Technical University of Varna  
  
Coursework Documentation

Faculty of Computing and Automation  
Department: Computer Science and Technology (CST)

Specialty: Computer Systems and Technologies (CST)

Subject: Basic Programming  
Group: 1B  
Coursework Topic: Football Tournament

Prepared by: Stefan Hristov Dimitrov

Faculty Number: 19621416

Football Tournament

A computer program must be written to implement an information system that manages a football championship (Registration Number, Team Name, Year of Establishment, list of played matches). The program stores and processes data about the participating teams and the matches played. The participants in the championship are between 3 and 5 teams.

The program should include:

A. A menu for selecting program functions.

B. Adding participants to the championship.

a. Adding a single participant (Name and City);

b. Adding a list of participants. An integer n is entered, followed by n participants.

C. Displaying all participants in the championship on the screen.

D. Conducting the tournament.

a. All teams play against each other in alternating venues, a random match result is generated, and it is recorded in a list of played matches for the two participating teams.

b. A table is displayed with the played matches and results.

E. Searching for teams in the championship:

a. Displaying the team with all its played matches by entering the team's name.  
 b. Displaying the results of the played matches between two teams, by entering their names.

F. Reports on the participating teams with a submenu:

a. Displaying all participants sorted by:

I. Year of establishment;  
 ii. Points won.

b. Displaying the team that has scored the most goals  
 c. Displaying the team with the best goal difference.

G. The data in the program should be saved to a file between two program starts.  
 H. Additional conditions:  
 a. For point B, when adding a new participant, there should be an option to choose whether to add a new one or to stop the input.  
 b. For point C, the teams should be displayed in a table format (Team Name, goals scored, goals conceded, points, and matches played with results).

c. For point G, the data should be saved in a binary file.  
  
  
**For the implementation of the program, the following were used: various types of libraries, structures, logical operators, arrays, two-dimensional arrays, loops, strings, conditional operators, a seed generator from local time, bubble sort method, file handling. The task was implemented using the C++ language.**

**Program Functions**

**otbor enter() -** Single team input.

**void otbor\_input(otbor o[], int counter, int nomer\_list) -** Input a list of teams.

**void otbor\_output(otbor o[]) -**  Output all entered teams.

**void izpisvane\_na\_match(match m[N][N], otbor o[], int k, int i)-** Display matches from the tournament with random results.

**void turnir(match m[N][N], otbor o[], int counter) –** Conduct the football tournament.

**void tursene\_s\_ime(match m[N][N], otbor o[], int counter) –** Display all matches played by a team by entering an existing team's name.

**void tursene\_po\_dve\_imena(match m[N][N], otbor o[], int counter) –**  Display matches between two teams by entering their names.

**void Spravki\_Podmenu(otbor o[], int counter, match m[N][N]) – S**ubmenu for reports on the participating teams.

**void Sortirane\_Godina(otbor o[], int counter) –** Sort the entered teams by year of establishment.

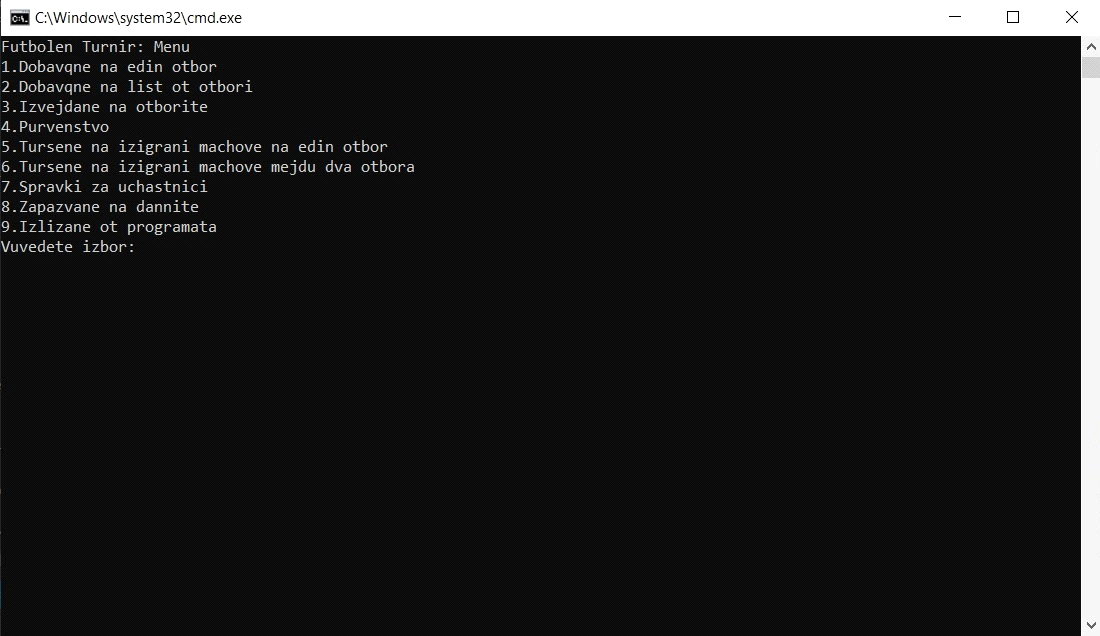
**void Sortirane\_Tochki(otbor o[], int counter, match m[N][N]) –** - Sort the entered teams by points won in the tournament.

**void Izvejdane\_Max\_Popadeniq(otbor o[], match m[N][N], int counter) –** Display the team with the most goals in the tournament.

