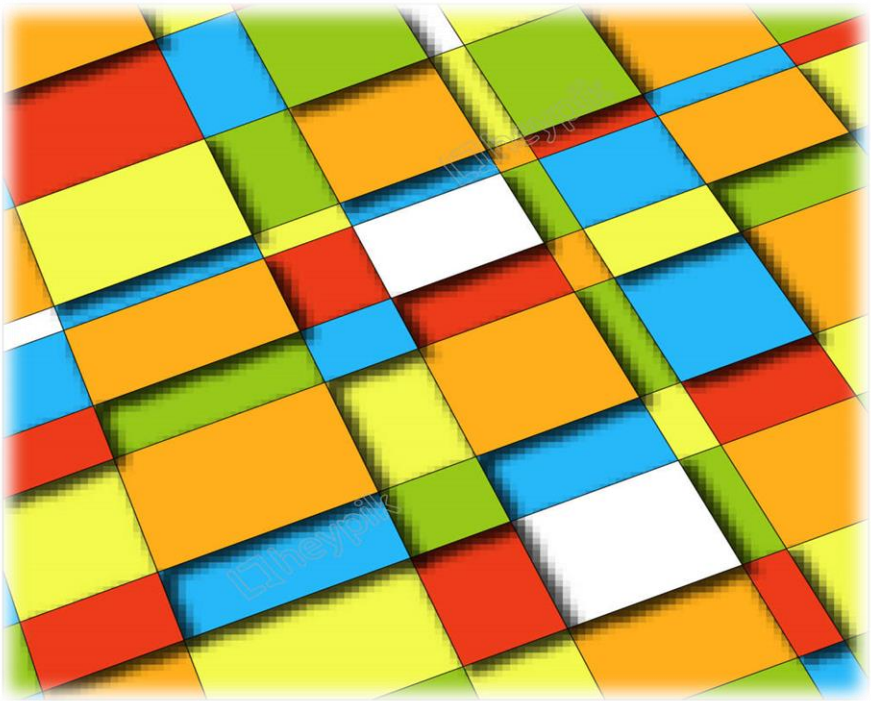
Programação I

Relatório Color Squares



Docente:

Teresa Gonçalves

Discentes:

Filipe Alfaiate nº43315

Vasco Barnabé nº42819

Diogo Porta-Nova nº42412

Introdução:

Perante um jogo bem conhecido no qual um tabuleiro (vertical) é preenchido com quadrados de cores diferentes e o jogador pode remover grupos com a mesma cor (o grupo consiste num conjunto de quadrados de lados partilhados) no qual, após cada movimento, o tabuleiro é atualizado de acordo com as regras a seguir apresentadas:

1) Gravidade: Indica que os quadrados acima da área vazia vão cair devido à gravidade;
2) Coluna: Indica que toda uma coluna que se apresentar vazia, irá colapsar movendo os blocos da direita para a esquerda de forma a fechar qualquer fenda/buraco/separação que exista.

Índice

1. Descrição do trabalho

Como nos foi implícito, este trabalho tem como finalidade desenvolver uma aplicação, que se possa jogar claro, denominada de Color Squares (Quadrados Coloridos).

Este jogo terá de conter 2 modos de jogos:

- ***Modo Iterativo***;
- ***Modo Automático***;

No modo iterativo o tabuleiro inicial é gerado de forma a selecionar aleatoriamente uma de 4 cores possíveis (1,2,3,4) para cada posição do tabuleiro. Em cada jogada, o jogador terá de escolher um desses mesmos quadrados do tabuleiro, sendo este atualizado de acordo e conforme as regras estipuladas para o jogo até que o tabuleiro fique vazio. É nesta altura que é apresentada a pontuação final do jogador.

No modo automático o tabuleiro e as jogadas são simplesmente "descarregados" de um ficheiro de texto e o programa terá e deverá de apresentar a pontuação final consoante esse mesmo ficheiro.

