# Projecto
# Big Two

Laboratórios de Algoritmia I
Laboratórios de Informática II

2015/2016

Pretende-se criar uma CGI na linguagem de programação **C** que corra no sistema operativo Linux (disponibilizado numa máquina virtual para quem precisar) que permita jogar ao **Big Two**.

## Definição do problema

O **Big Two** é um jogo muito popular no Extremo Oriente incluindo China, Hong Kong, Indonésia, Macau, Malásia, Singapura e Taiwan. Tal como a maioria destes jogos orientais, cada jogador pode jogar uma combinação de cartas (que pode conter uma, duas, três ou cinco cartas numa combinação de **Poker**) e pretende livrar-se de todas as suas cartas antes dos outros jogadores.

Para se familiarizar com o jogo, leia as regras no melhor sítio sobre jogos de cartas [1] até à secção de *Scoring* (incluindo esta secção). O seu programa deverá permitir a um utilizador jogar num *browser*, ordenar as cartas de várias maneiras, pedir ajuda para escolher que combinação de cartas deve jogar, ver as cartas jogadas na última ronda, escolher o nível de dificuldade dos adversários, configurar o aspeto do jogo (e.g., das cartas a jogar entre vários desenhos possíveis), etc.

A título de bonificação, deverá também implementar um jogador de **Big Two** inteligente que participará num torneio no fim do semestre.

## Calendarização e Entrega

| Etapa | Data de Entrega | Nota |
|---|---|---|
| 1ª etapa | 27 Mar | 6 valores |
| Defesa da 1ª etapa | 29 Mar a 04 Abr | |
| 2ª etapa | 3 Mai | 7 valores |
| Defesa da 2ª etapa | 16 Mai a 20 Mai | |
| 3ª etapa | 29 Mai | 7 valores |
| Defesa da 3ª etapa | 30 Mai a 3 Jun | |

A defesa de cada etapa é **presencial** e deverá ser feita por **todos** os elementos na semana correspondente à defesa dessa etapa e no turno prático correspondente. Se algum elemento ou grupo não defender, terá **zero** nessa etapa. A

entrega de uma etapa poderá ser feita na etapa seguinte (isto só é válido para a 1ª e 2ª etapas) mas a avaliação levará uma penalização de 25%.

Eis o que deverá ser entregue em cada etapa:

1. Um programa que:
   - Baralhe as cartas;
   - Permita jogar combinações válidas com uma, duas ou três cartas[1] através de um *browser* (e.g., selecionando as cartas que pretender jogar e carregando em seguida num botão para jogar);
   - Calcular o score no fim do jogo.
2. Um programa que:
   - Deverá também permitir jogar as combinações de **Poker** (i.e., de cinco cartas);
   - Deverá ordenar as cartas de várias formas possíveis (e.g., por valores, por naipes, por combinações);
   - Deverá sugerir ao jogador que combinação jogar a seguir.
3. Jogador inteligente de **Big Two**, documentação do código e análise do código gerado.

Para além disso também se avaliam os seguintes pontos na terceira etapa:

- Não ter avisos ou erros quando o código é compilado com as seguintes opções: `gcc -ansi -Wall -Wextra -pedantic -O2` do gcc;
- Legibilidade do código;
- Documentação do código;
- Relatório do projeto que explique as opções tomadas;

## Grupos

Os grupos de trabalho são compostos por 3 elementos e terão necessariamente de ser compostos por pessoas do mesmo turno prático. Nos casos em que o número de elementos no turno não seja divisível por 3 aceitam-se 2 grupos de 2 elementos se o resto da divisão do número por 3 for 1 e 1 grupo de 2 elementos se o resto der 2.

## Material a entregar em cada etapa

- Código fonte.
- Documentação gerada automaticamente pelo `Doxygen`;

[1] i.e., o programa deverá funcionar assumindo que não existem mãos de **Poker**

## Critérios obrigatórios

Os seguintes critérios tem que ser cumpridos ou a entrega não é válida:

- O programa tem que compilar sem erros com as opções `-Wall -Wextra -pedantic -ansi -O2` e funcionar na máquina virtual disponibilizada;
- O programa que implementar o jogador inteligente deverá ler as instruções do `stdin` e responder segundo o protocolo esperado através da escrita no `stdout`.

