**ПРИЛОЖЕНИЕ А**

program Laba3;

{$APPTYPE CONSOLE}

uses

System.SysUtils;

type

TTeamInfo = record

Code: integer;

Name: string[30];

Country: string[20];

TableNumber: integer;

end;

PTeamList = ^TTeamList;

TTeamList = record

Info: TTeamInfo;

Ptr: PTeamList;

end;

TPlayerInfo = record

Code: integer;

Name: string[50];

TeamCode: integer;

Position: string[20];

Success: integer;

PenaltyPoints: integer;

end;

PPlayerList = ^TPlayerList;

TPlayerList = record

Info: TPlayerInfo;

Ptr: PPlayerList;

end;

TTeamFile = file of TTeamInfo;

TPlayerFile = file of TPlayerInfo;

TTeamComparator = function(A, B: TTeamInfo): boolean;

TPlayerComparator = function(A, B: TPlayerInfo): boolean;

function PlayerSuccessComparator(A, B: TPlayerInfo): boolean;

begin

Result := A.Success > B.Success;

end;

function PlayerPenaltyComparator(A, B: TPlayerInfo): boolean;

begin

Result := A.PenaltyPoints > B.PenaltyPoints;

end;

function PlayerNameComparator(A, B: TPlayerInfo): boolean;

begin

Result := A.Name > B.Name;

end;

function TeamTableNumberComparator(A, B: TTeamInfo): boolean;

begin

Result := A.TableNumber > B.TableNumber;

end;

function TeamNameComparator(A, B: TTeamInfo): boolean;

begin

Result := A.Name > B.Name;

end;

function CheckPlayerName(Name: string; PlayerList: PPlayerList)

: boolean; overload;

var

CurrentPlayer: PPlayerList;

begin

Result := false;

CurrentPlayer := PlayerList^.Ptr;

while (CurrentPlayer <> nil) and not Result do

begin

if UpperCase(CurrentPlayer^.Info.Name) = UpperCase(Name) then

Result := true;

CurrentPlayer := CurrentPlayer^.Ptr;

end;

end;

function CheckTeamName(Name: string; TeamList: PTeamList): boolean; overload;

var

CurrentTeam: PTeamList;

begin

Result := false;

CurrentTeam := TeamList^.Ptr;

while (CurrentTeam <> nil) and not Result do

begin

if UpperCase(CurrentTeam^.Info.Name) = UpperCase(Name) then

Result := true;

CurrentTeam := CurrentTeam^.Ptr;

end;

end;

procedure MainMenu;

begin

Writeln('1. Чтение данных из файла');

Writeln('2. Просмотр списка');

Writeln('3. Сортировка данных');

Writeln('4. Поиск данных');

Writeln('5. Добавление данных');

Writeln('6. Удаление данных');

Writeln('7. Редактирование данных');

Writeln('8. Анализ игроков и команд');

Writeln('9. Выход без сохранения');

Writeln('10. Выход с сохранением');

end;

procedure ListMenu;

begin

Writeln('1. Список команд');

Writeln('2. Список футболистов');

Writeln('0. Назад');

end;

procedure PromptNotLoaded;

begin

Writeln('Списки не были загружены. Хотите ли вы создать пустые списки?');

Writeln('[1] Да [0] Нет');

end;

procedure PromptLoaded;

begin

Writeln('Списки уже были загружены. Повторная загрузка перезапишет текущие данные. Вы УВЕРЕНЫ, что хотите этого?');

Writeln('[1] Да [0] Нет');

end;

procedure CreateLists(var PlayerList: PPlayerList; var TeamList: PTeamList);

begin

New(PlayerList);

PlayerList^.Ptr := nil;

New(TeamList);

TeamList^.Ptr := nil;

Writeln('Списки успешно созданы!');

Writeln;

end;

procedure LoadLists(var PlayerList: PPlayerList; var TeamList: PTeamList);

var

PlayerFile: TPlayerFile;

TeamFile: TTeamFile;

PlayerElem: PPlayerList;

TeamElem: PTeamList;

begin

CreateLists(PlayerList, TeamList);

if FileExists('Players.dat') then

begin

PlayerElem := PlayerList;

AssignFile(PlayerFile, 'Players.dat');

Reset(PlayerFile);

While not EOF(PlayerFile) do

begin

New(PlayerElem^.Ptr);

PlayerElem := PlayerElem^.Ptr;

Read(PlayerFile, PlayerElem.Info);

end;

CloseFile(PlayerFile);

PlayerElem^.Ptr := nil;

end;

if FileExists('Teams.dat') then

begin

TeamElem := TeamList;

AssignFile(TeamFile, 'Teams.dat');

Reset(TeamFile);

While not EOF(TeamFile) do

begin

New(TeamElem^.Ptr);

TeamElem := TeamElem^.Ptr;

Read(TeamFile, TeamElem.Info);

end;