Imaginando que o tabuleiro tem dimensão 4 (4x4), a seguinte sequência de jogadas configuradas irá ser gerada:

- Escolher o valor x na posição (i,j) ⇔o valor 4 é apresentado na posição (2,2); Relembrando que na posição (i,j) o valor será 0 ⇔ posição (0,0) o valor será 0 no canto inferior esquerdo;
- Escolher o valor x na posição (i,j) ⇔ O valor 3 é apresentado na posição (2,1);

1	***2***	***3***	***2***
1	***3***	***4***	***2***
2	***4***	***4***	***1***
1	***2***	***3***	***2***

1	***-***	***-***	***2***
1	***2***	***-***	***2***
2	***3***	***3***	***1***
1	***2***	***3***	***2***

1	***-***	***2***	***-***
1	***-***	***2***	***-***
2	***2***	***1***	***-***
1	***2***	***1***	***-***

O jogador irá ganhar B pontos quando remover um grupo de um único quadrado; quando remover 2 quadrados pontua R pontos e por aí em diante agrupando grupos de 3 quadrados irá pontuar 6 pontos e assim sucessivamente.

A fórmula que indica o nº de pontos de uma jogada é a seguinte:

$$\text{Pontos}=\frac{num\_quadrados * (num\_quadrados+1)}{2}$$

2. Funções

2.1. void mostrar(int sz, int tabuleiro[ ] [sz])

```
 9  void mostrar(int sz, int tabuleiro[][sz]){
10      int i, j;
11      printf("\n");
12      for(i = sz-1; i >= 0; i--) {
13          for(j = 0; j < sz; j++){
14              if(tabuleiro[i][j] != -1){
15                  printf(" %d ", tabuleiro[i][j]);
16              } else {
17                  printf(" - ");
18              }
19          }
20          printf("\n");
21      }
22      printf("\n");
23  }
```

A função void mostrar(int sz, int tabuleiro[ ] [sz]) recebe como parâmetros o tamanho do tabuleiro de jogo (“sz”, do tipo inteiro) e todo o tabuleiro (“tabuleiro[ ] [sz]”, do tipo inteiro).

Esta função tem a funcionalidade de apresentar no ecrã o tabuleiro atualizado após cada jogada efetuada pelo jogador.

Foram criadas duas variáveis, “i” e “j” que servem para, através dos dois ciclos “for”, percorrer todo o tabuleiro (sendo “i” o número da linha e “j” o número da coluna). São necessárias duas variáveis uma vez que se trata de um array bidimensional.

No “if”, se a cor correspondente ao quadrado em questão for diferente de -1 (-1 pois nas funções int marcar, void coluna e void gravidade, a todo o quadrado que não representa nenhuma cor, é-lhe atribuído o valor -1, para o identificar como quadrado vazio) então essa cor será impressa no tabuleiro nesse quadrado. Caso não seja diferente de -1, isto é, aquele quadrado não tem

nenhuma cor atribuída, aparecerá no ecrã um simples “-“ para simbolizar a ausência de cor naquele quadrado.

Quando todo o tabuleiro tiver sido percorrido, o programa sai dos ciclos “for” fazendo os “printf(“\n”);” apenas por uma questão de apresentação no ecrã.

2.2. int marcar(int sz, int tabuleiro[ ] [sz], int x, int y)

```
26  int marcar(int sz, int tabuleiro[][sz], int x, int y){
27    int count;
28      if(x >= 0 && y >= 0 && x < sz && y < sz){
29          int peca = tabuleiro[x][y];
30          tabuleiro[x][y] = -1;
31          count = 1;
32
33          if(x > 0 && tabuleiro[x-1][y] == peca){
34              count += marcar(sz, tabuleiro, x-1, y);
35          }
36
37          if(y < (sz-1) && tabuleiro[x][y+1] == peca){
38              count += marcar(sz, tabuleiro, x, y+1);
39          }
40
41          if(x < (sz-1) && tabuleiro[x+1][y] == peca) {
42              count += marcar(sz, tabuleiro, x+1, y);
43          }
44
45          if(y > 0 && tabuleiro[x][y-1] == peca) {
46              count += marcar(sz, tabuleiro, x, y-1);
47          }
48
49          return count;
50
51      } else {
52          return -1;
53      }
54  }
```

A função int marcar(int sz, int tabuleiro[ ] [sz], int x, int y) recebe como parâmetros o tamanho do tabuleiro de jogo(“sz”, do tipo inteiro), todo o tabuleiro (“tabuleiro[ ] [sz]”, do tipo inteiro) e as coordenadas “x” e “y”, ambas inseridas

pelo jogador em cada jogada, ambas do tipo inteiro, em que “x” corresponde ao número da linha e “y” ao número da coluna.

