-надолу в кода.
   Избираме да е players.size() + 1 защото в момента имаме примерно N
играчи в players съответно players.size() == N;  понеже ще добавим още един
играч и то на определена позиция ни трябва players.size() + 1.

        int iteration_cnt = players.size() + 1;

        //Временна променлива от тип Player
        Player temp;

        for (int i = 1; i <= iteration_cnt; i++)
        {
            //Ако в new_player_list имаме N + 1 значи успешно сме добавили
нашия играч и break-ваме от цикъла.
            if (new_player_list.size() == iteration_cnt)
                break;


            Player p;

            // Проверка за това дали случайно не са свършили играчите в
нашия лист players.
            if (players.size() != 0) {

                // Вземаме си един играч.
                p = players.front();

                // Махаме го то players.
                players.pop_front();

            }
```

```
            // Ако i е различно от m значи не сме на позицията, която
искаме да добавим нашия играч.
            if (i != m) {

                // Проверяваме дали във временната променлива temp имаме
записано поне едно нещо ако нямаме това означава, че още  не сме присвоили
player на нея ще го видим малко по-надолу в кода защо го правим това с temp
променливата.
        Чрез проверката се застраховаме да нямаме играч в списъка с празни
данни.

                if (temp.get_country() != ""
                    || temp.get_current_points() != 0
                    || temp.get_first_name() != ""
                    || temp.get_first_places_cnt() != 0
                    || temp.get_num_in_tournament() != 0
                    || temp.get_num_in_world_rankings() != 0
                    || temp.get_second_name() != "") {

                    new_player_list.push_back(temp);
                }

       // Добавяме в new_player_list играча който взехме от players.

// Проверяваме дали p не е променлива която сме направили без да сме успели
да и зададем стойности понеже примерно в листа players е нямало повече
играчи.
                if (p .get_country() != ""
                    || p.get_current_points() != 0
                    || p.get_first_name() != ""
                    || p.get_first_places_cnt() != 0
                    || p.get_num_in_tournament() != 0
                    || p.get_num_in_world_rankings() != 0
                    || p.get_second_name() != "") {

                    new_player_list.push_back(p);
                }
            }
            else {
                // Понеже взехме играч и за да не го загубим при следващо
вземане ако има такова тук присвояваме на temp информацията за текущия
играч p.
                temp = p;
                // i == m значи сме на позицията която искаме да добавим
нашия играч и го добавяме.
                new_player_list.push_back(new_p);
            }
        }

        // Присвояваме на players всичките играчи, които добавихме по-горе
в new_player_list.
        players = new_player_list;
    }
// Функция, която използваме, за да махнем всички играчи от нашия турнир.
    void removeAll() {
// Завъртаме while цикъл докато в players имаме играчи
        while (!players.empty())
        {
// Махаме играчите един по един докато не приключи цикъла.
            players.pop_front();
        }
```

```
    }


// Празен конструктор на структурата Tournament.
Tournament::Tournament()
{
}

// Функцията addOnePlayer използваме за формиране на един играч тя приема параметър от тип int който се казва cnt понеже функцията addOnePlayer я извикваме и когато формираме N на брой играчи при формиране на първия играч имаме нужда от cin.ignore(); , но от там нататък нямаме нужда.

Player addOnePlayer(int cnt) {

// Инициализираме си този низ line с който ще четем данни за играча за променливите които са от тип int ще ги конвертираме и така ще ги присвоим, а за тези, които са от тип string просто ще ги присвоим.
   string line;

    // Данните от начало са "празни" като за целите числа това е 0, а за низовете това е празен низ това го правим с цел да изпълним условие от задачата, което гласи да направим проверка дали на даден играч не липсват данни ако има 0 за целите числа или празен низ за низовете ще потвърдим че в турнира имаме играч на когото му липсват данни.

    int num_in_tournament = 0, num_in_world_rankings = 0;
    string first_name = "", second_name = "", country = "";
    int current_points = 0, first_places_cnt = 0;

    if (cnt == 0)
        cin.ignore();


    cout << "Въведи нов играч: " << endl;
    // Въвеждане на информация за играча.

    cout << "Number of the player in tournament: ";
// Прочитаме един ред с данни за това какъв е номера на играча в турнира ако това не отговаря на число няма да се конвертира и стойността num_in_tournament ще остане 0 това важи и за другите променливи от тип int.
    getline(cin, line);

// Конвертираме данните.
    stringstream converter(line);
// Присвояваме ги на променливата.
    converter >> num_in_tournament;

    cout << "Number in world rankings: ";
// Прочитаме данни.
    getline(cin, line);
// Конвертираме ги
    stringstream converter1(line);
// Присвояваме
    converter1 >> num_in_world_rankings;
    cout << "First name: ";

// Прочитаме данни и направо ги присвояваме на променливата first_name, която отговаряше за името на играча понеже тя е от тип string.
    getline(cin, first_name);
```

```
    cout << "Last name: ";
// Прочитаме данни и направо ги присвояваме на променлива second_name
понеже тя е от тип string.
    getline(cin, second_name);

    cout << "Country: ";
// Прочитаме данни и направо ги присвояваме на променливата country понеже
тя е от тип string и няма нужда от конвертиране.
    getline(cin, country);

    cout << "Current points: ";
// Прочитаме един ред с данни.
    getline(cin, line);
// Конвертираме.
    stringstream converter2(line);
// Присвояваме на променливата current_points.
    converter2 >> current_points;

    cout << "First places count: ";
// Прочитаме данни
    getline(cin, line);
// Конвертираме ги.
    stringstream converter3(line);
// Присвояваме данните на променливата.
    converter3 >> first_places_cnt;
    cout << endl;

    // Формираме си нашия играч чрез конструктора като подаваме стойността
на всички променливи, които сме прочели.
    Player player(num_in_tournament, num_in_world_rankings, first_name,
second_name,
        country, current_points, first_places_cnt);

    // Връщаме играча.
    return player;
}

// Функция за формиране на N на брой играчи получава параметър int n, който
е броят на играчите които искаме да формираме.

list<Player> add_n_players(int n) {
// Правим си един list от тип Player в който ще добавим всеки един играч,
който сме направили с извикване на функцията addOnePlayer.

    list<Player> list_of_players;
// Тази променлива cnt я обяснихме във функцията addOnePlayer защо я
използваме.
    int cnt = 0;

    for (int i = 0; i < n; i++)
    {
        //Формираме един играч чрез функцията addOnePlayer, която е по-
горе.
        Player player = addOnePlayer(cnt);
        cnt++;
        //Добавяме играча в нашия лист.
        list_of_players.push_back(player);
    }
    // Връщаме листа на main функцията където с функцията от структурата
Tournament set_n_players ще добавим в нашия турнир n-на брой играчи.
    return list_of_players;
```

```
}

// Функция за принтиране на играчи получава като параметър list с играчи.
void printer_for_players(list<Player> players_list) {

    // В тази функция ще използваме итератор.
    list<Player>::iterator it;

// Нашият for цикъл ще започне от първия играч в турнира и ще продължи до последния като вътре в самия цикъл ще принтираме данните на текущия играч.
    for (it = players_list.begin(); it != players_list.end(); it++)
    {


