

# **UNIVERSIDADE DE BRASÍLIA**

## **Introdução à Ciência da Computação**

### **Disciplina: 113913**

---



**Prof:** **Luiz Augusto F. Laranjeira**  
**[luiz.laranjeira@gmail.com](mailto:luiz.laranjeira@gmail.com)**

---

**Universidade de Brasília – UnB**  
**Campus Gama**

---



---

# **11. VETORES**

---



- Um **vetor** é um conjunto unidimensional de posições de armazenamento de dados do mesmo tipo. Cada posição de armazenamento é chamado de elemento do vetor.
- No conteúdo deste material vamos utilizar o termo **vetor**, como a tradução do termo original em inglês **array** (unidimensional).
- Na declaração de um vetor escreve-se o tipo, seguido do nome (identificador) e o tamanho do vetor (número de elementos que ele contém) entre os colchetes ([ ]).
- **<tipo de dado> <identificador> [<quantidade de elementos>]**
- **Exemplo:** float nota[5];





- 
- Os termos em português **vetor** e **matriz** são ambos utilizados como tradução do termo original em inglês **array**. No contexto do nosso curso um outro elemento da definição de um array servirá como diferenciador entre um vetor e uma matriz.
  - Este outro elemento é o número de dimensões do array. A dimensão de um array é um assunto que será abordado mais à frente nesta aula, mas por enquanto:
  - O termo vetor referencia sempre um array com uma única dimensão (unidimensional);
  - O termo matriz referencia um array com duas ou mais dimensões.
  - Exemplos: float nota[5]; // vetor  
int tabuleiro [8][8]; // matriz



```
float  nota[5];           // vetor  
int   tabuleiro [8][8];  // matriz
```



- **No primeiro exemplo, declara-se um vetor (array unidimensional) de 5 números reais com o nome de **nota**. O compilador reservará um espaço em memória contígua suficiente para conter estes 5 elementos. Cada elemento será referenciado pelo nome do vetor e um índice. Ex: **notas[4]**;**
- **No segundo exemplo, declara-se uma matriz (array bidimensional) de 64 números inteiros com o nome de **tabuleiro**. O compilador reservará um espaço em memória contígua suficiente para conter estes 64 elementos. Cada elemento será referenciado pelo nome da matriz e dois índices. Ex: **tabuleiro[3][7]**;**
- **A memória contígua significa que os dados serão armazenados em seqüência contínua (um seguido do outro).**



- As **declarações de vetores ou matrizes** diferenciam-se da declaração de variáveis comuns pelo uso de conjuntos de colchetes ('[ ]') preenchidos com números inteiros.
- Cada número entre colchetes indica o tamanho do vetor ou o tamanho de uma das dimensões da matriz.
- Os valores dentro dos colchetes são diferentes de acordo com o contexto que estão sendo empregados. O primeiro é a declaração das variáveis, que indica o tamanho do vetor. O segundo é a referência de um elemento específico dentro do vetor.



int nota[5];

≠

nota[3]

NOTA

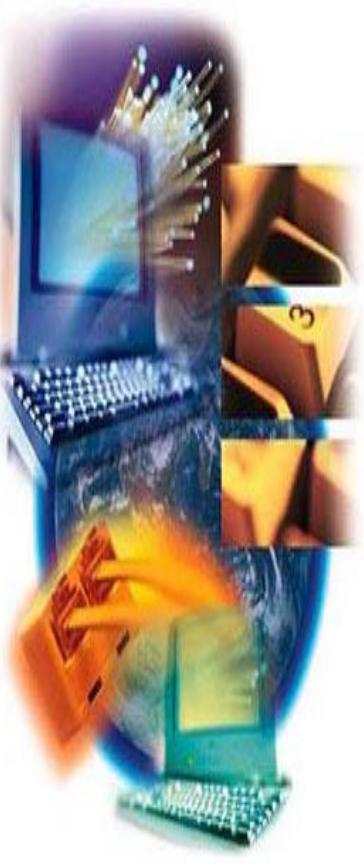
7	5	3	8	9
0	1	2	3	4

Cria um vetor  
de cinco  
Elementos.

