Технически университет Варна  
  
Документация на курсова работа

ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ  
Катедра: Компютърни науки и технологии(КНТ)

Специалност: Компютърни системи и технологии(КСТ)

Дисциплина: Базово програмиране  
Група: 1Б  
Тема на курсовата работа: Футболен турнир

Изработил: Стефан Христов Димитров

Факултетен №: 19621416

Футболен турнир

Да се напише компютърна програма, реализираща информационна система, която поддържа футболно първенство (Номер на регистрация, Име на отбор, Година на основаване, списък с изиграни срещи). Програмата съхранява и обработва данни за участващите отбори и изиграните мачове. Участниците в първенството са от 3 до 5

Програмата трябва да съдържа:

A. Меню за избор на функциите в програмата.

B. Добавяне на участници в първенството.

a. Добавяне на един участник (Име и град);

b. Добавяне на списък от участници. Въвежда се цяло число **n**  и след него **n** на брой участници.

C. Извеждане на всички участници от първенството на екрана

D. Провеждане на първенството.

a. Всички отбори се срещат помежду си в разменени гостувания, на случаен принцип се генерира резултат от срещата и се записва в списък с изиграни срещи за двата участващи отбора;

b. Извежда се таблица с изиграните срещи и резултати.

E. Търсене на отбори в първенството:

a. Извеждане на отбора с всички изиграни мачове по въведено име на отбор;  
 b. Извеждане на резултатите от изиграните мачове за два отбора, по въведени имена;

F. Справки за участниците отбори с под меню

a. Извеждане на всички участници в подреден ред по:

I. Година на основаване на отбора;  
 ii. Спечелени точки.

b. Извеждане на отбора отбелязал най-много попадения;  
 c. Извеждане на отбора с най-добра голова разлика.

G. Данните в програмата да могат да се запазват във файл между две стартирания на програмата.  
 H. Допълнителни условия:  
 a. За точка B при добавянето на нов участник да се даде възможност за избор дали да се въведе нов или да прекъсне  
 въвеждането.  
 b. За точка C да се реализира извеждането на отборите в табличен вид (Име на отбор, вкарани голове, допуснати голове, точки и изиграни мачове с резултат).

c. За точка G да се записват данните в двоичен файл.  
  
  
**За реализирането на програмата са използвани: Различни видове библиотеки, структури, логически оператори, масиви, двуизмерен масив, цикли, низове, условни оператори, генератор със сийд от локално време, сортитране по метод на мехурчето, работа с файлове. За изпълнението на задачата е използван езикът C++.**

**Функции в програмата**

**otbor enter() -** Единично въвеждане на отбор.

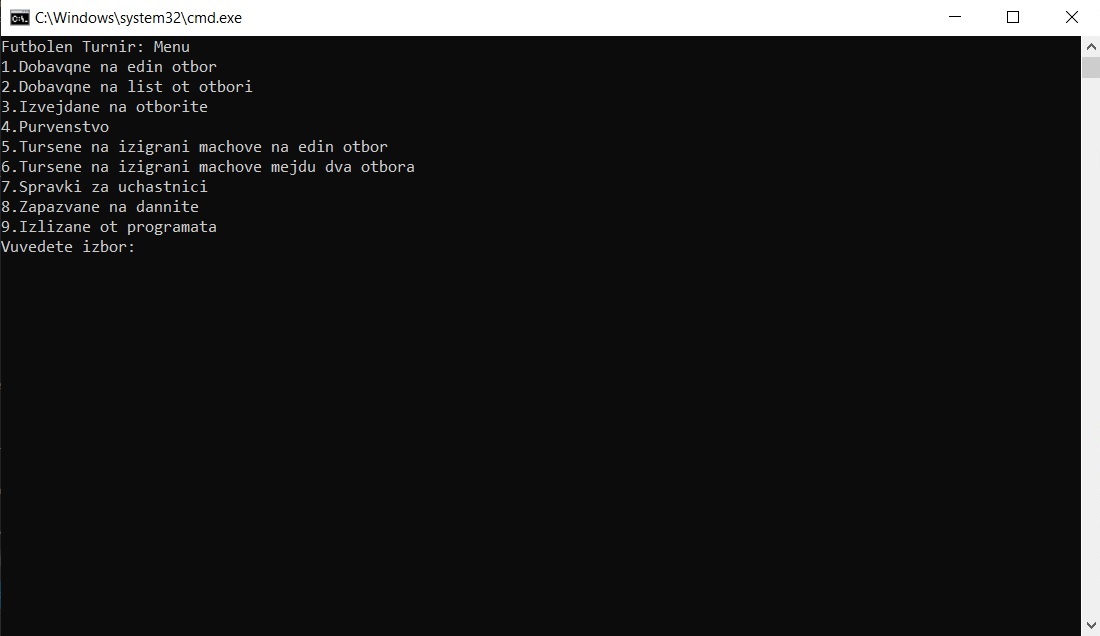
**void otbor\_input(otbor o[], int counter, int nomer\_list) -** Въвеждане на лист отбори.

**void otbor\_output(otbor o[]) -** Извеждане на всички въведени отбори.  
**void izpisvane\_na\_match(match m[N][N], otbor o[], int k, int i)-** Изписване на мачовете от турнира с произволен резултат.

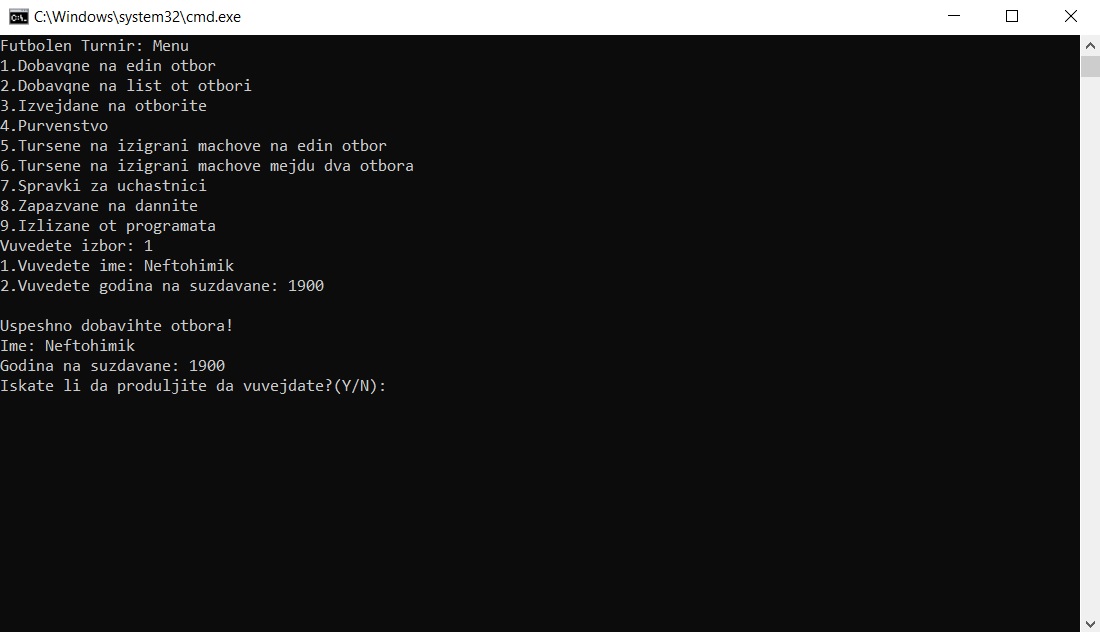
**void turnir(match m[N][N], otbor o[], int counter) –** Провеждане на футболния турнир

**void tursene\_s\_ime(match m[N][N], otbor o[], int counter) –** Изписване на всички изиграни мачове на отбор чрез въвеждане на име на съществуващ отбор  
**void tursene\_po\_dve\_imena(match m[N][N], otbor o[], int counter) –** Изписване на мачовете между два отбора чрез въвеждане на две имена на съществуващи отбори.  
**void Spravki\_Podmenu(otbor o[], int counter, match m[N][N]) –** Под меню за справки за участващите отбори.  
**void Sortirane\_Godina(otbor o[], int counter) –** Сортиране на въведените отбори по година на създаване.  
**void Sortirane\_Tochki(otbor o[], int counter, match m[N][N]) –** Сортиране на въведените отбори по спечелени точки от турнира.  
**void Izvejdane\_Max\_Popadeniq(otbor o[], match m[N][N], int counter) –** Извеждане на отбора с най-много попадения в турнира.  
**void Izvejdane\_Max\_Razlika(otbor o[], match m[N][N] –** Извеждане на отбора с най-добра голова разлика в турнира.  
**void Izkopirane\_Otbori(otbor o[], int counter, otbor o1[]) -** Изкопира данните на отборите ( Име и година) от o[] във o1[] за да не се разместят данните във o[] при сортиране по метод на мехурчето.

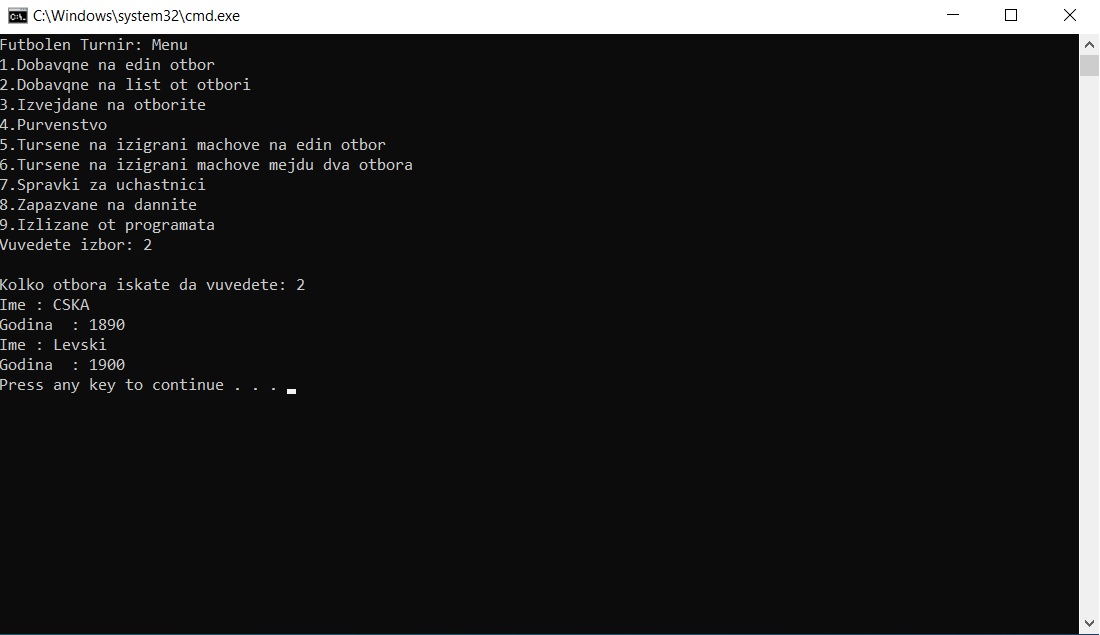
**void Save(otbor o[], match m[N][N], int counter) –** Запазва Името и годината на създаване на отборите и мачовете от последния турнир във двоичен файл, като изтрива старите данни във файла.

****

Начален изглед на програмата. Тук има switch case меню и два do while цикъла. Първият ни кара да въведем число от 1 до 9 като изисква на ново избор всеки път който се въведе число което не се намира в тези рамки. Вторият повтаря менюто докато не изберем опция 9 за изход.

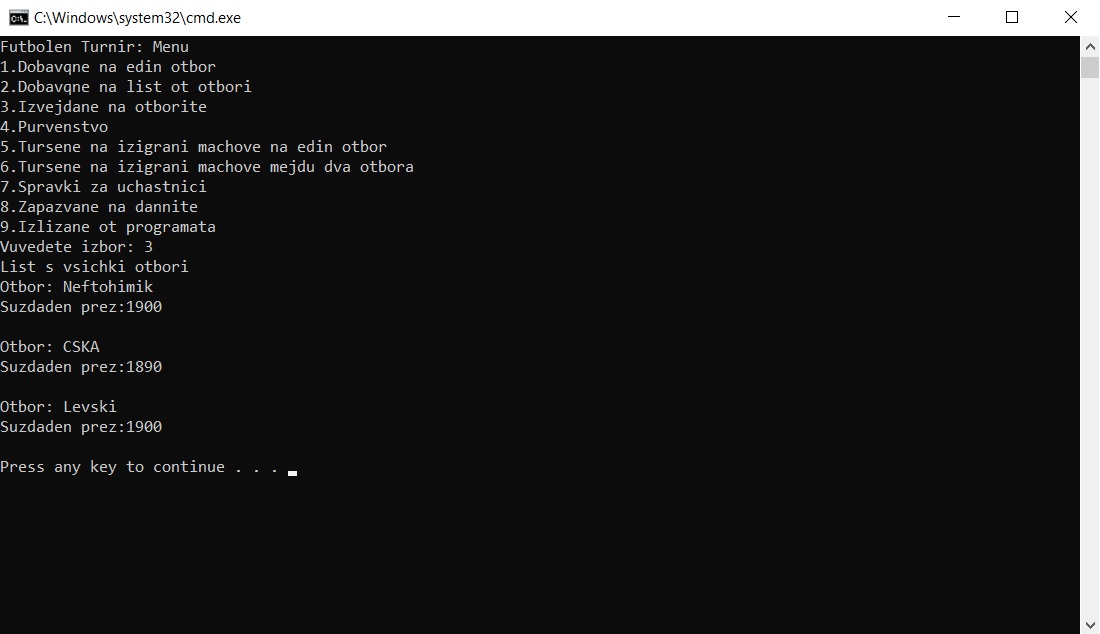
   
otbor enter()

Опция 1 ни кара да въведем 1 отбор и ако желаем можем да продължим да въвеждаме чрез избор y/Y- за да продължим или N/n за да спрем което ни връща в началното меню. Тук се въвеждат името и годината на създаване на отбора и не ни позволява да въведем повече от N(5) отбора.



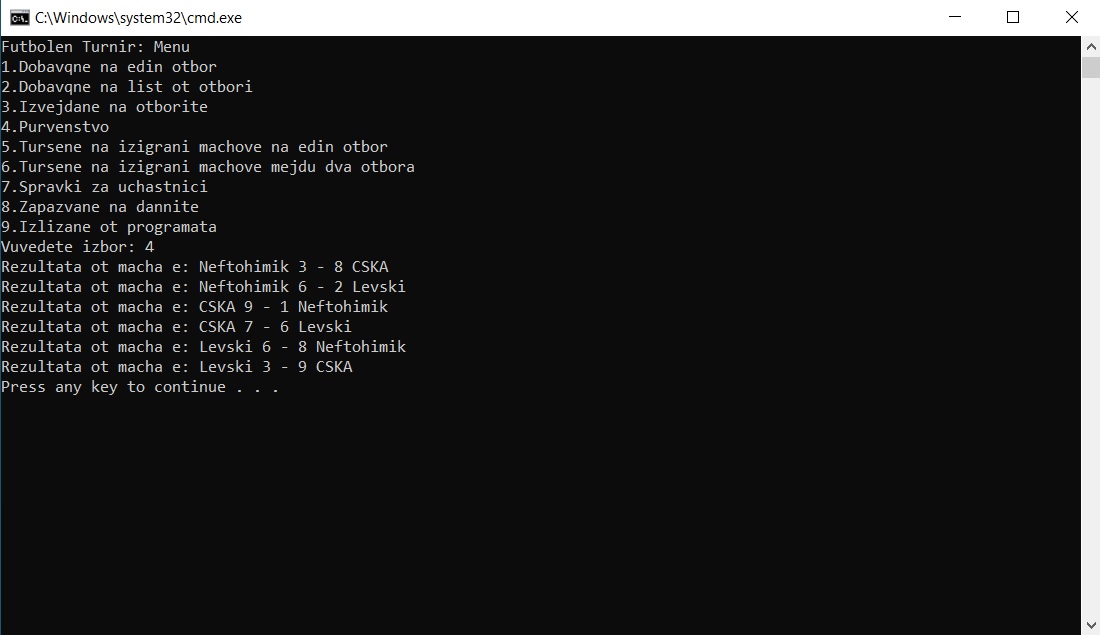
otbor\_output(o, counter, nomer\_list)

Опция 2 ни кара да въведем лист от отбори като първи ни кара да въведем броя на отборите които искаме да въведем и след това да добавим името и годината на основаване. Тя ни кара да се придържим до лимита на отбори (до 5) като преди да се включи първо проверява да ли сме стигнали 5 отбора и после проверява дали листа от отбори които искаме да въведем ще направи броя на общо въведените отбори по-голям от N(5).



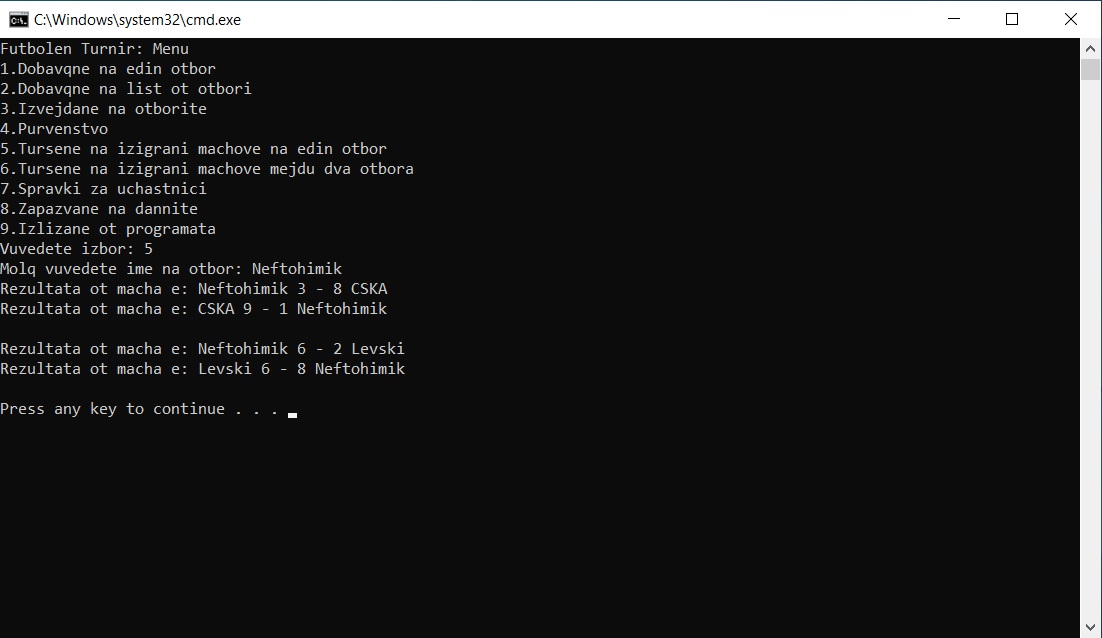
otbor\_output(o)

Опция 3 извежда всички отбори един след друг.



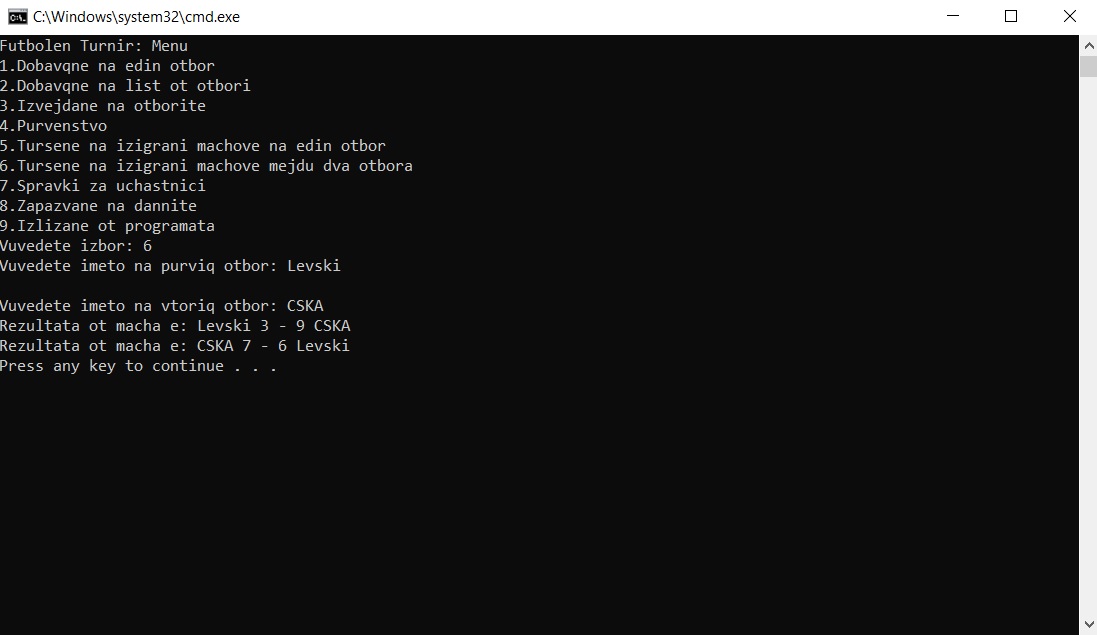
turnir(m, o, counter)

Опция 4 провежда турнира. Първо проверява дали има поне 3 записани отбора след което чрез двуизмерен масив кара всеки отбор да играе срещу всеки (чрез два for цикъла. А чрез if statement предотвратява отбора да играе срещу себе си) като изписва произволни попадения за двата отбора.

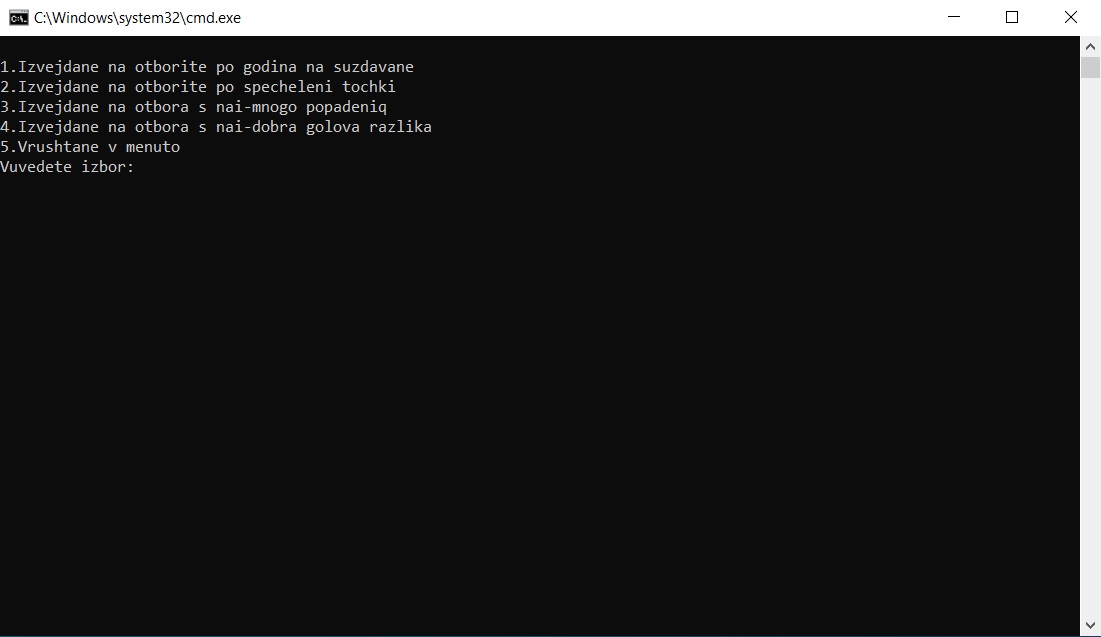


tursene\_s\_ime(m, o, counter)

Опция 5 ни кара да въведем име на отбор. Проверява дали въведения отбор съществува и изписва всичките мачове които той е изиграл в турнира. Като когато въведения отбор не съществува ни връща обратно в началното меню.

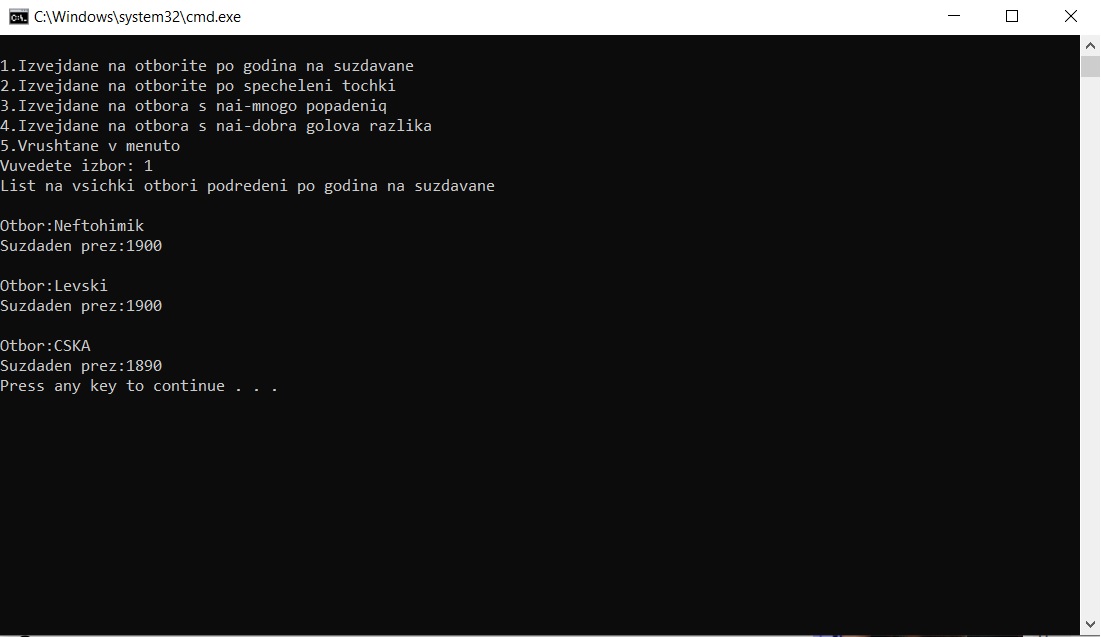
  
tursene\_po\_dve\_imena(m, o, counter)

Опция 6 ни кара да въведем имена на два отбора. Проверява дали и двата отбора съществуват и изписва мачовете между тези два отбора от турнира. Ако дори и един от въведените отбори не съществува ни връща в началното меню.



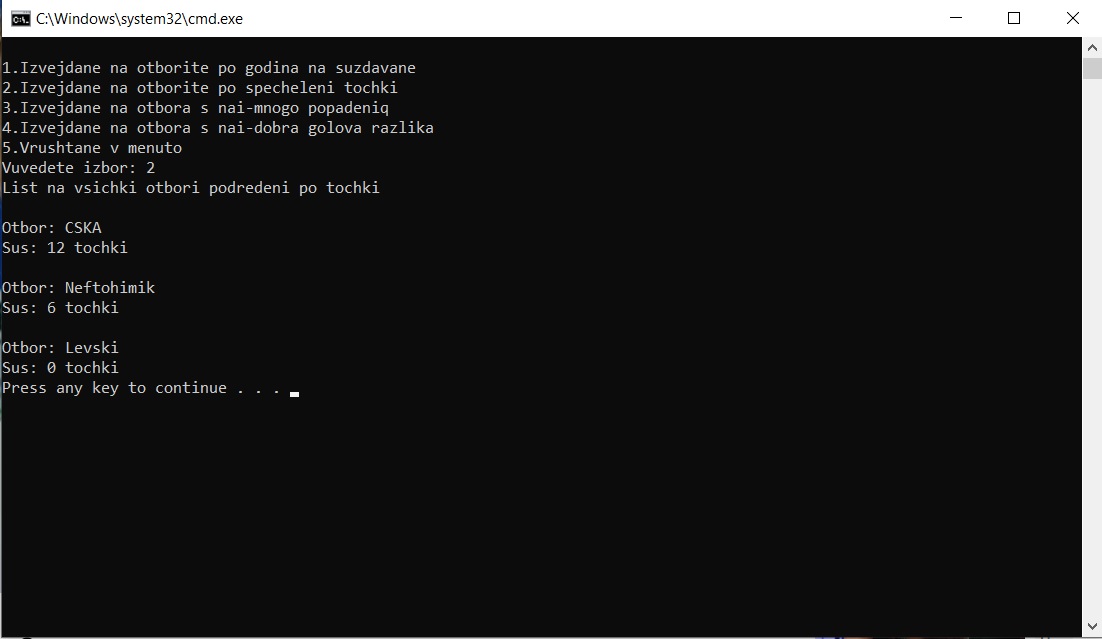
Spravki\_podmenu(o, counter, m)

Опция 7 е под меню. Като функция е сходна ката менюто в началото на програмата обаче за лимит има от 1 до 5. Като при избиране на опция 5 излиза от под менюто и се връща в главното. (За по-лесно разбиране на функциите в под менюто ще се наричат опция: 7.1, 7.2, 7.3 и 7.4)



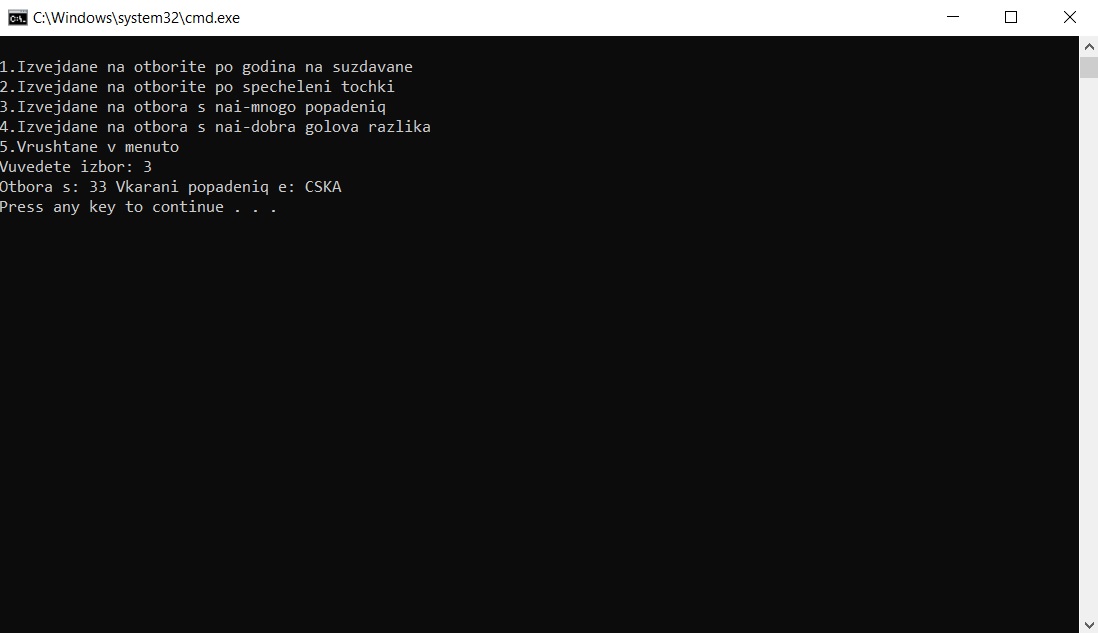
Sortirane\_Godina(o, counter)

Опция 7.1 чрез Bubble sort сортира отборите подредени по година на основаване.



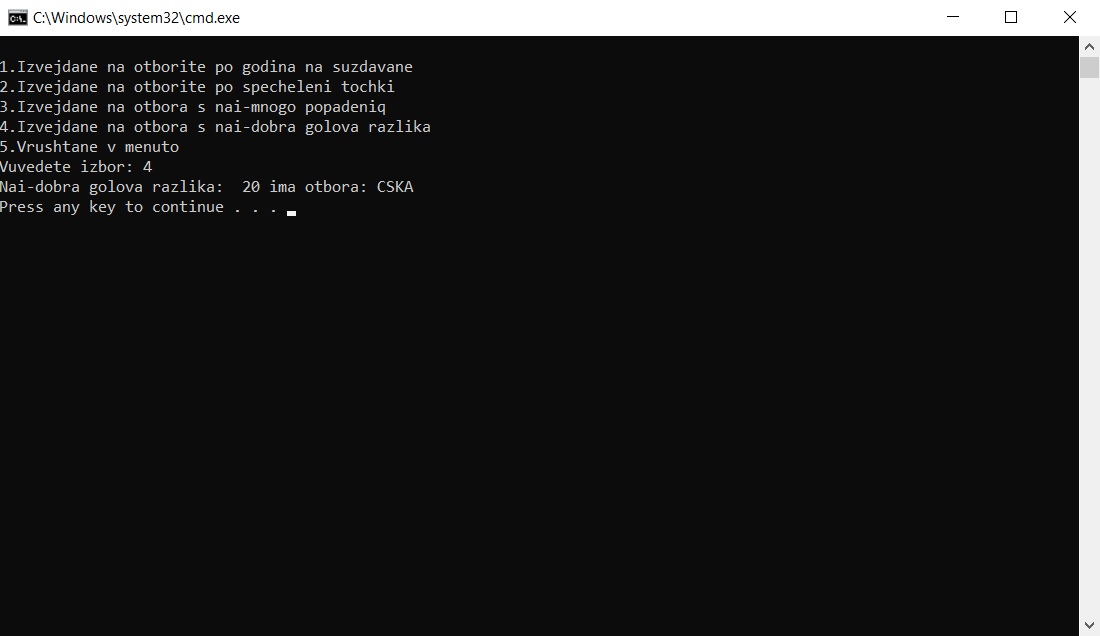
Sortirane\_Tochki(o, counter, m)

Опция 7.2 изчислява точките на отборите (дава им 3 точки на победа, 1 точка на равенство и 0 точки при загуба) и прави Bubble sort за да ги подреди по точки.



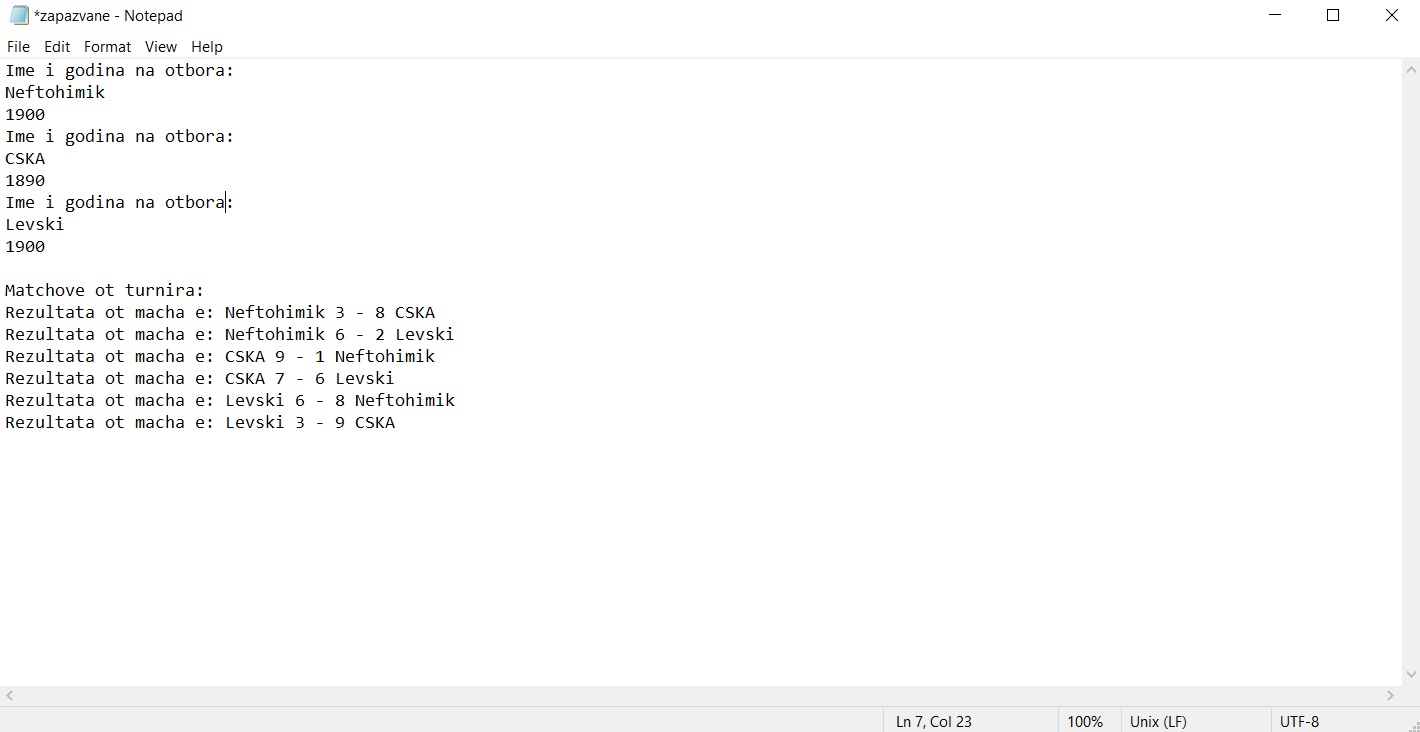
Izvejdane\_Max\_Popadeniq(o, m, counter)

Опция 7.3 изчислява всички попадения на всеки отбор в турнира и прави Bubble sort за да изкара отбора с най-много попадения



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

Опция 7.4 изчислява головата разлика на всеки отбор (прибавя разликата между головете на всеки мач на спечелилия отбор и я премахва от загубилия) и извежда отбора с най-добра голова разлика.



Save(o, m, counter)

Опция 8 от главното меню запазва информацията на отборите (Име и година) и мачовете от турнира.

#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();

}