Algoritmos II

ALOCAÇÃO DINÂMICA
DE MATRIZES

Conceito

- Também conhecidos como vetores dinâmicos multidimensionais.
- Uma matriz dinâmica é um vetor de vetores.
- Primeiro se cria um vetor dinâmico unidimensional de ponteiros, e após um vetor dinâmico para cada elemento do vetor.
- Existem duas formas de alocar matrizes dinamicamente:

Exemplo

Declara um ponteiro para ponteiro e aloca uma área de tamanho LIN para dados do tipo ponteiro para inteiro.

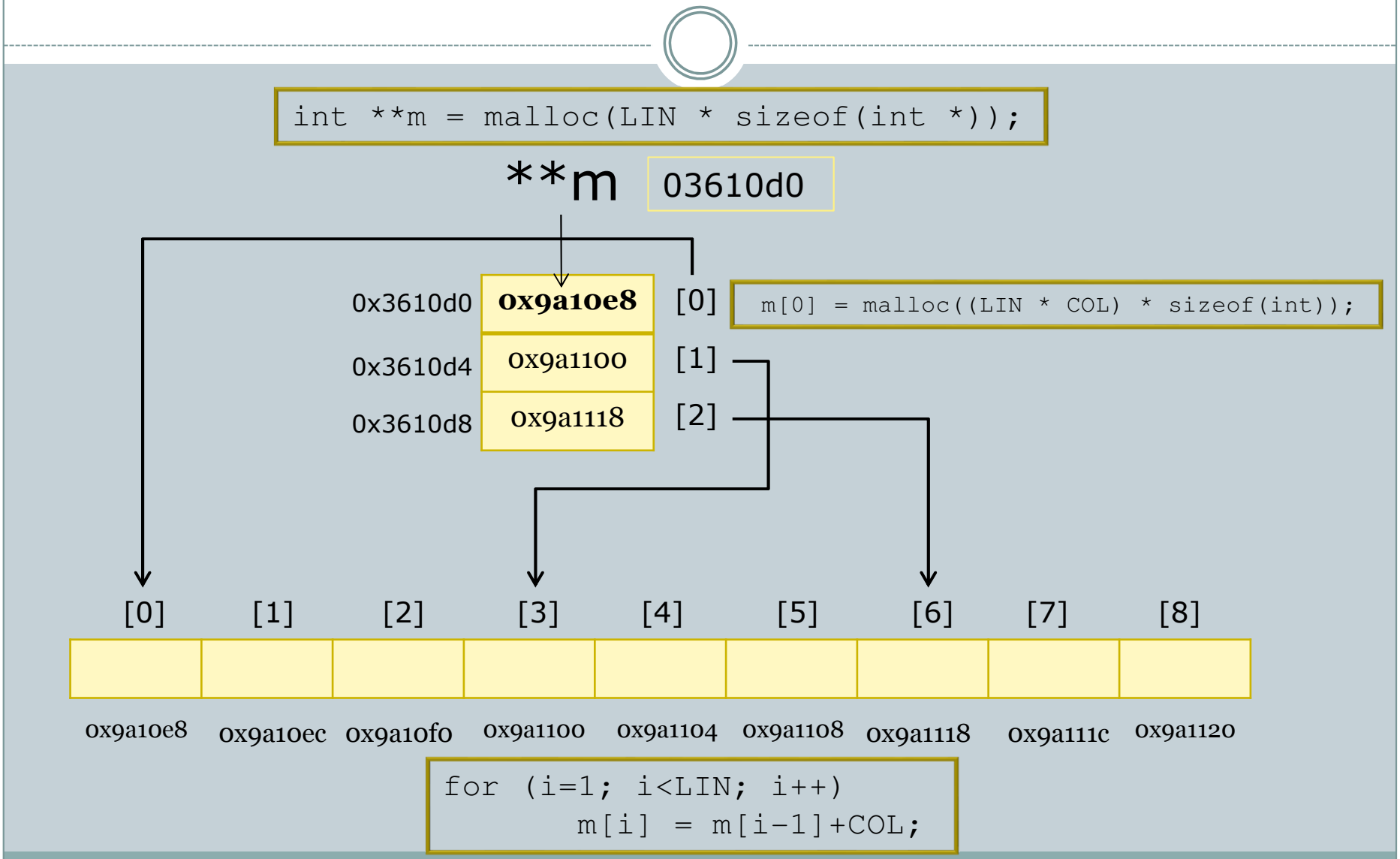
```
#include <stdio.h>
#include <stdlib.h>

int main() {
 int LIN = 3, COL = 3, i, j;

 int **m = malloc(LIN * sizeof(int *));

 m[0] = malloc((LIN * COL) * sizeof(int));

 for (i=1; i<LIN; i++)
   m[i] = m[i-1]+COL;

 for(i=0; i<LIN; i++)
   for(j=0; j<COL; j++)
     m[i][j] = i+j;

 for(i=0; i<LIN; i++){
   for(j=0; j<COL; j++){
     printf("%d",m[i][j]);
   }
   printf("\n");
 }
 free(m[0]);
 free(m);
 return 0;
}
```

