 INSTITUTO FEDERAL GOIANO Campus Urutaí	INSTITUTO FEDERAL GOIANO – CÂMPUS URUTAI		
	CONTEÚDO VETOR – MÓDULO 2		
	Curso:	GTI	Turma: II
	Disciplina:	LTPI	Data: 30/11/2020
	Professora:	Vívian Cirino de Lima	

MATRIZ UNIDIMENSIONAL – VETOR (ARRAYS)

Observe o esquema abaixo:

1. Estruturas de controle

- 1.1. **Estrutura Sequencial** – os comandos são executados na sequência um após o outro, sem interrupção.
- 1.2. **Estruturas Condicionais, de Decisão ou de Seleção** – os comandos são executados de acordo com as situações definidas nas condições.
 - 1.2.1. **Estrutura Condicional Simples** – uma condição (uso do *if*)
 - 1.2.2. **Estrutura Condicional Composta** – existência de um bloco V (uso do *if*) e um bloco F (uso do *else*).
 - 1.2.3. **Estrutura Condicional Encadeada** – existência de um agrupamento de condições, enquanto for necessário.
 - 1.2.3.1. **Estrutura de Seleção de Múltipla Escolha** – Composta por uma série de estruturas de seleção simples encadeadas, em que observamos as seguintes prioridades:
 - Todas as condições nas decisões são de igualdade.
 - Todas as condições comparam uma mesma expressão a uma constante.
 - Todas as constantes consideradas são de tipo inteiro ou caractere.
- 1.3. **Estruturas de Repetição** – Serve para repetir a execução de um bloco de comandos.
 - 1.3.1. Repetição Contada – *for*
 - 1.3.2. Repetição com precondição (*while*)
 - 1.3.3. Repetição com poscondição (*do/while*)

2. Estruturas de Dados Compostas

- 2.1. Homogêneas – Vetor e Matriz

Em computação um **Vetor** (Array) ou **Arranjo** é o nome de uma matriz unidimensional considerada a mais simples das estruturas de dados. Geralmente é constituída por dados do mesmo tipo (homogêneos) e tamanho que são agrupados continuamente na memória e acessados por sua posição (índice - geralmente um número inteiro) dentro do vetor. Na sua inicialização determina-se o seu tamanho que geralmente não se modifica mesmo que utilizemos menos elementos. Na verdade, essa estrutura nada mais é do que uma variável que sob o mesmo nome consegue armazenar vários dados, diferenciando-os por meio de índices. O conceito é semelhante ao conceito das matrizes, em matemática.

Os tipos construídos são denominados de Estrutura de Dados Composta, que consiste em um conjunto de informações armazenados em um repositório. É dividida em duas classificações:

- Estruturas de Dados Compostas Homogêneas
- Estruturas de Dados Compostas Heterogêneas

Estrutura de Dados Compostas Homogêneas

É formado por variáveis que aceitam um número indeterminado de informações a partir de um mesmo tipo primitivo. Ele é dividido em duas partes:

- Estruturas de Dados Compostas Homogêneas Unidimensionais, conhecido como **Vetor**; e
- Estruturas de Dados Compostas Homogêneas Multidimensionais, conhecido como **Matriz**.

EXEMPLOS:

Vetor

V =

4	7	2	5	3
---	---	---	---	---

Matriz

M =

3	8	1	5
0	2	4	7
2	5	9	3

Cada elemento dos arrays podem ser referenciados através de índices. Exemplos:

V[0] = 4	M[0][0] = 3
V[1] = 7	M[1][2] = 4
V[4] = 3	M[2][0] = 2

VETORES

Vetores necessitam de apenas um índice para individualizar um elemento do conjunto. Possui diversas posições para armazenarem dados e isso deve ser explicitado na declaração do vetor.

Sintaxe de Declaração de Vetor

Tipo identificador[total de posições];
onde:
tipo é o tipo de dado (*int, float, etc.*).
identificador é o nome da variável que se deseja declarar;
total de posições *total de posições do* intervalo de variação do índice;

EXEMPLO:

Declarar uma variável composta de 8 elementos numéricos de nome NOTA.

```
float nota[8]; //vetor possui 8 posições (índice de 0 a 7)
```

fará com que passe a existir um conjunto de 8 elementos do tipo real, individualizáveis pelos índices 1, 2, 3,

..., 8.

nota[0]	nota[1]	nota[2]	nota[3]	nota[4]	nota[5]	nota[6]	nota[7]

Outros exemplos de declarações de vetores:

```
int idade[20];
float md[5];
char nome[3][30];
```

Para processarmos individualmente todos os componentes de um vetor, é aconselhável o uso da estrutura **for**, para que possamos variar o índice do vetor.