Aponta para o elemento  
três que é o quarto do  
vetor.



- A declaração e a referência a um elemento demonstram situações diferentes, pois a declaração apresenta a quantidade de elementos que um vetor terá, enquanto que a referência apresenta o número do índice do vetor que se deseja o elemento



float nota[5]; → declaração de vetor de cinco valores

nota[3] → referência ao elemento 3 que é o quarto elemento do vetor

- A solicitação de gravação de um elemento, além da quantidade declarada, pode provocar consequências inesperadas, pois o compilador vai calcular a posição de memória onde este elemento será armazenado e vai gravá-lo. No caso do exemplo anterior, suponha que seja referenciado em um programa o sexto valor do vetor (nota[5]) e a declaração seja mantida como sendo de cinco elementos.



- O compilador multiplica o deslocamento (5) pelo tamanho de cada elemento (4 bytes no exemplo). Então ele passa por todos os bytes desde o início do vetor e grava o novo valor na posição calculada.
- O compilador não verifica o tamanho máximo do vetor e faz a gravação em memória baseado em seus cálculos, não importando sobre o que ele vai estar gravando o novo valor. Este novo valor será sobreposto ao valor que ocupava anteriormente aquela(s) posição(ões) de memória. As consequências são imprevisíveis caso aquela(s) posição(ões) de memória guardasse(m) algum outro dado utilizado pelo programa ou sistema operacional.



- A inicialização de um vetor de tipos fundamentais (inteiros ou caracteres), pode ser feita na sua declaração.

```
int vetor[5] = {10,2,3,40,50};
```

- 
- A omissão do tamanho do vetor ocasionará a criação de um array de tamanho suficiente para conter os elementos inicializados que foram atribuídos a ele.

```
int vetor[ ] = {10,2,3,40,50};
```

- Esta declaração criará um vetor igual ao anterior. Não se pode inicializar mais elementos do que os que foram declarados no vetor. Pode-se descobrir o número de elementos de um vetor usando a expressão:

**const int numelementos = sizeof(vetor) / sizeof(vetor[0]);**

- Os elementos não inicializados explicitamente em um vetor são inicializados com o valor 0.

```
int vetor[5] = {10,20};
```



## 1. Vetores

- 
- **Tipo de dado composto usado para representar uma coleção de variáveis de um mesmo tipo**
  - **Estrutura de dados homogênea unidimensional**
  - **Ex: Ler a nota de 3 alunos e calcular a média**

```
int nota0, nota1, nota2;
```

```
printf("entre com a 1a. nota");
```

```
scanf("%d", &nota0);
```

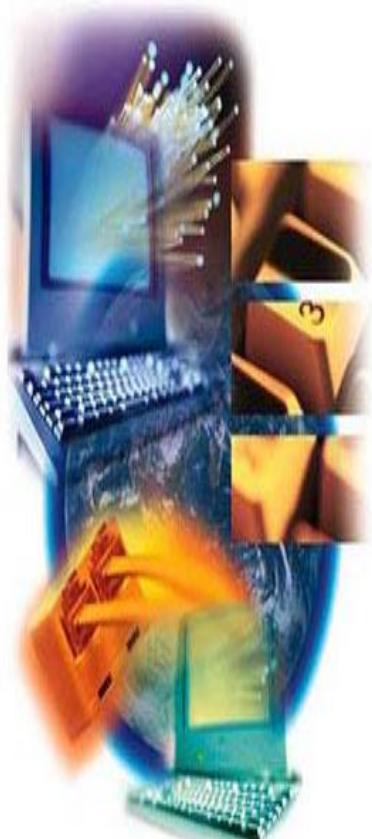
```
: : :
```

```
printf("média = %d", (nota0 + nota1 + nota2) / 3);
```



