Trabalho Prático de TPI 2005/2006

Paulo Cortez

2005/2006

Robot Race

Licenciatura em Engenharia Electrónica e Computadores

Enunciado: **Robot Race**

- A empresa *EEIC*[1], da qual a sua equipa faz parte, foi contratada para desenvolver um programa para gerir uma nova modalidade: as corridas de robôs **Robot Race**;
- À sua equipa foi alocado o módulo de gestão e simulação das corridas **Robot Race**, a ser desenvolvido na linguagem de programação de **C++**, através do programa MinGW.

[1] Engenheiros Espertos, Inventivos e Criadores.

Requisitos

- No robodomo existem diversas provas ao longo do ano. Em cada prova, é importante saber o estado do tempo (bom, chuva ou neve), a data de realização (por exemplo: 12/7/2005), bem como a classificação final de todos os robôs;
- Em cada corrida, participam todos os robôs disponíveis, sendo que no mínimo têm de existir 2 concorrentes. O campeonato decide-se ao longo de um número variável de corridas. Em cada corrida, uma vitória conta 10 pontos, o segundo lugar 9, etc... , até ao 10º classificado, que recebe um ponto;
- Cada robô tem um código com 2 algarismos, uma cor (azul, vermelho ou branco), potência (em cavalos) e peso (em kilogramas);

Requisitos (continuação)

- As robomotas têm apenas 2 rodas, um patrocinador (por exemplo: "Tintas Cin"), e combustível de "mistura", enquanto os robocarros e robocamiões usam "diesel";
- As robosides, são robomotas especiais com "sidekick", tendo por isso 3 rodas e um peso extra correspondente ao "sidekick";
- Ambos os robocarros e robocamiões têm 4 rodas. Contudo, os robocarros são mais baratos, pelo que têm só 2 patrocinadores, enquanto que os robocamiões têm 3. Os robocamiões são também mais pesados, pois além do peso da parte dianteira, é necessário saber o peso do atrelado;
- Cada marca (por exemplo "Robferrari") participa com um máximo de 2 robôs;

Requisitos (continuação)

- Na simulação de uma corrida, a classificação final de cada robô deve ser gerada de modo aleatório;

```
#include <cstdlib> // acesso a RAND_MAX
#include <ctime>   // acesso a time(...)
int main() // números aleatórios em C++:
{ int r;
  srand((unsigned)time(0));// usar só uma vez
  r=rand(); // int aleatório entre 0 e RAND_MAX
  r= (rand()%10)+1; // aleatório entre 1 e 10
}
// http://www.daniweb.com/techtalkforums/thread1769.html
```

- Quanto mais realista a simulação, melhor (peso? potência?);

Operações exigidas

- Memorizar todos os dados importantes, podendo a introdução dos dados ser efectuada de 2 formas: a partir do teclado ou de ficheiro (tudo de uma vez);
- Guardar num ficheiro único, todos os dados relevantes;
- Remover um robô ou uma corrida;
- Mostrar os dados totais de um robô e de uma corrida;
- Gerar (simular) uma corrida;
- Mostrar listagens com:
  - todos os robôs;
  - todas as corridas;
- Mostrar as seguintes estatísticas:
  - Classificação actual de robôs no campeonato (lista com os robôs e pontos, desde o 1o ao último lugar);
  - Classificação actual das marcas (lista com marcas e pontos);

Regras para os Grupos

- Os grupos deverão ser constituídos por 3[2], de **preferência** pertencendo ao mesmo turno prático. Após a formação do grupo, deverá ser entregue ao docente[3] os seguintes dados de cada elemento: Nome, Número de aluno e e-mail[4].
- Todos os elementos do grupo devem participar activamente na execução do trabalho (ver forma de entrega)!

[2] Em casos especiais (por exemplo: estudantes trabalhadores), poderão ser aceites grupos com 4 (máximo) ou 2 elementos (no mínimo), desde que seja dada autorização pelo docente.

[3] De preferência, durante as aulas práticas, senão via via e-mail.

[4] Pelo menos um por grupo, sendo da responsabilidade do aluno que este e-mail funcione e que seja lido de forma regular.

