Programação

Trabalho Prático

Licenciatura em Engenharia Informática: 1º ano - 2º semestre 2018/2019

Karts@ISEC

Notas prévias:

- O enunciado é possivelmente vago, genérico e incompleto em alguns pontos. O que se pretende é que os alunos avaliem as opções existentes e escolham a que considerarem mais apropriada para cada uma das situações com que se depararem. Todas as escolhas devem ser referidas e justificadas no relatório.
- O programa deve ser implementado em C *standard*, *i.e.,* não deve ter instruções que o tornem específico para um determinado ambiente/plataforma de desenvolvimento. Deverá ser respeitada a norma C99.
- O programa entregue deve ter uma interface simples e amigável, indicando o que pode ser feito em cada situação. Não são valorizados programas com interfaces gráficos.
- Deve distribuir o código fonte por vários ficheiros. Para além do ficheiro com código disponibilizado com este enunciado, deverão existir, no mínimo, outros dois ficheiros com código fonte.
- Deverá utilizar *header files* para gerir as dependências entre os vários ficheiros com código fonte.
- Todos os ficheiros devem ter a identificação do aluno (nome completo e número), em comentário, nas linhas iniciais dos ficheiros.

1. Introdução

Pretende-se criar um programa para efetuar a simulação completa de um campeonato de corridas de automóveis. A aplicação gere um conjunto de pilotos e um conjunto de automóveis. Além disso deve permitir simular a realização de corridas em pista. As corridas podem ser individuais, servindo apenas para treino, ou fazer parte de um campeonato. Neste último caso, os pilotos vão acumulando pontuação ao longo das corridas e, no final do campeonato, o que tiver mais pontos é declarado vencedor.

O programa deve manter um registo informático dos pilotos e automóveis existentes. Deve além disso, gerir a realização de corridas individuais e de campeonatos.

2. Pilotos e Carros

O programa deve armazenar e gerir a informação seguinte:

2.1 Pilotos

- Nome: Sequência com tamanho máximo de 100 caracteres
- Id: valor inteiro positivo que deve ser único entre todos os pilotos
- Data de Nascimento
- Peso: valor positivo
- Experiência: valor real superior ou igual a 0.0. Quando um novo piloto é criado, a sua experiência tem o valor 0.0
- 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

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

2.2 Carros

- Id: valor inteiro positivo que deve ser único entre todos os carros
- Potência: valor inteiro positivo
- Avariado: um carro pode ou não estar avariado. Um carro fica avariado quando está envolvido num acidente. Depois disso fica 1 corrida sem poder participar

Quando o programa não está a ser executado, a informação de pilotos e carros está armazenada em 2 ficheiros de texto, **obrigatoriamente com o formato descrito a seguir.**

No ficheiro de pilotos, duas linhas consecutivas armazenam a informação de cada piloto, surgindo na primeira o seu nome e na seguinte o resto da informação (pela ordem em que surge na lista anterior). Existe uma linha em branco entre pilotos. A seguir mostra-se um exemplo para um ficheiro com 4 pilotos.

Exemplo de um ficheiro de texto com pilotos

```
Paulo Andrade
2  23  12  1995  76  0.5  0

Faisca
3   1   1  1980  50  9.5  1

Diana Alves Pombo
4  1  10  1990  55  4.5  0

Ana Luisa Freitas
7  12  7  1976  68  1.0  3
```

O último valor da segunda linha relativa a cada piloto é um contador que indica quantas corridas ele vai estar sem competir. No exemplo em cima, há um piloto que não pode competir na próxima corrida e outro que vai estar 3 corridas parado. Este contador resulta dos impedimentos descritos em cima (lesão ou castigo).

No ficheiro de carros existe uma linha para cada viatura, surgindo a informação pela ordem apresentada na lista anterior. Existe uma linha em branco entre carros. A seguir mostra-se um exemplo para um ficheiro com 3 carros.

Exemplo de um ficheiro de texto com 3 carros

```
2  120   1

3  220   0

4  90    0
```

O último valor é um contador que indica durante quantas corridas o carro não vai poder competir devido a avaria.

No início da execução, a informação de pilotos e carros é obrigatoriamente transferida para **vetores dinâmicos de estruturas**. Caso a informação não exista, ou esteja num formato inválido, deverá ser apresentada uma mensagem adequada e o programa termina imediatamente.

Durante a execução, o programa deverá atualizar a informação dos pilotos e/ou carros presente nos vetores dinâmicos, em função das penalizações impostas aos pilotos e das corridas que se forem realizando. **Os ficheiros de texto devem ser atualizados apenas no final da execução do programa**.

