ISEC

Relatório do projeto de P 2019

João Pedro Neves Gonçalves Nº 21280302 Turma P7

Estruturas

Pilotos

```
typedef struct piloto Pi, *pPi;
// defenição do piloto
struct piloto {
 // identificação basica
 char nome[100]; // Sequência com tamanho máximo de 100 caracteres
 int Id; // valor inteiro positivo que deve ser único entre todos os pilotos

 // data de nascimento
 int Dnas; // dia de nascimento
 int Mnas; // mes de nascimento
```

```
 int Anas; // ano de nascimento

 // caracteristicas
 int peso; // peso do piloto;valor positivo
 float
     exp; // experiencia do piloto; valor real superior ou igual a 0.0.
          // Quando um novo piloto é criado, a sua experiência tem o valor 0.0
 int imp; // Impedimento: um piloto pode estar impedido de participar em
               // corridas devido a lesão ou penalização. Um piloto fica
               // magoado quando tem um acidente. Nessa altura deverá 1 ficar 2
               // corridas sem competir (podem ser corridas individuais ou
               // parte de campeonato).  Pode, além disso, ser castigado devido
               // a comportamento incorreto. A penalização varia entre 1 e 3
               // corridas sem competir
};
```

Estrutura dos pilotos para guardar os pilotos vindo dos ficheiro.

Carros

```
typedef struct piloto Pi, *pPi;
// defenição do carro
struct carro {
```

```
 int Id;  // id do carro;valor inteiro positivo que deve ser único entre todos
          // os carros

 int pot; // potencia do carro; valor inteiro positivo

 int avar; // se o carro estiver variado 1, se nao 0; um carro pode ou não
           // estar avariado. Um carro fica avariado quando está envolvido num
           // acidente. Depois disso fica 1 corrida sem poder participar
};
```

Estrutura dos carros para guardar os carros vindo dos ficheiros.

SaveS

```
typedef struct saverS sS, *psS;
// saverS -é uma estrutura para guardar os ponteiros e valores para todas as
// variaveis importantes do programa, assim em vez de se passar todos os valores
// como argumentos passa se como uma unica estrutura para simplificar o numero
// de argumentos que as funçoes tem e poder se buscar o que seja necessario para
// cada função

struct saverS
{
 pPi pPilotos; // ponteiro para os pilotos
 int nPilotos; // numero de pilotos
```

```
 pCar pCarros; // ponteiro para os carros
 int nCarros;  // numero de carros
};
```

PilCam

```
typedef struct pilcam Cam, *pCam;
//Pilotos do Campeonato(pilcam)-vai buscar todos os pilotos na estrutura e guarda o seu index
//para guardar os seus valores perante o campeonato em que esta a decorrer.
struct pilcam{

   int piloto;//index do piloto
   int nCorridas;//numero de corridas em que correu

   float pontos;//pontos acumulados

   int correr; //esta a correr, 1 se esta a correr na corrida actual ou 0 se não estiver a correr
   int acidente;//se ouve acidente durane a corrida
   float gainpts;//pontos ganhos numa corrida por volta

   pCam prox;//ponteiro para o anterior
   pCam ant;//ponteiro para o seguinte
};
```

