Refinamentos sucessivos

Objetivos:

- Estudar a técnica de refinamentos sucessivos

Estrutura de dados atual

```
Program Jogo2048;
USES CRT;
CONST
    MAX = 4; DIR=1; ESQ=2; CIMA=3; BAIXO=4;
    PERCENT = 5; MAXVALORPECA = 4;
TYPE tabuleiro = array [0..MAX+1,0..MAX+1] of integer;
Var
    T: tabuleiro;
    Direcao, Pontos: integer;
```

Algoritmo atual

```
Begin
    IniciarJogo (T,Pontos);
    MostrarJogoAtual (T,Pontos);
    While ExisteMovimento (T) do
    Begin
        Direcao:= EscolheDirecao;
        MovimentarPecas (T,Direcao,Pontos);
        InserirNovaPeca (T);
        MostrarJogoAtual (T,Pontos);
    End;
End.
```

Detalhamento das subrotinas

- Falta a implementação do procedimento:

  MovimentarPecas (T,Direcao,Pontos)

Detalhamento das subrotinas

• Como movimentar as peças do jogo na direção desejada, segundo as regras?

• Para simplificar o problema, vamos estudar o caso de um deslocamento para a direita em um vetor

• Depois faremos as modificações necessárias para adaptar a solução para uma matriz e também para as outras direções

Abre parênteses...

- Vamos considerar o caso de um vetor de 1..MAX inteiros que pode ter elementos nulos em qualquer posição, inclusive em todas

- Como deslocar todos os valores para a direita realizando operações de fusão+soma de elementos conforme as regras do jogo?

Movimentar um vetor à direita

- Pelas regras do jogo, quando uma peça é movimentada para a direita, se ela se choca com outra peça de igual valor, as duas se tornam uma só, sendo que o valor desta é a soma dos valores das duas peças originais

- Porém, uma peça que é resultado de uma fusão+soma não pode ser fundida+somada outra vez

Movimentar um vetor à direita

- Podemos imaginar vários algoritmos, mais ou menos eficientes. Basicamente eles vão diferir na eficiência ou na complexidade do código ou ainda no uso ou não de estruturas auxiliares

Movimentar um vetor à direita

Primeira opção:

- Passar três vezes pelo vetor: (1) compactar todos à direita, eliminando os zeros; (2) somar elementos consecutivos que são iguais; (3) outra compactação

Movimentar um vetor à direita

Segunda opção:

- Usar vetor auxiliar para facilitar o processo de soma e eliminação dos zeros e depois copiar de volta

Movimentar um vetor à direita

Terceira opção:

- Tentar passar pelo vetor apenas uma única vez sem vetor auxiliar

Movimentar um vetor à direita

- Vamos implementar a primeira opção:

Passar três vezes pelo vetor: (1) compactar todos à direita, eliminando os zeros; (2) somar elementos consecutivos que são iguais; (3) outra compactação

Compactar um vetor à direita

- Primeiro subproblema:

Compactar um vetor de 1..MAX elementos inteiros movendo todos à direita, eliminando os zeros. O vetor pode estar cheio e não ter zeros. Também pode conter apenas zeros. Os elementos mais a esquerda que sobrarem devem ficar com valor nulo.

Compactar um vetor à direita

• Ideia: K é o maior índice do vetor que pode apontar para conteúdos nulos. À direita de K, todos os elementos do vetor são não nulos garantidamente

				K			
0 ou N	0 ou N	0 ou N	0 ou N	0 ou N	N1	N2	N3

Compactar um vetor à direita

• Observar que o conteúdo apontado por K pode ser nulo ou não, o importante é que à sua direita não há mais nulos

				K			
0 ou N	0 ou N	0 ou N	0 ou N	0 ou N	N1	N2	N3

Compactar um vetor à direita

- Então basta percorrer os índices de K até 1 procurando o primeiro elemento não nulo

				i			
				K			
0 ou N	0 ou N	0 ou N	0 ou N	0 ou N	N1	N2	N3

Compactar um vetor à direita

- Primeira situação: i = k e v[i] = v[k] é não nulo

				i			
				K			
0 ou N	0 ou N	0 ou N	0 ou N	N	N1	N2	N3

Compactar um vetor à direita

- Solução: K:= K - 1; i:= i - 1

				i			
				K			
0 ou N	0 ou N	0 ou N	0 ou N	N	N1	N2	N3

Compactar um vetor à direita

- Resultado da operação:

			i				
			K				
0 ou N	0 ou N	0 ou N	0 ou N	N	N1	N2	N3

Compactar um vetor à direita

- Segunda situação: i < k. Neste caso todos os elementos de i+1 até K (inclusive) são nulos

	i						
				K			
0 ou N	N	0	0	0	N1	N2	N3

Compactar um vetor à direita

- Solução: v[K]:= v[i]; v[i]:= 0;
  K:= k – 1; i:= i – 1;

i							
			K				
0 ou N	0	0	0	N	N1	N2	N3

Compactar um vetor à direita

```
Procedure CompactarDireita (Var V: vetor);
Var i,K: integer;
Begin
    (* código no próximo slide *)
End;
```

