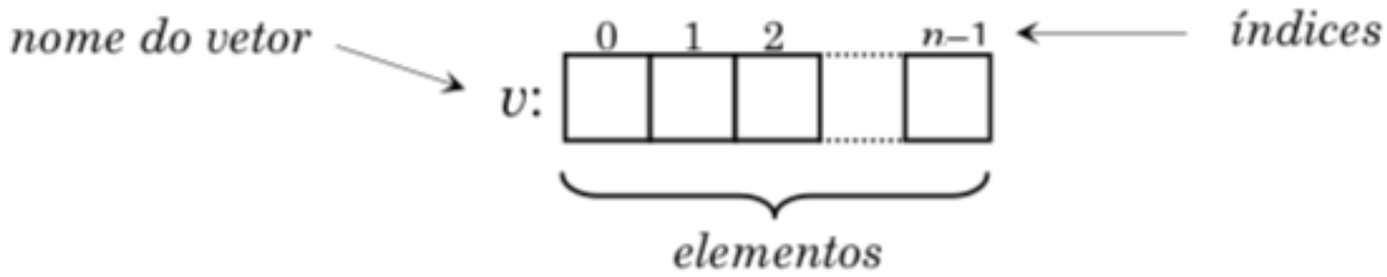
Laboratório de Programação - Linguagem C

Prof. Cezar Macegoza

Vetores

Um **vetor** é uma coleção de variáveis de um mesmo tipo, que compartilham o mesmo nome e que ocupam posições consecutivas de memória. Cada uma dessas variáveis denomina-se elemento e é identificada por um índice.



Em C, os vetores são sempre indexados a partir do zero, e portanto, o último elemento de um vetor de tamanho n ocupa a posição n-1 do vetor.

Vetores

Para criar um vetor, basta declarar uma variável com [n], sendo n o número de elementos a serem alocados no vetor.

```
int v[5];
```

Neste caso, temos um vetor de números interior com 5 espaços alocados para elementos inteiros;

Exercício

1. Crie um vetor para armazenar as temperaturas diárias desta semana e apresente na tela.

Exercício

2. Codifique um programa para solicitar $_5$ números, via teclado, e exibi-los na ordem inversa àquela em que foram fornecidos.

Vetores

Inicialização de vetores

Em C, vetores globais e estáticos são automaticamente zerados pelo compilador. Mas, se for desejado, podemos inicializá-los explicitamente no momento em os declaramos.

```
#include <stdio.h>
void main(void) {
   static float moeda[5] = {1.00, 0.50, 0.25, 0.10, 0.05};
   ...
}
```

Note que apenas expressões e valores constantes são permitidas numa lista de valores iniciais. O uso de variáveis causa erro de compilação.

Vetores

Se a lista de valores iniciais tem mais elementos que a capacidade do vetor, ocorre um erro de compilação. Entretanto, se tem menos que o necessário, as posições excedentes do vetor permanecem zeradas.

```
#include <stdio.h>
#define max 5
void main(void) {
   static int A[max] = {9, 3, 2, 7};
   auto int i;
   for(i=0; i<max; i++)
      printf("%d", A[i]);
}
```

Vetores

Vetores de tamanho implícito

```
#include <stdio.h>
int main() {
static char ds[] = {'D', 'S', 'T', 'Q', 'Q', 'S', 'S'};
 }
```

Como o tamanho é omitido, o compilador cria o vetor ds com 7 posições.

Exercício

3. Codifique um programa que indique a quantidade mínima de cédulas equivalente a uma dada quantia em dinheiro. Considere apenas valores inteiros e cédulas de 1, 5, 10, 50 e 100 reais.

Quantia? R$ 209
2 cédulas de R$100,00
1 cédula de R$5,00
4 cédulas de R$1,00

Endereços dos vetores

O formato %p é usado em C para a exibição de endereços. A importância desse fato é que, quando passamos um vetor como argumento a uma função, estamos na verdade passando o seu endereço. E, ao contrário do que ocorre com outros tipos, a passagem de vetores é feita por referência.

```
#include <stdio.h>

int main()

{

    int x[3], y[4];

    printf("x= %p e y= %p\n ", x, y);

}
```

Passagem de parâmetros tipo vetor

Em C, todos os parâmetros são passados por valor, exceto vetores, que são sempre passados por referência.

Exemplo para facilitar o entendimento:

Dadas as temperaturas registradas diariamente, durante uma semana, determine em quantos dias a temperatura esteve acima da média.