Sendo esta função do tipo inteiro, ela vai retornar um valor inteiro, vai retornar o valor da variável count (que foi criada no início da função, variável esta que é do tipo inteiro portanto), ou seja, todos os quadrados que façam parte do grupo do quadrado (x,y), para isso, fez-se o seguinte:

O primeiro “if” da função vai verificar se o quadrado selecionado está dentro do tabuleiro, e se sim, a variável “peca” criada (do tipo inteiro) vai receber o valor do quadrado selecionado pelo jogador. De seguida, esse quadrado irá receber o valor -1 (fica vazio), para que na função “void mostrar” esse -1 seja transformado em “-“. Além disso, a variável count passa a ter o valor de 1.

Após isto, 4 if’s irão ser testados: o primeiro irá testar se existe uma cor igual em baixo; se sim, então o valor da variável count aumenta utilizando a própria função, ou seja, por recursividade (usando a própria função para verificar o quadrado abaixo do vizinho encontrado, aumentando o valor de count sempre que se encontra um quadrado vizinho da mesma cor do quadrado escolhido pelo jogador); todos os outros if’s seguem o mesmo raciocínio, sendo que o segundo if verifica se existe cor igual do lado direito, o terceiro verifica se existe cor igual no topo, e o quarto verifica se existe cor igual do lado esquerdo.

No final, a função retorna count, o número de quadrados com a mesma cor do quadrado selecionado e que são vizinhos deste, ou vizinhos dos vizinhos deste e por aí adiante.

Caso o quadrado selecionado já não contenha nenhuma cor, então a função retorna -1, uma vez que assim, a função “int pontuação” que retorna os pontos da jogada segundo o número de quadrados, retornará 0 pontos.

2.3. void gravidade(int sz, int tabuleiro[ ] [sz])

```
58  void gravidade(int sz, int tabuleiro[][sz]){
59      int i, j;
60      for(j = 0; j < sz; j++){
61        for(i = 0; i < (sz-1); i++){
62          if(tabuleiro[i][j] == -1 && tabuleiro[i+1][j] != -1) {
63              tabuleiro[i][j] = tabuleiro[i+1][j];
64              tabuleiro[i+1][j] = -1;
65              i = -1;
66              }
67          }
68      }
69  }
```

A função void gravidade(int sz, int tabuleiro[ ] [sz]) à semelhança de todas as outras funções do tipo void, não retorna nenhum valor.

Recebe como parâmetros o tamanho do tabuleiro, “sz” e todo o tabuleiro, “tabuleiro[ ] [sz]”, ambos do tipo inteiro.

São criadas duas variáveis, “i” e “j” (ambas do tipo inteiro) para percorrer todo o tabuleiro, à semelhança do que acontece na função “int mostrar”.

Assim, dentro do segundo ciclo for, temos um if. Os comandos inseridos dentro da condição if apenas serão executados se a peça do tabuleiro em questão for igual a -1 (ou seja, sem cor) e se a peça acima dessa for diferente de -1 (ou seja, tem cor). Faz-se esta verificação para evitar os casos em que as duas estão vazias e depois permanecemos sempre no mesmo estado (tendo sido esta uma das dificuldades na construção desta função).

Estas duas condições têm de ser verdadeiras para que (dentro do if) à peça vazia lhe seja atribuída a cor da peça acima. Consequentemente, à peça acima é atribuído o valor -1 para que esta fique vazia e lhe possa ser atribuída a cor da peça acima, e por aí adiante até chegar à penúltima posição da coluna (penúltima linha, daí a condição i<(sz-1) no segundo ciclo for, pois se a última posição estiver vazia(-1), não acontece nada).

Por fim, reinicializa-se a variável iteradora ”i” para começar de novo a percorrer a coluna desde o início.