## 1. Vetores

- Ex: Calcular a média de 300 alunos



```
#include<stdio.h>
#define N_ALUNOS 300
int main()
{   int      i;
    float    notas [ N_ALUNOS ], media = 0;

    for ( i = 0; i < N_ALUNOS; i++ ) {
        printf ("entre com a nota %d", i+1);
        scanf ("%f", &notas[ i ]);
        media += notas [ i ];
    }
    printf (" Média = %f \n", media / N_ALUNOS);

    for ( i = 0; i < N_ALUNOS; i++ ) {
        printf ("\n A nota do %d aluno= ", i+1);
        printf ("%f \n", notas[ i ]);
    }
}
```



## 1. Vetores

- 
- An abstract collage of various computer-related images, including a monitor displaying code, a keyboard, circuit boards, and colorful geometric shapes, set against a dark background.
- Em ‘C’ não existe declaração de vetor dinâmico.
  - O tamanho de um vetor tem que ser determinado em tempo de compilação.

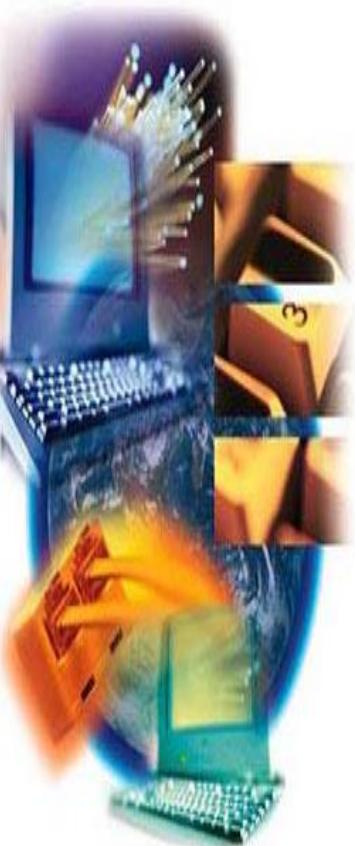
- Ex:

```
int alunos;  
int notas [ alunos ];  
: : :  
printf ("entre com o número de alunos");  
scanf ("%d", &alunos);
```

**NÃO É ACEITO !!!**



## 1. Vetores



**Solução:**

- **Declarar um vetor que suporte um número máximo de elementos:**

Ex:      int alunos;  
              int notas [ 70 ];  
              :    :    :

printf (“entre com o número de alunos”);  
scanf (“%d”, &alunos);

**ACEITO ...**



## **1. Vetores**

- **C não realiza verificação de limites em vetores**
- **Nada impede o acesso além do fim do vetor**
- **Faça sempre que necessário a verificação dos limites**





## 1. Vetores

An abstract collage of various computer-related images, including a monitor displaying code, a keyboard, circuit boards, and colorful geometric shapes, all set against a dark background.

```
#include <stdio.h>
#define TAMANHO 100
int main( )
{
    int      quantidade, media = 0;
    float    notas [ TAMANHO ];
    // quantidade deve ser ≤ TAMANHO
    printf ( "quantas notas devo ler ?" );
    scanf("%d", &quantidade);
    for ( i = 0; i < quantidade; i++ ) {
        printf ( "entre com a nota %d", i+1 );
        scanf("%d", &notas [ i ]);
    }
    :
    :
    for ( i = 0; i < quantidade; i++ )
        media += notas [ i ];
    :
    :
}
```



## 1. Vetores

### Passando um vetor como parâmetro de uma função



```
#include <stdio.h>
int maximum( int [ ] );      /* ANSI function prototype */

int main( )
{
    int values[5], i, max;
    printf("Entre com 5 numeros:\n");
    for( i = 0; i < 5; ++i )
        scanf("%d", &values[i] );
    max = maximum( values );
    printf("\nValor Maximo = %d\n", max );
}
```