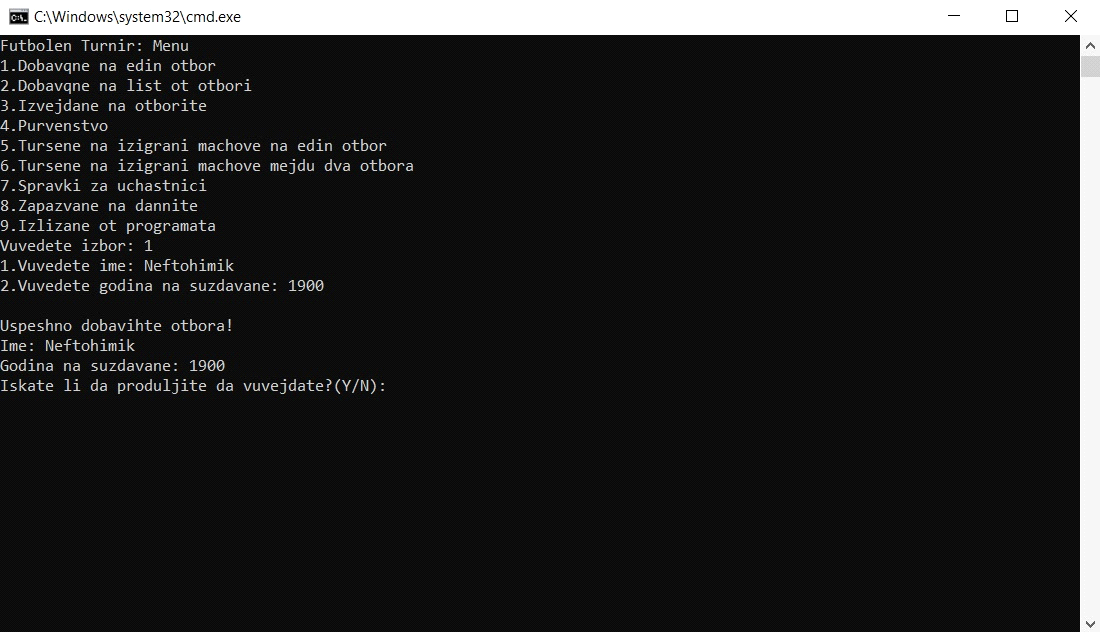
**void Izvejdane\_Max\_Razlika(otbor o[], match m[N][N] –** Display the team with the best goal difference in the tournament.

**void Izkopirane\_Otbori(otbor o[], int counter, otbor o1[]) -** Copy the team data (Name and Year) from o[] to o1[] to prevent data rearrangement in o[] when sorting using the bubble sort method.

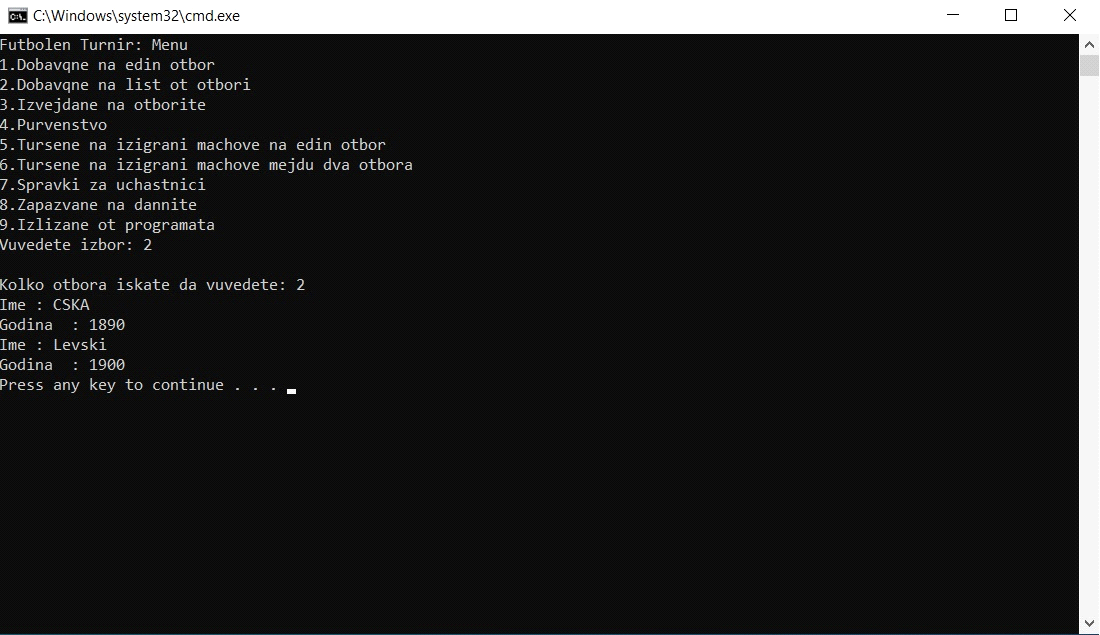
**void Save(otbor o[], match m[N][N], int counter) –** Save the name and year of establishment of the teams and the matches from the last tournament in a binary file, overwriting the old data in the file.



Initial view of the program. There is a switch case menu and two do-while loops. The first loop prompts us to enter a number between 1 and 9, requiring a new selection every time we input a number outside this range. The second loop repeats the menu until we select option 9 to exit.

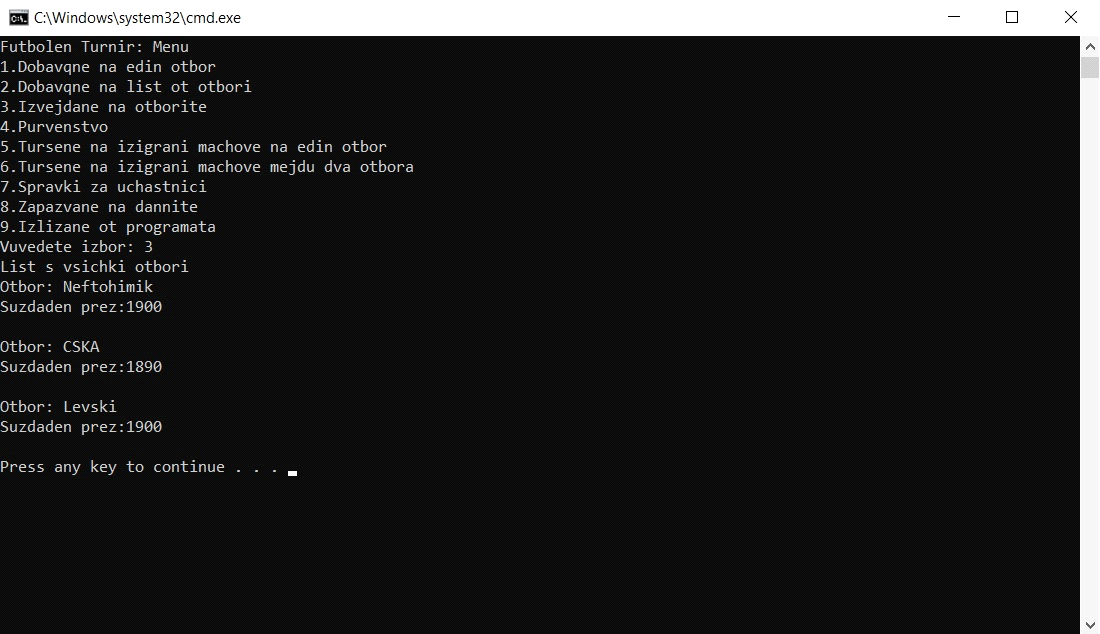
   
otbor enter()

Option 1 prompts us to enter a single team, and if we want, we can continue entering teams by choosing y/Y to continue or N/n to stop, which returns us to the main menu. Here, the team name and year of establishment are entered, and it does not allow us to enter more than N(5) teams.