Compactar um vetor à direita

```
K:= MAX;
For i:= MAX downto 1 do
    If v[i] <> 0 then
    Begin
            If v[K] = 0 then
            Begin
                    v[K]:= v[i]; v[i]:= 0;
            End;
            K:= K - 1
    End;
```

Aglomerar um vetor à direita

- Segundo subproblema:

Fazer o processo de fusão+soma de um vetor de 1..MAX elementos inteiros que está garantidamente compactado à direita, isto é, podem haver elementos nulos no início, mas a partir do primeiro não nulo, todos os seguintes são não nulos até MAX. O vetor pode estar cheio.

Aglomerar um vetor à direita

• Considerando que o vetor já está compactado à direita, este vetor tem a seguinte propriedade: existe um índice K tal que a partir dele não existem elementos nulos à direita

			K				
0	0	0	N1	N2	N3	N4	N5

Aglomerar um vetor à direita

• Estamos procurando por índices consecutivos de igual conteúdo, desde o índice máximo até K, que por sinal pode ser 1

			K		i		MAX
0	0	0	N1	N1	N	N2	N2

Aglomerar um vetor à direita

- Quando encontrados, basta somar os conteúdos em um deles e tornar o outro nulo

			K			i	
0	0	0	N1	N1	N	N2	N2

Aglomerar um vetor à direita

- Por exemplo, v[i] = v[i+1]

			K			i	
0	0	0	N1	N1	N	N2	N2

Aglomerar um vetor à direita

- Logo: v[i+1]:= 2*v[i+1]; v[i]:= 0; i:= i – 2;

			K	i			
0	0	0	N1	N1	N	0	2*N2

Aglomerar um vetor à direita

- Observar que se N = N2 não há problema de soma indevida, pois i andou duas posições

			K	i			
0	0	0	N1	N1	N2	0	2*N2

Aglomerar um vetor à direita

- Quando v[i] <> v[i+1] basta i:= i - 1

			K		i		
0	0	0	N1	N1	N	N2	N3

Aglomerar um vetor à direita

- Quando v[i] <> v[i+1] basta i:= i - 1

			K	i			
0	0	0	N1	N1	N	N2	N3

Aglomerar um vetor à direita

- Assim, dependendo do caso, em uma iteração ocorre i:= i – 1 ou i:= i – 2.

- Logo, a implementação deve ser feita com um W*hile*

- Se alguns zeros forem somados não há problema

Aglomerar um vetor à direita

```
Procedure AglomerarDireita (Var V: vetor);
Var i: integer;
Begin
    (* código no próximo slide *)
End;
```

Aglomerar um vetor à direita

```
i:= MAX - 1;
While (i >= 1) AND (v[i] <> 0) do
Begin
    If v[i] = v[i+1] then
    Begin
        v[i+1]:= 2*v[i];   v[i]:= 0;   i:= i – 2;
    End
    Else i:= i – 1;
End;
```

Fecha parênteses...

- Agora podemos voltar à implementação do jogo 2048 adaptando a solução do problema anterior para o caso de matrizes e também para que os deslocamentos ocorram nas quatro direções permitidas

Detalhamento das subrotinas

- Para implementar:

  MovimentarPecas (T,Direcao,Pontos)

- Vamos usar o algoritmo seguinte

Detalhamento das subrotinas

```
Procedure MovimentarPecas (var T: tabuleiro; Direcao: integer; var Pontos: integer);
Var Moveu: boolean;
Begin
    Moveu:= false;
    Repeat
        Case Direcao of
            DIR: MovimentarDireita (T,Pontos,Moveu);
            ESQ: MovimentarEsquerda (T,Pontos,Moveu);
            CIMA: MovimentarCima (T,Pontos,Moveu);
            BAIXO: MovimentarBaixo (T,Pontos,Moveu);
        End;
    Until Moveu;
End;
```

Detalhamento das subrotinas

Temos que implementar quatro procedimentos:

- MovimentarDireita
- MovimentarEsquerda
- MovimentarCima
- MovimentarBaixo

Vejamos o caso da direita, os outros são parecidos

Detalhamento das subrotinas

```
Procedure MovimentarDireita (var T: tabuleiro; var Pontos: integer; var Moveu: boolean);
Begin
        CompactarDireita (T,Moveu);
        AglomerarDireita (T,Pontos,Moveu);
        CompactarDireita (T,Moveu);
End;
```

Detalhamento das subrotinas

- Como compactar a matriz à direita?

