TÉCNICO LISBOA

Mestrado em Engenharia Electrotécnica e de Computadores

PROGRAMAÇÃO

Projecto - Mastermind

Ano Lectivo 2018/2019

Projecto de Programação 2018/2019

O projecto de Programação 2018/2020, Mastermind, será desenvolvido ao longo do semestre dividindo-se em 2 etapas designadas por Projecto Intermédio e Projecto Final, respectivamente. O enunciado detalhado do trabalho a desenvolver em cada uma das etapas acima referidas descreve-se de seguida.

Projecto Intermédio - Mastermind

O projecto intermédio pretende aferir a capacidade dos alunos desenvolverem programas elementares em linguagem C respondendo a um desafio proposto. Neste projecto pretende-se que seja implementado um programa que permita ao utilizador jogar Mastermind adivinhando a chave secreta gerada pelo computador. O trabalho será realizado por grupos de 2 alunos e deverá ser entregue até 5 de Abril via FENIX.

1. Regras do Jogo Mastermind Tradicional

O jogo Mastermind tradicional é um jogo para 2 jogadores em que um destes jogadores gera uma chave secreta de 4 pinos de cor, considerando um universo de 6 cores, e o outro jogador tenta adivinhar a chave secreta no menor número de jogadas. A avaliação de cada tentativa do jogador que tenta adivinhar a chave consiste em indicar quantas cores existem e estão na coluna certa (pino de resposta preto) e quantas cores existem mas estão na coluna errada (pino de resposta branco). A figura 1 ilustra uma sequência de jogadas e a respectiva avaliação até o jogador descobrir a chave secreta. Se houver cores duplicadas na tentativa (exemplo da jogada 2), só lhes é atribuído um pino branco ou preto por cada pino dessa cor existente na chave.

Fig. 1: Exemplo de jogo.

2. Regras do Jogo Mastermind a Implementar

O jogo Mastermind a implementar baseia-se na versão original mas introduz algumas variantes. O programa deverá manter um registo simples dos resultados de um grupo de jogadores. O programa executa uma sequência de jogos para cada jogador e no fim apresenta uma estatística dos resultados obtidos. A configuração do Mastermind pode variar em termos da dimensão da chave e da complexidade dessa chave por variação do número de cores e por permitir ou não repetição de cores na chave.

3. Fases de Desenvolvimento

A implementação deve seguir as fases de desenvolvimento descritas nas sub-secções seguintes e a interface deverá ser única e exclusivamente no terminal de texto.

3.1. Inicializações

A inicialização do jogo consiste em introduzir os seguintes parâmetros de inicialização (os valores introduzidos devem ser todos validados):

- Número de jogadores (1 a 4)
- Nome de cada jogador (max. 20 caracteres)
- Número de jogos a realizar por cada jogador (1 a 5)
- Número de cores (6 a 12)
- Dimensão da chave secreta (4 a 8)
- Indicar se a chave pode ter repetições de cores (S/N)
- Número de máximo de tentativas (10 a 20)
- Tempo máximo por jogo para cada jogador (60 a 300s)

3.2. Jogo

A implementação do programa envolve definir um ciclo que vá passando tantas vezes por cada jogador quantos os jogos definidos para cada jogador. Em cada novo jogo, para cada jogador, é gerada uma chave secreta que o jogador tentará descobrir até ao limite de tempo ou de jogadas. A forma como são representadas as chaves e tentativas internamente, assim como, o tipo de dados escolhido para codificar as cores fica ao critério de cada grupo, contudo, sempre que enviar as cores para o terminal estas devem ser representadas por letras maiúsculas de 'A' a 'L'. A chave deve ser representada por uma sequência de n caracteres de 'A' a 'L'), em que n é a dimensão da chave, e onde cada caractere representa uma cor diferente. O programa deverá a cada tentativa do jogador responder indicando o número de cores existentes e no lugar certo (tradicional pino preto) e o número de cores existentes mas no lugar errado (tradicional pino branco). Por exemplo nenhuma preta e nenhuma branca deverá ser "P0B0", duas pretas e uma branca deverá ser "P2B1".

3.3. Estatísticas

As estatísticas correspondem ao relatório final que deve ser apresentado no fim dos jogos antes de o programa concluir a execução.