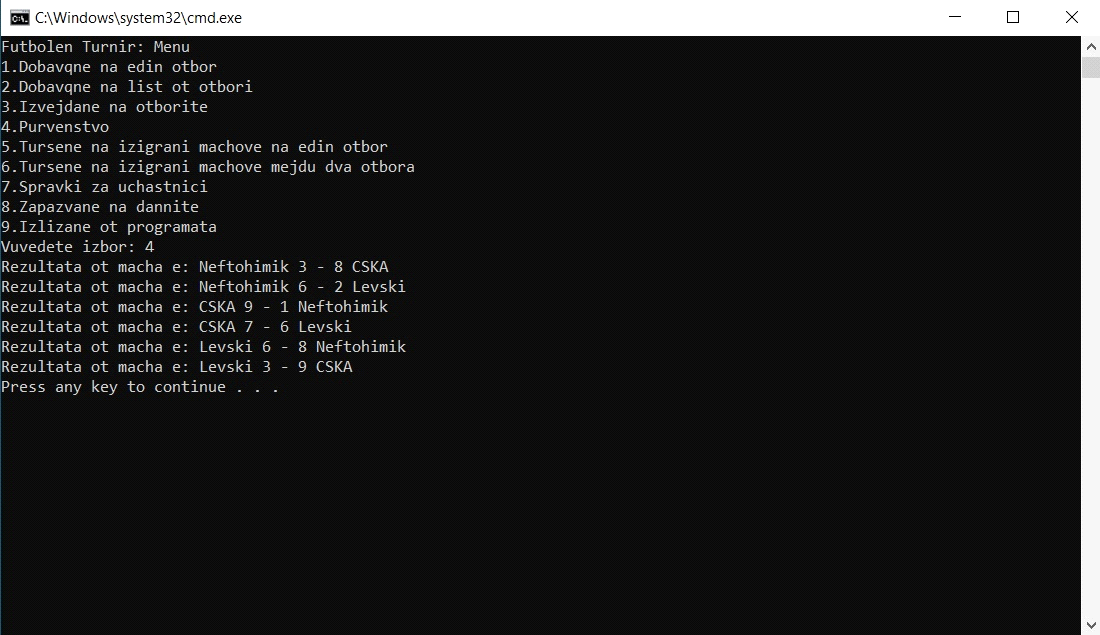
otbor\_output(o, counter, nomer\_list)

Option 2 prompts us to enter a list of teams. First, it asks us to enter the number of teams we want to input, and then we add the name and year of establishment. It ensures that the team limit (up to 5) is respected by first checking if we’ve reached 5 teams and then verifying whether the list of teams we want to add will exceed the total number of N(5) teams.



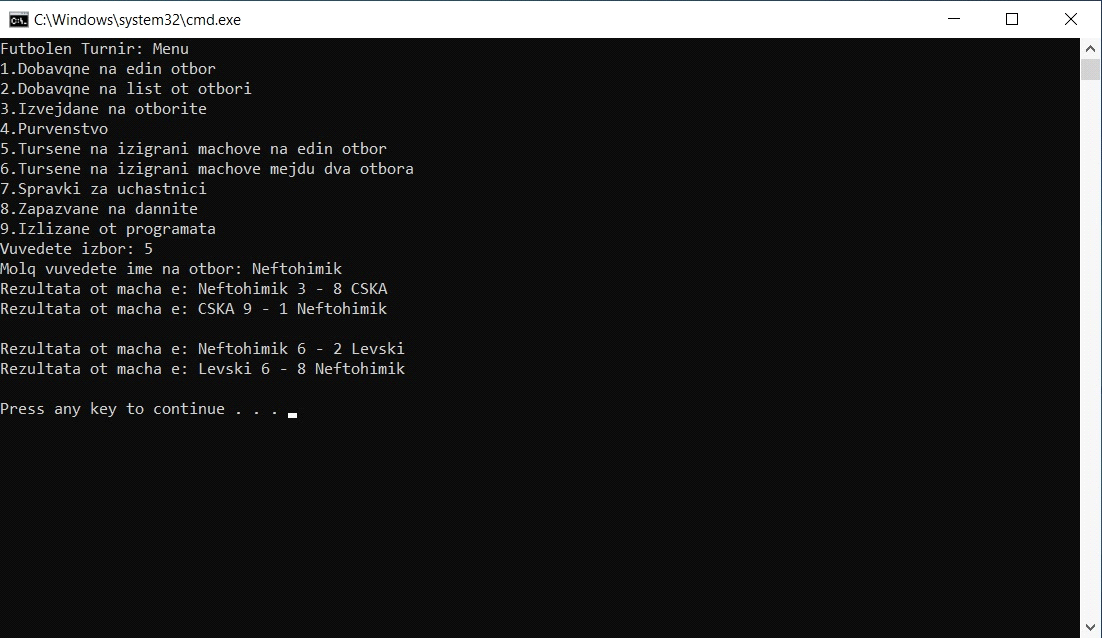
otbor\_output(o)

Option 3 displays all the teams one by one.