## 1. Vetores

An abstract collage on the left side of the slide, composed of various images related to computing and technology, including a monitor displaying code, a keyboard, circuit boards, and colorful geometric shapes.

```
int maximum( int values[5] )
{
    int max_value, i;
    max_value = values[0];
    for( i = 0; i < 5; ++i )
        if( values[i] > max_value )
            max_value = values[i];
    return max_value;
}
```

**Saida:**

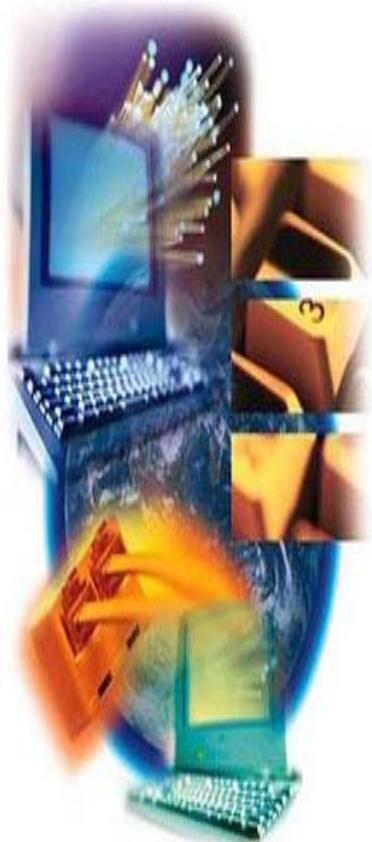
Entre com 5 numeros:

7 23 45 9 121

**Valor Maximo = 121**



## 1.A Exercício



**Escreva um programa em C que calcule o novo salário de um grupo de 100 funcionários. O usuário deverá entrar com o valor do salário atual e do número de filhos de cada funcionário. Após estes dados serem fornecidos para um funcionário o programa deverá escrever na tela o valor do novo salário do funcionário segundo a expressão:**

$$\text{NovoSalario} = \text{SalarioAtual} + 200 * \text{NumFilhos}$$



## 1.A Solução



```
#include <stdio.h>
int main () {
    float salario[100];
    int i;
    int nfilhos[100];

    for (i=0; i<100; i++) {
        printf("Digite o salário do funcionário %d\n",i+1);
        scanf("%f",&salario[i]);
        printf("Digite o número de filhos do funcionário %d\n",i+1);
        scanf("%d",&nfilhos[i]);
        salario[i]+= 200*nfilhos[i];
        printf("O novo salário do funcionário %d é de R$%.2f\n",i+1,
               salario[i]);
    }
    system("pause");
    return 0;
}
```



## **1.B Exercício**



**Escreva um novo programa em C que solucione o problema anterior. Só que agora, o número de funcionários é variável. (Aqui presume-se que haverá no máximo 100 funcionários).**



## 1.B Solução



```
#include <stdio.h>
int main () {
    float salario[100];
    int i, nrofunc;
    int nfilhos[100];

    printf("Digite o número de funcionários\n");
    scanf("%d",&nrofunc);
    for (i=0; i<nrofunc && i<100; i++) {
        printf("Digite o salário do funcionário %d\n",i+1);
        scanf("%f",&salario[i]);
        printf("Digite o número de filhos do funcionário %d\n",i+1);
        scanf("%d",&nfilhos[i]);
        salario[i]+= 200*nfilhos[i];
        printf("O novo salário do funcionário %d é de R$%.2f\n",i+1,
    salario[i]);
    }
    system("pause");
    return 0;
}
```



## **1.C Exercício**



**Escreva um algoritmo que leia N notas (N será informado pelo usuário e é sempre menor ou igual a 100), calcule a média das notas e imprima na tela todas as notas que ficaram abaixo dessa média.**



