Технически университет – Варна

# Факултет: Факултет по изчислителна техника и автоматизация (ФИТА)

Катедра: Софтуерни и интернет технологии (СИТ)

Специалност: Софтуерни и интернет технологии (СИТ)

Тема на проекта:

**Турнир по тенис**

Изготвил: Явор Константинов Стоянов  
Факултетен номер: 20621620

**Задание на проекта:**

**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.**

**A. Меню с избор на функциите в програмата**

**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**

**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.**

**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.**

**E. Съставяне на двубои за турнира.**

**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.**

**F. Одит на спортистите в подменю**

**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**

**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 код**

**Реализация на проекта по условие**

**А. Меню с избор на функциите в програмата**

Функция int menu(), използва се за визуализация в компилатора на възможните опции в програмата „Турнир по тенис“.

int menu()

{

short int c; //choice

cout << "\n \t### Tennis tournament ## Menu ### ";

cout << "\n ";

cout << "\n === Adding competitors in the tournament: === ";

cout << "\n 1. Adding a competitor ";

cout << "\n 2. Adding a list of competitors ";

cout << "\n === Showing competitors: === ";

cout << "\n 3. Show all competitors: ";

cout << "\n 4. Competitors with less won trophies (1st places) ";

cout << "\n 5. Competitors from specific country ";

cout << "\n === Edit === ";

cout << "\n 6. Edit competitor data by number in the tournament";

cout << "\n === Audit - sub menu === ";

cout << "\n 7. Audit";

cout << "\n === Duel === ";

cout << "\n 8. Duel Demo Test";

cout << "\n 9. Duel Test";

cout << "\n === Additional === ";

cout << "\n 10. Save file in binary";

cout << "\n 11. Save file from binary to text file";

cout << "\n 12 Save file from text to binary file";

cout << "\n 13. Save file in readable text file";

cout << "\n 14. Exit ";

cout << "\n !!! Follow the instructions or ERRORS may occur !!! ";

cout << "\n ";

do {

cout << "\n Your choice: ";

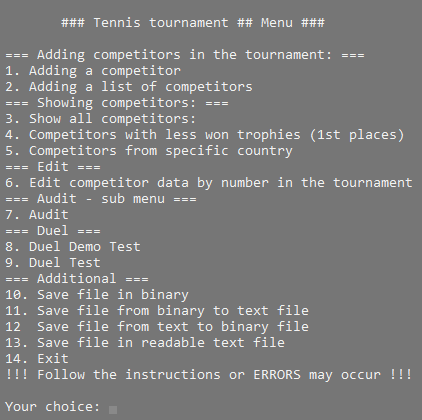
cin >> c;

} while (c < 1 || c > 14);

return c;

}

Функцията приема входни данни от 1 до 14 като всяко число отговаря на определената на него опция в менюто. Въвеждането на число извън този диапазон води до запитване от програма какъв е нашият избор.

Изходни данни на функцията int menu() :

**В. Добавяне на състезатели в турнира.**

**a. Добавяне на нов състезател**

Функция void addOneCompetitor(competitor arr[], int n) , се използва за добавяне на 1 ( един ) нов състезател в турнира ( програмата ). При използването й се създава предварително структура ( typedef struct competitors ) с данните на състезателите (номер в турнира, номер в световната ранглиста, име, фамилия, държава, текущи точки, брой спечелени купи(1ви места)) както и функция ( competitor input() ), която позволява въвеждането на данните на състезателя като след въвеждане изкарва на екрана въведените данни за проверка от потребителя.

Създаване на структура състезатели:

typedef struct competitors //defining structure

{ //using char to leave fields empty when information is missing in adding competitors

int numInTournament; //number in the current tournament

char name[30]; //first name

char surname[30]; //last name

char country[30]; //country

int numWorldRank; //number in the World rank list

int currentPoints; //points in the current tournament

int firstPlaces; //won trophies ( 1st places only )

}competitor;

Създаване на функция, която позволява въвеждането на данните на състезателя като след въвеждане изкарва на екрана въведените данни за проверка от потребителя :

competitor input() //structure for adding competitors

{

competitor a = { 0 }; //defines as an empty structure

cout << "\n Confirm number in the current tournament ";

cout << "\n Number in the current tournament: ";

cin >> a.numInTournament;

cout << "\n Confirm number in the World rank list ";

cout << "\n Number in World rank list (only int number) : ";

cin >> a.numWorldRank;

cout << "\n First Name: ";

cin.ignore(); //ignore or clear one or more characters from the input buffer

cin.getline(a.name, 30);

cout << "\n Last Name: ";

cin.getline(a.surname, 30);

cout << "\n Country: ";

cin.getline(a.country, 30);

cout << "\n Current points (only number) : ";

cin >> a.currentPoints;

cout << "\n Number of won 1st places (only int number) : ";

cin >> a.firstPlaces;

cout << "\n ################################" <<

"\n Number in the current tournament: " << a.numInTournament <<

"\n Number in World rank list: " << a.numWorldRank <<

"\n First Name: " << a.name <<

"\n Last Name: " << a.surname <<

"\n Country: " << a.country <<

"\n Current points: " << a.currentPoints <<

"\n Number of won 1st places: " << a.firstPlaces <<

"\n ################################ " << endl;

return(a); //returning full structure

}

Създаване на функция за въвеждане и запазване в програмата на 1 (един) състезател :

void addOneCompetitor(competitor arr[], int n) //adding ONE competitor in the tournament and append it

{

competitor b; //defines structure

cout << "\n Number in the current tournament (cheks if the number is free or it is taken): ";

cin >> b.numInTournament;

cout << endl;

if (b.numInTournament < 101)

{

//check for unique number in the tournament

for (int i = 0; i < n; i++)

{

if (arr[i].numInTournament == b.numInTournament)

{

cout << "Number in the current tournament is taken \n";

return;

}

}

cout << "\n Number in the world rank list (cheks if the number is free or it is taken): ";

cin >> b.numWorldRank;

cout << endl;

//check for unique number in the tournament

for (int i = 0; i < n; i++)

{

if (arr[i].numWorldRank == b.numWorldRank)

{

cout << "Number in the world rank list is taken \n";

return;

}

}

//opens binary file for appending

fp.open(Filename, ios::binary | ios::app);

//checks for error

if (!fp) { cout << "\n Error in file \n"; exit(1); }

cout << "\n Append new competitor to the tournament";

b = input();

//writting and appending competitor in the end

fp.write((char\*)&b, sizeof(competitor));

fp.close();

}

else if (b.numInTournament > 100)

{

cout << "\n Maximum number is 100 !!! \n";

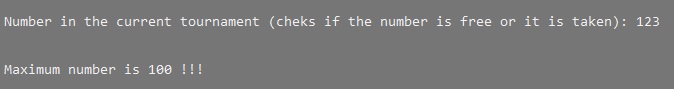
return;

}

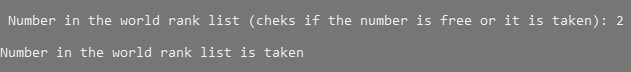
}

Входни и изходни данни при добавяне на състезател:

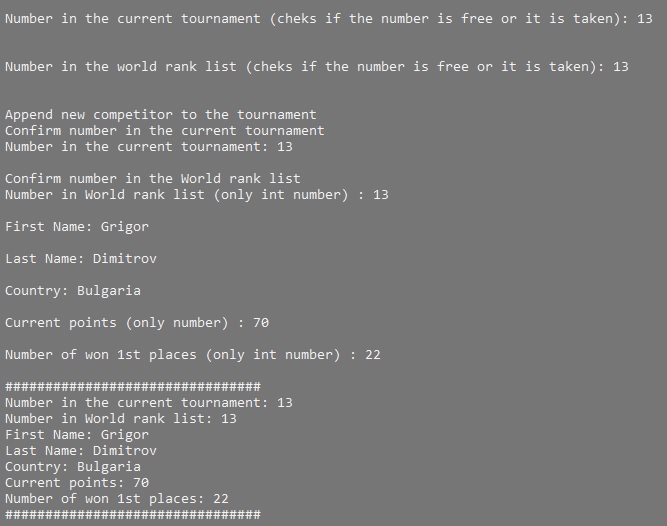
При заето място в турнира:



При заето място в световната ранг листа:



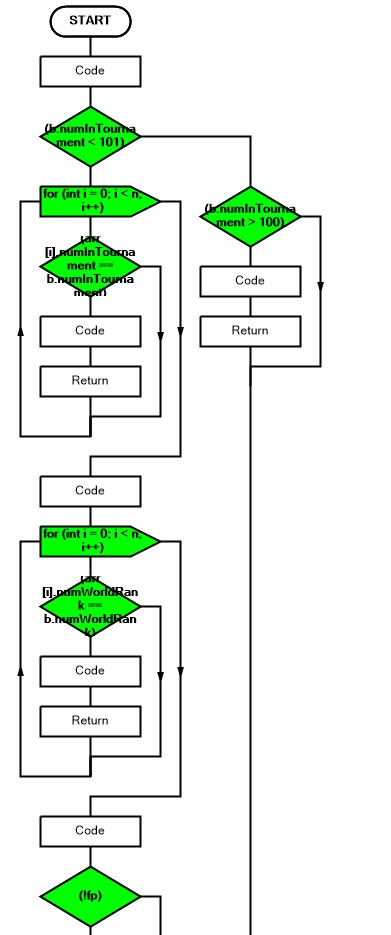
Въвеждане на данни на нов състезател:

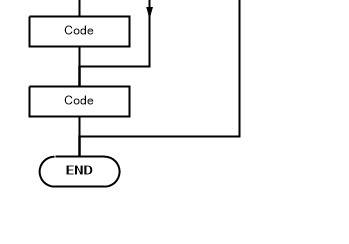


Как работи void addOneCompetitor?

