Programação Imperativa
IPCA – EST

1º ano – LEDJD
Trabalho Prático I
Data de entrega: 3 de dezembro de 2021

As resoluções deste trabalho prático devem ser submetidas na plataforma `https://elearning1.ipca.pt/2122/` **apenas por um dos elementos do grupo**, até ao dia 3 de dezembro de 2021.

1 Objectivos

Com este trabalho prático pretende-se sedimentar os conhecimentos apresentados nas aulas, relativos a:

- Métodos rigorosos de análise de problemas (requisitos) e desenvolvimento de software;
- Métodos de programação imperativa suportados em algoritmos e estruturas de dados lineares e sua implementação em C.
- Técnicas de *debugging*, análise de programas e geração de código robusto e rápido.

2 Enunciado

À semelhança de algumas *suites* de jogos de consola que aglomeram um conjunto de diferentes jogos num só pacote, o IPCA Games Arch (IGA) oferece uma série de jogos tradicionais num só programa C. Implemente a seguinte lista de requisitos para construir o IGA.

Qualquer melhoria devidamente fundamentada aos requisitos apresentados, ou desenvolvimento de outros jogos, será considerada como melhoria ao trabalho prático.

2.1 Requisitos

1. **Menu de jogos.** Implemente o menu do IGA, apresentando um conjunto de opções para o utilizador selecionar. O jogador seleciona cada opção introduzindo o número associado a essa opção. O jogador é alertado caso introduza um carater inválido de opção. O menu do IGA deve permitir ao jogador efetuar, pelo menos, as seguintes operações:

- Selecionar um jogo.
- Visualizar os seus pontos por jogo e pontos globais. Os pontos globais do jogador são o somatório dos seus pontos em todos os jogos.
- Visualizar a tabela de pontos dos 10 melhores jogadores de cada jogo.
- Visualizar a tabela de pontos global dos 10 melhores jogadores.
- Introduzir dados de 2 jogadores. No caso de jogos de apenas 1 jogador, utilize apenas um dos jogadores introduzidos.
- Sair do IGA.

Nota: pode utilizar a função int system(const char *command) da biblioteca stdlib.h para limpar a linha de comandos, p.e. executando a instrução system("cls").

2. **Adivinha o número.** Escreva um programa em C que gere um número inteiro aleatório entre 0 e 100, e permita ao utilizador tentar adivinhar o número gerado. À medida que o utilizador for sugerindo números o programa deve informar se o número sugerido é maior ou menor que o número gerado aleatoriamente. Caso o utilizador acerte no número o programa deve terminar mostrando o número de tentativas que o utilizador necessitou para adivinhar o número aleatório. A pontuação deste jogo é data pela fórmula $P = 10 - NT$, em que NT é o número de tentativas falhadas. Caso P seja negativo, a pontuação é 0.

   Pode utilizar a seguinte função, que retorna um número aleatório entre os valores passados nos argumentos `min` e `max`.