Os resultados a apresentar deverão ser os seguintes:

- Nome do vencedor do torneio: Jogador com mais chaves certas em menos tempo médio de jogo.
- Nome do vencedor para o jogo mais rápido: Chave acertada em menos tempo (desempata pelo nº de jogadas).
- Nome do vencedor para o jogo mais curto: Chave acertada em menos jogadas (desempata pelo tempo de jogo).

4. Avaliação do Projecto Intermédio

A avaliação do projecto intermédio terá em conta os seguintes pontos:

- Submissão do trabalho obrigatoriamente via sistema FENIX até dia 5 de Abril, por cada dia (ou fracção de dia) de atraso a penalização será de 25% sobre a nota total.
- Critério de avaliação:
    - Relatório com fluxogramas (máximo 5 páginas), listagem devidamente comentada e boa estruturação do código (25% - 5 valores)
    - Leitura e validação de todos os parâmetros de inicialização (12.5% - 2,5 valores)
    - Correcta execução do jogo, conforme definido em 3.2 (50% - 10 valores)
    - Cálculo correcto e apresentação das estatísticas indicadas (12.5% - 2,5 valores)

A boa documentação do trabalho, assim como, a boa estruturação do código será valorizada na avaliação dos trabalhos. Por boa documentação entende-se um comentário geral no início do programa, um cabeçalho no início de cada função, comentários explicativos ao longo do código e a utilização de nomes claros para todas as funções e variáveis do programa. Por boa estruturação do código entende-se a clara divisão do código em funções curtas e com uma funcionalidade clara e, ainda, a definição e utilização de estruturas de dados de forma eficiente.

Nota1: Os alunos devem submeter um único ficheiro em formato zip com o nome *numero_do_grupo*.zip que inclua o ficheiro P*numero_do_grupo*.c em que o número do grupo deve ter três dígitos, por exemplo 001 para o grupo 1 e 002 para o grupo2. O código deverá ter um comentário inicial identificando os elementos do grupo e autores do trabalho e o relatório deverá ser um ficheiro em formato PDF. O código deverá ser compilado, com a opção – Wall (todos os *warnings*), e executado sem erros nem avisos em linha de comando. A não execução do programa corresponderá à anulação do projecto, ou seja, é melhor um programa incompleto que execute do que um programa completo que não compile!

Nota2: Os projectos submetidos serão passados por um sistema de verificação de plágios. A detecção de plágio ou a utilização de código não original corresponderá à **reprovação automática na disciplina e abertura de processo disciplinar**.

Projecto Final - Mastermind

O projecto final consiste numa evolução do projecto intermédio. Nesta fase os alunos deverão fazer uso dos conhecimentos sobre apontadores, estruturas e alocação dinâmica de memória para resolver o problema proposto. Em termos de funcionalidades pretende-se que o programa passe a ter a funcionalidade de descobrir uma chave secreta gerada pelo utilizador, ou pelo próprio programa, e ainda mantenha um registo dos jogos realizados.

5. Fases de Desenvolvimento

A implementação deve seguir as fases de desenvolvimento, descritas nas sub-secções seguintes.

5.1. Implementação do Algoritmo de Jogo

O algoritmo de jogo a implementar embora não sendo o mais eficiente é seguramente bastante simples de implementar. Resumidamente, o algoritmo deverá começar por gerar uma tentativa para adivinhar a chave, de forma aleatória e de acordo com os parâmetros de inicialização (ver projecto intermédio). Posteriormente, deverá percorrer o espaço de soluções (chaves possíveis) e escolher uma nova combinação que seja compatível com as respostas recebidas nas tentativas anteriores. O jogo acaba quando gerar a chave correcta. O algoritmo a implementar deve ter um parâmetro (a introduzir pelo utilizador) que indica o número de tentativas geradas aleatoriamente antes de iniciar a pesquisa sistemática conforme descrito anteriormente.