2.4. void coluna(int sz, int tabuleiro[ ] [sz])

```
73  void coluna(int sz, int tabuleiro[][sz]){
74    int i, j;
75    for(j = 0; j < (sz-1); j++){
76      if(tabuleiro[0][j] == -1){
77        for(i = 0; i < sz; i++){
78          tabuleiro[i][j] = tabuleiro[i][j+1];
79          tabuleiro[i][j+1] = -1;
80        }
81      }
82    }
83  }
```

A função void coluna(int sz, int tabuleiro[ ] [sz]) à semelhança de todas as outras funções do tipo void, não retorna nenhum valor.

Recebe os mesmos parâmetros que a função “void gravidade”, o tamanho do tabuleiro, “sz” e todo o tabuleiro, “tabuleiro[ ] [sz]”, ambos do tipo inteiro.

São criadas duas variáveis “i” e “j”, ambas do tipo inteiro, que vão auxiliar a percorrer posições do tabuleiro, juntamente com os dois ciclos for.

Inicialmente, foi pensado verificar-se toda a coluna, se esta estivesse toda vazia, aí então se recorreria à regra da coluna vazia (quando toda a coluna está vazia, ela colapsa movendo os quadrados da direita para a esquerda para fechar a separação), mas posteriormente, verificou-se que havia uma maneira mais simples de executar a mesma função e que simplificaria o código. Assim, aqui verifica-se se a primeira posição da coluna é vazia, pois uma vez que existe a função gravidade, se a primeira posição da coluna está vazia então toda a coluna está vazia.

2.5. int pontuação(int num_quadrados)

```
86  int pontuacao(int num_quadrados){
87    return (num_quadrados*(num_quadrados + 1)) / 2;
88  }
```

A função int pontuação(int num_quadrados) é uma função do tipo inteiro, retornando assim um valor inteiro, e tem “num_quadrados” como parâmetro, do tipo inteiro também, que corresponde ao número de quadrados pelo qual o grupo do quadrado (x,y) inserido pelo jogador na jogada é composto.

Este valor (“num_quadrados”) é o valor retornado pela função “int marcar”.

Assim, esta função retorna a pontuação de cada jogada, que é calculada através da expressão:

$$\text{Pontos}=\frac{num\_quadrados * (num\_quadrados+1)}{2}$$

2.6. int jogada(int sz, int tabuleiro[ ] [sz], int x, int y)

```
91  int jogada(int sz, int tabuleiro[][sz], int x, int y){
92    if(tabuleiro[y][x] == -1){
93      printf("Jogada Invalida\n");
94      return 0;
95    }
96
97    int num_quadrados = marcar(sz, tabuleiro, y, x);
98    gravidade(sz, tabuleiro);
99    coluna(sz, tabuleiro);
100   return pontuacao(num_quadrados);
101 }
```

A função int jogada(int sz, int tabuleiro[ ] [sz], int x, int y) é uma função do tipo inteiro, tendo como parâmetros: o tamanho do tabuleiro(“sz”), todo o

tabuleiro(“tabuleiro[ ] [sz]”), a coordenada “x” e a coordenada “y” de cada jogada. Todos estes parâmetros são do tipo inteiro.

Esta função retorna um valor. Para isso, vai avaliar se a jogada inserida pelo jogador é inválida ou não; se for, então a mensagem “Jogada Invalida” surge no ecrã e o número de pontos desta jogada é, obviamente, 0. Caso a jogada não seja inválida, é criada a variável “num_quadrados” do tipo inteiro. A esta variável vai ser atribuído o valor retornado pela função “int marcar” (número de quadrados do grupo do quadrado inserido pelo jogador). Posteriormente, são chamadas as funções “void gravidade” e “void coluna” para que o tabuleiro seja atualizado segundo as regras pretendidas.

Para concluir, esta função invoca a função “int pontuacao” e retorna o número de pontos obtidos na jogada, o valor calculado nesta função.

2.7. int fimJogo(int sz, int tabuleiro[ ] [sz])

```
104  int fimJogo(int sz, int tabuleiro[][sz]){
105    int i, j;
106    for(i = 0; i < sz; i++){
107      for(j = 0; j < sz; j++){
108        if(tabuleiro[i][j] != -1){
109          return 0;
110        }
111      }
112    }
113    return 1;
114  }
```

A função int fimJogo(int sz, int tabuleiro[ ] [sz]) tem, à semelhança de muitas outras funções já aqui referenciadas, o tamanho do tabuleiro “sz” e todo o tabuleiro “tabuleiro[ ] [sz]” como parâmetros, ambos do tipo inteiro.