CloseFile(TeamFile);

TeamElem^.Ptr := nil;

end;

end;

procedure DisposeLists(var PlayerList: PPlayerList; var TeamList: PTeamList);

var

PlayerElementOld: PPlayerList;

TeamElementOld: PTeamList;

begin

while PlayerList^.Ptr <> nil do

begin

PlayerElementOld := PlayerList;

PlayerList := PlayerList^.Ptr;

Dispose(PlayerElementOld);

end;

Dispose(PlayerList);

PlayerList := nil;

while TeamList^.Ptr <> nil do

begin

TeamElementOld := TeamList;

TeamList := TeamList^.Ptr;

Dispose(TeamElementOld);

end;

Dispose(TeamList);

TeamList := nil;

end;

procedure SaveToFile(PlayerList: PPlayerList; TeamList: PTeamList);

var

PlayerHead: PPlayerList;

TeamHead: PTeamList;

PlayerFile: TPlayerFile;

TeamFile: TTeamFile;

begin

PlayerHead := PlayerList;

TeamHead := TeamList;

if not FileExists('Players.dat') then

FileClose(FileCreate('Players.dat'));

AssignFile(PlayerFile, 'Players.dat');

ReWrite(PlayerFile);

While PlayerList^.Ptr <> nil do

begin

PlayerList := PlayerList^.Ptr;

Write(PlayerFile, PlayerList^.Info);

end;

CloseFile(PlayerFile);

if not FileExists('Teams.dat') then

FileClose(FileCreate('Teams.dat'));

AssignFile(TeamFile, 'Teams.dat');

ReWrite(TeamFile);

While TeamList^.Ptr <> nil do

begin

TeamList := TeamList^.Ptr;

Write(TeamFile, TeamList^.Info);

end;

CloseFile(TeamFile);

DisposeLists(PlayerHead, TeamHead);

end;

function GetPlayerTeam(TeamList: PTeamList; Code: integer): TTeamInfo;

begin

TeamList := TeamList^.Ptr;

while (TeamList^.Info.Code <> Code) do

TeamList := TeamList^.Ptr;

Result := TeamList^.Info;

end;

function CheckCode(Code: integer; TeamList: PTeamList): boolean;

overload

begin

Result := false;

while (TeamList^.Ptr <> nil) and not Result do

begin

TeamList := TeamList^.Ptr;

if TeamList^.Info.Code = Code then

Result := true;

end;

end;

function PlayerCodeComparator(A, B: TPlayerInfo): boolean;

begin

Result := A.Code > B.Code;

end;

function PlayerTeamCodeComparator(A, B: TPlayerInfo): boolean;

begin

Result := A.TeamCode > B.TeamCode;

end;

function TeamCodeComparator(A, B: TTeamInfo): boolean;

begin

Result := A.Code > B.Code;

end;

function TeamCountryComparator(A, B: TTeamInfo): boolean;

begin

Result := A.Country > B.Country;

end;

procedure SortPlayer(PlayerList: PPlayerList; Compare: TPlayerComparator);

var

Base, Min: PPlayerList;

Temp: TPlayerInfo;

begin

if PlayerList <> nil then

begin

Base := PlayerList^.Ptr;

if Base <> nil then

begin

while Base^.Ptr <> nil do

begin

PlayerList := Base^.Ptr;

Min := Base;

while PlayerList <> nil do

begin

if Compare(Min^.Info, PlayerList^.Info) then

Min := PlayerList;

PlayerList := PlayerList^.Ptr;

end;

Temp := Min^.Info;

Min^.Info := Base^.Info;

Base^.Info := Temp;

Base := Base^.Ptr;

end;

end

end

else

Writeln('Список пустой');

end;

procedure SortTeam(TeamList: PTeamList; Compare: TTeamComparator);

var

Base, Min: PTeamList;

Temp: TTeamInfo;

begin

if TeamList <> nil then

begin

Base := TeamList^.Ptr;

if Base <> nil then

begin

while Base^.Ptr <> nil do

begin

TeamList := Base^.Ptr;

Min := Base;

while TeamList <> nil do

begin

if Compare(Min^.Info, TeamList^.Info) then

Min := TeamList;

TeamList := TeamList^.Ptr;

end;

Temp := Min^.Info;

Min^.Info := Base^.Info;

Base^.Info := Temp;

Base := Base^.Ptr;

end;

end

end

else

Writeln('Список пустой');

end;

function CheckCode(Code: integer; PlayerList: PPlayerList): boolean;

overload

begin

Result := false;

while (PlayerList^.Ptr <> nil) and not Result do

begin

PlayerList := PlayerList^.Ptr;

if PlayerList^.Info.Code = Code then

Result := true;

end;

end;

function CheckCode(Code: integer; PlayerList: PPlayerList;

