Alocação Dinâmica

Profa. Elloá B. Guedes

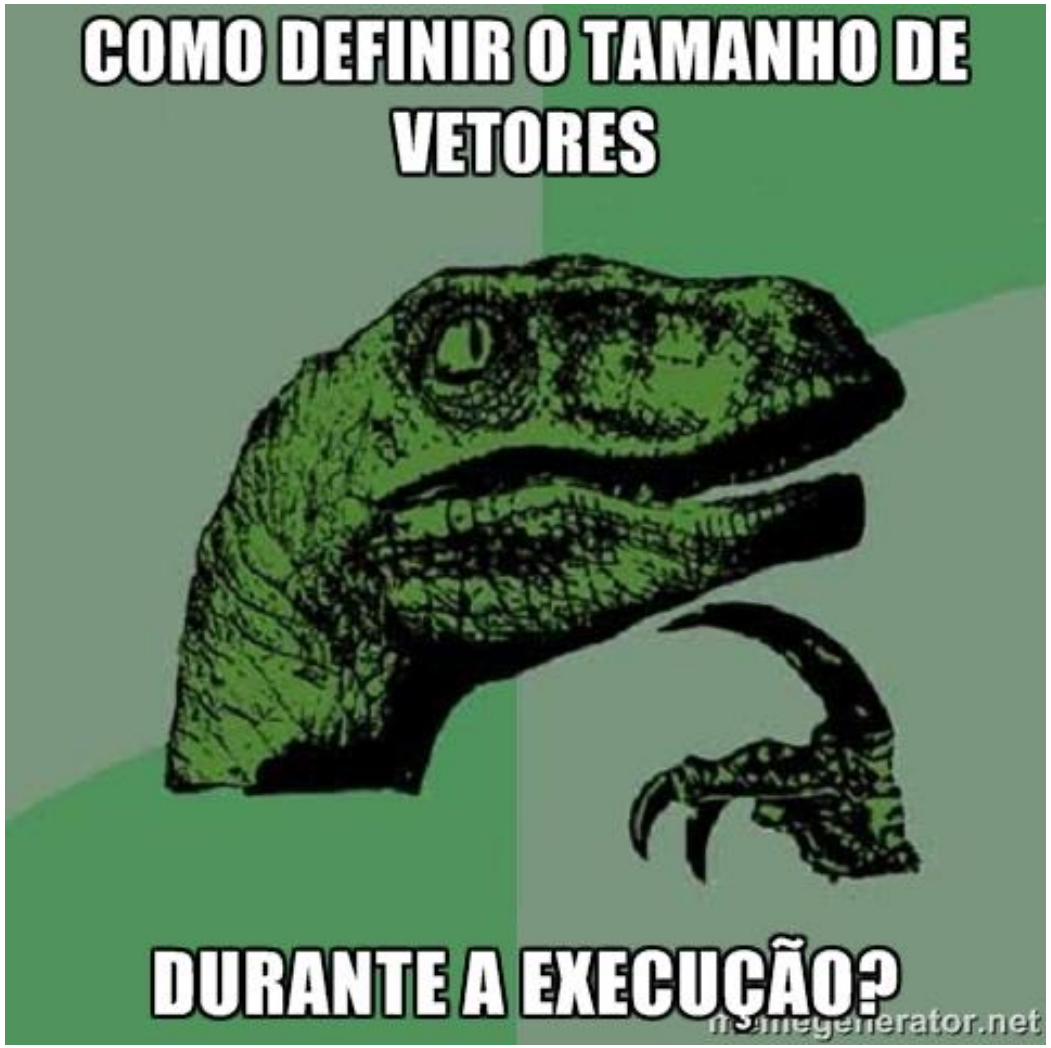
www.elloaguedes.com

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

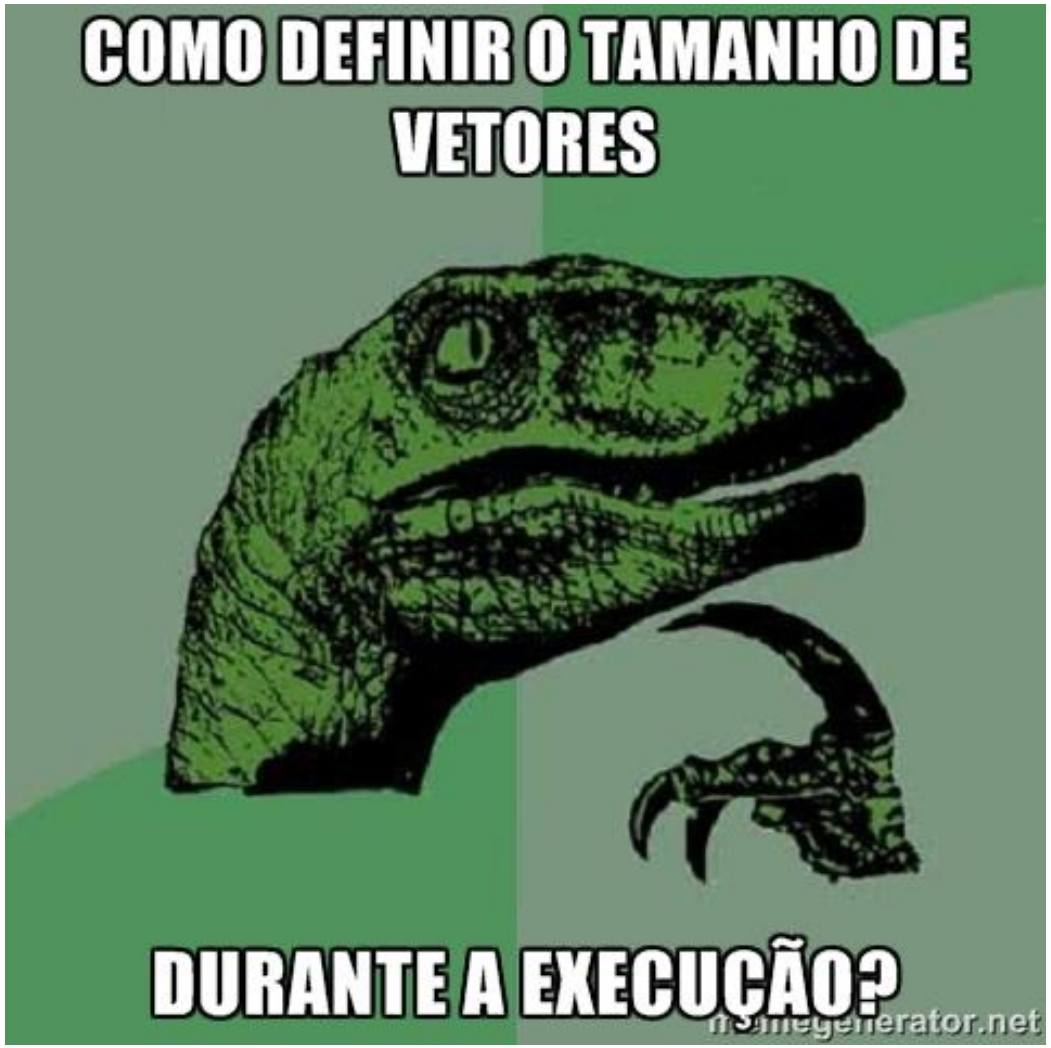
Problema

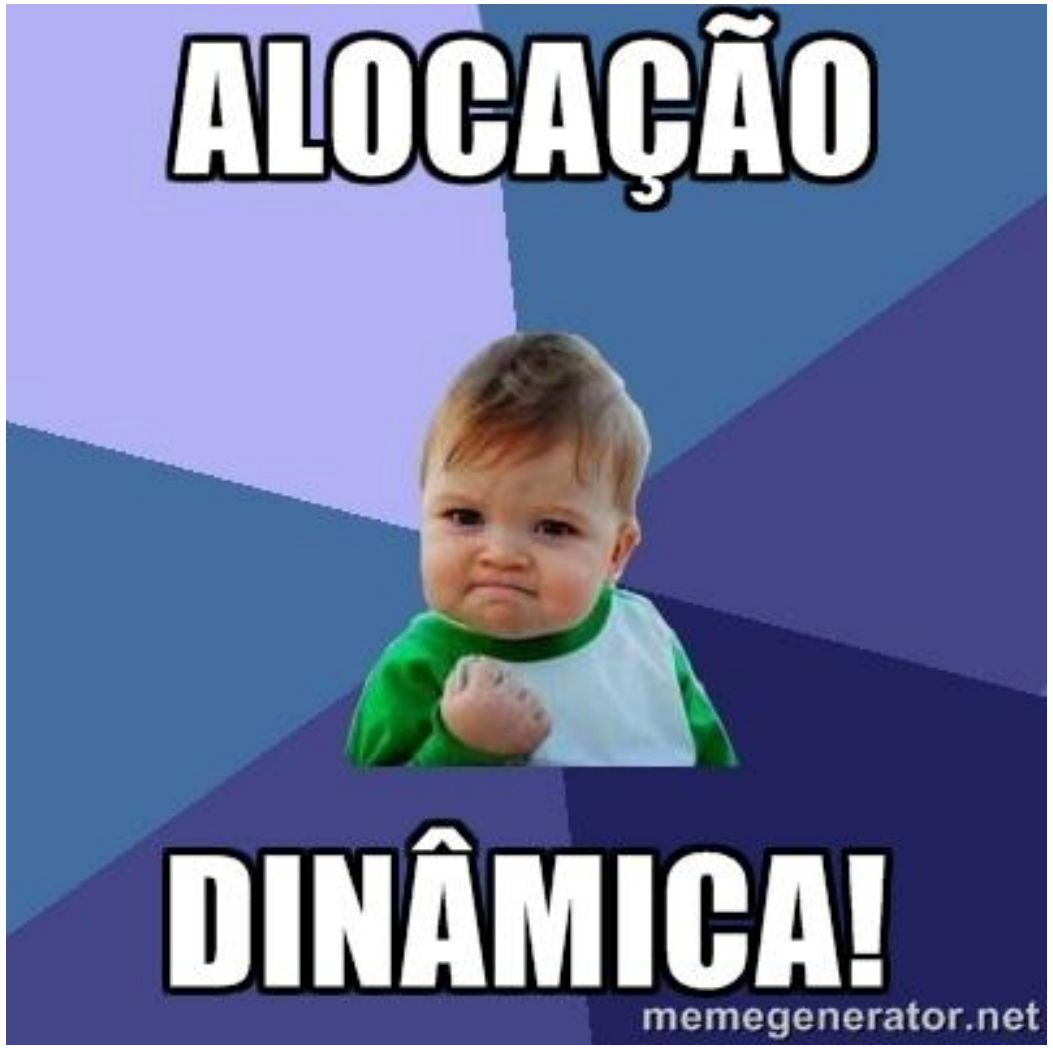
- Sempre que criávamos vetores ou strings era necessário saber o tamanho de antemão!

Questão da Aula de Hoje!



Questão da Aula de Hoje!





Alocação Dinâmica de Memória

- Uma maneira de solicitar memória durante a execução do programa
  - Dinâmica = durante a execução
  - Estática = durante a compilação