Функцията void addOneCompetitor запазва 1 (един) нов състезател в края на вече съществуващ или наново създаден файл. Преди запазването на състезателя, при въвеждане на желаното място в турнира, програмата проверява дали вече то не е заето или не надвишава допустимият брой участници – 100. Аналогична проверка за заетост се извършва и за мястото в световната ранг листа. При заетост на желано място програмата изписва на екрана съобщение, с което съобщава на потребителя, че мястото е заето. При свободни места в турнира, програмата, вече извършила проверките за свободни места, дава началото на записването на нов състезател чрез функцията competitor input().

Блок схема на void addOneCompetitor():





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

Функция void addManyCompetitors(competitor arr[], int n), се използва за добавяне на n на брой нови състезатели в турнира ( програмата ). При използването й, се вика функцията

void addOneCompetitor() за създаването на всеки състезател, един по един до достигане n на брой въведени състезатели.

void addManyCompetitors(competitor arr[], int n) //adding N number competitors in the tournament and appends them

{

int comp = 0; //comp = competitors

do {

cout << "\n Enter the num of new competitors you want to add (0 < num < 100) : ";

cin >> comp; //Add n competitors

} while (comp < 1 || comp > 101);

//check for free places in the tournament

if (comp + n < 100)

{

for (int i = 0; i < comp; i++)

{ //Call the function below this

cout << "\n Competitor to add No: " << i + 1;

addOneCompetitor(arr, n); //calling function for adding each competitor

}

}

else //if the tournament is full ( 100 competitors )

{

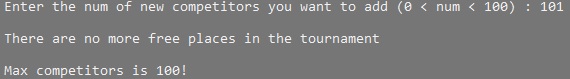
cout << "\n There are no more free places in the tournament \n ";

cout << "\n Max competitors is 100! ";

}

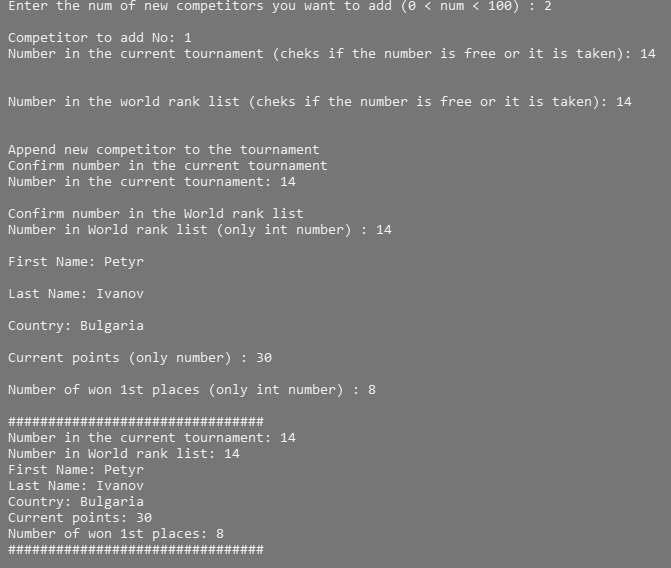
Входни и изходни данни при добавяне n на брой състезатели:

При надхвърляне броя на свободните места :

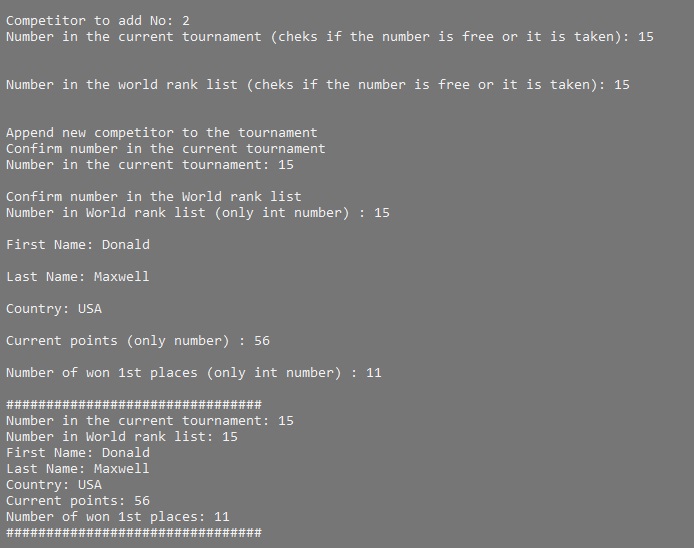


Въвеждане на данни на новите състезатели:

1.1



1.2

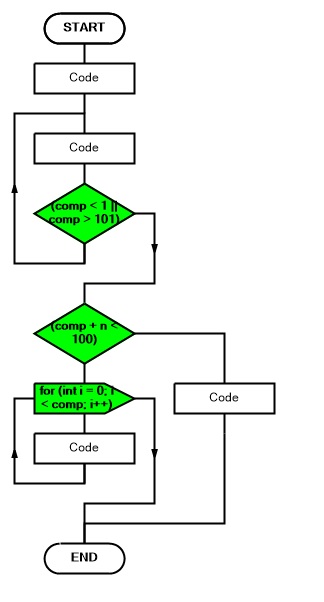


Как работи void addManyCompetitors?

Функцията void addManyCompetitors се изпълнява само когато и само тогава, когато желаният брой състезатели е 100 и при наличие на свободни места в турнира напр. 60. След успешното преминаване проверката за свободни места, функцията извиква функцията

void addOneCompetitor(), за да бъдат създадени записите на новите n на брой състезатели.

Блок схема на void addManyCompetitors :



**C. Извеждане на всички състезатели на екрана**

Функцията void showAllCompetitors извежда на екрана всички записани в състезатели подредени по номер в турнира. При извеждането на списък с всички участници, функцията използва 2 функции:

- sortNumInTournament(arr, n); - сортира състезателите по номер в турнира

-display(arr, i); - извежда на екрана всеки състезател

void sortNumInTournament(competitor arr[], int n) //sorts by size according to the place in the tournament

{ competitor temp; //defines structure / holding variable

int i, j;

for (i = 0; i < n - 1; i++) // element to be compared

{

for (j = i + 1; j < n; j++) // rest of the elements

{

if (arr[i].numInTournament > arr[j].numInTournament) //Ascending Order

{

temp = arr[i]; //swap

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

Функцията void sortNumInTournament обхожда всички елементи като се сравняват по начина на exchange sort. Когато първото преминаване през масива приключи, сортирането взема втория елемент и го сравнява с всеки следващ елемент на елементите за размяна на масива, които не са в ред. Този процес на сортиране продължава, докато се подреди целият масив.

