

# Quem Quer Ser Milionário?

## Trabalho Prático 2

### Programação

Departamento de Informática

Universidade da Beira Interior

**Data limite para entrega:** Terça-feira, 5 de Janeiro de 2021, 19h00m

**Grupos:** Este trabalho é para ser realizado em grupos de dois elementos. Os grupos podem ser formados por elementos de turnos práticos distintos mas apenas entre grupos que são lecionados pelo mesmo docente, i.e. alunos dos turnos práticos PL1, PL2, PL3 e PL4 podem formar grupo entre si e os alunos do PL5 e PL6 também. Alunos dos turnos PL5 e PL6 não podem formar grupo com alunos dos turnos PL1, PL2, PL3 e PL4.

**Trabalhos que não compilem não serão considerados, sendo nesse caso atribuída a nota de zero valores. Não serão abertas excepções.**

## 1 Trabalho a Realizar

*Quem Quer Ser Milionário?*<sup>1</sup> é um concurso de questões de escolha-múltipla cujo objectivo é acertar sucessivamente em várias questões para alcançar a quantia máxima em jogo, sendo que cada questão corresponde a uma determinada quantia.

O objectivo deste trabalho é implementar em C um programa que simule uma versão simplificada deste concurso e que permita gerir as questões do concurso, bem como registar dados dos participantes e resultados obtidos. O programa deve satisfazer os requisitos especificados nesta secção.

### 1.1 Funcionalidades do Programa

O programa deve começar por perguntar ao utilizador se pretende ler a informação sobre as questões em modo binário ou texto. Deve de seguida proceder com a leitura do conteúdo do ficheiro para memória utilizando o modo indicado. As questões são lidas/escritas no ficheiro *db.txt* (para texto) ou *db.bin* (para binário). Assuma que, no máximo, podem existir

---

<sup>1</sup>[https://pt.wikipedia.org/wiki/Quem\\_Quer\\_Ser\\_Milionario](https://pt.wikipedia.org/wiki/Quem_Quer_Ser_Milionario)

50 questões de cada nível de dificuldade (ver informação sobre os níveis de dificuldade na Secção 1.1.2). A operação indicada tem de ser realizada com sucesso antes da apresentação de qualquer menu.

O passo seguinte consiste na apresentação do seguinte **menu principal**:



1. Gerir Concurso
2. Iniciar Concurso

0. Sair

Em todas as opções existentes no programa, após a execução da acção associada com a opção indicada, deve ser sempre dada a opção ao utilizador de repetir a acção ou voltar ao menu anterior. Por exemplo, se o utilizador indicar que pretende iniciar um concurso, logo que o concurso termine é perguntado ao utilizador se pretende iniciar um novo concurso ou voltar ao menu anterior. A saída do programa será sempre feita pela opção **Sair** do menu principal (i.e. a opção **Sair** de todos os outros menus volta sempre ao menu anterior).

### 1.1.1 Opção *Gerir Concurso*

Se o utilizador seleccionar a opção 1, então é apresentado um menu com as seguintes opções:

1. Listar questões: permite mostrar no ecrã toda a informação registada para cada uma das questões existentes e qualquer outra informação que ache relevante; deve mostrar no ecrã, no máximo, 5 de cada vez, mostrando as 5 seguintes sempre que o utilizador carregar na tecla . Se o utilizador pretender interromper a visualização das questões deve escrever **quit** ou apenas **q**.
2. Procurar questões: dada uma string (palavra ou frase), mostra no ecrã toda a informação registada, e qualquer outra informação que ache relevante, para todas as questões nas quais ocorre essa string (na questão e/ou nas opções de resposta); a procura tem de ser *case insensitive*, i.e. tem de ignorar se as letras são maiúsculas ou minúsculas. Por exemplo, uma procura pela string “Picasso” deve mostrar resultados que contêm “picasso”, “PICASSO”, “piCasso”, etc. Se existirem vários resultados estes devem ser mostrados no ecrã, no máximo, 5 de cada vez, mostrando os 5 seguintes sempre que o utilizador carregar na tecla . Se o utilizador pretender interromper a visualização das questões deve escrever **quit** ou apenas **q**.
3. Adicionar questão: permite adicionar uma nova questão ao conjunto de questões que foram lidas de ficheiro; esta alteração tem de se reflectir nos ficheiros *db.txt* e *db.bin* logo que o utilizador indique que não pretende adicionar mais questões.
4. Remover questão: dado um identificador de uma questão, remove essa questão do conjunto de questões que foram lidas de ficheiro; esta alteração tem de se reflectir nos ficheiros *db.txt* e *db.bin* logo que o utilizador indique que não pretende remover mais questões.

