



INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Charqueadas

EDUCAÇÃO
PÚBLICA
100%
GRATUITA

Array: Vetores e Matrizes

Programação Estruturada

Prof. André del Mestre

Arrays

Fundamentos

- **Definição:** Sequencia de elementos de mesmo tipo
 - Vetor - Array unidimensional
 - Matriz - Array bidimensional
- **Por que usar arrays?**
 - Estruturar dados semelhantes
 - Lidar com alta quantidade de dados semelhantes

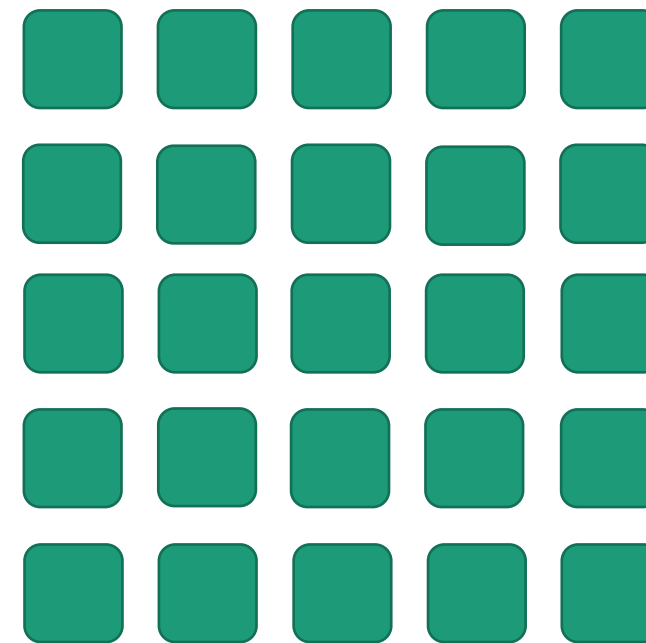
Variavel



Vetor



Matriz



Arrays

Fundamentos

- **Por que usar arrays?**

- Imagina que eu quero processar notas da turma inteira (32 alunos) e escolher qual aluno desejo saber se aprovou ou não.
- esse programa serve?

```
int main () {  
    float nota;  
    printf("digite a nota do aluno\n");  
    scanf("%f", &nota);  
    if(nota >= 6.0) {  
        printf("Aluno(a) aprovado(a)\n");  
    } else {  
        printf("Aluno(a) em reavaliacao\n");  
    }  
    return 0;  
}
```

Estruturas de Seleção

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?

```
int main () {  
    float nota;  
    for(i=0;i<32;i++) {  
        printf("digite a nota do aluno\n");  
        scanf("%f", &nota);  
        if(nota>=6.0)  
            printf("Aluno aprovado");  
        else  
            printf("Aluno em reavaliacao");  
    }  
    return 0;  
}
```

Estruturas de Repetição

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?

```
int main () {  
    float nota; int id_aluno;  
    for(i=0;i<32;i++) {  
        printf("digite a nota do aluno\n");  
        scanf("%f", &nota);  
    }  
    printf("Qual id do aluno que você ");  
    printf("quer saber se esta aprovado?\n");  
    scanf("%i", &id_aluno);  
    if(nota>=6.0)  
        printf("Aluno(a) aprovado(a)\n");  
    else  
        printf("Aluno(a) em reavaliacao\n");  
}
```


Tentando outra Solução

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?

A partir daqui
Não se chega na
solução



```
int main () {
    float nota; int id_aluno;
    for(i=0;i<32;i++) {
        printf("digite a nota do aluno\n");
        scanf("%f", &nota);
    }
    printf("Qual id do aluno que você ");
    printf("quer saber se esta aprovado?\n");
    scanf("%i", &id_aluno);
    if(nota>=6.0)
        printf("Aluno(a) aprovado(a)\n");
    else
        printf("Aluno(a) em reavaliacao\n");
}
```

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?



```
Digite a nota do aluno:  
9.5
```

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?

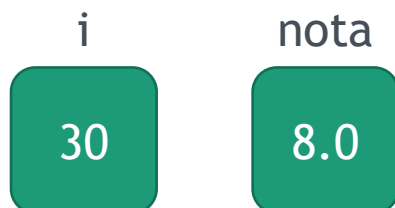


```
Digite a nota do aluno:  
9.5  
Digite a nota do aluno:  
8.5
```


Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?



```
Digite a nota do aluno:  
9.5  
Digite a nota do aluno:  
8.5  
.....  
Digite a nota do aluno:  
8.0
```

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?

i	nota
31	5.5

Digite a nota do aluno:

9.5

Digite a nota do aluno:

8.5

.....

Digite a nota do aluno:

8.0

Digite a nota do aluno:

5.5

Qual id do aluno que você quer saber se esta aprovado?

0

REAVALIACAO

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?

i	nota
31	5.5

```
int main () {  
    float nota; int id_aluno;  
    for(i=0;i<32;i++) {  
        printf("digite a nota do aluno\n");  
        scanf("%f", &nota);  
    }  
    printf("Qual id do aluno que você ");  
    printf("quer saber se esta aprovado?\n");  
    scanf("%i", &id_aluno);  
    if(nota>=6.0)  
        printf("Aluno(a) aprovado(a)\n");  
    else  
        printf("Aluno(a) em reavaliacao\n");  
}
```

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?



```
Digite a nota do aluno:  
9.5
```

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?



```
Digite a nota do aluno:  
9.5  
Digite a nota do aluno:  
8.5
```

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?



```
Digite a nota do aluno:  
9.5  
Digite a nota do aluno:  
8.5  
.....  
Digite a nota do aluno:  
8.0
```

Arrays

Fundamentos

- **Por que usar arrays?**
 - Imagina que eu quero processar notas da turma inteira (32 alunos) e **escolher qual aluno** desejo saber se aprovou ou não.
 - esse programa serve?



Digite a nota do aluno:

9.5

Digite a nota do aluno:

8.5

.....

Digite a nota do aluno:

8.0

Digite a nota do aluno:

5.5

Qual id do aluno que você quer saber se esta aprovado?

0

APROVADO

Utilizando Arrays

Arrays

Declaração

x= 

vetor= 

matriz= 

```
int main () {  
    int x, vetor[7], matriz[5][5];  
  
    float outra_matriz[10][10];  
  
    char outro_vetor[7];  
}
```

Arrays

Inicialização

x= 

vetor= 

matriz= 

```
int main () {  
    int x;  
  
    int vetor[7];  
  
    int matriz[5][5];  
}
```

Arrays

Inicialização

x=

-1

vetor=

0	1	0	1	0	1	0
---	---	---	---	---	---	---

matriz=

0	1	0	1	0
0	0	0	0	0
2	3	4	3	2
0	0	0	0	0
0	1	0	1	0

```
int main () {  
    int x = -1;  
  
    int vetor[7] = {0,1,0,1,0,1,0};  
  
    int matriz[5][5] = {  
        {0,1,0,1,0},  
        {0,0,0,0,0},  
        {2,3,4,3,2},  
        {0,0,0,0,0},  
        {0,1,0,1,0}  
    };  
}
```

Arrays

Atribuição

x=

-1

vetor=

?