O algoritmo de jogo a implementar deverá seguir os seguintes passos:

1. O utilizador insere a chave secreta ou esta é gerada pelo próprio computador (modo computador vs computador).
2. O programa gera um número de tentativas aleatórias até atingir o limite de tentativas aleatórias indicadas durante a leitura de parâmetros de inicialização.
3. O programa dá resposta às tentativas geradas em 2 (por exemplo “P0B0” para nenhuma preta e nenhuma branca ou “P2B1” para duas pretas e uma branca). Se acertou em alguma delas acaba o jogo (passo 6), caso contrário, volta para o passo 2 se houver mais tentativas aleatórias ou segue para o passo 4, se não houver.
4. O programa gera uma nova tentativa compatível com os resultados das tentativas anteriores, ou seja, uma nova tentativa que produza as mesmas respostas do que as obtidas em todas as tentativas anteriores.
5. O programa dá resposta à nova tentativa e se acertar na chave secreta acaba o jogo (passo 6), caso contrário, segue para o passo 4.
6. Fim de jogo. O jogo realizado é armazenado na estrutura de dados de registo de jogo conforme descrito na secção 5.2.

Para esta fase final o corpo docente disponibilizará um par de ficheiros (oraculo.h e oraculo.o) que os alunos deverão incorporar na sua implementação. O ficheiro de interface oraculo.h possui as assinaturas das funções codificadas em oraculo.o, assim como a definição de um tipo de variável para que a informação produzida em oraculo.o seja acessível pelo programa cliente que os alunos vão desenvolver.

Basicamente, este módulo serve para gerar e manter secreta uma chave, de acordo com as especificações dadas pelo programa cliente, e depois receber tentativas de solução que classifica, permitindo o acesso a essa classificação.

5.2. Registo de Jogo

Os jogos realizados devem ser registados numa lista de listas conforme indicado na figura:

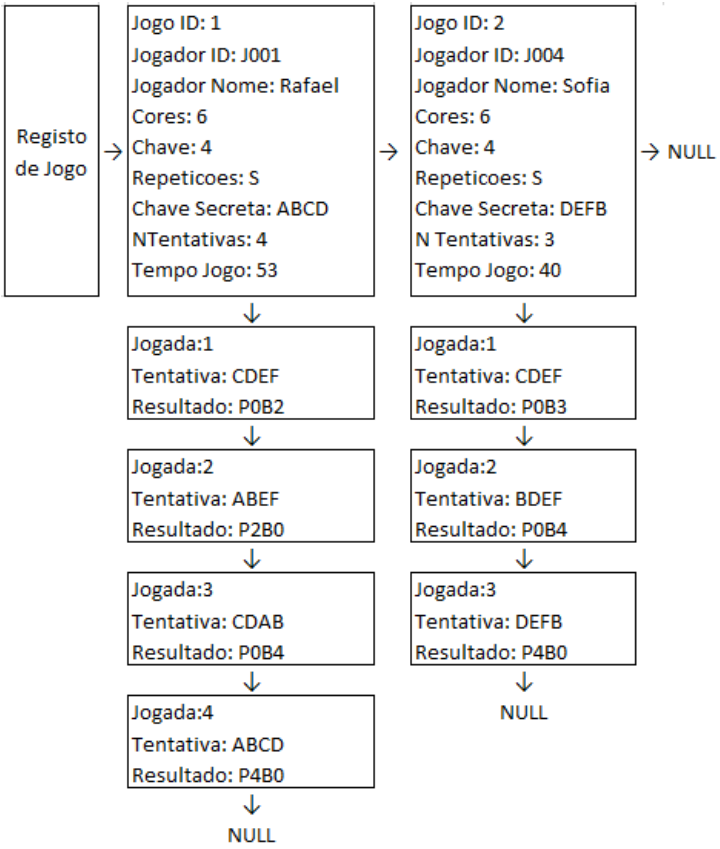

Fig. 2: Estrutura de dados para registo dos jogos.

No arranque do programa deve agora ser dada a possibilidade carregar o histórico dos jogos realizados de um ficheiro *historico.dat* e, no fim do programa, guardado o registo de jogos actualizado no ficheiro *historico.dat*.

O formato do ficheiro *historico.dat* deve ser o seguinte tendo em conta o exemplo da lista mostrada anteriormente:

```
1 J001 Rafael 6 4 S ABCD 4 53
1 CDEF P0B2
2 ABEF P2B0
3 CDAB P0B4
4 ABCD P4B0
2 J004 Sofia 6 4 S DEFB 3 40
1 CDEF P0B3
2 BDEF P0B4
3 DEFB P4B0
.....
```

Fig. 3: Exemplo de estrutura de dados armazenada no ficheiro historico.dat

Por fim, deve ainda ser possível ordenar a lista por ordem de jogo mais rápido para jogo mais lento e de jogo mais curto para jogo mais longo, sempre tendo em conta as diferentes categorias de jogo (número de cores, dimensão da chave e opção de repetição). Assim, antes de armazenar no ficheiro historico.dat o programa pergunta ao utilizador se pretende ordenar os jogos do mais rápido para o mais lento (fast) ou do mais curto para o mais longo (short), colocando no ficheiro o registo com ou sem ordenação conforme a resposta do utilizador.