5. Sair: volta ao menu anterior.

Note que qualquer novo concurso que venha a ser realizado após a adição ou remoção de questões tem de utilizar o conjunto de questões actualizado.

### 1.1.2 Opção *Iniciar Concurso*

Se o utilizador seleccionar a opção 2 então é iniciado um novo concurso. Deverá implementar o concurso tendo em conta o seguinte:

1. Um concurso começa com uma mensagem que pede ao concorrente para indicar o seu nome (primeiro e último nome), o qual deve ser armazenado para utilização sempre que relevante.
2. Um concurso consiste em 9 questões distintas. Para cada questão são apresentadas 4 opções de resposta sendo apenas uma a resposta correcta. Assuma que cada questão e cada opção de resposta tem no máximo 256 caracteres.
3. As questões estão divididas em três níveis de dificuldade: *fácil*, *médio* e *difícil*. A dificuldade das questões vai aumentando ao longo do concurso, sendo as 3 primeiras de nível *fácil*, as 3 seguintes de nível *médio*, e as últimas 3 de nível *difícil*.
4. Se o participante acertar todas as questões ganha o valor total (i.e. €20 000). Se errar uma questão perde toda a quantia acumulada até aquele ponto, excepto nos casos em que ultrapassou um patamar de segurança. Os patamares de segurança garantem uma determinada quantia mesmo que o concorrente erre uma questão após ter ultrapassado esse patamar.

Existem dois patamares de segurança. O primeiro corresponde a responder correctamente a todas as questões de nível fácil (patamar dos €500). O segundo corresponde a responder correctamente a todas as questões de nível fácil e médio (patamar dos €5000). Na Tabela 1 os patamares de segurança encontram-se identificados a negrito.

	Nível	Quantia
Questão 1	Fácil	€100
Questão 2	Fácil	€300
<b>Questão 3</b>	<b>Fácil</b>	<b>€500</b>
Questão 4	Médio	€2000
Questão 5	Médio	€3500
<b>Questão 6</b>	<b>Médio</b>	<b>€5 000</b>
Questão 7	Difícil	€10 000
Questão 8	Difícil	€15 000
Questão 9	Difícil	€20 000

Tabela 1: Quantia e nível de dificuldade associado com cada questão.

5. O concorrente pode decidir parar a qualquer momento, caso em que ganha o valor acumulado até aquele ponto. Esta decisão é indicada respondendo à questão com o carácter 'X'.
6. As quantias que o concorrente acumula por responder correctamente a cada questão encontram-se detalhadas na Tabela 1. Note que o valor mostrado é o valor **total** acumulado após responder correctamente a cada uma das questões. Por exemplo, se responder correctamente à **Questão 2** o concorrente ganha €200 extra ficando com um total acumulado de €300.
7. Num determinado concurso todas as questões apresentadas têm de ser distintas (i.e. não podem ser repetidas questões durante um concurso). As questões a responder e a ordem em que são apresentadas têm de ser aleatórias. Ou seja, o programa tem de escolher as questões aleatoriamente dentro de cada nível de dificuldade.

**Dica:** Para gerar números pseudo-aleatórios em C pode utilizar as funções:

```
int rand(void);
```

```
void srand(unsigned int seed);
```

ambas da `stdlib.h`. Investigue e utilize estas funções.

