# PROGRAMAÇÃO ESTRUTURADA

Vetores

- São agregados de dados homogêneos
- Possuem tamanho fixo
- Os elementos estão armazenados em posições contíguas da memória
- Considerada a estrutura de dados mais simples
- Cada elemento do vetor pode ser acessado individualmente, especificando a sua posição.

Declaração em C:

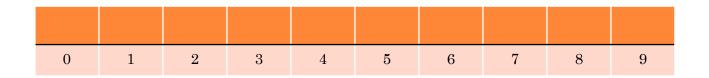
```
tipo nome vetor[tamanho];
```

Exemplo:

```
int vetNum[10];
```

- Vetor de 10 posições que armazena um número inteiro
- O primeiro elemento do vetor fica na posição 0
- O último elemento do vetor na posição 9 (tamanho-1)

```
int vetNum[10];
```



```
vetNum[0] = 11;

vetNum[5] = -2

scanf("%d", &vetNum[8]) \longrightarrow 18
```

```
int vetNum[10];
```

11				-2		18	
0	1	2					

```
vetNum[0] = 11;

vetNum[5] = -2

scanf("%d", &vetNum[8]) \longrightarrow 18
```

#### EXEMPLO - VETORES

inteiros em um vetor
e depois calcular a
sua média

lores

inteiro: ");

Armazenar 10 valores

```
int main (void) {
    int i, soma=0, vetor[10];
    float media = 0;
    // lendo e armazenando os valores
    for (i=0; i<10; i++) {
        printf("Digite um número inteiro: ");
        scanf("%d", &vetor[i]);
    // calculando a média dos números do vetor
    for (i=0; i<10; i++)
        soma += vetor[i];
   media = soma/10.0;
   printf("A média é: %.2f", media);
    return 0;
```

- Tamanho do Vetor
  - Como o tamanho do vetor é fixo, é possível usar uma constante para facilitar as alterações no código
  - Quando não sabemos o tamanho do vetor, podemos declarar um vetor com um tamanho bem grande → desperdício de espaço
  - Em versões mais novas dos compiladores C, é possível perguntar ao usuário o tamanho do vetor e depois fazer a declaração
  - Também veremos no decorrer do curso → alocação dinâmica

#### EXEMPLO - VETORES

## inteiros em um vetor e depois calcular a

Armazenar 10 valores

```
sua média
#define TAM 10
int main (void) {
    int i, soma=0, vetor[TAM];
    float media = 0;
    // lendo e armazenando os valores
    for (i=0; i<TAM; i++) {
        printf("Digite um número inteiro: ");
        scanf("%d", &vetor[i]);
    // calculando a média dos números do vetor
    for (i=0; i<TAM; i++)
        soma += vetor[i];
    media = soma/(TAM *1.0);
    printf("A média é: %.2f", media);
    return 0;
```

#### EXEMPLO - VETORES

```
int main (void) {
    int i, soma=0, tam;
    float media = 0:
    // descobrindo o tamanho do vetor
    printf("Digite o tamanho do vetor: ");
    scanf("%d", &tam);
    int vetor[tam];
    // lendo e armazenando os valores
    for (i=0; i < tam; i++) {
        printf("Digite um número inteiro: ");
        scanf("%d", &vetor[i]);
    // calculando a média dos números do vetor
    for (i=0; i<tam; i++)
        soma += vetor[i];
    media = soma/(tam * 1.0);
    printf("A média é: %.2f", media);
    return 0;
```

Armazenar 10 valores inteiros em um vetor e depois calcular a sua média

### VETORES COMO PARÂMETROS

- Ao se passar um vetor como parâmetro para uma função, passamos apenas o endereço da sua primeira posição
  - Devemos usar um tipo ponteiro
  - Ponteiro são variáveis capazes de armazenar endereços
  - Veremos o conceito de ponteiros em aulas futuras

#### EXEMPLO - VETORES

```
#define TAM 10

float calculaMedia(int *v) {
   int i, soma=0;
   for (i=0; i<TAM; i++)
       soma += v[i];

   return (soma/(TAM*1.0));
}</pre>
```

```
int main (void) {
   int i, vetor[TAM];
   float media;
  // lendo os valores do vetor
   for (i=0; i<TAM; i++) {
       printf("Digite um número: ");
       scanf("%d", &vetor[i]);
  media = calculaMedia(vetor);
  printf("A média é: %.2f", media);
  return 0;
```

#### Exercícios usando Vetores

- Ler 20 números inteiros e depois imprimí-los na ordem contrária em que foram lidos.
- Ler N notas dos alunos de uma turma e calcular a média. Ao final imprimir a quantidade de alunos que ficou acima da média calculada.
- 3. Preencher 2 vetores de X posições cada, com números inteiros. Fazer uma função que receba os dois vetores como parâmetros e que retorne a quantidade de posições que possuem números distintos.

