

Alocação Dinâmica - Vetores e Matrizes (em C)

Eduardo Piveta

Vetores e Matrizes com alocação dinâmica em C

- Armazena-se um ponteiro para a primeira posição de um vetor.
- Matrizes são comumente representadas como vetores de vetores.
- Uma vez alocados, seus elementos são acessados da mesma forma que vetores estáticos.

Alocando um vetor de inteiros

exAlocacaoDinamica.cpp

vetorInteiros.cpp

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(int argc, char *argv[]) {
6      int* v = (int*) malloc(10 * sizeof(int));
7      if (v!=NULL){
8          printf("\nMemoria alocada com sucesso");
9          // Realiza operacoes
10     }
11     free(v);
12     return 0;
13 }
14
```

Librando a memória

exAlocacaoDinamica.cpp

vetorInteiros.cpp

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(int argc, char *argv[]) {
6      int* v = (int*) malloc(10 * sizeof(int));
7      if (v!=NULL){
8          printf("\nMemoria alocada com sucesso");
9          // Realiza operacoes
10     }
11     free(v);
12     return 0;
13 }
14
```

Acessando os elementos

```
4 ☐ int* criar(int tam){
5   |   return (int*) malloc(tam * sizeof(int));
6   | }
7
8 ☐ void mostrar(int* v, int n){
9   |   int i;
10  |   for(i = 0; i < n; i++)
11  |       printf("\nv[%d] = %d", i, v[i]);
12  | }
13
14 ☐ int main(int argc, char *argv[]) {
15  |   int* v = criar(10);
16  |   ☐ if (v!=NULL){
17  |       |   int i;
18  |       |   for(i = 0; i < 10; i++)
19  |       |       v[i] = i * 100;
20  |       |   mostrar(v, 10);
21  |       | }
22  |   free(v);
23  |   return 0;
24  | }
```

Passando v. dinâmicos por referência

```
4 int* criar(int tam){
5     return (int*) malloc(tam * sizeof(int));
6 }
7
8 void mostrar(int* v, int n){
9     int i;
10    for(i = 0; i < n; i++)
11        printf("\nv[%d] = %d", i, v[i]);
12 }
13
14 int main(int argc, char *argv[]) {
15     int* v = criar(10);
16     if (v!=NULL){
17         int i;
18         for(i = 0; i < 10; i++)
19             v[i] = i * 100;
20         mostrar(v, 10);
21     }
22     free(v);
23     return 0;
24 }
```

Retornando v. dinâmicos

```
4 ☐ int* criar(int tam){  
5     return (int*) malloc(tam * sizeof(int));  
6 }  
7  
8 ☐ void mostrar(int* v, int n){  
9     int i;  
10    for(i = 0; i < n; i++)  
11        printf("\nv[%d] = %d", i, v[i]);  
12 }  
13  
14 ☐ int main(int argc, char *argv[]) {  
15     ☐ int* v = criar(10);  
16     ☐ if (v!=NULL){  
17         int i;  
18         for(i = 0; i < 10; i++)  
19             v[i] = i * 100;  
20         mostrar(v, 10);  
21     }  
22     free(v);  
23     return 0;  
24 }
```

Alocando uma matriz de inteiros

```
int** criar(int m, int n){  
    int** matriz = (int**) malloc(m * sizeof(int*));  
    for (int i = 0; i < m; i++)  
        matriz[i] = (int*) malloc(n * sizeof(int));  
    return matriz;  
}
```

```
int main(int argc, char *argv[]) {  
    int** m = criar(3, 4);  
    if (m!=NULL) {  
        int count = 0;  
        for(int i = 0; i < 3; i++)  
            for(int j = 0; j < 4; j++)  
                m[i][j] = count++;  
        mostrar(m, 3, 4);  
    }  
    liberar(m, 3);  
    return 0;  
}
```