- Basta adaptar os algoritmos estudados para o caso de compactar vetores à direita

Detalhamento das subrotinas

- Sabemos que se um algoritmo funciona para um vetor, então ele deve funcionar para uma linha (ou coluna) específica de uma matriz

Detalhamento das subrotinas

Adaptação ao caso de uma linha da matriz:

- Seja M um valor fixo
- Trocar as ocorrências de v[x] por T[M,x]

Detalhamento das subrotinas

Adaptação ao caso de uma linha da matriz:

- Em seguida basta colocar o código em um laço:

For M:= 1 to MAX Do

Detalhamento das subrotinas

- A pontuação é atualizada sempre que houver uma soma, o que ocorre na procedure Aglomerar

- As duas procedures de compactar e aglomerar podem provocar movimentos e isto tem que ser reportado

Detalhamento das subrotinas

```
Procedure CompactarDireita (Var T: tabuleiro; var Moveu: boolean);

Var i,K,M: integer;

Begin

    For M:= 1 to MAX do

    Begin

        (* código no próximo slide *)

    End;

End;
```

Detalhamento das subrotinas

```
K:= MAX;
For i:= MAX downto 1 do
        If T[M,i] <> 0 then
        Begin
                If T[M,K] = 0 then
                Begin
                    T[M,K]:= T[M,i]; T[M,i]:= 0; Moveu:= true;
                End;
                K:= K – 1;
        End;
```

Detalhamento das subrotinas

```
Procedure AglomerarDireita (Var T: tabuleiro; Var Pontos: integer; var Moveu: boolean);

Var i,M: integer;

Begin
    For M:= 1 to MAX do
    Begin
            (* código no próximo slide *)
    End;
End;
```

Detalhamento das subrotinas

```
i:= MAX - 1;
While (i >= 1) AND (T[M,i] <> 0) do
Begin
        If T[M,i] = T[M,i+1] then
        Begin
                T[M,i+1]:= 2*T[M,i+1]; T[M,i]:= 0; i:= i – 2;
                Pontos:= Pontos + T[M,i+1]; Moveu:= true;
        End
        Else i:= i – 1;
End;
```

Detalhamento das subrotinas

Como movimentar para baixo?

- A diferença de movimentar para a direita e movimentar para baixo é simples:

basta inverter linhas e colunas!

Detalhamento das subrotinas

Como movimentar para baixo?

- Trocar as ocorrência de T[M,X] por T[X,M]

Detalhamento das subrotinas

Como movimentar para a esquerda?

- Para movimentar para a direita, o algoritmo percorreu o vetor do final para o início

- Logo, para movimentar para a esquerda, basta inverter a direção, isto é, percorrer o vetor do início para o final

Detalhamento das subrotinas

```
Procedure CompactarEsquerda (Var T: tabuleiro; var Moveu: boolean);

Var i,K,M: integer;

Begin

    For M:= 1 to MAX do

    Begin

            (* código no próximo slide *)

    End;

End;
```

Detalhamento das subrotinas

```
K:= 1;                                  // antes era K:= MAX;
For i:= 1 to MAX do                     // antes era For i:= MAX downto 1
        If T[M,i] <> 0 then
        Begin
                If T[M,K] = 0 then
                Begin
                        T[M,K]:= T[M,i]; T[M,i]:= 0; Moveu:= true;
                End;
                K:= K + 1;              // antes era K:= K – 1;
        End;
```

Detalhamento das subrotinas

```
Procedure AglomerarEsquerda (Var T: tabuleiro; Var Pontos: integer; var Moveu: boolean);

Var i,M: integer;

Begin
    For M:= 1 to MAX do
    Begin
            (* código no próximo slide *)
    End;
End;
```

Detalhamento das subrotinas

```
i:= 2;
While I <= MAX do
Begin
        If T[M,i] = T[M,i-1] then
        Begin
                T[M,i-1]:= 2*T[M,i]; T[M,i]:= 0; i:= i + 2;
                Pontos:= Pontos + T[M,i-1]; Moveu:= true;
        End
        Else i:= i + 1;
End;
```

Detalhamento das subrotinas

Como movimentar para cima?

- Para movimentar para cima, basta inverter linhas e colunas no algoritmo de movimentar para a esquerda!

Detalhamento das subrotinas

- Trocar as ocorrências de T[M,X] por T[X,M]

Detalhamento das subrotinas

- Em resumo, temos quatro procedimentos diferentes, mas parecidos, para se movimentar elementos nas quatro direções

Finalmente...