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

int generateRandomInt(int min, int max) {
        srand((unsigned)time(NULL));
        return min + rand() % (max - min);
}
```

3. **Adivinha a carta.** Implemente um jogo em que o computador seleciona uma carta aleatoriamente de um baralho com 52 cartas de 4 naipes e o jogador tenta adivinhar a carta. Em cada tentativa o jogador introduz um naipe e um valor da carta. O número de cada carta numérica corresponde ao seu valor, o às tem o valor 1, o valete o valor 11, a dama o valor 12 e o rei o valor 13. O computador compara a tentativa do jogador com a carta selecionada aleatoriamente no início do jogo, informado se está totalmente correta, se o naipe está correto e se o valor da carta está acima ou abaixo. A pontuação deste jogo é dada pela fórmula $P = 6 - NT$, em que NT é o número de tentativas falhadas. Caso P seja negativo, a pontuação é 0.

   Nota: utilize a função int generateRandomInt(int min, int max) para retirar cartas aleatoriamente do baralho.

4. **Vinte-e-um.** Implemente o jogo de cartas vinte-e-um, utilizando apenas 1 baralho de 52 cartas de 4 naipes, em que apenas um jogador joga contra o computador. O objetivo do jogador é conseguir um valor total de cartas na sua mão superior ao da mesa (computador), mas sem ultrapassar o valor 21. Caso o jogador ou a mesa ultrapassem o valor 21, perdem imediatamente. O valor de cada carta numérica é o seu respetivo número, o às vale 1 e as figuras valem 10. O jogo começa com o computador a dar uma carta para o jogador e outra para a mesa (o computador). De seguida, o jogador pode escolher pedir mais uma carta, ou manter a(s) carta(s) que tem. Caso decida manter a(s) carta(s) que tem, o jogador não pode depois voltar a pedir mais cartas. O computador retira sempre cartas para si, até ganhar ao jogador, fazendo mais pontos que ele e não ultrapassando 21, ou ultrapassa 21, perdendo neste último caso. O jogador ganha 1 ponto por cada vitória e perde 1 ponto por cada derrota.

5. **Jogo do Galo (para dois).** Implemente o jogo do galo tradicional, que consiste num tabuleiro de 3×3, onde em cada jogada um dos jogadores pode colocar uma cruz numa casa e o outro jogador um círculo. Ganha o jogador que conseguir efetuar uma linha horizontal, vertical ou diagonal de símbolos iguais. O jogador que joga primeiro é selecionado aleatoriamente pelo computador. O jogador que começa a jogar primeiro, ganha 1 ponto por cada vitória e perde 2 pontos por cada derrota. O jogador que começa a jogar em segundo lugar, ganha 2 pontos por cada vitória e perde 1 ponto por cada derrota.

6. **Jogo do Galo vs CPU.** Melhore o jogo anterior, acrescentando uma opção para o jogador jogar contra o computador.

7. **Jogo da Forca.** Implemente o jogo da forca, em que o computador propõe o enigma (palavra ou frase) e o jogador tenta adivinhar o enigma letra-a-letra, podendo ainda em qualquer altura tentar adivinhar o termo completo de uma só vez. O jogador dispõe de 6 tentativas para introduzir letras que não existam no enigma até perder o jogo. O jogador ganha 1 ponto por cada letra que acerte, independentemente do número de ocorrências dessa letra no enigma, e perde 1 ponto por cada letra que não exista no enigma. Quando o jogador adivinha o termo completo, ganha 1 ponto por cada letra escondida.

8. **Quatro-em-Linha.** Implemente o jogo quatro-em-linha para 2 jogadores. O objetivo do jogo é ir colocando peças numa das 7 colunas de um tabuleiro com 6 linhas, até se alinhar (na horizontal, vertical ou diagonal) 4 peças do mesmo símbolo (do jogador ou do computador). Como o tabuleiro está na vertical, o efeito da gravidade faz com que cada peça colocada num coluna desça até à primeira casa livre a contar da base. O jogador ganha 4 pontos por cada vitória e perde 4 pontos por cada derrota.

9. **Quatro-em-Linha vs CPU.** Melhore o jogo anterior, acrescentando uma opção para o jogador jogar contra o computador.

2.2 Melhoria

Como melhoria ao trabalho prático, sugerem-se os seguintes desafios:

1. **Ficheiros.** Grave e leia informação para ficheiros de modo a manter o estado do IGA relativamente a jogadores e suas pontuações.

2. **Galo Cervejeiro.** Implemente o jogo galo cervejeiro, semelhante ao tradicional Jogo do Galo, em que o jogadores se defrontam sobre um tabuleiro quadrado (3×3), podendo em cada jogada: colocar numa casa vazia uma caneca de cerveja cheia, meada, vazia; ou beber um golo de cerveja de uma caneca não-vazia colocada numa casa do tabuleiro (cada golo é sempre exatamente igual a meia caneca). Ganha o primeiro que conseguir por em linha (horizontal, vertical, ou diagonal) três canecas com o mesmo nível (cheia, meada, vazia). O jogador que joga primeiro é selecionado aleatoriamente pelo computador. O jogador que começa a jogar primeiro, ganha 2 pontos por cada vitória e perde 4 pontos por cada derrota. O jogador que começa a jogar em segundo lugar, ganha 4 pontos por cada vitória e perde 2 pontos por cada derrota.

3. **Galo Cervejeiro vs CPU.** Melhore o jogo anterior, acrescentando uma opção para o jogador jogar contra o computador.

4. **Master-Mind.** Implemente o jogo master-mind, em que o jogador tenta adivinhar uma sequência de 4 cores (símbolos) selecionadas aleatoriamente pelo computador. Em cada tentativa de adivinhar o enigma o computador indica o número de cores (símbolos) corretas e o número de cores (símbolos) no sítio correto corretas. O jogador ganha 20 pontos por cada vitória e perde 5 pontos por cada derrota, não podendo ficar com pontuação negativa.

5. **Master-Mind vs CPU.** Melhore o jogo anterior, acrescentando uma opção para o jogador jogar contra o computador.

6. **Mine-Sweeper.** No jogo mine-sweeper, o jogador tenta encontrar um conjunto M de minas num campo quadrado de dimensão N, minado aleatoriamente. Nesta versão, o jogador pode pedir ajuda (até A vezes), sendo-lhe indicada uma célula sem minas pelo computador. Ao destapar uma célula sem minas, todas as células adjacentes sem minas são também destapadas. O jogador perde jogo se destapar uma célula minada. O jogador ganha 1 ponto por cada mina encontrada, 20 pontos se conseguir encontrar todas as minas e perde 1 ponto por cada ajuda. Implemente o jogo mine-sweeper em que $M = 10$, $N = 10$ e $A = 5$.

7. **Mine-Sweeper Multi-nível.** Melhore o jogo anterior, acrescentando diferentes níveis de dificuldade fazendo variar os valores de M, N e A.