Retornando uma matriz de uma função

```
int** criar(int m, int n){  
    int** matriz = (int**) malloc(m * sizeof(int*));  
    for (int i = 0; i < m; i++)  
        matriz[i] = (int*) malloc(n * sizeof(int));  
    return matriz;  
}
```

```
int main(int argc, char *argv[]) {  
    int** m = criar(3, 4);  
    if (m!=NULL) {  
        int count = 0;  
        for(int i = 0; i < 3; i++)  
            for(int j = 0; j < 4; j++)  
                m[i][j] = count++;  
        mostrar(m, 3, 4);  
    }  
    liberar(m, 3);  
    return 0;  
}
```

Liberando a memória

```
void liberar(int** matriz, int m) {  
    for (int i = 0; i < m; i++)  
        free(matriz[i]);  
    free(matriz);  
}
```

```
int main(int argc, char *argv[]) {  
    int** m = criar(3, 4);  
    if (m!=NULL) {  
        int count = 0;  
        for(int i = 0; i < 3; i++)  
            for(int j = 0; j < 4; j++)  
                m[i][j] = count++;  
        mostrar(m, 3, 4);  
    }  
    liberar(m, 3);  
    return 0;  
}
```

Passando a matriz por referência

```
void liberar(int** matriz, int m) {  
    for (int i = 0; i < m; i++)  
        free(matriz[i]);  
    free(matriz);  
}
```

```
int main(int argc, char *argv[]) {  
    int** m = criar(3, 4);  
    if (m != NULL) {  
        int count = 0;  
        for(int i = 0; i < 3; i++)  
            for(int j = 0; j < 4; j++)  
                m[i][j] = count++;  
        mostrar(m, 3, 4);  
    }  
    liberar(m, 3);  
    return 0;  
}
```

Acessando os elementos

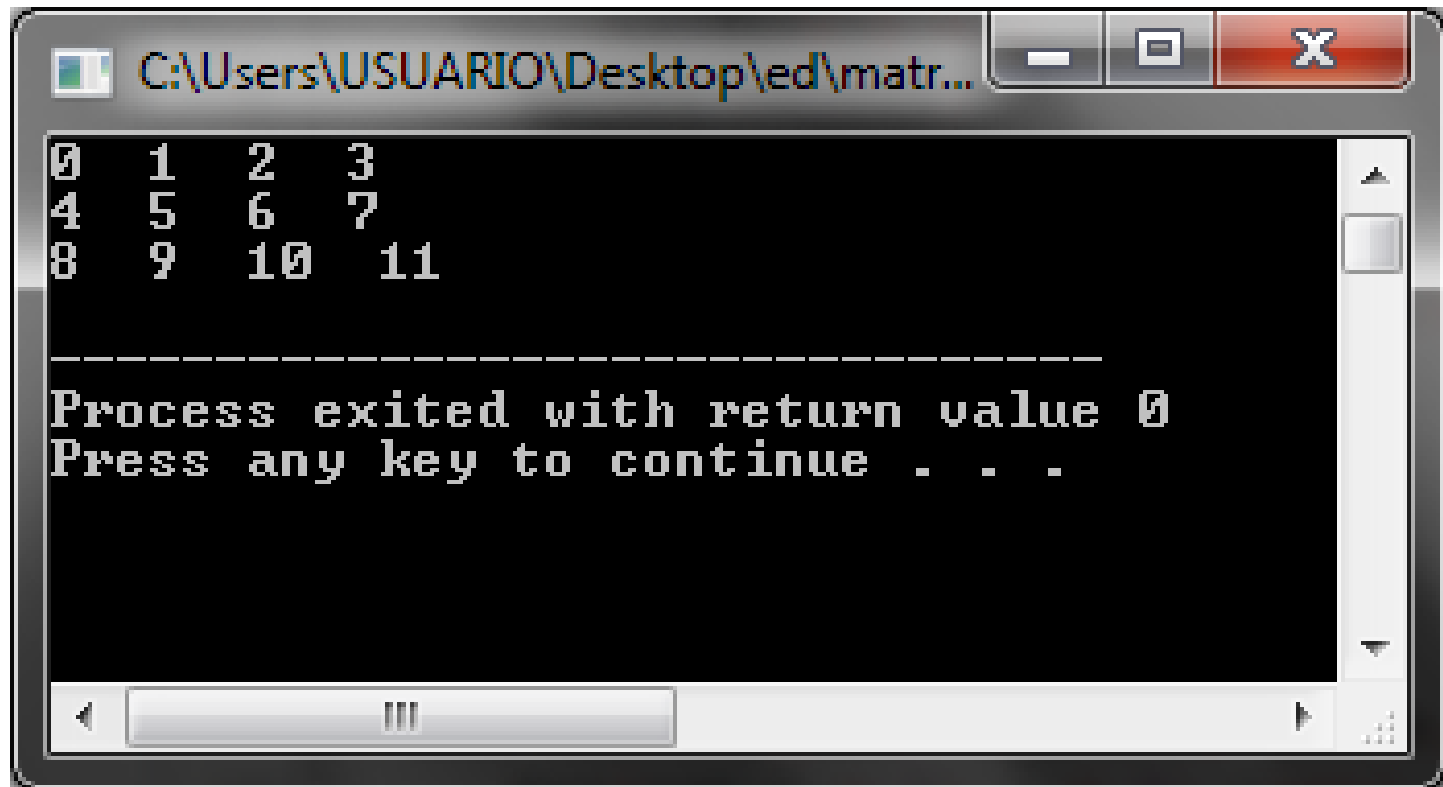
```
int main(int argc, char *argv[]) {  
    int** m = criar(3, 4);  
    if (m!=NULL) {  
        int count = 0;  
        for(int i = 0; i < 3; i++)  
            for(int j = 0; j < 4; j++)  
                m[i][j] = count++;  
        mostrar(m, 3, 4);  
    }  
    liberar(m, 3);  
    return 0;  
}
```

