# Programação 2015/2016

Mestrado em Engenharia Electrotécnica e de Computadores (MEEC)

# Blackjack II


# Projeto de Programação – Entrega Final

## 1 Introdução

O objetivo deste projeto é continuar o desenvolvimento do jogo de cartas *blackjack* mas desta vez utilizando também jogadores que não são humanos, isto é entidades artificiais para as quais é necessário programar a sua estratégia de jogo. Quanto mais eficiente for a estratégia de jogo da entidade artificial (EA) que vai jogar contra a casa maior é a probabilidade do jogador EA vencer o jogo do *blackjack* e ter mais lucro.

Neste documento não vão ser detalhadas nenhumas regras do jogo *blackjack*, nem da forma como se calcula a pontuação ou o valor do dinheiro. Estas regras estão no enunciado do projeto – entrega intermédia.

## 2 Funcionamento

O funcionamento do programa *blackjack* pode ser sumarizado da seguinte forma:

- Leitura dos ficheiros de configuração, ver Secção 5.

- Tal como na entrega intermédia, o jogador pode realizar 4 opções:
    - **Hit** – Tecla H: Recebe uma carta.
    - **Stand** – Tecla S: Passa o jogo para o próximo jogador.
    - **New** – Tecla N: Nova ronda do jogo.
    - **Quit** – Tecla Q: Sai do jogo e escreve o ficheiro de estatísticas.
- Além das quatro opções acima definidas, devem acrescentar as seguintes opções:
    - **Double** – Tecla D: Duplica o valor da aposta, recebe exatamente mais uma carta e compromete-se a fazer **stand**, isto é, passa o controlo para o próximo jogador.
    - **Surrender** – Tecla R: Desiste do jogo e a casa fica apenas com metade do valor da aposta enquanto o jogador fica com o resto.
    - **Bet** – Tecla B: Esta opção pode ser usada apenas quando um jogo terminar (antes de ser selecionada a opção N) sendo necessário ler do *stdin* o nome do jogador para o qual se aplica o aumento/diminuição da aposta e o novo valor da aposta.
- Quando ocorrer **blackjack,** o controlo deve passar automaticamente para o próximo jogador.
- As opções **double** e **surrender** apenas podem ser seleccionadas logo após terem sido distribuídas as cartas à casa e ao jogador. Note-se, assim, que não é possível escolher essas opções após um jogador fazer **hit**.
- Caso não tenha sido escolhido **double** ou **surrender**, um jogador humano pode realizar vários pedidos de **hit** até que realize um pedido de **stand** usando sempre o teclado. Naturalmente, se exceder 21 pontos deve ocorrer **bust** e o controlo passa para o próximo jogador.
- Um jogador EA, tal como um jogador humano, pode fazer **hit**, **stand**, **double** ou **surrender** a partir de um conjunto de regras lidas de um ficheiro de configuração. Sempre que fizer uma escolha deve ser adicionado um atraso de 1s. Este valor de atraso pode ser modificado através das teclas das setas, podendo ser aumentado (tecla da seta ②) ou diminuído (tecla da seta ④).
- No final de cada jogo é necessário atualizar o valor monetário com que cada jogador fica. Podem ser usados valores com vírgula flutuante com apenas uma casa decimal.
- A tecla novo jogo apenas tem efeito quando todas as cartas estiverem distribuídas.
- No final de cada ronda pode ser inserido um novo jogador da mesa numa posição livre, escolhendo a tecla A e selecionando o lugar onde se pretende adicionar o jogador através do rato. Os dados do novo jogador devem ser lidos através do *stdin*.
- Quando um jogador (seja EA ou humano) não tiver mais dinheiro para apostar deve ser removido da mesa.
- O programa só pode ser terminado no final de cada ronda do jogo pressionando a tecla q. Nesta situação o ficheiro de estatísticas deve ser escrito. Este ficheiro deve incluir estatísticas de jogadores que tenham sido removidos da mesa por qualquer motivo.