CheckAsTeamCode: boolean): boolean;

overload

begin

Result := false;

if CheckAsTeamCode then

begin

while (PlayerList^.Ptr <> nil) and not Result do

begin

PlayerList := PlayerList^.Ptr;

if PlayerList^.Info.TeamCode = Code then

Result := true;

end;

end;

end;

procedure WriteInfo(TeamInfo: TTeamInfo); overload;

begin

Writeln(Format('|%5d|%-27s|%-22s|%11d|', [TeamInfo.Code, TeamInfo.Name,

TeamInfo.Country, TeamInfo.TableNumber]));

end;

procedure WriteInfo(PlayerInfo: TPlayerInfo); overload;

begin

Writeln(Format('|%5d|%-25s|%6d|%-12s|%7d|%13d|', [PlayerInfo.Code,

PlayerInfo.Name, PlayerInfo.TeamCode, PlayerInfo.Position,

PlayerInfo.Success, PlayerInfo.PenaltyPoints]));

end;

procedure WriteList(TeamList: PTeamList); overload;

begin

if TeamList^.Ptr <> nil then

begin

Writeln('|--------------------------------------------------------------------|');

Writeln(Format('| %-3s | %-25s | %-20s | %9s |', ['Код', 'Название',

'Страна', 'Таблица']));

Writeln('|-----|---------------------------|----------------------|-----------|');

repeat

TeamList := TeamList^.Ptr;

WriteInfo(TeamList^.Info);

until (TeamList^.Ptr = nil);

Writeln('|--------------------------------------------------------------------|');

end

else

begin

Writeln('Список пустой');

Writeln;

end;

end;

procedure WriteList(PlayerList: PPlayerList); overload;

begin

if PlayerList^.Ptr <> nil then

begin

Writeln('|-------------------------------------------------------------------------|');

Writeln(Format('|%-5s|%-25s|%-6s|%-12s|%-7s|%-13s|', ['Код', 'ФИО', 'КодК',

'Амплуа', 'Успех', 'Штраф. очки']));

Writeln('|-----|-------------------------|------|------------|-------|-------------|');

repeat

PlayerList := PlayerList^.Ptr;

WriteInfo(PlayerList^.Info);

until (PlayerList^.Ptr = nil);

Writeln('|-------------------------------------------------------------------------|');

end

else

begin

Writeln('Список пустой');

Writeln;

end;

end;

procedure SearchPlayer(PlayerList: PPlayerList; Compare: TPlayerComparator;

Key: TPlayerInfo);

begin

if PlayerList <> nil then

begin

Writeln('|-------------------------------------------------------------------------|');

Writeln(Format('|%-5s|%-25s|%-6s|%-12s|%-7s|%-13s|', ['Код', 'ФИО', 'КодК',

'Амплуа', 'Успех', 'Штраф. очки']));

Writeln('|-----|-------------------------|------|------------|-------|-------------|');

while PlayerList^.Ptr <> nil do

begin

PlayerList := PlayerList^.Ptr;

if not(Compare(PlayerList^.Info, Key) or Compare(Key, PlayerList^.Info))

then

begin

WriteInfo(PlayerList^.Info);

end;

end;

Writeln('|-------------------------------------------------------------------------|');

end

else

Writeln('Список пустой');

end;

procedure SearchTeam(TeamList: PTeamList; Compare: TTeamComparator;

Key: TTeamInfo);

begin

if TeamList <> nil then

begin

Writeln('|---------------------------------------------------------------------------|');

Writeln(Format('|%-5s|%-25s|%-6s|%-12s|%-7s|%-13s|', ['Код', 'Название',

'Страна', 'Таблица']));

Writeln('|-----|-------------------------|------|------------|-------|-------------|');

while TeamList^.Ptr <> nil do

begin

TeamList := TeamList^.Ptr;

if not(Compare(TeamList^.Info, Key) or Compare(Key, TeamList^.Info)) then

begin

WriteInfo(TeamList^.Info);

end;

end;

Writeln('|---------------------------------------------------------------------------|');

end

else

Writeln('Список пустой');

end;

procedure SearchPlayerByName(PlayerList: PPlayerList; Name: string);

var

PlayerKey: TPlayerInfo;

begin

PlayerKey.Name := Name;

SearchPlayer(PlayerList, PlayerNameComparator, PlayerKey);

end;

procedure SearchTeamByName(TeamList: PTeamList; Name: string);

var

TeamKey: TTeamInfo;

begin

TeamKey.Name := Name;

SearchTeam(TeamList, TeamNameComparator, TeamKey);

end;

procedure ReadPlayer(var TeamCode: integer; var Name, Position: string;

var Success, PenaltyPoints: integer; TeamList: PTeamList);

var

Error: boolean;

begin

repeat

WriteList(TeamList);

Write('Введите код команды: ');

Readln(TeamCode);

