



**Estácio**

# **ARA0098 - ESTRUTURA DE DADOS EM C**

---

PROFESSOR: LUCAS SAMPAIO LEITE

# Exercícios para casa...

---

1. Faça um programa, com uma função que necessite de três argumentos, e que forneça a soma desses três argumentos. Utilize passagem por referência.
2. Faça um programa, com uma função que necessite de um argumento. A função retorna o valor de caractere 'P', se seu argumento for positivo, e 'N', se seu argumento for zero ou negativo.
3. Faça um programa para imprimir de acordo com a imagem abaixo para um  $n$  informado pelo usuário. Use uma função que receba um valor  $n$  inteiro e imprima até a  $n$ -ésima linha.



O diagrama mostra a saída esperada para um valor de  $n$  informado pelo usuário. A saída é uma pirâmide de números, onde cada linha contém uma sequência de números consecutivos, começando com 1 na primeira linha e terminando com  $n$  na  $n$ -ésima linha. O exemplo abaixo assume  $n=5$ .

```
1
2 2
3 3 3
...
n n n ... n
```

# Desafio...

---

- ❑ Leia um número decimal (até 3 dígitos) e escreva o seu equivalente em numeração romana. Utilize funções para obter cada dígito do número decimal e para a transformação de numeração decimal para romana ( 1 = I, 5 = V, 10 = X, 50 = L, 100 = C, 500 = D, 1.000 = M; e utilize um vetor guardando a tradução para cada um dos dígitos).



# Motivação...

---

```
C alocacao_dinamica.c > ...  
1  #include <stdio.h>  
2  #include <stdlib.h>  
3  int main(){  
4  
5      int x, y;  
6      float c;  
7      char nome[50];  
8  
9      return 0;  
10 }
```

# Motivação...

---

- ❑ Precisamos desenvolver um programa que processe os valores dos salários de uma determinada empresa.
- ❑ Uma solução possível e simples seria armazenar em um vetor grande.

```
C alocacao_dinamica.c > ...  
1  ✓ #include <stdio.h>  
2  #include <stdlib.h>  
3  int main(){  
4  
5      float salarios[1000];  
6  
7      return 0;  
8  }
```

# Motivação...

---

- ❑ Precisamos desenvolver um programa que processe os valores dos salários de uma determinada empresa.
- ❑ Uma solução possível e simples seria armazenar em um vetor grande.

```
C alocacao_dinamica.c > ...  
1  #include <stdio.h>  
2  #include <stdlib.h>  
3  int main(){  
4  
5      float salarios[1000];  
6  
7      return 0;  
8  }
```

❑ Qual a eficiência disso????

# Sabemos que....

---

- ☐ Os vetores são agrupamentos sequenciais de dados de um mesmo tipo de memória.
- ☐ Um ponteiro guarda o endereço de um dado na memória.
- ☐ O nome de um vetor é um ponteiro para o primeiro elemento do array.

# Sabemos que....

---

- ☐ Os vetores são agrupamentos sequenciais de dados de um mesmo tipo de memória.
- ☐ Um ponteiro guarda o endereço de um dado na memória.
- ☐ O nome de um vetor é um ponteiro para o primeiro elemento do array.

☐ Solução: solicitar um bloco de memória e colocar a sua primeira posição em um ponteiro.



# Alocação dinâmica de memória

- ❑ Solução: solicitar um bloco de memória e colocar a sua primeira posição em um ponteiro.

Endereço Conteúdo Nome

0x1000		
0x1004		
0x1008	0x0000	a
0x1012		
0x1016		
0x1020		
0x1024		
0x1028		
0x1032		
0x1036		
0x1040		
0x1044		
0x1048		
0x1052		
0x1056		
0x1060		



Endereço Conteúdo Nome

0x1000		
0x1004		
<del>0x1008</del>	0x1028	a
0x1012		
0x1016		
0x1020		
0x1024		
0x1028		Vetor dinâmico  a
0x1032		
0x1036		
0x1040		
0x1044		
0x1048		
0x1052		
0x1056		
0x1060		

# Alocação dinâmica de memória

---

- ❑ A linguagem C usa 4 funções para alocação dinâmica, disponíveis na biblioteca `<stdlib.h>`:
  - ❑ `malloc`;
  - ❑ `calloc`;
  - ❑ `realloc`;
  - ❑ `free`.
- ❑ Existe também o operador `sizeof`.

