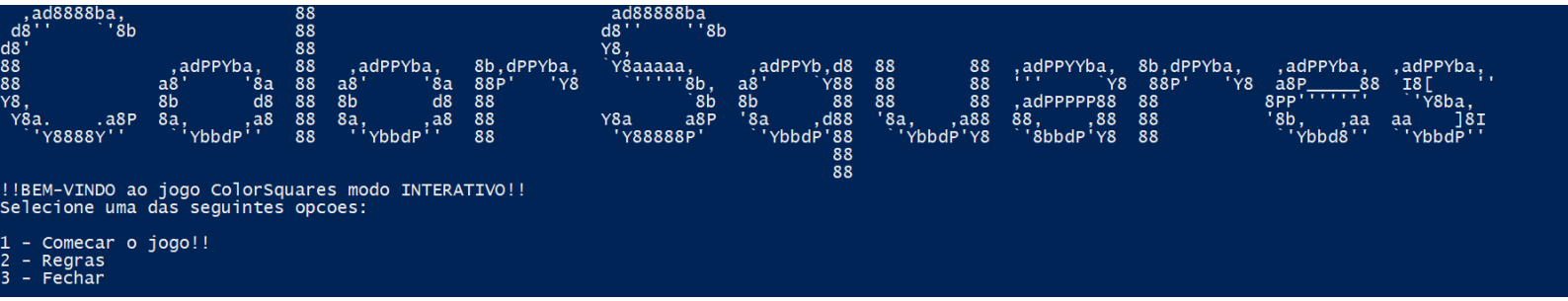
UNIVERSIDADE DE ÉVORA

Color Squares



Trabalho elaborado por:Miguel de Carvalho nº43108

Nuno Sousa nº43014

Ricardo Oliveira nº42647

Professora:Teresa Gonçalves

Relatório

Após uma boa interpretação do trabalho proposto, começou-se por fazer as funções indicadas: marcar, pontuação, gravidade, coluna, jogada, mostrar. Passa-se a explicar a função de cada uma delas em seguida.

- Função "coluna" -

Esta função do tipo void consegue reconhecer e mover uma coluna vazia no tabuleiro do jogo. Depois de encontrar uma, irá a movimentar para o lado direito do mesmo, juntamente com todas as outras que posteriormente tinha detetado, nunca alterando o tamanho do tabuleiro. Dando assim a ilusão de que a coluna colapsou e que os blocos se moveram para fechar a separação.

A função recebe os seguintes argumentos**: int tabuleiro** e **int sz,** sendo respetivamente, a matriz correspondente à estrutura do tabuleiro e o tamanho do mesmo.

Irá percorrer todas as colunas do tabuleiro, contando o número de espaços vazios (zeros) que encontra em cada uma delas. Se for uma coluna vazia, ou seja, o número de zeros for igual ao tamanho (sz) do tabuleiro, a função para cada um dos zeros igualará ao valor da mesma linha e da próxima coluna, tornando de seguida esse igual a 0, com isto é dada a ilusão que o zero se desloca ao longo da linha da esquerda para a direita. Dando o exemplo de uma só linha da coluna vazia (de zeros):

0 1 2 3 ⟶ 1 1 2 3 ⟶ 1 0 2 3 ⟶ . . . . ⟶ 1 2 3 0

Na função são declaradas as seguintes variáveis:

- **int n_z**: é encarregue de contar o número de zeros (espaços vazios) que se encontram em cada uma das colunas do tabuleiro.
- **int ocupacao**: é responsável por contar o número de colunas vazias presentes no tabuleiro.
- **int p_actual**: tem como função registar qual a coluna que está a ser avaliada no momento.
- **int limite**: atribui um limite para o qual é necessário avaliar as colunas do tabuleiro, sendo definido como *limite=sz-ocupacao*, logo, retira ao tamanho do tabuleiro as colunas vazias detetadas, não as avaliando, tornando assim a função mais efeciente.
- **int temp** e **int i**: serão variáveis auxiliares que ajudarão a restringir o número de vezes que um ciclo se repete.

Análise detalhada:

No início da função, é atribuído o valor 0 (zero) a cada uma das seguintes variáveis: *n_z*, ocupacao e *p_actual*, já que estes são todos contadores que, irão registar o que já foi referido acima.