?

?

?

?

?

?

matriz=

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

```
int main () {  
    int x, vetor[7], matriz[5][5];  
    x=-1;  
}
```

Arrays

Atribuição

x=

-1

vetor=

9

8

3

3

-1

?

?

matriz=

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

```
int main () {  
    int x, vetor[7], matriz[5][5];  
    x=-1;  
  
    vetor[0]=9;    vetor[1]=8;  
    scanf("%i", &vetor[2]);  
    scanf("%i", vetor+3); //scanf("%i", &vetor[3]);  
    vetor[4]=x;  
}
```

Arrays

Atribuição

x=

vetor=

matriz=

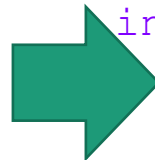
9	?	?	?	?
?	?	?	?	?
?	?	?	?	?
?	?	3	?	?
?	?	?	?	8

```
int main () {  
    int x, vetor[7], matriz[5][5];  
    x=-1;  
  
    vetor[0]=9;    vetor[1]=8;  
    scanf("%i", &vetor[2]);  
    scanf("%i", vetor+3); //scanf("%i", &vetor[3])  
    vetor[4]=x;  
  
    matriz[0][0]=9;    matriz[4][4]=8;  
    scanf("%i", &matriz[3][2]);  
}
```

Vetores e Estruturas de Repetição

Vetores e Estruturas de Repetição

Declaração




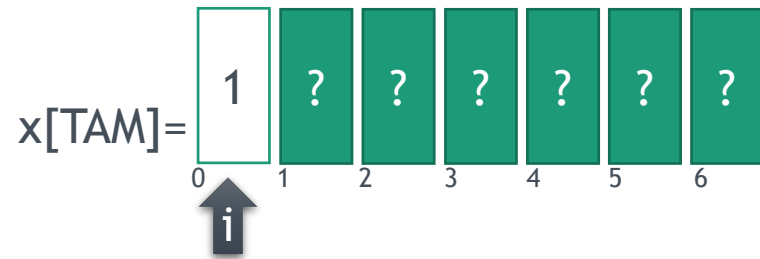
```
int main () {  
    int i, x[7], num;
```



ATENCAO AOS INDICES!

Vetores e Estruturas de Repetição


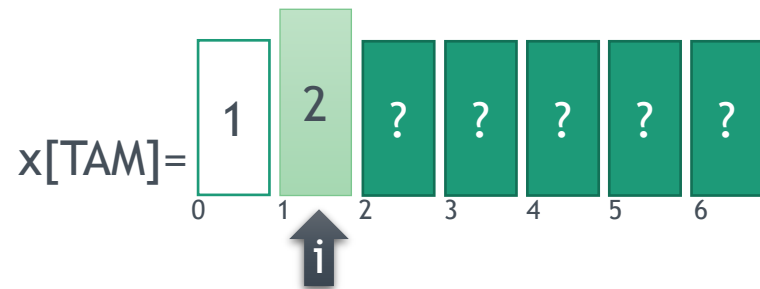
Inicialização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++) {  
        x[i]=i+1;  
    }  
}
```

Vetores e Estruturas de Repetição


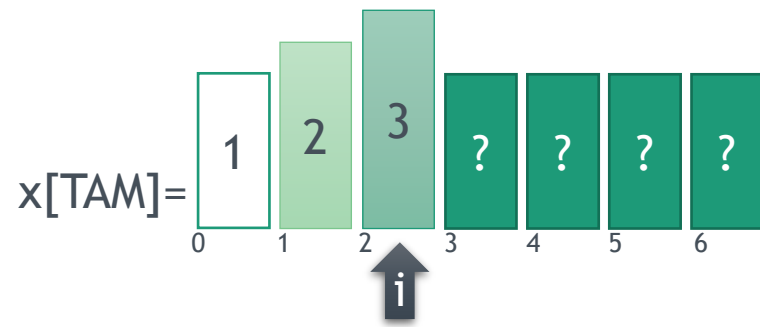
Inicialização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++) {  
        x[i]=i+1;  
    }  
}
```

Vetores e Estruturas de Repetição


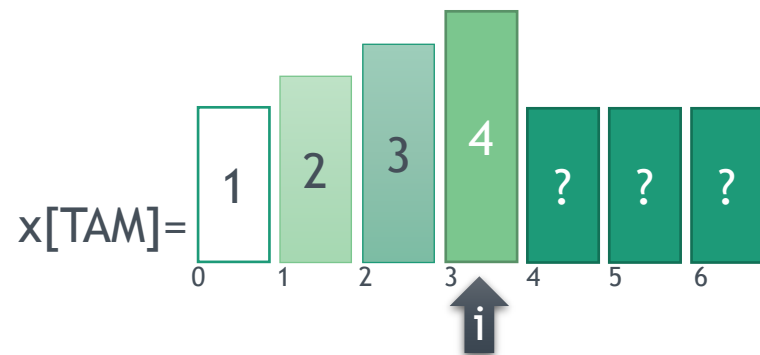
Inicialização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++) {  
        x[i]=i+1;  
    }  
}
```

Vetores e Estruturas de Repetição


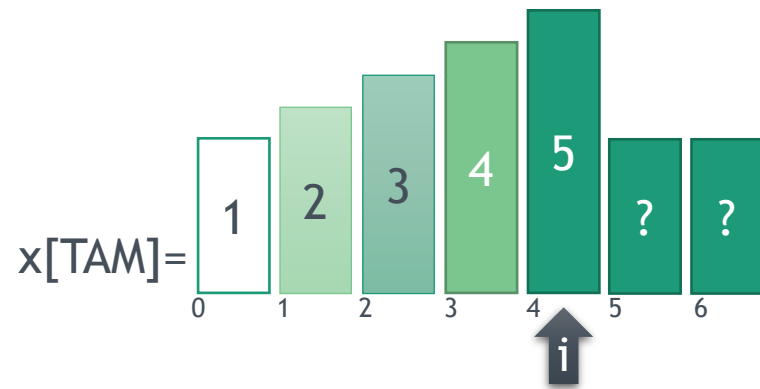
Inicialização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++) {  
        x[i]=i+1;  
    }  
}
```