        // Достъпваме на нашия текущ играч от листа номера в турнира.
        int number_in_tournament = it->get_num_in_tournament();

        // Достъпваме на нашия текущ играч от листа номера в световната ранглиста.
        int number_in_world_rankings = it->get_num_in_world_rankings();

        // Достъпваме на нашия текущ играч от листа името.
        string first_name = it->get_first_name();

        // Достъпваме на нашия текущ играч от листа фамилията.
        string second_name = it->get_second_name();

        // Достъпваме на нашия текущ играч от листа държавата.
        string country = it->get_country();

        // Достъпваме на нашия текущ играч от листа текущите точки.
        int current_points = it->get_current_points();

        // Достъпваме на нашия текущ играч от листа боря на спечелените първите места(купи).
        int first_places_cnt = it->get_first_places_cnt();


        // Принтираме информацията за текущия играч.

        cout << endl;
        cout << "Number in tournament: " << number_in_tournament << endl << "Number in the world rankings:"
            << number_in_world_rankings << endl;
        cout << "First name: " << first_name << endl << "Last name: " << second_name << endl;
        cout << "Country: " << country << endl << "Current points of player: " << current_points << endl;
        cout << "First places count: " << first_places_cnt << endl;

    }
}


// Функция за това да видим в нашия турнир кой е играча или ако са няколко играча с равен брой най-малко спечелени купи.
void compare_players_wons(Tournament tournament) {

    // Вземаме всички играчи, които участват в турнира.
    list <Player> player_list = tournament.get_data();
```

```
    // Временна променлива от тип Player
    Player temp;
    // Лист в който ще сложим играча или ако са няколко на брой играчите с
най-малко спечелени купи.
    list<Player> players_less_wons;
    temp.first_places_cnt = INT16_MAX;

    // Въртим while цикъл, докато има играчи в player_list.
    while (!player_list.empty())
    {
        // Вземаме един играч.
        Player p = player_list.front();
        // Махаме го от player.list.
        player_list.pop_front();
        // Проверяваме ако на текущия играч броя на купите е по-малък или
равен на тези на временната променлива
        if (p.get_first_places_cnt() <= temp.get_first_places_cnt()) {
 // Присвояваме на temp информацията за текущия играч, защото неговите купи
са по-малко и така в temp ще има информация за играча с най-малко спечелени
купи до тук.
            temp = p;

        }

    }

    // Слагаме в листа играча с най-малко спечелени купи в турнира.
    players_less_wons.push_back(temp);
    // Вземаме още един път всички играчи в турнира.
    list <Player> sec_pl_list = tournament.get_data();
    // Въртим while цикъла, докато в sec_pl_list има играчи.
    while (!sec_pl_list.empty())
    {
// Вземаме играч на който ще проверяваме дали броя спечелени купи не е
равен на броя на спечелените купи на нашия играч,който до тука има най-
малък брой спечелени купи. Ако са равни това означава, че в турнира има
няколко играча с равен
        брой спечелени най-малко купи.

        Player p = sec_pl_list.front();
        // Махаме играча.
        sec_pl_list.pop_front();
        // Проверяваме като при проверката гледаме името да е различно, за
да не запишем един и същи играч два пъти.
        if (p.get_first_places_cnt() == temp.get_first_places_cnt() &&
p.get_first_name() != temp.get_first_name())
            // Ако се окаже, че има такъв играч също го добавяме в листа
players_less_wons.
            players_less_wons.push_back(p);
    }

    // Принтираме играча или ако са няколко играчите.
    printer_for_players(players_less_wons);
}

// Функция за това да принтираме всички играчи от определена държава тя
приема като параметри нашия турнир с играчи и името на държавата от която
искаме да са всички играчи.
void target_country_printer(Tournament tournament, string target_country) {

    // Лист с всички играчи в турнира
```

```
    list<Player> players = tournament.get_data();

    // Лист в който ще съберем всички играчи, които са от определена държава.
    list<Player> target_country_players;

// Завъртаме while цикъл докато в листа players има играчи.
    while (!players.empty())
    {
// Вземаме си един играч от players.
        Player player_for_compare = players.front();
// Махаме го от players.
        players.pop_front();

        // Правим проверка ако на текущия играч държавата е същата като държавата от която искаме да принтираме всички играчи добавяме го в нашия лист със играчи само от тази държава.
        if (player_for_compare.get_country() == target_country)
            target_country_players.push_back(player_for_compare);
    }

    // Принтираме играчите, които са от тази определена държава.
    printer_for_players(target_country_players);
}


// Функция която ще валидира това да няма двама състезателя с еднакви номера в турнира.
bool validate_tournament_num(Tournament& tournament, int num) {
    // Вземаме всички състезатели в турнира.
    list<Player> players = tournament.get_data();

    // Въртим while цикъл докато players все още има състезатели.
    while (!players.empty())
    {
        // Вземаме състезателя.
        Player p = players.front();

        // Махаме го от players.
        players.pop_front();

        // num е променлива, която идва от функцията introduce_new_data_for_player и това всъщност е новият номер на играча и ако в турнира има играч чийто номер съвпада връщаме true и ще поискаме нов номер във функцията introduce_new_data_for_player.

        if (p.get_num_in_tournament() == num)
            return true;
    }
    // Ако няма играч с този номер който искаме да сложим на нашия играч връщаме false.
    return false;
}

// Функция в която въвеждаме новите данни на играча.
Player introduce_new_data_for_player(Tournament& t) {

    string line;
    int num_in_tournament = 0, num_in_world_rankings = 0;
    string first_name = "", second_name = "", country = "";
    int current_points = 0, first_places_cnt = 0;
```

```
    cout << "Въведи новите данни на играча: " << endl;
    cout << "Number of the player in tournament: ";

    // Прочитаме един низ който ще бъде нашият нов номер в турнира.
    cin.ignore();
    getline(cin, line);

    // Конвертираме го към int.
    stringstream converter(line);
    converter >> num_in_tournament;

    // Извикваме validate_tournament_num за да валидираме, че новият номер
на играча в турнира няма да съвпадне с този на някой друг играч. В случай,
че съвпадне завъртаме while цикъл докато не получим удовлетворяващ за нас
номер.

    while (validate_tournament_num(t, num_in_tournament))
    {
        cout << "Вече в турнира съществува играч с този номер моля въведете
ново число: ";
        getline(cin, line);
        stringstream converter(line);
        converter >> num_in_tournament;
    }

    cout << "Number in world rankings: ";

    // Прочитаме низ, който ще бъде новият ни номер в световната ранглиста.
    getline(cin, line);
    // Конвертираме го към int.
    stringstream converter1(line);
    converter1 >> num_in_world_rankings;

    cout << "First name: ";
    // Прочитаме ново име.
    getline(cin, first_name);
    // Прочитаме нова фамилия.
    cout << "Last name: ";
    getline(cin, second_name);
    // Прочитаме нова държава.
    cout << "Country: ";
    getline(cin, country);


    cout << "Current points: ";
    // Прочитаме низ, който ще представлява нашите нови текущи точки в
турнира.
    getline(cin, line);
    // Конвертираме го към int.
    stringstream converter2(line);
    converter2 >> current_points;

    cout << "First places count: ";
    // Прочитаме низ, който ще представлява броя на новите ни спечелени
първи места(купи).
    getline(cin, line);
    // Конвертираме към int.
    stringstream converter3(line);
    converter3 >> first_places_cnt;
    cout << endl;
    // Правим си нашия играч с новите му данни.
```