Assim, de forma muito sucinta, pois a estrutura da função é bastante idêntica a outras já analisadas anteriormente, são criadas duas variáveis do tipo

inteiro, “i” e “j” para, com o auxílio de dois ciclos for, todo o tabuleiro ser percorrido.

Todos os quadrados vão ser analisados. Caso exista algum que não está vazio e contém uma cor, a função retorna 0 (um inteiro). Caso todo o tabuleiro esteja vazio, ou seja, todos os quadrados estão vazios (têm o valor -1 atribuído), então a função retorna 1. Assim, quando esta função for invocada na main, será possível ao programa concluir quando é que o jogo terminou.

2.8. void gerarTabuleiro(int sz, int tabuleiro[ ] [sz])

```
117  void gerarTabuleiro(int sz, int tabuleiro[][sz]){
118    int i, j;
119    for(i = 0; i < sz; i++){
120      for(j = 0; j < sz; j++){
121        tabuleiro[i][j] = (rand() % 4) + 1;
122      }
123    }
124  }
```

A função void gerarTabuleiro(int sz, int tabuleiro tabuleiro[ ] [sz]) é uma função que não retorna nenhum valor, mas que é invocada na função main para que seja criado o tabuleiro inicial, com valores aleatórios entre 1 e 4.

Para isto, são criadas duas variáveis “i” e “j” do tipo inteiro, e utilizando dois ciclos for, todo o tabuleiro é percorrido e a todos os quadrados é atribuído um valor através da expressão “(rand() % 4) + 1” (nesta expressão soma-se o valor 1 uma vez que rand()%4 apenas escolhe valores entre 0 e 3), em que é utilizada a função rand() como auxiliar, da biblioteca <stdlib.h>.

3. Main do Modo Iterativo

3.1. Modo Iterativo

```c
int main(){



  int x, y, sz, pontuacao = 0;

  printf("============= COLOR SQUARES =============\n\n\n");
  printf("Indique o tamanho do tabuleiro(Max=20): ");
  scanf("%d", &sz);
  printf("\n");
  printf("\n");
  printf("A dimensao do tabuleiro sera %d x %d.\n\n", sz, sz);
  printf("Em cada jogada tera que escolher um quadrado do tabuleiro.\n");
  printf("Quanto maior for o numero de quadrados do grupo do quadrado selecionado, maior sera a pontuacao(em cada jogada).\n\n\n");

  int tabuleiro[sz][sz];

  gerarTabuleiro(sz, tabuleiro);

  while(fimJogo(sz, tabuleiro) != 1) {
    mostrar(sz, tabuleiro);
    printf("(0,0) Ponto Inferior Esquerdo\n\n");
    printf("Indique a cordenada X: ");
    scanf("%d", &x);
    printf("Indique a cordenada Y: ");
    scanf("%d", &y);
    pontuacao += jogada(sz, tabuleiro, x, y);
  }

  printf("\n--= FIM DE JOGO =--\n\n");
  printf("Obteve uma pontuacao de %d pontos\n\n", pontuacao);

    return 0;
}
```

A função int main() é a função que contém as mensagens principais que vão aparecer no ecrã para o jogador e é onde todas as outras funções vão ser invocadas para que o programa funcione consoante desejado.

São no início criadas quatro variáveis, todas elas do tipo inteiro, e são elas: “x”, “y”, “sz” e “pontuacao”; esta última é inicializada com o valor 0, pois corresponde à pontuação no início do jogo.

De seguida, surgem alguns “printf’s” que mostram mensagens que surgem no ecrã do jogador, durante o jogo.

É pedido ao jogador o tamanho do tabuleiro, o qual é registado através da operação “scanf(“%d”, &sz)”.

Aparecem mais algumas mensagens com algumas regras do jogo, e posteriormente é criado o tabuleiro com as dimensões inseridas pelo jogador (“int

tabuleiro[sz] [sz]") e através da chamada da função "gerarTabuleiro(sz, tabuleiro)", todas as posições do tabuleiro serão preenchidas por números entre 1 e 4 em que cada número corresponde a uma cor.

