Programação 2016/2017

Mestrado em Engenharia Electrotécnica e de Computadores (MEEC)

Projeto de Programação: Avaliação Intermédia

1 Introdução

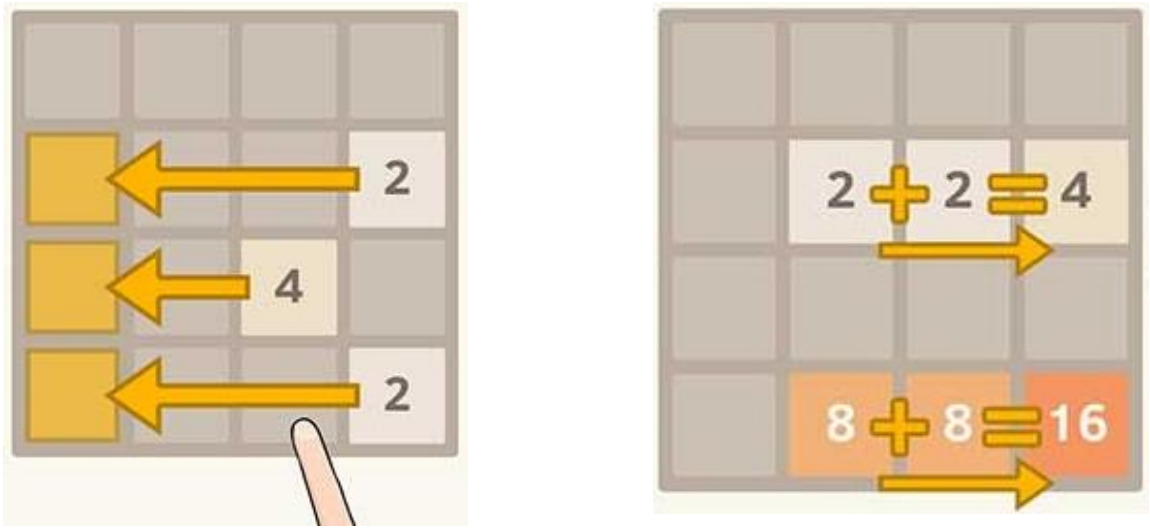
Com este projeto pretende-se que os alunos desenvolvam um jogo de raciocínio intitulado 1024, em que o objectivo é deslizar peças numeradas numa grelha e combiná-las para criar uma peça com um dado valor. Este jogo é inspirado no jogo 2048 que é jogado num tabuleiro 4x4 com peças numéricas que deslizam suavemente quando os jogadores as movem usando quatro teclas. O jogo 2048 foi criado num único fim de semana em março de 2014 por *Gabriele Cirulii* e recebeu mais de 4 milhões de downloads numa semana. Para poderem perceber melhor a dinâmica do jogo podem consultar http://2048game.com/.

2 Dinâmica do Jogo 1024

A dinâmica deste jogo funciona de acordo com as seguintes regras:

- O jogador pode mover as peças escolhendo uma das 4 direções: cima, baixo, direita e esquerda. Todas as peças se devem mover de acordo com a direção escolhida, caso não estejam impedidas por um obstáculo.
- Após cada jogada, uma nova peça com o valor de 2 ou 4 aparece aleatoriamente num local vazio.

- As peças deslizam na direção escolhida até que sejam paradas por qualquer outra peça ou pela borda do tabuleiro, como ilustrado na Figura 1 a).
- Se duas peças com o mesmo valor colidirem durante a movimentação, elas fundir-se-ão numa única peça com o valor igual à soma dos valores das duas peças que colidiram, como ilustrado na Figura 1 b).
- A peça resultante não pode fundir-se com outra peça novamente na mesma jogada. No entanto, várias peças podem fundir-se após a escolha da direção por parte do jogador.

