UNIVERSIDADE DE ÉVORA

CURSO DE ENGENHARIA INFORMÁTICA

Programação I

João Silveirinha nº42575
Leonardo Catarro nº43025
Daniel Montinho nº41894

Introdução

O objetivo deste trabalho é desenvolver um jogo de tabuleiro vertical chamado “Color Squares”. Este mesmo tabuleiro é preenchido com quadrados de diferentes cores (Neste caso, com números de 1 a 4), e o jogador pode remover grupos com o mesmo número. Um grupo consiste em quadrados com lados partilhados.

Existem dois modos de jogo, o interativo e o automático.

No modo interativo, o tabuleiro inicial é gerado definindo aleatoriamente uma de quatro cores possíveis (1, 2, 3 e 4), para cada posição do tabuleiro. Em cada jogada, o jogador escolhe um dos quadrados do tabuleiro, sendo este atualizado de acordo com as regras do jogo, até que o tabuleiro fique vazio.

No modo de jogo automático, o tabuleiro inicial e as jogadas são lidas de um ficheiro de texto.

Desenvolvimento do jogo

Para a realização do jogo foram utilizadas as seguintes **funções:**

- int marcar(int tabuleiro[][], int sz, int x, int y)

```
{
int num_quadrados = 0;
for(x= 0; x< sz; x++)
  {
    for(y=0; y< sz; y++)
           {
              if(tabuleiro[x][y]==0)
                   {
                       num_quadrados++;
                   }
           }
  }
return num_quadrados;
}
```

Esta função marca no tabuleiro todos os quadrados que fazem parte do grupo do quadrado (x,y), e posteriormente devolve o número de quadrados do grupo, para que estes possam ser utilizados na função pontuação.

Nesta função, a variável “num_quadrados” funciona como um contador, para que sempre que um quadrado(x,y) for 0, a função ir contando estes quadrados. No final, a função vai retornar o número de quadrados (x, y) iguais a 0.

- int movimentos (int sz, int tabuleiro[][20], int x, int y)

```
{
if((x>=0 && x<sz) && (y>=0 && y<sz))
  {
int g = tabuleiro[x][y];

tabuleiro[x][y] = 0;

if(tabuleiro[x+1][y] == g){ movimentos (sz, tabuleiro, x+1, y);} else
if (tabuleiro[x-1][y] == g){movimentos (sz, tabuleiro, x-1, y);}   else
if (tabuleiro[x][y+1] == g){ movimentos (sz, tabuleiro, x, y+1);}
else if(tabuleiro[x][y-1] == g){ movimentos (sz, tabuleiro, x, y-1);}
  }
return 0;
 }
```

Esta função averigua os possíveis quadrados que possam fazer parte do mesmo grupo, ou seja, verifica se um quadrado e o seu adjacente tem a mesma cor (1, 2, 3, ou 4).

Condicionou-se as posições de x e y, para que estas fiquem dentro do tabuleiro. De seguida, foi criada uma variável auxiliar g (int g = tabuleiro[x][y]), pois verificou-se que sem esta, a função não funciona exatamente como é pedido. Posteriormente, verificaram-se se as peças acima da escolhida e a escolhida, têm a mesma cor (1, 2, 3, 4).

Por fim, chamou-se a função de forma recursiva, onde já estão implementadas as alterações das posições da peça, para que possam ser verificadas todas as posições acima, à esquerda, direita e abaixo da peça escolhida.

Caso não fosse necessário criar uma variável auxiliar, as condições ficariam da seguinte forma: (tabuleiro[x+1][y] == tabuleiro[x][y]), por exemplo.

- int pontuacao (int num_quadrados)

```
{
int pontos;
pontos= (num_quadrados*(num_quadrados + 1))/2;
return pontos;
}
```

Esta função calcula a pontuação de cada jogada.

(A fórmula utilizada para o cálculo dos pontos está presente no enunciado do trabalho.)