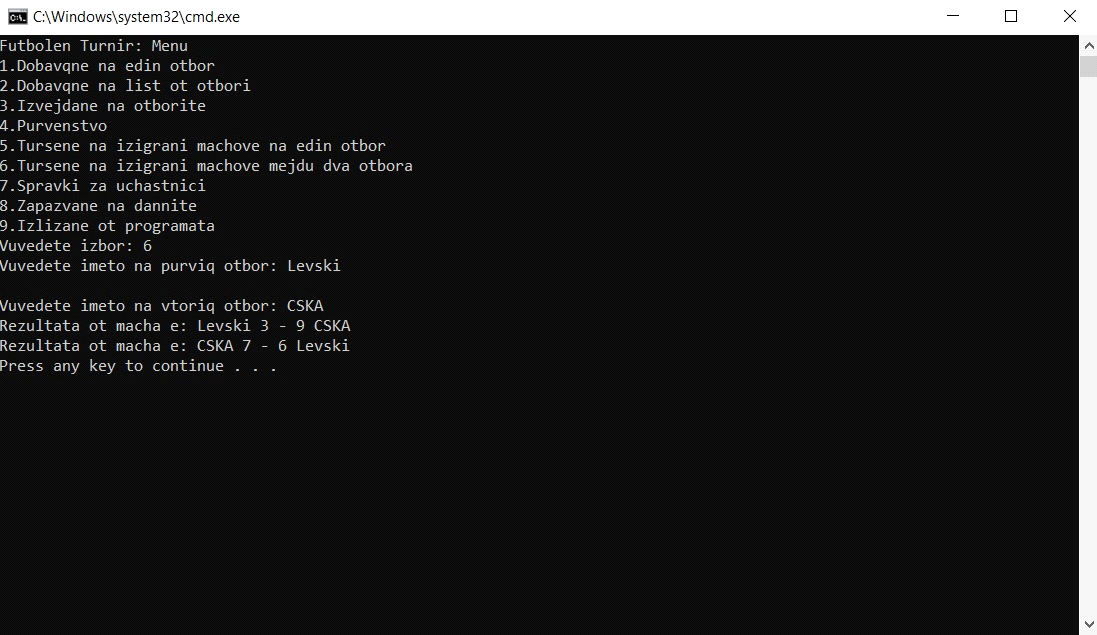
turnir(m, o, counter)

Option 4 conducts the tournament. First, it checks if there are at least 3 registered teams, and then, using a two-dimensional array, it makes every team play against each other (using two for loops). An if statement prevents teams from playing against themselves, and random goals are displayed for both teams.

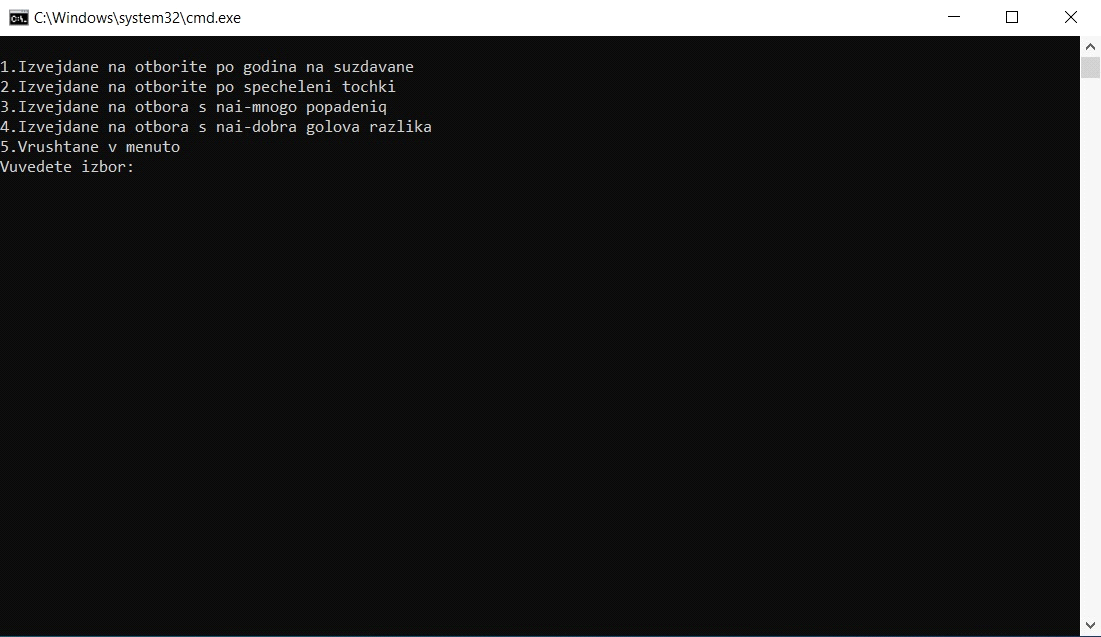


tursene\_s\_ime(m, o, counter)

Option 5 prompts us to enter a team name. It checks if the entered team exists and displays all the matches that team has played in the tournament. If the entered team does not exist, it returns us to the main menu.

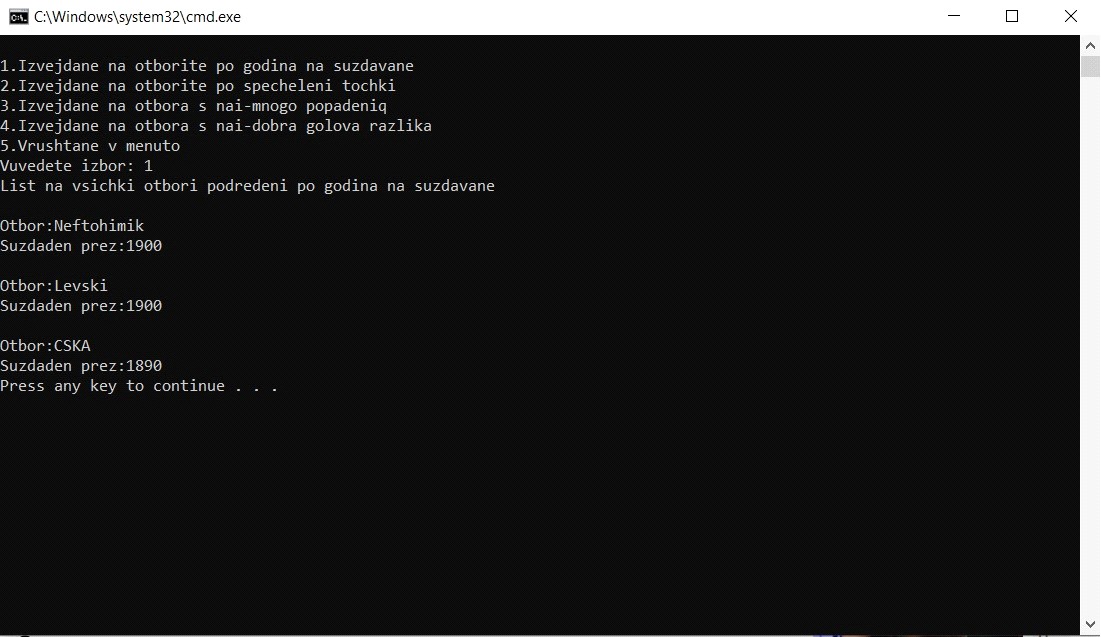
  
tursene\_po\_dve\_imena(m, o, counter)

Option 6 prompts us to enter the names of two teams. It checks if both teams exist and displays the matches between the two teams from the tournament. If even one of the entered teams does not exist, it returns us to the main menu.