Funções

```
//funcção inicial do programa
psS inicializa()

//le o os pilotos no ficheiro
int readPilotos(char *file, pPi *pPilotos, int *nPilotos) ;

//verifica os pilotos do ficheiro
int verificaPiloto(Pi piloto, int n, int piId[]);

//sala a struct de pilotos no ficheiro apropriado
int salvaP(psS saveS) ;

//printa a estrura inteira dos pilotos
void printPi(psS saveS);

/menu principal do programa
int mainmenu(psS saveS);

//menu inicial para uma corrida
int corridamenu(psS saveS, int voltas, int comp, int MaxAll);
```

```
//adiciona uma estrutura na lista ligada de competidor de uma corrida
pCon adicionaCon(pCon inicio, int piloto, int carro);

//ajustar incrementa o numero de


//menu para ver as pontuações ou sair
int menufinalcor(psS saveS, pCon combina, int nMaxP, int voltas);

//apaga as estruturas principais do saveS e das estruturas dos pilotos e carros
void freeall(psS saveS);

//ver as pontoações com detalhe
int vercorrida(psS saveS,pCon combina, int nMaxP, int voltas);

//função docampeonato
int campeonatomenu(psS saveS);

//corre para selecionar os participantes
pCam selectPil(psS saveS);

//o campeonato recebe como input o saveS(já explicado), part(a lista ligada os pilotos
participantes),numdone(numero de corridas já corridas),numall(numero de todas as corridas), voltas por
corrida, comprimento da pista, e quantos participantes podem correr
int campeonato(psS saveS, pCam part, int numdone, int numall, int voltas, int comp, int maxpart);
```

```c
//meter os participantes no ficheiro binario
int gravaBi(pCam part, int numdone, int numall, int voltas, int comp, int maxpart);

//ler o ficheiro binario
int lerBi(psS saveS);

//le o ficheiro dos carros e mete los numa estrutura
int readCar(char *file, pCar *pCarros, int *nCarros);

//verifica os valoros na estrutura dos carros
int verificaCarro(Car carro, int n, int carId[]);

//salva a estrutura num ficheiro apropriado
int salvaC(psS saveS);

//printa a estrura inteira dos carros
void printCar(psS saveS) ;

//faz a estrutura geral
psS makegeral(pPi pilotos, int npilotos, pCar carros, int ncarros);

// calcular idade
// indexpiloto- index de onde esta o piloto no array dos pilotos
int calIda(psS saveS, int indexpiloto);

//seleciona os Carros e os pilotos de forma aleatoria
```

```
pCon selCarPil(int nMaxP, psS saveS);

//apaga um membro da lista ligada combina(Con)
pCon delCon(pCon dels);

//gotohead vai para a cabeça(inicio) da lista
pCon gth(pCon combina);

//printa a posição de que os pilotos estao diacordo com a lista ligada
//espera - esta variavel só deve ser metida no final da corrida para fazer esperar e calcular o XP
void verPos(pCon combina, psS saveS, int voltastotal, int voltaact, int esperar);

//função para limpar a memoria
void freecorr(pCon combina);

//muda a experiencia nos corredores
void calPontos(psS saveS, pCon combina);

//substitui o "total" em todos os constintuintes da lista ligada "combina" pelo tempo total que fizeram ate a
"volta" x
pCon totalde(pCon combina,int volta);

//algoritmo da corrida
pCon fazercorrida(psS saveS, pCon combina, int voltas, int comp,int nMaxP, pCam campeonato);

//adiciona os pontos para os participantes do campeonato
```

```
void adicionapontos(pCon combina, pCam part);

//novo algoritmo para ordenar a lista ligada
pCon ordterm2(pCon inicio);

//remover penalizaçoes
void rempen(psS saveS);

//apaga um valor de uma lista
int *movete(int *array, int value, int max);

//trocas os conteodos de duas estruturas do tipo combina
void swap(pCon finder) ;

//erdena a lista ligada combina
pCon bubbleSort(pCon start) ;

//conrrige os pontos para o anterior
void fixant(pCon start);
```

Como funciona

Este é um programa de simulação de corridas, ele pode ser compilado em gcc com o comando

gcc main.c campeonato.c carros.c corrida.c menus.c pilotos.c utils.c -o run -std=c99

(é necessario o gcc para executar este comando)

e corrido com o executável run que se pode ser declarado como executavel com o comando

chmod +x run

e executado num terminal bash com

./run

Na execução teremos menus e opções para navegar por ele, o menu inicial está escondido para ocupar menos espaço mas pode ser mostrado digitando 5 e clicando Enter, pode se sair para trás de qualquer menu digitando 0 e depois enter.