# Alocação dinâmica de memória

---

- ❑ Alocar memória do tipo int é diferente de alocar memória do tipo char;
- ❑ Tipos diferentes podem ter tamanhos diferentes na memória.

Tipo	nº de bytes
char	1 byte
int	4 bytes
float	4 bytes
double	8 bytes

# Alocação dinâmica de memória

---

```
C alocacao_dinamica.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct ponto{
5      int xAxis ,yAxis;
6  };
7
8  int main(){
9
10     printf("char: %ld bytes \n", sizeof(char));
11     printf("int: %ld bytes \n", sizeof(int));
12     printf("float: %ld bytes \n", sizeof(float));
13     printf("double: %ld bytes \n", sizeof(double));
14     printf("struct ponto: %ld bytes \n", sizeof(struct ponto));
15
16     return 0;
17 }
```

# Alocação dinâmica de memória

```
C alocacao_dinamica.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct ponto{
5      int xAxis ,yAxis;
6  };
7
8  int main(){
9
10     printf("char: %ld bytes \n", sizeof(char));
11     printf("int: %ld bytes \n", sizeof(int));
12     printf("float: %ld bytes \n", sizeof(float));
13     printf("double: %ld bytes \n", sizeof(double));
14     printf("struct ponto: %ld bytes \n", sizeof(struct ponto));
15
16     return 0;
17 }
```



```
char: 1 bytes
int: 4 bytes
float: 4 bytes
double: 8 bytes
struct ponto: 8 bytes
```

# Alocação dinâmica de memória

---

- ❑ O operador `sizeof()` retorna o número de bytes necessários para alocar um único elemento de um determinado tipo.
- ❑ Forma geral:
  - ❑ `sizeof(nome_do_tipo)`.

```
int x = sizeof(int);  
printf("x = %d bytes \n", x);
```



```
x = 4 bytes
```

# Alocação dinâmica de memória

---

- ❑ Comando malloc:

- ❑ Faz parte da biblioteca <stdlib.h>.

- ❑ Aloca dinamicamente um bloco consecutivo de bytes na memória e retorna o endereço deste bloco.

```
void* malloc(unsigned int num);
```

- ❑ Isto permite escrever programas mais flexíveis.

- ❑ Exemplo de uso: alocar um vetor de tamanho definido pelo usuário...

# Alocação dinâmica de memória

---

- ❑ A função `malloc()` recebe por parâmetro a quantidade de bytes a ser alocada e retorna:
  - ❑ Ponteiro para a primeira posição do array.
  - ❑ `NULL`: em caso de erro;

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

int main(){

    int *i = malloc(200); //cria um vetor de 50 inteiros (200 bytes)

    char *c = malloc(200); //cria um vetor de 200 char (200 bytes)

    return 0;
}
```



# Alocação dinâmica de memória

---

- ❑ Na alocação dinâmica de memória, deve-se levar em conta o tamanho do tipo.

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

int main(){

    int *i = (int*) malloc(200);
    char *c = (char*) malloc(50);

    int *i = (int*) malloc(50 * sizeof(int)); //cria um vetor de 50 inteiros
    char *c = (char*) malloc(50 * sizeof(int)); //cria um vetor de 50 char

    return 0;
}
```