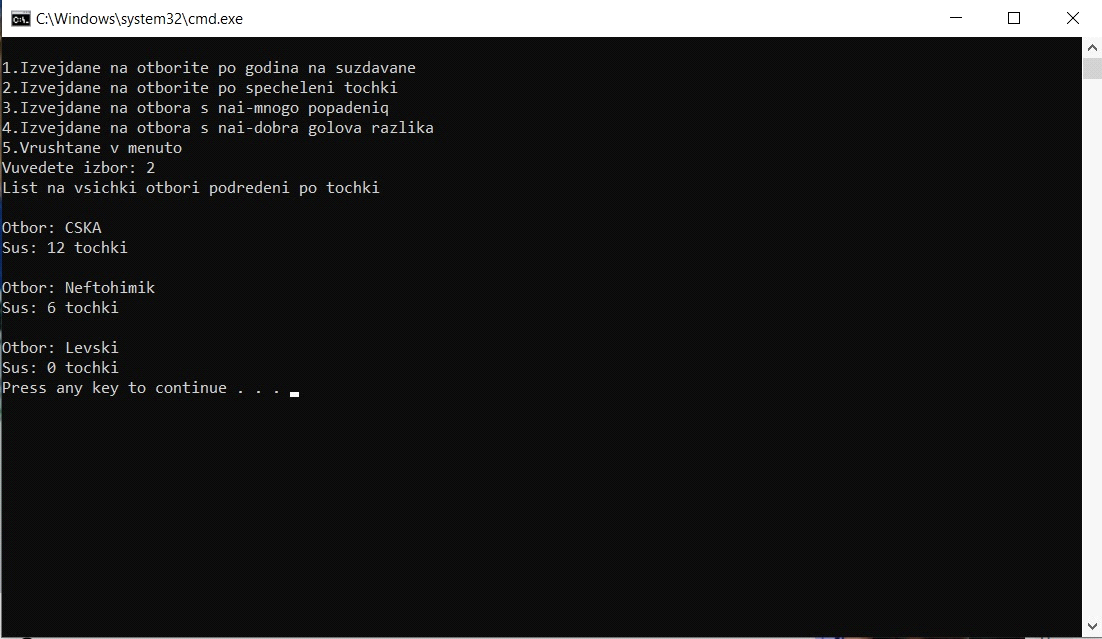
Spravki\_podmenu(o, counter, m)

Option 7 is a submenu. As a function, it is similar to the main program menu, but the limit is from 1 to 5. Selecting option 5 exits the submenu and returns to the main menu. (For easier understanding of the functions in the submenu, they will be referred to as options: 7.1, 7.2, 7.3, and 7.4)



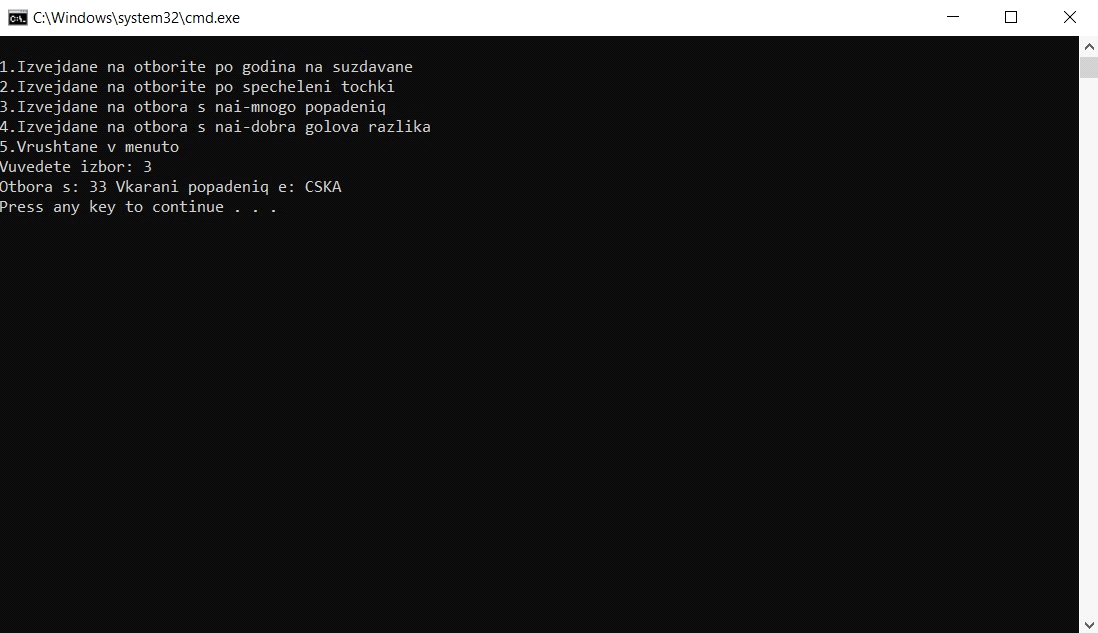
Sortirane\_Godina(o, counter)

Option 7.1 sorts the teams using Bubble Sort, ordered by their year of establishment.