Vetores e Estruturas de Repetição


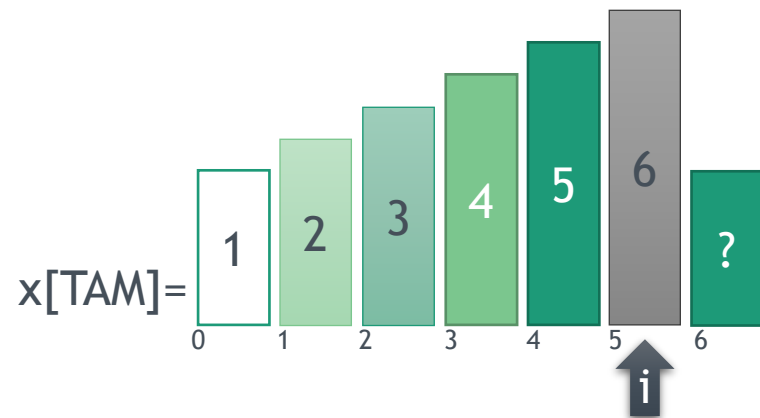
Inicialização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++) {  
        x[i]=i+1;  
    }  
}
```

Vetores e Estruturas de Repetição


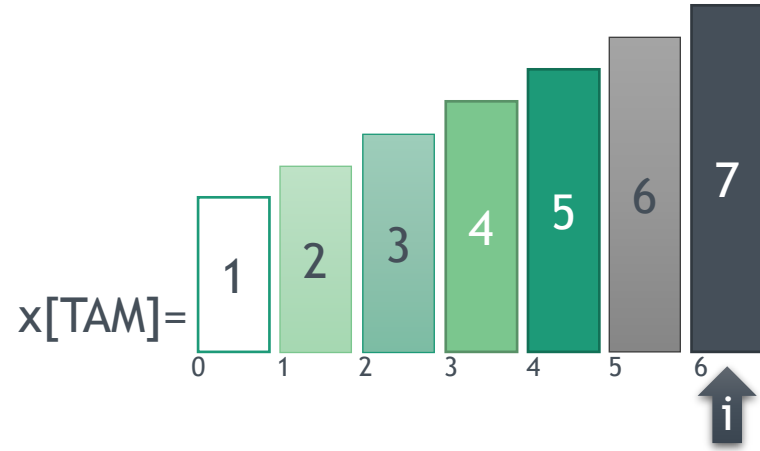
Inicialização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++) {  
        x[i]=i+1;  
    }  
}
```

Vetores e Estruturas de Repetição

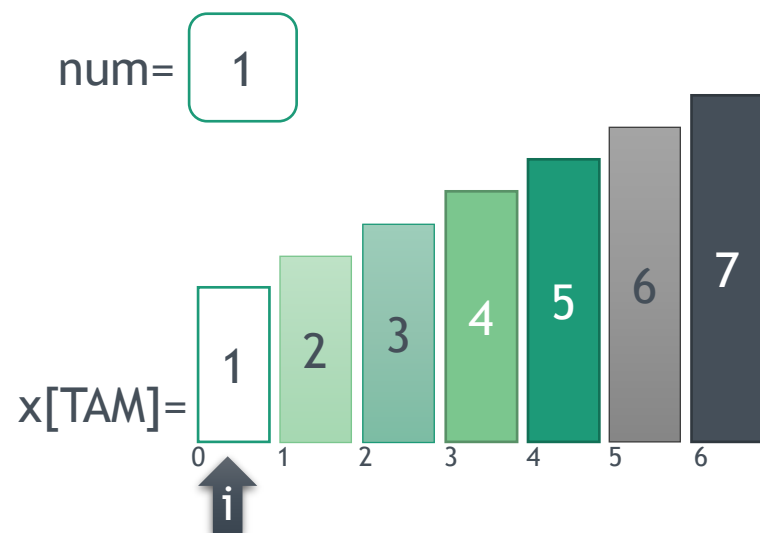
Inicialização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++) {  
        x[i]=i+1;  
    }  
}
```

Vetores e Estruturas de Repetição

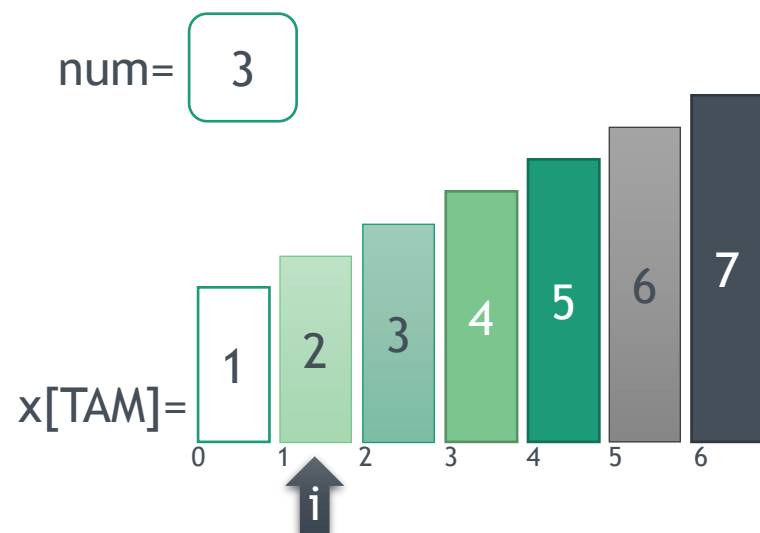
Utilização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++){  
        x[i]=i+1;  
    }  
    ...  
    num=0;  
    for(i=0; i<7; i++){  
        num += vet[i];  
    }  
}
```