Функцията void display се използва и в други функции с цел по-малко редове!

void display(competitor arr[], int i) //displays the entered data

{

cout << "\n ################################" <<

"\n Number in the current tournament: " << arr[i].numInTournament <<

"\n Number in World rank list: " << arr[i].numWorldRank <<

"\n First Name: " << arr[i].name <<

"\n Last Name: " << arr[i].surname <<

"\n Country: " << arr[i].country <<

"\n Current points: " << arr[i].currentPoints <<

"\n Number of won 1st places: " << arr[i].firstPlaces <<

"\n ################################" << endl;

}

Крайна функция за извеждане на всички състезатели:

void showAllcompetitors(competitor arr[], int n) //showing all competitors in the tournament sorted by number in the tournament

{

int i;

int k = 0;

cout << "\n All competitors: \n";

//all available competitors

for (i = 0; i < n; i++)

{

//numbered sequentially

cout << "\n " << i + 1 << ".";

//sorts by number in the tournament

sortNumInTournament(arr, n);

//displays each competitor after the sorting by num

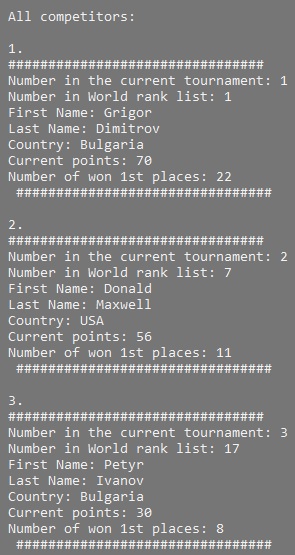
display(arr, i);

k++;

}

}

Примерни изходни данни за извеждане на всички състезатели:



Как работи void showAllcompetitors?

Функцията void showAllcompetitors изкарва на екрана всички записани състезатели като ги номерира последователно в списък, в който са подредени по номер в турнира.

**a. Извеждане на състезател с най-малко спечелени купи**

Функцията **void lessTrophies** извежда на екрана състезателя с най- малко спечелени купи чрез проверка в целия масив

**void lessTrophies(competitors arr[], int n) //showing the competitor with less won trophies ( 1st places )**

{

cout << "\n Competitor with less won trophies (1st places) \n";

int count = 0;

int minTrophies = arr[0].firstPlaces;

//to find the number of minimum cups

for (int i = 0; i < n; i++)

{

if (minTrophies > arr[i].firstPlaces)

{

minTrophies = arr[i].firstPlaces;

}

}

//to go around the competitors again and get those who have exactly as many cups as minTrophies

for (int i = 0; i < n; i++)

{

if (minTrophies == arr[i].firstPlaces)

{

display(arr, i);

count++;

}

}

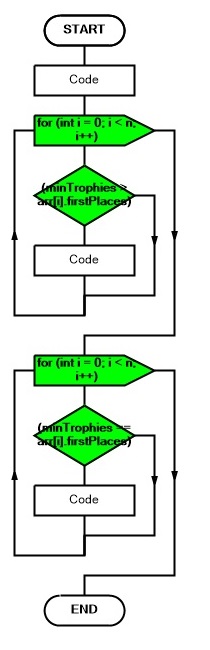
**}**

Изходни данни за извеждане на състезател с най- малко купи:



Как работи **void lessTrophies** ?

Функцията **void lessTrophies** първо обхожда въведените състезатели докато намери състезателя с точно толкова купи колкото minTrophies. След това повторно обхожда въведените състезатели, за да изведе тези / този колкото купи имат равни на minTrophies.

Блок схема на **void lessTrophies** :

**b. Извеждане на състезатели от определена държава**

Функцията **void specificCountry(competitor arr[], int n)** извежда на екрана всички състезатели от желана страна. Във функцията се извиква функция display(arr, i) за визуализация на данните на екрана.

**void specificCountry(competitor arr[], int n) //shows all competitor from wanted specific country**

{

char country[30]; //defines country

int flag = 0, count = 0;

cin.ignore();

cout << "Enter wanted country: ";

cin.getline(country, 30);

// element to be compared

for (int i = 0; i < n; i++)

{

/\*starts comparing the first character of each string.

If they are equal to each other,

it continues with the following pairs

until the characters differ or until a

terminating null-character is reached\*/

if (!strcmp(country, arr[i].country))

{

cout << "\nFound: "; flag = 1;

//displays each competitor from wanted country

display(arr, i);

count++;

}

}

if (count > 0) //checks how many competitors from country are found

{

cout << "\n Total competitors found: " << count << "\n";

}

else //if there are no competitors from wanted country

{

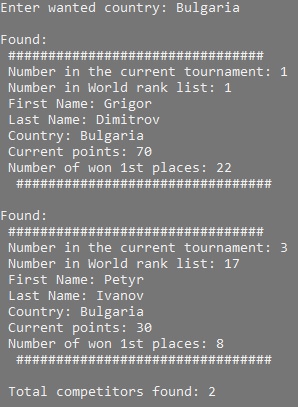
cout << "\n Country not found. \n";

}

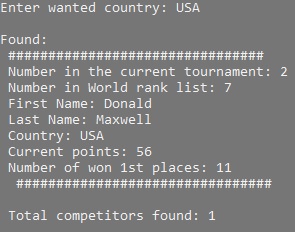
**}**

Примерни входни и изходни данни за извеждане на състезатели от определена страна:

При липса на търсена страна: 



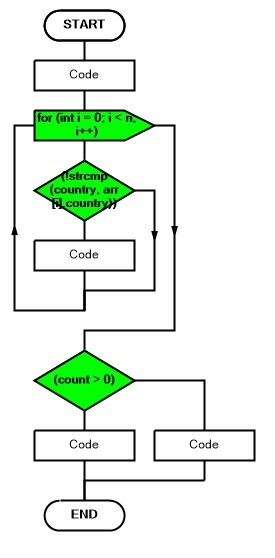
При извеждане на състезател/и от търсена страна:



1.1

1.2

Как работи **void specificCountry** ?

Функцията търси чрез обхождане на целия масив, състезатели от търсената страна. При намирането на състезател/и програмата извежда и точният брой на състезатели от търсената страна. При липса на страна, компилатора извежда съобщение.  
  
Блок схема нa **void specificCountry** : 

**D. Коригиране на данни за състезател.**

**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.**

Функцията void editCompetitor(competitor arr[], int numInTournament, int n) изпълнява условията на а, б и с като при подточка с идентичността на номерата на играчите се установява в началото при добавяне на състезател/и , при коригиране не се допуска промяна на място в турнира и място в световната ранг листа. Във функцията се извиква inputUpdated(arr, i) за въведени и извеждане на коригираните данни.

void editCompetitor(competitor arr[], int numInTournament, int n) //edits competitor info by number in the tournament

{

int count = 0;

//all available competitors

for (int i = 0; i < n; i++)

{

// Check if someone of the elements are equal to inputed tournament number

if (arr[i].numInTournament == numInTournament)

{

competitor b = inputUpdated(arr, i); // new element

b.numInTournament = arr[i].numInTournament; //Save old tournament number

arr[i] = b; //replace current competitor with the new one or updates information

count++;

}

}

if (count == 0) //checks if there is not a competitor with the searched number in the tournament

{

cout << "\n Competitor not found. \n";

}

}

Функция за въвеждане и извеждане на коригираните данни на състезател:

competitor inputUpdated(competitor arr[], int i) //structure for editing competitors information

{

competitor update; //defines structure

cout << "Confirm number in the current tournament ";

cin >> update.numInTournament;

cout << "\n Number in the world rank list: " << arr[i].numWorldRank;

cout << "\n Confirm number in the number in World rank list: ";

cin >> update.numWorldRank;

cout << "\n First Name: ";

cin.ignore(); //ignore or clear one or more characters from the input buffer

cin.getline(update.name, 30);

cout << "\n Last Name: ";

cin.getline(update.surname, 30);

cout << "\n Country: ";

cin.getline(update.country, 30);

cout << "\n Current points (only number) : ";

cin >> update.currentPoints;

cout << "\n Number of won 1st places (only int number) : ";

cin >> update.firstPlaces;

cout << "\n ########### UPDATED ############" <<

"\n Number in the current tournament: " << update.numInTournament <<

"\n Number in World rank list: " << update.numWorldRank <<

"\n First Name: " << update.name <<

"\n Last Name: " << update.surname <<

"\n Country: " << update.country <<

"\n Current points: " << update.currentPoints <<

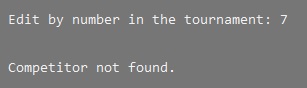
"\n Number of won 1st places: " << update.firstPlaces <<

"\n ################################ " << endl;

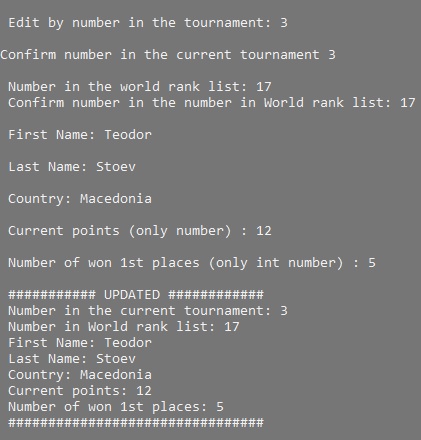
return(update); //returning updated full structure

}

Примерни входни и изходни данни за коригиране на състезател:

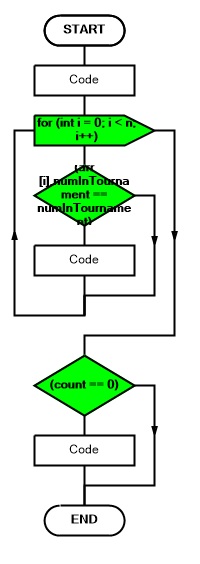


При липса на състезател за корекция на данни:

При нормално коригиране на данни на състезател: 

Как работи void editCompetitor?

