Aula 16 – Funções e Vetores

Até agora, todos os parâmetros que usamos nas funções eram variáveis simples. Vetores podem também ser parâmetros de funções.

Quando um vetor é passado como parâmetro, o que é passado é o endereço ou localização do vetor. Desta forma, é possível que se tenha acesso a todos os elementos do vetor, pois os elementos são contíguos. Portanto, se houver alteração do valor de algum elemento do vetor dentro da função, essa alteração é definitiva.

P71) Escreva uma função **`void zera(a, n)`** que zera os `n` primeiros elementos do vetor `a` de inteiros.

```
void zera (int a[], int n) {
   int i = 0;
   while (i < n) a[i++] = 0;
   /* não precisa retornar nada */
}
```

Na função acima, os parâmetros são o vetor e quantidade de elementos que desejamos zerar.

Alguns exemplos de chamada da função `zera`:

```
/* exemplo de chamada da função zera no programa principal */
int main() {
   int x[100], y[30], z[50];
   int k = 20;
   /* zerar todo o vetor x */
   zera (x,100);
   /* zerar os 30 primeiros de x */
   zera (x, 30);
   /* zerar todo o vetor y */
   zera (y,30);
   /* zerar os k primeiros de z */
   zera (z, k);
}
```

P72) Escreva uma função **`int conta(a, n, x)`** que devolve como resultado, o número de elementos iguais a **`x`** que aparecem no vetor **`a`** de **`n`** elementos.

```
int conta (int a[], int n, int x) {
   int i = 0,
       cc = 0;
   while (i < n) {
       if (a[i] == x) cc++;
       i++;
   }
   return cc;
}
```

Veja abaixo, alguns exemplos de chamadas da função `conta`:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int vet[200];
  int n, k;
  /* atribui a k o número de nulos de x */
  k = conta (vet, 200, 0);
  /* imprime o número de -1s nos 50 primeiros elementos de vet */
  printf("\nnumero de elementos iguais a -1 = %5d", conta (vet, 50, -1));
  /*imprime quantas vezes cada número de 0 a 9 aparece nos n primeiros */
  for (k = 0; k < 10; k++)
     printf("\n%5d aparece %5d vezes", k, conta (vet, n, k));
  /* Lembram-se daquele exercício que verificava quantas vezes cada
     elemento do vetor se repetia ??? */
  for (k = 0; k < n; k++)
     printf("\nvet[%3d] = %5d aparece %5d vezes", k, vet[k],
              conta (vet, n, vet[k]));
  system("pause");return 0;
}
```

P72a) Escreva uma função **`void ContaRepetidos(int Vet[],int N)`** que imprime a quantidade de vezes que cada elemento ocorre no vetor **`Vet`** de **`N`** elementos. Use a função **`conta`** acima. Por exemplo, se N for 10 a função **`ContaRepetidos`** deve imprimir:

```
Elemento 0 ocorre <x> vezes
Elemento 1 ocorre <y> vezes
...
Elemento 9 ocorre <z> vezes
```

Note que se um elemento ocorre k vezes irão aparecer k linhas dizendo que o elemento ocorre k vezes.

P72b) Idem, imprimindo apenas uma linha para cada elemento repetido, isto é, se o elemento aparece k vezes, imprimir a linha com a quantidade apenas na primeira vez que ele aparece. Já resolvemos esse problema anteriormente. Basta para cada elemento verificar se ele já apareceu antes. Use também a função **`conta`** acima.

P72c) A função **`conta`** acima, conta sempre a partir do elemento 0 do vetor. Faça uma pequena modificação de modo que ela conte de um índice inicial (**`inicio`**) até um índice final (**`fim`**).

P72d) Repita o P72b com a nova função **`conta`**.

P73) Escreva uma função **`int trocavet(char a[],char b[],int n)`** que troca o conteúdo dos 2 vetores **`a`** e **`b`** de **`n`** elementos.

