Matrizes Esparsas - Funções
Disciplina de Algoritmos e Estruturas de Dados I
Curso de Engenharia de Computação
Professor: Silvio Luiz Bragatto Boss

1) Tipo de Dado Abstrato para Matrizes Esparsas

```
1
2  typedef struct NoTag{
3     int linha, coluna;
4     int valor;
5     struct NoTag *direita, *abaixo;
6  }no;
7
8  typedef struct MatrizE{
9    int m,n; //dimensao da matriz
10   struct NoTag *inicio, *fimLinha, *fimColuna;
11 }MatrizE;
```

2) Função *Main*

```
1
2  void main(void)
3  {
4
5    MatrizE M;
6
7    LeMatriz(&M);
8
9  }
```

3) Função que efetua a leitura dos elementos da Matriz Esparsa

```
1
2  void LeMatriz(MatrizE *pmat)
3  {
4
5     int quantLinha, quantColuna; //dimensao da linha e coluna
6     int opcao,i,j,valor;
7
8     printf("Insira a quantidade de linhas da Matriz ");
9     scanf("%d",&quantLinha);
10
11    printf("Insira a quantidade de colunas da Matriz ");
12    scanf("%d",&quantColuna);
13
14    CriaMatriz(pmat,quantLinha,quantColuna);
15
16   do
17    {
18     printf("\n1 - Insere elemento na Matriz ");
19     printf("\n2 - Mostra Matriz ");
20     printf("\n3 - Encerrar Programa ");
21     printf("\nInsira sua opcao ");
22     scanf("%d",&opcao);
23
24     switch(opcao){
25        case 1:
26              ObtemDados(&i,&j,&valor);
27              Insere(pmat,i,j,valor);
28              break;
29        case 2:
30              Mostra(pmat);
31              break;
32        case 3:
```

```
33              break;
34        default:
35              printf ("\n Digito nao valido");
36              break;
37      }
38    }while (opcao!= 3);
39 }
```

4) Função que Obtém os dados para ser Inseridos na Matriz Esparsa

```
1
2 void ObtemDados(int *i, int *j, int *valor)
3 {
4
5    int linha,coluna,item;
6
7    printf("Digite a linha para inserir o elemento na matriz ");
8    scanf("%d",&linha);
9
10   printf("Digite a coluna para inserir o elemento na matriz ");
11   scanf("%d",&coluna);
12
13   printf("Digite o valor para ser inserido na matriz ");
14   scanf("%d",&item);
15
16   *i = linha;
17   *j = coluna;
18   *valor = item;
19 }
```

5) Função que Cria o "Esqueto" da Matriz Esparsa

```
1
2 void CriaMatriz(MatrizE *Maux, int quantLinha, int quantColuna)
3 {
4    int cont;
5
6    CriaNoPrincipal(Maux);
7
8    Maux->m = quantLinha;
9    Maux->n = quantColuna;
10
11  //Cria o esqueleto da Matriz
12
13  for(cont=1; cont<=quantLinha; cont++)
14    InicializaLinha(Maux);
15
16  for(cont=1; cont<=quantColuna; cont++)
17    InicializaColuna(Maux);
18
19 }
```

6) Função que Cria o Nó Principal da Matriz Esparsa

```
1
2 void CriaNoPrincipal ( MatrizE *pmat )
3 {
4
5  no *temp; //*temp e o espaco destinado para o no
6
7  temp = (no *) malloc (sizeof (no ) ) ;
8
9  temp->linha = -1;
10 temp->coluna = -1;
```

```
11
12  temp->direita = temp; //Fecha o ciclo
13  temp->abaixo = temp;
14
15  pmat->inicio = temp;
16  pmat->fimLinha = temp;
17  pmat->fimColuna = pmat->fimLinha;
18 }
```

7) Função que Inicializa as Linhas do "Esqueleto" da Matriz Esparsa

```
1
2 void InicializaLinha ( MatrizE *pmat )
3 {
4   no *temp;
5
6   temp = (no *) malloc ( sizeof ( no  ) );
7
8   temp->linha = -1;
9   temp->coluna = 0;
10
11  pmat->fimLinha->abaixo = temp;
12  pmat->fimLinha = temp;
13
14  temp->abaixo = pmat->inicio ; //Fecha o circulo
15  temp->direita = temp;
16 }
```

8) Função que Inicializa as Colunas do "Esqueleto" da Matriz Esparsa

```
1
2 void InicializaColuna ( MatrizE *pmat )
3 {
4   no *temp;
5
6   temp = ( no *) malloc ( sizeof ( no ) );
7
8   temp->linha = 0;
9   temp->coluna = -1;
10
11  pmat->fimColuna->direita = temp;
12  pmat->fimColuna = temp;
13
14  temp->direita = pmat->inicio; //Fecha o circulo
15  temp->abaixo = temp;
16 }
```

9) Função que Insere um Elemento na Matriz Esparsa (Passado como parâmetro a linha e coluna para sua inserção).

```
1
2 void Insere ( MatrizE *pmat, int i, int j, int elem)
3 {
4
5   no *auxL; //auxilia a percorrer as linhas
6   no *auxC;  //auxilia a percorrer as colunas
7   no *temp;
8
9   int cont;
10
11  //Cria um novo no
12
13  temp = ( no *) malloc ( sizeof ( no ) );
14
15  temp->linha = i;
```

```
 temp->coluna = j;
 temp->valor = elem;

  //Percorre as linhas com um auxiliar

 auxL = pmat->inicio->abaixo;

 // i - 1 pois ja deslocou uma vez
 // percorre ate encontrar a linha desejada

 for ( cont=1; cont <= i-1; cont++)
     auxL = auxL->abaixo;

 for ( cont = 1; cont <= j; cont++)
 {
   if ( temp->coluna < auxL->direita->direita->coluna )
   {
      temp->direita = auxL->direita->direita;
      auxL->direita->direita = temp;
      break;
 }
 else if ( auxL->direita->linha == -1)
 {
    temp->direita = auxL->direita;
    auxL->direita = temp;
    break;
 }
 else
    auxL = auxL->direita;
 }

  //Percorre as linhas com um auxiliar

 auxC = pmat->inicio->direita;

 for ( cont = 1; cont <= j-1; cont++)
    auxC = auxC->direita;

 for ( cont = 1; cont <= i; cont++)
 {
   if ( temp->linha < auxC->abaixo->abaixo->linha )
   {
      temp->abaixo = auxC->abaixo->abaixo;
      auxC->abaixo->abaixo = temp;
      break;
   }
   else if (auxC->abaixo->coluna == -1)
   {
      temp->abaixo = auxC->abaixo;
      auxC->abaixo = temp;
      break;
   }
   else
      auxC = auxC->abaixo;
 }
}
```

10) Função que Exibe os elementos da Matriz Esparsa.

```

void Mostra( MatrizE *pmat)
{

  int i,j;
  no *temp;

  temp = pmat->inicio->abaixo;
```

```
9
10    for(i=1; i<=pmat->m; i++)
11    {
12     for(j=1; j<=pmat->n; j++)
13     {
14       if(i == temp->direita->linha && j == temp->direita->coluna)
15       {
16          printf("%d ", temp->direita->valor);
17          temp = temp->direita;
18       }
19       else
20      printf("%d ", 0);
21
22     }
23     printf("\n");
24     temp = temp->direita->abaixo;
25    }
26 }
```