O comportamento da casa é igual ao que estava definido na entrega intermédia. No entanto, a casa deve efectuar **stand** sempre que tenha mais do que 16 pontos, mesmo que tenha um Às na sua mão. Quando não houver mais cartas do baralho para distribuir devem ser carregados o mesmo número de baralhos inicialmente lidos do stdin. Em nenhuma situação podem carregar novos baralhos sem que as cartas a distribuir tenham sido esgotadas.

O jogo deve funcionar em qualquer uma das seguintes situações: 1) só com jogadores EA; 2) só com jogadores humanos; e 3) com jogadores EA e humanos.

# 3 Entidades artificiais

As entidades artificiais pretendem simular um jogador perito em *blackjack*. Uma estratégia de jogo eficiente aumenta a probabilidade do jogador EA ganhar o jogo do *blackjack*, isto é, de ganhar mais dinheiro à casa. Note-se que o jogador EA não pode aceder ao conjunto de cartas por distribuir nem saber a carta da casa que está com a face voltada para baixo. Resumindo, o jogador EA deve implementar a estratégia delineada na Figura 1.

**4-8 Decks, Dealer Stands on Soft 17**

| Player | Dealer's card | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| hard | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
| 4-8 | H | H | H | H | H | H | H | H | H | H |
| 9 | H | Dh | Dh | Dh | Dh | H | H | H | H | H |
| 10 | Dh | Dh | Dh | Dh | Dh | Dh | Dh | Dh | H | H |
| 11 | Dh | Dh | Dh | Dh | Dh | Dh | Dh | Dh | Dh | H |
| 12 | H | H | S | S | S | H | H | H | H | H |
| 13 | S | S | S | S | S | H | H | H | H | H |
| 14 | S | S | S | S | S | H | H | H | H | H |
| 15 | S | S | S | S | S | H | H | H | Rh | H |
| 16 | S | S | S | S | S | H | H | Rh | Rh | Rh |
| 17+ | S | S | S | S | S | S | S | S | S | S |
| soft | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
| 13 | H | H | H | Dh | Dh | H | H | H | H | H |
| 14 | H | H | H | Dh | Dh | H | H | H | H | H |
| 15 | H | H | Dh | Dh | Dh | H | H | H | H | H |
| 16 | H | H | Dh | Dh | Dh | H | H | H | H | H |
| 17 | H | Dh | Dh | Dh | Dh | H | H | H | H | H |
| 18 | S | Ds | Ds | Ds | Ds | S | S | H | H | H |
| 19+ | S | S | S | S | S | S | S | S | S | S |

**Figura 1: Regras *blackjack* para a implementação de uma estratégia ganhadora.**

Na Figura 1, H corresponde a **hit**, S corresponde a **stand**, Dh/Ds corresponde a **double** e Rh corresponde a **surrender**. As colunas correspondem ao valor da carta a descoberto da casa (*dealer*) e as linhas correspondem ao valor acumulado das duas cartas que o jogador EA possui. Como pode observar na Figura 1, existem duas tabelas, uma para o caso de nenhuma carta do jogador EA ser um Ás (*hard*) e outra para o caso de uma ou mais cartas serem um Ás (*soft*).

**Implementação opcional**

Um jogador EA também pode usar uma estratégia de contar cartas e assim ganhar vantagem em relação a outros jogadores. A estratégia de contagem mais popular é a estratégia Hi-Lo que consiste em contar cada carta que seja distribuída atribuindo um valor para cada tipo de carta distribuído:

- Ás, Rei, Dama, Valete e 10 valem -1
- 7, 8 e 9 valem 0
- 2, 3, 4, 5 e 6 valem +1

Naturalmente, cartas com baixo valor devem valer mais, uma vez que é provável que a seguir venham cartas com elevado valor (por exemplo Ás) e cartas com alto valor devem valor menos, uma vez que é mais provável que a seguir venham cartas de menor valor.