- void gravidade (int sz, int tabuleiro[][20])

```
{
int grav=0;
while (grav<sz){
for(int col=0; col<sz; col++){

for(int linha=0; linha<sz; linha++){
if(linha>0 && tabuleiro[linha][col]==0){
tabuleiro[linha][col]=tabuleiro[linha-1][col];
tabuleiro[linha-1][col]=0;}
                }
        }
                grav++;
        }
}
```

Esta função atualiza o tabuleiro de forma a que sempre que hajam zeros (depois de terem aparecido devido ao "rebentar" dos grupos de cores), estes desapareçam e as peças acima "caiam" e substituam as posições dos zeros. A função utilizada é do tipo void, pois não retorna nada (retorna vazio).

Foi criada uma variável "grav", que funciona como um contador, e por isso tem que ser declarada e igualada a 0 fora do ciclo while. Este ciclo while é utilizado para que o contador (grav), não exceda o tamanho do tabuleiro.

De seguida, foram criadas duas **condições** para que a função "gravidade" funcione corretamente:

- **(linha>0):** Pois se a linha for igual a 0, não é necessário aplicar o que já foi dito em cima. (Lembrando que neste momento ainda não foi feita a inversão do tabuleiro e por isso a posição (0,0) ainda é no canto superior esquerdo).
- **(tabuleiro[linha][col]==0):** Averigua se a posição(x,y) é igual 0.

Se ambas as condições forem verdadeiras, o tabuleiro é atualizado e as peças "caem" e substituem os 0.

Após o que foi dito e de terem sido fechadas as condições, o contador atua (apenas dentro do ciclo while).

- void coluna (int sz, int tabuleiro[][20])

```
{
int l, col;
for(col=0; col<sz; col++)
        {
int vazio = 0;
for (l=0; l<sz; l++)
                {
if(tabuleiro[l][col] == 0)
                        {
vazio++;
                        }
        }
if(vazio==sz)
         {
for (int i= 0; i<sz; i++)
                {
int j, bx;
j = col;
bx = tabuleiro [i][j];
while (j< sz-1)
                {
tabuleiro [i][j]=tabuleiro[i][j+1];
          tabuleiro[i][j+1]= bx;
j++;}}  }        } }
```

Esta função atualiza o tabuleiro de forma a que quando toda a coluna estiver vazia (coluna de zeros), esta colapsa e move os quadrados da direita para a esquerda de forma a fechar a separação entre as colunas.

Mais uma vez, foi utilizado um contador (vazio), para o número de zeros (sempre que a posição da peça fosse igual a 0). Mas, desta vez, foi necessário colocar o contador apenas dentro do “for” refente à coluna, para que apenas funcionasse para a coluna.

Depois, colocou-se a condição (vazio==sz), pois apenas se o contador (vazio), ou seja, o número de zeros da coluna, for igual ao tamanho da coluna é que esta colapsa.

Foram criadas duas variáveis auxiliares para que a função funcionasse corretamente pois sem a sua utilização, a função conteria “bugs” (j = col) (bx = tabuleiro [i][j]), respetivamente.

Depois, foi criado um ciclo “while (j<sz-1)”, para que a função funcione só até haver uma coluna (pois a primeira coluna é a 0).

- int jogada (int sz, int tabuleiro[][20], int x, int y)

```
{
int pontos=0;
movimentos (sz, tabuleiro, y, x);
coluna (sz, tabuleiro); gravidade
(sz, tabuleiro);
int num_quadrados = marcar (sz, tabuleiro, x, y);
pontuacao(num_quadrados);
pontos = pontos + pontuacao(num_quadrados);
printf ("Os seus pontos são: %d\n\n", pontos);
return 0;
}
```

Esta função executa uma jogada, devolvendo a pontuação da mesma. Para o funcionamento desta função, foi necessário chamar as outras funções que fazem parte do jogo, de um modo recursivo (tal como foi pedido no enunciado).

Para além disso, foram ainda declaradas as variáveis “pontos” e “num_quadrados”, esta ultima teve que ser igualada à chamada recursiva da função marcar, pois é devido ao retorno do número de quadrados da função marcar que a função pontuação (“pontuação (num_quadrados)”) funciona. A variável pontos serve para incrementar pontos á função pontuação a cada jogada.

Por último deu-se print ao número de pontos de forma a que estes sejam mostrados no terminal a cada jogada.