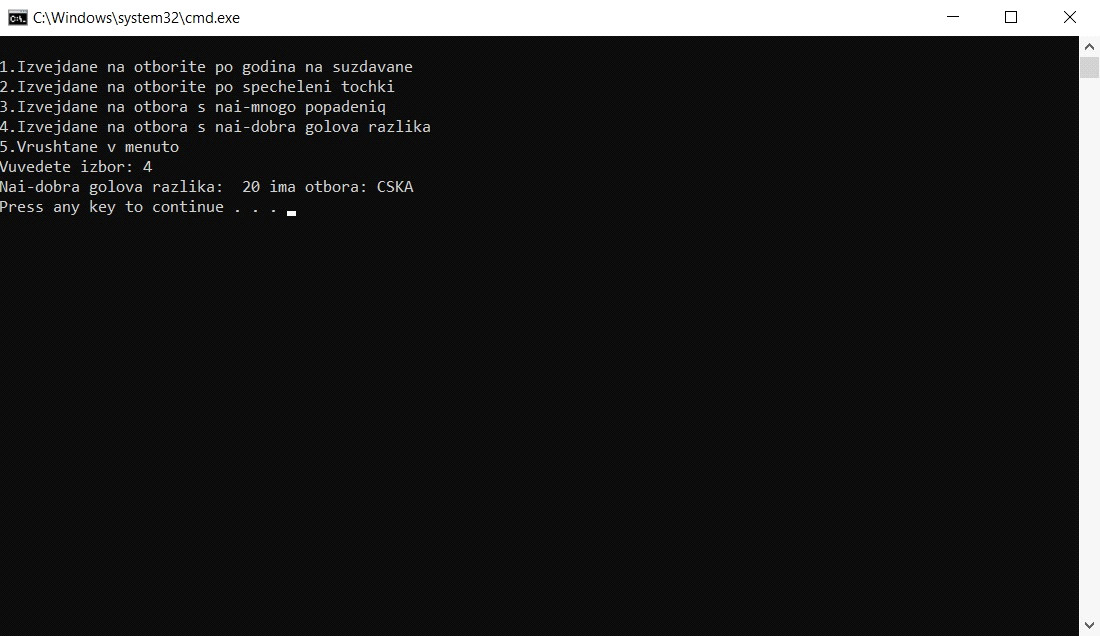
Sortirane\_Tochki(o, counter, m)

Option 7.2 calculates the points for the teams (awarding 3 points for a win, 1 point for a draw, and 0 points for a loss) and uses Bubble Sort to arrange them by points.



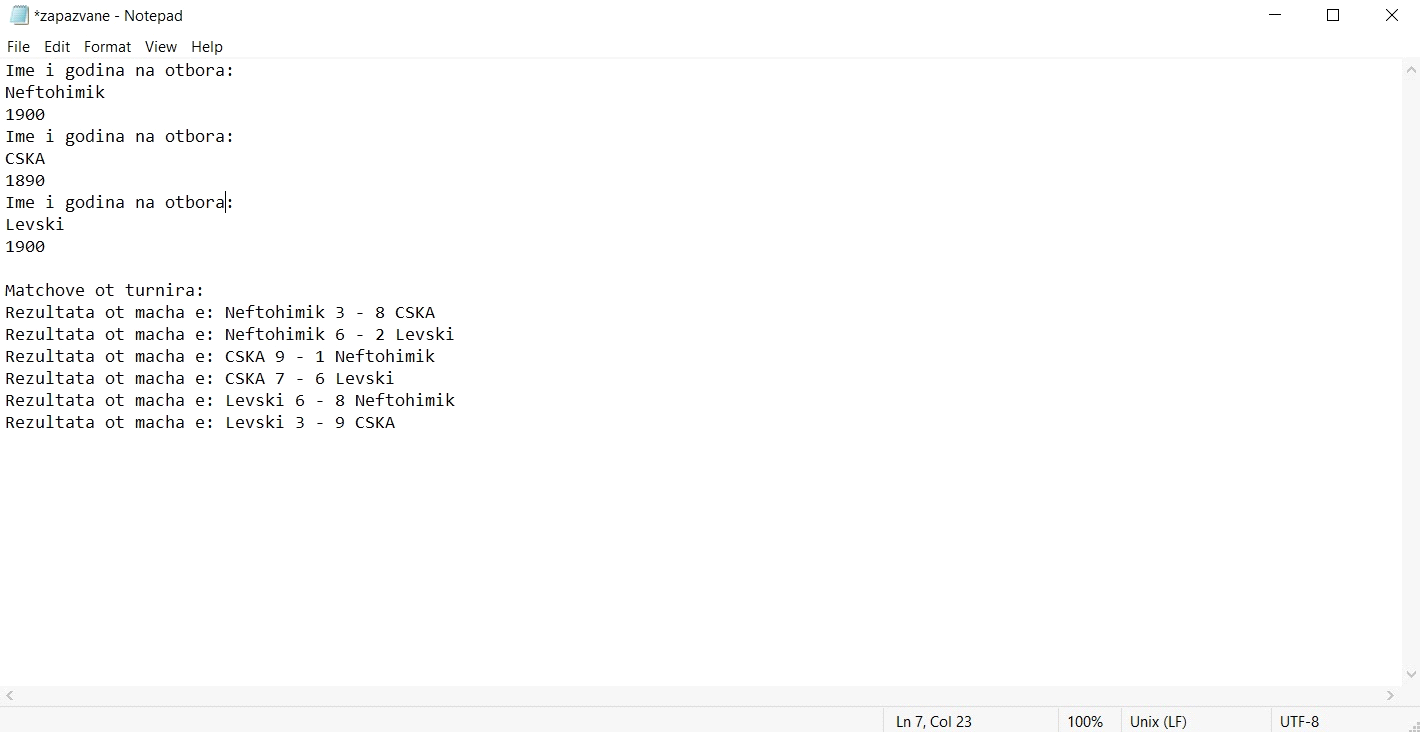
Izvejdane\_Max\_Popadeniq(o, m, counter)

Option 7.3 calculates all the goals scored by each team in the tournament and uses Bubble Sort to find the team with the most goals.



Izvejdane\_Max\_Razlika(o, m, counter)

Option 7.4 calculates the goal difference for each team (adding the goal difference for the winning team and subtracting it for the losing team) and displays the team with the best goal difference.



Save(o, m, counter)

Option 8 in the main menu saves all teams' information (Name and Year) and the matches from the tournament.