Vetores e Estruturas de Repetição

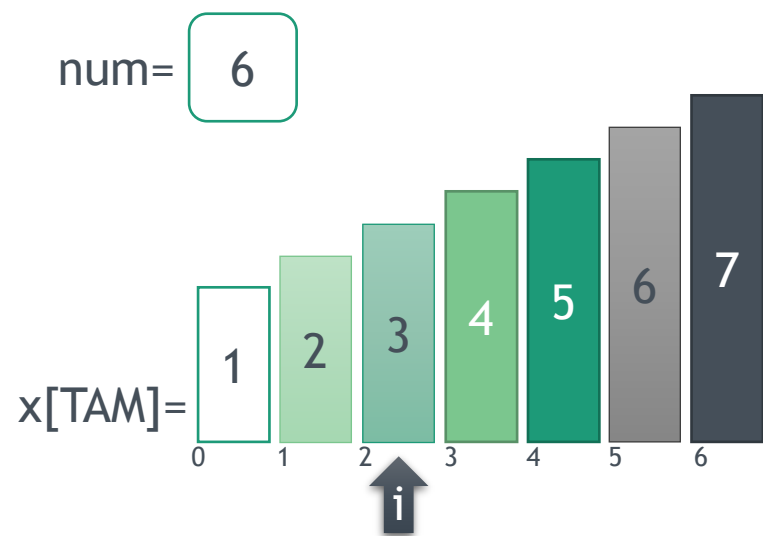
Utilização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++){  
        x[i]=i+1;  
    }  
    ...  
    num=0;  
    for(i=0; i<7; i++){  
        num += vet[i];  
    }  
}
```

Vetores e Estruturas de Repetição

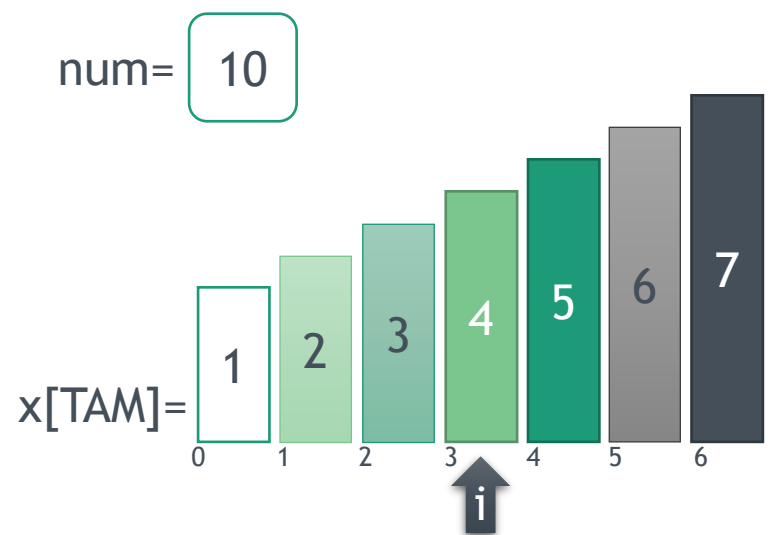
Utilização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++){  
        x[i]=i+1;  
    }  
    ...  
    num=0;  
    for(i=0; i<7; i++){  
        num += vet[i];  
    }  
}
```

Vetores e Estruturas de Repetição

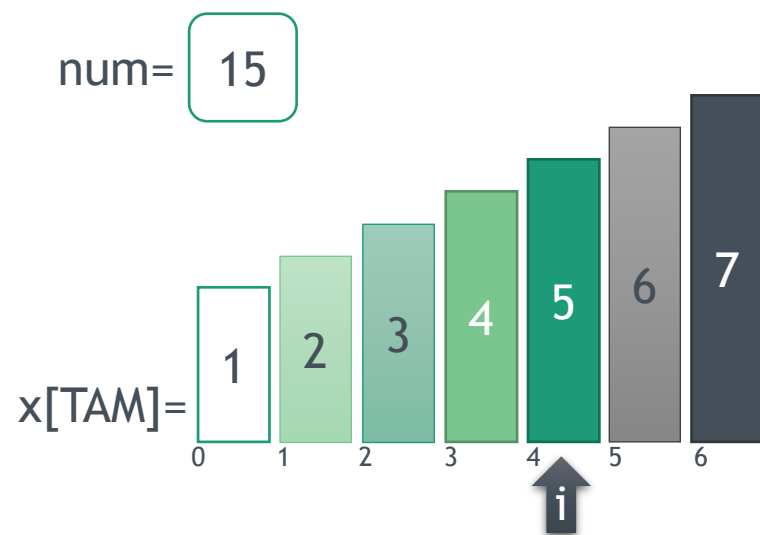
Utilização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++){  
        x[i]=i+1;  
    }  
    ...  
    num=0;  
    for(i=0; i<7; i++){  
        num += vet[i];  
    }  
}
```

Vetores e Estruturas de Repetição

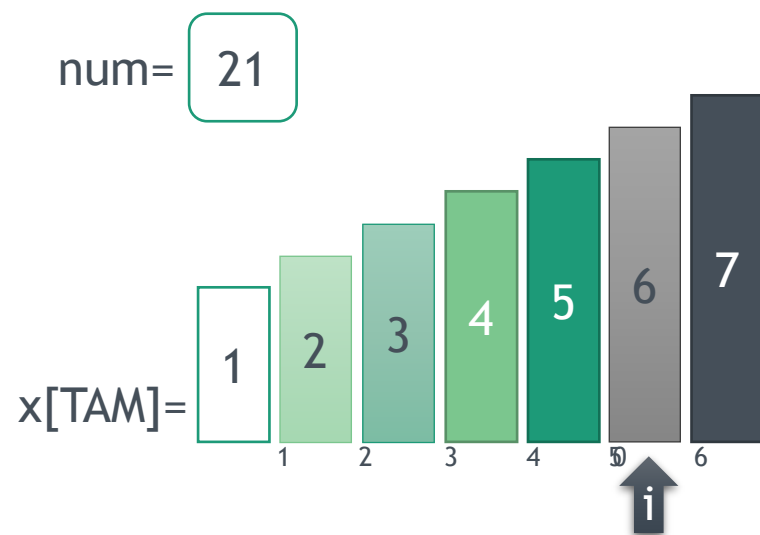
Utilização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++){  
        x[i]=i+1;  
    }  
    ...  
    num=0;  
    for(i=0; i<7; i++){  
        num += vet[i];  
    }  
}
```

Vetores e Estruturas de Repetição

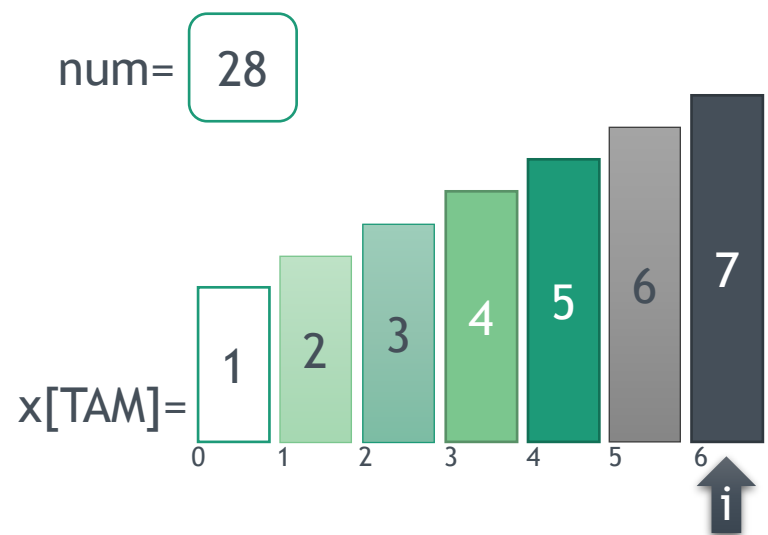
Utilização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++){  
        x[i]=i+1;  
    }  
    ...  
    num=0;  
    for(i=0; i<7; i++){  
        num += vet[i];  
    }  
}
```

Vetores e Estruturas de Repetição

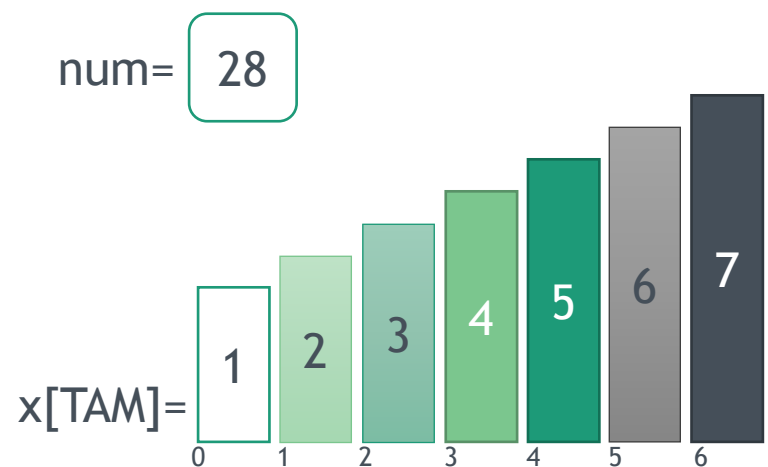
Utilização