- Utilização da biblioteca <stdlib.h>

Função malloc de <stdlib.h>

- Aloca um conjunto de bytes indicado pelo programador, devolvendo um ponteiro para o bloco de bytes criados ou NULL, quando a alocação falha

- Protótipo:

```
void *malloc(qtde_de_bytes)
```

Função malloc de <stdlib.h>

- Retorno: ponteiro para void

- Significa um ponteiro para qualquer tipo de dado
  - Dê um cast para o tipo de ponteiro que você quiser

Exemplo

- Usuário vai informar uma quantidade de números a ser lida
- Aloque dinamicamente um vetor que comporte estes números
- Calcule a média e imprima os números ao final

Exemplo

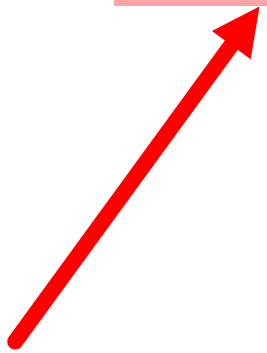
```c
#include <stdio.h>
#include <stdlib.h>

int main(){

    int tam;
    int *nums;
    int media = 0;
    printf("Informe a quantidade dos numeros a serem lidos: ");
    scanf("%d",&tam);
```

Exemplo

```
nums = (int*)malloc(tam*sizeof(int));
```

Exemplo

```
nums = (int*)malloc(tam*sizeof(int));
```



Tamanho do tipo de dados que será alocado

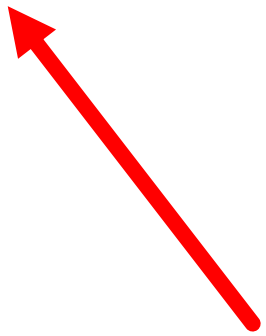
Exemplo

```
nums = (int*)malloc(tam*sizeof(int));
```

