Aula 15 – Variáveis Indexadas (vetores)

Além das variáveis normais já conhecidas, podemos ter também variáveis indexadas. Tais variáveis são referenciadas por um nome e um índice.
Especialmente úteis para armazenar seqüências de valores com as mesmas características.

Variáveis indexadas de um só índice são mais conhecidas como **vetores**. Variáveis indexadas com mais de um índice são conhecidas como **matrizes**.

Os elementos de um vetor estão localizados contiguamente na memória.

a[0]	a[1]	a[2]	. . .	a[9]

Declaração:

```
int a[10];         /*declara 10 variáveis a[0], a[1], . . ., a[9]*/
float b[100];      /*declara 100 variáveis b[0], b[1], . . ., b[99]*/
double c[345];     /*declara 345 variáveis c[0], c[1], . . . , c[344]*/
```

Declara-se sempre o tamanho máximo de elementos, mesmo que não sejam todos utilizados.

Uso – Como se fossem variáveis simples

```
/* atribuição */
a[1] = 0;
b[5] = b[2];
c[i] = c[j];

/* zerar o vetor a */
for (i=0; i < 10; i++) a[i] = 0;

/* zerar o vetor a (outra forma) */
i = 0;
while (i < 10) a[i++] = 0;

/* ler e imprimir os 100 elementos do vetor b */
for (i = 0; i < 100; i++) {
    scanf("%d", &b[i]);
    printf ("%d", b[i]);
}

/* contar quantos nulos tem nos primeiros 100 elementos de c */
cont = 0;
for (j = 0; j < 100; j++)
    if (c[i] == 0.0) cont++;

/* calcular o maior, o menor é a média dos n primeiros elementos de c */
max = min = soma = c[0];
for (i = 1; i < n; i++) {
     if (c[i] > max) max = c[i];
     if (c[i] < min) min = c[i];
     soma = soma + c[i];
}
printf ("máximo = %10.5lf – mínimo = %10.5lf – média = %10.5lf",
         max, min, soma/n);
```

```
/* inverter a ordem dos n primeiros elementos do vetor a, isto é:
   trocar a[0] com a[n-1], a[1] com a[n-2], etc . */
for (i = 0; i < n/2; i++) {
     aux = a[i];
     a[i] = a[n-1-i];
     a[n-1-i] = aux;
}
```

Observe na solução acima que se **n** for ímpar, o elemento médio fica intacto.

Tente agora resolver o seguinte problema sem usar vetores:

P53) Dado **n<=100** e uma sequência de **n** elementos, imprimir a sequência na ordem inversa à que foi lida.

O problema é que para imprimir a seqüência na ordem inversa, é necessário em primeiro lugar, armazenar todos os elementos na memória. Além disso, não sabemos quantos elementos tem a seqüência a priori, pois n também é um dado do problema. Se soubéssemos que a sequência teria exatamente 5 elementos, por exemplo, seria fácil resolver.

Com vetores fica fácil:

```
#include <stdio.h>
#include <stdlib.h>
#define nmax 100
/* dado n inteiro 0<n<=100, e um vetor de n elementos inteiros,
   imprimir o vetor na ordem inversa a que foi lido */
int main() {
 int vet[nmax],  /* vetor de nmax elementos */
     n,  /* número de elementos */
     i;

 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);

 /* ler o vetor de n elementos */
 for (i = 0; i < n; i++) {
        printf("\n - entre com o proximo elemento %5d:", i);
        scanf("%d", &vet[i]);
 }

 /* imprima os elementos na ordem inversa */
 printf("\n *** ordem inversa ***");
 for (i = n-1; i >= 0; i--)
       printf("\n * elemento %5d = %5d", i, vet[i]);
 system("pause"); return 0;
}
```

Sobre o tamanho do vetor

O vetor deve sempre ser declarado com uma quantidade fixa de elementos. Ocorre que muitas vezes usamos parte do vetor apenas. Não há problema nisso. Observe o programa acima. O **n** é no máximo 100. Portanto declaramos o vetor com 100 elementos. Entretanto, vamos usar apenas os primeiros **n** elementos do vetor. Se o n lido for 5, usaremos apenas 5 elementos e os demais 95 ficarão sem uso.

P54) Dado `n` inteiro, **`0<n<=100`**, e uma seqüência de `n` números entre `0` e `99`, determinar quantos estão entre `0` e `9`, entre `10` e `19`, ..., entre `90` e `99`.