#include <iostream>

#include <iomanip>

#include <string>

#include<conio.h>

#include <random>

#include <cstdlib>

#include <ctime>

#include <fstream>

#include <time.h>

using namespace std;

const int N = 5;

int counter = 0;

struct otbor

{

char ime[20];

int godina;

};

otbor enter()

{

cin.ignore();

otbor o;

char input\_ime[20];

char input\_godina[20];

cout << "1.Vuvedete ime: ";

cin.getline(o.ime, 20, '\n');

cout << "2.Vuvedete godina na suzdavane: ";

cin >> o.godina;

cout << endl << "Uspeshno dobavihte otbora!";

cout << endl << "Ime: " << o.ime;

cout << endl << "Godina na suzdavane: " << o.godina;

return o;

};

struct match

{

int home;

int away;

};

void otbor\_input(otbor o[], int counter, int nomer\_list);

void otbor\_output(otbor o[]);

void turnir(match m[N][N], otbor o[], int counter);

void tursene\_s\_ime(match m[N][N], otbor o[], int counter);

void izpisvane\_na\_match(match m[N][N], otbor o[], int k, int i);

void tursene\_po\_dve\_imena(match m[N][N], otbor o[], int counter);

void Spravki\_podmenu(otbor o[], int counter, match m[N][N]);

void Sortirane\_Godina(otbor o[], int counter);

void Sortirane\_Tochki(otbor o[], int counter, match m[N][N]);

void Izvejdane\_Max\_Popadeniq(otbor o[], match m[N][N], int counter);

void Izvejdane\_Max\_Razlika(otbor o[], match m[N][N], int counter);

void Izkopirane\_Otbori(otbor o[], int counter, otbor o1[]);

void Save(otbor o[],match m[N][N], int counter);

int main()

{

otbor o[N];

int nomer\_list = 0;

char a = 'y';

int izbor;

match m[N][N];

do {

system("cls");

cout << "Futbolen Turnir: Menu";

cout << endl << "1.Dobavqne na edin otbor";

cout << endl << "2.Dobavqne na list ot otbori";

cout << endl << "3.Izvejdane na otborite";

cout << endl << "4.Purvenstvo";

cout << endl << "5.Tursene na izigrani machove na edin otbor";

cout << endl << "6.Tursene na izigrani machove mejdu dva otbora";

cout << endl << "7.Spravki za uchastnici";

cout << endl << "8.Zapazvane na dannite";

cout << endl << "9.Izlizane ot programata";

cout << endl << "Vuvedete izbor: ";

cin >> izbor;

while (izbor < 0 || izbor > 9) {

cout << "Molq vuvedete chislo ot 1 do 9!: ";

cin >> izbor;

}

switch (izbor)

{

case 1:

if (counter != N)

{

while (counter < N && (a == 'Y' || a == 'y')) {

o[counter] = enter();

counter++;

cout << endl << "Iskate li da produljite da vuvejdate?(Y/N): ";

do {

cin >> a;

} while ((a != 'N' && a != 'n') && (a != 'Y' && a != 'y'));

if (counter == N)

{

cout << "Dostignahte maximalniq broi otbori" << endl;

system("pause");

}

}

}

else cout << "Dostignahte maximalniq broi otbori" << endl;

a = 'y';

system("pause");

break;

case 2:

if (counter == N || nomer\_list == N)

{

cout << "Dostignahte maximalniq broi otbori" << endl;

system("pause");

}

else {

cout << endl << "Kolko otbora iskate da vuvedete: ";

cin >> nomer\_list;

while (counter + nomer\_list > N)

{

cout << "Molq vuvedete maximum 5 otbora: ";

cin >> nomer\_list;

}

otbor\_input(o, counter, nomer\_list);

counter = counter + nomer\_list;

}

break;

case 3:

otbor\_output(o);

system("pause");

break;

case 4:

if (counter < 3)

{

cout << "Molq purvo vuvedete pone 3 otbora" << endl;

}

else

{

turnir(m, o, counter);

}

system("pause");

break;

case 5:

tursene\_s\_ime(m, o, counter);

system("pause");

break;

case 6:

tursene\_po\_dve\_imena(m, o, counter);

system("pause");

break;

case 7:

Spravki\_podmenu(o, counter, m);

system("pause");

break;

case 8:

Save(o, m, counter);

break;

}

} while (izbor != 9);

return 0;

}

void otbor\_input(otbor o[], int counter, int nomer\_list)

{

do

{

cout << "Ime : ";

cin >> o[counter].ime;

cout << "Godina : ";

cin >> o[counter].godina;

nomer\_list--;

counter++;

} while (nomer\_list != 0);

}

void otbor\_output(otbor o[])

{

cout << "List s vsichki otbori" << endl;

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

{

cout << "Otbor: " << o[i].ime << endl;

cout << "Suzdaden prez:" << o[i].godina << endl << endl;

}

}

void turnir(match m[N][N], otbor o[], int counter)

{

srand(time(NULL));

for (int k = 0; k < counter; k++)

{

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

{

if (i != k)

{

m[k][i].home = rand() % 10 + 1;

m[k][i].away = rand() % 10 + 1;

izpisvane\_na\_match(m, o, k, i);

}

else

{

continue;

}

}

}

}

void tursene\_s\_ime(match m[N][N], otbor o[], int counter)

{

int otbor\_index = -1;

char ime[20];

cout << "Molq vuvedete ime na otbor: ";

cin >> ime;

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

{

if (!strcmp(ime, o[i].ime))

{

otbor\_index = i;

}

}

if (otbor\_index != -1)

{

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

if (otbor\_index != i)

{

izpisvane\_na\_match(m, o, otbor\_index, i);

izpisvane\_na\_match(m, o, i, otbor\_index);

cout << endl;

}

}

else

{

cout << "Tozi otbor ne sushtestvuva molq probvaite otnovo! ";

}

}

void izpisvane\_na\_match(match m[N][N], otbor o[], int k, int i)

{

if (k != i)

{

cout << "Rezultata ot macha e: " << o[k].ime << " " << m[k][i].home << " - " << m[k][i].away << " " << o[i].ime << endl;

}

}

void tursene\_po\_dve\_imena(match m[N][N], otbor o[], int counter)

{

int otbor\_index1 = -1;

int otbor\_index2 = -1;

char ime1[20];

char ime2[20];

cout << "Vuvedete imeto na purviq otbor: ";

cin >> ime1;

cout << endl << "Vuvedete imeto na vtoriq otbor: ";

cin >> ime2;

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

{

if (!strcmp(ime1, o[i].ime))

{

otbor\_index1 = i;

}

if (!strcmp(ime2, o[i].ime))

{

otbor\_index2 = i;

