UFRJ

INSTITUTO DE COMPUTAÇÃO
UFRJ

Programação de Computadores II

Ponteiros
– Alocação dinâmica - Matrizes e estruturas

Profa. Giseli Rabello Lopes

Sumário

- Alocação dinâmica de memória
  - Matrizes
  - Estruturas
- Exemplos

Adaptado de material preparado por Profa. Silvana Rossetto e Prof. Adriano Joaquim de Oliveira Cruz

Exemplo: matrizes com vetores bidimensionais

```
#include <stdio.h>
#define LIN 3
#define COL 2
int main ()
{
        int matriz [LIN][COL], i, j;
        for (i=0; i<LIN; i++)
                for (j=0; j<COL; j++)
                {
                        printf("Matriz[%d][%d]: ",i,j);
                        scanf ("%d", &matriz[i][j]);
                }
        for (i=0; i<LIN; i++)
        {
                for (j=0; j<COL; j++)
                        printf ("[%d] ", matriz[i][j]);
                printf("\n");
        }
        return 0;
}
```

Matrizes e Ponteiros

- Um ponteiro aponta para uma área de memória que é endereçada de maneira linear
- Vetores podem ser facilmente manipulados com ponteiros
- Em estruturas de dados com maior dimensionalidade, como matrizes, por exemplo, que são arranjos bidimensionais de dados, é necessário mapear o espaço bidimensional (ou de maior ordem) para uma dimensão
- No caso das matrizes é necessário mapear o endereço de cada elemento na matriz, que é definido por um par *(linha, coluna)* em um endereço linear

Matrizes e Ponteiros

- Considere uma matriz chamada matriz de tamanho **`LIN,COL`**
- Podemos usar uma solução que mapeie a matriz que é um objeto de duas dimensões em um vetor que tem apenas uma
- Neste caso o programa deve fazer a translação de endereços toda vez que precisar ler ou escrever na matriz
- A expressão **`matriz+(i*COL+j)`** calcula a posição do elemento **`matriz[i][j]`** a partir do primeiro elemento da matriz que está no endereço inicial **`matriz`**
- **Solução não muito boa**

Exemplo: matrizes com ponteiros

```c
#include <stdio.h>
#include <stdlib.h>
#define LIN 3
#define COL 2
int main ()
{
        int * matriz1 = (int *) malloc (LIN * COL * sizeof (int)), i, j;
        if (!matriz1)
        {
                printf ("Erro de alocacao de memoria\n");
                return -1;
        }
        for (i=0; i<LIN; i++)
                for (j=0; j<COL; j++)
                {
                        printf("Matriz[%d][%d]: ",i,j);
                        scanf ("%d", matriz1 + ((i * COL) + j));
                }
        for (i=0; i<LIN; i++)
        {
                for (j=0; j<COL; j++)
                        printf ("[%d] ", *(matriz1 + ((i * COL) + j)));
                        printf("\n");
        }
        free(matriz1);
        return 0;
}
```

Matrizes com Vetores de Ponteiros

- Uma possibilidade mais interessante é utilizar vetores de ponteiros
- Esta não é a notação ideal, mas é um passo na direção da notação mais efetiva
- Neste caso cada linha da matriz é apontada por um ponteiro armazenado no vetor de ponteiros