À medida que as cartas são distribuídas é necessário actualizar a contagem das cartas (valor cumulativo), somando ou subtraindo de acordo com o valor atribuído à carta distribuída. Sempre que for necessário efetuar uma decisão (isto é, decidir o valor da aposta) é necessário dividir esse valor acumulado pelo número de baralhos ainda não distribuídos. Quanto maior for esse valor, mais elevada deve ser a aposta e quanto menor for esse valor, mais baixa deve ser a aposta. Tipicamente, por cada valor de +1 o jogador EA deve duplicar o valor da aposta considerada mínima (lida do ficheiro de configuração), isto é, se a aposta mínima for N, para valores de +1, +2, +3, +4, +5, a aposta deve ser 2N, 4N, 8N, 10N e 12N. O uso desta estratégia vai ser avaliado com o programa a funcionar apenas com jogadores EA.

# 4 Estruturas de Dados

No desenvolvimento deste jogo, é obrigatório obedecerem às seguintes regras:

- Uma carta deve ser identificada através de uma estrutura de dados que tenha pelo menos o código da carta (de 1 a 13) e o naipe.
- Não podem ser usados vectores nem matrizes de tipos de dados elementares (int, char, float, etc.) para representarem um qualquer conjunto de cartas ou conjunto de jogadores.
- Devem usar uma lista (escolham o tipo de lista mais adequado) para representarem os jogadores que num dado momento estão a participar no jogo.
- Também devem usar uma lista para representar o conjunto de cartas a distribuir pela casa.
- Apenas podem usar um vector de estruturas para representarem as cartas que os jogadores e a casa possuem. Note-se que vai ser necessário um vector para cada jogador.
- Opcionalmente, podem usar uma pilha para representarem as cartas que o jogador e a casa possuem. Neste caso, a lista de jogadores deve estar ligada a uma pilha de cartas através de ponteiros.
- Devem representar por uma matriz 2D as ações que os jogadores EA devem fazer (ver Figura 1).

Também é importante salientar que o programa desenvolvido deverá ser estruturado em múltiplos ficheiros que permitam uma organização adequada do código.

# 5 Leitura de Ficheiros

O funcionamento do jogo deve ser ditado por um conjunto de parâmetros lidos através de um ficheiro de configuração usando funções apropriadas (ver Figura 2):

- Número de baralhos de 52 cartas que vai ser utilizado pela casa. Podem ser usados entre 4 e 8 baralhos.
- Número de jogadores que vão participar no jogo. Este valor deve estar compreendido entre 1 e 4.
- Para cada jogador devem indicar:

- Se o jogador é humano (HU) ou é uma entidade artificial (EA)
- Nome do jogador. O nome é representado por uma string com tamanho máximo de 8 caracteres.
- Valor inicial de dinheiro com que cada jogador começa. Não devem ser aceites valores inferiores a 10 e superiores a 500.
- Valor da aposta inicial de cada jogador. Nunca poderá ser superior a 25% do valor inicial de dinheiro que o jogador possui e inferior a 2 euros.

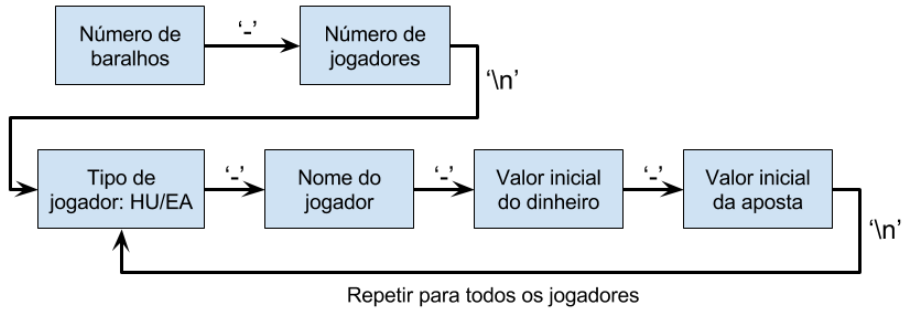


**Figura 2: Sintaxe do ficheiro de configuração do jogo.**

O funcionamento das entidades artificiais deve ser ditado por um conjunto de regras que estão representadas por um formato matricial e que indicam como é que a entidade artificial vai jogar. Estas regras devem também estar num ficheiro que deve ser lido no início do programa.

Os nomes dos dois ficheiros devem ser lidos através do argumento da função main, o que significa que quando o programa é executado é necessário indicar dois ficheiros: o ficheiro de configuração e o ficheiro de estratégia dos jogadores EA.