Error := false;

if not CheckCode(TeamCode, TeamList) then

begin

Error := true;

Writeln('Такой команды нет');

end;

until not Error;

Write('Введите ФИО футболиста: ');

Readln(Name);

Write('Введите амплуа футболиста: ');

Readln(Position);

Write('Введите успешность футболиста: ');

Readln(Success);

Write('Введите штрафные очки футболиста: ');

Readln(PenaltyPoints);

end;

procedure ReadTeam(var Name, Country: string; var TableNumber: integer);

begin

Write('Введите название команды: ');

Readln(Name);

Write('Введите страну команды: ');

Readln(Country);

Write('Введите номер в турнирной таблице: ');

Readln(TableNumber);

end;

procedure AddElement(PlayerList: PPlayerList; TeamList: PTeamList;

var Max: integer);

overload

var

TeamCode, Success, PenaltyPoints: integer;

Name, Position: string;

begin

if TeamList^.Ptr = nil then

begin

Writeln('Невозможно добавить футболиста: список команд пустой');

Writeln;

end

else

begin

ReadPlayer(TeamCode, Name, Position, Success, PenaltyPoints, TeamList);

if CheckPlayerName(Name, PlayerList) then

begin

Writeln('Ошибка: футболист с таким ФИО уже существует.');

Writeln;

end

else

begin

while PlayerList^.Ptr <> nil do

begin

PlayerList := PlayerList^.Ptr;

end;

New(PlayerList^.Ptr);

PlayerList := PlayerList^.Ptr;

PlayerList.Info.Code := Max + 1;

Inc(Max);

PlayerList.Info.TeamCode := TeamCode;

PlayerList.Info.Name := Name;

PlayerList.Info.Position := Position;

PlayerList.Info.Success := Success;

PlayerList.Info.PenaltyPoints := PenaltyPoints;

PlayerList.Ptr := nil;

Writeln('Футболист добавлен!');

Writeln;

end;

end;

end;

procedure AddElement(TeamList: PTeamList; var Max: integer);

overload

var

Name, Country: string;

TableNumber: integer;

begin

ReadTeam(Name, Country, TableNumber);

if CheckTeamName(Name, TeamList) then

begin

Writeln('Ошибка: команда с таким названием уже существует.');

Writeln;

end

else

begin

while TeamList^.Ptr <> nil do

begin

TeamList := TeamList^.Ptr;

end;

New(TeamList^.Ptr);

TeamList := TeamList^.Ptr;

TeamList.Info.Code := Max + 1;

Inc(Max);

TeamList.Info.Name := Name;

TeamList.Info.Country := Country;

TeamList.Info.TableNumber := TableNumber;

TeamList.Ptr := nil;

Writeln('Команда добавлена!');

Writeln;

end;

end;

procedure RedactElement(PlayerList: PPlayerList; Code: integer;

TeamList: PTeamList; Field: integer);

overload

var

TeamCode, Success, PenaltyPoints: integer;

Name, Position: string;

Error: boolean;

begin

PlayerList := PlayerList^.Ptr;

While (PlayerList^.Ptr <> nil) and (PlayerList^.Info.Code <> Code) do

begin

PlayerList := PlayerList^.Ptr;

end;

if PlayerList^.Info.Code <> Code then

Writeln('Футболист не найден')

else

begin

case Field of

1:

begin

repeat

WriteList(TeamList);

Write('Введите новый код команды: ');

Readln(TeamCode);

Error := false;

if not CheckCode(TeamCode, TeamList) then

begin

Error := true;

Writeln('Такой команды нет');

end;

until not Error;

PlayerList.Info.TeamCode := TeamCode;

end;

2:

begin

Writeln('Введите новое ФИО: ');

Readln(Name);

PlayerList.Info.Name := Name;

end;

3:

begin

Writeln('Введите новое амплуа: ');

Readln(Position);

PlayerList.Info.Position := Position;

end;

4:

begin

Writeln('Введите новую успешность: ');

Readln(Success);

PlayerList.Info.Success := Success;

end;

5:

begin

Writeln('Введите новые штрафные очки: ');

Readln(PenaltyPoints);

PlayerList.Info.PenaltyPoints := PenaltyPoints;

end;

end;

Writeln('Футболист изменен!');

end;

Writeln;

end;

procedure RedactElement(TeamList: PTeamList; Code: integer; Field: integer);

overload

var

Name, Country: string;

TableNumber: integer;

begin

TeamList := TeamList^.Ptr;

While (TeamList^.Ptr <> nil) and (TeamList^.Info.Code <> Code) do

begin

TeamList := TeamList^.Ptr;

end;

if TeamList^.Info.Code <> Code then

Writeln('Команда не найдена')

else

begin

case Field of

1:

begin

Writeln('Введите новое название команды: ');

Readln(Name);

TeamList.Info.Name := Name;

end;

2:

begin

Writeln('Введите новую страну: ');