## Entrega

A entrega é feita no servidor de redmine (lim.di.uminho.pt). Deverá ser colocado na opção "Files" do redmine ("Ficheiros" para os alunos que usam a versão portuguesa) um arquivo compactado com o comando `tar` do qual constem os seguintes ficheiros e pastas:

**identificacao** ficheiro com a identificação dos alunos (nome completo e número);

**analise.pdf** ficheiro com o relatório da análise do código gerado pelo compilador;

**code** pasta com o código fonte;

**code/jogador** pasta com o código fonte do jogador inteligente;

**doc** com a documentação `html`[2] gerada pelo `doxygen`.

A pasta deverá ter o nome PLg<nº do grupo>-et<nº da etapa>.tar.bz2
Nos casos em que o número do grupo seja só um algarismo este deverá ser precedido de um zero. Exemplos:

- PLg02-et1.tar.bz2 grupo 2 a entregar a etapa 1
- PLg11-et2.tar.bz2 grupo 11 a entregar a etapa 2

Para se usar o comando `tar` da forma correta:

1. Abre-se uma consola no Linux
2. Usando o comando `cd` vai-se para a diretoria que contém as pastas code e doc
3. Escreve-se o comando `tar jcf PLg11-et2.tar.bz2 identificacao code doc`

Caso algum destes requisitos não seja cumprido, o trabalho não será considerado entregue.

[2] i.e., deve conter dentro (e não em subpastas) o ficheiro index.htm ou index.html gerado pelo Doxygen assim como os restantes ficheiros necessários

## Análise do Código Gerado pelo Compilador

1. Vá buscar através do `redmine` o código que está no ficheiro `contar.c`;
2. Compile o código com o GCC (e.g. `gcc -02 contar.c`) que se encontra na máquina virtual e seguidamente use o `gdb` (e.g., `gdb a.out`);
3. Vá buscar o código gerado pelo compilador para a função escrevendo o seguinte no prompt do gdb:

   ```
   disassemble contar_valores
   ```

4. Guarde o código que obteve num ficheiro de texto;
5. Crie a tabela de alocação de registos;
6. Corra o programa e coloque um *breakpoint* na função;
7. Identifique a área de memória associada à variável `valores`, descubra quanto espaço ocupa (e explique porquê) e faça um esquema da organização dessa área de memória;
8. Indique como é feita a indexação da matriz e mostre quais são as linhas do código *assembly* que lhe correspondem;
9. Identifique que instruções em *assembly* correspondem a cada instrução em C;
10. Entregue um ficheiro com o resultado chamado `analise.pdf` contendo o resultado da sua análise (nomeadamente os pontos 4, 5, 7, 8 e 9) juntamente com o resto do trabalho colocando este ficheiro na raiz (ao mesmo nível do ficheiro `identificacao` e das pastas `code` e `doc`).

## Jogador Inteligente

Para implementar o jogador inteligente, deverá escrever um programa em **C** que leia as instruções do `stdin` e escreva o resultado no `stdout`. O jogador terá que seguir um protocolo muito rígido visto que ele irá jogar com mais três jogadores (feitos por outros grupos) e por isso o sistema terá que funcionar de forma completamente automática. Caso o seu programa não responda da forma correta, não poderá obviamente participar no torneio. Sugere-se que implemente este requisito de tal forma que ele possa ser integrado sem qualquer problema no seu jogo que funciona no *browser*.

### Protocolo

O seu programa deverá aceitar a seguinte linguagem:

**MAO** seguida com as cartas que compõe a mão do jogador;

**JOGOU** seguida da combinação de cartas que o adversário jogou;

**PASSOU** caso o adversário em causa tenha passado;

**JOGADA** que pede ao jogador para efetuar uma jogada;

**OK** que informa o jogador que o seu comando JOGADA foi executado com sucesso;

**NAO** que informa o jogador que o seu comando JOGADA não foi executado devido a alguma violação das regras de jogo;

**ACABOU** seguida do score do jogador para este jogo e do score cumulativo.

O protocolo será o seguinte: inicialmente, o seu programa recebe a instrução MAO que o informa da sua mão inicial. Seguidamente, receberá a instrução JOGADA quando for a sua vez (seguida de OK ou NAO para o informar se a sua jogada foi aceite) ou, para cada jogador, a instrução JOGOU ou PASSOU para o informar do que esse jogador fez. Quando o jogo acabar, receberá a instrução ACABOU.