Quantidade de dados do tipo a serem alocados

Exemplo

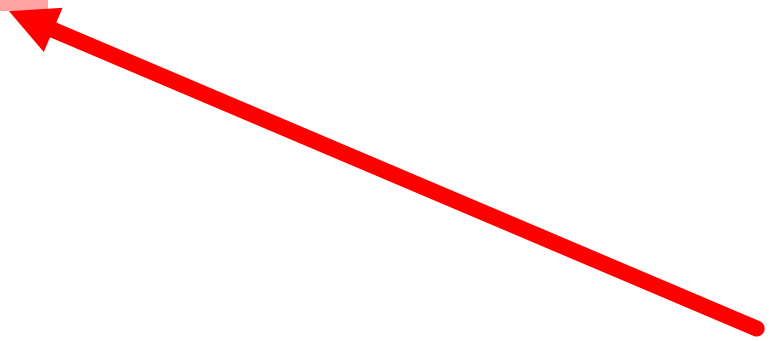
```
nums = (int*)malloc(tam*sizeof(int));
```



Cast para ponteiro do tipo de dados que estou trabalhando

Exemplo

```
nums = (int*)malloc(tam*sizeof(int));
```



Ponteiro do tipo de dados que estou trabalhando
Depois dessa parte, posso trabalhar como um vetor
Tamanho é conhecido (tam)

Exemplo

```
if (nums == NULL) {
    printf("Alocacao dinamica falhou!");
    return 1;
} else {
    for (int i = 0; i < tam; i++){
        printf("\nnum[%d] =",i);
        scanf("%d",&nums[i]);
        media += nums[i];
    }
    media /= tam;
    printf("\nA media eh: %d",media);

    printf("\nImpressao do vetor: ");
    for (int i = 0; i < tam; i++){
        printf("\nnum[%d] = %d",i,nums[i]);
    }
```

Exemplo

```
if (nums == NULL) {
    printf("Alocacao dinamica falhou!");
    return 1;
} else {
    for (int i = 0; i < tam; i++){
        printf("\nnum[%d] =",i);
        scanf("%d",&nums[i]);
        media += nums[i];
    }
    media /= tam;
    printf("\nA media eh: %d",media);

    printf("\nImpressao do vetor: ");
    for (int i = 0; i < tam; i++){
        printf("\nnum[%d] = %d",i,nums[i]);
    }
```

Função free de <stdlib.h>

- Libera a memória alocada dinamicamente após a sua execução

- Protótipo:

```
void free(void *ptr)
```

Exemplo

```
} else {
    for (int i = 0; i < tam; i++){
        printf("\nnum[%d] =",i);
        scanf("%d",&nums[i]);
        media += nums[i];
    }
    media /= tam;
    printf("\nA media eh: %f",media);

    printf("\nImpressao do vetor: ");
    for (int i = 0; i < tam; i++){
        printf("\nnum[%d] = %d",i,nums[i]);
    }

    free(nums);
```

Exercício

- Faça um programa que lê um valor do usuário e cria uma string do tamanho informado.
- Em seguida, leia a string e imprima na tela