Com o "while(p_actual<limite)" a função lerá todas as colunas até ao limite estabelecido. Se o primeiro elemento da coluna for 0 "if(tabuleiro[0][p_actual]==0)", então o resto dos elementos

serão lidos, até que seja encontrado um diferente de 0, sendo o *n_z* recetado para 0 e o ciclo terminado com um "break". Se for encontrado um 0: n_z++.

Se o valor de *n_z* for igual ao *sz* "if(n_z==sz)", então é porque foi detetada uma coluna vazia. Sendo assim, *temp* assumirá o valor 0 até que este assuma o valor de *sz*, *temp*++. E *i* o valor de *p_actual* até que este assuma o valor do *limite*, *i*++. Para todas as linhas de índice *temp* do tabuleiro e as repetivos elementos de indíce *i*, é atribuido o valor do elemento seguinte (*i+1)* e a este o valor zero: "tabuleiro[temp][i] = tabuleiro[temp][i+1]", "tabuleiro[temp][i+1] = 0". Com isto, é dada a ilusão que o zero se desloca ao longo da linha, da esquerda para a direita. Depois de todos os elementos da coluna vazia sejam "deslocados", *ocupacao*++ e o ciclo termina com um "break".

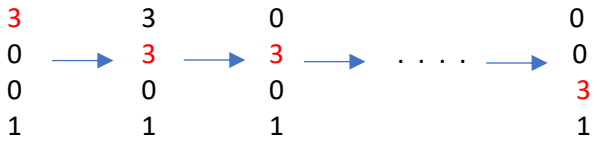
Se n_z for diferente de *sz*, então continuasse para a próxima coluna, p_actual++.

-Função "gravidade"-

Esta função void é responsável por simular a gravidade no jogo. Se um número se encontra por cima de um espaço vazio, este será "puxado" para baixo até que por debaixo dele se encontre outro número ou que chegue à base do tabuleiro.

A função recebe os seguintes arguementos**: int tabuleiro** e **int sz,** sendo respetivamente, a matriz correspondente à estrutura do tabuleiro e o tamanho do mesmo.

Irá percorrer todos os elementos do tabuleiro, coluna por coluna de cima para baixo. Se encontrar um 0 (espaço vazio) irá verificar se o elemento em cima é diferente de 0, se isso se verificar, então o a posição do zero toma o valor do número e o número o valor de 0. Após a troca dos dois elementos, o tabuleiro é novamente verificado até não existirem mais números sobre espaços vazios. Dando um exemplo em que isto acontece:



Na função são declaradas as seguintes variáveis:

- **int c** e **int l**: representam respetivamente o índice da coluna e da linha da matriz que constitui o tabuleiro.

Análise detalhada:

Para todas as colunas "for(int c = 0 ; c < sz ; c++)" e linhas "for(int l = 0 ; l < sz ; l++)" do tabuleiro, a função irá verificar após ser detetado um 0 (espaço vazio), se o valor abaixo é diferente de 0 e se este não se encontra na base do tabuleiro "l-1 >=0", da seguinte maneira: "if(tabuleiro[l][c] == 0 && tabuleiro[l-1][c] != 0 && l-1 >=0)". Se isto ocorrer, então o zero e o número por cima dele são trocados de posição: "tabuleiro[l][c] = tabuleiro[l-1][c]" em seguida de "tabuleiro[l-1][c] = 0". Efetuada a troca de valores, as variáveis *c* e *l* são recetadas e a matriz é analisada novamente até não existirem mais números sobre zeros.

-Função “mostrar”-

Esta função do tipo void é relativamente simples e tem como objetivo principal, mostrar visualmente ao jogador o estado atual do tabuleiro e transformar todos os zeros em traços, dando a ideia de um espaço sem número.

É chamada no final de cada jogada quando todas as outras funções já tenham terminado de alterar estrutura do tabuleiro, expondo a disposição do mesmo após a jogada.

Na função são declaradas as seguintes variáveis:

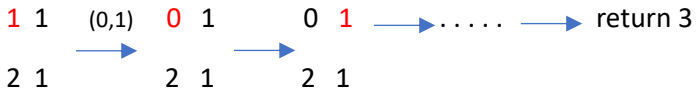
- **int c** e **int l**: representam respetivamente o índice da coluna e da linha da matriz que constitui o tabuleiro.

Para todas as colunas “for(int c = 0 ; c < sz ; c++)” e linhas “for(int l = 0 ; l < sz ; l++)” do tabuleiro, a função imprimerá todos os elementos da matriz: “printf("%d", tabuleiro[l][c])”.