```
    Player player(num_in_tournament, num_in_world_rankings, first_name,
second_name,
        country, current_points, first_places_cnt);
    return player;
}

// Функция в която ще изберем на кой играч искаме да зададем нови данни.
void new_data_for_player(Tournament& tournament) {
    // Вземаме си нашите играчи.
    list<Player> players = tournament.get_data();
    // Въвеждаме число примерно 3 и така избираме да променим данните на
третия играч в турнира.
    cout << "Въведете числото: ";
    int m;
    cin >> m;

    // Проверяваме дали числото което сме въвели не е отрицателно или пък
не е по-голямо от players.size(), което всъщност ни показва и колко играчи
имаме в players.
    if (m < 0)
    {

        cout << "Числото което сте въвели е невалидно: ";
        cout << "Няма как да има състезател на отрицателна позиция"
    }
else if(m > players.size()) {
Cout << "Числото което сте въвели е невалидно" << endl;
Cout << "В турнира има само " << players.size() << " състезателя" << endl;
}
else {
    // Завъртаме for цикъл от 1(първия играч) включително до m, за да
вземем играча, който искаме да му променим данните.
    for (int i = 1; i <= m; i++)
    {
        // Вземаме текущ играч.
        Player player = players.front();
        // Махаме го от players.
        players.pop_front();
        // Ако i е равно на m значи, че сме открили нашия играч.
        if (i == m) {
/ Ще проверим дали на този играч не ми липсват данни.
           list<Player>list_with_one_player;
            Player player_for_check = player;
            bool checker = player_for_check.get_num_in_tournament() == 0
                || player_for_check.get_num_in_world_rankings() == 0
                || player_for_check.get_first_name() == ""
                || player_for_check.get_second_name() == ""
                || player_for_check.get_country() == ""
                || player_for_check.get_current_points() == 0
                || player_for_check.get_first_places_cnt() == 0;

           // Ако липсват принтираме съобщението.
           if (checker) {
                cout << "Има липсваща информация за този състезател
информацията за него в момента изглежда ето така!!!" << endl;
                list_with_one_player.push_back(player_for_check);
                cout << endl;
                printer_for_players(list_with_one_player);
                list_with_one_player.pop_front();

            }
            cout << "Въведете новите данни: " << endl;
```

```
            // Махаме го от турнира със старите му данни.
            tournament.pop_player_of_index(m);
            // Извикваме си функцията за въвеждане на нови данни.
            Player new_player = introduce_new_data_for_player(tournament);
            // Слагаме нашия играч с новите му данни на позицията от която
го махнахме със старите му данни.
            tournament.set_player_index(m, new_player);
        }

    }
}
}



// Функция за принтиране на всички играчи в турнира.
void printer_all_players(Tournament tournament) {
    // Правим си лист с всички играчи, които участват  в турнира.
    list<Player> players = tournament.get_data();
    // Извикваме нашата функция, която получава като параметър лист от
Player и принтира информацията за всеки един играч.
    printer_for_players(players);
}

// Функция за получаване на състезател от определена позиция.
Player get_P_at_this_index(list<Player> _list, int _i) {

    list<Player>::iterator it = _list.begin();
    _i = _i - 1;
    for (int i = 0; i < _i; i++) {
        ++it;
    }
    return *it;
}

// Функция за махане на играч от определена позиция тя приема като
параметри лист със всички играчи, индекса на играча който искаме да махнем
и един брояч, който след това колко махания имаме дотук. По-надолу в
документацията ще видим по-подробно обяснение за това за какво точно
използваме този брояч.
list<Player> remove_p_index(list<Player> players, int index, int
remove_cnt) {
    list<Player> new_player_list;

    int i = 1;
    // Това с remove_cnt го правим понеже ако искаме да махнем играч от
първа позиция и remove_cnt е 0 това означава, че досега не сме махали
играчи и всички са така
 както са си били подредени, нека махнем първия играч  сега искаме да
махнем играча на втора позиция, но ние вече имаме едно махане и листа е с
един играч по-малко тоест всички позиции са с една назад и втория играч
реално в момента е на позицията на първия играч(който го махнахме) третият
на позицията на втория и т.н.!
    index = index - remove_cnt;

    if (index == 0) {
        index = 1;
    }

    // Въртим while цикъла докато нашите "players" не станат празни.
```

```
    while (!players.empty())
    {
        // Вземаме си играч от началото на players.
        Player p = players.front();
        // Махаме го от players.
        players.pop_front();

        // Ако i == index това значи, че сме намерили нашият играч и НЕ го
добавяме в  new_player_list, за да може в new_player_list да са всички
играчи без този, който сме искали да махнем.

        if (i != index) {
            new_player_list.push_back(p);

        }
        i++;
    }
    // Присвояваме на нашите players всички играчи, които сме записали в
new_player_list.

    return new_player_list;
}

// Функция за ръчно съставяне на двубои тя приема като параметър нашия
турнир
list<Player> handmake_duels(list<Player> players, Tournament tournament) {

    // лист в който ще запишем всички победители от текущи двубои и ще се
класират на следващо ниво
    list <Player> winners;

    // лист в който ще сложим двама състезателя (двубой), за да можем да
изберем кой от тях ще продължи напред.
    list<Player> current_duel;
    int remove_cnt = 0;

    while (!players.empty()) {
// с тези две числа ще изберем състезателя по неговата позиция примерно 3
отговаря на трети състезател.
        int first_p;
        int second_p;

        cout << endl;
        cout << "Въведете две числа с които да изберете кои състезатели ще
са в двубой!" << endl;
        cout << "Въведете число с което да изберете първият състезател за
двубоя.";
        cin >> first_p;
        cout << "Въведете число с което да изберете вторият състезател за
двубоя.";
        cin >> second_p;

        while (first_p < 0 || first_p > players.size() || second_p < 0 ||
second_p > players.size())
        {
            // Ако се окаже, че числото което е въвел потребителя не ни
удовлетворява искаме ново число.
            cout << "Числото което сте въвели е невалидно въведете ново
число: ";
            cin >> first_p;
            cin >> second_p;
```

```
        }
        if (first_p == second_p) {
            cout << "Числата, които сте въвели избират един и същи
състезател моля въведете ново число с което да избере втория съзтезтел.";
            cin >> second_p;
        }
        // Вземаме първия играч за двубоя
        Player first_player = get_P_at_this_index(players, first_p -
remove_cnt);
        // Вземаме втория играч за двубоя
        Player second_player = get_P_at_this_index(players, second_p -
remove_cnt);