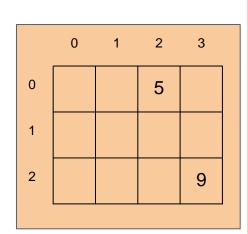
#### VETORES MULTIDIMENSIONAIS

- Os mais comuns são os vetores bidimensionais, como por exemplo as matrizes.
- Declaração:

```
tipo varVetor[dim1][dim2]...[dimN];
```

Exemplo Matriz:

```
int matNum[3][4];
matNum[0][2] = 5;
scanf("%d", &matNum[2][3]);
matNum[0,2] = 5; → errado!
```



### VETORES COMO PARÂMETROS

#### Exemplos:

```
func (int vetor[10]);
func (int vetor[]);
func (int *vetor)
func (int matriz[10][5]);
func (int matriz[][5]);
func (int (*matriz)[5]);
```

### EXERCÍCIOS

- 1. Faça um programa que preencha uma matriz 3x4 com números inteiros.
  - a. Faça uma função para preencher a matriz;
  - b. Faça uma função que escreva a matriz formatada;
- Faça um programa que preencha uma matriz NxN e que escreva a soma dos elementos da diagonal principal.
  - a. Faça uma função para preencher a matriz;
  - b. Faça uma função que escreva a matriz formatada;
  - c. Faça uma função que retorne a soma dos elementos da diagonal principal.

#### EXERCÍCIOS

- 3. Crie uma matriz N x N que representará as amizades entre um grupo de pessoas. A matriz AMIZADE é preenchida, em cada posição com o número 0 ou 1. O número 1 em uma posição i,j indica que a pessoa i é amiga de j. Por outro lado, o número 0 indica que i e j não são amigos.
  - Faça um programa que preencha a matriz de amizades e depois verifique se existe alguma inconsistência, já que a relação de amizade é recíproca.
  - Faça uma função que retorne quantas inconsistências existem na matriz.

- São usados para armazenar uma cadeia de caracteres
- Exemplo:

```
char nome[40];
char endereco[50];
```

 Internamente as cadeias de caracteres terminam com '\0', para que os programas possam encontrar o fim de uma cadeia

- Não é possível a atribuição direta de uma cadeia de caracteres a um vetor do tipo char.
- Exemplo:

```
char palavra[10];
palavra = "ARCO";
```

Só é possível atribuir posições:

```
palavra[0] = 'A';

palavra[1] = 'R';

palavra[4] = '\0';
```

- Pode ser usada a função strcpy que copia uma cadeia de caracteres para outra incluindo o '\0'.
- Exemplo:

```
strcpy(nome1, nome2);
```

copia o conteúdo de nome 2 para nome 1

```
strcpy(nome, "Aline");
```

equivale a nome = "Aline";

- Existem outras funções que podem ser usadas para manipulação de cadeias de caracteres
- Exemplos:

```
strncpy(s,ct,n)
```

→ copia no máximo n caracteres da cadeia ct para s.
Retorna s.

```
strcat(s,ct)
```

→ concatena a cadeia ct ao final da cadeia s. Retorna s.

```
strlen(ct)
```

→ retorna o tamanho de ct

#### Exemplos:

```
strncat(s,ct,n)
```

→ concatena no máximo n caracteres da cadeia ct ao final da cadeia s. Retorna s.

```
strcmp(cs,ct)
```

→ compara a cadeia cs com ct.

Retorna  $< 0 \rightarrow se cs < ct$ 

Retorna =  $0 \rightarrow \text{se cs} = \text{ct}$ 

Retorna >  $0 \rightarrow \text{se cs} > \text{ct}$ 

Estas funções são oferecidas pela biblioteca <string.h>

```
#include <string.h>
```

#### EXERCÍCIOS

- 4. Fazer um programa que leia códigos e preços dos produtos de uma loja e que escreva ao final o código do produto mais caro.
  - Considere que o código é uma cadeia de 3 caracteres.
- Faça um programa que leia uma cadeia de caracteres em formato de data: DD/MM/AAAA
  - Crie uma função que receba esta data como parâmetro e que retorne o ano em uma variável inteira.
  - Crie um a função que receba o ano como parâmetro e retorne se a data caiu em um ano bissexto.