Функцията void editCompetitor проверява дали някой отговаря на въведения номер в турнира и ако отговаря отваря функцията competitor inputUpdated за коригиране на данни като запазва номера в турнира и номера в световната ранг листа след корекция. При липса на състезател се извежда съобщение.



Блок схема на void editCompetitor :

**E. Съставяне на двубои за турнира.**

**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.**

**Частта със съставяне на двубои за турнира не е довършена изцяло като това ще бъде функция в програмата, която в бъдеще ще работи и ще предоставя на потребителя възможност за съставяне на двубои и проследяването им от 8мина до пълен финал. В програмата е написана частична функция за двубои 1 в 1 (** int duel(competitor arr[], int n) **) и пълна функция за двубои ( int** duelTest() **) между А и Б, чрез сравнение на 2 числа ( сила на състезател ).**

Функцията int duel, e функция в процес на написване за съставяне от 8мина до пълни финали между състезателите. Във функция int main() си партнира с функцията за извеждане на всички състезатели като по този начин се избират състезатели за двубой.

int duel(competitor arr[], int n)

{

int powerOne, powerTwo;

int i, j;

cout << "\n Enter competitor 1 by number in the tournament: ";

cin >> powerOne;

for (i = 0; i < n; i++)

{

// Check if someone of the elements are equal to inputed tournament number

if (arr[i].numInTournament == powerOne)

{

display(arr, i);

}

}

cout << "\n Enter competitor 2 by number in the tournament: ";

cin >> powerTwo;

for (j = 0; j < n; j++)

{

// Check if someone of the elements are equal to inputed tournament number

if (arr[j].numInTournament == powerTwo)

{

display(arr, i);

}

}

if (arr[i].currentPoints > arr[j].currentPoints)

{

cout << "\n Competitor 1 wins \n ";

}

else if (arr[i].currentPoints < arr[j].currentPoints)

{

cout << "\n Competitor 2 wins \n ";

}

else if (arr[i].currentPoints == arr[j].currentPoints)

{

cout << "\n No winner \n ";

}

return 0;

}

Функцията **int** duelTest() представлява обикновено сравнение на числа, представени като сила на състезател.

int duelTest()

{

int powerOne, powerTwo;

cout << "\n \t ONEvsONE duel - Demo / future function";

cout << "\n Enter competitor 1 power (int): ";

cin >> powerOne;

cout << "\n Enter competitor 2 power (int):: ";

cin >> powerTwo;

if (powerOne > powerTwo)

{

cout << "\n Competitor 1 wins \n ";

}

else if (powerOne < powerTwo)

{

cout << "\n Competitor 2 wins \n ";

}

else if (powerOne == powerTwo)

{

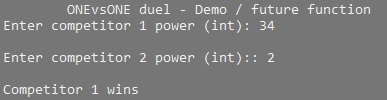
cout << "\n No winner \n ";

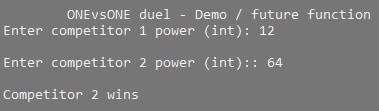
}

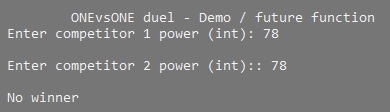
return 0;

}

Примерни входни и изходни данни на функцията:

Състезател 1 победител:

Състезател 2 победител: 

Състезателите са с равна сила: 

**F. Одит на спортистите в подменю**

**a. Извеждане на всички състезатели, сортирани по номер във Световната ранг листа**

**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.**

Подменюто съдържа всички желани действия в една функция **int submenuAudit(competitor arr[], int n)** като за сортирането по номер във Световната ранг листа, сортиране по азбучен ред и сортиране по брой спечелени купи в намаляващ ред са написани допълнителни функции извън подменюто, съответно **void** auditWorldrank(competitor arr[], int n), **void** auditAlphabeticalOrd(competitor arr[], int n**)**, **void** auditFirstPlaces(competitor arr[], int n**)**.

**void** submenuAudit(competitor arr[], int n)

{

int ch;

do {

cout << "\n \t #### SUB MENU #####";

cout << "\n 1. Competitors sorted by World rank ";

cout << "\n 2. Competitors from specific country sorted in alphabetical order";

cout << "\n 3. Competitors from specific country sorted by first places ";

cout << "\n 4. Back to main menu";

cout << "\n Your choice: ";

cin >> ch;

} while (ch < 1 || ch > 4);

switch (ch) {

case 1: auditWorldrank(arr, n); break;

case 2: auditAlphabeticalOrd(arr, n); break;

case 3: auditFirstPlaces(arr, n); break;

}

}

Функция за сортиране на състезатели по номер в световната ранг листа:

void auditWorldrank(competitor arr[], int n) //shows all competitors sorted by number in the World Rankings

{

int k = 0;

cout << "\n All competitors: \n";

//all available competitors

for (int i = 0; i < n; i++)

{

//numbered sequentially

cout << "\n" << i + 1 << "." << "\n \t";

//sorts by number in the world rank list

sortNumWorldRank(arr, n);

//displays each competitor after the sorting by world rank

display(arr, i);

k++;

}

}

void sortNumWorldRank(competitor arr[], int n) //sorts by size according to the place in the World rank list

{

competitor temp; //defines structure / holding variable

int i, j;

for (i = 0; i < n - 1; i++) // element to be compared

{

for (j = i + 1; j < n; j++) // rest of the elements

{

if (arr[i].numWorldRank > arr[j].numWorldRank) //Ascending Order

{

temp = arr[i]; //swap

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

Функцията void sortNumWorldRank обхожда всички елементи като се сравняват по начина на exchange sort. Когато първото преминаване през масива приключи, сортирането взема втория елемент и го сравнява с всеки следващ елемент на елементите за размяна на масива, които не са в ред. Този процес на сортиране продължава, докато се подреди целият масив.

Функция за сортиране на състезатели по азбучен ред:

void auditAlphabeticalOrd(competitor arr[], int n) //show all competitors from a watned country, sorted alphabetically

{

char country[30]; //defines country

int flag = 0, count = 0;

cin.ignore();

cout << "Enter wanted country: ";

cin.getline(country, 30);

// element to be compared

for (int i = 0; i < n; i++)

{

sortName(arr, n); //sorts competitors by name

/\*starts comparing the first character of each string.

If they are equal to each other,

it continues with the following pairs

until the characters differ or until a

terminating null-character is reached\*/

if (!strcmp(country, arr[i].country))

{

cout << "\nFound: "; flag = 1;

//displays each competitor from wanted country

display(arr, i);

count++;

}

}

if (count > 0) //checks how many competitors from country are found

{

cout << "\n Total countries found: " << count << "\n";

}

else //if there are no competitors from wanted country

{

cout << "\n Country not found. \n";

}

}

Функцията void auditAlphabeticalOrd търси чрез обхождане на целия масив, състезатели от търсената страна. При намирането на състезател/и програмата извежда и точният брой на състезатели от търсената страна. При липса на страна, компилатора извежда съобщение.

void sortName(competitor arr[], int n) //sorts by name

{

competitor temp; //defines structure / holding variable

int i, j;

for (i = 0; i < n - 1; i++) // element to be compared

{

for (j = i + 1; j < n; j++) // rest of the elements

{

/\*starts comparing the first character of each string.

If they are equal to each other,

it continues with the following pairs

until the characters differ or until a

terminating null-character is reached\*/

if (strcmp(arr[i].name, arr[j].name) > 0)

{

temp = arr[i]; //swap

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

Функцията void sortName обхожда всички елементи като започва да сравнява първия символ на всеки низ. Ако те са равни помежду си, той продължава със следващите, докато символите се различават или докато не се достигне до краен нулев знак.

Функция за сортиране на състезатели по брой спечелени купи ( 1ви места ) в намаляващ ред:

void auditFirstPlaces(competitor arr[], int n) //show all competitors from a wanted country, sorted by number of cups won in descending order

{

char country[30]; //defines country

int flag = 0, count = 0;

cin.ignore();

cout << "Enter wanted country: ";

cin.getline(country, 30);

// element to be compared

for (int i = 0; i < n; i++)

{

sortFirstPlaces(arr, n); //sorts competitors by won trophies

/\*starts comparing the first character of each string.