        // Махаме ги от players.
        players = remove_p_index(players, first_p, remove_cnt);
        remove_cnt++;
        players = remove_p_index(players, second_p, remove_cnt);
        remove_cnt++;

        //Слагаме ги в листа current_duel и от тук ще изберем кой от тях ще
продължи напред.

        current_duel.push_back(first_player);
        current_duel.push_back(second_player);
        cout << endl;
        cout << "Участниците в този двубой са: " << endl;
// Принтираме играчите, които участват в текущия двубой.
        printer_for_players(current_duel);

// Избираме победителя.
        cout << "Избери побетдителя от двубоя като въведеш число 1-отговаря
за първия 2-отговаря за втория състезател: " << endl;
        int num;
        cin >> num;

        // Вземаме победителя от текущия двубой
        Player winner_p = get_P_at_this_index(current_duel, num);
        //Слагаме победителя във winners.
        winners.push_back(winner_p);

// Махаме състезателите от current_duel, за да може на следваща итерация да
сложим други двама състезателя и така докато в players има състезатели.
        current_duel.pop_front();
        current_duel.pop_front();

    }

    // В players слагаме всички играчи(победителите), които имаме във
winners
    players = winners;
    // Махаме всички играчи от турнира.
    tournament.removeAll();
    // И слагаме само тези, които са победители.
    tournament.set_n_players(winners);
    return players;
}

// Функцията duels е функция за двубои приема като параметър турнира с
всички играчи в него
void duels(Tournament tournament) {
```

```
// Вземаме данните за играчите от турнира и ги присвояваме на листа players.
    list<Player> players = tournament.get_data();

// current_duel лист в който ще добавим двама играчи и така ще направим текущ двубой и накрая ще изберем кой от тях ще е победителя.
    list<Player> current_duel;

// winners лист в който ще добавим победителите от текущите двубои.
    list<Player> winners;

    //Проверяваме дали в players имаме 64 или повече играчи, което ще значи, че двубоите ще започнат от 64 финали
    if (players.size() > 64) {

        // Ако имаме примерно 100 играча това е 64 финали но имаме само 50 двубоя и ще завъртим for цикъла само до 50 защото след 50 нямаме играчи, които да участват в двубоите.
        int cnt = players.size() / 2;

        // С променливата duel_cnt ще броим колко двубоя сме направили ако са се провели по-малко от 64 двубоя и нямаме достатъчно състезатели въвеждаме нови такива в турнира.
        int duel_cnt = 0;

        // Програмата сама взема двама състезателя ние избираме само кой е победителя.
        for (int i = 1; i <= cnt; i++)
        {
// Проверяваме дали имаме поне двама състезателя в players, защото в противен случай няма как да направим двубой.
            if (players.size() >= 2) {


//Вземаме състезател от players и после  го махаме и от players и от турнира
                Player first_player = players.front();
                players.pop_front();
                tournament.pop_player_of_tournament();

// Вземаме още един състезател от players и пак го махаме и от players и от турнира
                Player second_player = players.front();
                players.pop_front();
                tournament.pop_player_of_tournament();

// Добавяме двамата състезатели в листа current_duel
                current_duel.push_back(first_player);
                current_duel.push_back(second_player);
                cout << endl;

//Принтираме кои са участниците в този двубой.
                cout << "Участниците в този двубой са: " << endl;
                printer_for_players(current_duel);

// Избираме победител.
                cout << "Избери побетдителя от двубоя като въведеш число 1-отговаря за първия 2-отговаря за втория състезател: " << endl;
                int num;
                cin >> num;
```

```
// Вземаме победителя от текущия двубой и го поставяме във winners това беше листът в който бяха победителите от текущите двубои.
                Player winner_p = get_P_at_this_index(current_duel, num);
                winners.push_back(winner_p);

// Махаме играчите от current_duel, за да може на следваща итерация да сме с нови състезатели.
                current_duel.pop_front();
                current_duel.pop_front();

// Увеличаваме брояча, който следи колко двубоя имаме проведени.
                duel_cnt++;
            }
        }

        // За останалите състезатели за които няма опоненти направо ги добавяме към победителите
Това го правим като завъртим while цикъл докато има играчи в players.
        while (!players.empty())
        {
// Вземаме играч и го махаме от players и от турнира и го слагаме във winners.
            Player p = players.front();
            players.pop_front();
            tournament.pop_player_of_tournament();
            winners.push_back(p);
        }

        // Ако са се провели по-малко от 64 двубоя и нямаме достатъчно състезатели въвеждаме нови такива в турнира.
        if (duel_cnt < 64) {

// Променливата remainder ще ни покаже колко състезателя не ни достигат за следващо ниво и ще трябва да добавим толкова, за да продължим.
            int remainder = 64 - duel_cnt;
            cout << "За съжаление няма достатъчно играчи в турнира, за да продължим към 32 финалите" << endl;
            cout << "За да продължим моля добавете " << remainder << " състезателя в турнира!" << endl;
            for (int i = 0; i < remainder; i++)
            {
// Формираме новия състезател чрез функцията addOnePlayer.
                Player p = addOnePlayer(i);
                winners.push_back(p);
            }

        }
        // Записваме във players всички победители от 64 финалите.
        players = winners;
    }

    // Начало на 32 финали.
    //Проверяваме дали в players имаме 32 или повече играчи, което ще значи, че двубоите ще започнат от 32 финали
    if (players.size() > 32 && players.size() <= 64) {

// cnt_f_32 променливата ще ни покаже колко двубоя може да проведем.
        int cnt_f_32 = players.size() / 2;

// d_cnt променлива, която ще следи за това колко двубоя имаме проведени.
        int d_cnt_32 = 0;
```

```
// winners_32 лист в който ще запишем всички победители от текущите двубои, които ще проведем.
        list<Player> winners_32;

        // Програмата сама взема двама състезателя ние избираме само кой е победителя.
        for (int i = 1; i <= cnt_f_32; i++)
        {
// Проверяваме в players дали имаме поне 2 състезателя, за да може да проведем двубой
            if (players.size() >= 2) {

// Вземаме първия играч и го махаме от players и от турнира.
                Player first_player = players.front();
                players.pop_front();
                tournament.pop_player_of_tournament();

// Вземаме втория играч и го махаме от players и от турнира.
                Player second_player = players.front();
                players.pop_front();
                tournament.pop_player_of_tournament();

// Слагаме първия и втория играч в листа current_duel.
                current_duel.push_back(first_player);
                current_duel.push_back(second_player);
                cout << endl;

// Принтираме кои са участниците в този двубой.
                cout << "Участниците в този двубой са: " << endl;
                printer_for_players(current_duel);

// Избираме победител
                cout << "Избери побетдителя от двубоя като въведеш число 1-отговаря за първия 2-отговаря за втория състезател: " << endl;
                int num;
                cin >> num;

// Вземаме победителя и го поставяме в листа winners_32.
                Player winner_p = get_P_at_this_index(current_duel, num);
                winners_32.push_back(winner_p);

// Махаме играчите от current_duel, за да може на следваща итерация да сме с нови състезатели.
                current_duel.pop_front();
                current_duel.pop_front();