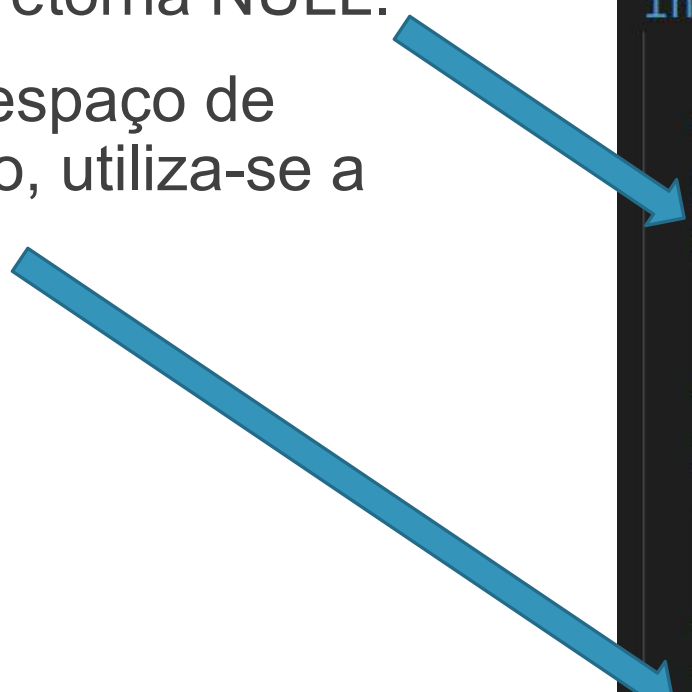
# Alocação dinâmica de memória

- ❑ Se não houver memória suficiente para alocação, a função malloc() retorna NULL.
- ❑ Para liberar o espaço de memória alocado, utiliza-se a função free().

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

int main(){

    int *p;
    p = (int*) malloc(5 * sizeof(int));
    if(p == NULL){
        printf("Erro: memória insuficiente! \n");
        exit(1);
    }
    for (int i=0; i<5; i++){
        printf("Digite p[%d]: ", i);
        scanf("%d", &p[i]);
    }
    free(p);
    return 0;
}
```



# Alocação dinâmica de memória

---

- ❑ A função `calloc()` também serve para alocar memória durante a execução do programa.
- ❑ Ela faz o pedido de memória ao computador e retorna um ponteiro com o endereço do início do espaço de memória alocado.

```
void* calloc (unsigned int num, unsigned int size);
```

# Alocação dinâmica de memória

---

- ❑ A função `calloc()` recebe por parâmetro o número de elementos no vetor a ser alocado e o tamanho de cada elemento do vetor e retorna:
  - ❑ Ponteiro para a primeira posição do array.
  - ❑ `NULL`: em caso de erro;

```
int *i = (int*) calloc(50, 4);  
char *c = (char*) calloc(50, 1);
```

```
int *i = (int*) calloc(50, sizeof(int)); //cria um vetor de 50 inteiros  
char *c = (char*) calloc(50, sizeof(int)); //cria um vetor de 50 char
```

# Alocação dinâmica de memória

---

- ☐ Vamos exercitar?
  - ☐ Crie um vetor dinâmico para armazenar números inteiros de 5 posições utilizando malloc e calloc.
  - ☐ Imprima os vetores.

Qual a diferença???

# Alocação dinâmica de memória

---

- ☐ Vamos exercitar?
  - ☐ Crie um vetor dinâmico para armazenar números inteiros de 5 posições utilizando malloc e calloc.
  - ☐ Imprima os vetores.

Qual a diferença???

malloc() apenas aloca a memória.  
calloc() aloca a memória e zera os bits

# Exercícios

---

1. Crie um programa que:
  - a) Aloque dinamicamente um vetor de 5 números inteiros;
  - b) Peça para o usuario digitar os 5 números no espaço alocado;
  - c) Mostre na tela os 5 números armazenados;
  - d) Libere a memória alocada.
2. Faça um programa que leia do usuário o tamanho de um vetor a ser lido e faça a alocação dinâmica de memória. Em seguida, leia do usuário seus valores e imprima o vetor lido.

# Exercícios

---

3. Faça um programa para armazenar em memória um vetor de dados contendo 1500 valores do tipo int, usando a função de alocação dinâmica de memória CALLOC:
  - a) Faça um loop e verifique se o vetor contém realmente os 1500 valores inicializados com zero (conte os 1500 zeros do vetor).
  - b) Atribua para cada elemento do vetor o valor do seu índice junto a este vetor.
  - c) Exibir na tela os 10 primeiros e os 10 últimos elementos do vetor.



# Dúvidas???

---