# 6 Escrita de Ficheiros

Tal como na entrega intermédia, é necessário escrever um ficheiro que indique no final do jogo a seguinte informação para cada jogador do *blackjack*:

- Nome e tipo do jogador
- Número de jogos que ganhou, empatou ou perdeu.
- Valor monetário com que o jogador ficou no final
- Valor monetário que a casa ganhou ou perdeu

# 7 Aspeto Gráfico

Devem usar o código da entrega intermédia para mostrar as cartas da casa e do jogador. Este código necessita de ser adaptado para se poderem usar listas em vez de vectores e matrizes bidimensionais. Por cima da área reservada a cada jogador (onde são mostradas as cartas distribuídas) é necessário mostrar:

- Nome do jogador

- Valor da aposta
- Número de pontos correspondentes às cartas que cada jogador possui ou BJ, BU para indicar que ocorreu **blackjack** ou **bust, respetivamente**.

Do lado direito da aplicação (espaço em branco) é necessário mostrar:

- Nome (primeiros 8 caracteres) e tipo de cada jogador.
- Valor actual do dinheiro de cada jogador. O valor da aposta deve ser descontado sempre que uma nova ronda seja iniciada.

Tal como na entrega intermédia, devem mostrar qual é o jogador que se encontra a jogar, por exemplo desenhando a área de jogo desse jogador com outra cor.

# 8 Desenvolvimento do Jogo

Também é fundamental que os alunos cumpram as regras que se seguem no desenvolvimento do jogo.

## 8.1 Desenvolvimento Faseado

O desenvolvimento deste projeto deverá ser feito de uma forma faseada, devendo os alunos garantir que todas as funcionalidades codificadas até ao momento estão a funcionar corretamente.

**É preferível um programa que implementa poucas funcionalidades, mas que funcionam corretamente, do que um programa totalmente desenvolvido mas que faz muito pouco.**

Assim sugerem-se os seguintes passos, pela ordem apresentada, para realização do projeto, organizados em duas fases.

**Fase 1:**

No final desta fase, as funcionalidades da entrega intermédia devem estar correctamente implementadas recorrendo a estruturas de dados, listas ligadas e usando um ficheiro de configuração. Desta forma, é necessário realizar as seguintes tarefas:

- Inicialização da biblioteca SDL e criação da interface gráfica.
- Criação das estruturas de dados necessárias para representar as cartas do jogador e da casa.
- Leitura do ficheiro de configuração necessário ao funcionamento do programa.
- Implementação da função baralhar cartas usando uma lista ligada de cartas.
- Recorrendo ao código realizado do projeto intermédio:
    - Adaptar as opções **hit**, **stand**, **new** and **quit**.
    - Verificar se ocorreu **bust** ou **blackjack.**
    - Calcular os pontos de um conjunto de cartas.
    - Calcular o dinheiro com que cada jogador fica no final de cada ronda.
    - Verificar todo o controlo do jogo, por exemplo, a distribuição de cartas para os jogadores e casa.

- Atualização da interface gráfica, modificando as funções fornecidas para que recebam como entrada as listas de cartas.
- Implementação das funcionalidades gráficas definidas na Secção 7.
- Escrita do ficheiro com os resultados dos jogos.

**Fase 2**

No final desta fase devem estar implementadas as novas funcionalidades e o funcionamento do jogo com entidades artificiais (EA), que segue um conjunto de regras definidas no ficheiro de estratégia.

- Remoção de um jogador da mesa sempre que este fique sem dinheiro ou sempre que o utilizador o sinalizar (opção A).
- Implementação das opções **double**, **surrender** e **bet**.
- Leitura do ficheiro de configuração com a estratégia de jogo para os jogadores EA.
- Implementação do funcionamento do jogo com jogadores EA de acordo com a estratégia definida.
- Implementação do atraso nos jogadores EA bem como o reconhecimento das teclas das setas.
- Implementação da estratégia hi-low para receber a bonificação.

Os alunos deverão garantir a robustez da aplicação verificando todos os casos de erro (por exemplo quando um parâmetro de entrada não seja válido).