// Увеличаваме брояча за дуелите d_cnt_32.
                d_cnt_32++;
            }
        }

        // За останалите състезатели за които няма опоненти направо ги добавяме към спечелилите.
        while (!players.empty())
        {
            Player p = players.front();
            players.pop_front();
            tournament.pop_player_of_tournament();
            winners_32.push_back(p);
        }
```

```
        // Ако са се провели по-малко от 32 двубоя и нямаме достатъчно
състезатели въвеждаме нови такива в турнира.
        if (d_cnt_32 < 32) {

// Променливата remainder ще ни покаже колко нови играча трябва да
формираме, за да продължим към следващото ниво.
            int remainder = 32 - d_cnt_32;

            cout << "За съжаление няма достатъчно играчи в турнира, за да
продължим към 16 финалите" << endl;
            cout << "За да продължим моля добавете " << remainder << "
състезателя в турнира!" << endl;
            for (int i = 0; i < remainder; i++)
            {
// Формираме новия състезател чрез функцията addOnePlayer.
                Player p = addOnePlayer(i);
                winners_32.push_back(p);
            }

        }

        // Слагаме в players всички състезатели, които са победители от 32
финалите така те продължават към 16 финалите.
        players = winners_32;
        tournament.set_n_players(winners_32);
    }
    // Края на 32 финали.

    // Начало на 16 финали.
    //Проверяваме дали в players имаме 16 или повече играчи, което ще
значи, че двубоите ще започнат от 16 финали
    if (players.size() > 16 && players.size() <= 32) {

// Променливата cnt_f_16 ще ни покаже колко двубоя можем да проведем.
        int cnt_f_16 = players.size() / 2;

// Променливата d_cnt_16 ще следи за това колко проведени двубоя имаме.
        int d_cnt_16 = 0;

//winners_16 лист в който ще добавим победителите от текущите двубои.
        list<Player> winners_16;

        // Програмата сама взема двама състезателя ние избираме само кой е
победителя.
        for (int i = 1; i <= cnt_f_16; i++)
        {

// Проверяваме дали във players имаме поне двама играчи ако е така това
значи, че може да проведем двубой.
            if (players.size() >= 2) {

// Вземаме първия играч от players и го махаме от players и от турнира.
                Player first_player = players.front();
                players.pop_front();
                tournament.pop_player_of_tournament();

// Вземаме втория играч и го махаме от players и от турнира.
                Player second_player = players.front();
                players.pop_front();
                tournament.pop_player_of_tournament();
```

```
// Слагаме първия и втория играч в current_duel това беше листът, който
отговаряше за текущите двубои.
                current_duel.push_back(first_player);
                current_duel.push_back(second_player);
                cout << endl;

// Принтираме участниците в текущия двубой.
                cout << "Участниците в този двубой са: " << endl;
                printer_for_players(current_duel);
//Избираме победител.
                cout << "Избери побетдителя от двубоя като въведеш число 1-
отговаря за първия 2-отговаря за втория състезател: " << endl;
                int num;
                cin >> num;

// Вземаме си победителя и го добавяме в листа winners_16.
                Player winner_p = get_P_at_this_index(current_duel, num);
                winners_16.push_back(winner_p);

// Махаме и двамата играчи от current_duel, за да може на следваща итерация
да сме с нови състезатели.
                current_duel.pop_front();
                current_duel.pop_front();

// Увеличаваме брояча, който следеше колко поведени двубоя имаме.
                d_cnt_16++;
            }
        }

        // За останалите състезатели за които няма опоненти направо ги
добавяме към спечелилите.
        while (!players.empty())
        {
            Player p = players.front();
            players.pop_front();
            tournament.pop_player_of_tournament();
            winners_16.push_back(p);
        }

        // Ако са се провели по-малко от 16 двубоя и нямаме достатъчно
състезатели въвеждаме нови такива в турнира.
        if (d_cnt_16 < 16) {
// Променливата remainder ни показва колко играчи не ни достигат, за да
продължим към следващото ниво съответно трябва да добавим толкова на брой
нови играчи.
            int remainder = 16 - d_cnt_16;
            cout << "За съжаление няма достатъчно играчи в турнира, за да
продължим към 16 финалите" << endl;
            cout << "За да продължим моля добавете " << remainder << "
състезателя в турнира!" << endl;

            for (int i = 0; i < remainder; i++)
            {
// Добавяме нов играч с функцията addOnePlayer.
                Player p = addOnePlayer(i);
                winners_16.push_back(p);
            }

        }
```

```
        // Слагаме във players всички победители от 16 финали и те
продължат към осмина финалите.
        players = winners_16;
        tournament.set_n_players(winners_16);
    }

    // Осмина финали извикваме функцията handmake_duels и така ще съставим
всеки един двубой от осмина финалите и ще изберем победителите, които ще
продължат към четвъртина финалите.

    if (players.size() > 8 && players.size() <= 16) {
// Извикваме функцията handmake_duels
        players = handmake_duels(players, tournament);
    }

    // Четвъртина финали извикваме функцията handmake_duels и така ще
съставим всеки един двубой от четвъртина финалите и ще изберем
победителите, които ще продължат към полуфиналите.

    if (players.size() > 4 && players.size() <= 8) {
        players = handmake_duels(players, tournament);
    }

    // Полуфинали извикваме функцията handmake_duels и така ще съставим
всеки един двубой от полуфиналите и ще изберем победителите, които ще са на
финал.
    if (players.size() > 2 && players.size() <= 4) {
        players = handmake_duels(players, tournament);
    }


    // Финал
    if (players.size() == 2) {
        cout << "Финалистите в турнира са: " << endl;
// Принтираме кои са участниците във финала.
        printer_for_players(players);

// first_p отговаря за позицията на първия играч.
        int first_p = 1;
// second_p отговаря за позицията на втория играч.
        int second_p = 2;
        int remove_cnt = 0;
        cout << endl;

// Вземаме първия играч и го махаме от players и увеличаваме remove_cnt за
който обяснихме по-горе за какво го ползваме.
        Player first_player = get_P_at_this_index(players, first_p -
remove_cnt);
        players = remove_p_index(players, first_p, remove_cnt);
        remove_cnt++;

// Вземаме втория играч и го махаме от players и увеличаваме remove_cnt.
        Player second_player = get_P_at_this_index(players, second_p -
remove_cnt);
        players = remove_p_index(players, second_p, remove_cnt);
        remove_cnt++;


// Слагаме във current_duel първия и втория играч.
        current_duel.push_back(first_player);
        current_duel.push_back(second_player);
```

```
        cout << endl;
// Избираме кой е победителя на финала.
        cout << "Избери побетдителя от двубоя като въведеш число 1-отговаря
за първия 2-отговаря за втория състезател: " << endl;
        int num;
        cin >> num;

// Вземаме победителя от финала и го принтираме.
        Player winner_p = get_P_at_this_index(current_duel, num);
        list<Player> winner_in_tournament;
        winner_in_tournament.push_back(winner_p);
        cout << "Победител в турнира е: " << endl;