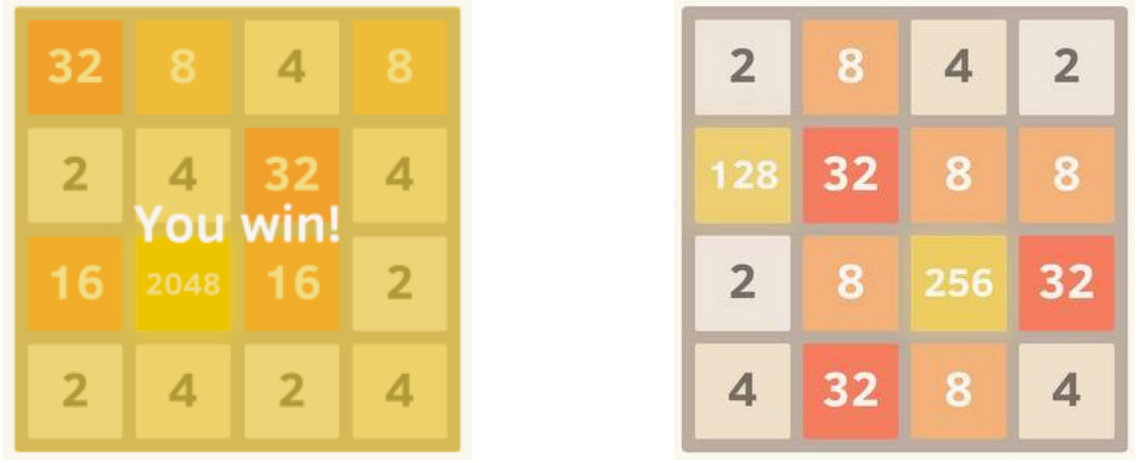


(a) (b)

Figure 1 - Dinâmica do jogo 1024: a) Movimento das peças quando é escolhido a direção ←; b) Fusões de duas peças.

As derrotas e vitórias deste jogo obedecem às seguintes regras:

- **Vitória**: Sempre que o jogador conseguir obter uma peça com o valor determinado pela dificuldade do jogo, deve ser assinalada vitória, como ilustrado na Figura 2 a). A dificuldade do jogo é um parâmetro lido no início do programa (*ver Seção 4*).
- **Derrota**: Ocorre uma derrota sempre que não for possível o jogador escolher uma direção que leve a uma fusão de peças e o tabuleiro esteja totalmente preenchido com peças, como ilustrado na Figura 2 b).



(a) (b)

Figure 2 - Dinâmica do jogo 1024: a) Vitória; b) Derrota. Em ambos os casos o objectivo era chegar a 2048.

Ao longo do jogo é essencial mostrar a dificuldade do jogo, a pontuação obtida e o tempo decorrido (ver Seção 5). A pontuação é um valor que deve ser iniciado a zero e incrementado sempre que ocorrer uma fusão de peças com o valor da peça obtida após a fusão.

3 Funcionamento do Programa

O funcionamento do programa pode ser sumarizado da seguinte forma:

- Leitura dos parâmetros de configuração (ver Seção 4).
- Mostrar o tabuleiro (quadrado) com o número de casas especificado.
- Um jogo deve começar apenas quando o utilizador carregar na tecla **n**. A qualquer altura pode ser iniciado um novo jogo, mesmo quando o jogo ainda não acabou.
- Inicialmente, duas peças devem ser colocadas em posições do tabuleiro aleatórias. Estas peças podem assumir os valores 2 e 4. O valor de cada peça (2 ou 4) deve ser gerado aleatoriamente.
- Depois de serem distribuídas as duas peças iniciais, é necessário permitir aos jogadores escolher qual é a direção em que pretendem movimentar as peças, utilizando para tal as setas do teclado.
- As peças devem ser movidas de acordo com a direção escolhida.
- Sempre que ocorrer uma colisão, deve ser efetuada uma fusão numa única peça.
- Devem atualizar a pontuação do jogo após cada jogada.
- Devem permitir ao jogador fazer *undo*, isto é, anular a última jogada feita, voltando o tabuleiro e pontuação ao estado antes de efetuar jogada. Não é necessário realizar mais do que um nível de *undo*. A tecla **u** deve ser usada para esta funcionalidade.
- O tempo do jogo deve começar a zero após o utilizado carregar na tecla **n** e deve ser incrementado sempre que passar um segundo de jogo. Após a vitória ou derrota o valor do tempo não deve ser incrementado.
- Naturalmente, não devem deixar um jogador continuar a jogar depois de este ter ganho ou perdido.
- Quando o programa terminar, deverá ser escrito um ficheiro de estatísticas contendo as estatísticas de todos os jogos concluídos (ver Seção 4). Estas estatísticas devem estar guardadas num vector.
- O jogador pode sair a qualquer momento da aplicação carregando na tecla **q**.

Naturalmente, é necessário atualizar a interface gráfica para mostrar o movimento das peças ao jogador, sempre que este efetuar uma jogada. Em relação ao código, é obrigatório que o tabuleiro com as peças seja representado como uma matriz 2D de inteiros. Não podem ser usadas estruturas de dados.