## 8.2 Documentação

O código produzido pelos alunos deverá ser comentado. Os comentários presentes no código deverão explicitar e explicar o funcionamento da aplicação assim como as decisões tomadas. As seguintes regras devem ser cumpridas:

- O código deve ser comentado sempre que realize alguma operação não óbvia.
- Os comentários devem ser claros, gramaticalmente corretos e usando frases simples.
- A declaração de todas as variáveis e constantes deve ser acompanhada de um comentário com uma breve descrição de para que servem.
- Cada bloco de código (seleção ou repetição) deve ser precedido de um breve comentário explicativo.
- Todos os programas devem ter um comentário inicial que identifique, no mínimo, o trabalho, o seu autor, o fim a que se destina e a data de realização.

## 8.3 Indentação

Um ponto fundamental na organização de escrita de código é a indentação, isto é, organização hierárquica das linhas de código, de acordo com âmbito onde elas se encontram. A indentação deixa o código fonte do programa mais organizado, mais legível, fácil de entender e de modificar, sendo uma parte essencial do trabalho.

## 8.4 Estrutura do Código

Todos os programas em C devem possuir a mesma estrutura genérica, composta pelas seguintes secções:

- Bloco de comentários.
- Diretivas #include.

- Constantes globais e variáveis globais (caso sejam necessárias).
- Declaração de funções.
- Função main().
- Definição de funções.

Como regra geral deve considerar que as funções devem caber num único ecrã, isto é, devem ter no máximo cerca de 30 linhas. Também deve cumprir as seguintes regras:

- Inicialize sempre as variáveis na sua declaração.
- Teste a validade dos parâmetros recebidos por uma função.
- Declare constantes e evite usar números no corpo das funções.
- Evite repetições de código, use funções, ciclos, etc.
- Evite o uso de variáveis globais.
- Não use **goto.**
- Escreva código simples e claro que um colega seu possa perceber !

## 8.5 Compilação

O compilador a usar na execução do projeto é o gcc em ambiente Linux. **Os projetos que não compilem, i.e. que tenham erros de sintaxe, não serão avaliados.** A existência de avisos durante a fase de compilação poderá ser indício da existência de problemas no código. Estes deverão ser eliminados corretamente ou ignorados com cuidado extremo.

## 8.6 Decisões do Projeto

Como em qualquer projeto de informática, o funcionamento do programa não está totalmente definido no enunciado, existindo algumas ambiguidades e omissões. Para resolver essas omissões os alunos deverão tomar algumas decisões aquando do desenvolvimento do projeto. Estas decisões devem ser fundamentadas, sem nunca ir contra o definido no enunciado.

## 8.7 Biblioteca SDL

Durante o desenvolvimento deste projeto devera ser usada a biblioteca SDL2. A aplicação deverá ser compilada usando as bibliotecas SDL2, SDL2_image, SDL2_ttf. Mais informação disponível aqui:

- http://wiki.libsdl.org/APIByCategory
- https://wiki.libsdl.org/FrontPage

# 9 Submissão

Os alunos deverão submeter o código desenvolvido através do sistema FENIX até 20 de Maio. Devem entregar o código correspondente ao programa desenvolvido, nomeadamente os ficheiros de texto com extensão **.c** e **.h** e os ficheiros de configuração usados. Não utilize um processador de texto (e.g. Word) para formatar o seu programa.

## 10 Plágio

Os trabalhos serão objeto de um sistema de deteção de plágio. Os alunos podem conversar entre si para discutir possíveis soluções para algum problema que tenham mas não podem partilhar código fonte. Nesta entrega final, o projecto deve ser realizado em grupos de dois alunos! Se uma cópia for detetada todos os alunos envolvidos na cópia serão penalizados.

## 11 Avaliação

A avaliação do projeto terá em conta diversos parâmetros:

- Funcionalidades implementadas
- Uso correcto das estruturas de dados
- Criação de listas, filas ou pilhas de forma adequada aos objetivos do projeto
- Eficiência e qualidade da implementação dos jogadores EA
- Qualidade do código produzido
- Estruturação da aplicação
- Comentários e legibilidade do código
- Tratamento de erros