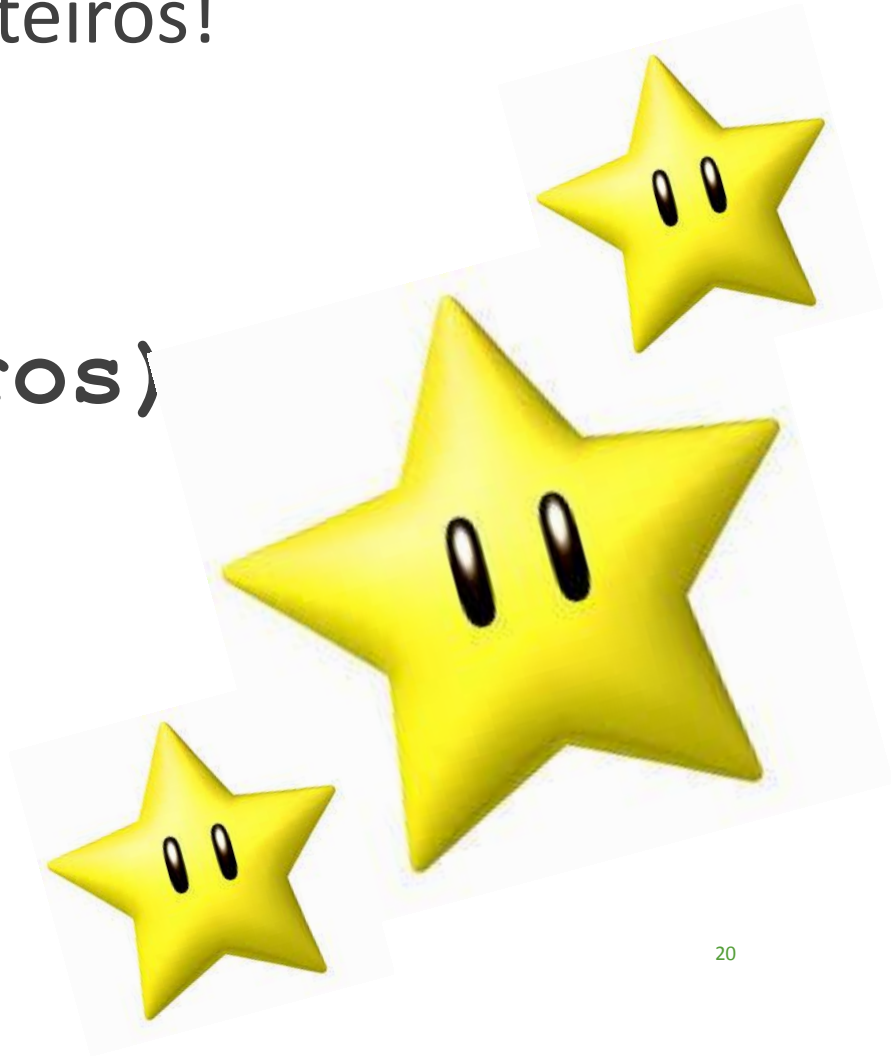
- Lembrete: é preciso guardar o \0

Um adendo – Funções que retornam ponteiros

- Em C, é possível que funções retornem ponteiros!

- Protótipo:

```
tipo *nomedafuncao(parametros)
```



Exercício

- Faça uma função em C que receba uma string e retorne um novo ponteiro com uma cópia do conteúdo desta string.

Função calloc de <stdlib.h>

- Aloca um conjunto de bytes indicado pelo programador, devolvendo um ponteiro para o bloco de bytes criados ou NULL, quando a alocação falha

- Inicializa todos os valores alocados para **ZERO**

- Protótipo:

```
void *calloc(qtde_de_bytes)
```

Função realloc de <string.h>

- Permite alterar o número de bytes alocados anteriormente

- Protótipo:

```
void *realloc(void *ptr,novo_tamanho);
```

Exemplo

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

int main(){

    char *string;
    int tam;

    printf("Informe a quantidade de caracteres a serem lidos: ");
    scanf("%d",&tam);
    tam++;
    string = (char*)malloc((tam)*sizeof(char));
```

Exemplo

```
if (string == NULL) {
    printf("Deu errado!");
} else {
    printf("Informe a string: ");
    fflush(stdin);
    fgets(string,tam,stdin);
    puts(string);

    printf("Informe a NOVA quantidade de caracteres a serem lidos: ");
    scanf("%d",&tam);
    tam++;
```

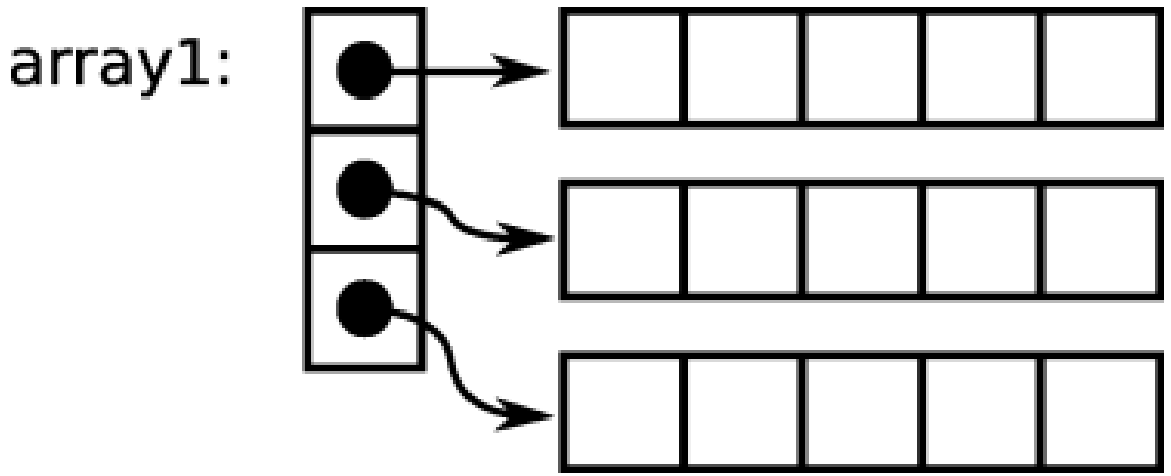
Exemplo