//Принтираме победителя от двубоя.
        printer_for_players(winner_in_tournament);
// Махаме играчите от current_duel.
        current_duel.pop_front();
        current_duel.pop_front();
    }



}
// В класа player_world_rankings_compare сравняваме играчите по номера им в
световната ранглиста.
class player_world_rankings_compare {
public:


    bool operator()(Player& a,
        Player& b)
    {

        // Вземаме на първия играч номера в световната ранглиста и го
сравняваме с номера на втория играч ако е по-малък връщаме true.

        if (a.get_num_in_world_rankings() < b.get_num_in_world_rankings())
{
            return true;
        }
        // В противен случай връщаме резултат false.
        return false;
    }
};

// Функцията players_sorted_world_rankings_num приема като параметър
турнира с играчите и в тази функция ние ще сравним играчите по техния номер
в световната ранглиста.
void players_sorted_world_rankings_num(Tournament tournament) {

    // Правим си лист като вземаме информацията за турнира от структурата
Tournament чрез функцията
    get_data() която връщаше списък с всички играчи.

    list<Player> players_list = tournament.get_data();

    // Правим си обект от класа player_wons_compare!
    player_world_rankings_compare cmp;

    //Сортираме нашия лист във възходящ ред чрез обекта cmp.
    players_list.sort(cmp);
```

```
    // Принтираме нашите играчи във възходящ ред според това на кого номера
е най-малък в световната ранглиста.
    printer_for_players(players_list);
}

class alphabeticly_sort {
public:


    bool operator()(Player& a,
        Player& b)
    {

// Вземаме името на първия играч и го сравняваме с името на втория ако
буквата с която започва името на първия е преди буквата с която започва
името на втория играч връщаме true.

 if (a.get_first_name() < b.get_first_name()) {
            return true;
        }
        // В противен случай връщаме резултат false.
        return false;
    }
};

// Във функцията target_country_sorted ще сортираме играчите от определена
държава по имената им подредени в азбучен ред.
void target_country_sorted(Tournament tournament) {

// На променливата target_country ще присвоим стойност за държавата от
която искаме да са играчите.
    string target_country;
    cout << "Моля въведете държава: ";
    cin >> target_country;

    // Лист с всички играчи в турнира
    list<Player> players = tournament.get_data();

    // Лист в който ще съберем всички играчи, които са от определена
държава.
    list<Player> target_country_players;

// Завъртаме while цикъла докато в players има играчи.
    while (!players.empty())
    {
// Вземаме текущ играч и го махаме от players.
        Player player_for_compare = players.front();
        players.pop_front();

        // Правим проверка ако на текущия играч държавата е същата като
държавата от която искаме да принтираме всички играчи добавяме го в нашия
лист с играчи само от тази държава.
        if (player_for_compare.get_country() == target_country)
            target_country_players.push_back(player_for_compare);
    }
// Правим си обект от класа alphabeticly_sort cmp  с който ще сортираме
нашия лист по азбучен ред според имената на играчите.
```

```
   alphabeticly_sort cmp;
// Сортираме
    target_country_players.sort(cmp);
// Принтираме сортирания лист.
    printer_for_players(target_country_players);
}

//В класа t_country_f_places ще сортираме играчите по брой спечелени купи в намаляващ ред.
class t_country_f_places {
public:


    bool operator()(Player& a,
        Player& b)
    {

        // Вземаме на първия играч броя спечелени първи места и сравняваме с тези на втория ако са по-малко връщаме резултат true.
        if (a.get_first_places_cnt() > b.get_first_places_cnt()) {
            return true;
        }
        // В противен случай връщаме резултат false.
        return false;
    }
};

// Във функцията target_country_first_places_sorter ще сортираме в намаляващ ред всички играчи от определена държава според това кой има по-малко спечелени първи места(купи)
void target_country_first_places_sorter(Tournament tournament) {

// Променлива на която ще присвоим стойност за това от коя държава искаме да са нашите играчи.
    string target_country;
    cout << "Моля въведете държава: ";
    cin >> target_country;

    // Лист с всички играчи в турнира
    list<Player> players = tournament.get_data();

    // Лист в който ще съберем всички играчи, които са от определена държава.
    list<Player> target_country_players;

// Завъртаме while цикъла докато в players има играчи.
    while (!players.empty())
    {
// Вземаме текущ играч и го махаме от players.
        Player player_for_compare = players.front();
        players.pop_front();

        // Правим проверка ако на текущия играч държавата е същата като държавата от която искаме да принтираме всички играчи добавяме го в нашия лист с играчи само от тази държава.
        if (player_for_compare.get_country() == target_country)
            target_country_players.push_back(player_for_compare);
    }
// Правим си обект от класа t_country_f_places с който ще сортираме нашия лист с играчи от определена държава в намаляващ ред според това кой има по-малко спечелени първи места(купи).
```

```
    t_country_f_places cmp;
// Сортираме
    target_country_players.sort(cmp);
// Принтираме сортирания лист.
    printer_for_players(target_country_players);

}

// Функция за запазване на  данните за играчите във файл, които са били в програмата.
void save_data(Tournament tournament) {
// лист със всички играчи в турнира.
    list<Player> players = tournament.get_data();


// с променливата myfile ще записваме в example.txt данните за играчите, които са в турнира.
    ofstream myfile;
// Отваряме example.txt
    myfile.open("example.txt");
// Променлива която ни показва колко играча имаме записани
    int players_cnt = players.size();
    myfile << players_cnt << endl;
    // Записваме данните на текущ играч във файла example.txt и това действие продължава докато в players има състезатели.
    while (!players.empty())
    {
        Player p = players.front();
        players.pop_front();
        myfile << p.get_num_in_tournament() << endl;
        myfile << p.get_num_in_world_rankings() << endl;
        myfile << p.get_first_name() << endl;
        myfile << p.get_second_name() << endl;
        myfile << p.get_country() << endl;
        myfile << p.get_current_points() << endl;
        myfile << p.get_first_places_cnt() << endl;
    }
    myfile.close();
}

// Функция за четене на данните, които са били запазени в example.txt така примерно ако сме имали трима играчи преди да спрем програмата и сме ги запазили с функцията save_data още при стартиране на програмата може да извикаме read_data_from_file и пак в турнира ще са същите трима играчи от предния път.
Tournament read_data_from_file() {
// Променливи на които ще присвоим стойности за даден играч.
    int t_num;
    int w_r_num;
    string f_name;
    string l_name;
    string country;
    int c_points;
    int f_places_cnt;
    Tournament tournament;
    ifstream infile;
    infile.open("example.txt");

// Променлива която ще ни покаже колко играча са били записани и ще завъртим for цикъл за да можем да вземем данните за всеки един играч, който е бил записан в example.txt
```

```
    int iter_cnt;
    infile >> iter_cnt;
// Присвояване на стойности за дадения играч
    for (int i = 0; i < iter_cnt; i++)
    {
        infile >> t_num;
        infile >> w_r_num;
        infile >> f_name;
        infile >> l_name;
        infile >> country;
        infile >> c_points;
        infile >> f_places_cnt;
// Правим си нашия играч със стойностите които са били записани в
example.txt
        Player player(t_num, w_r_num, f_name, l_name, country, c_points,
f_places_cnt);