Passando a matriz por referência

```
void mostrar(int** matriz, int m, int n){  
    for(int i = 0; i < m; i++){  
        for(int j = 0; j < n; j++){  
            printf("%d  ", matriz[i][j]);  
        }  
        printf("\n");  
    }  
}
```

```
int main(int argc, char *argv[]) {  
    int** m = criar(3, 4);  
    if (m!=NULL) {  
        int count = 0;  
        for(int i = 0; i < 3; i++){  
            for(int j = 0; j < 4; j++){  
                m[i][j] = count++;  
            }  
            mostrar(m, 3, 4);  
        }  
        liberar(m, 3);  
        return 0;  
    }  
}
```

Mostrando os elementos da matriz



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\USUARIO\Desktop\ed\matr...". The window contains a 3x4 matrix of numbers: 0 1 2 3, 4 5 6 7, and 8 9 10 11. Below the matrix, a dashed line separates it from the text "Process exited with return value 0" and "Press any key to continue . . .". The window has standard Windows controls (minimize, maximize, close) and a scrollbar on the right.

```
0 1 2 3
4 5 6 7
8 9 10 11

-----
Process exited with return value 0
Press any key to continue . . .
```

Matrizes de estruturas

- Comumente, necessitamos de vetores e matrizes de estruturas (e não somente de tipos simples).
- Os operadores e funcionalidades são iguais.
- No entanto, deve-se prestar atenção no que se deseja.

Exemplo: Matrizes de Estruturas

- Considere a seguinte estrutura (usando a sintaxe simplificada de C++):

```
struct Ponto{  
    int x;  
    int y;  
};
```


Exemplo: Matrizes de Estruturas

- O programador poderia querer, por exemplo:
 - Uma matriz estática
 - De pontos estáticos
 - De pontos dinâmicos
 - Uma matriz dinâmica
 - De pontos estáticos
 - De pontos dinâmicos

Exemplo: Matrizes de Estruturas

- O programador poderia querer, por exemplo:
 - Uma matriz estática
 - De pontos estáticos
 - De pontos dinâmicos
 - Uma matriz dinâmica
 - De pontos estáticos
 - De pontos dinâmicos