EXEMPLO

01. Fazer um programa que leia um vetor A contendo 30 números inteiros (com valores variando a partir de 1), calcule e exiba:

- a) o maior elemento;
- b) a posição (índice) do maior elemento.

```
#include <stdio.h>
int main()
{
    int a[30];
    int cont, pos, maior=0;
    for (cont=0; cont<30; cont++)
    {
        printf ("\n Digite valor do elemento %d\n", cont);
        scanf ("%d", &a[cont]);
        if (a[cont] > maior)
        {
            maior=a[cont];
            pos=cont+1;
        }
    }
    for (cont = 0; cont <30; cont++)
    {
        Printf ("\tElemento %d do vetor: %d", cont +1, a[cont]);
    }

    //exibição dos resultados
    printf ("\n Maior valor = %d se encontra na posicao %d", maior,pos);
    return(0);
}
```

Vetores

9.1 Armazenamento

Um **vetor** é um tipo de variável capaz de armazenar uma coleção de dados do mesmo tipo. Cada um dos dados armazenados num vetor, denominado **item**, é identificado por um número natural, a partir de 0, denominado **índice**.

Para indicar que uma variável é um vetor, devemos declará-la com o sufixo $[n]$, sendo n um valor inteiro positivo que estabelece o tamanho do vetor. Por exemplo, a declaração:

```
char v[3];
```

cria um vetor com três posições, cada uma delas capaz de armazenar um caractere. Na verdade, com esta única declaração, criamos as variáveis $v[0]$, $v[1]$ e $v[2]$, conforme ilustrado na Figura 9.1. Note que, como a indexação inicia-se em 0, o último item de um vetor de tamanho n é armazenado na posição $n-1$.

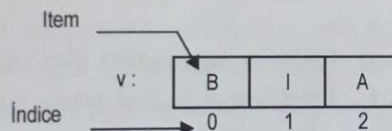


Figura 9.1 - Representação de um vetor na memória do computador.

O Problema 9.1 ilustra uma situação em que precisamos usar um vetor. O programa que resolve este problema é apresentado na Listagem 9.1 e sua tela de execução é apresentada na Figura 9.2.

Problema 9.1

Leia uma sequência de cinco números e exiba-a em ordem inversa.

```
1º número? 32 ↵
2º número? 76 ↵
3º número? 12 ↵
4º número? 99 ↵
5º número? 27 ↵
Ordem inversa: 27 99 12 76 32
```

Figura 9.2 - Tela de execução para um programa que resolve o Problema 9.1.

A lógica desse programa é simples. Basta ir lendo os números e guardando nas posições do vetor, da esquerda para a direita; em seguida, após o completo preenchimento do vetor, os itens são acessados da direita para a esquerda e exibidos.

```
/* inv5.c - exibe sequencia de 5 numeros em ordem inversa */  
  
#include <stdio.h>  
  
int main(void) {  
    int v[5], i;  
  
    for(i=0; i<5; i++) {  
        printf("%do. numero? ", i+1);  
        scanf("%d", &v[i]);  
    }  
  
    printf("\nOrdem inversa: ");  
  
    for(i=4; i>=0; i--)  
        printf("%d ", v[i]);  
  
    return 0;  
}
```

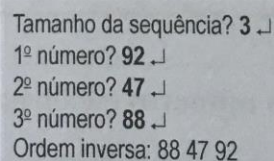
Listagem 9.1 - Programa que resolve o Problema 9.1.

9.2 Vetor com tamanho variável

De acordo com o padrão ISO, o tamanho de um vetor também pode ser indicado por uma variável. O Problema 9.2 ilustra uma situação em que isso é necessário. O programa que resolve este problema é apresentado na Listagem 9.2 e sua tela de execução é mostrada na Figura 9.3.

Problema 9.2

Leia uma sequência de n números e exiba-a em ordem inversa.



```
Tamanho da sequência? 3 ↵  
1º número? 92 ↵  
2º número? 47 ↵  
3º número? 88 ↵  
Ordem inversa: 88 47 92
```

Figura 9.3 - Tela de execução para um programa que resolve o Problema 9.2.


```

/* invn.c - exibe sequência de n numeros em ordem inversa */
#include <stdio.h>

int main(void) {
    int n, i;

    printf("Tamanho da sequencia? ");
    scanf("%d", &n);

    int v[n]; // cria vetor com tamanho igual ao valor de n

    for(i=0; i<n; i++) {
        printf("%do. numero? ", i+1);
        scanf("%d", &v[i]);
    }

    printf("\nSequencia invertida: ");
    for(i=n-1; i>=0; i--)
        printf("%d ", v[i]);

    return 0;
}

```

Listagem 9.2 - Programa que resolve o Problema 9.2.

Referências

Pereira, Silvio do Lago. **Algoritmos e Lógica de Programação em C**: Uma abordagem didática. 1ª ed. São Paulo: Érica, 2010.