// Слагаме играча в турнира.
        tournament.set_player(player);
    }

    return tournament;

}

// Меню
void menu() {
    setlocale(LC_ALL, "Bulgarian");
    cout << "                                                    Меню" <<
endl;
    cout << endl;
    cout << "Въведи едно число, за да избереш команда от менюто: " << endl;
    cout << endl;
    cout << "1: за формиране на нов играч в турнира" << endl;
    cout << endl;
    cout << "2: за формиране на n на брой играчи: " << endl;
    cout << endl;
    cout << "3: за извеждане на всички състезатели на екрана" << endl;
    cout << endl;
    cout << "4: за принтиране на играчите с най-малко спечелени купи" <<
endl;
    cout << endl;
    cout << "5: за принтиране на играчите от определена държава" << endl;
    cout << endl;
    cout << "6: коригиране на данни на състезател ако той участва в
турнира" << endl;
    cout << endl;
    cout << "7: за съставяне на турнирни двубои по схема до дадени финали"
<< endl;
    cout << endl;
    cout << "8: за извеждане на състезатели сортирани по номер в световната
ранглиста" << endl;
    cout << endl;
    cout << "9: за извеждане на състезатели от определена държава сортирани
по азбучен ред" << endl;
    cout << endl;
    cout << "10: за извеждане на състезатели от определена държава
сортирани по брой спечелени купи в намаляващ ред";
    cout << endl;
    cout << endl;
    cout << "11: за запазване на данните в програмата във файл" << endl;
```

```
    cout << endl;
    cout << "12: за прочитане на запазените данни от предишно стартиране на
програмата" << endl;
    cout << endl;
}


int main()
{
    // Променлива от тип Tournament в който ще си съхраняваме нашите играчи.
    Tournament tournament;

    setlocale(LC_ALL, "Bulgarian");

    while (true)
    {
        // Меню.
        menu();

        list<Player> players;
        Player player;
        int n;
        string str;

        // Опции в нашето меню.

       // Локална променлива command от тип int на която после ще присвоим
информацията от низа str по-долу, за да можем да работим със swtich.
        int command;
        // Прочитаме нашия низ.
        cout << "Въведи число, за да избереш команда от менюто или end за
край на програмата! ";
        cin >> str;



        // Проверяваме дали низа не е "end" ако е енд нашата програма приключва.
        if (str == ("end"))
            break;

        // Ако не е end конвертираме го към int и отиваме в кейса, който сме си
избрали от менюто и извършваме операциите.
        stringstream converter(str);
        converter >> command;

        cout << endl;
        // От тук ще изберем каква операция искаме да направим дали да формираме
един играч или N-на брой такива и т.н.
        switch (command)
        {
        case 1:
            player = addOnePlayer(0);

            tournament.set_player(player);
            break;

        case 2:

            cout << "n = ";
            cin >> n;
            players = add_n_players(n);
```

```
            tournament.set_n_players(players);
            break;

        case 3:
            printer_all_players(tournament);
            break;
        case 4:

            compare_players_wons(tournament);
            break;

        case 5:
            cout << "Въведи държава: ";

            cin >> str;

            target_country_printer(tournament, str);
            break;

        case 6:
            cout << "Искате да промените данните на определен играч добре "
<< endl;
            cout << "Трябва да въведете число, за да намерите играча  чийто
данни искате да промените ";
            cout << "Например 1 - за първия играч 2 - за втория и n - за n-
тия играч" << endl;
            new_data_for_player(tournament);
            break;

        case 7:
            duels(tournament);
            break;

        case 8:
            players_sorted_world_rankings_num(tournament);
            break;

        case 9:
            target_country_sorted(tournament);
            break;

        case 10:
            target_country_first_places_sorter(tournament);
            break;

        case 11:
            save_data(tournament);
            break;

        case 12:
            tournament = read_data_from_file();
            break;