}

}

if (otbor\_index1 == -1 || otbor\_index2 == -1)

{

cout << "Pone edin ot dvata otbora ne sushtestvua probvaite otnovo!";

}

else

{

izpisvane\_na\_match(m, o, otbor\_index1, otbor\_index2);

izpisvane\_na\_match(m, o, otbor\_index2, otbor\_index1);

}

}

void Spravki\_podmenu(otbor o[], int counter, match m[N][N])

{

int podmenu\_izbor;

do{

system("cls");

cout << endl << "1.Izvejdane na otborite po godina na suzdavane";

cout << endl << "2.Izvejdane na otborite po specheleni tochki";

cout << endl << "3.Izvejdane na otbora s nai-mnogo popadeniq";

cout << endl << "4.Izvejdane na otbora s nai-dobra golova razlika";

cout << endl << "5.Vrushtane v menuto";

cout << endl << "Vuvedete izbor: ";

cin >> podmenu\_izbor;

while (podmenu\_izbor < 0 || podmenu\_izbor > 5)

{

cout << "Vuvedete chislo ot 1 do 5!: ";

cin >> podmenu\_izbor;

}

switch (podmenu\_izbor)

{

case 1:

Sortirane\_Godina(o, counter);

system("pause");

break;

case 2:

Sortirane\_Tochki(o, counter, m);

system("pause");

break;

case 3:

Izvejdane\_Max\_Popadeniq(o, m, counter);

system("pause");

break;

case 4:

Izvejdane\_Max\_Razlika(o, m, counter);

system("pause");

break;

}

} while (podmenu\_izbor != 5);

}

void Izkopirane\_Otbori(otbor o[], int counter, otbor o1[])

{

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

{

o1[i] = o[i];

}

}

void Sortirane\_Godina(otbor o[], int counter)

{

otbor buff;

otbor o1[N];

Izkopirane\_Otbori(o, counter, o1);

cout << "List na vsichki otbori podredeni po godina na suzdavane" << endl;

for (int k = counter; k > 1; k--)

{

for (int i = 0; i < k - 1; i++)

if (o1[i].godina < o1[i + 1].godina)

{

buff = o1[i];

o1[i] = o1[i + 1];

o1[i + 1] = buff;

}

}

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

cout << endl <<"Otbor:"<< o1[i].ime << endl<<"Suzdaden prez:" << o1[i].godina << endl;

}

void Sortirane\_Tochki(otbor o[], int counter, match m[N][N])

{

int tochki[N];

int buffer;

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

{

tochki[i] = 0;

}

otbor buff;

otbor o1[N];

Izkopirane\_Otbori(o, counter, o1);

cout << "List na vsichki otbori podredeni po tochki" << endl;

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

{

o1[i] = o[i];

}

for (int k = 0; k < counter; k++)

{

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

{

if (i != k)

{

if (m[k][i].home == m[k][i].away)

{

tochki[k] = tochki[k] + 1;

tochki[i] = tochki[i] + 1;

}

if (m[k][i].home > m[k][i].away)

{

tochki[k] = tochki[k] + 3;

}

if (m[k][i].home < m[k][i].away)

{

tochki[i] = tochki[i] + 3;

}

}

}

}

for (int k = counter; k > 1; k--)

{

for (int i = 0; i < k - 1; i++)

if (tochki[i] < tochki[i + 1])

{

buff = o1[i];

o1[i] = o1[i + 1];

o1[i + 1] = buff;

buffer = tochki[i];

tochki[i] = tochki[i + 1];

tochki[i + 1] = buffer;

}

}

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

cout << endl << "Otbor: " << o1[i].ime << endl << "Sus: " << tochki[i] << " tochki" << endl;

}

void Izvejdane\_Max\_Popadeniq(otbor o[], match m[N][N], int counter)

{

int golove[N];

int buffer;

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

{

golove[i] = 0;

}

otbor o1[N];

Izkopirane\_Otbori(o, counter, o1);

otbor buff;

//Sabira popadeniqta na vseki otbor

for (int k = 0; k < counter; k++)

{

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

{

if (i != k)

{

golove[k] = golove[k] + m[k][i].home;

golove[i] = golove[i] + m[k][i].away;

}

}

}

for (int k = counter; k > 1; k--)

{

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

{

if (golove[i] < golove[i + 1])

{

buffer = golove[i];

golove[i] = golove[i + 1];

golove[i + 1] = buffer;

buff = o1[i];

o1[i] = o1[i + 1];

o1[i + 1] = buff;

}

}

}

cout <<"Otbora s: "<<golove[0]<<" Vkarani popadeniq e: "<<o1[0].ime<<endl;

}

void Izvejdane\_Max\_Razlika(otbor o[], match m[N][N], int counter)

{

int razlika[N];

int buffer;

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

{

razlika[i] = 0;

}

otbor o1[N];

Izkopirane\_Otbori(o, counter, o1);

otbor buff;

for (int k = 0; k < counter; k++)

{

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

{

if (i != k)

{

razlika[k] = razlika[k] + (m[k][i].home - m[k][i].away);

razlika[i] = razlika[i] - (m[k][i].home - m[k][i].away);

}

}

}

for (int k = counter; k > 1; k--)

{

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

{

if (razlika[i] < razlika[i + 1])

{

buffer = razlika[i];

razlika[i] = razlika[i + 1];

razlika[i + 1] = buffer;

buff = o1[i];

o1[i] = o1[i + 1];

o1[i + 1] = buff;

}

}

}

cout << "Nai-dobra golova razlika: " << razlika[0] << " ima otbora: " << o1[0].ime<<endl;

}

void Save(otbor o[], match m[N][N], int counter)

{

fstream save;

save.open("zapazvane.dat", ios::binary | ios::out | ios::trunc);

int k = 0;

int i = 0;

if (save.fail())

{

cout << "Faila ne moje da se otvori" << endl;

system("pause");

}

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

{

save << "Imena i godini na otborite: "<<endl << o[i].ime << endl << o[i].godina<<endl;

}

save <<endl<< "Matchove ot turnira:"<<endl;

for (int k = 0; k < counter; k++)

{

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

{

if (k != i)

{

save << "Rezultata ot macha e: " << o[k].ime << " " << m[k][i].home << " - " << m[k][i].away << " " << o[i].ime << endl;

}

}

}

save.close();

}