```
    realloc(string,tam*sizeof(char));

    if (string != NULL) {
            printf("Informe a string: ");
            fflush(stdin);
            fgets(string,tam,stdin);
            puts(string);
    }

}

return 0;
```

Alocação Dinâmica de Matrizes

- Uma matriz é uma vetor de ponteiros que apontam para ponteiros

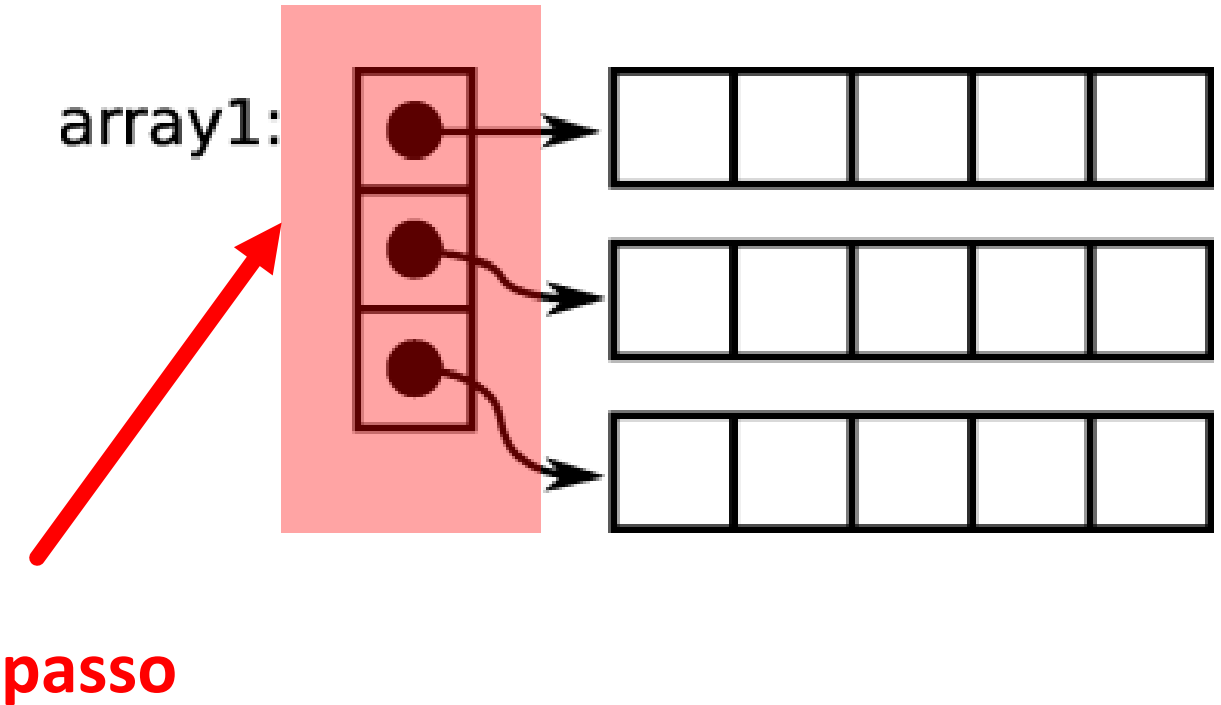
- 1º Passo:
  - Alocar ponteiros para ponteiros
- 2º Passo:
  - Alocar espaço para os ponteiros



Alocação Dinâmica de Matrizes

- Uma matriz é uma vetor de ponteiros que apontam para ponteiros

- 1º Passo:
  - Alocar ponteiros para ponteiros
- 2º Passo:
  - Alocar espaço para os ponteiros

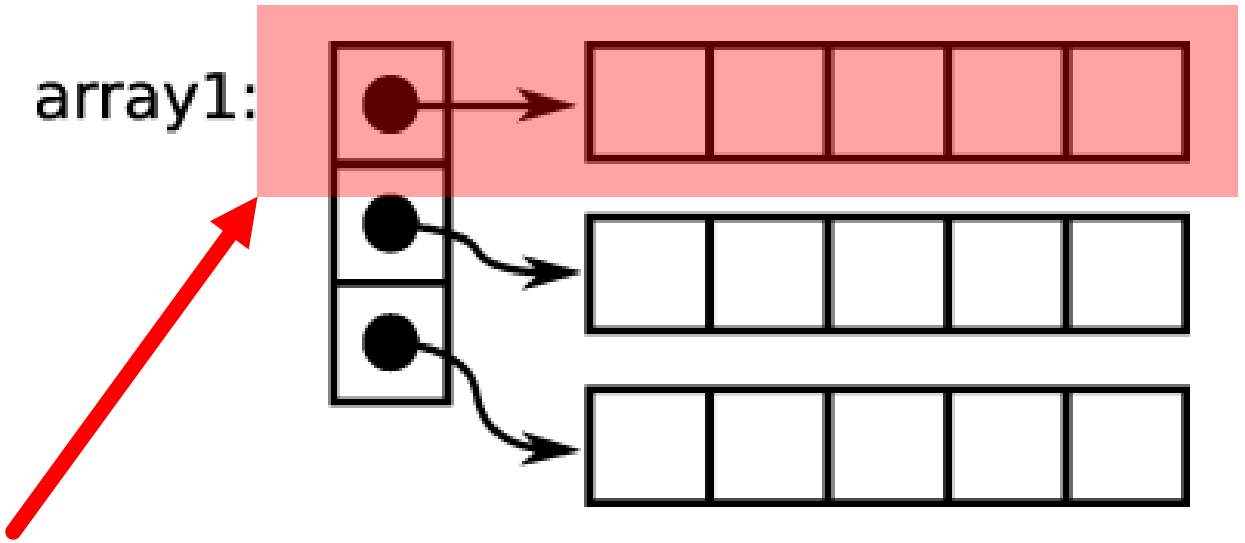


1º. passo

Alocação Dinâmica de Matrizes

- Uma matriz é uma vetor de ponteiros que apontam para ponteiros

- 1º Passo:
  - Alocar ponteiros para ponteiros
- 2º Passo:
  - Alocar espaço para os ponteiros



2º. passo

Exemplo

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

int main(){

  int ordem;
  int** matriz;

  scanf("%d",&ordem);
  matriz = (int**)malloc(ordem*sizeof(int*));

  for ( int i = 0; i < ordem; i++ ) {