Existem várias soluções para este problema. Vamos usar um vetor de contadores de frequências, `freq[0]` a `freq[9]`, que conterão a quantidade de elementos em cada intervalo. Note que dado um elemento `seq[i]` da seqüência, o contador que será incrementado devido a este elemento será `freq[seq[i]/10]`.

```
#include <stdio.h>
#include <stdlib.h>
#define nmax 100
int main() {
 int seq[nmax],/* vetor de nmax elementos */
     freq[10], /* freq[0] conterá a quantidade de elementos entre 0 e 9
                  freq[1] conterá a quantidade de elementos entre 10 e 19
                  assim por diante até
                  freq[9] que conterá a quantidade entre 90 e 99 */
           n,  /* numero de elementos */
           i;  /* contador */

 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);

 /* ler o vetor de n elementos */
 for (i = 0; i < n; i++)
       /* consistência - números entre 00 e 00 */
       while (1==1) /* esta comparação será sempre verdadeira */
          {printf("\n - entre com o %5d-esimo numero entre 00 e 99:", i);
           scanf("%d", &seq[i]);
           /* sai do while se foi digitado elemento entre 00 e 99 e
              volta para o for */
           if (seq[i] >= 0 && seq[i] <= 99) break;
           else printf("valor errado ####### digite novamente");
          }

 /* zerar o vetor de freqüências */
 for (i = 0; i < 10; i++) freq[i] = 0;

 /* incrementar a freqüência correspondente a cada valor */
 for (i = 0; i < n; i++) freq[seq[i]/10]++;

 /* imprima o vetor de freqüências */
 printf("\n *** vetor de frequências ***");

 for (i = 0; i < 10; i++)
    printf("\n * existem %5d elementos entre %5d e %5d", freq[i],
i*10,(i+1)*10 - 1);
 system("pause"); return 0;
}
```

Neste exemplo apareceu uma construção nova com o `while`. Quando colocamos **`while(1 == 1)`** estamos fazendo uma repetição infinita, pois a comparação será sempre verdadeira. Outra forma equivalente seria **`while(1)`**. Como veremos à frente, 0 está associado ao valor FALSO enquanto qualquer número diferente de zero está associado ao valor VERDADEIRO.

P55) Dado **n** inteiro, **0<n≤1000**, gerar uma seqüência de n números inteiros entre 0 e 999 usando a função rand(). Em seguida verificar quantos números estão entre 0 e 99, 100 e 199, ..., 900 e 999. Em seguida traçar um gráfico de freqüências da seguinte forma:

```
  0 -  99 20 * * * * * * * * * * * * * * * * * * * *
100 - 199  5 * * * * *
200 - 299 12 * * * * * * * * * * * *
300 - 399  3 * * *
:
:
:
900 - 999 10 * * * * * * * * * *
```

```
#include <stdio.h>
#include <stdlib.h>
#define nmax 1000

int main() {
 int seq[nmax],/* vetor de nmax elementos */
     freq[10], /* freq[0] conterá a quantidade de elementos entre 0 e 99
                  freq[1] conterá a quantidade de elementos entre 100 e 199.
                  Assim por diante até freq[9] que conterá a quantidade
                  de elementos entre 900 e 999 */
     n,       /* número de elementos */
     i,j;     /* contadores */

 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);

 /* Gerar n elementos entre 0 e 999.
    A função rand() gera um número aleatório entre 0 e 2^16-1.
    Portanto, basta pegar o resto do número gerado por 1000 */
 for (i = 0; i < n; i++) {
        seq[i] = rand()%1000;
        printf("%5d", seq[i]);
 }

 /* zerar o vetor de freqüências */
 for (i = 0; i < 10; i++) freq[i] = 0;

 /* incrementar a freqüência correspondente a cada valor */
 for (i = 0; i < n; i++) freq[seq[i]/100]++;

 /* imprima o gráfico de freqüências */
 printf("\n *** gráfico de frequencia ***\n");

 for (i = 0; i < 10; i++) {
       printf("\n %3d - %3d %3d ", i, (i+1)*100-1, freq[i]);
       /* imprima o número de asteriscos necessário */
       for (j = 0; j < freq[i]; j++) printf("* ");
 }
 system("pause"); return 0;
}
```

P56) Dado n inteiro, $0 < n \leq 100$ e uma seqüência com n elementos, contar quantas vezes cada elemento ocorre, isto é, para cada um dos n elementos da seqüência calcular quantas vezes ele aparece repetido na seqüência.

```
#include <stdio.h>
#include <stdlib.h>
#define nmax 100

int main() {
 int seq[nmax],  /* vetor de nmax elementos */
     n,  /* numero de elementos */
     i,j,cont;  /* contador */

 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);

 /* ler o vetor de n elementos */
 for (i = 0; i < n; i++) {
      printf("\n - entre com o %5d-esimo numero:", i);
      scanf("%d", &seq[i]);
 }

 /* para cada elemento seq[i] verificar quantas vezes ele ocorre */
 printf("\n\n******* numero de vezes que cada elemento ocorre *******");
 for (i = 0; i < n; i++) {
        cont = 0;
        for (j = 0; j < n; j++)
        if (seq[i] == seq[j]) cont++;
        printf("\n *elemento %5d = %5d * ocorre %5d vezes", i, seq[i],
        cont);
 }
 system("pause"); return 0;
}
```

Na solução acima, se um elemento aparece mais de uma vez, ele será impresso mais de uma vez também.