```
int main () {  
    int i, x[7], num;  
    for(i=0; i<7; i++){  
        x[i]=i+1;  
    }  
    ...  
    num=0;  
    for(i=0; i<7; i++){  
        num += vet[i];  
    }  
}
```

Vetores e Estruturas de Repetição

Visualização



```
x[0] = 1
x[1] = 2
x[2] = 3
x[3] = 4
x[4] = 5
x[5] = 6
x[6] = 7
```

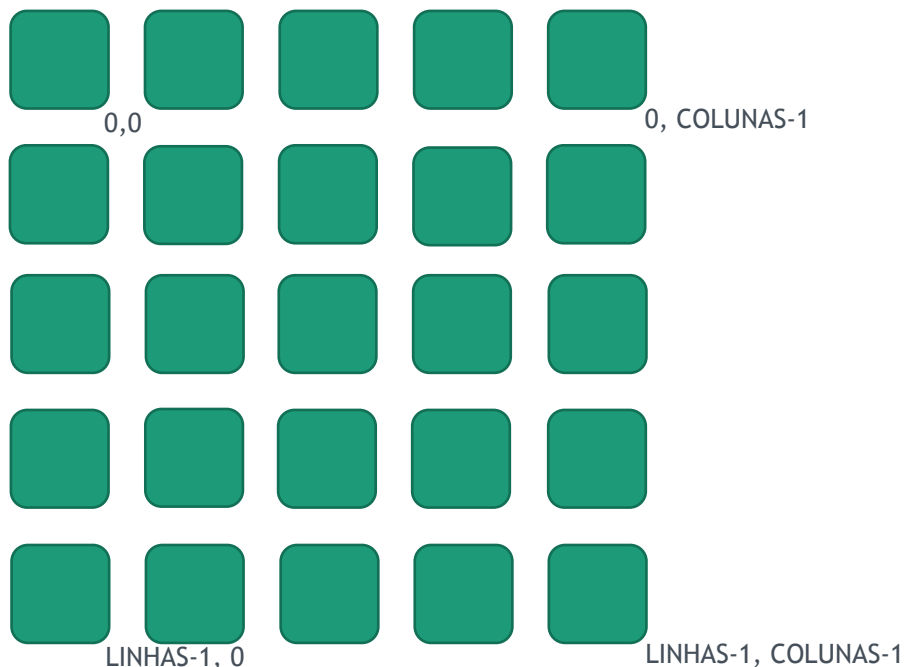
```
int main () {
    int i, x[7], num;
    for(i=0; i<7; i++){
        x[i]=i+1;
    }
    ...
    num=0;
    for(i=0; i<7; i++){
        num += vet[i];
    }
    ...
    for(i=0; i<7; i++){
        printf("x[%i] = %i\n", i, x[i]);
    }
}
```



Matrizes e Estruturas de Repetição

Matrizes e Estruturas de Repetição

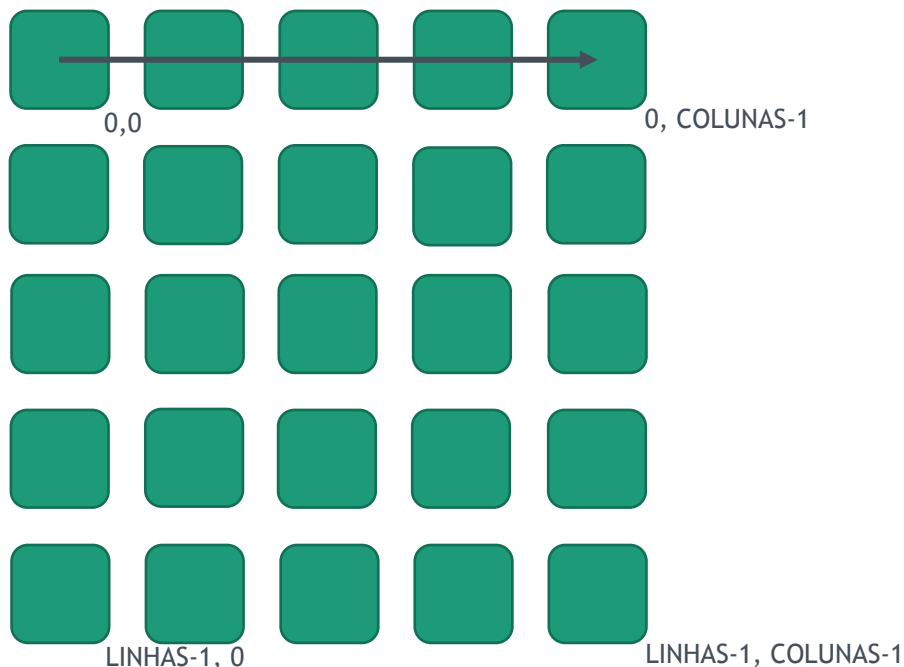
Declaração



