Definição e exemplos iniciais

Uma matriz é uma série de variáveis do mesmo tipo referenciadas por um único identificador (nome), onde cada variável é diferenciada através de um número chamado “índice” que fica entre colchetes no fim do nome da variável (sem espaços).

Um vetor nada mais é do que uma matriz unidimensional (que contém apenas uma dimensão).

Por exemplo, uma *string* em C é um vetor (ou uma matriz unidimensional). Quando fazemos a declaração de uma *string* estamos na verdade declarando um vetor.

Veja os exemplos nos programas abaixo:

1) Saber qual foi a 6ª letra digitada numa *string* ou num vetor chamado **nome** de 40 posições:

```
#include <stdio.h>
int main()
{
   char nome[40];
   printf("Digite um nome: ");
   scanf("%s",&nome);
   printf("\nO caracter na posicao 6 do nome digitado é %c.",nome[5]);
 return 0;
}
```

Em destaque podemos ver primeiro a declaração do vetor nome, que abriga 40 posições do tipo caractere, que seqüenciados chamamos de *string*.

No segundo destaque, podemos ver como estamos buscando apenas a letra que ocupa a posição de número 6 dentro do vetor nome. Reparem na máscara do comando printf e scanf sublinhados no programa.

2) Imagine o seguinte problema: Calcular a média aritmética das notas de 5 alunos. Poderíamos resolver assim:

```
#include <stdio.h>
int main()
{
   float nota1,nota2,nota3,nota4,nota5,soma,media;
   printf("Digite a 1ª nota: ");
   scanf("%f",&nota1);
   printf("\nDigite a 2ª nota: ");
   scanf("%f",&nota2);
   printf("\nDigite a 3ª nota: ");
   scanf("%f",&nota3);
   printf("\nDigite a 4ª nota: ");
   scanf("%f",&nota4);
   printf("\nDigite a 5ª nota: ");
   scanf("%f",&nota5);
   soma=nota1+nota2+nota3+nota4+nota5;
   media=soma/5;
   printf("\nA media das notas e %3.2f.",media);
return 0;
```

```
}
```

Esse mesmo programa pode ser escrito da seguinte forma com a utilização de um vetor:

```
#include <stdio.h>
int main()
{
  float notas[5],soma,media;
  int cont;
  for(cont=0;cont<5;cont++)
  {
  printf("Digite a %dª nota: ",cont+1);
  scanf("%f",&notas[cont]);
  }
  soma=0;
  for(cont=0;cont<5;cont++)
       soma=soma+notas[cont];
  media=soma/5;
  printf("\nA media das notas e %3.2f.",media);
return 0;
}
```

A princípio, temos a impressão que o trabalho é o mesmo, porém imagine o mesmo problema, mas desta vez para calcular a média das notas de 2000 alunos. Seria um pouco extenso e trabalhoso se utilizarmos a primeira forma. Mas se utilizarmos vetores e matrizes, basta alterarmos os valores 5 para 2000 que o programa funciona sem nenhuma outra alteração.

Declaração

Como qualquer variável, um vetor ou matriz deve ser declarado no início do programa, como visto no exemplo anterior. Um vetor ou matriz pode assumir qualquer tipo de dados que podemos atribuir a qualquer variável. Veja alguns exemplos de declarações:

```
int sequencia[23];
float notas[45];
char letras[26];
```

Quando efetuamos uma declaração de um vetor ou matriz, o número entre colchetes mostra qual o valor máximo de posições que ela irá possuir. Desta forma, dentro do programa, poderemos variar de zero até o número explicitado na declaração do vetor para acessar ou atribuir dados a ele. Esse número pode ser substituído quando necessário por uma variável do tipo int (como na utilização de cont no exemplo acima).

Para acessarmos um “indivíduo” específico, então, podemos utilizar uma linha de comando assim: sequencia[7], ou notas[26], ou letras[19]. Com a utilização de uma variável inteira chamada i, por exemplo, podemos fazer assim: sequencia[i], letras[i] ou notas[i].

Lembrete importante: A linguagem C não verifica a dimensão da matriz/vetor em uma atribuição acima do seu valor limite declarado. Se você declarar um vetor inteiro de 10 posições (int vet[10]) e utilizar uma atribuição de um elemento acima do 10 (vet[30]=4;) você poderá estar atribuindo dados a outras regiões da memória não destinados à utilização do programa. Isto acarretará em resultados imprevisíveis do programa (como por exemplo até o travamento do computador e a perda do serviço não salvo). Você, como programador, tem a responsabilidade

de criar mecanismos para que isto não ocorra, providenciando sempre a checagem dos valores e laços antes da execução do programa.

Exercícios:

1) Faça um programa que possua um vetor de 10 números inteiros. Calcule a soma dos números no vetor, após terem sidos digitados por um usuário.

2) Faça um programa que tenha três vetores. Nos dois primeiros, leia 7 números reais e no outro calcule, nas mesmas posições de índices, a soma dos números do vetor 1 e 2. Exemplo:

    1. Vet1 : [2,3,0....]
    2. Vet2 : [9,-8,2,...]
    3. Vet3 : [11,-5,2,...]

3) Faça um programa que leia uma string de 10 caracteres. Depois da leitura, mostrar a string ao contrário, por exemplo

    1. String lida: Estudando!
    2. Resultado: !odnadutsE