Readln(Country);

TeamList.Info.Country := Country;

end;

3:

begin

Writeln('Введите новый номер в таблице: ');

Readln(TableNumber);

TeamList.Info.TableNumber := TableNumber;

end;

end;

Writeln('Команда изменена!');

end;

Writeln;

end;

procedure DeleteElement(PlayerList: PPlayerList; Code: integer);

overload

var

IsFound: boolean;

Element: PPlayerList;

begin

if PlayerList <> nil then

begin

IsFound := false;

While (PlayerList^.Ptr <> nil) and (not IsFound) do

begin

if PlayerList^.Ptr^.Info.Code = Code then

begin

IsFound := true;

Element := PlayerList^.Ptr;

PlayerList^.Ptr := Element^.Ptr;

Dispose(Element);

end

else

PlayerList := PlayerList^.Ptr;

end;

if IsFound then

Writeln('Футболист удален')

else

Writeln('Футболист не найден');

end

else

Writeln('Список пустой');

Writeln;

end;

procedure DeleteElement(TeamList: PTeamList; Code: integer;

PlayerList: PPlayerList);

overload

var

IsFound: boolean;

Element: PTeamList;

begin

if TeamList <> nil then

begin

// Проверяем, есть ли футболисты в этой команде

if not CheckCode(Code, PlayerList, true) then

begin

IsFound := false;

While (TeamList^.Ptr <> nil) and (not IsFound) do

begin

if TeamList^.Ptr^.Info.Code = Code then

begin

IsFound := true;

Element := TeamList^.Ptr;

TeamList^.Ptr := Element^.Ptr;

Dispose(Element);

end

else

TeamList := TeamList^.Ptr;

end;

if IsFound then

Writeln('Команда удалена')

else

Writeln('Команда не найдена');

end

else

Writeln('Ошибка: в этой команде есть футболисты');

end

else

Writeln('Список пустой');

Writeln;

end;

procedure CalculateMax(PlayerList: PPlayerList; TeamList: PTeamList;

out PMax, TMax: integer);

begin

PMax := 0;

TMax := 0;

while PlayerList^.Ptr <> nil do

begin

PlayerList := PlayerList^.Ptr;

if PlayerList^.Info.Code > PMax then

PMax := PlayerList^.Info.Code;

end;

while TeamList^.Ptr <> nil do

begin

TeamList := TeamList^.Ptr;

if TeamList^.Info.Code > TMax then

TMax := TeamList^.Info.Code;

end;

end;

procedure SortPlayersBySuccess(PlayerList: PPlayerList);

begin

SortPlayer(PlayerList, PlayerSuccessComparator);

end;

procedure SortPlayersByPenalty(PlayerList: PPlayerList);

begin

SortPlayer(PlayerList, PlayerPenaltyComparator);

end;

procedure SortPlayersByName(PlayerList: PPlayerList);

begin

SortPlayer(PlayerList, PlayerNameComparator);

end;

procedure SortTeamsByTableNumber(TeamList: PTeamList);

begin

SortTeam(TeamList, TeamTableNumberComparator);

end;

procedure SortTeamsByName(TeamList: PTeamList);

begin

SortTeam(TeamList, TeamNameComparator);

end;

procedure PlayerSorts;

begin

Writeln('[0] По коду');

Writeln('[1] По ФИО');

Writeln('[2] По успешности');

Writeln('[3] По штрафным очкам');

Writeln('[4] По коду команды');

Writeln('[10] Главное меню');

end;

procedure PlayerSearch;

begin

Writeln('[0] По коду');

Writeln('[1] По ФИО');

Writeln('[2] По успешности');

Writeln('[3] По штрафным очкам');

Writeln('[4] По коду команды');

Writeln('[10] Главное меню');

end;

procedure TeamSorts;

begin

Writeln('[0] По коду');

Writeln('[1] По названию');

Writeln('[2] По стране');

Writeln('[3] По номеру в таблице');

Writeln('[10] Главное меню');

end;

procedure TeamSearch;

begin

Writeln('[0] По коду');

Writeln('[1] По названию');

Writeln('[2] По стране');

Writeln('[3] По номеру в таблице');

Writeln('[10] Главное меню');

end;

procedure PlayerRedacts;

begin

Writeln('[1] Код команды');

Writeln('[2] ФИО');

Writeln('[3] Амплуа');

Writeln('[4] Успешность');

Writeln('[5] Штрафные очки');

Writeln('[10] Главное меню');

end;

procedure TeamRedacts;

begin

Writeln('[1] Название');

Writeln('[2] Страна');

Writeln('[3] Номер в таблице');

Writeln('[10] Главное меню');

end;

procedure SaveToTextFile(const FileName: string; const Content: string);

var

TextFile: Text;

begin

AssignFile(TextFile, FileName);

ReWrite(TextFile);