Escolha uma das seguintes funcionalidades **avançadas** e faça a sua implementação:

- Implementação de um sistema para permitir guardar (*save*) um jogo que esteja a decorrer, sempre que o utilizador saia da aplicação. Quando o jogador iniciar de novo a aplicação, deve ser possível continuar a jogar o jogo a partir do estado guardado. Não é preciso definir nenhuma nova tecla.
- Fazer mais do que um nível de *undo*, por exemplo, suportar até 25 níveis de *undo*.

Não deverão implementar estas funcionalidades avançadas sem terem as restantes funcionalidades a funcionar corretamente, uma vez que poderão ser penalizados caso o façam.

4 Leitura dos Parâmetros de Jogo e Escrita de Ficheiros

O funcionamento do jogo deve ser ditado por um conjunto de parâmetros que devem ser introduzidos pelo utilizador (através do teclado) antes de começar qualquer jogo:

- **Tamanho do tabuleiro:** Apenas devem suportar tabuleiros quadrados. Não pode haver tabuleiros com mais de 11x11 casas.
- **Nome do jogador**: Este nome não pode ser superior a 8 letras.
- **Dificuldade do jogo:** Uma vitória será alcançada logo após o jogador conseguir obter uma peça com o valor estabelecido por este parâmetro. Os níveis de dificuldade devem ter uma potência de 2 e não podem ser superiores a 2^{23} e inferiores a 2^{4}.

Cada parâmetro de entrada deve ser validado e caso o utilizador não tenha introduzido um parâmetro válido deve ser pedido de novo ao utilizador esse parâmetro, sinalizando a ocorrência de erro.

Quando o programa terminar é necessário escrever um ficheiro com o nome *stats.txt* que indique a seguinte informação para cada jogo:

- Nome do jogador.
- Pontuação máxima.
- Peça com o maior valor que o jogador conseguiu obter.
- Tempo que demorou a jogar a este jogo.

5 Aspecto Gráfico

A aplicação *1024* deverá ter um aspeto gráfico semelhante ao que está ilustrado na Figura 3.

Figure 3 – Interface gráfica do jogo 1024. No topo da interface devem mostrar o nível, a pontuação e os segundos que decorreram.

A aplicação deverá mostrar o nome da aplicação (1024) e o autor da aplicação (nome e número do aluno). Para realizar a parte gráfica devem utilizar o código de apoio ao projeto. No entanto, note-se que necessitam de modificar o código de forma a mostrar algumas estatísticas do jogo, nomeadamente o nível de dificuldade, a pontuação e o tempo. Quando o utilizador ganhar ou perder um jogo, também devem indicar a vitória ou derrota na janela da aplicação.

6 Desenvolvimento do Jogo

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

6.1 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 não faz nada ou muito pouco.

Assim sugerem-se os seguintes passos, pela ordem apresentada, para realização do projeto:

- Inicialização da biblioteca SDL e criação da janela gráfica (código fornecido pelo Prof.).
- Leitura dos parâmetros de funcionamento do programa.
- Geração aleatória de novas peças e movimento das peças existentes.
- Fusão correcta das peças.
- Atualização da interface gráfica.
- Interação com utilizador e reconhecimento das teclas das setas, **n** e **q**.
- Cálculo correto da pontuação e do tempo decorrido.
- Escrita na janela do jogo do nível, pontuação e tempo decorrido.
- Implementação da opção *undo* (tecla **u**).
- Escrita do ficheiro com os resultados dos jogos (estatísticas).
- Implementação de uma das funcionalidades de escolha:

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

6.2 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.

6.3 Indentação

Um ponto fundamental na organização da 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.

6.4 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, devem considerar que as funções devem caber num único ecrã, isto é, devem ter no máximo cerca de 30 linhas. Também devem 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 !

6.5 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.

6.6 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.

6.7 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

7 Submissão

Os alunos deverão submeter o código desenvolvido através do sistema FENIX até 15 de Abril. Apenas será necessário entregar o código correspondente ao programa desenvolvido.

Só será aceite um ficheiro de texto com extensão **.c**. Não utilize um processador de texto (e.g. Word) para formatar o seu programa.

8 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 intermédia, todo o código deve ser realizado individualmente! Se uma cópia for detetada, os alunos envolvidos na cópia serão penalizados.

9 Avaliação do Projeto

A avaliação do projeto terá em conta diversos parâmetros:

- Funcionalidades implementadas.
- Cumprimento das regras do jogo e das especificações do projeto.
- Estruturação da aplicação, nomeadamente o uso de funções.
- Tratamento de erros.
- Comentários e legibilidade do código.