- void mostrar(int sz, int tabuleiro[][20])

```
{
for (int i = 0; i <sz; i++)
  {
for (int j = 0; j < sz ; j++)
    {
if(tabuleiro[i][j]==0)
      {
printf("-");
            }
else {
printf("%d", tabuleiro[i][j]);
                        }
    }
printf("\n");
  }
 }
```

Esta função mostra o tabuleiro atualizado a cada jogada feita.
Para além disso, devido a condição imposta, quando a peça é igual a 0, a função dá print ao “-”, ou seja transforma o 0 neste mesmo caracter.

- void aleatorio(int sz, int tabuleiro[][20])

```
{
    srand(time(NULL));
  for (int i = 0; i < sz; i++)
  {
    for (int j = 0; j < sz; j++)
    {
      tabuleiro[i][j] = rand() %4 + 1;
    }
  }
}
```

Esta função gera os valores aleatórios a serem colocados de forma aleatória no tabuleiro.
Esta última função não consta no documento onde estão as funções comuns aos dois modos de jogo pois esta é apenas utilizada para o modo interativo.

De realçar que, embora não tivesse sido referido anteriormente, todas as funções, à exceção da “jogada”, tem 2 “for”, um para as linhas, e outro para as colunas, de forma a que tudo fique dentro do tabuleiro como é pedido.

Dentro da função “main”, constam os outputs e inputs para o tamanho do tabuleiro e para as posições das jogadas, bem como a chamada recursiva da função “void aleatorio” para gerar os valores aleatórios, a chamada recursiva da função jogada e mostrar.
Por último consta então a inversão do tabuleiro “coluna_x= sz-1-coluna_x”, para que a posição (0,0) seja no canto inferior esquerdo.

Em relação à separação do programa em ficheiros, verificou-se algumas dificuldades em ligar tudo de forma a que funcione tal como se o programa apenas estivesse contido num só ficheiro. Porém, neste modo interativo, tudo ficou a funcionar como pedido.

No **modo automático**, mais uma vez, os diferentes ficheiros ficaram bem interligados, mas o mesmo não foi finalizado devidamente.

```
int main(int argc, char **argv)
{
FILE *fp = fopen("text", "r");    int
tamanho;
fscanf (fp, "%d", &tamanho);   printf("%d\n", tamanho);
int tabuleiro[20][20];
int i, j;
for(i=0; i < tamanho;i++)
        {
for(j=0; j< tamanho; j++)
                {
fscanf(fp, "%1d", &tabuleiro[i][j]);
printf("%1d", tabuleiro[i][j]);
                }
printf("\n");
        }
int numero_jogadas;
fscanf(fp, "%d", &numero_jogadas);
printf("%d\n", numero_jogadas);
return 0;
}
```

Primeiramente foi criado um ficheiro cujo nome é “text”, que contém exatamente aquilo que o enunciado pedia.

Com a seguinte linha “FILE *fp = fopen("text", "r");”, o ficheiro foi aberto, e depois devido ao “r”, foi lido.

Depois, a variável tamanho foi declarada para que o tamanho pudesse ser lido do ficheiro de texto (fscanf (fp, "%d", &tamanho);), e foi dado o print do tamanho para que este pudesse ser apresentado no terminal (printf("%d\n", tamanho);).

De seguida, foram colocados dois “for”, mais uma vez, um para as linhas e outro para as colunas do tabuleiro do ficheiro de texto, e dentro dos “for”, o tabuleiro foi lido do ficheiro de texto e foi feito o print no mesmo no terminal.

Por fim, foi declarada a variável “numero_jogadas”, para que o número de jogadas pudesse ser lido do ficheiro de texto e ser feito o print no terminal.

Conclusão

De um modo geral este trabalho ficou quase completo visto que faltou apenas implementar a parte do código para as jogadas serem lidas do ficheiro de texto e a consequente pontuação.

Verificou-se que esta última parte do trabalho não foi realizado, talvez não por falta de conhecimento sobre a matéria, mas por falta de raciocínio, visto que não foi encontrada uma maneira de relacionar as jogadas já pré-definidas do ficheiro de texto com variáveis, para que estas possam ser lidas do ficheiro de texto.