Write(TextFile, Content);

CloseFile(TextFile);

end;

var

PlayerKey: TPlayerInfo;

TeamKey: TTeamInfo;

PlayerMax, TeamMax: integer;

StrInput: String;

Input, Input2: integer;

IsLoaded: boolean;

PlayerList: PPlayerList;

TeamList: PTeamList;

OutputContent: string;

CurrentPlayer: PPlayerList;

CurrentTeam: PTeamList;

TempPlayerList: PPlayerList;

BestPlayer: TPlayerInfo;

MostPenaltyPlayer: TPlayerInfo;

FoundPlayersForTeam: boolean;

Input2\_local: integer;

begin

IsLoaded := false;

repeat

MainMenu;

Readln(Input);

case Input of

1:

begin

LoadLists(PlayerList, TeamList);

CalculateMax(PlayerList, TeamList, PlayerMax, TeamMax);

IsLoaded := true;

end;

9:

Exit;

else

begin

PromptNotLoaded;

Readln(Input);

if Input = 1 then

begin

CreateLists(PlayerList, TeamList);

IsLoaded := true;

end;

end;

end;

until IsLoaded;

repeat

MainMenu;

Readln(Input);

case Input of

1:

begin

PromptLoaded;

Readln(Input);

if Input = 1 then

begin

DisposeLists(PlayerList, TeamList);

LoadLists(PlayerList, TeamList);

CalculateMax(PlayerList, TeamList, PlayerMax, TeamMax);

end;

Input := 1;

end;

2:

begin

ListMenu;

Readln(Input);

case Input of

1:

WriteList(TeamList);

2:

WriteList(PlayerList);

end;

Input := 2;

end;

3:

begin

ListMenu;

Readln(Input);

case Input of

1:

begin

if TeamList^.Ptr <> nil then

begin

TeamSorts;

Readln(Input2);

case Input2 of

0:

SortTeam(TeamList, TeamCodeComparator);

1:

SortTeam(TeamList, TeamNameComparator);

2:

SortTeam(TeamList, TeamCountryComparator);

3:

SortTeam(TeamList, TeamTableNumberComparator);

end;

WriteList(TeamList);

end

else

Writeln('Список пустой');

end;

2:

begin

if PlayerList^.Ptr <> nil then

begin

PlayerSorts;

Readln(Input2);

case Input2 of

0:

SortPlayer(PlayerList, PlayerCodeComparator);

1:

SortPlayer(PlayerList, PlayerNameComparator);

2:

SortPlayer(PlayerList, PlayerSuccessComparator);

3:

SortPlayer(PlayerList, PlayerPenaltyComparator);

4:

SortPlayer(PlayerList, PlayerTeamCodeComparator);

end;

WriteList(PlayerList);

end

else

Writeln('Список пустой');

end;

end;

Input := 3;

end;

4:

begin

ListMenu;

Readln(Input);

case Input of

1:

begin

if TeamList^.Ptr <> nil then

begin

TeamSearch;

Readln(Input2);

case Input2 of

0:

begin

Write('Введите код команды: ');

Readln(Input);

TeamKey.Code := Input;

SearchTeam(TeamList, TeamCodeComparator, TeamKey);

end;

1:

begin

Write('Введите название команды: ');

Readln(StrInput);

TeamKey.Name := StrInput;

SearchTeam(TeamList, TeamNameComparator, TeamKey);

end;

2:

begin

Write('Введите страну команды: ');

Readln(StrInput);

TeamKey.Country := StrInput;

SearchTeam(TeamList, TeamCountryComparator, TeamKey);

end;

3:

begin

Write('Введите номер в турнирной таблице: ');

Readln(Input);

TeamKey.TableNumber := Input;

SearchTeam(TeamList, TeamTableNumberComparator,

TeamKey);

end;

end;

end

else

Writeln('Список пустой');

end;

2:

begin

if PlayerList^.Ptr <> nil then

begin

PlayerSearch;

Readln(Input2);

case Input2 of

0:

begin

Write('Введите код футболиста: ');

Readln(Input);

PlayerKey.Code := Input;

SearchPlayer(PlayerList, PlayerCodeComparator,

PlayerKey);

end;

1:

begin

Write('Введите ФИО футболиста: ');

Readln(StrInput);

PlayerKey.Name := StrInput;

SearchPlayer(PlayerList, PlayerNameComparator,

PlayerKey);

end;

2:

begin

Write('Введите успешность футболиста: ');

Readln(Input);

PlayerKey.Success := Input;

SearchPlayer(PlayerList, PlayerSuccessComparator,

PlayerKey);

end;

3:

begin

Write('Введите штрафные очки футболиста: ');

Readln(Input);

PlayerKey.PenaltyPoints := Input;

SearchPlayer(PlayerList, PlayerPenaltyComparator,

PlayerKey);

end;