No problema a seguir fazer a mesma coisa, só que se o elemento já foi impresso com o número de vezes que ele ocorre não mostrar mais de uma vez.

P57) Idem sem repetição

```
#include <stdio.h>
#include <stdlib.h>
#define nmax 100
int main() {
 int seq[nmax],  /* vetor de nmax elementos */
     n,  /* numero de elementos */
     i,j,cont;  /* contador */

 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);

 /* ler o vetor de n elementos */
 for (i = 0; i < n; i++) {
        printf("\n - entre com o %5d-esimo numero:", i);
```

```
        scanf("%d", &seq[i]);
 }

 /* para cada elemento verificar quantas vezes ele ocorre na sequência */
 printf("\n\n******* numero de vezes que cada elemento ocorre *******");
 for (i = 0; i < n; i++) {
        /* Verifica se já foi contado. Para isso, basta verificar se é
           igual a algum elemento anterior ao i */
        for (j = i - 1; j >= 0; j--)
             if (seq[i] == seq[j]) break; /* já foi contado */
        /* só contar se ainda não foi contado */
        if (j < 0) {
          cont = 0;
          /* Basta contar a partir do i, pois antes não tem */
          for (j = i; j < n; j++)
          if (seq[i] == seq[j]) cont++;
          printf("\n *valor %5d * ocorre %5d vezes", seq[i], cont);
        }
 }
 system("pause"); return 0;
}
```

P58) Dado inteiro `n`, `0<n≤1000` e uma seqüência de `n` números, determinar o elemento mais próximo da média.
Para resolver esse é necessário primeiro calcular a média e em seguida, verificar qual elemento cujo valor absoluto de sua diferença com a média é mínimo.

P59) Dados `n,m` inteiros, `0<n,m ≤100` e dois vetores, **a** com **n** elementos e **b** com **m** elementos,construir um terceiro vetor contendo os elementos que estão em **a** e **b** ao mesmo tempo, ou seja, a intersecção de **a** com **b**.

```
#include <stdio.h>
#include <stdlib.h>
#define nmmax 100
int main() {
 int a[nmmax],b[nmmax],c[nmmax];  /* vetores dados de n e m elementos */
 int n, m, k, /* numero de elementos dos vetores a b c */
     i,j;     /* contadores */

 /* ler n e m com consistência */
 while (0 == 0) {  /* repita sempre */
    printf("\ndigite os valores de n e m separados por branco(s):");
    scanf("%d%d", &n, &m);
    if ((n > 0 && n <= nmmax) && (m > 0 && m <= nmmax)) break;
    else printf("\n***n ou m digitados errados");
 }

 /* Um conjunto não deve ter elementos repetidos.
    Vamos deixar essa consistência como exercício. Para isso, a cada
    elemento lido deve-se verificar se já foi lido anterior igual. */

 /* ler o vetor a de n elementos */
 printf("\n--------- digitacao do conjunto a\n");
 for (i = 0; i < n; i++) {
      printf("\n - entre com o %5d-esimo numero:", i);
      scanf("%d", &a[i]);
```

```
 }

/* ler o vetor b de m elementos */
 printf("\n--------- digitacao do conjunto b\n");
 for (i = 0; i < m; i++) {
      printf("\n - entre com o %5d-esimo numero:", i);
      scanf("%d", &b[i]);
 }

 /* construir o vetor c com a intersecção de a com b */
 printf("\n\n******* interseccao de a com b");
 k = 0; /* k é o número de elementos de c */
 for (i = 0; i < n; i++)
    /* procurar a[i] no vetor b */
    for (j = 0; j < m; j++)
       if (a[i] == b[j]) {
           /* encontrou então pode colocar em c e sair */
           c[k++] = a[i]; break; /* sai do for mais interno */
       }

 /* imprimir o vetor c */
 for (i = 0; i < k; i++)
       printf("\n elemento %5d = %5d", i, c[i]);
 system("pause"); return 0;
}
```

P60) Idem com os elementos que estão em **a** ou **b**, ou seja a união de **a** com **b**.

P61) Idem com os elementos que estão em **a** mas não estão em **b**, ou seja a diferença de **a** com **b**.

Atribuição de Valores aos Elementos do Vetor na Declaração

Assim como no caso das variáveis simples, pode-se atribuir valores a vetores na declaração.

```
int a[10] = {0,1,2,3,4,5,6,7,8,9};
double b[5] = {0.1, 0.2, 0.3, 0.4};
char c[26] = {"abcdefghijklmnopqrstuvwxyz"};
char d[10] = {48, 49, 50, 51, 52, 53, 54, 55, 56, 57};
```