-Função “marcar”-

Esta função do tipo int tem como principal objetivo detectar os grupos formados pelos quadrados do tabuleiro, eleminá-los e devolver o número de quadrados que perteciam ao grupo.

Quando o jogador introduz as coordenadas de uma jogada, a função irá procurar nos quatro lados desse quadrado se algum tem o mesmo valor que o escolhido. Se encontrar eleminará o valor do anterior e passará a ao próximo, repetindo o processo. Vejamos um caso em que isto acontece:



A função recebe os seguintes argumentos**: int tabuleiro** e **int sz,** sendo respetivamente, a matriz correspondente à estrutura do tabuleiro e o tamanho do mesmo**. int x** e **int y** que, correpondem às coordenadas da jogada.

Na função são declaradas as seguintes variáveis:

- **int num_quadrados**: é a variável que armazena o número actual de quadrados contados.
- **int fixo**: é a variável que armazena o valor do quadrado em estudo.

Análise detalhada:

É usada recursividade no uso da função, já que assim é mais prático a contagem de quadrados bem como aplicar exatamente o mesmo processo a cada novo quadrado que encontra. “if((x < sz && y < sz) && (x >= 0 && y >= 0)” é usado para limitar os quadrados que a função estuda, dentro do tabuleiro.

Começa por guardar o valor do quadrado na variável *fixo* e apaga o seu valor (igualar a 0). O número de quadrados começa no valor 1 porque, quando a função foi chamada pelo menos um quadrado faz parte do grupo.

Analisemos quando a função encontra um quadrado igual à direita do em estudo (x+1), o número de quadrados contados será igual ao atual mais o seguinte, sendo que a função de repete até nao encontrar mais quadrados com o valor *fixo,* zerando todos os que analisa. No final dá return *num_quadrados.*

-Função "pontuação"-

Tem apenas uma intenção, cálcular e devolver a pontuação de uma jogada da seguinte forma: "pontos = num_quadrados * (num_quadrados + 1) / 2". Então esta função apenas tem como argumento **int num_quadrados**, que recebe da função "marcar".

-Função "jogada"-

A sua intenção é reunir ambos a função "marcar" e "pontuação" em uma só. Permitindo assim ao programa retirar um grupo de quadrados e a pontuação da jogada.

A função recebe os seguintes argumentos**: int tabuleiro** e **int sz,** sendo respetivamente, a matriz correspondente à estrutura do tabuleiro e o tamanho do mesmo**. int x** e **int y** que, correpondem às coordenadas da jogada

Foram utilizadas outras funções específicas ao modo iterativo que, tornam o programa mais agradável e prático:

-Função "fim"-

É uma função int simples cujo objetivo é saber se o jogo terminou; para isso a função irá verificar se o elemento da matriz que se encontra no canto inferior esquerdo é igual a 0. No modo iterativo, graças às funções "gravidade" e "coluna" haverá sempre um número diferente de 0 no canto enquanto o jogo não terminar, logo, se houver "if(tabuleiro[sz-1][0] == 0)" a função dará return 1, ou então return 0 se não for 0.

A função recebe os seguintes argumentos**: int tabuleiro** e **int sz,** sendo respetivamente, a matriz correspondente à estrutura do tabuleiro e o tamanho do mesmo.

-Função "intro_inter"-

O propósito desta função do tipo int é proporcionar ao jogador um menu (recurso à função printf) para mais fácil entender o funcionamento do jogo, oferencendo opções tais como ver as regras do mesmo; permitindo assim, que qualquer pessoa possa jogar sem contexo

A função dá return 1 se o jogador quiser começar o jogo e return 0 se o quiser fechar.

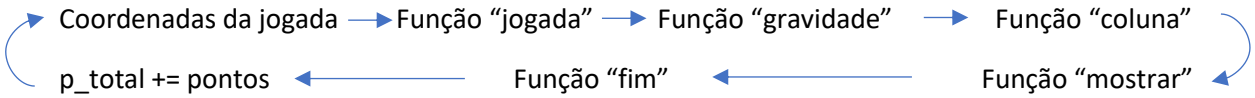
-Modo Iterativo-

Tendo todas as funções necessárias, começou-se a construir a "main" para o modo iterativo. A boa organização da mesma foi essencial para o programa funcionar corretamente, bem como a ordem em que as funções são chamadas.