```
int cont=0, i, j, LINHAS=5, COLUNAS=5;
int m[LINHAS][COLUNAS];
for(i=0; i<LINHAS; i++){
    for(j=0; j<COLUNAS; j++){
        m[i][j] = cont;
        cont++;
    }
}
```

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

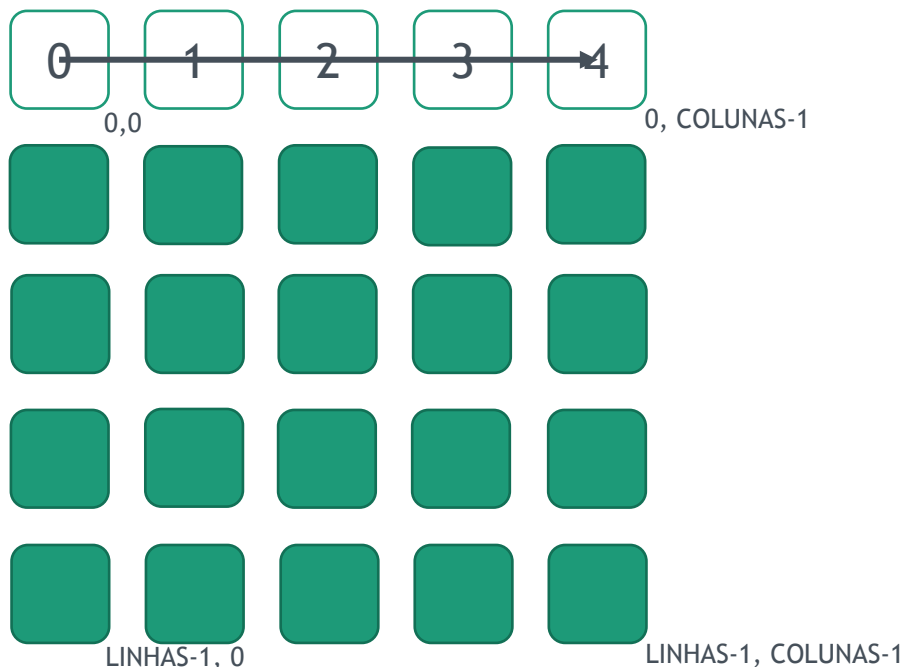


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;  
int m[LINHAS][COLUNAS];  
for(i=0; i<LINHAS; i++){  
    for(j=0; j<COLUNAS; j++){  
        m[i][j] = cont;  
        cont++;  
    }  
}
```

Vai preencher a primeira linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

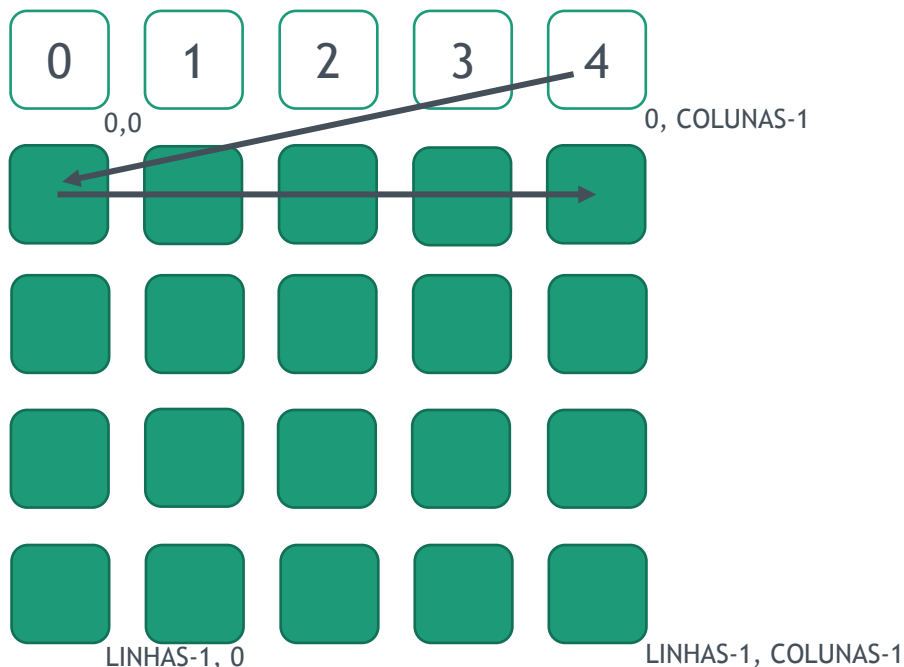


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;
int m[LINHAS][COLUNAS];
for(i=0; i<LINHAS; i++){
    for(j=0; j<COLUNAS; j++){
        m[i][j] = cont;
        cont++;
    }
}
```

Preenche a
primeira linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

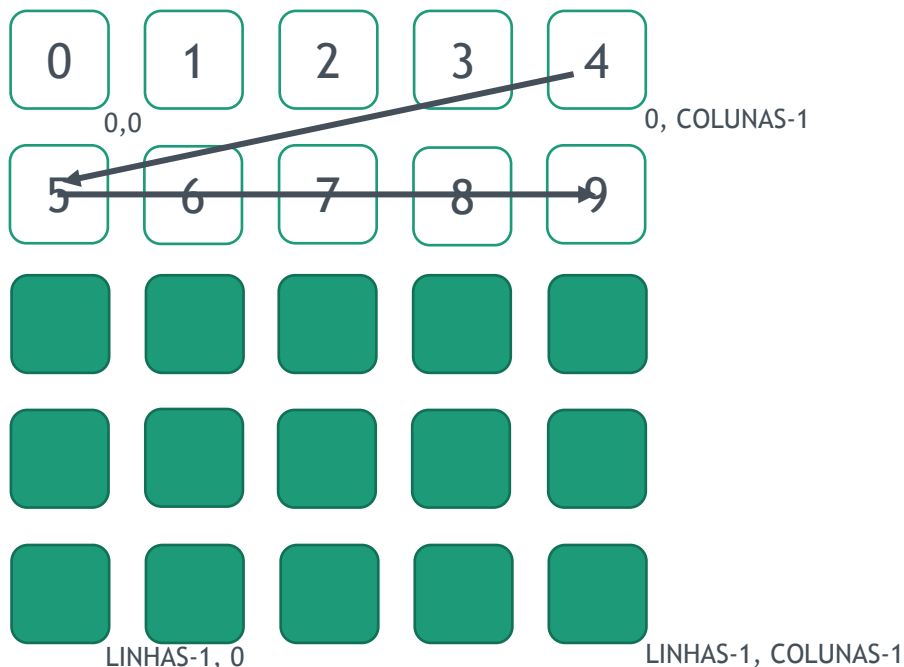


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;
int m[LINHAS][COLUNAS];
for(i=0; i<LINHAS; i++){
    for(j=0; j<COLUNAS; j++){
        m[i][j] = cont;
        cont++;
    }
}
```