If they are equal to each other,

it continues with the following pairs

until the characters differ or until a

terminating null-character is reached\*/

if (!strcmp(country, arr[i].country))

{

cout << "\nFound: "; flag = 1;

//displays each competitor from wanted country

display(arr, i);

count++;

}

}

if (count > 0) //checks how many competitors from country are found

{

cout << "\n Total countries found: " << count << "\n";

}

else //if there are no competitors from wanted country

{

cout << "\n Country not found. \n";

}

}

Функцията void auditFirstPlaces търси чрез обхождане на целия масив, състезатели от търсената страна. При намирането на състезател/и програмата извежда и точният брой на състезатели от търсената страна. При липса на страна, компилатора извежда съобщение.

void sortFirstPlaces(competitor arr[], int n) //sorts by size according to the 1st places / won trophies

{

competitor temp; //defines structure / holding variable

int i, j;

for (i = 0; i < n - 1; i++) // element to be compared

{

for (j = i + 1; j < n; j++) // rest of the elements

{

if (arr[i].firstPlaces < arr[j].firstPlaces) //descending Order

{

temp = arr[i]; //swap

arr[i] = arr[j];

arr[j] = temp;

}

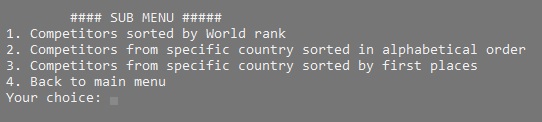
}

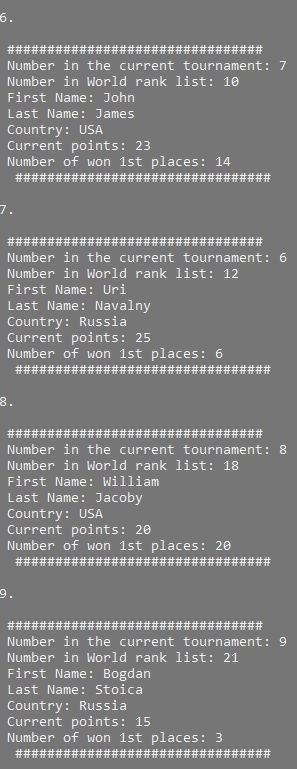
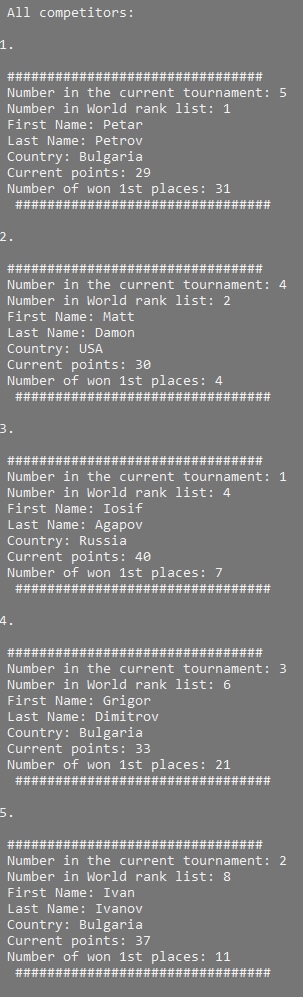
}

}

Функцията void sortNumWorldRank обхожда всички елементи като се сравняват по начина на exchange sort. Когато първото преминаване през масива приключи, сортирането взема втория елемент и го сравнява с всеки следващ елемент на елементите за размяна на масива, които не са в ред. Този процес на сортиране продължава, докато се подреди целият масив.

Примерни входни и изходни данни на точка F:

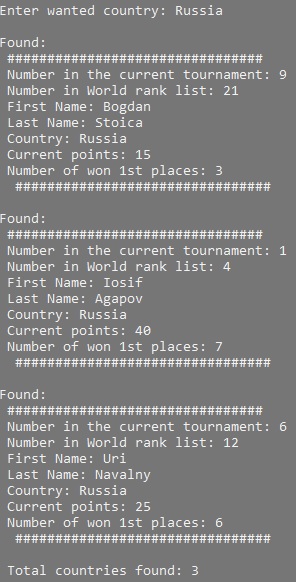
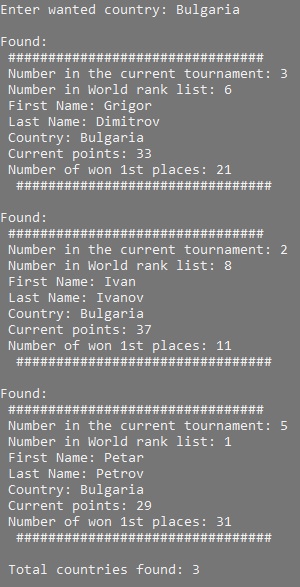




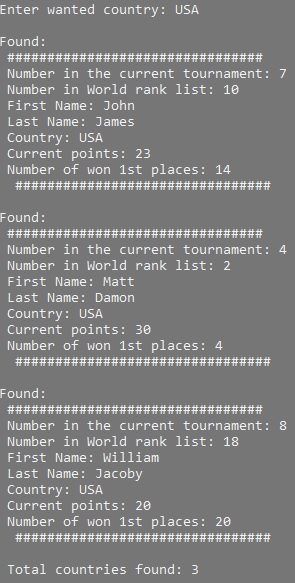
1.2

1.1

Примерни входни и изходни данни на подточка b, за извеждане на състезатели от дадена страна сортирани по азбучен ред:

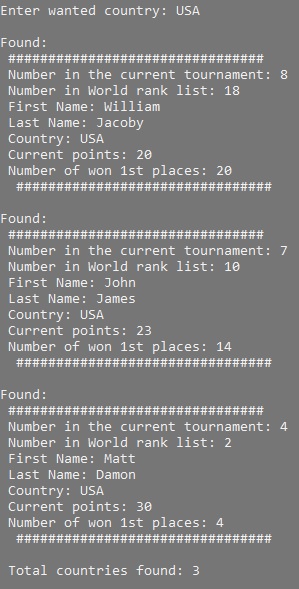
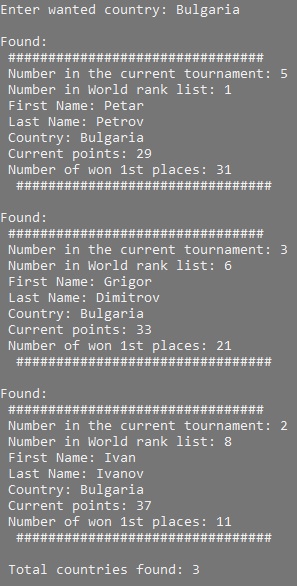


2.1 2.2

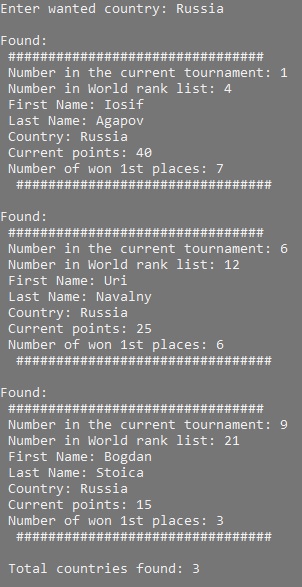


2.3

Примерни входни и изходни данни на подточка b, за извеждане на състезатели от дадена страна сортирани по брой спечели купи в намаляващ ред :



3.1 3.2



3.3

Как работи подменюто?

Избира се опция от менюто  
{

1. Функцията void sortNumWorldRank обхожда всички елементи като се сравняват по начина на exchange sort. Когато първото преминаване през масива приключи, сортирането взема втория елемент и го сравнява с всеки следващ елемент на елементите за размяна на масива, които не са в ред. Този процес на сортиране продължава, докато се подреди целият масив. Извеждат се на екрана екрана вече подредените състезатели чрез функцията display(arr, i).

2. Изборът на страна става по алгоритъма използван във функцията **void specificCountry.** Функцията void sortName обхожда всички елементи като започва да сравнява първия символ на всеки низ. Ако те са равни помежду си, той продължава със следващите, докато символите се различават или докато не се достигне до краен нулев знак. Като изборът на страна става по алгоритъма използван във функцията **void specificCountry.**

3. Изборът на страна става по алгоритъма използван във функцията **void specificCountry.** Функцията void sortNumWorldRank обхожда всички елементи като се сравняват по начина на exchange sort. Когато първото преминаване през масива приключи, сортирането взема втория елемент и го сравнява с всеки следващ елемент на елементите за размяна на масива, които не са в ред. Този процес на сортиране продължава, докато се подреди целият масив.