Aloca uma área de tamanho LIN*COL para dados do tipo inteiro e armazena seu endereço na primeira posição do vetor m.

Rotina para fazer cada elemento do vetor apontar para a posição correspondente ao inicio de cada linha da matriz, pulando a primeira posição, pois esta já foi inicializada.

Na primeira linha libera a área alocada para a matriz, e na segunda a área alocada para o vetor de ponteiros.

Conceito


int **m = malloc(LIN * sizeof(int *));
**m
03610d0
0x3610d0
0x9a10e8
[0]
m[0] = malloc((LIN * COL) * sizeof(int));
0x3610d4
0x9a1100
[1]
0x3610d8
0x9a1118
[2]
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
0x9a10e8
0x9a10ec
0x9a10f0
0x9a1100
0x9a1104
0x9a1108
0x9a1118
0x9a111c
0x9a1120
for (i=1; i<LIN; i++)
m[i] = m[i-1]+COL;

Exemplo

Declara um ponteiro para ponteiro e aloca uma área de tamanho LIN para dados do tipo ponteiro para inteiro.

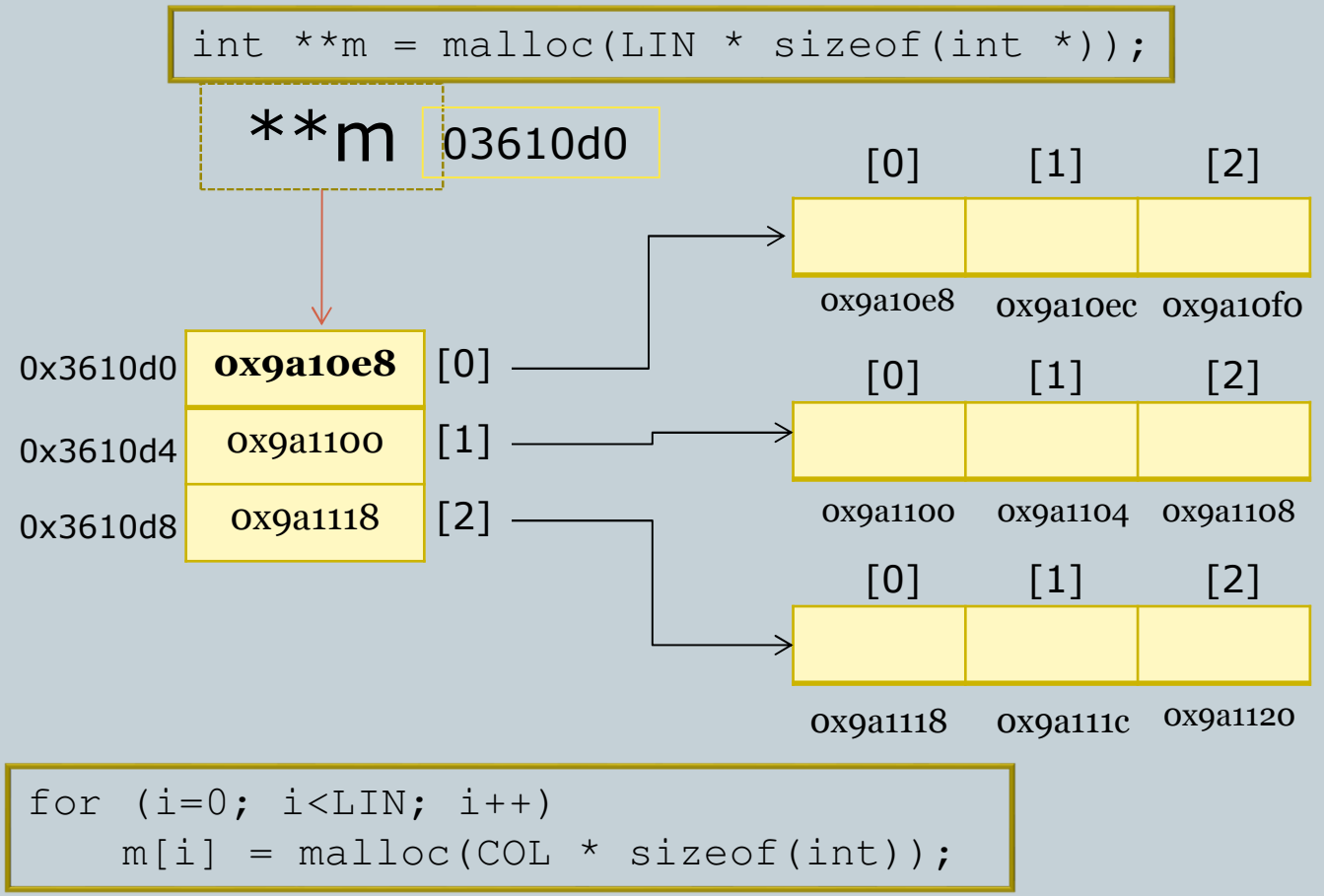
Aloca para cada posição do vetor de ponteiros um vetor de inteiros, que corresponde as posições da matriz