Vai preencher a
segunda linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

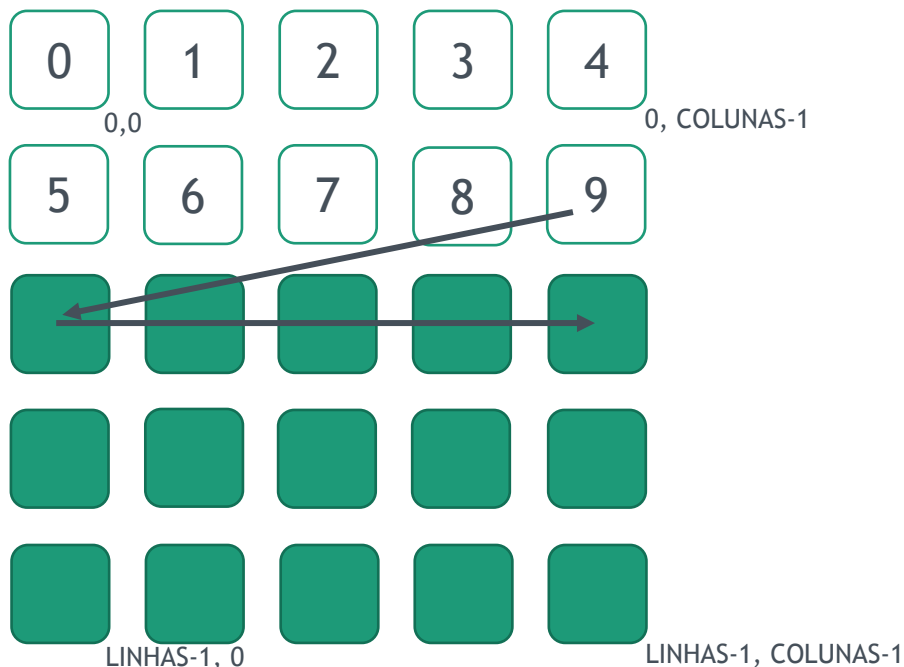


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;  
int m[LINHAS][COLUNAS];  
for(i=0; i<LINHAS; i++){  
    for(j=0; j<COLUNAS; j++){  
        m[i][j] = cont;  
        cont++;  
    }  
}
```

Preenche a
segunda linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

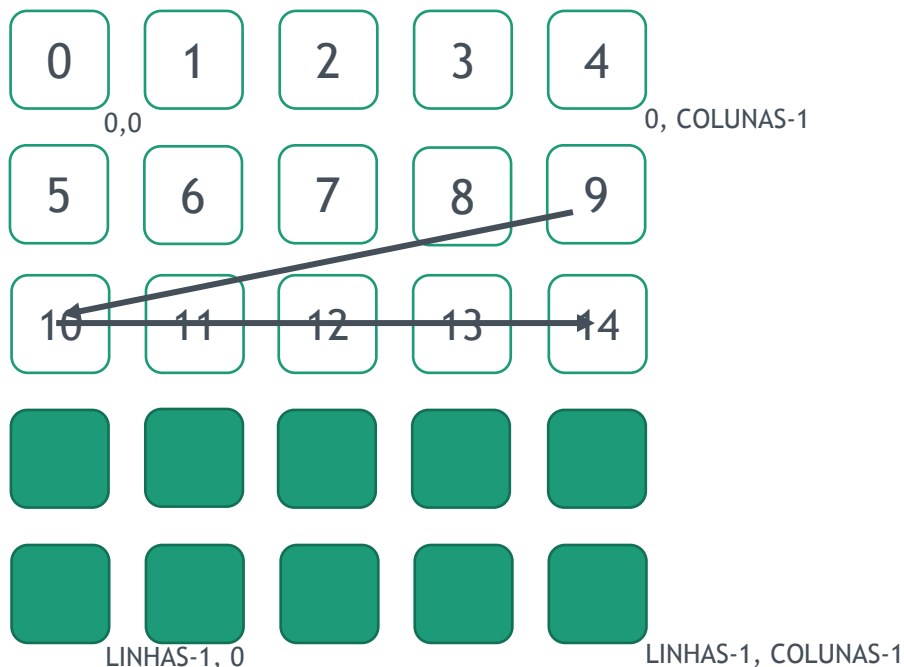


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;
int m[LINHAS][COLUNAS];
for(i=0; i<LINHAS; i++){
    for(j=0; j<COLUNAS; j++){
        m[i][j] = cont;
        cont++;
    }
}
```

Vai preencher a
terceira linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