        default:
            break;
        }



    }    return 0;}
```

Упътване за употреба

Когато стартираме програмата ще видим меню от което като въведем число ще можем да изберем опция за дадена операция или при въвеждане на end да прекратим работата на програмата.

Опция 1 - добавяне на нов състезател в турнира: Когато избере тази опция потребителят ще трябва да въведе данните на играча, който иска да формира те са: номер в турнира, номер в световната ранглиста, име, фамилия, държава, текущи точки и брой спечелени първи места(купи).

Опция 2 - добавяне на N на брой играчи: Когато избере тази опция потребителят ще трябва да въведе число за N, което ще покаже колко играча иска да формира. След като въведе числото програмата ще изиска от него да въведе данните за играчите.

Опция 3 - за извеждане на всички състезатели в турнира на екрана: Когато избере тази опция ще се изведат данните за всеки един състезател в турнира.

Опция 4 - за принтиране на играч или ако са няколко играчи с най-малко спечелени купи: При избор на тази опция ще се изведе информация за това кой е играча с най-малко спечелени купи. Ако има няколко играча, които има равен брой най-малко спечелени купи ще се изведе информацията за тях.

Опция 5 – за принтиране на играчи от определена държава: При избор на тази опция програмата ще изиска от потребителя да въведе държава и ще изведе на екрана играчите чийто държава съвпада с въведената.

Опция 6 - за коригиране на данни за състезател ако той участва в турнира: Когато се избере тази опция програмата ще изиска да се въведе число, което да определи на кой състезател искаме да променим данните примерно 2 – избира вторият състезател. Ако се въведи число което е невалидно примерно 5 а ние имаме само 4 състезателя програмата ще изведе съобщение, че числото е невалидно. В другия случай ако се въведе отрицателно число отново ще изведе съобщение за невалидно число. При въвеждане на число, което удовлетворява програмата ще се провери дали на този състезател не му липсват данни и след това ще се изискат от потребителя новите данни на играча. Като е взето под внимание, че играчи с еднакви номера в турнира не може да има!

Опция 7 – за съставяне на турнирни двубои: При избор на тази опция програмата ще провери колко състезателя има в нашия турнир ако са повече от 64 двубоите ще започнат от 64 финали. Ако са повече 32, но по-малко или равно на 64 турнирите ще започнат от 32 финали. Ако са повече от 16, но по-малко или равно на 32, двубоите ще започнат от 16 финали. Като в тези двубои програмата сама взема двама състезателя, а потребителят ще трябва да избере само победител. Програмата ще изиска от него да избере победител в текущия двубой като въведе 1 за първия състезател в двубоя или 2 за втория по този начин ще се избере победителя в текущия двубой. Когато се стигне до осмина, четвъртина и полуфиналите всеки двубой ще се прави ръчно. Като потребителят въвежда число примерно 3 така взема третия състезател подред после

въвежда 5 и взема петия състезател подред и това е текущ двубой. Ако някое от числата е невалидно ще бъде поискано ново число за избор на състезател. Пак ще трябва да избере победител по същия начин, който беше описан по-горе. Когато се стигне до финала ще се изведат на екрана финалистите и потребителят ще трябва да избере победител в турнира.

Опция 8 – за извеждане на състезатели сортирани по номер в световната ранглиста: Когато се избере тази опция от менюто на екрана ще се изведат състезателите сортирани по номер в световната ранглиста във възходящ ред.

Опция 9 – за извеждане на състезатели от определена държава сортирани по азбучен ред: При избор на тази опция програмата ще изиска от потребителя да въведе държава и ще изведе на екрана играчите сортирани по азбучен ред по име.

Опция 10 – за извеждане на всички състезатели от определена държава сортирани по брой спечелени места в намаляващ ред: Когато бъде избрана тази опция потребителят ще трябва да въведе име на държава и всички играчи, които са в турнира и държава им съвпада с въведената ще бъдат сортирани по брой спечелени първи места(купи) в намаляващ ред и изведени на екрана.

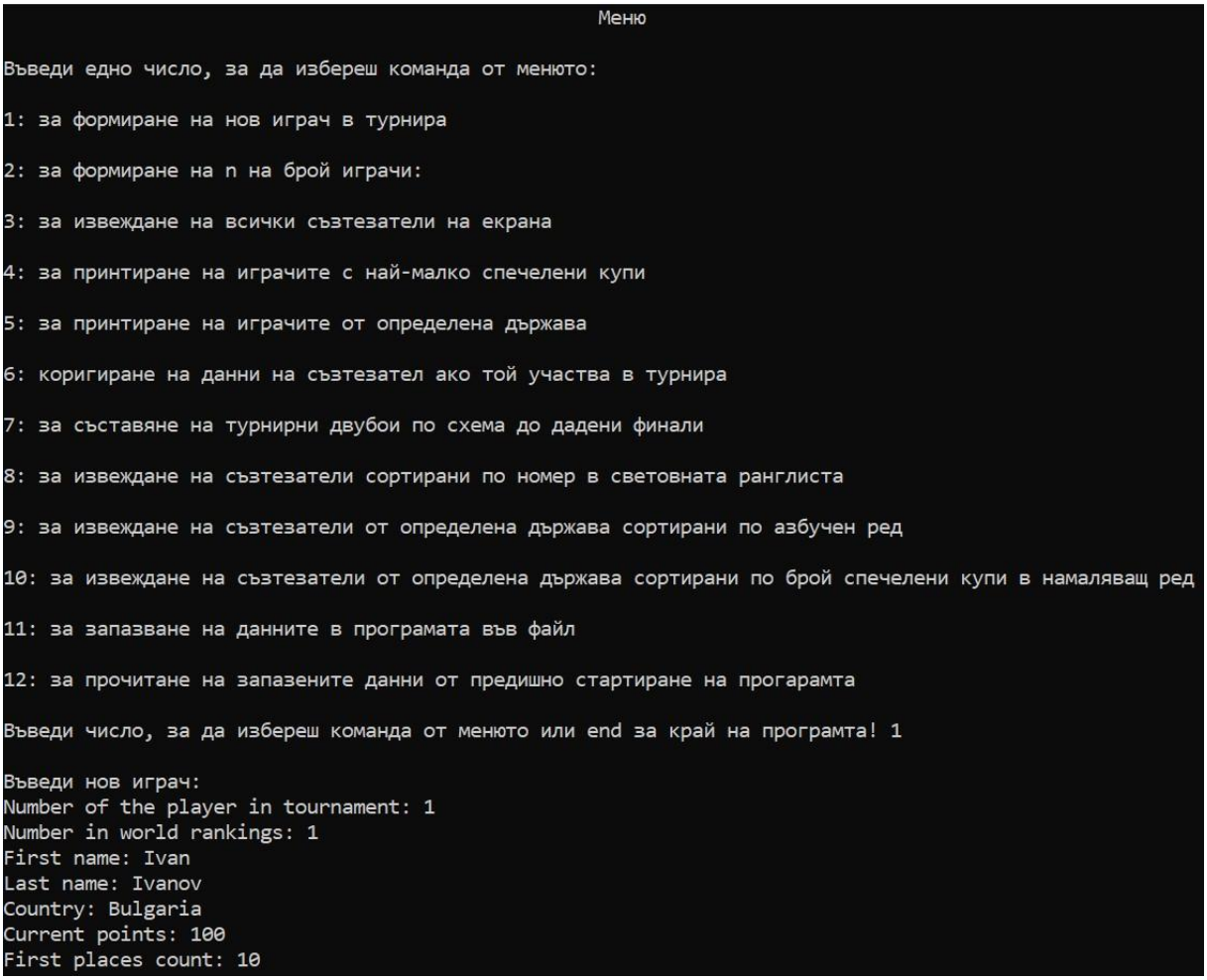
Опция 11 – за запазване на данните в програмата във файл: При избор на тази опция потребителят ще има възможността да запише данните за всеки един състезател във текстов файл. Като записването става автоматично програмата записва данните за всеки един играч във текстовия файл.

Опция 12 – за прочитане на запазените данни от предишно стартиране на програмата: Когато се избере тази опция от менюто ако е била избрана опция 12 и в текстовия файл има записани данни за състезателите потребителят ще може да ги прочете и в турнира вече ще са същите състезатели, които са били в текстовия файл.

Примерно действие на програмата

Снимки на изгледа с примерни входни данни:

Примерни входни данни с избор 1 от менюто, който е за формиране на един състезател



Примерни входни данни с избор 2 от менюто, който е за формиране на N на брой играчи.

```
Въведи число, за да избереш команда от менюто или end за край на програмта! 2

n = 2
Въведи нов играч:
Number of the player in tournament: 1
Number in world rankings: 1
First name: Petar
Last name: Petrov
Country: Bulgaria
Current points: 100
First places count: 5

Въведи нов играч:
Number of the player in tournament: 2
Number in world rankings: 5
First name: Joe
Last name: Joe
Country: England
Current points: 80
First places count: 1
```

Снимки на изгледа с примерни изходни данни:

Примерни изходни данни с избор 3 от менюто, който е за извеждане на всички състезатели в турнира

```
                                        Меню

Въведи едно число, за да избереш команда от менюто:

1: за формиране на нов играч в турнира

2: за формиране на n на брой играчи:

3: за извеждане на всички състезатели на екрана

4: за принтиране на играчите с най-малко спечелени купи

5: за принтиране на играчите от определена държава

6: коригиране на данни на състезател ако той участва в турнира

7: за съставяне на турнирни двубои по схема до дадени финали

8: за извеждане на състезатели сортирани по номер в световната ранглиста

9: за извеждане на състезатели от определена държава сортирани по азбучен ред

10: за извеждане на състезатели от определена държава сортирани по брой спечелени купи в намаляващ ред

11: за запазване на данните в програмата във файл

12: за прочитане на запазените данни от предишно стартиране на прогарамта

Въведи число, за да избереш команда от менюто или end за край на програмта! 3


Number in tournament: 4
Number in the world rankings:15
First name: Atanas
Last name: Ivanov
Country: Bulgaria
Current points of player: 98
First places count: 4
```

Изходни данни с избор 7 от менюто, който е за съставяне на турнирни двубои и това са финалистите от двубоите: