Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „Софтуерно и Интернет Технологии“

**СЕМЕСТРИАЛНА ДОМАШНА РАБОТА**

по дисциплината „Базово програмиране”

на тема: „Футболно първенство”

Вариант: 47

|  |  |
| --- | --- |
| Изготвил: Християн Керкенезов | Проверил: |
| Специалност: Софтуерни и Интернет Технологии |  |
| Група: 2 |  |
| Факултетен номер: 22621617 |  |

2023

Съдържание

[I. Задание на проекта 3](#_Toc120821980)

[II. Анализ на решението 4](#_Toc120821981)

[1. Структура за данните в програмата 4](#_Toc120821982)

[2. Реализация на условие A 4](#_Toc120821983)

[3. Реализация на условие B 4](#_Toc120821984)

[4. Реализация на условие C 5](#_Toc120821985)

[5. Реализация на условие D 5](#_Toc120821986)

[6. Реализация на условие E 6](#_Toc120821987)

[7. Реализация на условие F 7](#_Toc120821988)

[8. Реализация на условие … - допълнение първо 7](#_Toc120821989)

[9. Реализация на условие … - допълнение второ 8](#_Toc120821990)

[10. Реализация на допълнение трето 8](#_Toc120821991)

[III. Упътване за употреба 9](#_Toc120821992)

[1. Впишете съответната част от проекта 9](#_Toc120821993)

[2. Впишете съответната част от проекта 9](#_Toc120821994)

[IV. Примерно действие на програмата 10](#_Toc120821995)

[1. Условие A 10](#_Toc120821996)

[2. Условие B 10](#_Toc120821997)

[3. Условие C 10](#_Toc120821998)

[4. Условие D 10](#_Toc120821999)

[5. Условие E 10](#_Toc120822000)

[6. Условие F 11](#_Toc120822001)

[7. Допълнение първо 11](#_Toc120822002)

[8. Допълнение второ 11](#_Toc120822003)

[9. Допълнение трето 11](#_Toc120822004)

Задание на проекта

Да се напише компютърна програма, реализираща информационна система, която поддържа футболно първенство (номер на регистрация, име на отбор, град, година на основаване, брой картотекирани играчи, брой спечелени първенства). Максималния брой различни отбори в първенството е 16.

Анализ на решението

Структура за данните в програмата

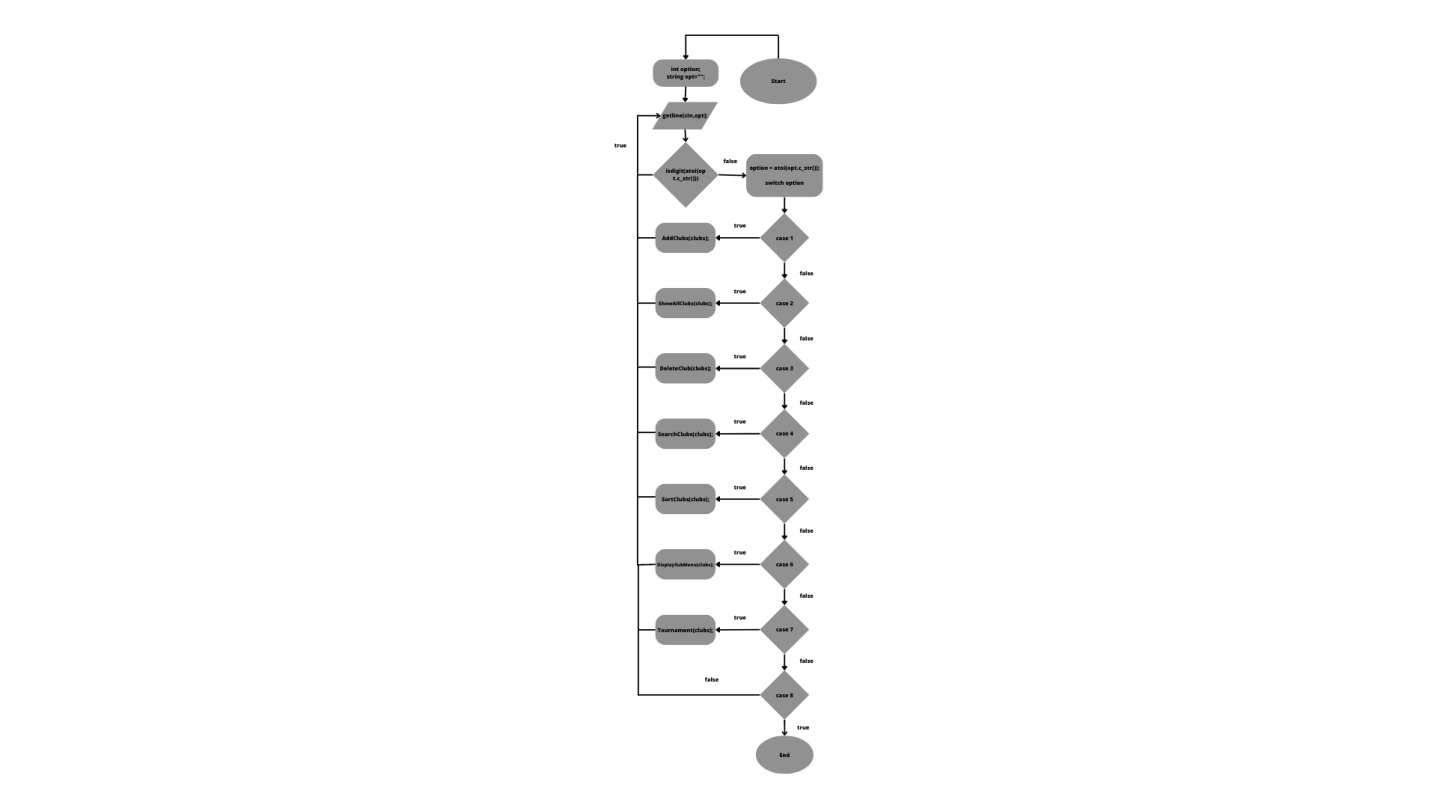
|  |  |  |
| --- | --- | --- |
| Структура | Обяснение | Примерени стойности |
| class FClub {  public:  char cName[50], city[50];  int registrationNumber = 0, year, numberOfPlayers;  int wins = 0, losses = 0, draws = 0;  }; | -cName – име на отбора  -city – град  -registration number – номер на регистрация  -year – година на основаване  -numberOfPlayers – брой картотекирани играчи  -wins – победи  -losses – загуби  -draws - равни игри | Име1  Град1  1  2003  16  1  1  1 |

Реализация на условие A

Анализ на алгоритъма, който трябва да се реализира

Меню за избор на функциите от програмата.

Блок схема на алгоритъма



Функция, с която е реализиран алгоритъма

void DisplayMenu(FClub clubs[]);//Func --A—

void DisplayMenu(FClub clubs[]) {

int option;

string opt = "";

do {

do {

cout << "Choose option:\n" <<

"1 - Add Football club\n" <<

"2 - Display all clubs\n" <<

"3 - Delete club\n" <<

"4 - Sort clubs\n" <<

"5 - Sub menu\n" <<

"6 - Organize tournament\n" <<

"7 - Leave\n";

getline(cin, opt);

} while (isdigit(atoi(opt.c\_str())));

option = atoi(opt.c\_str());

switch (option)

{

case 1:

AddClubs(clubs);

break;

case 2:

ShowAllClubs(clubs);

break;

case 3:

DeleteClub(clubs);

break;

case 4:

SortClubs(clubs);

break;

case 5:

DisplaySubMenu(clubs);

break;

case 6:

Reset();

Tournament(clubs);

break;

case 7:

exit(0);

break;

default:

cout << "Please enter valid number\n";

Delay();

Reset();

continue;

}

Delay();

cout << "\nPress any button to continue\n";

\_getch();

Reset();

} while (true);

}

#### Входни данни на функцията

FClub clubs[] – Това е масивът, в който се съдържат отборите.

#### Изходни данни на функцията или данни, които се извеждат

Тази функция играе ролята на мост между различните услуги, от които потребителят може да избира.

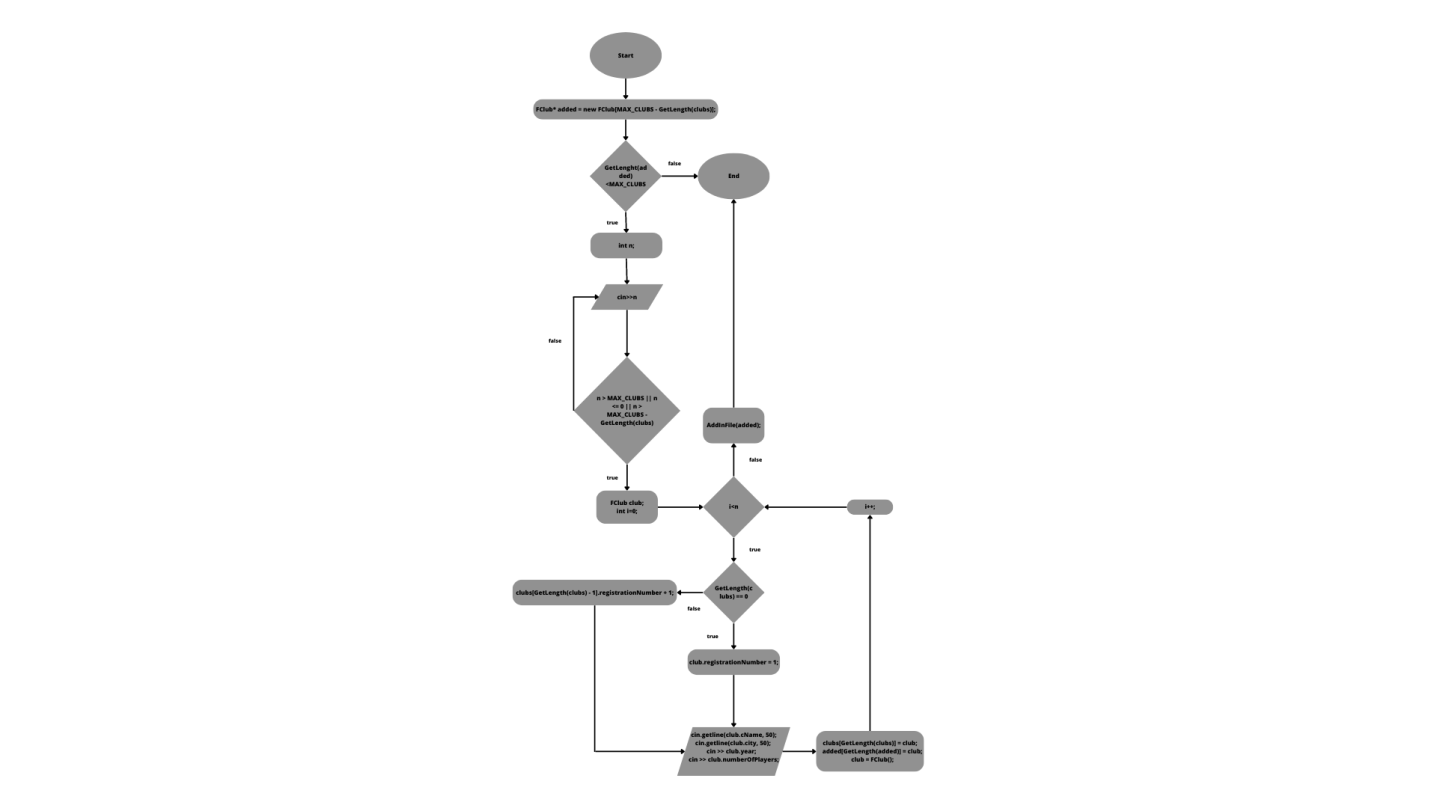
Реализация на условие B

Анализ на алгоритъма, който трябва да се реализира

Добавяне на отбор:

1. Трябва да позволява въвеждането на различен брой нови отбори, докато не достигнат максималния си брой в първенството (16).

Блок схема на алгоритъма



Функция, с която е реализиран алгоритъма

void AddClubs(FClub clubs[]);//Func --B—

bool CheckClubs(FClub clubs[]);

int CheckHowManyClubs(FClub clubs[]);

void AddClubs(FClub\* clubs) {

FClub\* added = new FClub[MAX\_CLUBS - CheckHowManyClubs(clubs)];

if (CheckHowManyClubs(clubs) < MAX\_CLUBS)

{

int n;

do {

Reset();

cout << "Type number of clubs you want to enter:";

cin >> n;

} while (n > MAX\_CLUBS || n <= 0 || n > MAX\_CLUBS - CheckHowManyClubs(clubs));

FClub club;

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

{

cin.ignore();

cout << "\nEnter the data for club " << i + 1 << endl;

if (CheckHowManyClubs(clubs) == 0)

club.registrationNumber = 1;

else club.registrationNumber = clubs[CheckHowManyClubs(clubs) - 1].registrationNumber + 1; //NUMBER

cout << "\nEnter the club name: "; cin.getline(club.cName, 50); //NAME

cout << "Enter the club's city: "; cin.getline(club.city, 50); //CITY

cout << "Enter when the club has been created(year): "; cin >> club.year; //YEAR

cout << "Enter the number of players: "; cin >> club.numberOfPlayers; //PLAYERS

clubs[CheckHowManyClubs(clubs)] = club;

added[CheckHowManyClubs(added)] = club;

club = FClub();

}

AddInFile(added);

}

else

{

cout << "The max number of clubs is " << MAX\_CLUBS << endl;

}

}

bool CheckClubs(FClub clubs[]) {

if (clubs[0].registrationNumber != 0)

return true;

return false;

}

int CheckHowManyClubs(FClub clubs[]) {

int count = 0;

for (int i = 0; i < MAX\_CLUBS; i++) {

if (!CheckClubs(clubs) || clubs[i].registrationNumber == 0) {

return count;

}

count++;

}

}

#### Входни данни на функцията

AddClubs() – CheckClubs() - CheckHowManyClubs() – FClub clubs[] – Това е масивът, в който се съдържат отборите.

#### Изходни данни на функцията или данни, които се извеждат

Void AddClubs() – Служи за добавяне на отбори(1-16 наведнъж) в масива.

Bool CheckClubs() – Определя дали масивът е празен чрез номера на регистрация(връща true ако регистрационният номер на първия клуб е 0 и false в противен случай)

Int CheckHowManyClubs() – Връща броят на отборите в масива

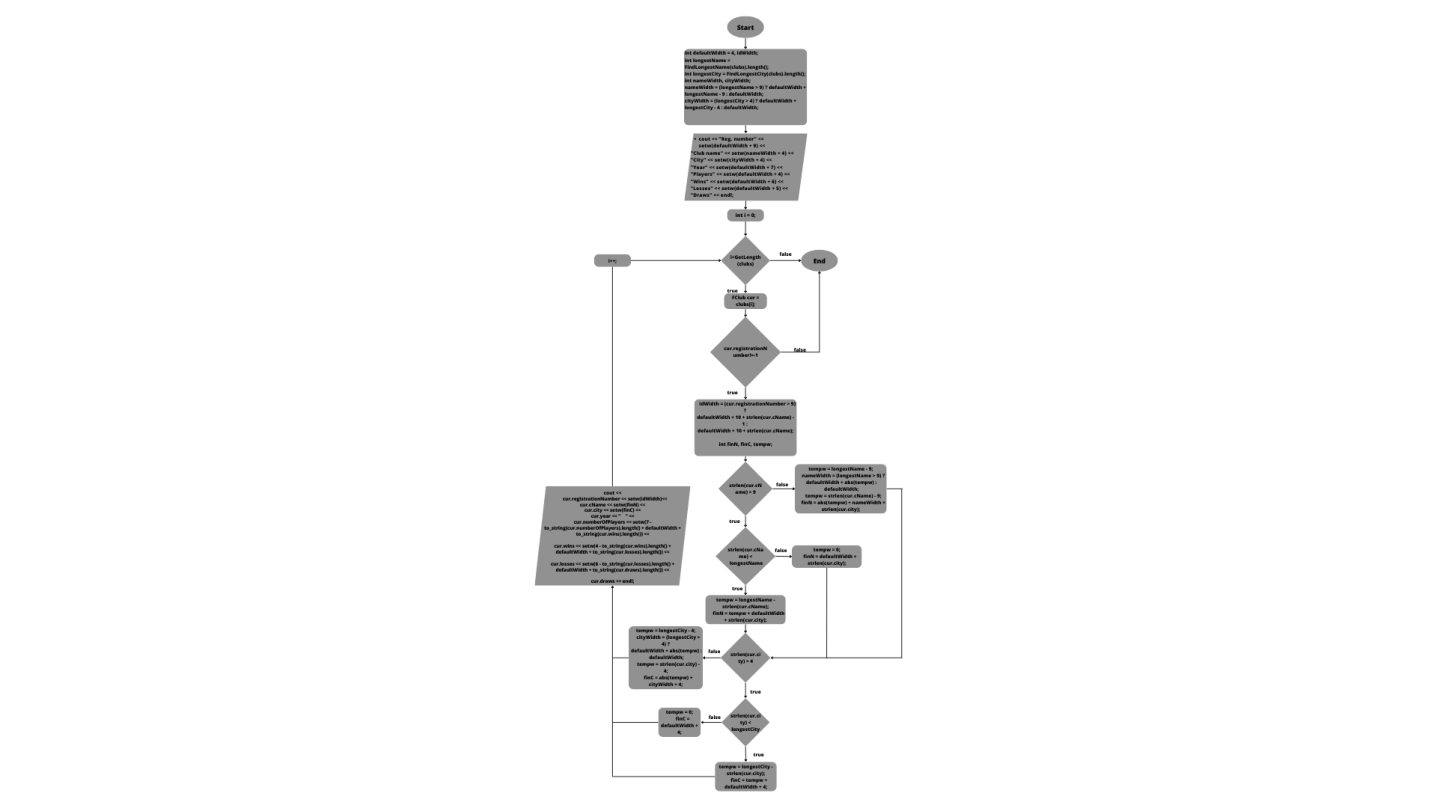
Реализация на условие C

Анализ на алгоритъма, който трябва да се реализира

Извеждане на екрана

1. Извеждане на всички отбори в оформен вид

Блок схема на алгоритъма



Функция, с която е реализиран алгоритъма

void ShowAllClubs(FClub clubs[]);//Func --C--

string FindLongestName(FClub clubs[]);

string FindLongestCity(FClub clubs[]);

string ConvertCharToString(char a[], int lenght);

void ShowAllClubs(FClub clubs[]) {

Reset();

int defaultWidth = 4, idWidth;

int longestName = FindLongestName(clubs).length();

int longestCity = FindLongestCity(clubs).length();

int nameWidth, cityWidth;

nameWidth = (longestName > 9) ? defaultWidth + longestName - 9 : defaultWidth;

cityWidth = (longestCity > 4) ? defaultWidth + longestCity - 4 : defaultWidth;

cout << "Reg. number" << setw(defaultWidth + 9) << "Club name" << setw(nameWidth + 4) << "City" << setw(cityWidth + 4) << "Year" << setw(defaultWidth + 7) << "Players" << setw(defaultWidth + 4) << "Wins" << setw(defaultWidth + 6) << "Losses" << setw(defaultWidth + 5) << "Draws" << endl;

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

FClub cur = clubs[i];

if (cur.registrationNumber != -1)

{

idWidth = (cur.registrationNumber > 9) ? defaultWidth + 10 + strlen(cur.cName) - 1 : defaultWidth + 10 + strlen(cur.cName);

int finN, finC, tempw;

if (strlen(cur.cName) > 9)

{

if (strlen(cur.cName) < longestName)

{

tempw = longestName - strlen(cur.cName);

finN = tempw + defaultWidth + strlen(cur.city);

}

else

{

tempw = 0;

finN = defaultWidth + strlen(cur.city);

}

}

else {

tempw = longestName - 9;

nameWidth = (longestName > 9) ? defaultWidth + abs(tempw) : defaultWidth;

tempw = strlen(cur.cName) - 9;

finN = abs(tempw) + nameWidth + strlen(cur.city);

}

if (strlen(cur.city) > 4)

{

if (strlen(cur.city) < longestCity)

{

tempw = longestCity - strlen(cur.city);

finC = tempw + defaultWidth + 4;

}

else

{

tempw = 0;

finC = defaultWidth + 4;

}

}

else {

tempw = longestCity - 4;

cityWidth = (longestCity > 4) ? defaultWidth + abs(tempw) : defaultWidth;

tempw = strlen(cur.city) - 4;

finC = abs(tempw) + cityWidth + 4;

}

cout <<

cur.registrationNumber << setw(idWidth) <<

cur.cName << setw(finN) <<

cur.city << setw(finC) <<

cur.year << " " <<

cur.numberOfPlayers << setw(7 - to\_string(cur.numberOfPlayers).length() + defaultWidth + to\_string(cur.wins).length()) <<

cur.wins << setw(4 - to\_string(cur.wins).length() + defaultWidth + to\_string(cur.losses).length()) <<

cur.losses << setw(6 - to\_string(cur.losses).length() + defaultWidth + to\_string(cur.draws).length()) <<

cur.draws << endl;

}

}

}

string FindLongestName(FClub clubs[]) {

int k = 0;

for (int i = 1; i < CheckHowManyClubs(clubs); i++)

{

if (strlen(clubs[k].cName) < strlen(clubs[i].cName))

{

k = i;

}

}

return ConvertCharToString(clubs[k].cName, strlen(clubs[k].cName));

}

string FindLongestCity(FClub clubs[]) {

int k = 0;

for (int i = 1; i < CheckHowManyClubs(clubs); i++)

{

if (strlen(clubs[k].city) < strlen(clubs[i].city))

{

k = i;

}

}

return ConvertCharToString(clubs[k].city, strlen(clubs[k].city));

}

string ConvertCharToString(char a[], int length) {

string s = "";

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

{

s += a[i];

}

return s;

}

#### Входни данни на функцията

ShowAllClubs() – FindLongestCity() – FindLongestName() - FClub clubs[] – Това е масивът, в който се съдържат отборите.

ConvertCharToString() –

char a[] – масивът, който ще се преобразува

int length – броят на символи в масива

#### Изходни данни на функцията или данни, които се извеждат

Void ShowAllClubs() –извежда информацията за всички отбори

String FindLongestCity() – връща града с най голям брой символи от масива за отбори

String FindLongestName() – връща името на отбора с най голям брой символи от масива за отбори

String ConvertCharToString() – превръща масив от символи в тип string

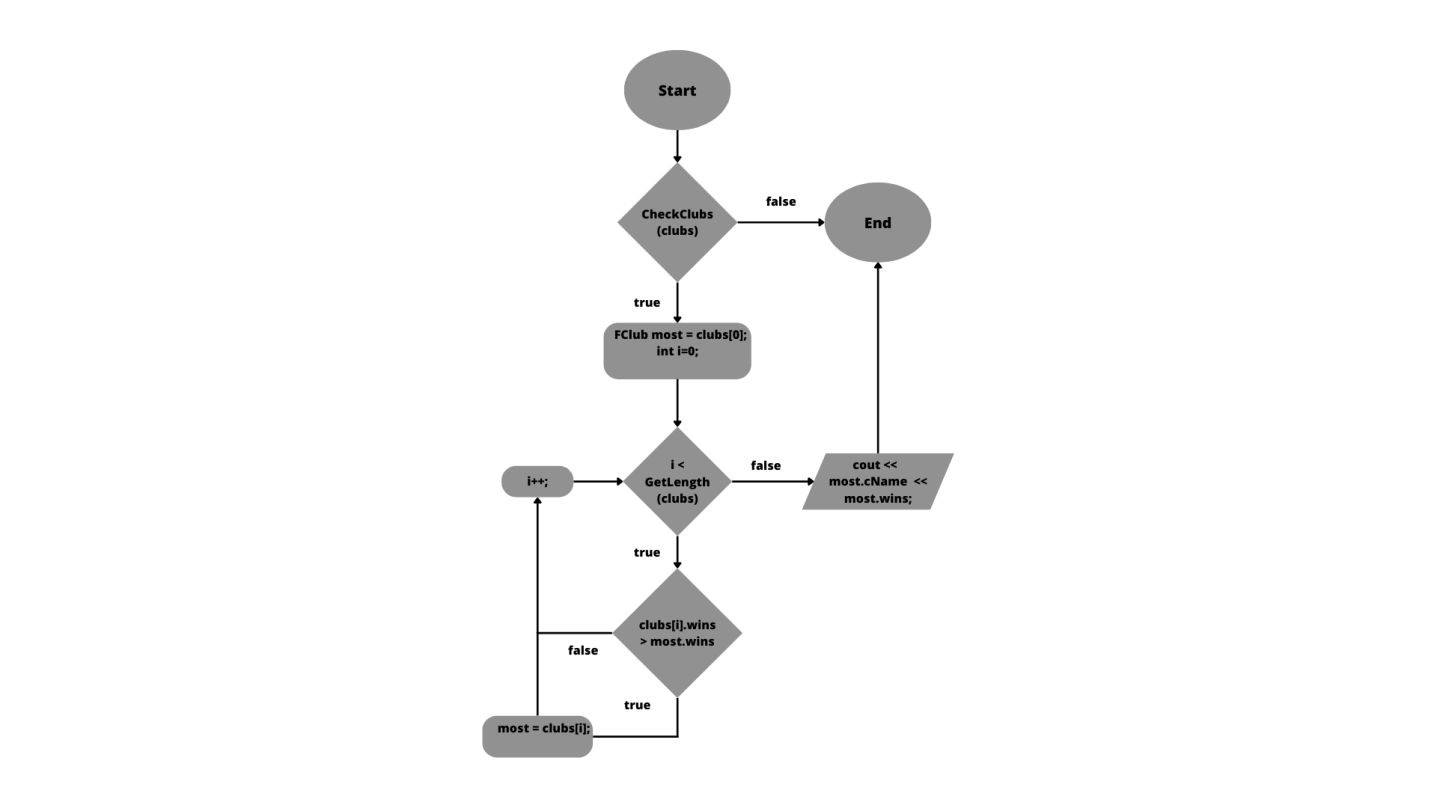
Реализация на условие D

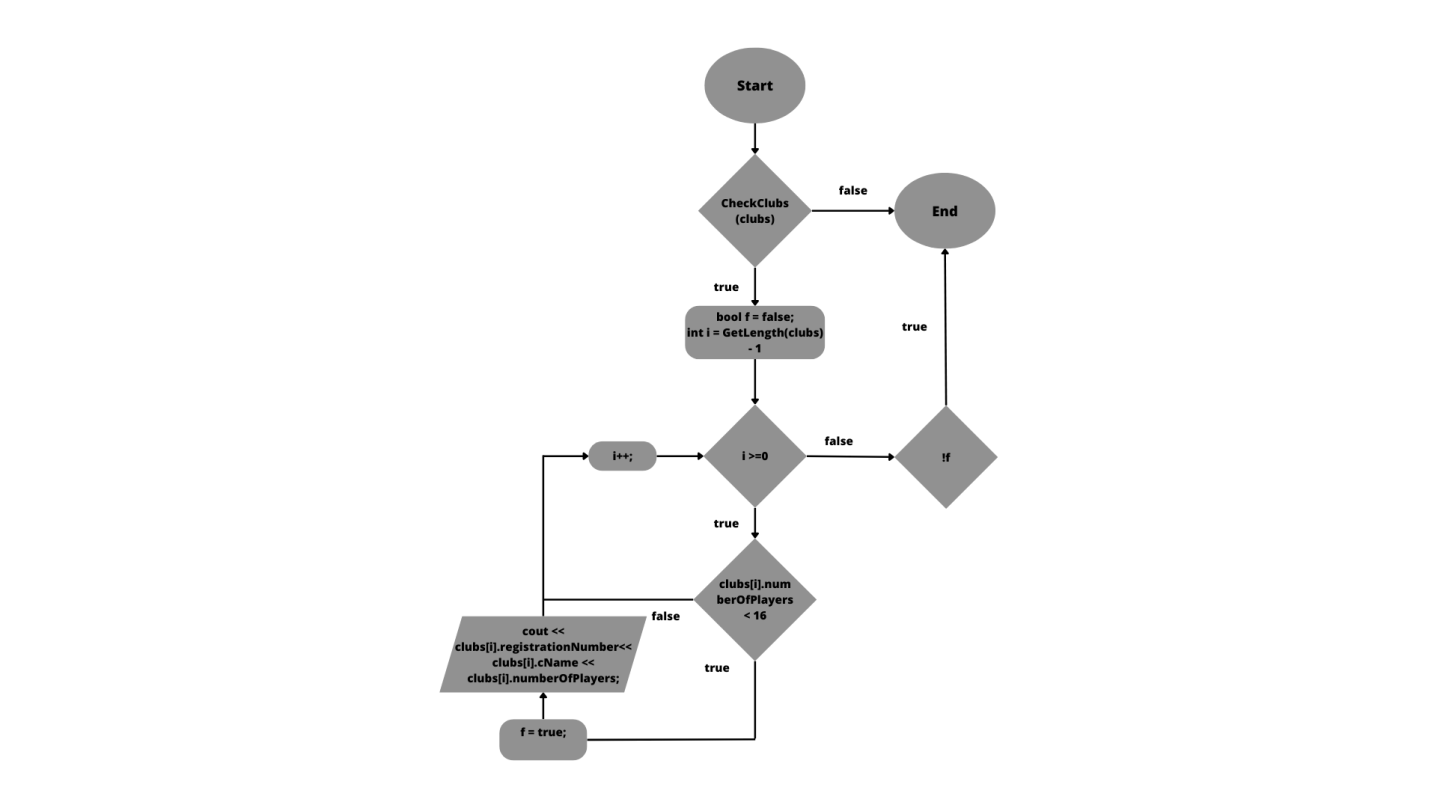
Анализ на алгоритъма, който трябва да се реализира

Търсене и извеждане на екрана:

1. Намиране и извеждане на отбора с най-голям брой спечелени първенства
2. Търсене на всички отбори, чийто брой картотекирани играчи е под 16.

Блок схема на алгоритъма





Функция с която е реализиран алгоритъма

void GetClubWithMostWins(FClub clubs[]);//Func --D--(a)

void GetClubsWithLeastPlayers(FClub clubs[]);//Func --D--(b)

void GetClubWithMostWins(FClub clubs[]) {

if (CheckClubs(clubs))

{

FClub most = clubs[0];

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

if (clubs[i].wins > most.wins)

{

most = clubs[i];

}

}

cout << "The club with the most wins is: " << most.cName << " with " << most.wins << " wins";

}

else cout << "There is no clubs added.";

}

void GetClubsWithLeastPlayers(FClub clubs[]) {

if (CheckClubs(clubs))

{

bool f = false;

for (int i = CheckHowManyClubs(clubs) - 1; i >= 0; i--)

{

if (clubs[i].numberOfPlayers < 16)

{

f = true;

cout << "Reg. number: " << clubs[i].registrationNumber << setw(4 + 9) << "Club name: " << clubs[i].cName << setw(4 + 7) << "Players: " << clubs[i].numberOfPlayers << endl;

}

}

if (!f)

{

cout << "There are no clubs with players less than 16";

return;

}

}

else cout << "There is no clubs added.";

}

#### Входни данни на функцията

GetClubWithMostWins() - GetClubsWithLeastPlayers() - FClub clubs[] – Това е масивът, в който се съдържат отборите.

#### Изходни данни на функцията или данни, които се извеждат

void GetClubWithMostWins() – Показва името на отбора с най много победи и колко са

void GetClubsWithLeastPlayers() – Извежда списък с отборите, които имат брой играчи по-малко от 16

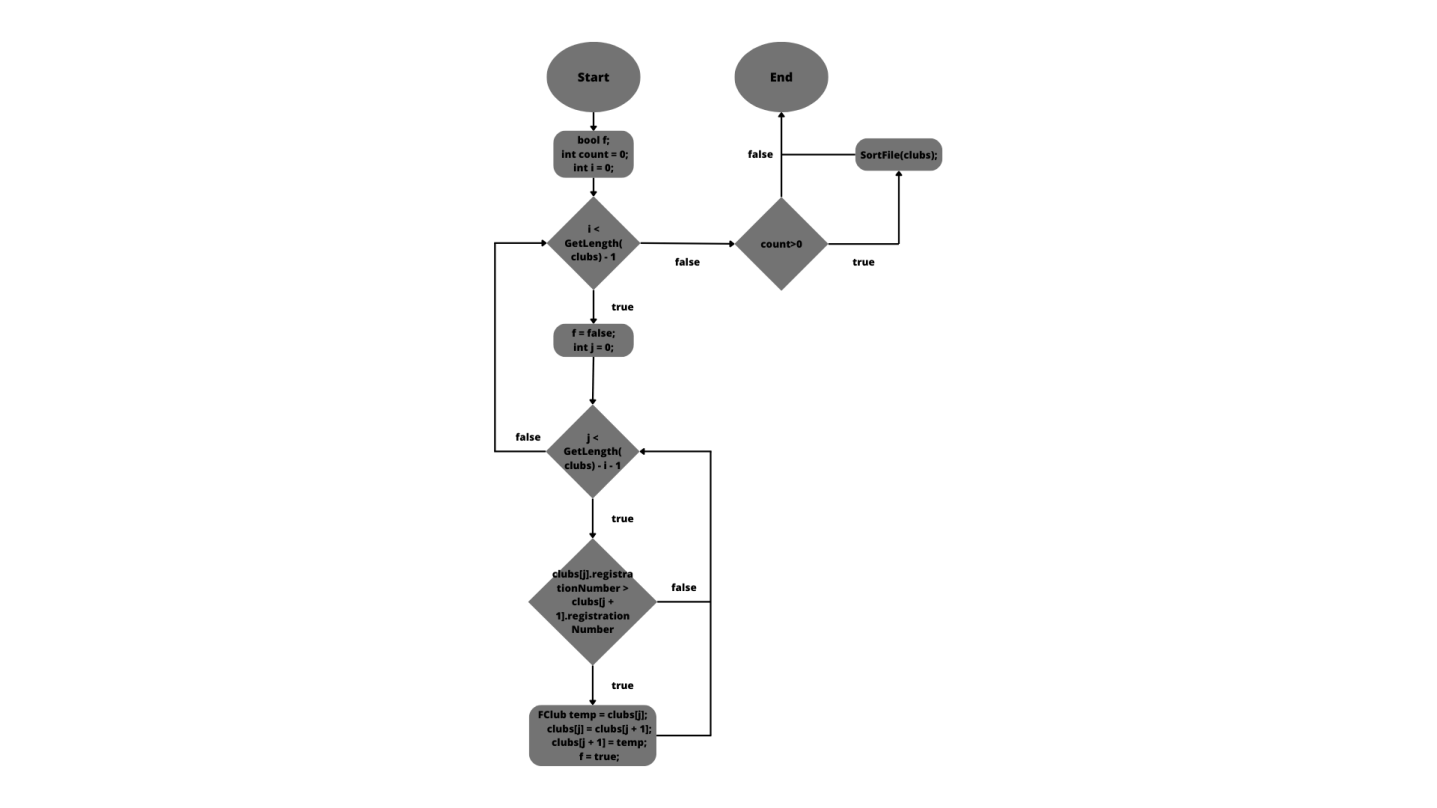
Реализация на условие E

Анализ на алгоритъма, който трябва да се реализира

Подреждане на основния масив с отбори, без да се извежда на екрана.

1. Подреждане на отборите в нарастващ ред на номера на регистрация

Блок схема на алгоритъма



Функция с която е реализиран алгоритъма

void SortClubs(FClub clubs[]);//Func --E--(a)

void SortClubs(FClub clubs[]) {

bool f;

int count = 0;

int option;

string opt = "";

do {

Reset();

cout << "Choose option:\n" <<

"1 - Sort by registration number\n" <<

"2 - Sort by year of creation\n" <<

"3 - Sort by number of tabulated players\n" <<

"4 - Back\n";

getline(cin, opt);

} while (isdigit(atoi(opt.c\_str())));

option = atoi(opt.c\_str());

switch (option) {

case 1:

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

{

f = false;

for (int j = 0; j < CheckHowManyClubs(clubs) - i - 1; j++)

{

if (clubs[j].registrationNumber > clubs[j + 1].registrationNumber)

{

FClub temp = clubs[j];

clubs[j] = clubs[j + 1];

clubs[j + 1] = temp;

f = true;

}

}

if (!f) break;

count++;

}

if (count > 0) {

SortFile(clubs);

}

break;

case 2:

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

{

f = false;

for (int j = 0; j < CheckHowManyClubs(clubs) - i - 1; j++)

{

if (clubs[j].year > clubs[j + 1].year)

{

FClub temp = clubs[j];

clubs[j] = clubs[j + 1];

clubs[j + 1] = temp;

f = true;

}

}

if (!f) break;

count++;

}

if (count > 0) {

SortFile(clubs);

}

break;

case 3:

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

{

f = false;

for (int j = 0; j < CheckHowManyClubs(clubs) - i - 1; j++)

{

if (clubs[j].numberOfPlayers < clubs[j + 1].numberOfPlayers)

{

FClub temp = clubs[j];

clubs[j] = clubs[j + 1];

clubs[j + 1] = temp;

f = true;

}

}

if (!f) break;

count++;

}

if (count > 0) {

SortFile(clubs);

}

break;

break;

case 4:

return;

}

}

#### Входни данни на функцията

SortClubs() - FClub clubs[] – Това е масивът, в който се съдържат отборите.

#### Изходни данни на функцията или данни, които се извеждат

Void SortClubs() – Сортира масивът с отбори(без да го извежда на екрана) по регистрационен номер, година на създаване и брой картотекирани играчи

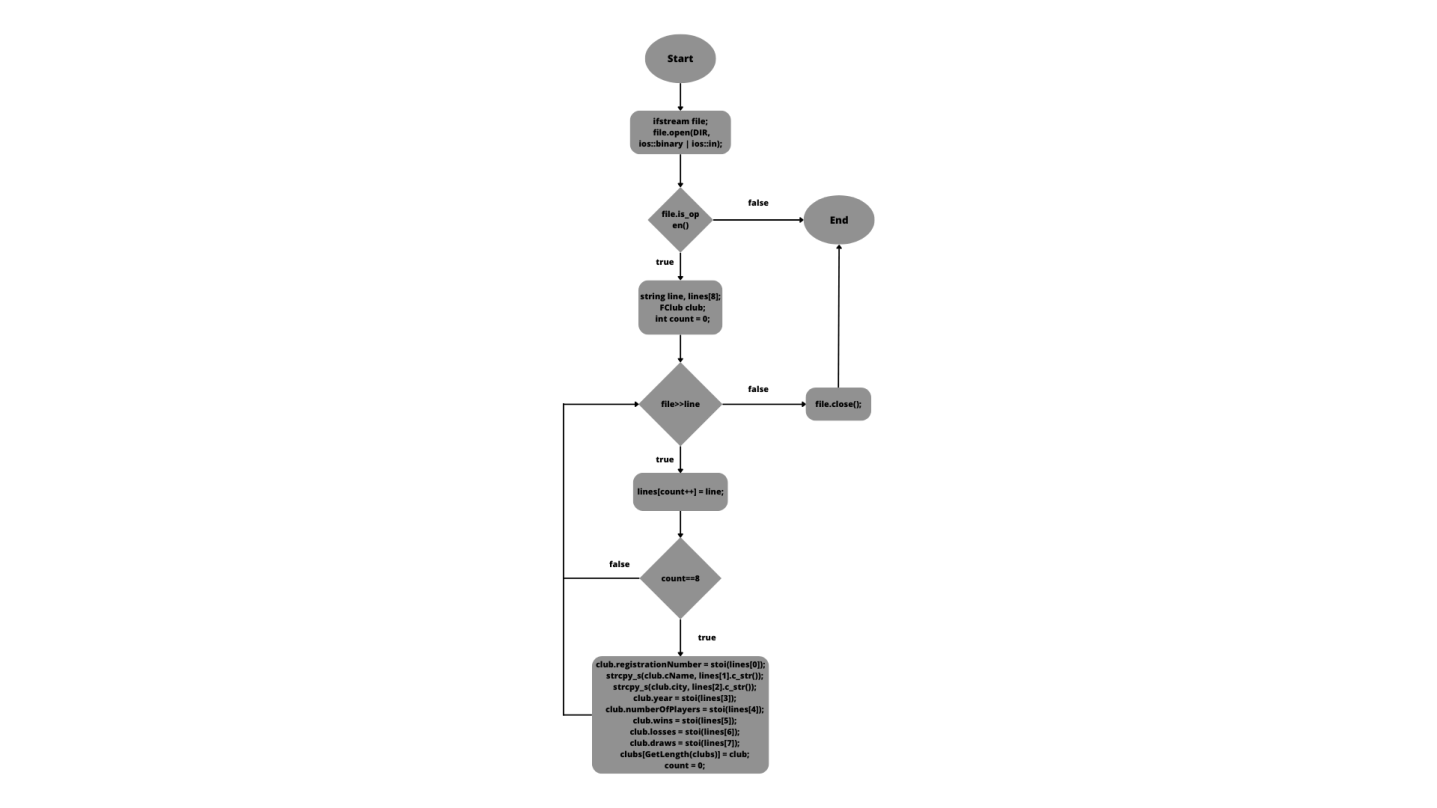
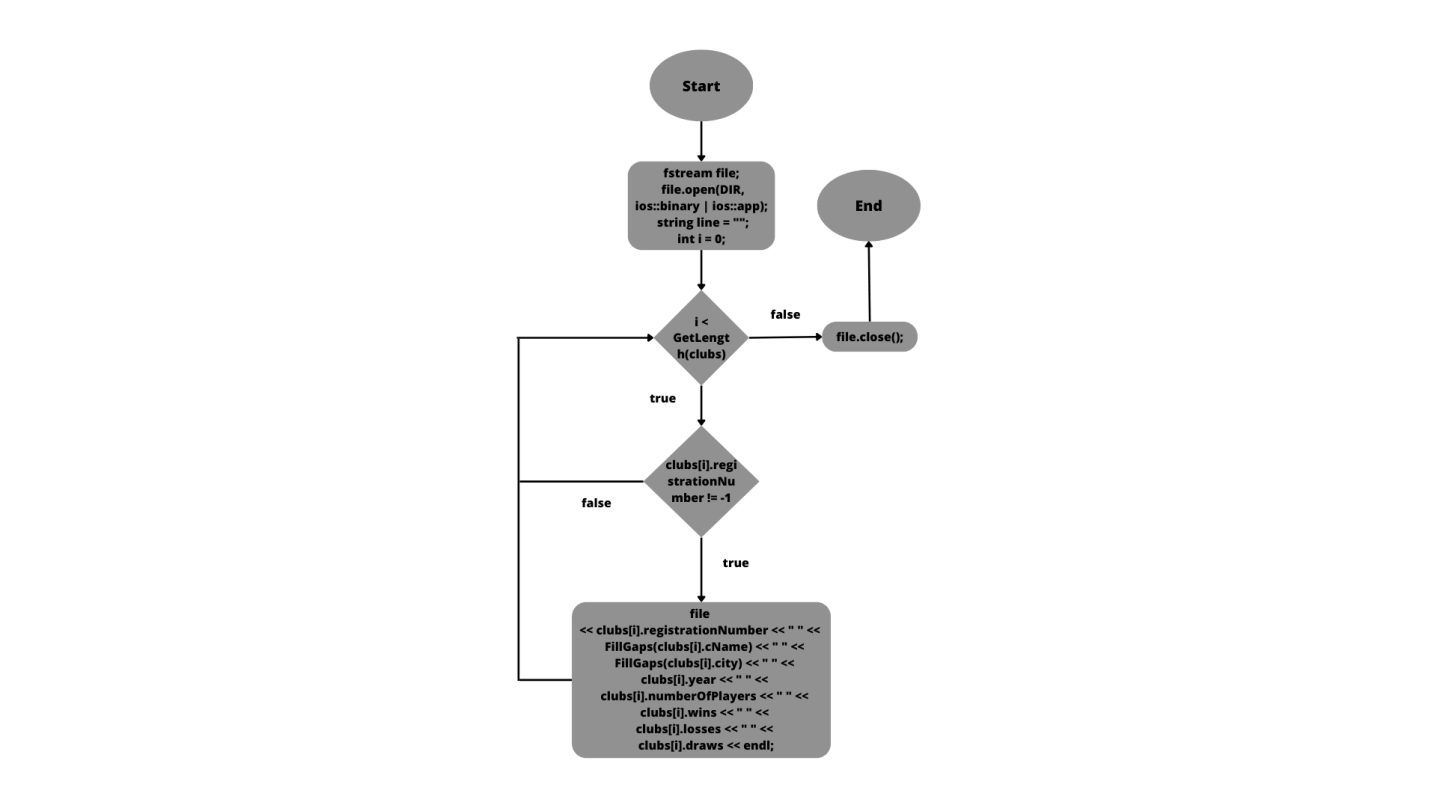
Реализация на условие F

Анализ на алгоритъма, който трябва да се реализира

Управление на файл:

1. Извеждане на масива с отбори във файл (двоичен)
2. Въвеждане на масива с отбори от файл (двоичен)

Блок схема на алгоритъма



Функция с която е реализиран алгоритъма

void AddInFile(FClub clubs[]);//Func --F--(a)

void FillClubsFromFile(FClub clubs[]);//Func --F--(b)

void SortFile(FClub clubs[]);

string FillGaps(char a[]);

void AddInFile(FClub clubs[]) {

fstream file;

file.open(DIR, ios::binary | ios::app);

string line = "";

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

if (clubs[i].registrationNumber != -1)

file << clubs[i].registrationNumber << " " << FillGaps(clubs[i].cName) << " " << FillGaps(clubs[i].city) << " " << clubs[i].year << " " << clubs[i].numberOfPlayers << " " << clubs[i].wins << " " << clubs[i].losses << " " << clubs[i].draws << endl;

}

file.close();

}

void FillClubsFromFile(FClub clubs[]) {

ifstream file;

file.open(DIR, ios::binary | ios::in);

if (file.is\_open())

{

string line, lines[8];

FClub club;

int count = 0;

while (file >> line)

{

lines[count++] = line;

if (count == 8)

{

club.registrationNumber = stoi(lines[0]);

strcpy\_s(club.cName, lines[1].c\_str());

strcpy\_s(club.city, lines[2].c\_str());

club.year = stoi(lines[3]);

club.numberOfPlayers = stoi(lines[4]);

club.wins = stoi(lines[5]);

club.losses = stoi(lines[6]);

club.draws = stoi(lines[7]);

count = 0;

clubs[CheckHowManyClubs(clubs)] = club;

}

}

file.close();

}

}

void SortFile(FClub clubs[]) {

fstream file;

file.open(DIR, ios::out);

file.clear();

file.close();

AddInFile(clubs);

}

string FillGaps(char a[]) {

string s = "";

for (int i = 0; i < strlen(a); i++)

{

if (a[i] == ' ')

{

s += '\_';

}

else s += a[i];

}

return s;

}

#### Входни данни на функцията

AddInFile() - FillClubsFromFile() - SortFile() - FClub clubs[] – Това е масивът, в който се съдържат отборите.

FillGaps() - char a[] – символен низ

#### Изходни данни на функцията или данни, които се извеждат

void AddInFile() –добавя информацията за масива с отбори във файл(двоичен)

void FillClubsFromFile() – добавя информацията от файл в масива с отбори при стартиране на програмата

void SortFile() – изчиства файла и го запълва пак

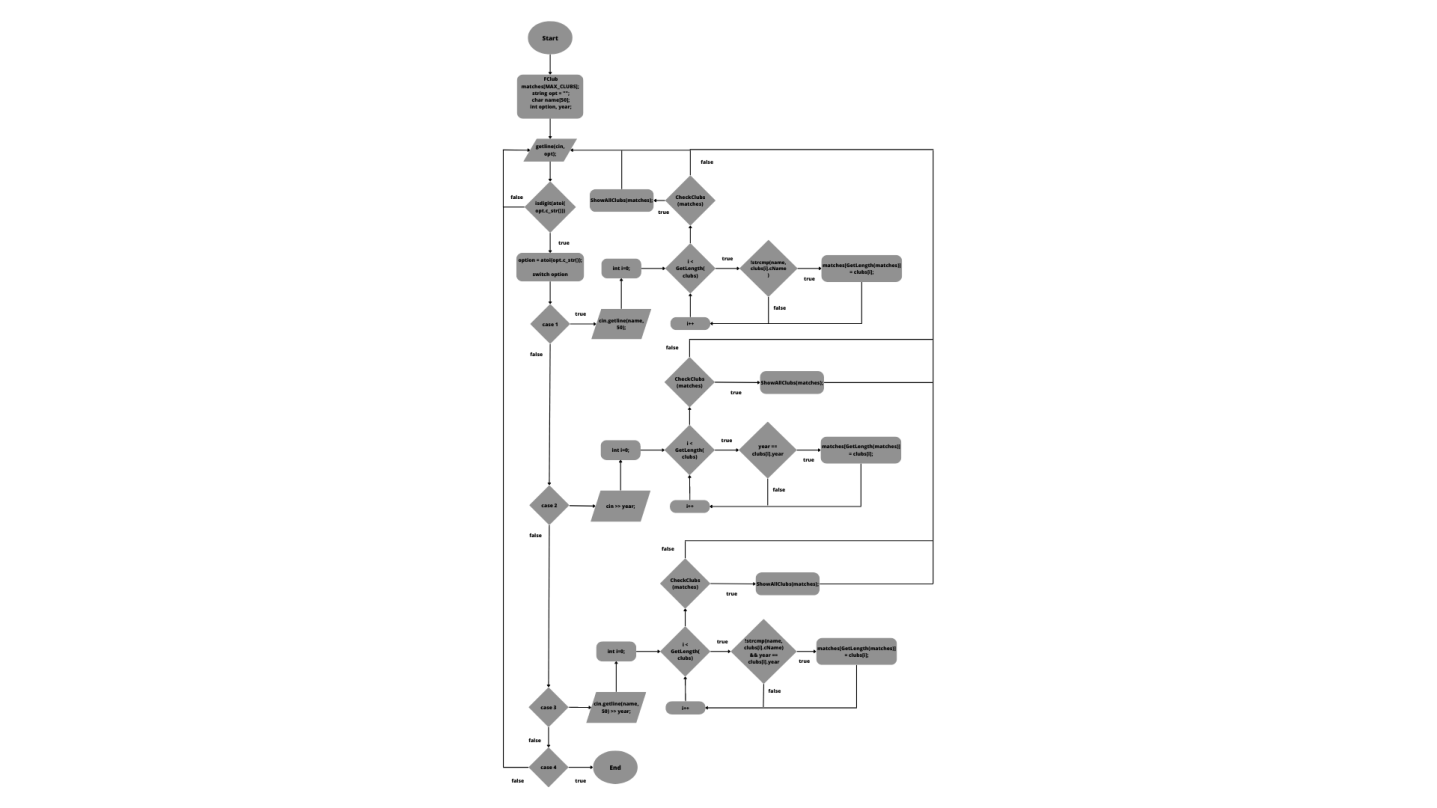
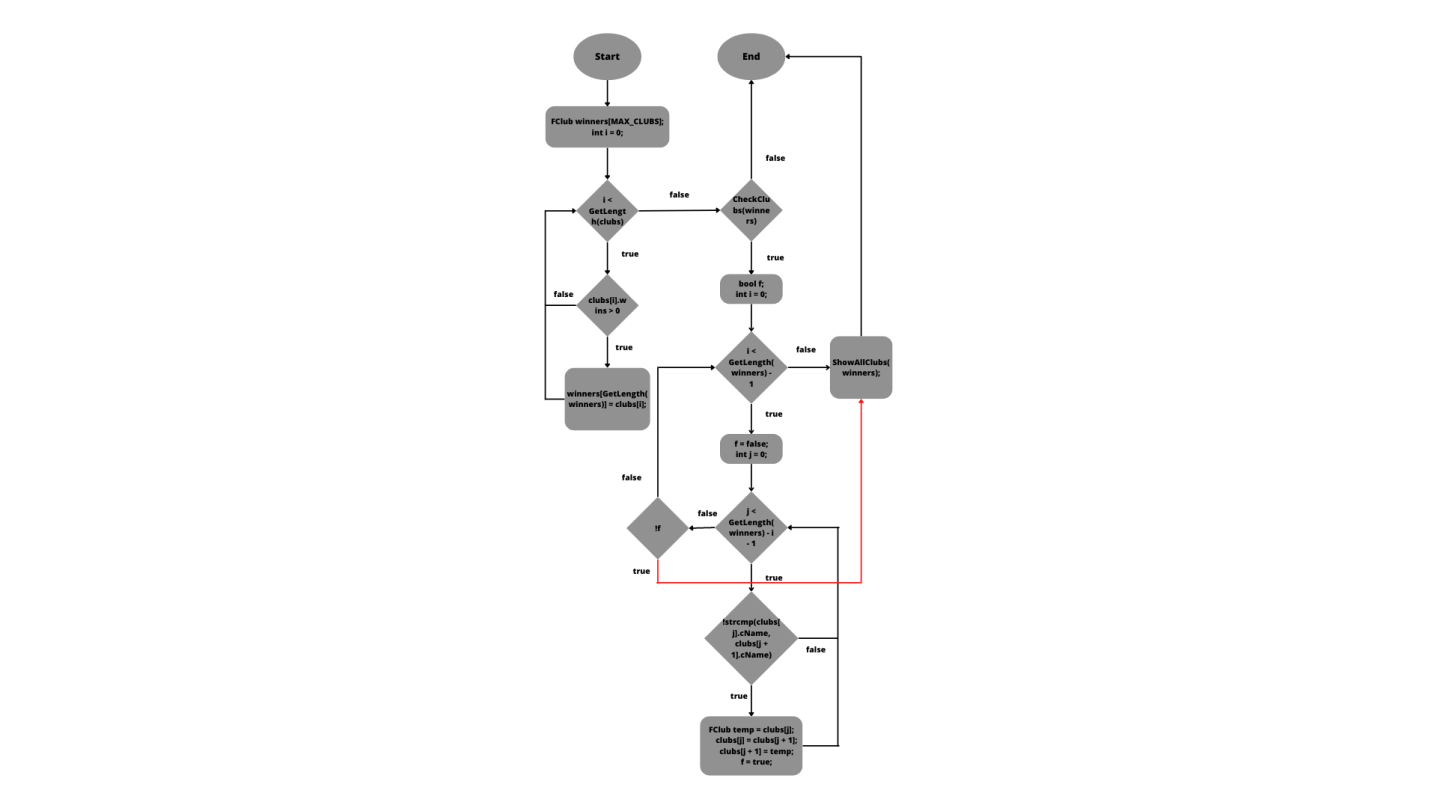
string FilGaps() – връща символен низ без празни интервали

Реализация на условие … - допълнение първо

Анализ на алгоритъма, който трябва да се реализира

1. Създайте подменю, в което се влиза от основното, с нови функции за:
2. Извеждане на отборите със спечелени първенства (брой спечелени първенства > 0), подредени по име на отбора.
3. Търсене и извеждане на отборите по град и година на основаване

Блок схема на алгоритъма



Функция с която е реализиран алгоритъма

void DisplaySubMenu(FClub clubs[]);//Func --G--

void ShowClubsWithWins(FClub clubs[]);//Func-- --G--(a)

void SearchClubs(FClub clubs[]);//Func --G--(b)

void DisplaySubMenu(FClub clubs[]) {

int option;

string opt = "";

do {

do {

Reset();

cout << "Choose option:\n" <<

"1 - Show the club with the most wins\n"<<

"2 - Show the clubs with the least tabulated players\n"<<

"3 - Show clubs with wins more than 0, sorted by name\n" <<

"4 - Search club by name and year\n" <<

"5 - Back\n";

getline(cin, opt);

} while (isdigit(atoi(opt.c\_str())));

option = atoi(opt.c\_str());

switch (option)

{

case 1:

GetClubWithMostWins(clubs);

break;

case 2:

GetClubsWithLeastPlayers(clubs);

break;

case 3:

ShowClubsWithWins(clubs);

break;

case 4:

SearchClubs(clubs);

break;

case 5:

return;

default:

cout << "Please enter valid number\n";

break;

}

Delay();

cout << "\nPress any button to continue\n";

\_getch();

Reset();

} while (true);

}

void ShowClubsWithWins(FClub clubs[]) {

FClub winners[MAX\_CLUBS];

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

if (clubs[i].wins > 0)

{

winners[CheckHowManyClubs(winners)] = clubs[i];

}

}

if (CheckClubs(winners))

{

bool f;

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

{

f = false;

for (int j = 0; j < CheckHowManyClubs(winners) - i - 1; j++)

{

if (!strcmp(clubs[j].cName, clubs[j + 1].cName))

{

FClub temp = clubs[j];

clubs[j] = clubs[j + 1];

clubs[j + 1] = temp;

f = true;

}

}

if (!f)break;

}

ShowAllClubs(winners);

}

else cout << "There aren't clubs with wins more than 0\n";

}

void SearchClubs(FClub clubs[]) {

FClub matches[MAX\_CLUBS];

string opt = "";

char name[50];

int option, year;

do {

do {

Reset();

cout << "Choose how do you want to search:\n" <<

"1 - By name\n" <<

"2 - By year\n" <<

"3 - By name and year\n" <<

"4 - Back\n";

getline(cin, opt);

} while (isdigit(atoi(opt.c\_str())));

option = atoi(opt.c\_str());

switch (option)

{

case 1:

cout << "Type the name of the club:" << endl;

cin.getline(name, 50);

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

if (!strcmp(name, clubs[i].cName))

{

matches[CheckHowManyClubs(matches)] = clubs[i];

}

}

if (CheckClubs(matches))

{

ShowAllClubs(matches);

}

else cout << "There aren't clubs with that name\n";

break;

case 2:

cout << "Enter the year of the club:" << endl;

cin >> year;

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

if (year == clubs[i].year)

{

matches[CheckHowManyClubs(matches)] = clubs[i];

}

}

if (CheckClubs(matches))

{

ShowAllClubs(matches);

}

else cout << "There aren't clubs created at that year\n";

break;

case 3:

cout << "Type name and year\n";

cin.getline(name, 50) >> year;

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

if (!strcmp(name, clubs[i].cName) && year == clubs[i].year)

{

matches[CheckHowManyClubs(matches)] = clubs[i];

}

}

if (CheckClubs(matches))

{

ShowAllClubs(matches);

}

else cout << "There aren't clubs with that name created at that year\n";

break;

case 4:

return;

default:

cout << "Please enter valid number\n";

break;

}

Delay();

cout << "\nPress any button to continue\n";

\_getch();

Reset();

} while (true);

}

#### Входни данни на функцията

DisplaySubMenu() - ShowClubsWithWins() - SearchClubs() - FClub clubs[] – Това е масивът, в който се съдържат отборите.

#### Изходни данни на функцията или данни, които се извеждат

void DisplaySubMenu() – Тази функция играе ролята на мост между различните услуги, от които потребителят може да избира.

void ShowClubsWithWins() – Показва отборите, които имат брой победи повече от 0

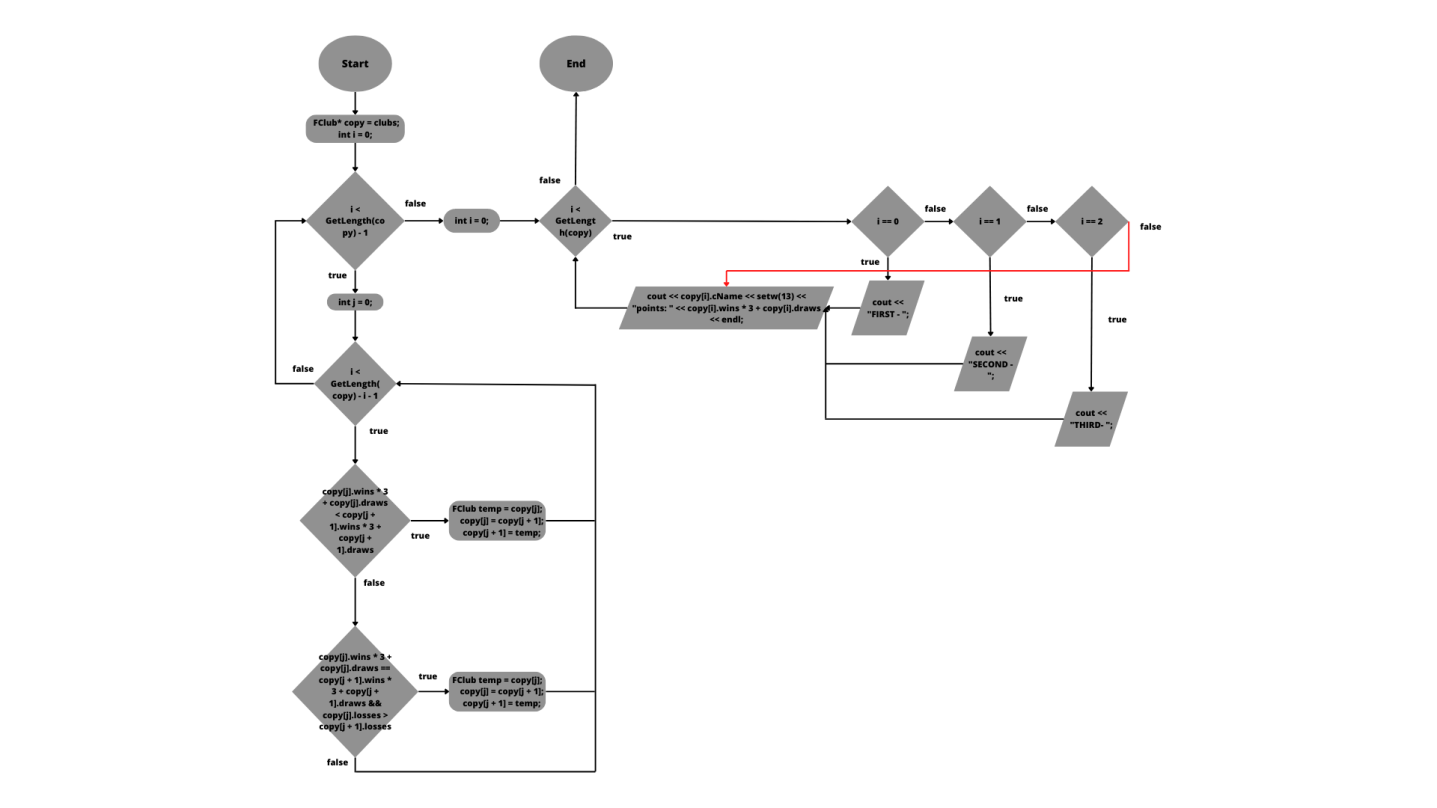
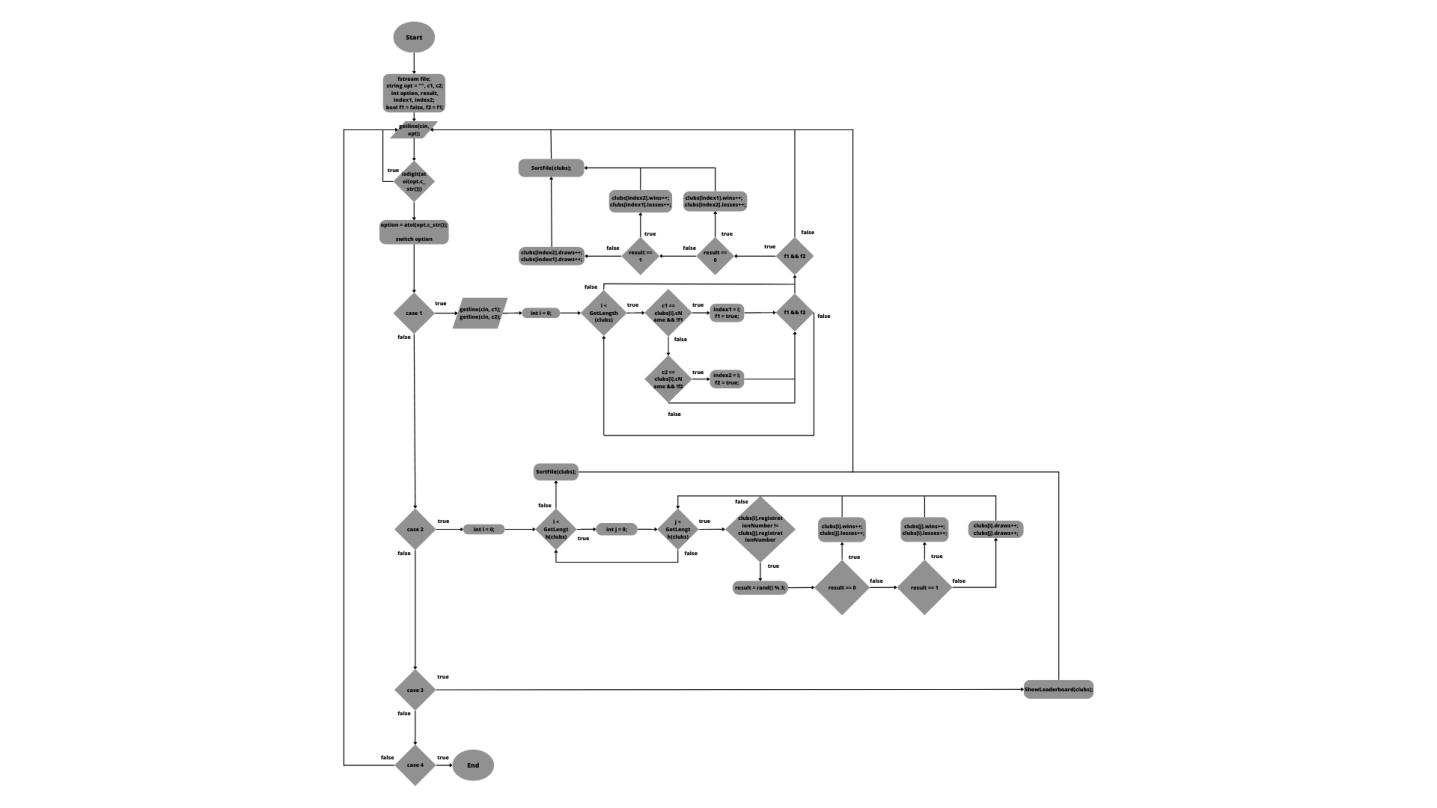
void SearchClubs() – Да опцията на потребителя да търси отборите по – име, година, име и година

Реализация на условие … - допълнение второ

Анализ на алгоритъма, който трябва да се реализира

1. Организиране на първенство
2. Добавете в структурата на отбора брой победи, брой загуби и брой равни мачове, като по подразбиране да имат стойност 0.
3. Да се добави функционалност за организиране на срещи между отборите (еднократно всеки срещу всеки или общо по 15 на отбор). Да се генерира случаен резултат за отбора-домакин – победа, равен или загуба. Стойностите на променливите, съответстващи на резултата и за двата отбора, участващи в дадената среща, да се увеличават с 1.
4. Да се добави функционалност за извеждане на класиране на отборите по брой постигнати точки, като за победа се присъждат 3 т, за равен – 1 т., а за загуба – 0 т.

Блок схема на алгоритъма



Функция с която е реализиран алгоритъма

void Tournament(FClub clubs[]);//Func --H—

void ShowLeaderboard(FClub clubs[]);

void Tournament(FClub clubs[]) {

fstream file;

string opt = "", c1, c2;

int option, result, index1, index2;

bool f1 = false, f2 = f1;

do {

do {

cout << "Choose:\n" <<

"1 - Team vs Team\n" <<

"2 - All vs All\n" <<

"3 - Show Leaderboard\n" <<

"4 - Back\n";

getline(cin, opt);

} while (isdigit(atoi(opt.c\_str())));

option = atoi(opt.c\_str());

switch (option) {

case 1:

cout << "Type any 2 club names you choose\n";

cout << endl << "Club 1:\n";

getline(cin, c1);

cout << endl << "Club 2:\n";

getline(cin, c2);

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

if (c1 == clubs[i].cName && !f1)

{

index1 = i;

f1 = true;

}

else if (c2 == clubs[i].cName && !f2)

{

index2 = i;

f2 = true;

}

if (f1 && f2)break;

}

if (f1 && f2)

{

result = rand() % 3;

if (result == 0)

{

clubs[index1].wins++;

clubs[index2].losses++;

cout << c1 << " won the match\n";

}

else if (result == 1)

{

clubs[index2].wins++;

clubs[index1].losses++;

cout << c2 << " won the match\n";

}

else

{

clubs[index2].draws++;

clubs[index1].draws++;

cout << "The match was a draw\n";

}

file.open(DIR, ios::out);

file.clear();

file.close();

AddInFile(clubs);

}

else

{

cout << "The clubs' names you entered are not valid\n";

break;

}

break;

case 2:

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

for (int j = 0; j < CheckHowManyClubs(clubs); j++)

{

if (clubs[i].registrationNumber != clubs[j].registrationNumber)

{

result = rand() % 3;

if (result == 0)

{

clubs[i].wins++;

clubs[j].losses++;

}

else if (result == 1)

{

clubs[j].wins++;

clubs[i].losses++;

}

else {

clubs[i].draws++;

clubs[j].draws++;

}

}

}

}

file.open(DIR, ios::out);

file.clear();

file.close();

AddInFile(clubs);

break;

case 3:

ShowLeaderboard(clubs);

break;

case 4:

return;

default:

cout << "Please enter valid number\n";

Delay();

Reset();

continue;

}

Delay();

cout << "\nPress any button to continue\n";

\_getch();

Reset();

} while (true);

}

void ShowLeaderboard(FClub clubs[]) {

FClub\* copy = clubs;

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

{

for (int j = 0; j < CheckHowManyClubs(copy) - i - 1; j++)

{

if (copy[j].wins \* 3 + copy[j].draws < copy[j + 1].wins \* 3 + copy[j + 1].draws)

{

FClub temp = copy[j];

copy[j] = copy[j + 1];

copy[j + 1] = temp;

}

else if (copy[j].wins \* 3 + copy[j].draws == copy[j + 1].wins \* 3 + copy[j + 1].draws && copy[j].losses > copy[j + 1].losses)

{

FClub temp = copy[j];

copy[j] = copy[j + 1];

copy[j + 1] = temp;

}

}

}

for (int i = 0; i < CheckHowManyClubs(copy); i++)

{

if (i == 0)

cout << "FIRST - " << copy[i].cName << setw(13) << "points: " << copy[i].wins \* 3 + copy[i].draws << endl;

else if (i == 1)

cout << "SECOND - " << copy[i].cName << setw(13) << "points: " << copy[i].wins \* 3 + copy[i].draws << endl;

else if (i == 2)

cout << "THIRD - " << copy[i].cName << setw(13) << "points: " << copy[i].wins \* 3 + copy[i].draws << endl;

else cout << copy[i].cName << setw(13) << "points: " << copy[i].wins \* 3 + copy[i].draws << endl;

}

}

#### Входни данни на функцията

Tournament() - ShowLeaderboard() - FClub clubs[] – Това е масивът, в който се съдържат отборите.

#### Изходни данни на функцията или данни, които се извеждат

void Tournament() – Дава избор на потребителя между – мач [1 на 1; всички срещу всички; показване на точките на отборите(сортирани)]

void ShowLeaderboard() – Показва кой отбор колко точки има сортирани от най-много към най-малко

Реализация на допълнение трето

Анализ на алгоритъма, който трябва да се реализира

1. Данните в програмата да се попълват автоматично от файл при стартиране и да се записват автоматично във файл при затваряне на програмата.

Блок схема на алгоритъма

Тук се поставя блок схема на описания алгоритъм.

Функция с която е реализиран алгоритъма

void AddInFile(FClub clubs[]);//Func --F--(a)

void FillClubsFromFile(FClub clubs[]);//Func --F--(b)

void SortFile(FClub clubs[]);

string FillGaps(char a[]);

void AddInFile(FClub clubs[]) {

fstream file;

file.open(DIR, ios::binary | ios::app);

string line = "";

for (int i = 0; i < CheckHowManyClubs(clubs); i++)

{

if (clubs[i].registrationNumber != -1)

file << clubs[i].registrationNumber << " " << FillGaps(clubs[i].cName) << " " << FillGaps(clubs[i].city) << " " << clubs[i].year << " " << clubs[i].numberOfPlayers << " " << clubs[i].wins << " " << clubs[i].losses << " " << clubs[i].draws << endl;

}

file.close();

}

void FillClubsFromFile(FClub clubs[]) {

ifstream file;

file.open(DIR, ios::binary | ios::in);

if (file.is\_open())

{

string line, lines[8];

FClub club;

int count = 0;

while (file >> line)

{

lines[count++] = line;

if (count == 8)

{

club.registrationNumber = stoi(lines[0]);

strcpy\_s(club.cName, lines[1].c\_str());

strcpy\_s(club.city, lines[2].c\_str());

club.year = stoi(lines[3]);

club.numberOfPlayers = stoi(lines[4]);

club.wins = stoi(lines[5]);

club.losses = stoi(lines[6]);

club.draws = stoi(lines[7]);

count = 0;

clubs[CheckHowManyClubs(clubs)] = club;

}

}

file.close();

}

}

void SortFile(FClub clubs[]) {

fstream file;

file.open(DIR, ios::out);

file.clear();

file.close();

AddInFile(clubs);

}

string FillGaps(char a[]) {

string s = "";

for (int i = 0; i < strlen(a); i++)

{

if (a[i] == ' ')

{

s += '\_';

}

else s += a[i];

}

return s;

}

#### Входни данни на функцията

AddInFile() - FillClubsFromFile() - SortFile() - FClub clubs[] – Това е масивът, в който се съдържат отборите.

FillGaps() - char a[] – символен низ

#### Изходни данни на функцията или данни, които се извеждат

void AddInFile() –добавя информацията за масива с отбори във файл(двоичен)

void FillClubsFromFile() – добавя информацията от файл в масива с отбори при стартиране на програмата

void SortFile() – изчиства файла и го запълва пак

string FilGaps() – връща символен низ без празни интервали

Упътване за употреба

Представете какво трябва да се въведе за правилното използване на програмата, какво се извежда в резултат на нея. Какви са очакваните резултати и какво ще се случи, ако се въведат невалидни данни. Изтрийте този текст, след като се запознаете с него.

Главно меню

Очаквани входни данни

При тази функция очакваните входни данни са цели числа от 1-7, ако не се спази това условие потребителя ще трябва пак да даде команда на програмата докато не се въведе правилен вход.

Примерен вход:

"1" – При този вход програмата ще поизка от потребителя да въведе, колко футболни отбора ще въвежда и данните за тях като потребителя трябва да въведе искащите се входни данни за да се изпълни правилно тази щаст от програмата

"2" – При този вход програмата не изисква други входни данни от потребителя

"3" – При този вход пограмата изисква номер на клуб (цяло число)

"4" – При този вход програмата изисква кой вид сортиране да се направи като се избира вида с въвеждането на цяло число

"5" - При този вход програмата влиза в подменюто, което отново ще изисква цяло число от 1-5

"6" – При този вход програмата изисква какъв мач да се организира или да се изведе таблица с точките на всички отбори, като отново се изисква вход от цяло число

"7" – При този вход ще се излезе от програмата

При други входни данни програмата ще повтаря искането на цяло число 1-7

Очакван резултат от изпълнението на конкретния фрагмент от проекта

"1" – При този вход програмата не връща нищо от правилното си изпълнение освен вкарването на въведените отбори във файла

"2" – Изкарва се таблица с цялата информация за всеки един футболен

"3" – При правилно въведен номер на отбор ще се изпише: "Deleting complete!", ако не е правилно програмата ще се върне обратно в главното меню

"4" – При този вход програмата не връща нищо от правилното си изпълнение освен сортирането на на отборите по въведеният начин

"5" – При този вход програмата ще покаже подменюто

"6" – При този вход програмата ще покаже опциите за организиране на турнир между отборите

"7" – При този вход не се връща нищо

Подменю

Очаквани входни данни

При тази функция очакваните входни данни са цели числа от 1-5, ако не се спази това условие потребителя ще трябва пак да даде команда на програмата докато не се въведе правилен вход.

Примерен вход:

"1; 2; 3" – При тези входни данни не се изисква други входни данни

"4" – При този вход програмата ще поиска тип на търсене на отбор (цяло число 1-4)

"5" - При този вход програмата се връща в главното меню

При други входни данни програмата ще повтаря искането на цяло число 1-5

### Очакван резултат от изпълнението на конкретния фрагмент от проекта

"1" – Показва името и победите на отбора с най-много победи

"2" – Програмата връща всички отбори с брой на картотекирани играчи по-малко от 16

"3" – Програмата връща всички отбори, които имат повече от 0 победи сортирани по името им

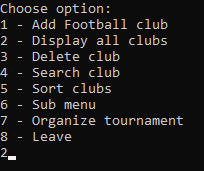
"4" – Програмата връща отборите, които съвпадат с въведените входни данни

"5" – Програмата ще покаже главното меню

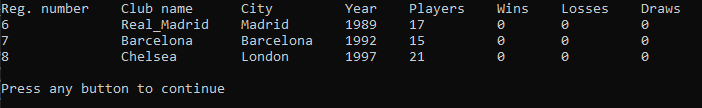
Примерно действие на програмата

Условие A

Снимка на изгледа с примерни входни данни

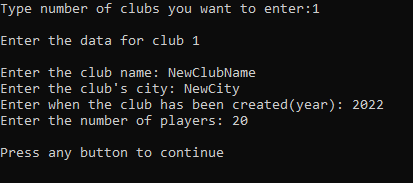


Снимка на изгледа с примерни изходни данни



Условие B

Снимка на изгледа с примерни входни данни



Снимка на изгледа с примерни изходни данни



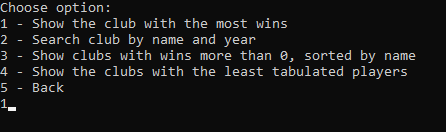
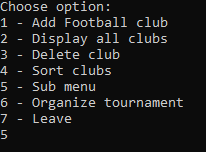
Условие C

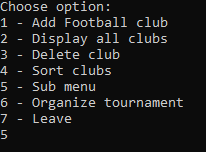
Снимка на изгледа с примерни изходни данни

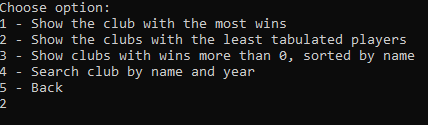


Условие D

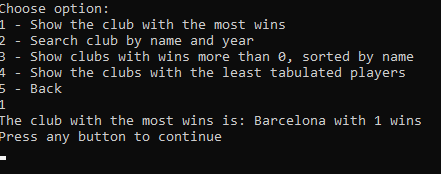
Снимка на изгледа с примерни входни данни

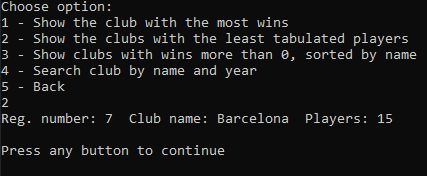






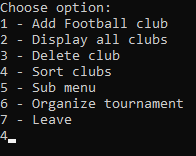
Снимка на изгледа с примерни изходни данни

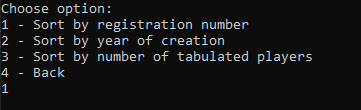




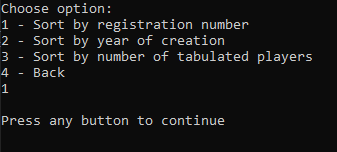
Условие E

Снимка на изгледа с примерни входни данни



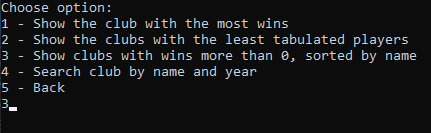
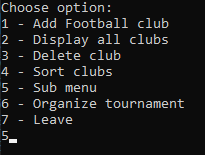


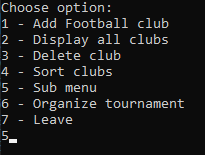
Снимка на изгледа с примерни изходни данни

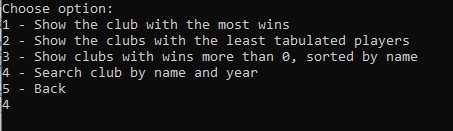


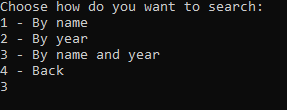
Допълнение първо

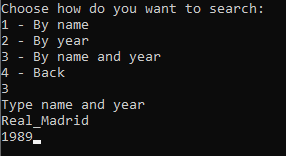
Снимка на изгледа с примерни входни данни



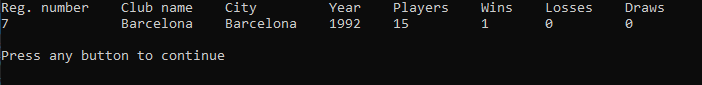


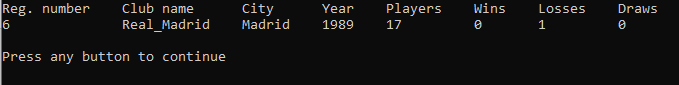






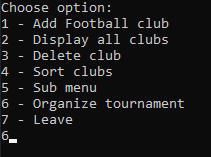
Снимка на изгледа с примерни изходни данни

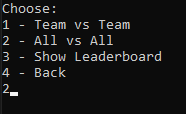


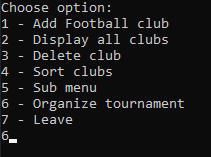


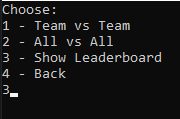
Допълнение второ

Снимка на изгледа с примерни входни данни









Снимка на изгледа с примерни изходни данни