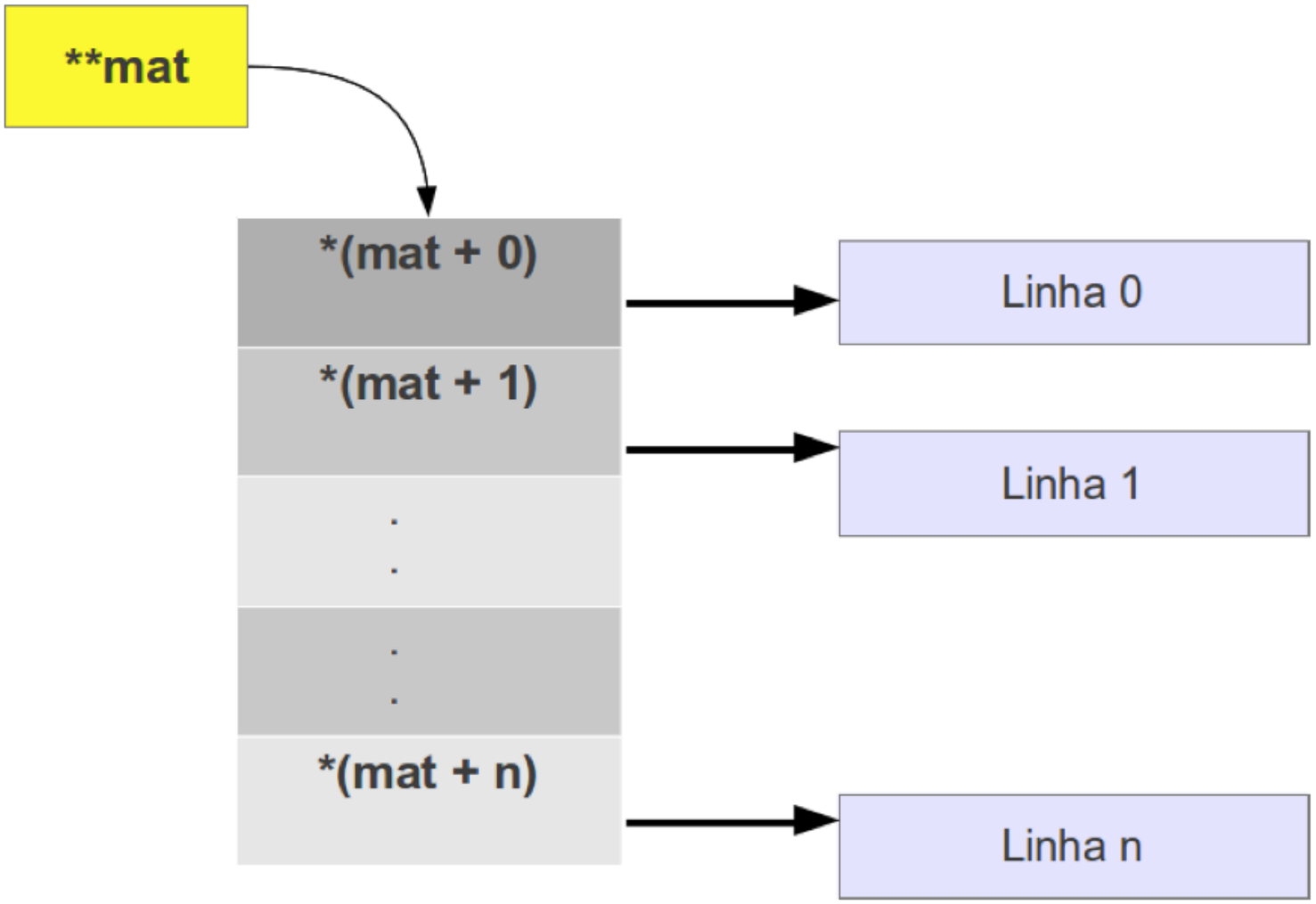
Exemplo: matrizes com vetores de ponteiros

```
int *matriz2[LIN], i, j;
for (i=0; i<LIN; i++)
{
        matriz2[i] = (int *) malloc (COL * sizeof (int));
        if (!matriz2[i])
        {
                printf ("Erro de alocacao de memoria\n");
                return -1;
        }
}
for (i=0; i<LIN; i++)
        for (j=0; j<COL; j++)
        {
                printf ("Matriz[%d][%d]: ",i,j);
                scanf ("%d", matriz2[i] + j);
        }
for (i=0; i<LIN; i++)
{
        for (j=0; j<COL; j++)
                printf ("[%d] ", *(matriz2[i] + j));
        printf("\n");
}
```