## 1.C Solução



```
#include <stdio.h>
int main () {
    float media, notas[100];
    int i, n;

    printf("Digite o número de alunos\n");
    scanf("%d",&n);
    media = 0.0;
    for (i=0; i<n; i++) {
        printf("Digite a %dª nota\n",i+1);      scanf("%f",&notas[i]);
        media+=notas[i];
    }
    media = (float) media/n;
    printf("A média das notas informadas é %.2f\n",media);
    for (i=0; i<n; i++) {
        if (notas[i]<media)
            printf("A nota %d é %.2f e está abaixo da média de %.2f\n",
i+1,
                notas[i], media);
    }
    system("pause");
    return 0;
}
```



## **1.D Exercício**

A vertical collage of images related to computing and technology. It includes a monitor displaying code, a keyboard, a hand interacting with a 3D cube, and various abstract digital patterns.

**Faça um algoritmo que leia um conjunto de N inteiros (N será lido e é sempre menor ou igual a 100), encontre e mostre o maior deles.**



## 1.D Solução

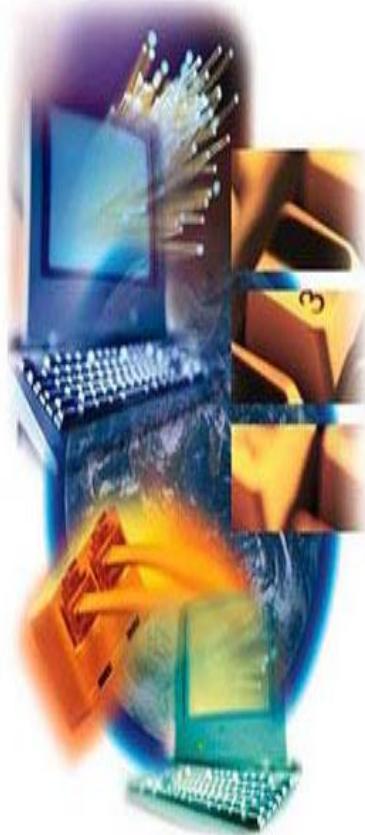


```
#include <stdio.h>
int main () {
    int i, n, maior, numeros[100];

    printf("Digite a quantia de números a serem lidos: \n");
    scanf("%d",&n);
    for (i=0; i<n; i++) {
        printf("Digite o %do número: \n",i+1);
        scanf("%d",&numeros[i]);
    }
    maior = numeros[0];
    for (i=0; i<n; i++) {
        if (numeros[i]>maior)
            maior = numeros[i];
    }
    printf("O maior número é %d\n ",maior);
    system("pause");
    return 0;
}
```



## **1.D Exercício -- CRÍTICA**



Note que no exemplo anterior o vetor é varrido totalmente por duas vezes, uma para ler os dados e outra para determinar o maior elemento. No caso de dois vetores grandes, isso pode aumentar o tempo de execução. Como poderíamos melhorar o algoritmo apresentado, de forma que se obtenha o mesmo resultado mas que varra o vetor totalmente apenas uma vez?

### **RESPOSTA:**

Determinando o maior número durante a leitura dos números. Assim, caso os números não fossem mais ser utilizados, poderíamos até mesmo nem armazenar os valores num vetor (mas aqui supomos que o programa pudesse ter outros usos para os valores, então armazenamos mesmo assim).



## 1.D Nova solução



```
#include <stdio.h>
int main () {
int main () {
    int i, n, maior, numeros[100];

    printf("Digite a quantia de números a serem lidos: \n");
    scanf("%d",&n);
    for (i=0; i<n; i++) {
        printf("Digite o valor do número %d: \n",i+1);
        scanf("%d",&numeros[i]);
        if (i==0)
            maior = numeros[0];
        else if (numeros[i]>maior)
            maior = numeros[i];
    }
    printf("O maior número é %d\n ",maior);
    system("pause");
    return 0;
}
```