4:

begin

Write('Введите код команды футболиста: ');

Readln(Input);

PlayerKey.TeamCode := Input;

SearchPlayer(PlayerList, PlayerTeamCodeComparator,

PlayerKey);

end;

end;

end

else

Writeln('Список пустой');

end;

end;

Input := 4;

end;

5:

begin

ListMenu;

Readln(Input);

case Input of

1:

AddElement(TeamList, TeamMax);

2:

AddElement(PlayerList, TeamList, PlayerMax);

end;

Input := 5;

end;

6:

begin

ListMenu;

Readln(Input);

case Input of

1:

begin

if TeamList^.Ptr <> nil then

begin

WriteList(TeamList);

Write('Введите код команды, которую хотите удалить: ');

Readln(Input);

DeleteElement(TeamList, Input, PlayerList);

end

else

Writeln('Список пустой');

end;

2:

begin

if PlayerList^.Ptr <> nil then

begin

WriteList(PlayerList);

Write('Введите код футболиста, которого хотите удалить: ');

Readln(Input);

DeleteElement(PlayerList, Input);

end

else

Writeln('Список пустой');

end;

end;

Input := 6;

end;

7:

begin

ListMenu;

Readln(Input);

case Input of

1:

begin

if TeamList^.Ptr <> nil then

begin

WriteList(TeamList);

Write('Введите код команды, которую хотите изменить: ');

Readln(Input);

TeamRedacts;

Readln(Input2);

RedactElement(TeamList, Input, Input2);

end

else

Writeln('Список пустой');

end;

2:

begin

if PlayerList^.Ptr <> nil then

begin

WriteList(PlayerList);

Write('Введите код футболиста, которого хотите изменить: ');

Readln(Input);

PlayerRedacts;

Readln(Input2);

RedactElement(PlayerList, Input, TeamList, Input2);

end

else

Writeln('Список пустой');

end;

end;

Input := 7;

end;

8:

begin

Writeln('Анализ игроков и команд:');

Writeln('1. Топ-10 самых успешных игроков');

Writeln('2. Топ-10 игроков с максимальной суммой штрафных очков');

Writeln('3. Определить самого успешного игрока и игрока с максимальной суммой штрафных очков для каждой команды');

Writeln('0. Назад');

Readln(Input);

case Input of

1: // Топ-10 самых успешных

begin

if PlayerList^.Ptr <> nil then

begin

SortPlayersBySuccess(PlayerList);

OutputContent := 'Топ-10 самых успешных игроков:' + #13#10;

OutputContent := OutputContent +

Format('|%-5s|%-25s|%-6s|%-12s|%-7s|%-13s|',

['Код', 'ФИО', 'КодК', 'Амплуа', 'Успех', 'Штраф. очки']

) + #13#10;

OutputContent := OutputContent +

'|-----|-------------------------|------|------------|-------|-------------|'

+ #13#10;

Writeln('Топ-10 самых успешных игроков:');

Writeln('|-------------------------------------------------------------------------|');

Writeln(Format('|%-5s|%-25s|%-6s|%-12s|%-7s|%-13s|',

['Код', 'ФИО', 'КодК', 'Амплуа', 'Успех', 'Штраф. очки']));

Writeln('|-----|-------------------------|------|------------|-------|-------------|');

Input2\_local := 0;

CurrentPlayer := PlayerList^.Ptr;

while (CurrentPlayer <> nil) and (Input2\_local < 10) do

begin

WriteInfo(CurrentPlayer^.Info);

OutputContent := OutputContent +

Format('|%5d|%-25s|%6d|%-12s|%7d|%13d|' + #13#10,

[CurrentPlayer^.Info.Code, CurrentPlayer^.Info.Name,

CurrentPlayer^.Info.TeamCode,

CurrentPlayer^.Info.Position, CurrentPlayer^.Info.Success,

CurrentPlayer^.Info.PenaltyPoints]);

CurrentPlayer := CurrentPlayer^.Ptr;

Inc(Input2\_local);

end;

OutputContent := OutputContent +

'|---------------------------------------------------------------------------|'

+ #13#10;

Writeln('|-------------------------------------------------------------------------|');

SaveToTextFile('top\_successful\_players.txt', OutputContent);

Writeln('Результаты сохранены в файл top\_successful\_players.txt');

end

else

Writeln('Список футболистов пустой');

end;

2: // Топ-10 по штрафным очкам

begin

if PlayerList^.Ptr <> nil then

begin

SortPlayersByPenalty(PlayerList);

OutputContent :=

'Топ-10 игроков с максимальной суммой штрафных очков:' +

#13#10;

OutputContent := OutputContent +

Format('|%-5s|%-25s|%-6s|%-12s|%-7s|%-13s|',

['Код', 'ФИО', 'КодК', 'Амплуа', 'Успех', 'Штраф. очки']

) + #13#10;