O programa não tem operações para adicionar ou eliminar pilotos ou viaturas. Todas estas operações poderão ser feitas diretamente no ficheiro de texto, apenas quando o programa não está a ser executado. Durante a realização de um campeonato não é possível adicionar / eliminar carros ou pilotos. Pode assumir que essas operações não são efetuadas nos ficheiros durante o decurso do campeonato.

2.3 Gestão dos Impedimentos de Pilotos / Avarias de Carros

Um piloto pode estar impedido de participar numa corrida devido a acidente ou penalização. Se no decurso de uma corrida tiver um acidente, o valor do seu impedimento toma o valor 2. Além disso, pode ser penalizado por comportamento incorreto. Esta penalização é atribuída pelo utilizador do programa, dado o Id do piloto, e tem um valor entre 1 e 3 (a somar ao valor do impedimento que o piloto tenha nessa altura).
O valor do impedimento de um piloto vai sendo decrementado em uma unidade por cada corrida em que ele não participa.

A avaria de um carro resulta de um acidente e toma sempre o valor 1, impedindo que ele participe na próxima corrida.

3. Realização de Corridas

3.1 Corridas Individuais / Treino

Antes do início de uma corrida individual devem ser definidas as suas características, nomeadamente:

- Número de voltas: valor inteiro entre 5 e 10
- Comprimento da pista em metros: valor inteiro entre 500 e 1000
- Número máximo de carros que podem participar (capacidade da pista)

Depois das definições, deverão ser associados carros não avariados a pilotos que não estejam impedidos de participar[1]. A atribuição é feita de forma aleatória e o programa deverá fazer a gestão correta de todas as condicionantes existentes para a realização de cada corrida: por exemplo poderão existir mais carros disponíveis do que o número que pode participar na corrida e nesse caso será necessário não considerar alguns; ou poderão existir mais pilotos do que carros e será preciso escolher aleatoriamente os pilotos que irão correr.
Antes do início da corrida, o programa mostra os emparelhamentos feitos. Deve mostrar igualmente quais os pilotos e carros que não vão participar, indicando o motivo.

Um par carro/piloto demora um determinado número de segundos em cada volta. Esta duração tem em consideração os seguintes fatores: idade, peso e experiência do condutor, potência do carro e comprimento da pista. A função estocástica *calculaSegundos()* está disponível no ficheiro de código *utils.c* (ver secção 4) e, dadas estas características, devolve o número de segundos para uma volta.

Em cada volta, cada par piloto/carro tem uma probabilidade de 5% de ter um acidente. Um acidente implica que o par piloto/carro desiste da corrida e tem a penalização indicada anteriormente.

[1] O contador dos pilotos impedidos e dos carros avariados deve ser atualizado nesta altura.

No final de cada volta, o programa mostra a classificação nessa altura. A listagem deve incluir os pilotos que já desistiram. Deve estar ordenada pela classificação atual, incluir a identificação do par piloto/carro e mostrar detalhadamente o número de segundos que cada equipa demorou em cada uma das voltas já realizadas. No caso de desistência, deve surgir em que volta se verificou o abandono da prova. A seguir mostra-se um exemplo de uma possível classificação no final da terceira volta de uma corrida.

Classificação no Final da Terceira Volta
1º: Paulo Andrade (Id: 2) / Carro 4: 50 + 75 + 66: 191 segundos
2º: Diana Alves Pombo (Id: 4) / Carro 3: 49 + 80 + 65: 194 segundos
...

Durante a corrida, o programa deve ir mostrando as classificações em cada volta de forma sequencial, sem necessitar de intervenção do utilizador. Deve existir um intervalo de 5 segundos entre a visualização da classificação de 2 voltas consecutivas. No final da prova é mostrada a classificação final.

Depois de terminada a corrida, o utilizador pode **rever a visualização das classificações detalhadas**. Devem existir, pelo menos, 2 formas de revisão da visualização:

- Visualização da classificação detalhada volta a volta
- Visualização detalhada da classificação numa volta específica, indicada pelo utilizador

Numa corrida individual os pilotos não recebem pontuação, mas a sua experiência é atualizada de acordo com as seguintes regras:

- Se terminaram a corrida, acumulam 0.5 pontos de experiência por cada volta em que se encontravam em primeiro
- Se desistiram, perdem 1 ponto de experiência (desde que ela não fique negativa)

Toda a informação relativa à gestão da informação durante a preparação e decorrer de uma corrida tem obrigatoriamente que ser armazenada em **memória dinâmica do tipo lista ligada**.