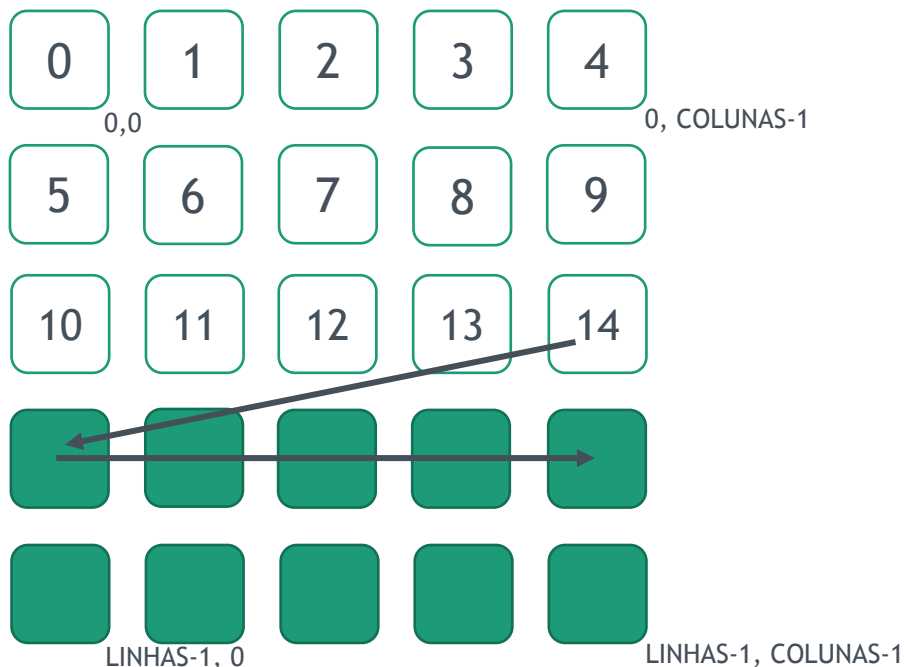


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;
int m[LINHAS][COLUNAS];
for(i=0; i<LINHAS; i++){
    for(j=0; j<COLUNAS; j++){
        m[i][j] = cont;
        cont++;
    }
}
```

Preenche a
terceira linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

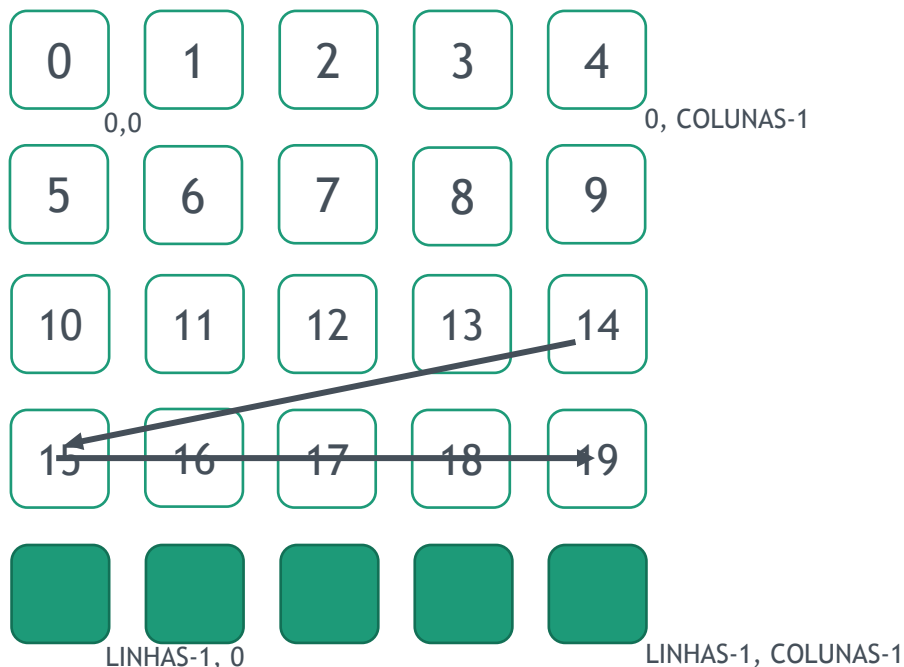


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;  
int m[LINHAS][COLUNAS];  
for(i=0; i<LINHAS; i++){  
    for(j=0; j<COLUNAS; j++){  
        m[i][j] = cont;  
        cont++;  
    }  
}
```

Vai preencher a quarta linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

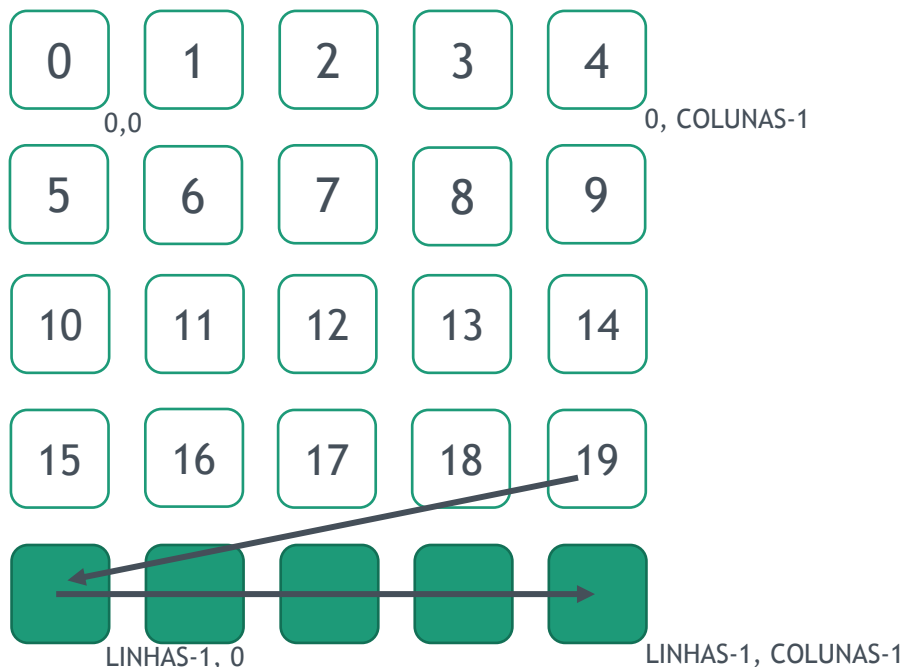


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;
int m[LINHAS][COLUNAS];
for(i=0; i<LINHAS; i++){
    for(j=0; j<COLUNAS; j++){
        m[i][j] = cont;
        cont++;
    }
}
```

Preenche a
quarta linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

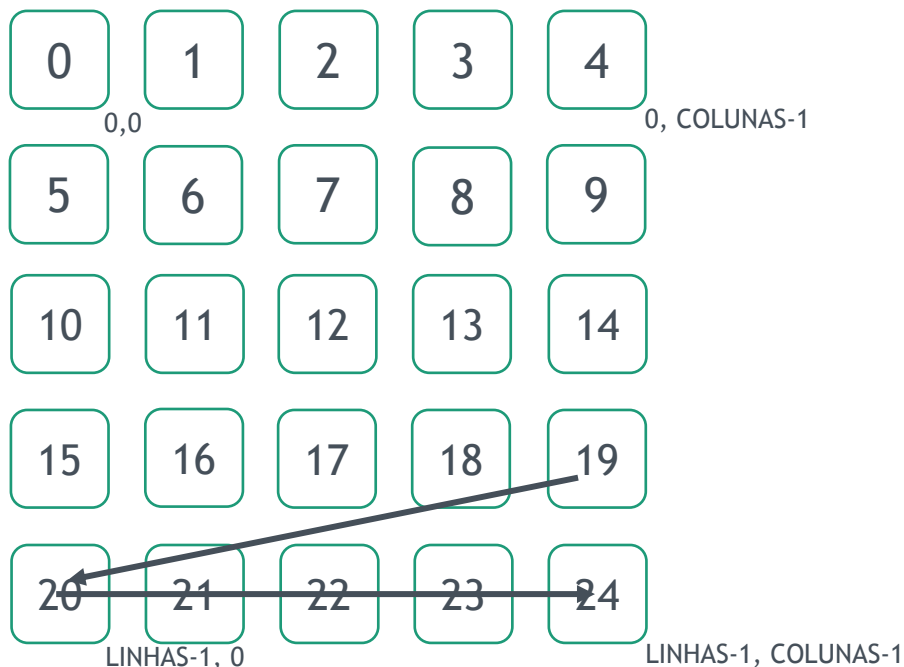


```
int cont=0, i, j, LINHAS=5, COLUNAS=5;  
int m[LINHAS][COLUNAS];  
for(i=0; i<LINHAS; i++){  
    for(j=0; j<COLUNAS; j++){  
        m[i][j] = cont;  
        cont++;  
    }  
}
```

Vai preencher a
ultima linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!



```
int cont=0, i, j, LINHAS=5, COLUNAS=5;
int m[LINHAS][COLUNAS];
for(i=0; i<LINHAS; i++){
    for(j=0; j<COLUNAS; j++){
        m[i][j] = cont;
        cont++;
    }
}
```

Preenche a
ultima linha!

Matrizes e Estruturas de Repetição

A ordem de preenchimento importa!

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

0,0

0, COLUNAS-1

LINHAS-1, 0

LINHAS-1, COLUNAS-1

```
int cont=0, i, j, LINHAS=5, COLUNAS=5;
int m[LINHAS][COLUNAS];
for(i=0; i<LINHAS; i++){
    for(j=0; j<COLUNAS; j++){
        m[i][j] = cont;
        cont++;
    }
}
```

**MATRIZ
PREENCHIDA!!**

MUITO
OBRIGADO

Prof. André del Mestre

www.ifsul.edu.br
almmartins@charqueadas.ifsul.edu.br