OutputContent := OutputContent +

'|-----|-------------------------|------|------------|-------|-------------|'

+ #13#10;

Writeln('Топ-10 игроков с максимальной суммой штрафных очков:');

Writeln('|-------------------------------------------------------------------------|');

Writeln(Format('|%-5s|%-25s|%-6s|%-12s|%-7s|%-13s|',

['Код', 'ФИО', 'КодК', 'Амплуа', 'Успех', 'Штраф. очки']));

Writeln('|-----|-------------------------|------|------------|-------|-------------|');

Input2\_local := 0;

CurrentPlayer := PlayerList^.Ptr;

while (CurrentPlayer <> nil) and (Input2\_local < 10) do

begin

WriteInfo(CurrentPlayer^.Info);

OutputContent := OutputContent +

Format('|%5d|%-25s|%6d|%-12s|%7d|%13d|' + #13#10,

[CurrentPlayer^.Info.Code, CurrentPlayer^.Info.Name,

CurrentPlayer^.Info.TeamCode,

CurrentPlayer^.Info.Position, CurrentPlayer^.Info.Success,

CurrentPlayer^.Info.PenaltyPoints]);

CurrentPlayer := CurrentPlayer^.Ptr;

Inc(Input2\_local);

end;

OutputContent := OutputContent +

'|---------------------------------------------------------------------------|'

+ #13#10;

Writeln('|-------------------------------------------------------------------------|');

SaveToTextFile('top\_penalty\_players.txt', OutputContent);

Writeln('Результаты сохранены в файл top\_penalty\_players.txt');

end

else

Writeln('Список футболистов пустой');

end;

3: // Самый успешный и по штрафным очкам для каждой команды

begin

OutputContent :=

'Самый успешный игрок и игрок с максимальной суммой штрафных очков для каждой команды:'

+ #13#10#13#10;

if (TeamList^.Ptr <> nil) and (PlayerList^.Ptr <> nil) then

begin

Writeln('Самый успешный игрок и игрок с максимальной суммой штрафных очков для каждой команды:');

CurrentTeam := TeamList^.Ptr;

while CurrentTeam <> nil do

begin

Writeln('Команда: ', CurrentTeam^.Info.Name);

OutputContent := OutputContent + 'Команда: ' +

CurrentTeam^.Info.Name + #13#10;

FoundPlayersForTeam := false;

TempPlayerList := PlayerList^.Ptr;

BestPlayer.Success := -1;

MostPenaltyPlayer.PenaltyPoints := -1;

while TempPlayerList <> nil do

begin

if TempPlayerList^.Info.TeamCode = CurrentTeam^.Info.Code

then

begin

FoundPlayersForTeam := true;

if TempPlayerList^.Info.Success > BestPlayer.Success

then

BestPlayer := TempPlayerList^.Info;

if TempPlayerList^.Info.PenaltyPoints >

MostPenaltyPlayer.PenaltyPoints then

MostPenaltyPlayer := TempPlayerList^.Info;

end;

TempPlayerList := TempPlayerList^.Ptr;

end;

if FoundPlayersForTeam then

begin

Writeln(' Самый успешный игрок:');

WriteInfo(BestPlayer);

OutputContent := OutputContent +

' Самый успешный игрок:' + #13#10;

OutputContent := OutputContent +

Format('|%5d|%-25s|%6d|%-12s|%7d|%13d|' + #13#10,

[BestPlayer.Code, BestPlayer.Name, BestPlayer.TeamCode,

BestPlayer.Position, BestPlayer.Success,

BestPlayer.PenaltyPoints]);

Writeln(' Игрок с максимальными штрафными очками:');

WriteInfo(MostPenaltyPlayer);

OutputContent := OutputContent +

' Игрок с максимальными штрафными очками:' + #13#10;

OutputContent := OutputContent +

Format('|%5d|%-25s|%6d|%-12s|%7d|%13d|' + #13#10,

[MostPenaltyPlayer.Code, MostPenaltyPlayer.Name,

MostPenaltyPlayer.TeamCode, MostPenaltyPlayer.Position,

MostPenaltyPlayer.Success,

MostPenaltyPlayer.PenaltyPoints]);

end

else

begin

Writeln(' В этой команде нет футболистов.');

OutputContent := OutputContent +

' В этой команде нет футболистов.' + #13#10;

end;

Writeln;

OutputContent := OutputContent + #13#10;

CurrentTeam := CurrentTeam^.Ptr;

end;

SaveToTextFile('team\_best\_players.txt', OutputContent);

Writeln('Результаты сохранены в файл team\_best\_players.txt');

end

else

Writeln('Один из списков (команд или футболистов) пустой.');

end;

end;

end;

9:

DisposeLists(PlayerList, TeamList);

10:

SaveToFile(PlayerList, TeamList);

end;

until (Input = 10) or (Input = 9);

end.