GUIA DE IMPLEMENTAÇÃO

1. Conceitos gerais

O funcionamento do nosso programa assenta muito na navegação em matrizes definidas na memória. A matriz principal é a matriz do tabuleiro, a matriz yx[17][26]. O próprio nome indica-nos de que modo podemos encontrar uma coordenada rapidamente na matriz completa, na primeira coordenada indicar o valor de y (linha) e na segunda coordenada o valor de x (coluna), como se de um referencial cartesiano se tratasse mas o par ordenado surge na ordem inversa (yx[y][x]). Ou seja, se queremos encontrar a coordenada A2 (com A=1, B=2…), teríamos que colocar yx[2][1]. Este conceito é fulcral e transversal no nosso programa.

2. Implementação dos modos de posicionamento e disparo

Quanto à implementação dos modos de posicionamento gostávamos de referir primeiro que em ambos os modos (p1 e p2) temos duas instruções do{}while(…), (uma para o valor y e outra para o valor x da coordenada) que apontam para os cantos superiores esquerdos das matrizes do tabuleiro sequencialmente. Isto permite-nos trabalhar matriz a matriz com o mesmo referencial – o canto superior esquerdo da matriz atual, indicando qualquer coordenada da matriz em função de y e x. Assim a coordenada do meio da matriz atual será sempre dada por yx[y-1][x+1] (por exemplo).

Em relação ao modo de posicionamento 1, apenas referir que a variável tentativa_v conta as tentativas de colocação de qualquer uma das 43 variantes, se este valor chegar a 3 é colocada a peça de Id. Global 5, como pedido (através do bloco if (tentativa<=3){qualquer variante}else{peça IG 5}). Já em relação ao modo de posicionamento 2, o seu funcionamento é análogo só que desta vez sorteamos primeiro um tipo de peça e depois dentro de cada peça sorteamos uma das suas variantes. Se essa variante não der para ser colocada, a flag e_v é ativa e todas as variantes dessa peça serão testadas (contando as variáveis que vão sendo testadas – tentativa_v++). Se nenhuma variante der, a flag erro_peca é ativa e regista-se na flag tentativa_p[1, 2, 3, 4, 5, 6, 7 ou 8] que essa peça (e respetivas variantes) já foi tentada colocar. Depois é sorteada outra peça escolhida pelo utilizador (excluindo aquelas peças que o programa não conseguiu colocar). O processo repetir-se-á por todas peças e se nenhuma peça puder ser colocada na matriz atual (e ainda houver peças para colocar) a flag erro_tabuleiro é ativa e a contagem i_tabuleiro é incrementada. Se esta contagem chegar a 1000 o programa toma as diligências apropriadas.

No que toca ao modo de disparo 1 gostaríamos apenas de relevar a matriz de flags d_yx que vai registando as coordenadas que já foram aleatorizadas para que não voltem a calhar. Quanto aos modos de disparo queremos salientar que juntámos o modo d_2 e d_3 dada a sua semelhança. Em relação a estes modos de disparo gostávamos de elaborar apenas um pouco mais dada a sua complexidade relativa. A junção dos dois modos em si não é complicada. Por meio de blocos if(…){}else{} garantimos que a void quando modo_dis = 2 se comporta de modo diferente de quando modo_dis = 3. Na realidade a complexidade reside na matriz sequencia[2][9] na qual, na primeira linha, gravámos os valores que serão adicionados a y e na segunda linha os valores que serão adicionados a x, por ordem. Assim, a 1ª coluna desta matriz (tendo SEMPRE como referência o canto superior esquerdo da matriz) selecionará a coordenada do meio da matriz, e a 2ª coluna selecionará a coordenada imediatamente acima da do meio (e assim sucessivamente seguindo a ordem destes modos de disparo que o enunciado refere). Assim, em cada matriz, ele disparará seguindo sempre a sequência do enunciado até que todas as peças que essa matriz contém sejam descobertas ou, se não tiver peça nenhuma, quando percorrer a matriz toda pela ordem.

Por fim, gostávamos apenas de referir o uso da matriz d_yx no modo de disparo 3. No instante imediatamente antes de passar à próxima matriz, o nosso programa analisa as peças que foram descobertas e em torno delas (em toda a volta) ativa a matriz de flags d_yx (com o valor ‘1’). Para este efeito servimo-nos da sequência[2][9] para que sejam colocados ‘1’ em torno de todas as peças que foram descobertas na matriz atual. Este processo acaba por colocar o valor ‘1’ DENTRO de algumas das coordenadas da matriz atual mas isso não interfere com o programa na medida em que este processo é realizado imediatamente antes do programa passar à próxima matriz, importando apenas os ‘1’ que transgridem os limites da matriz em que estávamos anteriormente.

Como dissemos, talvez este seja o pormenor mais subtil do nosso programa. Esperamos que o esclarecimento um pouco mais elaborado tenha sido útil. Queríamos reforçar que achamos fundamental a leitura das notas que acompanham o nosso programa para uma compreensão do nosso código mais efetiva.