4. Подменюто се напуска.

}

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

***Данните в програмата биват запазвани след всяко стартиране. При желание данните да бъдат изтрити трябва да бъде изтрит файлът TURNIR.dat от папката на проекта или където се намира .exe файлът на програмата.***

Използваните функции за запазване във файл са void saveInFile(competitor arr[], int n) и част от void addOneCompetitor(competitor arr[], int n), която при всяко добавяне на състезател директно го запазва във файл. Файлът бива генериран автоматично след 1вият добавен състезател в програмата. Функцията void saveInFile е добавена като опция в менюто за подсигуряване запазването на данните преди напускане на програмата.

Функцията void saveInFile :

void saveInFile(competitor arr[], int n) //saving binary file

{

//opens a file named Filename ( file constanta ) as a binary for writing

fp.open(Filename, ios::binary | ios::out);

//interrupts in case of a problem

if (!fp) { cout << "\n Error in file \n"; exit(1); }

//writes down the whole tournament program data which has n number competitors \* (multiplied) by one competitor structure

fp.write((char\*)arr, sizeof(competitor) \* n);

fp.close(); //closing file

}

Функцията void addOneCompetitor :

**void addOneCompetitor(competitor arr[], int n) //adding ONE competitor in the tournament and append it**

{

competitor b; //defines structure

// . . .

//opens binary file for appending

fp.open(Filename, ios::binary | ios::app);

//checks for error

if (!fp) { cout << "\n Error in file \n"; exit(1); }

cout << "\n Append new competitor to the tournament";

b = input();

//writting and appending competitor in the end

fp.write((char\*)&b, sizeof(competitor));

fp.close();

}

// . . .

**}**

**H. Допълнителни условия:**

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

Данните в програмата биват записани в двоичен файл автоматично.

За да запазите данните от двоичен файл в текстов, се използва функцията int saveTXT(). За запазването да данни от текстов в двоичен файл се използва функцията int saveBINARY().

Добавена е и опция за запазването в текстов, даващ възможност за прочитане от потребителя void saveReadableTXT(competitor arr[], int n).

Функциите следва да бъдат използвани в следната последователност:

1. След добавянето на състезатели е генериран файл, готов за употреба от всички функции в програмата.

2. При използването на функцията int saveBINARY() трябва преди това задължително да бъде използвана функцията saveTXT() , защото това може да доведе до генерирането на неизползваеми файлове.

Функцията int saveTXT() :

int saveTXT() //saves file with all competitors in txt

{ /\* this function gets the data from already existing

binary file and copy it in a new txt file\*/

ifstream fin; //only for reading fin

fin.open("TURNIR.dat"); //opens binary file

ofstream fout; //only writting fout

fout.open("TURNIR.txt"); //opens txt file

char ch;

//reading all the way to the last symbol from the binary file

while (!fin.eof())

{

fin.get(ch);

fout << ch;

}

//closing both files

fin.close();

fout.close();

return 0;

}

Функцията int saveBINARY() :

int saveBINARY() //saves file with all competitors in binary from existing txt file

{

/\* this function gets the data from already existing

txt file and copy it in a new binary file\*/

ifstream fin; //only for reading fin

fin.open("TURNIR.txt"); //opens txt file

ofstream fout; //only writting fout

fout.open("TURNIR\_NEW.dat"); //opens binary file

char ch;

//reading all the way to the last symbol from the binary file

while (!fin.eof())

{

fin.get(ch);

fout << ch;

}

//closing both files

fin.close();

fout.close();

return 0;

}

Функцията int saveTXT() копира съдържанието от автоматично генерираният двоичен файл от програмата и го запазва в нов текстов файл. Функцията saveBINARY() копира съдържанието от създаденият от потребителя текстов файл чрез функцията int saveTXT() и го запазва в нов двоичен файл.

!!! Няма как да бъде създаден 1во двоичният файл чрез saveBINARY() , преди да бъде употребена функцията int saveTXT() !!!

Функцията void saveReadableTXT е независима от останалите функции за запазване.

Функцията void saveReadableTXT:

void saveReadableTXT(competitor arr[], int n)

{

fstream file;

file.open("TURNIR\_READABLE.txt", ios::out);

for (int i = 0; i < n; i++)

{

file << arr[i].numInTournament << endl;

file << arr[i].numWorldRank << endl;

file << arr[i].name << endl;

file << arr[i].surname << endl;

file << arr[i].country << endl;

file << arr[i].currentPoints << endl;

file << arr[i].firstPlaces << endl;

}

file.close();

}

**int main() function**

Функцията int main() използва функцията int loadFile(competitor arr[]),

за зареждане на двоичният файл генериран автоматично от програмата преди всяка функция, която има нужда до достъп до списъка със състезателите. Int main работи на принципа на цикъл със след условие с вложен switch, за избор на опциите в менюто.

int main()

{

competitor arr[N]; //array of structs with limit N = 100

int choice;

int n = 0;

n = loadFile(arr);

do {

choice = menu();

switch (choice) {

case 1: addOneCompetitor(arr, n); n++; break;

case 2: { system("cls"); } addManyCompetitors(arr, n); break;

case 3: {system("cls"); } {n = loadFile(arr); showAllcompetitors(arr, n); }; break;

case 4: {system("cls"); } {n = loadFile(arr); lessTrophies(arr, n); }; break;

case 5: {system("cls"); } { n = loadFile(arr); specificCountry(arr, n); }; break;

case 6: {system("cls"); } { n = loadFile(arr);

cout << "\n Edit by number in the tournament: ";

int numInTournament;

cin >> numInTournament;

cout << endl;

editCompetitor(arr, numInTournament, n);

saveInFile(arr, n);

};

break;

case 7: n = loadFile(arr); submenuAudit(arr, n); break;

case 8: {system("cls"); }

{ cout << "\n \t ONEvsONE duel - Demo / future function";

n = loadFile(arr);

showAllcompetitors(arr, n);

duel(arr, n);

};

break;

case 9: {system("cls"); } duelTest(); break;

case 10: saveInFile(arr, n);

{

cout << "\n File saved as TURNIR.dat as binary \n";

}

break;

case 11: saveTXT();

{

cout << "\n File saved as TURNIR.txt as txt \n";

}

break;

case 12: saveBINARY();

{

cout << "\n File saved as TURNIR\_NEW.dat as binary \n";

}

break;

case 13: saveReadableTXT(arr, n);

{

cout << "\n File saved as TURNIR\_READABLE.txt as txt \n";

}

break;

}

} while (choice != 14);

return 0;

}

Функцията int loadFile :

int loadFile(competitor arr[]) //loads binary file

{

long pos; //position

int n = 0, i; //n = 0 = empty tournament

competitors b; //defines structure

//open the binary file for reading

fp.open(Filename, ios::binary | ios::in);

//in case of a problem it returns back to main()

if (!fp) { cout << "\n TURNIR.dat file not exist\n"; return n; }

//take the number of bytes and closes it

fp.seekg(0l, ios::end);

pos = fp.tellg();

fp.close();

n = pos / (sizeof(competitor));

//open the file for reading again

fp.open(Filename, ios::binary | ios::in);

//checks for error

if (!fp) { cout << "\n Error in file \n"; exit(1); }

//read structure by structure people and put them in an array

for (i = 0; i < n; i++)

{

fp.read((char\*)&b, sizeof(competitor));

arr[i] = b;

}

fp.close(); //closing file

return n; //with return, I return n to the calling function

}

**Упътване за употреба**

**Програмата е направена лесна за употреба с нужните обяснения по време на работа с нея и по време на работа с кода на програмата.**

**1. Избор на опция от менюто / подменюто**

**Всички възможни опции са визуализирани на екрана , при надписването на число НЕ фигуриращо в менюто, програмата няма да изведе нищо или ще изведе отново редът, с който очаква вашият избор.**

**Не бива да се въвеждат думи, букви, символи и т.н. това може да доведе до повреда в програмата и нейното нужно рестартиране.**

**2. Добавяне на състезател / състезатели**

**2.1 Добавяне на 1 (един) състезател**

**При добавянето на състезател програмата следва да провери за заетостта на мястото в турнира и в световната ранг листа, защото не може да има двама състезателя с едни и същи номера. След проверката за уникалност на номерата, се добавят данните на състезателя като при успешно преминаване на проверката следва да бъдат въведени отново желание номера в турнира и световната ранг листа. Те са само и единствено цели положителни числа. Следва добавяне на име, фамилия и страна на състезателя. Трите полета трябва да бъдат попълнени само с букви, не се приемат цифри, знаци и т.н. тъй като това може да повлияе на методите на сортиране на програмата. Накрая се добавят текущи точки в турнира и брой спечелени 1ви (първи) места. При наличието на състезател с желаните данни за въвеждане ( номер в турнира, номер в световната ранг листа), програмата съобщава за заетостта на мястото и ни връща обратно в основното меню.**