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  int LIN = 3, COL = 3, i, j;

  int **m = malloc(LIN * sizeof(int *));

  for (i=0; i<LIN; i++)
    m[i] = malloc(COL * sizeof(int));


 for(i=0; i<LIN; i++)
   for(j=0; j<COL; j++)
      m[i][j] = i+j;

 for(i=0; i<LIN; i++) {
   for(j=0; j<COL; j++) {
     printf("%d",m[i][j]);
   }
   printf("\n");
 }

 for (i=0; i<LIN; i++)
   free(m[i]);

 free(m);

 return 0;
}
```

Libera cada um dos vetores de dados alocados

Libera o vetor de ponteiros

Exemplo


int **m = malloc(LIN * sizeof(int *));
**m
03610d0
[0]
[1]
[2]
0x9a10e8
0x9a10ec
0x9a10f0
0x3610d0
0x9a10e8
[0]
0x3610d4
0x9a1100
[1]
0x3610d8
0x9a1118
[2]
[0]
[1]
[2]
0x9a1100
0x9a1104
0x9a1108
[0]
[1]
[2]
0x9a1118
0x9a111c
0x9a1120
for (i=0; i<LIN; i++)
m[i] = malloc(COL * sizeof(int));

Algoritmos II

ARITMÉTICA DE PONTEIROS

Conceito

- Como o vetor, é uma aritmética de endereços, e não de valores.

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  int *ptr, i,j;
  int    mat[3][3]={10,20,30,40,50,60,70,80,90};
  printf("%d\n",*(*( mat + 2 ) + 1));
  for (i=0;i<3;i++)
    for (j=0;j<3;j++)
      *(*(mat+i)+j) = *(*(mat+i)+j) * j;
  for (i=0;i<3;i++) {
    for (j=0;j<3;j++) {
      printf("%d\t",*(*(mat+i)+j));
    }
    printf("\n");
  }
  return 0;
}
```

No parênteses mais interno está deslocando a linha, e no segundo a coluna.

Neste laço é como se fizesse:
mat[i][j] = mat[i][j] * j;

Aqui está imprimindo a matriz modificada.

Algoritmos II

OPERADOR

Conceito

- Operador que pode ser usado para simplificar a notação para especificar os membros de uma struct.
- Combina as ações do operador * e do operador ponto (.).

```
#include <stdio.h>
#include <stdlib.h>

typedef struct{
   int dia, mes, ano;
}data;

int main () {
  data *hoje;
  hoje = malloc(sizeof(data));
  scanf("%d",&hoje->dia);
  scanf("%d",&hoje->mes);
  scanf("%d",&hoje->ano);
  printf("\n%02d/%02d/%04d",hoje->dia,hoje->mes,hoje->ano);
  return 0;
}
```

As notações abaixo possuem o mesmo significado:

data->dia

e

(*data).dia