Matrizes com Ponteiros para Ponteiros

- No exemplo anterior, o número de linhas da matriz é fixa
- Há uma mistura de notação de ponteiros com matrizes
- Vamos considerar um exemplo onde tanto o número de linhas como o de colunas é desconhecido
- Agora vamos criar um vetor de ponteiros que irá armazenar o endereço inicial de cada linha

Ponteiros para ponteiros



Exemplo: matrizes de ponteiros para ponteiros

```
int i, j, L, C;
int **matriz3;
puts("Informe o numero de linhas e colunas");
scanf("%d %d", &L, &C);
matriz3 = (int **) malloc(L * sizeof(int *));
if(!matriz3)
{
        printf ("Erro de alocacao de memoria\n");
        return -1;
}
for (i=0; i<L; i++)
{
        *(matriz3 + i) = (int *) malloc (C * sizeof (int));
        if (! (*(matriz3 + i)))
        {
                printf ("Erro de alocacao de memoria\n");
                return -1;
        }
}
...
```

Exemplo: matrizes de ponteiros para ponteiros

```
...
for (i=0; i<L; i++)
{
        for (j=0; j<C; j++)
        {
                        printf ("Matriz[%d][%d]: ",i,j);
scanf ("%d", *(matriz3 + i) + j);
        }
}
for (i=0; i<L; i++)
{
        for (j=0; j<C; j++)
                        printf ("[%d] ", *(*(matriz3 + i) + j));
        printf("\n");
}
```

Alocando espaço para estruturas

- Para alocar espaço para estruturas apontadas por ponteiros é necessário usar o operador unário **`sizeof`**
- O tamanho de uma estrutura é sempre igual ou maior que a soma dos tamanhos dos seu componentes
- Considera como os dados são armazenados na memória dos computadores

Alocando espaço para estruturas

```
#include <stdio.h>
typedef struct _ALUNO
{
    char nome[40];
    float n1, n2, media;
} ALUNO;
int main (void)
{
    ALUNO *maria;
    maria = (ALUNO *) malloc (sizeof(ALUNO));
    if (!maria) exit(1);
    gets(maria->nome);
    scanf("%f %f", &(maria->n1), &(maria->n2));
    maria->media = (maria->n1 + maria->n2)/2;
    printf("A media de %s vale %0.2f\n", maria->nome,
        maria->media);
    return 0;
}
```

Alocando espaço para vetores de estruturas

```
#include <stdio.h>
#include <stdlib.h>
typedef struct _func
{
  char nome[40];
  float salario;
} Tfunc ;
void le (Tfunc *cadastro, int funcionarios)
{
  int i;
  char linha[40];
  for (i=0; i<funcionarios; i++)
  {
    puts("Nome ?");
    fgets((cadastro+i)->nome, 39, stdin);
    puts ("Salario ?"); fgets(linha, 39, stdin);
    sscanf(linha, "%f", &((cadastro+i)->salario));
  }
```

Alocando espaço para vetores de estruturas (cont.)

```
float media(Tfunc *cadastro, int funcionarios)
{
  float media=0.0;
  int i;
  for (i=0; i<funcionarios; i++)
  {
    media += (cadastro+i)->salario;
  }
  return media /= funcionarios;
  }
```

Alocando espaço para vetores de estruturas (cont.)

```
int main(void)
{
  Tfunc *cadastro;
  int funcionarios;
  char linha[40];
  puts("Quantos funcionarios ?");
  fgets(linha, 39, stdin);
  sscanf (linha, "%d" , &funcionarios);
  cadastro = (Tfunc *)malloc(funcionarios * sizeof (Tfunc));
  if (!cadastro) exit (1) ;
  le(cadastro, funcionarios);
  printf("Salario medio = %.2f\n",
    media(cadastro, funcionarios) );
  return 0;
}
```