5.3. Execução do Programa

O programa pode ser utilizado em modo interactivo, em que o jogador insere os dados durante a execução do programa, ou em modo de teste. No modo de teste o programa é executado com um conjunto de argumentos e sem qualquer interação com o jogador.

No modo interativo o programa deve ser lançado apenas com o seguinte comando na janela de texto:

- nomedoprograma

No modo de teste o programa executa-se com o seguinte comando:

- nomedoprograma -i init.dat -h historico.dat -o fast/short

O argumento -i indica que a seguir é indicado o ficheiro de inicializações, init.dat, ilustrado na fig. 4. Neste caso, pretende-se que seja o programa a executar autonomamente o jogo permitindo assim de uma forma fácil testar o programa e gerar um ficheiro de histórico rapidamente. O argumento -h indica que a seguir surge o nome do ficheiro com o histórico e, por fim, o argumento, -o indica que no fim do programa o histórico deve ser ordenado tendo em conta o melhor tempo de jogo (fast) ou o jogo mais curto (short). Os argumentos (aos pares) podem ou não estar presentes e podem ter qualquer ordem.

```
computer // nome do jogador (sem espaços)
100      // número de jogos
10       // número de cores
6        // dimensão da chave
S        // com ou sem repetições
5        // número de tentativas aleatórias
20       // número máximo de tentativas
```

Fig. 4: Exemplo do ficheiro init.dat

O modo de teste pode ainda ser executado simplesmente com o comando:

- nomedoprograma -h historico.dat -o fast/short

Neste caso, é apenas feita uma leitura do registo de jogos, de seguida é realizada uma ordenação do registo de jogos e, por fim, o registo de jogos é de novo armazenado no ficheiro historico.dat.

6. Avaliação do Projecto Final

A avaliação do projecto Final terá em conta os seguintes pontos:

- Submissão do trabalho obrigatoriamente via sistema FENIX até dia 24 de Maio, por cada dia (ou fracção de dia) de atraso a penalização será de 25% sobre a nota total.
- Critério de avaliação
    - Relatório com fluxogramas (máximo 5 páginas), listagem devidamente comentada e boa estruturação do código (25% - 5 valores)
    - Leitura e escrita correcta do registo de jogos em ficheiro (25% - 5 valores)
    - Correcta execução do jogo (25% - 5 valores)
    - Execução do programa nos modos interactivos e de teste (25% - 5 valores)

A boa documentação do trabalho, assim como, a boa estruturação do código será valorizada na avaliação dos trabalhos. Por boa documentação entende-se um comentário geral no início do programa, um cabeçalho no início de cada função, comentários explicativos ao longo do código e a utilização de nomes claros para todas as funções e variáveis do programa. Por boa estruturação do código entende-se a clara divisão do código em funções curtas e com uma funcionalidade clara e, ainda, a definição e utilização de estruturas de dados de forma eficiente.

Nota1: Os alunos devem submeter um único ficheiro em formato zip com o nome *numero_do_grupo*.zip que inclua os vários ficheiros do projecto devendo o principal ter o nome PF*numero_do_grupo*.c em que o número do grupo deve ter três dígitos, por exemplo 001 para o grupo 1 e 002 para o grupo2. O código deverá ter um comentário inicial identificando os elementos do grupo e autores do trabalho e o relatório deverá ser um ficheiro em formato PDF. O código deverá ser compilado, com a opção – Wall (todos os warnings), e executado sem erros ou avisos em linha de comando. A não execução do programa corresponderá à anulação do projecto, ou seja, é melhor um programa incompleto que execute que um programa completo que não compile!

Nota2: Os projectos submetidos serão passados por um sistema de verificação de plágios. A detecção de plágio ou a utilização de código não original corresponderá à **reprovação automática na disciplina e abertura de processo disciplinar**.