Matriz estática de pontos estáticos

```
// Matriz estatica de pontos estaticos  
Ponto p0[10][10];  
p0[0][0].x = 10;  
printf("\np0.x: %d\n", p0[0][0].x);
```

Matriz estática de pontos estáticos

```
// Matriz estatica de pontos estaticos  
Ponto p0[10][10];  
p0[0][0].x = 10;  
printf("\np0.x: %d\n", p0[0][0].x);
```

Matriz estática

Matriz estática de pontos estáticos

```
// Matriz estatica de pontos estaticos
Ponto p0[10][10];
p0[0][0].x = 10;
printf("\np0.x: %d\n", p0[0][0].x);
```

Pontos estáticos

Escrita e leitura
estáticas

Exemplo: Matrizes de Estruturas

- O programador poderia querer, por exemplo:
 - Uma matriz estática
 - De pontos estáticos
 - De pontos dinâmicos
 - Uma matriz dinâmica
 - De pontos estáticos
 - De pontos dinâmicos

Matriz estática de pontos dinâmicos

```
// Matriz estatica de pontos dinamicos
Ponto* p1[10][10];
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        p1[i][j] = (Ponto*) malloc(sizeof(Ponto));
p1[0][0]->x = 10;
printf("\np1->x: %d\n", p1[0][0]->x);
```

Matriz estática de pontos dinâmicos

```
// Matriz estatica de pontos dinamicos
Ponto* p1[10][10];
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        p1[i][j] = (Ponto*) malloc(sizeof(Ponto));
p1[0][0]->x = 10;
printf("\np1->x: %d\n", p1[0][0]->x);
```

Matriz estática

Matriz estática de pontos dinâmicos

```
// Matriz estatica de pontos dinamicos
```

```
Ponto* p1[10][10];
```

```
for (int i = 0; i < 10; i++)
```

```
    for (int j = 0; j < 10; j++)
```

```
        p1[i][j] = (Ponto*) malloc(sizeof(Ponto));
```

```
p1[0][0]->x = 10;
```

```
printf("\np1->x: %d\n", p1[0][0]->x);
```

Pontos dinâmicos

Escrita e leitura
dinâmicas

Matriz estática de pontos dinâmicos

```
// Matriz estatica de pontos dinamicos
Ponto* p1[10][10];
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        p1[i][j] = (Ponto*) malloc(sizeof(Ponto));
p1[0][0]->x = 10;
printf("\np1->x: %d\n", p1[0][0]->x);
```

A alocação dos pontos é dinâmica
(cada um dos pontos da matriz
é alocado dinamicamente)

Exemplo: Matrizes de Estruturas

- O programador poderia querer, por exemplo:
 - Uma matriz estática
 - De pontos estáticos
 - De pontos dinâmicos
 - Uma matriz dinâmica
 - De pontos estáticos
 - De pontos dinâmicos

Matriz dinâmica de pontos estáticos

```
Ponto** alocarMatrizDinamicaDePontosEstaticos(int nrLinhas, int nrColunas){
    Ponto** matrix = (Ponto**) malloc(nrLinhas * sizeof(Ponto*));
    for (int i = 0; i < nrLinhas; i++)
        matrix[i] = (Ponto*) malloc(nrColunas * sizeof(Ponto));
    return matrix;
}
```

```
// Matriz dinamica de pontos estaticos
```

```
Ponto** p2 = alocarMatrizDinamicaDePontosEstaticos(10, 10);
p2[0][0].x = 10;
printf("\np2.x: %d\n", p2[0][0].x);
```

Matriz dinâmica

Matriz dinâmica de pontos estáticos

```
Ponto** alocarMatrizDinamicaDePontosEstaticos(int nrLinhas, int nrColunas){  
    Ponto** matrix = (Ponto**) malloc(nrLinhas * sizeof(Ponto*));  
    for (int i = 0; i < nrLinhas; i++)  
        matrix[i] = (Ponto*) malloc(nrColunas * sizeof(Ponto));  
    return matrix;  
}
```

```
// Matriz dinamica de pontos estaticos
```

```
Ponto** p2 = alocarMatrizDinamicaDePontosEstaticos(10, 10);  
p2[0][0].x = 10;  
printf("\np2.x: %d\n", p2[0][0].x);
```