Segue-se um ciclo while. Este ciclo vai trabalhar enquanto o jogo não terminar, ou seja, enquanto o tabuleiro não estiver completamente vazio. Para isso, invoca-se a função "int fimJogo" e se esta retornar o valor 0 (ou seja, diferente de 1, tal como diz a condição dentro do ciclo while), então será permitido ao jogador executar mais uma jogada.

A cada jogada é invocada a função "void mostrar" que atualiza o tabuleiro a cada jogada efetuada. Aparecem ainda no ecrã do jogador as mensagens para que este insira a coordenada "x" e a coordenada "y", para identificar qual o quadrado que quer selecionar do tabuleiro.

Algo muito importante neste ciclo: A variável "pontuacao" é atualizada todas as jogadas, uma vez que a esta se soma sempre o valor retornado pela função "int jogada", ou seja, o valor de pontos daquela jogada. Esta função ("int jogada") é muito importante pois é nela que se concentram todas as funções que atualizam o tabuleiro segundo as regras do jogo.

Assim, todas estas operações de verificar se todo o tabuleiro está vazio ou não, de apresentar o tabuleiro atualizado, de pedir as coordenadas da jogada ao jogador, e aplicar as regras do jogo no tabuleiro e de registar os pontos da jogada, são aplicadas a cada jogada efetuada pelo jogador.

Quando o tabuleiro estiver totalmente vazio, ou seja, quando a função "int fimJogo" retornar o valor 1, o ciclo while deixa de funcionar e o Jogo termina. Aparece a mensagem "Fim de Jogo" no ecrã do utilizador e é apresentada a pontuação final.

3.2. Modo Automático

3.2.1. Variáveis:

- i e j - são variáveis iteradoras que vão ser utilizadas nos ciclos for;
- sz - irá conter o tamanho do tabuleiro lido a partir do ficheiro;
- pontuacao - irá guardar a pontuação do jogo;
- num_jogadas - irá conter o número de jogadas lidas a partir do ficheiro;
- cores - irá conter as linhas do tabuleiro lidas a partir do ficheiro;
- tabuleiro – array bidimensional que vai conter o tabuleiro de jogo;
- ch - variável que vai conter um caracter lido a partir do ficheiro quando estamos a ler as jogadas;
- index - variável iteradora;
- jogadas – array bidimensional que vai conter as coordenadas das jogadas lidas a partir do ficheiro

3.2.2.Programa

É aberto o ficheiro “jogadas.txt” no modo de leitura e guardado na variável apontador f do tipo FILE. Depois é verificado se o ficheiro foi aberto com sucesso. Caso contrário mostra mensagem de erro e o programa termina. Depois começa a ler as linhas a partir do ficheiro. Primeiro lê o tamanho do tabuleiro que corresponde à primeira linha do ficheiro. Utiliza-se a função fscanf com a string de formato %d que irá retornar um inteiro que por sua vez irá ser guardado na variável sz. Como já se tem o tamanho do tabuleiro pode-se declarar o tabuleiro como um array bidimensional onde as duas dimensões têm o mesmo tamanho (variável sz). Depois o tabuleiro é preenchido. Como já se sabe o tamanho do tabuleiro executa-se um cliclo for com sz iterações para ler, a partir do ficheiro, cada uma das linhas que contêm as cores de cada posição no tabuleiro. O ciclo exterior serve para ler as linhas a partir ficheiro e guardá-las como um número inteiro na variável cores. Este ciclo vai de sz-1 até 0 porque na implementação do programa a array é lida ao contrário, ou seja, o ponto 0,0 é no topo esquerdo. Depois o ciclo interior, que também vai iterar de sz-1 até 0 (do fim do array até ao início), serve para separar individualmente cada dígito do inteiro e guardá-lo no array do tabuleiro. Por exemplo: cores = 14111