**След успешно добавяне на състезател, програмата извежда оградено поле за проверка на въведените данни.**

**2.2 Добавяне на n на брой състезатели**

**При добавянето на n на брой състезатели, програмата първо проверява дали има свободно място в турнира за желаният брой състезатели да бъдат добавени. При наличието на свободни места се добавят състезателите както следва по т. 2.1. При не достатъчно място в турнира, за желаният брой състезатели да бъдат добавени, програмата извежда на екрана съобщение след което ни връща обратно в основното меню.**

**3. Извеждане на всички състезатели на екрана**

**Извеждат се на екрана всички записани състезатели в програмата, сортирани по номер в турнира.**

**3.1 Извеждане на състезател с най- малък брой спечелени купи ( 1ви места )**

**Извежда се на екрана състезателя с най- малък брой спечелени купи**

**3.2 Извеждане на състезатели от определена страна**

**Програмата предоставя възможност на потребителя да търси състезатели от определена ( желана от потребителя ) страна. При избор на опцията, програмата очаква да бъде въведена желаната страна. При наличие на състезател/и от въпросната страна , програмата ги изкарва на екрана. При липсващ/и състезател/и от търсената страна, програмата изкарва съобщение на екрана за липсата на страната в записа на програмата и ни връща обратно в менюто.**

**4. Коригиране на данни за състезател**

**Опцията позволява на потребителя да коригира, актуализира данни на състезател като за да се направи корекция, програмата първо проверява дали съществува състезател с търсеното място в турнира. При съвпадение потребителя потвърждава въведения номер в турнира. Следва потвърждаване на номер в световната ранг листа и след това идва коригирането на данни на състезател ( име, фамилия, страна, текущи точки, брой спечелени купи ). Въвеждането на данни става аналогично на т. 2.1. При не намиране на състезател с търсеният номер в турнира, програмата извежда съобщение на екрана за липсата на състезател и отново ни връща в менюто.**

**5. Съставяне на двубои за турнира – бъдеща опция**

**Опцията е още в разработка!**

**Опцията двубои ще дава възможност на потребителя да симулира двубои между въведените в програмата състезатели. Потребителят ще има избор из между:**

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

**5.2 Добавяне на резултати за двубоите.**

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

**6. Одит на спортистите в подменю**

**Дава допълнителни опции за сортиране на състезателите в турнира**

**6.1 Опция за сортиране на състезатели по номер в световната ранг листа**

**Извежда на екрана всички състезатели, сортирани по номер в световна ранг листа**

**6.2 Опция за извеждане на всички състезатели от дадена държава, сортирани в азбучен ред**

**Опцията работи на принципа на т. 3.2 като изведените страни са сортирани по азбучен ред**

**6.3 Опция за извеждане на всички състезателите от дадена държава, сортирани по брой спечелени купи в намаляващ ред**

**Опцията работи на принципа на т. 3.2 като изведените страни са сортирани по брой спечелени купи ( 1ви места ) в намаляващ ред**

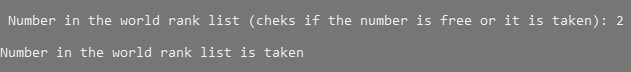
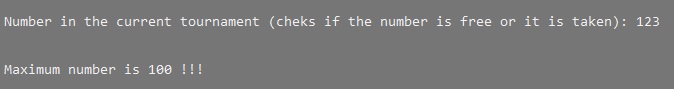
**7. Данните в програмата могат да се запазват във файл – двоичен, текстов, текстов за четене**

**Данните в програмата могат да се запишат от двоичен в текстов както и от текстов в двоичен фаил.**

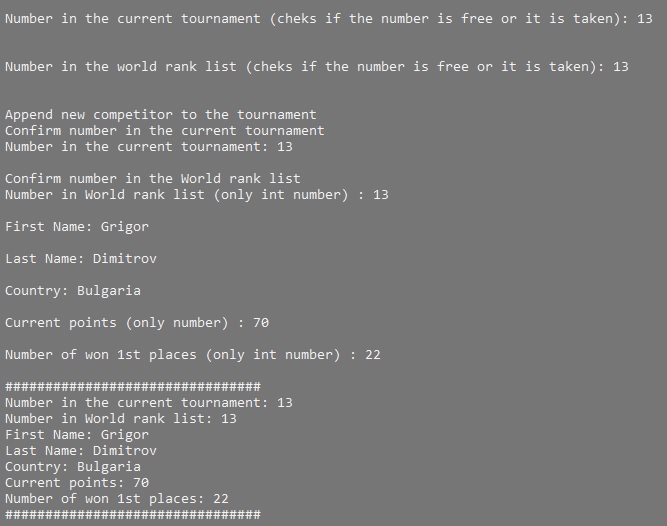
**Примерно действие на програмата**

**1. Добавяне на състезател**

Проверка на номер в турнира:

Проверка на номер в световната ранг листа:

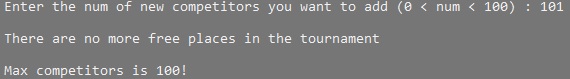
Въвеждане на данни на нов състезател:



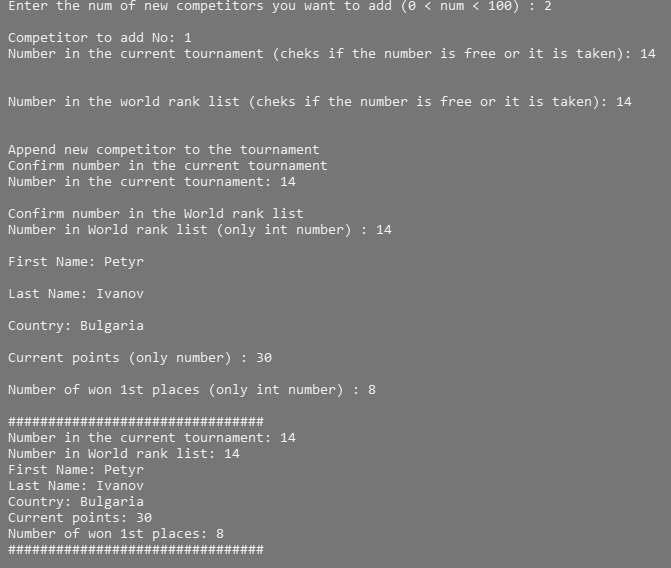
**2. Добавяне на n на брой състезатели**

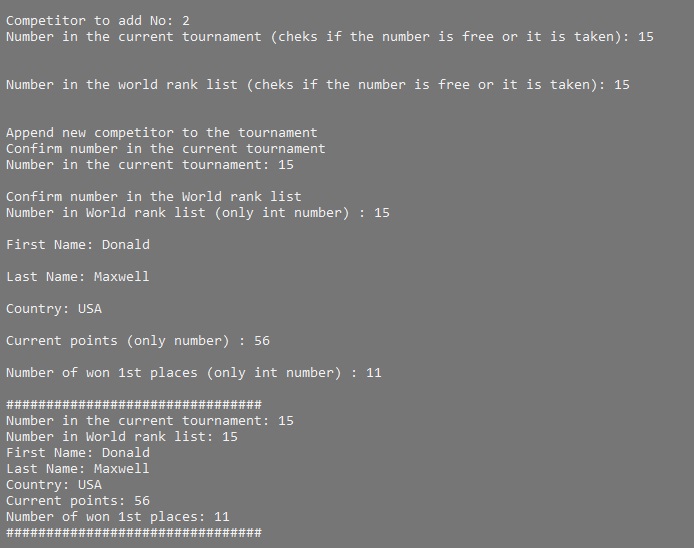
Проверка за свободни в турнира:



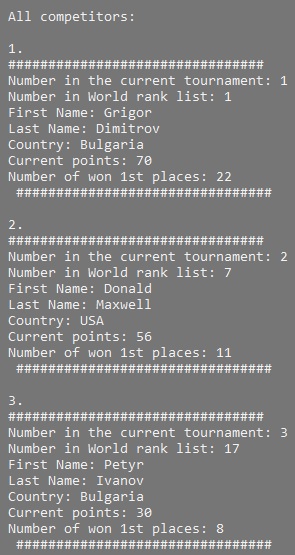


Въвеждане на данни на нови състезатели:

 1.1

 1.2

**3. Извеждане на всички състезатели:**



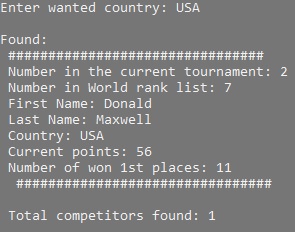
**4. Извеждане на състезатели с най-малко спечелени купи:**

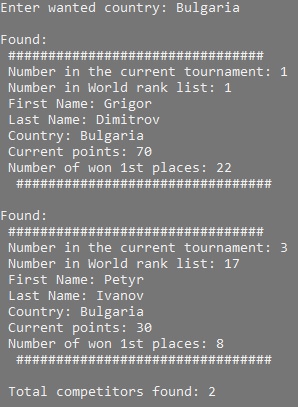


**5. Извеждане на състезатели от определена държава:**

При липса на търсена страна:

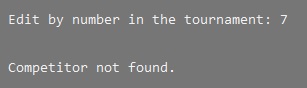
При наличието на търсената страна:



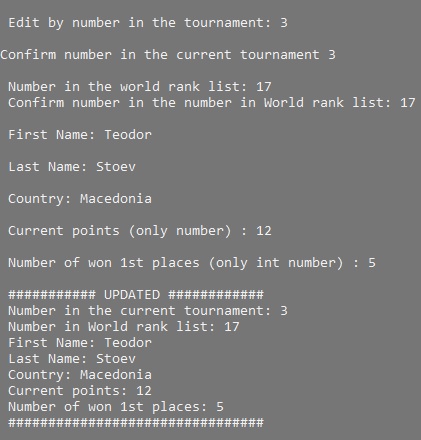


**6. Коригиране на данни за състезател:**

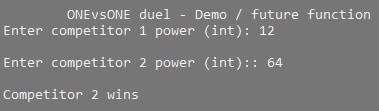
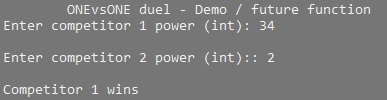
При липса на състезател:



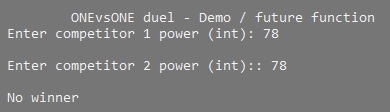
Коригиране на данни на състезател:



**7. Съставяне на двубои за турнира:**

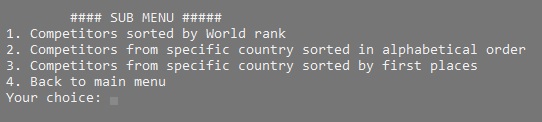
При победител състезател 1 : 

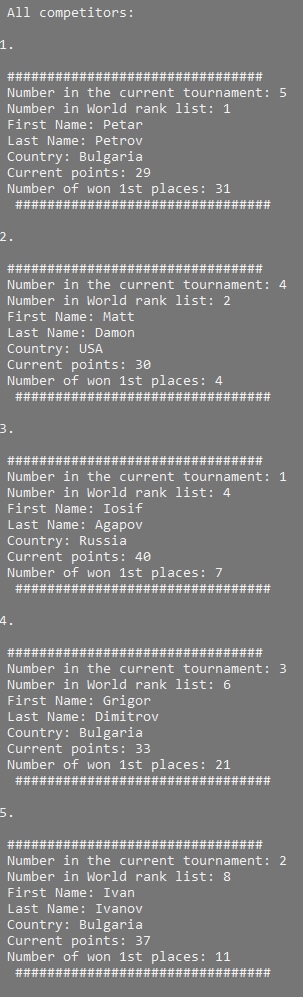
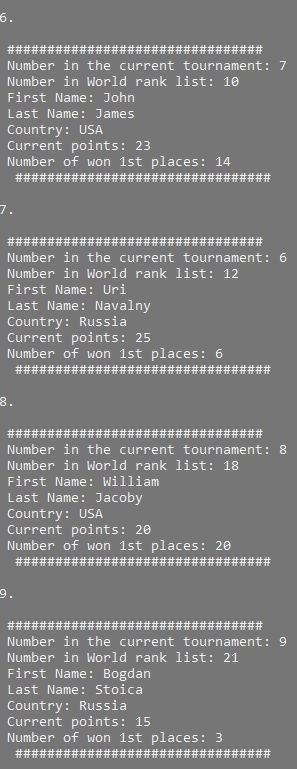
При победител състезател 2 :

При равен брой сила: 

**8. Одит на спортистите в подменю**

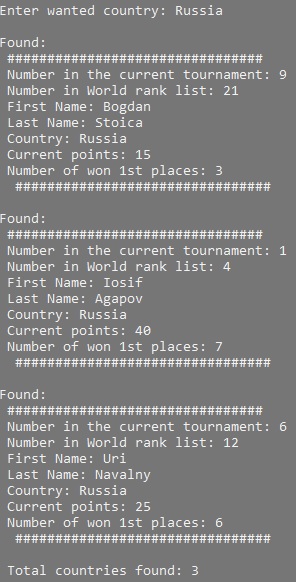
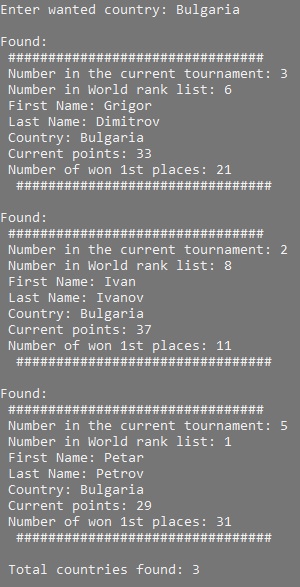
Изглед на подменюто:



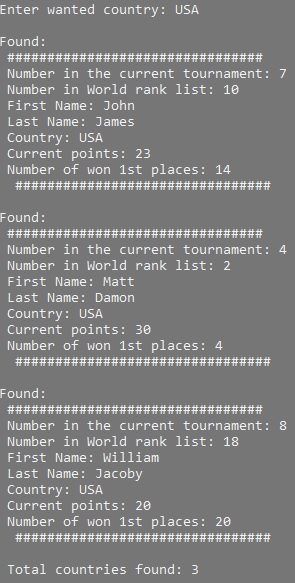
**9. Извеждане на всички състезатели, сортирани по н****омер във Световната ранглиста**

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

При наличие на търсената страна:



10.1 10.2

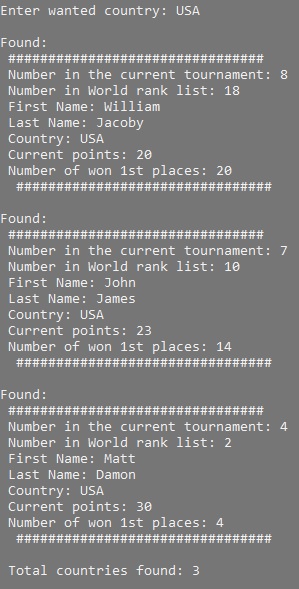
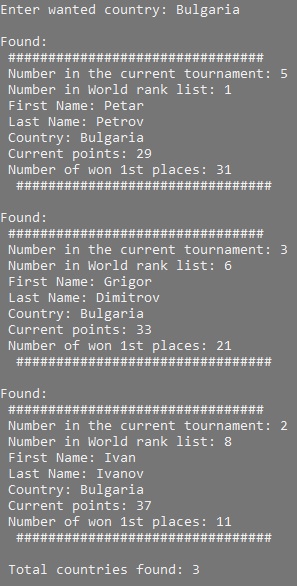


10.3

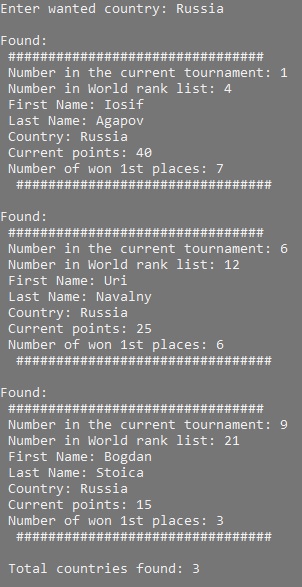
При липса на търсената страна:



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



11.1 11.2



11.3

При липса на търсената страна: 

**12. Запазване в двоичен файл**



**13. Запазване от двоичен в текстов файл**



**14. Запазване от текстов в двоичен файл**



**15. Запазване в текстов файл за четене от потребителя**



**######################################################################################################################################################################**

**Кодът на програмата „Турнир по тенис“,**

**заедно с всички блок схеми може да бъде намерен на :**

**https://github.com/StoyanovBG/TennisTournament.git**