    matriz[i] = (int*)malloc(ordem*sizeof(int));
  }

  return 0;


}
```

Exercício

- Faça um programa que leia dois valores inteiros do usuário referentes ao número de linhas e de colunas de uma matriz

- Aloque dinamicamente uma matriz de inteiros com esta dimensão

- Leia a matriz do teclado
- Imprima a matriz

Solução

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

int main(){

  int linha, coluna;
  int **matriz;
  scanf("%d %d",&linha,&coluna);

  matriz = (int**)malloc(linha*sizeof(int*));

  if (matriz != NULL) {
    for ( int i = 0; i < linha; i++ ) {
        matriz[i] = (int*)malloc(coluna*sizeof(int));
        if (matriz[i]== NULL){
            return 1;
        }
    }
```

Solução

```
} else {
  return 1;
}

// Leitura
  for (int i =0; i < linha; i++)  {
      for (int j = 0; j < coluna; j++){
          printf("\nmatriz[%d][%d] = ",i,j);
      }
  }

//Impressao
  for (int i =0; i < linha; i++)  {
      for (int j = 0; j < coluna; j++){
          printf("%3d", matriz[i][j]);
      }
      printf("\n");
  }

  free(matriz);
```

```
int __cdecl printf (const char * __restrict__ _Format, ..
```