

Karts@ISEC

Programação

Nº 21270416

Maria Fonseca Ferreira

# Índice

[Índice 1](#_Toc10411623)

[Estrutura de dados 4](#_Toc10411624)

[Estrutura dos Pilotos 4](#_Toc10411625)

[Estrutura dos Carros 4](#_Toc10411626)

[Estrutura das Equipas 4](#_Toc10411627)

[Estrutura das Corridas 5](#_Toc10411628)

[Estrutura dos Campeonatos 5](#_Toc10411629)

[Estruturas dinâmicas 6](#_Toc10411630)

[Vetor dinâmico dos Pilotos: 6](#_Toc10411631)

[Vetor dinâmico dos Carros: 6](#_Toc10411632)

[Lista Ligada Equipas 6](#_Toc10411633)

[Lista Ligada Corridas 6](#_Toc10411634)

[Lista Ligada Campeonatos 6](#_Toc10411635)

[Ficheiros Utilizados 7](#_Toc10411636)

[carros.c 8](#_Toc10411637)

[int num\_carros(char \*nomeficheiro); 8](#_Toc10411638)

[Esta função abre o ficheiro “carros.txt” para leitura. Caso haja problemas com a abertura do ficheiro, a função termina. Senão começa a contar os caracteres ‘\n’. No fim, incrementa a variável contador mais uma vez pois não existe o caracter ‘\n’ no último carro. Retorna o número de carros. 8](#_Toc10411639)

[carros\* inicializa\_carros(char \*nomeficheiro, int total\_c); 8](#_Toc10411640)

[void listar\_carros(carros \*p, int total\_c); 8](#_Toc10411641)

[void procura\_carro(carros \*p, int total\_c); 8](#_Toc10411642)

[carros\* atualiza\_carros\_fim\_corrida(pequipas eq, carros \*c, int total\_c); 8](#_Toc10411643)

[pilotos.c 8](#_Toc10411644)

[int num\_pilotos(char \*nomeficheiro); 8](#_Toc10411645)

[pilotos\* inicializa\_pilotos(char \*nomeficheiro, int total); 9](#_Toc10411646)

[void listar\_pilotos(pilotos \*p, int total); 9](#_Toc10411647)

[void procura\_piloto(pilotos \*p, int total); 9](#_Toc10411648)

[pilotos\* atualiza\_pilotos\_fim\_corrida(pequipas eq, pilotos \*p, int total\_p); 9](#_Toc10411649)

[pilotos\* adiciona\_penalizacao(pilotos \*p, int total); 9](#_Toc10411650)

[void gravar\_vetorP(pilotos \*p, char \*nomefich, int total) ; 9](#_Toc10411651)

[pilotos\* atualiza\_experiencia\_campeonato(pcampeonato camp, pilotos \*p, int total\_p); 9](#_Toc10411652)

[corrida.c 9](#_Toc10411653)

[int num\_equipas(int num\_max, carros \*c, int total\_c, pilotos \*p, int total\_p); 9](#_Toc10411654)

[Função que retorna o número de pares possível. 9](#_Toc10411655)

[pequipas criaEquipa(pequipas eq, int total\_e, carros \*c, int total\_c, pilotos \*p, int total\_p); 10](#_Toc10411656)

[Função que cria as equipas de forma random. 10](#_Toc10411657)

[void listar\_equipas(pequipas eq); 10](#_Toc10411658)

[pequipas getinfo(pequipas aux, int index); 10](#_Toc10411659)

[Função que retorna informação sobre uma equipa. 10](#_Toc10411660)

[pcorridas adicionaNo(int \*flag, int v,int num\_voltas, int comprimento, int nr\_carros, pequipas infoequipa, int total\_e, pcorridas corrida, carros \*c, int total\_c, pilotos \*p, int total\_p); 10](#_Toc10411661)

[Função que adiciona nó à corrida por cada volta de uma equipa. 10](#_Toc10411662)

[void mostraCorridaCompleta(int num\_voltas, pcorridas corrida, pilotos \*p, int total\_p); 10](#_Toc10411663)

[Função que mostra a classificação final volta a volta e as voltas em que as equipas tenham sofrido um acidente. 10](#_Toc10411664)

[pequipas colocaAcidente(pequipas eq, int corredor); 10](#_Toc10411665)

[Função em que caso uma equipa sofra um acidente durante a corrida será “ativada” a variável *ocorreu\_acidente* na lista ligada *eq*. 10](#_Toc10411666)

[pcorridas atribuiClassificao(int num\_voltas,pcorridas corrida); 10](#_Toc10411667)

[Função que atribui classificação em cada volta. 10](#_Toc10411668)

[void mostraResultadosVolta(int v, pcorridas corrida, pilotos \*p, int total\_p); 10](#_Toc10411669)

[Função que mostra resultados de cada volta durante o decorrer da corrida. 10](#_Toc10411670)

[pequipas calculaTempoFinal( pequipas eq, pcorridas corrida); 10](#_Toc10411671)

[pequipas classificacaoFinal( pequipas eq, pcorridas corrida, int total\_e); 10](#_Toc10411672)

[Função que atribui a classificação final às equipas que participaram na corrida. 10](#_Toc10411673)

[pequipas contaPrimeirosLugares(int num\_voltas, pequipas eq,int total\_e, pcorridas corrida); 10](#_Toc10411674)

[void mostraClassificaoFinal(int num\_voltas, pequipas eq,int total\_e, pcorridas corrida, pilotos \*p, int total\_p); 10](#_Toc10411675)

[void mostraVoltaEspecifica(pcorridas corrida, pilotos \*p, int total\_p); 11](#_Toc10411676)

[Função que mostra uma volta indicada pelo utilizador. 11](#_Toc10411677)

[void liberta(pcorridas corrida); 11](#_Toc10411678)

[pequipas inicializa\_corrida( carros \*c, int total\_c, pilotos \*p, int total\_p); 11](#_Toc10411679)

[utils.c 11](#_Toc10411680)

[void initRandom(); 11](#_Toc10411681)

[int intUniformRnd(int a, int b); 11](#_Toc10411682)

[int probEvento(float prob); 11](#_Toc10411683)

[void espera(unsigned int seg); 11](#_Toc10411684)

[Função que demora aproximadamente seg segundos a executar. 11](#_Toc10411685)

[void obtemData(int \*dia, int \*mes, int \*ano); 11](#_Toc10411686)

[int retornaIdade(int dia, int mes, int ano); 11](#_Toc10411687)

[Retorna idade do piloto. 11](#_Toc10411688)

[int calculaSegundos(int idadeP, int pesoP, float expP, int PotC, int metros); 12](#_Toc10411689)

[void base(); 12](#_Toc10411690)

[void menu(); 12](#_Toc10411691)

[void submenu1(); 12](#_Toc10411692)

[void submenu2(); 12](#_Toc10411693)

[void submenu3(); 12](#_Toc10411694)

[void submenu4(); 12](#_Toc10411695)

[void submenu5(); 12](#_Toc10411696)

[void libertaEquipas(pequipas equipas); 12](#_Toc10411697)

[campeonato.c 12](#_Toc10411698)

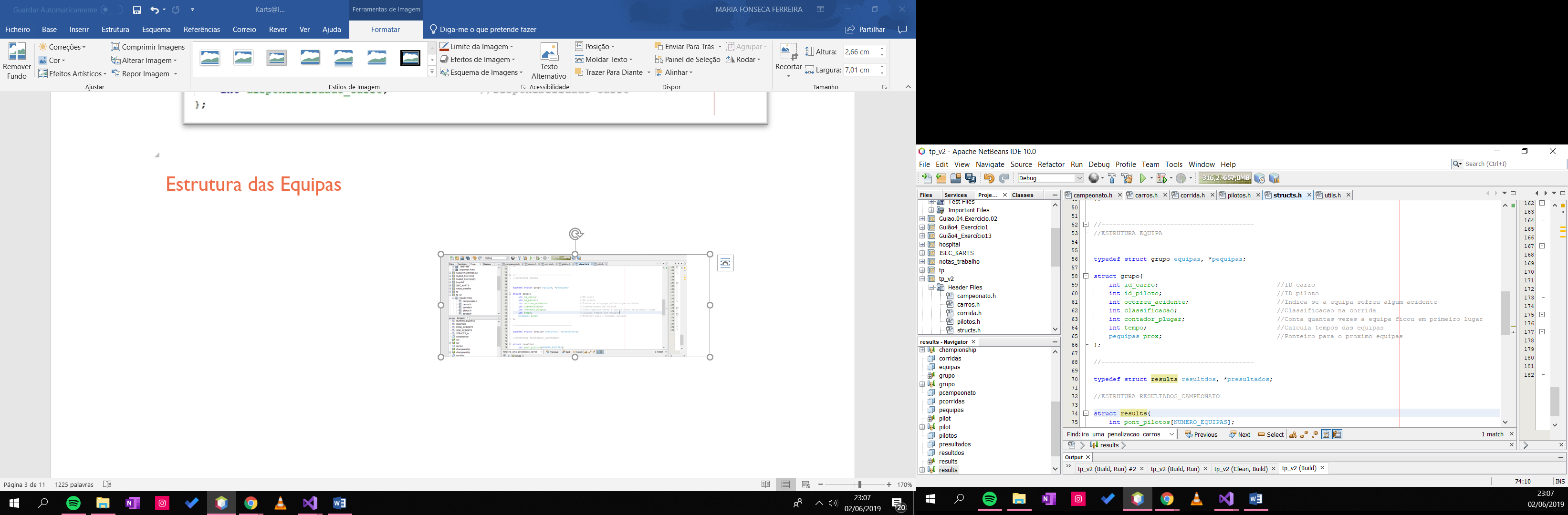
[Opções tomadas em termos de implementação 14](#_Toc10411699)

# Estrutura de dados

## Estrutura dos Pilotos

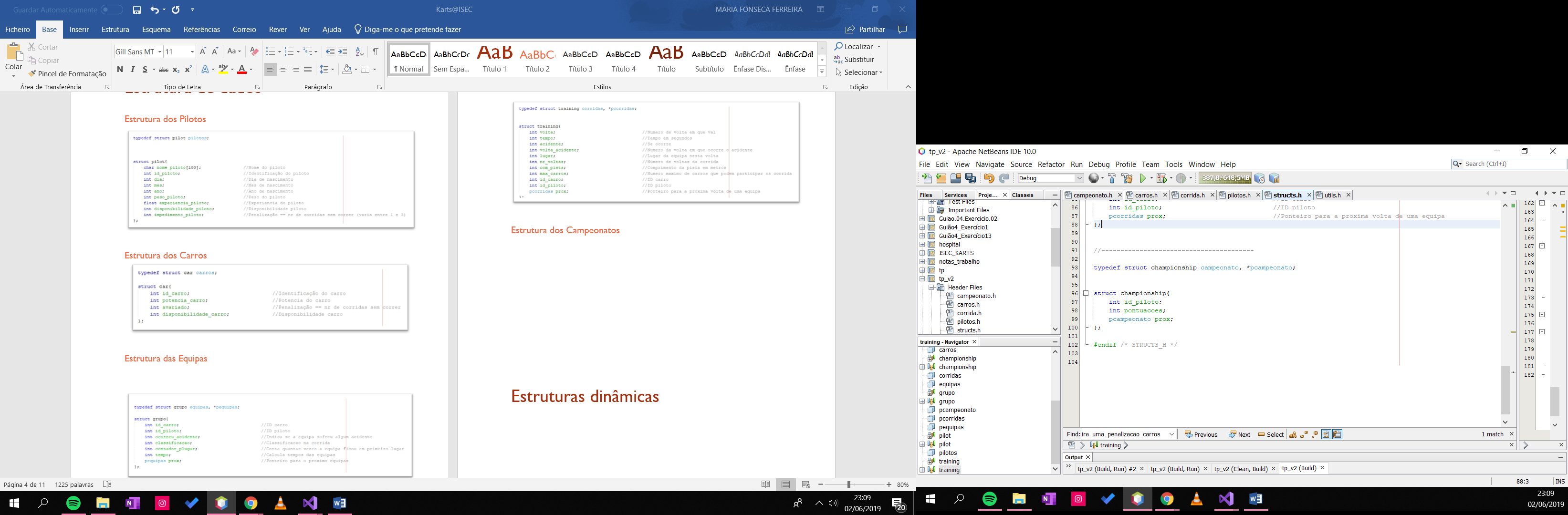
## Estrutura dos Carros

## Estrutura das Equipas



## Estrutura das Corridas

## Estrutura dos Campeonatos



# Estruturas dinâmicas

Neste trabalho prático foram utilizadas 6 estruturas dinâmicas. Como era requerido no enunciado eram necessários dois vetores dinâmicos, um para os pilotos e outro para os carros, uma lista ligada para as corridas e outra para os resultados dos campeonatos. Optei ainda por acrescentar uma lista ligada para as equipas formadas para uma corrida.

## 

## Vetor dinâmico dos Pilotos:

Tem todos os pilotos existentes no ficheiro pilotos.txt, é utilizado para guardar e alterar a informação relativa aos pilotos. Contém o ID, o nome, data de aniversário, peso do piloto, experiencia do piloto, disponibilidade do piloto, e a disponibilidade deste.

## 

## Vetor dinâmico dos Carros:

Tem todos os pilotos existentes no ficheiro carros.txt, é utilizado para guardar e alterar a informação relativa aos carros. Contém o ID, a potência, o número de voltas que o carro está impedido de correr e a disponibilidade deste.

## Lista Ligada Equipas

Tem as equipas que efetuam uma corrida, é utlizado como meio de troca de atualizações de informações. Contem o ID do carro e do piloto, indicação se a equipa sofreu algum acidente, conta em quantas voltas a equipa ficou em primeiro lugar, o somatório do tempo da corrida e a classificação na corrida.

## Lista Ligada Corridas

Tem toda a informação sobre uma corrida, cada nó corresponde à volta de uma equipa. Contém o número da volta, de voltas, o tempo com que a equipa concluiu a volta em segundos, se ocorreu um acidente nessa volta, o número da volta do acidente, a classificação da equipa nessa volta, comprimento da pista, número máximo de carros, o ID do carro e do piloto.

## Lista Ligada Campeonatos

Utilizada para guardar a informação sobre a pontuação de cada piloto no Campeonato. Contém apenas o ID do piloto e a sua pontuação.

# Ficheiros Utilizados

**main.c:** contém as funções de inicialização, bem como o processamento do menu.

**utils.c:** processamento de menus, código disponibilizado e libertação de memória.

**utils.h:** ficheiro header que faz a ligação do ficheiro que contém as funções do ficheiro utils.c para outros ficheiros.

**structs.h:** inicializa as estruturas relevantes ao programa.

**pilotos.c:** processamento da informação relativa aos pilotos.

**pilotos.h:** ficheiro header que faz a ligação do ficheiro que contém as funções dos pilotos para outros ficheiros.

**carros.c:** processamento da informação relativa aos carros.

**carros.h:** ficheiro header que faz a ligação do ficheiro que contém as funções dos carros para outros ficheiros.

**corrida.c:** processamento da informação relativa às corridas.

**corrida.h:** ficheiro header que faz a ligação do ficheiro que contém as funções da corrida para outros ficheiros.

**campeonato.c:** processamento da informação relativa ao campeonato.

**campeonato.h:** ficheiro header que faz a ligação do ficheiro que contém as funções do campeonato para outros ficheiros.

**carros.txt:** ficheiro de texto que contém informação relativa aos carros.

**pilotos.txt:** ficheiro de texto que contém informação relativa aos pilotos.

**campeonato.dat:** ficheiro binário que contém informação relativa ao campeonato.

## carros.c

### int num\_carros(char \*nomeficheiro);

### Esta função abre o ficheiro “carros.txt” para leitura. Caso haja problemas com a abertura do ficheiro, a função termina. Senão começa a contar os caracteres ‘\n’. No fim, incrementa a variável contador mais uma vez pois não existe o caracter ‘\n’ no último carro. Retorna o número de carros.

### carros\* inicializa\_carros(char \*nomeficheiro, int total\_c);

A função começa por abrir o ficheiro “carros.txt”. Caso haja problemas

com a abertura do ficheiro, a função termina e retorna NULL. Senão, começa a

contruir o vetor dinâmico. Primeiro, cria um espaço de memória com o tamanho

de uma estrutura de carros multiplicado pela variável total. Se correr mal, a

função termina, fecha o ficheiro e apresenta uma mensagem de erro. Senão,

começa a ler do ficheiro as informações sobre os carros e a preencher no vetor. Retorna o vetor.

### void listar\_carros(carros \*p, int total\_c);

Função que lista toda a informação sobre os pilotos.

### void procura\_carro(carros \*p, int total\_c);

Função que mostra toda a informação sobre o carro procurado pelo utilizador. Caso esse carro não exista o utilizador é informado.

### carros\* atualiza\_carros\_fim\_corrida(pequipas eq, carros \*c, int total\_c);

Função que atualiza toda a informação (acidentes) sobre os carros no final de uma corrida.

## pilotos.c

### int num\_pilotos(char \*nomeficheiro);

Esta função abre o ficheiro “pilotos.txt” para leitura. Caso haja problemas com a abertura do ficheiro, a função termina. Senão começa a contar os caracteres ‘\n’. No fim, incrementa a variável contador mais uma vez pois não existe o caracter ‘\n’ na ultima guitarra. Retorna o número de pilotos.

### pilotos\* inicializa\_pilotos(char \*nomeficheiro, int total);

A função começa por abrir o ficheiro “pilotos.txt”. Caso haja problemas

com a abertura do ficheiro, a função termina e retorna NULL. Senão, começa a

contruir o vetor dinâmico. Primeiro, cria um espaço de memória com o tamanho

de uma estrutura de pilotos multiplicado pela variável total. Se correr mal, a

função termina, fecha o ficheiro e apresenta uma mensagem de erro. Senão,

começa a ler do ficheiro as informações sobre os pilotos e a preencher no vetor. Retorna o vetor.

### void listar\_pilotos(pilotos \*p, int total);

Função que lista toda a informação sobre os pilotos.

### void procura\_piloto(pilotos \*p, int total);

Função que mostra toda a informação sobre o piloto procurado pelo utilizador. Caso esse piloto não exista o utilizador é informado.

### pilotos\* atualiza\_pilotos\_fim\_corrida(pequipas eq, pilotos \*p, int total\_p);

Função que atualiza toda a informação (experiência e impedimentos) sobre os pilotos no final de uma corrida.

### pilotos\* adiciona\_penalizacao(pilotos \*p, int total);

Função que permite ao utilizador adicionar penalização a piloto.

### void gravar\_vetorP(pilotos \*p, char \*nomefich, int total) ;

Função que guarda o vetor dinâmico dos pilotos no ficheiro “pilotos.txt”.

### pilotos\* atualiza\_experiencia\_campeonato(pcampeonato camp, pilotos \*p, int total\_p);

Função que adiciona ao piloto vencedor do campeonato mais 10 pontos de experiência.

## corrida.c

### int num\_equipas(int num\_max, carros \*c, int total\_c, pilotos \*p, int total\_p);

### Função que retorna o número de pares possível.

### pequipas criaEquipa(pequipas eq, int total\_e, carros \*c, int total\_c, pilotos \*p, int total\_p);

### Função que cria as equipas de forma random.

### void listar\_equipas(pequipas eq);

Função que lista as equipas formadas.

### pequipas getinfo(pequipas aux, int index);

### Função que retorna informação sobre uma equipa.

### pcorridas adicionaNo(int \*flag, int v,int num\_voltas, int comprimento, int nr\_carros, pequipas infoequipa, int total\_e, pcorridas corrida, carros \*c, int total\_c, pilotos \*p, int total\_p);

### Função que adiciona nó à corrida por cada volta de uma equipa.

### void mostraCorridaCompleta(int num\_voltas, pcorridas corrida, pilotos \*p, int total\_p);

### Função que mostra a classificação final volta a volta e as voltas em que as equipas tenham sofrido um acidente.

### pequipas colocaAcidente(pequipas eq, int corredor);

### Função em que caso uma equipa sofra um acidente durante a corrida será “ativada” a variável *ocorreu\_acidente* na lista ligada *eq*.

### pcorridas atribuiClassificao(int num\_voltas,pcorridas corrida);

### Função que atribui classificação em cada volta.

### void mostraResultadosVolta(int v, pcorridas corrida, pilotos \*p, int total\_p);

### Função que mostra resultados de cada volta durante o decorrer da corrida.

### pequipas calculaTempoFinal( pequipas eq, pcorridas corrida);

Função que calcula a soma individual de tempos das equipas na corrida.

### pequipas classificacaoFinal( pequipas eq, pcorridas corrida, int total\_e);

### Função que atribui a classificação final às equipas que participaram na corrida.

### pequipas contaPrimeirosLugares(int num\_voltas, pequipas eq,int total\_e, pcorridas corrida);

Função que conta quantas vezes uma equipa ficou em primeiro lugar durante uma corrida.

### void mostraClassificaoFinal(int num\_voltas, pequipas eq,int total\_e, pcorridas corrida, pilotos \*p, int total\_p);

Função que mostra a classificação final e as voltas em que as equipas tenham sofrido um acidente.

### void mostraVoltaEspecifica(pcorridas corrida, pilotos \*p, int total\_p);

### Função que mostra uma volta indicada pelo utilizador.

### void liberta(pcorridas corrida);

Quando a lista corrida já não é necessária, o espaço ocupado pelos seus elementos é libertado.

### pequipas inicializa\_corrida( carros \*c, int total\_c, pilotos \*p, int total\_p);

Função que efetua uma corrida. Começa por pedir ao utilizador informações sobre a corrida como o número de voltas, o comprimento da pista e o numero máximo de carros que podem participar. De seguida obtém-se o número de equipas possível formar, formam-se e listam-se as equipas. Inicia a corrida numa lista ligada onde cada nó corresponde a cada volta de uma equipa. É atribuída e mostrada a classificação de cada equipa volta a volta e a suma da corrida.

## utils.c

### void initRandom();

Inicializa o gerador de números aleatórios.

### int intUniformRnd(int a, int b);

Devolve um valor inteiro aleatório distribuído uniformemente no intervalo [a, b].

### int probEvento(float prob);

Devolve o valor 1 com probabilidade prob. Caso contrário, devolve 0.

### void espera(unsigned int seg);

### Função que demora aproximadamente seg segundos a executar.

### void obtemData(int \*dia, int \*mes, int \*ano);

Obtém e devolve a data e hora atuais.

### int retornaIdade(int dia, int mes, int ano);

### Retorna idade do piloto.

### int calculaSegundos(int idadeP, int pesoP, float expP, int PotC, int metros);

Calcula e devolve o número de segundos por volta de acordo com as informações recebidas por parâmetro.

### void base();

Imprime titulo.

### void menu();

Imprime o menu.

### void submenu1();

Imprime o menu dos pilotos.

### void submenu2();

Imprime o menu dos carros.

### void submenu3();

Imprime o menu dos carros.

### void submenu4();

Imprime o menu das corridas.

### void submenu5();

Imprime o menu do campeonato.

### void libertaEquipas(pequipas equipas);

Quando a lista equipas já não é necessária, o espaço ocupado pelos seus elementos é libertado.

## campeonato.c

void listar\_campeonato(pcampeonato camp, pilotos \*p, int total\_p);

Função que mostra as pontuações de todos os pilotos no final de um campeonato.

pcampeonato adicionaNoCampeonato(pcampeonato camp, int idP);

Função que adiciona um um piloto à lista ligada do Campeonato.

pequipas inicializa\_campeonato( int nrcorrida, int total\_corridas, carros \*c, int total\_c, pilotos \*p, int total\_p);

Função que efetua uma corrida de um campeonato. Começa por pedir ao utilizador informações sobre a corrida como o número de voltas, o comprimento da pista e o número máximo de carros que podem participar. De seguida obtém-se o número de equipas possível formar, formam-se e listam-se as equipas. Inicia a corrida numa lista ligada onde cada nó corresponde a cada volta de uma equipa. É atribuída e mostrada a classificação de cada equipa volta a volta e a suma da corrida.

pequipas calculaTempo(pequipas eq,pcorridas corrida);

Função que soma os tempos de cada equipa numa corrida.

pcampeonato atualizaPontuacao(pcampeonato camp, pequipas eq, int total\_p);

Função chamada no final de cada corrida de um campeonato.

pcampeonato iniciaCampeonato(pcampeonato camp, pilotos \*p, int total\_p);

Função que inicia a lista ligada de pontuações.

void libertaCampeonato(pcampeonato camp);

Quando a lista campeonato já não é necessária, o espaço ocupado pelos seus elementos é libertado.

# Opções tomadas em termos de implementação

Uma das decisões mais importantes tomadas neste trabalho prático foi colocar as equipas em uma lista ligada. A opção foi tomada para que eu pudesse alterar valores mais facilmente. Serve como um intermediário e permite alterar de forma simples os valores dos pilotos, carros, corridas e campeonatos. A lista ligada contém as equipas que efetuam uma corrida, contém o ID do carro e do piloto, indicação se a equipa sofreu algum acidente, conta em quantas voltas a equipa ficou em primeiro lugar, o somatório do tempo da corrida e a classificação na corrida.