- Terminamos a implementação do jogo 2048!
- Vejamos os jogo funcionando...

Considerações

- Do ponto de vista de um jogo em um tabuleiro 4x4 nossa implementação é suficiente

- Mas pensando em um tabuleiro maior podemos fazer muitas otimizações

Considerações

- Algumas destas otimizações estão sugeridas nos slides anteriores desta aula e nos slides da aula anterior

- No restante da aula, e na próxima, vamos trabalhar em algumas delas, as outras ficam como exercício

Problema 1

- Melhorar as procedures de deslocamento
- Implementar uma única procedure para movimentos na horizontal (direita e esquerda)

Problema 1

- Vejamos novamente o algoritmo para movimentar à esquerda

Problema 1

```
Procedure CompactarEsquerda (Var T: tabuleiro; var Moveu: boolean);

Var i,K,M: integer;

Begin

    For M:= 1 to MAX do

    Begin

            (* código no próximo slide *)

    End;

End;
```

Problema 1

```
K:= 1;                                  // antes era K:= MAX;
For i:= 1 to MAX do                     // antes era For i:= MAX downto 1
        If T[M,i] <> 0 then
        Begin
                If T[M,K] = 0 then
                Begin
                        T[M,K]:= T[M,i]; T[M,i]:= 0; Moveu:= true;
                End;
                K:= K + 1;              // antes era K:= K – 1;
        End;
```

Problema 1

- Podemos usar um parâmetro booleano indicando se o movimento é para a direita ou esquerda

- Este parâmetro permite definir os limites dos laços

Problema 1

```
Procedure CompactarHorizontal (Var T: tabuleiro; var Moveu: boolean; direita: boolean);

Var i,K,M,sinal: integer;

Begin
    For M:= 1 to MAX do
    Begin
            (* código no próximo slide *)
    End;
End;
```

Problema 1

```
(* definindo os limites de início do laço e soma/subtração *)
    If direita then
    Begin
        K:= MAX; i:= MAX; sinal:= 1;
    End
    Else
    Begin
        K:= 1; i:= 1; sinal:= -1
    End;
```

Problema 1

```
While (direita AND (i >= 1)) OR (not direita AND (i <= MAX))  Do
Begin
    If T[M,i] <> 0 then
    Begin
                If T[M,K] = 0 then
                Begin
                        T[M,K]:= T[M,i]; T[M,i]:= 0; Moveu:= true;
                End;
                K:= K – 1*sinal;
    End;
    i:= i – 1*sinal;
End;
```

Problema 1

```
Procedure AglomerarHorizontal
(Var T: tabuleiro; Var Pontos: integer; direita: boolean);
Var i,M,sinal: integer;
Begin
        For M:= 1 to MAX do
        Begin
                (*  código no próximo slide *)
        End;
End;
```

Problema 1

```
If direita then
Begin
        i:= MAX – 1;
        sinal:= 1;
End
Else
Begin
        i:= 2;
        sinal:= -1;
End;
```

Problema 1

```
While (direita AND (I >= 1)) OR (not direita AND (i <= MAX)) do
Begin
    If T[M,i] = T[M,i+1*sinal] then
     Begin
          T[M,i+1*sinal]:= 2*T[M,i+1*sinal]; T[M,i]:= 0;
          Pontos:= Pontos + T[M,i+1*sinal];
           i:= i - 2*sinal;
     End
     Else i:= i - 1*sinal;
End;
```

Problema 2

- De maneira análoga, pode-se unificar os dois procedimentos que fazem movimentos na vertical

- Fica como exercício

Detalhamento das subrotinas

```
Procedure MovimentarPecas (var T: tabuleiro; Direcao: integer; var Pontos: integer);
Var Moveu: boolean;
Begin
    Moveu:= false;
    Repeat
        Case Direcao of
            DIR: MovimentarHorizontal (T,Pontos,Moveu,true);
            ESQ: MovimentarHorizontal (T,Pontos,Moveu,false);
            CIMA: MovimentarVertical (T,Pontos,Moveu,false);
            BAIXO: MovimentarVertical (T,Pontos,Moveu,true);
        End;
    Until Moveu;
End;
```

Detalhamento das subrotinas

```
Procedure MovimentarHorizontal (var T: tabuleiro; var Pontos: integer; var Moveu: boolean; direita: boolean);
Begin
        CompactarHorizontal (T,Moveu,direita);
        AglomerarHorizontal (T,Pontos,direita);
        CompactarHorizontal (T,Moveu,direita);
End;
```

Problema 3

- Na verdade, é possível fazer um único procedimentos que faz movimentos em direção, bastando passar dois parâmetros booleanos

- Um para indicar movimentos esquerda/direita e outro para movimentos na horizontal/vertical

- Fica também como exercício