Por exemplo, o programa seguinte gera 5 números pseudo-aleatórios entre 0 e 9:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    srand(time(0));

    for(int i = 0; i < 5; i++)
        printf("%d\n", rand() % 10);

    return 0;
}
```

8. A ordem das opções de resposta apresentadas também tem de ser aleatória. Cada resposta está associada com uma letra ('A', 'B', 'C' ou 'D'). O concorrente indica através do teclado a letra associada com a sua resposta seguida de . Exemplo:

Qual era o primeiro nome de Picasso?

- A. Pablo
- B. Diego
- C. Jorge
- D. Nuno

Resposta: A

9. Durante todo o concurso é apresentado um sumário do estado actual. Este deve incluir o valor total acumulado, o número da questão (de 1 a 9), o nível de dificuldade da questão, e valor que acumula se responder correctamente.
10. Quando o concurso termina, ou por resposta errada ou decisão do concorrente, uma mensagem apropriada deve ser mostrada indicando qual o resultado obtido e qual a quantia que o concorrente leva para casa.
11. Sempre que um concurso termina, o nome do jogador e a quantia ganha são registados num ficheiro com o nome *resultados.txt* que mantém a informação de todos os concursos que decorreram e os resultados obtidos. Esta informação mantém-se entre execuções distintas do programa sendo incrementada a cada execução. O ficheiro deverá conter a informação em formato de texto como se segue:

Nome: Joana Ferreira  
Quantia: 20000

12. **EXTRA OPCIONAL:** Existem dois tipos de ajudas disponíveis: *50:50* e *troca*. Na ajuda *50:50* o computador elimina aleatoriamente duas respostas erradas. Note que as respostas eliminadas têm de ser aleatoriamente escolhidas. Na ajuda *troca* a questão actual é trocada por outra do mesmo nível de dificuldade. O concorrente pode utilizar cada ajuda apenas uma vez. A informação sobre as ajudas que o concorrente ainda tem disponíveis tem de ser apresentada no sumário do estado actual (referido no Ponto 9).

## 2 Algumas considerações gerais

Os requisitos indicados nas secções anteriores referem-se às funcionalidades gerais esperadas. No entanto, existem vários detalhes que são esperados em qualquer programa que não serão detalhados neste enunciado mas são esperados de qualquer programador. Exemplos incluem, mas não estão limitados a:

- Todos os inputs em todo o programa deverão ser validados
- Deverá imprimir mensagens que permitem uma interação com o programa que seja simples e intuitiva

Note que é esperado que use as abordagens e conceitos mais apropriados para a implementação deste programa, os quais foram já abordados no contexto desta Unidade Curricular. Ter um trabalho que funciona não é indicativo de que vai obter toda a pontuação: esta será obtida por trabalhos que utilizem estruturas de dados adequadas, algoritmos simples, correctos e eficientes, que separam as várias partes do programa em funções apropriadas, entre outros. Note que deve desenhar os seus algoritmos com cuidado e que deve

testar bem o programa (utilizando também casos limite) de forma a detectar erros antes da submissão. Só porque um programa funciona para alguns exemplos não é prova irrefutável de que funciona para todos. Lembre-se:

**Ao testar um programa conseguimos mostrar a presença de erros mas não a sua ausência.<sup>2</sup>**

Note que todas as funcionalidades têm de ser implementadas utilizando a linguagem C. A utilização de chamadas ao sistema para que este execute tarefas do programa não é aceite. Note também que apenas pode utilizar código standard do C, tal como temos feito nas aulas.

### 3 Entrega do Trabalho

Para que a sua submissão seja considerada **terá que usar como nome do ficheiro submetido os números de aluno dos elementos do grupo, e respectivos turnos práticos, separados pelo caracter ‘\_’ (underscore)**. Por exemplo, se os alunos com números a12345 do PL1 e Z123 do PL2 formam grupo, o ficheiro deverá chamar-se `a12345PL1_Z123PL2.c`

A entrega do trabalho deve ser feita via Moodle. No início do ficheiro de código tem de identificar claramente o nome, número de aluno, e turno prático dos elementos do grupo (num comentário). Todas as funções devem estar devidamente documentadas com comentários no código.

Deverá submeter no Moodle um ficheiro `.zip` (não são aceites outros formatos) contendo os seguintes ficheiros:

1. O ficheiro `.c` onde se encontra a implementação