Passagem de parâmetros tipo vetor

```
#include <stdio.h>
int conta(float t[], float m)
{
    int i, c = 0;
    for (i = 0; i < 7; i++)
        if (t[i] > m)
            c++;
    return c;
}
float media(float t[])
{
    int i;
    float s = 0;
    for (i = 0; i < 7; i++)
        s += t[i];
    return s / 7;
}
```

Passagem de parâmetros tipo vetor

```
void obtem(float t[])
{
    int i;
    printf("Informe as temperaturas: ");
    for (i = 0; i < 7; i++)
    {
        printf("%d o valor? ", i + 1);
        scanf("%f", &t[i]);
    }
}
int main()
{
    float temp[7], m;
    obtem(temp);
    m = media(temp);
    printf("Estatística: %d", conta(temp, m));
}
```

Exercício

4. Usando as funções desenvolvidas, altere o programa apresentado de modo que sejam exibidos a temperatura média, a mínima, a máxima .

String

Em C, uma string é uma série de caracteres terminada com um caracter nulo, representado por '\0'. Como o vetor é um tipo de dados capaz de armazenar uma série de elementos do mesmo tipo e a string é uma série de caracteres, é bastante natural que ela possa ser representada por um vetor de caracteres. Essa representação possibilita que os caracteres que formam a string sejam acessados individualmente, o que proporciona grande flexibilidade na sua manipulação.

```
#include <stdio.h>
int main(void)
{
    printf("\nEspaço alocado = %lu bytes", sizeof("verde e amarelo"));
}
```

Inicialização de Strings

Como qualquer outro vetor, strings também podem ser inicializadas quando são declaradas. Nessa inicialização, podemos usar a sintaxe padrão, em que os caracteres são fornecidos entre chaves e separados por vírgulas, ou então podemos usar a sintaxe própria para strings, na qual os caracteres são fornecidos entre aspas.

Inicialização de Strings

A saída da string x certamente termina após a letra m ter sido exibida, pois o '\0' é encontrado.

```
#include <stdio.h>
int main()
{
    char x[] = "um";
    printf("Palavras\n");
    printf("%s\n",x);
}
```

Inicialização de Strings

Quanto à string y, entretanto, não há como saber quando a saída terminará pois, como não foi colocado o terminador, o compilador irá exibir todos os caracteres armazenados após o s, até que um '\0' seja encontrado.

```
#include <stdio.h>
int main()
{
    char y[] = {'d', 'o', 'i', 's'};
    printf("%s\n",y);
}
```

Manipulação de Strings

Como a string não é um tipo de dados básico da linguagem C, operações simples como atribuição e comparação não podem ser feitas diretamente com os operadores disponíveis.

```
#include <stdio.h>
int main()
{
    char x[] = "um";
    char y[] = "um";
    printf("%s == %s resulta em %s\n", x, y, x == y ? "verdade" : "falso");
}
```

Isso acontece porque na expressão x==y não esta- mos comparando o conteúdo dos vetores x e y, mas sim seus endereços que, obviamente, devem ser diferentes.

Manipulação de Strings

```
#include <stdio.h>
int strcmp(char s[], char t[])
{
    int i = 0;
    while (s[i] == t[i] && s[i] != '\0')
        i++;
    return s[i] - t[i];
}
int main()
{
    char x[] = "um";
    char y[] = "um";
    char z[] = "dois";
    printf("\n %s = %s = %s",x, y, strcmp(x,y)==0 ? " V " : " F " );
    printf("\n %s != %s = %s",x, y, strcmp(x,y)!=0 ? " V " : " F " );
    printf("\n %s < %s = %s",x, z, strcmp(x,z)<0 ? " V " : " F " );
    printf("\n %s > %s = %s",x, z, strcmp(x,z)>0 ? " V " : " F " );
    printf("\n %s <= %s = %s",z, y, strcmp(z,y)<=0 ? " V " : " F " );
    printf("\n %s => %s =%s",z, z, strcmp(z,z)>=0 ? " V " : " F " );

}
```

Exercício

5. Codifique a função strcat(s,t), que concatena a string t ao final da string s. Por exemplo, se x armazena "facil" e y armazena "idade", após a chamada strcat(x,y), x estará armazenando "facilidade".

Exercício

6. Codifique a função strlen(s), que devolve o número de caracteres armazenados na string s. Lembre-se de que o terminador '\0' não faz parte da string e, portanto, não deve ser contado.

Exercício

7. Codifique a função strpos(s,c), que devolve a posição da primeira ocorrência do caracter c na string s; ou –1, caso ele não ocorra em s.

Matriz

Uma matriz é uma estrutura de dados homogênea bidimensional, cujos elementos são distribuídos em linhas e colunas. Se A é uma matriz m×n, então suas linhas são indexadas de 0 a m–1 e suas colunas de 0 a n–1. Para acessar um particular elemento de A, escrevemos A[i][j], sendo i o número da linha e j o número da coluna que o elemento ocupa.

Matriz

```
#include <stdio.h>
int main()
{
    int i, j;
    for (i = 0; i < 3; i++)
    {
        printf("\n");
        for (j = 0; j < 4; j++)
            printf("[%d][%d] ", i, j);
    }
}
```

Exercício

8. Codifique um programa para ler uma matriz quadrada de ordem n e exibir apenas os elementos da diagonal principal.