O seu programa deverá responder escrevendo no *stdout* uma combinação de cartas separadas por espaços ou a palavra PASS0.

Cada carta é descrita por dois carateres, em que o primeiro indica o valor da carta (2 A K Q J T 9 8 7 6 5 4 3) e o segundo o naipe (S: ♠, H: ♥, C: ♣, D: ♦).

## Exemplo

Para ilustrar o funcionamento do sistema, imagine que o seu programa recebe a seguinte mão:
A ♠ K ♦ J ♥ J ♦ 10 ♣ 8 ♠ 8 ♥ 8 ♦ 7 ♠ 6 ♥ 5 ♥ 4 ♦ 3 ♦

Como esta mão contém o 3 ♦, você é o primeiro a jogar e por isso o sistema pede-lhe que envie uma jogada. O sistema fá-lo passando o seguinte para o *stdin* do seu programa:

```
MAO AS KD JH JD TC 8S 8H 8D 7S 6H 5H 4D 3D
JOGADA
```

O seu programa decide responder jogando a sequência 7 ♠ 6 ♥ 5 ♥ 4 ♦ 3 ♦, que inclui o 3 ♦, imprimindo o seguinte no *stdout*:

```
7S 6H 5H 4D 3D
```

A seguir é impresso o comando OK e depois os outros 3 jogadores jogam: o primeiro decide passar, o segundo decide jogar um *flush* de paus (3 ♣ 4 ♣ 5 ♣ 7 ♣ 9 ♣) e o último um *full house* (6 ♠ 6 ♦ 6 ♣ 4 ♥ 4 ♠).

```
OK
PASSOU
JOGOU 3C 4C 5C 7C 9C
JOGOU 6S 6D 6C 4H 4S
JOGADA
```

Agora você decide jogar o *full house* de oitos e valetes (J ♥ J ♦ 8 ♠ 8 ♥ 8 ♦):

```
JH JD 8S 8H 8D
```

O seu programa recebe as seguintes instruções visto que o primeiro e o segundo jogadores passaram e o terceiro jogou um *full house* (10 ♠ 10 ♥ 10 ♦ 5 ♠ 5 ♦).

```
OK
PASSOU
PASSOU
JOGOU TS TH TD 5S 5D
JOGADA
```

Você não tem nenhuma combinação possível e por isso decide passar:

```
PASSO
```

E descobre que o primeiro jogador tinha um *poker* de rainhas (Q ♠ Q ♥ Q ♣ Q ♦ 8 ♣) guardado até agora.

```
JOGOU QS QH QC QD 8C
PASSOU
PASSOU
JOGADA
```

E você passa novamente:

```
PASSO
```

O jogo continua e você não se consegue livrar de mais nenhuma carta, ficando assim com o score de 3 (lembre-se que neste jogo, quanto menor o score, melhor). Como este é o primeiro jogo, o seu score é igual ao acumulado. Caso se tivesse livrado de todas as suas cartas, o seu score seria de zero.

```
ACABOU 3 3
```

### Avaliação

A avaliação dos jogadores será feita mediante um torneio. Para se candidatarem ao torneio, o grupo deverá criar uma pasta chamada `jogador` debaixo da pasta `code` com o código necessário, que deverá compilar (com as opções descritas nos critérios obrigatórios) sem *warnings* nem erros. O programa deverá ainda seguir o protocolo dado acima, e responder em menos de **1 segundo de CPU**.

Caso o programa cumpra todos os requisitos, este entrará num torneio com todos os outros programas concorrentes. A classificação dependerá da sua prestação nesse torneio.

## Referências

[1] Big Two (PAGAT), `https://www.pagat.com/climbing/bigtwo.html`