Linha do array tabuleiro

14111 % 10 = 1

14111 / 10 = 1411

Outra iteração do ciclo interior

1411 % 10 = 1

1411 / 10 = 141

E assim por diante até o ciclo chegar ao 0, ou seja, percorre o tamanho do tabuleiro que está definido em sz. Depois é lido a partir do ficheiro o número de jogadas e guardado o valor na variável num_jogadas. Depois são lidas as jogadas a partir do ficheiro. Esta leitura vai ser feita caracter a caracter. É definido o valor 0 para a variável iteradora i e são definidas a variável ch do tipo char que vai guardar o carácter lido a partir do ficheiro, a variável index do tipo inteiro e inicializada com o valor 0 que vai servir como variável iteradora e é definido um array bidimensional com uma dimensão do tamanho do número de jogadas e a outra dimensão igual a 2, uma vez que cada jogada é constituída por dois valores. É inicializado um ciclo while que vai ser executado até ser atingido o fim do ficheiro. Dentro do ciclo while é lido um caracter a partir do ficheiro. Depois é verificado se o caracter é maior ou igual ao carácter zero (‘0’) e se a conversão de char para inteiro da variável ch é menor que sz. Aqui faz-se a comparação com sz porque as coordenadas das jogadas não podem ultrapassar o tamanho do tabuleiro. Na expressão de comparação ch >= ‘0’ não se faz a conversão do char para inteiro porque os outros caracteres (espaço, virgula, etc...) quando convertidos para inteiro têm valor 0 e aí quando encontrássemos um carácter desses ia ser considerado válido. Depois é guardada a primeira coordenada no array das jogadas e o valor da variável índice toma o valor de 1, ou seja, lê-se a primeira coordenada incrementando a variável index para depois se guardar a segunda coordenada da mesma jogada (ou mesma linha, valor de i). É também

verificado se o caracter lido é a vírgula e este passo é importante porque quando encontramos uma vírgula significa que já lemos uma jogada e vamos avançar para a próxima. E então a variável iteradora i é incrementada para passar para a próxima linha do array das jogadas e a variável index toma o valor 0, uma vez que o próximo caracter válido a ser lido vai ser a primeira coordenada da jogada [0]. Quando se diz caracter válido é porque no ciclo while só são considerados os números, que são as coordenadas das jogadas e a vírgula, que é o caracter que serve para avançar para a próxima jogada. Os outros caracteres são ignorados no ciclo. Depois o jogo é efetuado num ciclo for que vai percorrer todas as jogadas onde são passados à função jogada o tamanho, o tabuleiro, a primeira coordenada e a segunda coordenada.

4. Ficheiros

Os ficheiros usados no código são:

- colorSquares.h : com os protótipos das funções comuns a ambos os programas;
- colorSquares.c : com a implementação das referidas funções;
- scIter.c : com o jogo para o modo iterativo;
- csAuto.c : com o jogo para o modo automático;

O comando *gcc -c colorSquare.c* compila o ficheiro *colorSquare.c* e gera o ficheiro objeto *colorSquares.o*.

O comando *gcc -o csIter colorSquares.o csIter.o* liga os ficheiros objetos e cria o executável *csIter*.

Conclusão

Neste trabalho, desenvolvido em Linguagem C, aprofundou-se o conhecimento anteriormente adquirido sobre uma matriz bidimensional.

Inicialmente a ajuda de um papel e uma caneta foi extremamente essencial, pois sabe-se que não se escreve um código antes de se pensar acerca dele. Visto que é bastante mais fácil entender como se escreve o código de um programa quando se tem um esboço a partir do qual ele se desenvolve.

Um dos maiores desafios encontrados na realização deste trabalho foi como selecionar os quadrados que fariam parte do grupo selecionado pelo jogador. Inicialmente, a melhor solução para este problema, aparentou ser a utilização de ciclos for que percorressem a matriz e comparasse as peças vizinhas do quadrado selecionado, as vizinhas das vizinhas e por aí em diante. Após muitos erros e falhas na tentativa de arranjar um algoritmo que selecionasse os quadrados de uma mesma cor, foi averiguado que a melhor solução seria o uso da própria função através da recursividade.

Apesar de diversos erros, dúvidas e muito esforço, aprendeu-se como criar um jogo de “Color Squares” utilizando todas as ferramentas apreendidas ao longo do semestre pelo discente.