Na main são declaradas as seguintes variáveis:

- **int sz**: armazena o valor do tamanho que o jogador pretende para o seu tabuleiro.
- **int c** e **int l**: representam respetivamente o índice da coluna e da linha da matriz que constitui o tabuleiro.
- **int x** e **int y**: conservam as coordenadas da jogada que o utilizador jogou.
- **int pontos** e **int p_total**: arquiva os pontos de uma jogada, soma os pontos de todas as jogas efetuadas, respetivamente.
- **int comecar:** guarda o valor da função "intro_inter".
- **int verifica_fim**: guarda o valor da função "fim".
- **int tabuleiro[50][50]**: é a matriz responsável pela estrutura do tabuleiro de jogo.
- **int i**: é uma variável auxiliar que ajudará a restringir o número de vezes que um ciclo se repete.

Ao iniciar o jogo a função "intro_inter" é chamada, que decidirá se o jogo é iniciado. Se começar, é pedido um tamanho para a tabuleiro que será aleatoriamente gerado com uso à função "rand". Após a criação do tabuleiro é chamada a função "mostrar" e a função "fim". Em seguida é criado um ciclo que irá parar quando o valor de verifica_fim mudar para 1:



No final do programa é imprimido a pontuação final "printf("Pontos totais: %d\n", p_total)".

-Modo Automático-

Neste modo é um ficheiro de texto que se encontra na mesma pasta que o resto dos ficheiros do programa, é lido de uma maneira organizada. O texto tem que ter a seguinte organização: na primeira linha, o tamanho da matriz; nas linhas abaixo terá que estar representada a matriz; em seguida o número de jogadas pretendidas; e na seguinte as coordenadas da jogada numa forma ordenada: x y, x y, x y,....

Na main são declaradas as seguintes variáveis:

- **int tabuleiro[50][50]**: é a matriz responsável pela estrutura do tabuleiro de jogo.
- **int sz**: armazena o valor do tamanho que o jogador pretende para o seu tabuleiro.
- **char fNome[50]**: armazena um array de caracteres como o nome do ficheito de texto.
- **char linha_coor[300]**: armazena um array de caracteres da linha respetiva às coordenadas da jogada.

- **char linha_mov[10]**: armazena um array de caracteres da linha respetiva ao número de jogadas.
- **char c_sz[3]**: armazena um array de caracteres da linha respetiva ao tamanho do tabuleiro.
- **char linha_tab[21]:** armazena um array de caracteres das várias linhas do tabuleiro.
- **int mov**: guarda o número de jogadas introduzidas.
- **int x** e **int y**: conservam as coordenadas da jogada que o utilizador jogou.
- **int p**: é uma variável auxiliar que controla o modo de como x e y são lidos sequêncialmente.
- **int elemento:** é o valor inteiro correspondente ao caracter do array **linha_tab[21]** em estudo.
- **int pontos** e **int p_total**: arquiva os pontos de uma jogada, soma os pontos de todas as jogas efetuadas, respetivamente.
- **int a** e **int i**: serão variáveis auxiliares que ajudarão a restringir o número de vezes que um ciclo se repete.

Ao iniciar o executável, o programa pede ao jogador o nome do ficheiro que queira que seja lido. Em seguida, o programa abrirá o ficheiro e começará a ler o seu conteúdo. Foram utilizadas as funções: “fgets” que, lê os caracteres de uma linha do ficheiro e armazena-os numa string, sempre que é chamada novamente lerá a linha seguinte sem ler a anterior; e a “sscanf” que transforma uma string num número inteiro.

Usando estas duas funções pode-se ler e colocar o tamanho do tabuleiro na variável sz. Com o mesmo método é possível ler todas as linhas correspondentes à matriz repetindo a função “fgets” “sz vezes”. Para cada um dos caracteres de uma linha da matriz, podemos os converter para inteiro e armazenar-los na variável *elemento* subtraindo o caracter ‘0’. Repete-se o procedimento para o número de jogadas.

As coordenadas são lidas “mov vezes” e os valores de *x* e de *y* estão a 5 posições dos próximos valores para *x* e *y*. Sendo que o primeiro valor de x está no índice 0 e o de y no índice 2. Logo, numa forma ordenada é possível transformar os elementos da string *linha_coor* que correspondem aos valores de x e y. Se o valor de tabuleiro[y][x] for igual a zero, então o programa não executará a jogada, se for diferente o programa jogará automáticamente num ciclo semelhante ao Modo Iterativo, porém não chamará a função “mostrar”, dando assim ao utilizador apenas a sua pontuação final.