2. Um ficheiro *README* onde indica quais as funcionalidades implementadas, quais as funcionalidades que ficaram por implementar, bem como qualquer informação que ache relevante para a compreensão de algum detalhe em particular; note que o texto do *README* deve ser breve e directo; pode, e deve, apresentar a informação por pontos (*bullet points*) para que seja mais conciso e simples de ler
3. Os ficheiros que contêm as questões utilizadas no seu trabalho (*db.txt* e *db.bin*)

A submissão deve ser feita através da conta de apenas um dos elementos do grupo. Garanta que fez todos os passos necessários para a submissão: terá de aceitar a declaração de submissão e finalizar a submissão carregando no botão “Submeter o trabalho”. O trabalho só estará submetido quando receber um email do Moodle a confirmar a submissão.

---

<sup>2</sup>*Program testing can be used to show the presence of bugs but never to show their absence!*, Edsger Dijkstra em <https://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD303.html>

## 4 Classificação e Critérios de Correção

A nota do trabalho será baseada nos seguintes aspectos:

1. Implementação do concurso completo e qualidade da interacção com o utilizador (até 9 valores).
2. Implementação de todos os menus, todas as funcionalidades de gestão do concurso e qualidade da interacção com o utilizador (até 6 valores).
3. Estilo de programação, uso de algoritmos e estruturas de dados mais apropriados, facilidade de leitura do código, nomes de variáveis e funções bem escolhidos, qualidade (e não quantidade) dos comentários, tamanho das funções, entrega de todos os ficheiros requeridos, e utilização dos elementos discutidos nas aulas (até 5 valores).

**Conduta:** Chama-se a atenção para o Código de Integridade e para o Regulamento Disciplinar dos Estudantes da Universidade da Beira Interior. **Submissões com código semelhante serão investigadas.**

Casos de fraude académica serão levados muito a sério. **Plágio resultará em reprovação na disciplina de todos os envolvidos e será tratado de acordo com o regulamento disciplinar em vigor, o que poderá ter consequências graves.** O corpo docente da cadeira será o único juiz do que se considera ou não plágio. Não vale a pena copiar código e apenas alterar nomes de variáveis, funções, adicionar ou remover chavetas, etc. O corpo docente utilizará software especializado na detecção de plágio em trabalhos de programação. Relembremos que quem faz o trabalho tem o direito e o dever de o manter para si e de não partilhar nem deixar consultar a sua solução. Relembremos também que quem tenta passar o trabalho dos outros como seu está a cometer fraude e, em caso de utilização de trabalho de outros colegas, está, directamente e conscientemente, a prejudicar esses colegas. Plágio contempla não apenas a submissão, integral ou parcial, de trabalho de colegas mas também submissão de trabalho de outras fontes, tais como websites e livros. Discussões sobre o trabalho prático entre colegas de grupos diferentes são permitidas (e encorajadas). No entanto, apenas devem discutir ideias e nunca partilhar ou discutir os detalhes da vossa implementação.

## 5 Recomendações

Esta secção lista recomendações importantes que deverá seguir e ter em consideração:

- Leia todo o enunciado com atenção, procurando perceber o objectivo das várias funcionalidades pedidas. Em caso de dúvida de interpretação, utilize as aulas, o horário de dúvidas ou o canal **#avaliacoes** no Slack da UC para esclarecer as suas questões. O trabalho será também discutido nas aulas teóricas.

- Um ficheiro com o nome `a12345PL1_Z123PL2.c` deverá ser compilado utilizando o comando:

```
gcc -o qqsm a12345PL1_Z123PL2.c -Wall -std=c99
```

e só deverá ser submetido para avaliação após compilação com sucesso e sem avisos.

**Trabalhos que não compilem não serão considerados, sendo nesse caso atribuída a nota de zero valores. Não serão abertas excepções.**

Se no momento da entrega o trabalho não estiver a compilar e pretende que este seja considerado para avaliação, deverá então remover todo o código que provoca erros e submeter uma versão que compile. Para evitar esta situação, termine a implementação do seu programa dias antes da data limite para entrega e não deixe a submissão para o último dia.

- Inicie a resolução deste trabalho assim que possível. Deixar a realização do projecto para os últimos dias é desaconselhado, pois poderá ficar sem tempo para resolver problemas que surjam.