Alocação da matriz é dinâmica

Matriz dinâmica de pontos estáticos

```
Ponto** alocarMatrizDinamicaDePontosEstaticos(int nrLinhas, int nrColunas){  
    Ponto** matrix = (Ponto**) malloc(nrLinhas * sizeof(Ponto));  
    for (int i = 0; i < nrLinhas; i++)  
        matrix[i] = (Ponto*) malloc(nrColunas * sizeof(Ponto));  
    return matrix;  
}
```

```
// Matriz dinamica de pontos estaticos
```

```
Ponto** p2 = alocarMatrizDinamicaDePontosEstaticos(10, 10);  
p2[0][0].x = 10;  
printf("\np2.x: %d\n", p2[0][0].x);
```

Pontos estáticos

Escrita e leitura
estáticas

Exemplo: Matrizes de Estruturas

- O programador poderia querer, por exemplo:
 - Uma matriz estática
 - De pontos estáticos
 - De pontos dinâmicos
 - Uma matriz dinâmica
 - De pontos estáticos
 - De pontos dinâmicos

Matriz dinâmica de pontos dinâmicos

```
Ponto*** alocarMatrizDinamicaDePontosDinamicos(int nrLinhas, int nrColunas){
    Ponto*** matrix = (Ponto***) malloc(nrLinhas * sizeof(Ponto**));
    for (int i = 0; i < nrLinhas; i++)
        matrix[i] = (Ponto**) malloc(nrColunas * sizeof(Ponto*));
    for (int i = 0; i < nrLinhas; i++)
        for (int j = 0; j < nrColunas; j++)
            matrix[i][j] = (Ponto*) malloc(sizeof(Ponto));
    return matrix;
}

// Matriz dinamica de pontos dinamicos
Ponto*** p3 = alocarMatrizDinamicaDePontosDinamicos(10, 10);
p3[0][0]->x = 10;
printf("\np3->x: %d\n", p3[0][0]->x);
```

Matriz dinâmica
(alocação dinâmica)

Matriz dinâmica de pontos dinâmicos

```
Ponto*** alocarMatrizDinamicaDePontosDinamicos(int nrLinhas, int nrColunas){
    Ponto*** matrix = (Ponto***) malloc(nrLinhas * sizeof(Ponto**));
    for (int i = 0; i < nrLinhas; i++)
        matrix[i] = (Ponto**) malloc(nrColunas * sizeof(Ponto*));
    for (int i = 0; i < nrLinhas; i++)
        for (int j = 0; j < nrColunas; j++)
            matrix[i][j] = (Ponto*) malloc(sizeof(Ponto));
    return matrix;
}
```

// Matriz dinamica de pontos dinamicos

```
Ponto*** p3 = alocarMatrizDinamicaDePontosDinamicos(10, 10);
p3[0][0]->x = 10;
printf("\np3->x: %d\n", p3[0][0]->x);
```

Pontos dinâmicos
(alocação dinâmica,
leituras e escritas dinâmicas)

Comparando...

```
// Matriz estatica de pontos estaticos
Ponto p0[10][10];
p0[0][0].x = 10;
printf("\np0.x: %d\n", p0[0][0].x);

// Matriz estatica de pontos dinamicos
Ponto* p1[10][10];
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        p1[i][j] = (Ponto*) malloc(sizeof(Ponto));
p1[0][0]->x = 10;
printf("\np1->x: %d\n", p1[0][0]->x);

// Matriz dinamica de pontos estaticos
Ponto** p2 = alocarMatrizDinamicaDePontosEstaticos(10, 10);
p2[0][0].x = 10;
printf("\np2.x: %d\n", p2[0][0].x);

// Matriz dinamica de pontos dinamicos
Ponto*** p3 = alocarMatrizDinamicaDePontosDinamicos(10, 10);
p3[0][0]->x = 10;
printf("\np3->x: %d\n", p3[0][0]->x);
```