3.2 Campeonato

Quando o programa inicia o modo campeonato terão que ser definidas quantas corridas irão ser realizadas. Um campeonato tem sempre entre 3 e 8 provas. As corridas não terão que ser feitas numa única execução do programa. Podem ser feitas, por exemplo, 2 corridas e o programa termina. Durante a interrupção, toda a informação relevante é guardada em **ficheiros binários**. Na próxima execução do programa, o campeonato recomeça no ponto em que foi interrompido. Durante o modo campeonato não há corridas individuais.

A realização de cada uma das corridas do campeonato segue as mesmas regras definidas anteriormente, no que diz respeito, às suas características, associação piloto/carro e processamento das voltas.
No final de uma corrida do campeonato, os pilotos acumulam experiência de acordo com as regras anteriores: Para efeitos do campeonato, são pontuados da seguinte forma:

- Pontuações 5, 4, 3, 2, 1 para os primeiros cinco classificados
- Além disso recebem mais 0.5 pontos por cada volta que terminaram em primeiro

Só pilotos que terminaram a corrida é que recebem pontuação.

Durante o modo campeonato, a classificação dos pilotos é obrigatoriamente mantida numa **estrutura dinâmica do tipo lista ligada**. Esta lista deve estar ordenada pela pontuação que eles tenham nessa altura. O primeiro critério de desempate é o número de corridas em que participaram e o segundo critério de desempate é a idade do piloto.

No final do campeonato, o piloto campeão acumula 10 pontos de experiência.

4. Código Disponibilizado

O ficheiro *utils.c* contém algumas funções auxiliares que podem ser úteis durante a implementação do trabalho:

void initRandom();

Inicializa o gerador de números aleatórios. Deve ser chamada apenas uma vez no início da execução do programa.

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, int *h, int *m, int *s);*

Obtém e devolve a data e hora atuais

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.

O ficheiro *utils.c* contém igualmente uma função *testes()* que serve apenas para ilustrar alguns exemplos de chamadas das funções disponibilizadas.

5. Normas para a realização do trabalho prático

O trabalho deve ser **realizado individualmente**. O trabalho só pode ser entregue uma vez e numa única data. A nota obtida é válida em todas as épocas de avaliação do ano letivo 2018/2019.

Data limite de entrega do trabalho prático: **23.55** do dia **2 de Junho de 2019**.

Material a entregar:

- **Até às 23.55 do dia 02/06/2019**: entregar via moodle um ficheiro compactado em formato ZIP, contendo o relatório, o código fonte comentado e os ficheiros de dados necessários para o funcionamento do programa. Caso tenham usado o Netbeans ou o CodeBlocks para a implementação, deverão incluir no ZIP todos os ficheiros do respetivo projeto.
- O nome do ficheiro ZIP deve obrigatoriamente ter o seguinte formato: Prog_pX_NumAluno.zip, em que X é a identificação da turma prática que frequenta.

- **No início da defesa**: entregar, ao professor responsável pela defesa, uma cópia impressa do relatório. O conteúdo do documento deve ser igual ao submetido via moodle.

Defesa:
Os trabalhos serão sujeitos a **defesa obrigatória** em data a anunciar. As defesas poderão incluir:

i) Demonstração do funcionamento do programa
ii) Explicação detalhada do código
iii) Implementação de alterações / novas funcionalidades

As defesas serão feitas nos computadores dos laboratórios, utilizando o Netbeans ou o Codeblocks em ambiente Windows. Antes da submissão deverão certificar-se que o trabalho submetido está pronto para correr num destes ambientes.

Relatório

Deve ser entregue um relatório contemplando os seguintes pontos:

- Apresentação das principais estruturas de dados, justificando as escolhas feitas
- Apresentação detalhada das estruturas dinâmicas implementadas
- Descrição dos ficheiros utilizados
- Justificação para as opções tomadas em termos de implementação
- Pequeno manual de utilização.

Avaliação

A cotação do trabalho é de **7 valores**.

Esta componente da avaliação não tem nota mínima.

A deteção de plágio parcial ou total implica a anulação imediata de todos os trabalhos envolvidos.

Cotações

- Funcionalidades da Secção 2: 25%
- Funcionalidades da Secção 3.1: 35%
- Funcionalidades da Secção 3.2: 30%
- Organização de Código: 5%
- Relatório: 5%

Critérios de Avaliação para as Funcionalidades Implementadas

- Definição das estruturas de dados
- Correção das funcionalidades implementadas
- Manipulação de estruturas dinâmicas
- Manipulação de ficheiros
- Simplicidade/funcionalidade do interface com o utilizador