Avaliação via os seguintes parâmetros:

- Entrega do processo de desenho (em papel): **descrição e diagrama detalhado**[5] **das classes**;
- Código devidamente **comentado**[6] e **formatado**[7];
- Minimização de código via **classes e herança**[8];
- Uso de **memória dinâmica** (opcionalmente podem utilizar outras matérias leccionadas nas aulas teóricas como por exemplo: **operadores** ou **excepções**).

---

[5] Com o nome e atributos privados das classes, embora não seja necessário descrever o nome das funções membro (parte inferior da caixa de cada classe)

[6] Significa comentar somente aquilo que é necessário por forma a que o código fique legível a qualquer utilizador externo. Pelo menos no ficheiro principal (`.cc`) que inclui a função `main`, deve aparece o número dos alunos do grupo.

[7] Uso de TABS para alinhamento de funções e blocos, linhas em branco entre funções, etc...

[8] A regra de ouro é, desde que obedeça ao que é exigido, quanto menos linhas de código tiver, melhor é a Programação por Objectos.

Avaliação via os seguintes parâmetros:

- Encapsulamento e separação de classes por ficheiros `.h` e `.cc`;
- As operações que o programa executa (ver **Operações exigidas**);
- Entrega de um **ficheiro de exemplo**[9] que possa ser lido pelo programa;
- **Avaliação efectuada pelo docente nas aulas práticas**, durante a execução do trabalho;
- **Demonstração** da aplicação ao docente durante a entrega do trabalho;
- **Respostas dadas pelo grupo** durante a entrega do trabalho;

[9] Um bom ficheiro exemplo deverá ser legível e conter diversos tipos de dados, semelhantes ao que se esperariam encontrar numa aplicação real, por forma a poder avaliar todas as potencialidades da aplicação.

O que não será avaliado

1. Funcionalidades que não foram pedidas pelo enunciado;
2. Interface com o utilizador[10];
3. Demasiadas validações: o que importa é que seja simples introduzir os dados e que estes não contenham erros;
4. Não será exigido um relatório, explicações que achem necessárias devem ser incluidas via comentários no código fonte.
5. Plágio e/ou cópia entre grupos, havendo sanções, que no limite poderão incluir a proibição de efectuar a cadeira durante este ano lectivo por todos os elementos envolvidos;

[10] Interessa apenas que a inserção e manipulação da aplicação seja simples, não interessa perder demasiado tempo com interfaces muito elaborados com menús.

Entrega dos trabalhos

- A entrega do trabalho é efectuada obrigatoriamente durante as aulas práticas[11], devendo estar presentes TODOS os elementos do grupo.
- Durante a entrega, deverá ser demonstrado o software. Será escolhido aleatoriamente um elemento do grupo, que responderá a todas as questões. A nota final será dada com base na impressão obtida por este elemento!!!
- Antes da demonstração, o código deve ser todo compilado, à frente do docente;
- Todo o material (código, projecto, ficheiros exemplo) deve ser entregue via CD, memória USB ou por e-mail, devendo aguardar por um e-mail de confirmação vindo do docente neste último caso (todo o material deve estar compactado, a partir de uma única pasta num ficheiro .zip);
- É da responsabilidade dos alunos que os ficheiros enviados não contenham virus. Penalizações ocorrerão caso contrário.

[11] A excepção vai para os estudantes trabalhadores, que em casos

Entrega dos trabalhos

- Para o desenho de software deve ser utilizado o papel (não é necessário uma capa);
- Quem entregar até ao dia 9 de Dezembro tem um bónus extra de 1 valor[12];
- O último dia para entrega é no dia 16 de Dezembro, os trabalhos não entregues até esse dia levam a nota 0!!!
- Antes a entrega, cada grupo deve marcar com o docente[13] (durante as aulas ou via e-mail) uma hora para entregar o trabalho. Cada entrega deverá demorar cerca de $\pm$15 minutos.

[12] Este bónus só é atribuído para trabalhos que tenham mais de 10 valores, até um limite máximo de 19 valores

[13] Se não o fizer, poderá ter de esperar mais tempo pela sua vez.