```
int trocavet (char a[], char b[], int n) {
   int i = 0;
```

```
    char aux;
    while (i < n) {
        aux = a[i];
        a[i] = b[i];
        b[i] = aux;
        i++;
    }
}
```

O que será impresso pelo programa abaixo? Veja resposta mais abaixo.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char x[5] = {"abcde"},
         y[5] = {"edcba"};
    int k;

    trocavet(x, y, 5);
    /* imprime x depois da troca */
    for (k = 0; k < 5; k++) printf("\nx[%2d] = %1c", k, x[k]);

    /* imprime y depois da troca */
    for (k = 0; k < 5; k++) printf("\ny[%2d] = %1c", k, y[k]);
    system("pause");return 0;
}
```

```
x[ 0] = e
x[ 1] = d
x[ 2] = c
x[ 3] = b
x[ 4] = a
y[ 0] = a
y[ 1] = b
y[ 2] = c
y[ 3] = d
y[ 4] = e
```

P74) Escreva uma função **`int busca(double a[],int n,double x)`**, que procura **`x`** no vetor **`a`** de **`n`** elementos, devolvendo como resultado o índice do elemento que é igual a **`x`** ou **`-1`** caso não encontre, Embora possa existir mais de um elemento igual a **`x`**, devolva o índice do primeiro encontrado.

```
int busca (int a[], int n, int x) {
    int i = 0;
    while (i < n) {
        if (a[i] == x) return i; /* encontrou */
        i++;
    }
    return -1; /* não encontrou */
}
```

Abaixo um exemplo de chamada e que será impresso pelo programa:

```
/* Gerar nummax números aleatórios entre 0 e 29.
   Verificar se são primos.
   Fazer isso comparando com todos os primos menores ou iguais a 29
*/
#include <stdio.h>
#include <stdlib.h>
#define nummax 20
int main() {
   int primos[10] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
   int n, k;
   for (k = 0; k < nummax; k++) {
       n = rand () % 30;  /* n estará entre 0 e 29 */
       if (busca (primos, 10, n) >= 0)
             printf("\n%5d e um numero primo", n);
       else printf("\n%5d nao e numero primo", n);
   }
 system("pause");return 0;
}
```

```
   11 e um numero primo
   17 e um numero primo
    4 nao e numero primo
   10 nao e numero primo
   29 e um numero primo
    4 nao e numero primo
   18 nao e numero primo
   18 nao e numero primo
   22 nao e numero primo
   14 nao e numero primo
    5 e um numero primo
    5 e um numero primo
    1 nao e numero primo
   27 nao e numero primo
    1 nao e numero primo
   11 e um numero primo
   25 nao e numero primo
    2 e um numero primo
   27 nao e numero primo
    6 nao e numero primo
```

P75) Escreva uma função **`int comp(char a[],char b[],int n)`** que compara os 2 vetores **a** e **b** de **n** elementos, devolvendo:

`n` se `a[i]=b[i]` para $0 \leq i < n$

`k` se `a[k]≠b[k]` e `a[i]=b[i]` para $0 \leq i < k$

```
int comp (char a[], char b[], int n) {
   int i = 0;
   while (i < n) {
       if (a[i] != b[i]) break;
       i++;
   }
   return i; /* se todos forem iguais a i, sai do while valendo n */
}
```

O que será impresso pelo programa abaixo?

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   char a[] = {0,1,2,3,4,5,6,7,8,9},
        b[] = {0,1,2,3,4,5,6,6,6,6};

   printf("\n*******numero de elementos iguais  = %5d", comp (a, b, 5));
   printf("\n*******numero de elementos iguais  = %5d", comp (a, b, 10));
   system("pause");return 0;
 }
```

```
*******numero de elementos iguais  =     5
*******numero de elementos iguais  =     7
```

P75a) Escreva uma função `int SemRepeticao(int A[],int N,int B[])` que recebe um vetor `A` de `N` elementos e devolve um vetor `B`, onde `B` é o próprio `A` sem elementos repetidos. O número de elementos de `B` (menor ou igual a `N`) é devolvido como retorno da função.

P75b) Idem `int SemRepeticao(int A[],int N)` eliminando as repetições e devolvendo no próprio A.

Sugestão: Uma forma de resolver seria usar um vetor B declarado dentro da função (local à função), jogando em B apenas os não repetidos e transferir o conteúdo de B para o A. Entretanto existe uma maneira de fazer sem usar o vetor auxiliar B. Tente resolver desta maneira.

P76) Escreva uma função `int Uniao(int A[],int N,int B[],int M,int C[])` que recebe os vetores `A` e `B` de `N` e `M` elementos e devolve o vetor `C` união entre `A` e `B`, ou seja, os elementos que estão em `A` ou em `B`. O valor de retorno da função é a quantidade de elementos de `C`. Pode supor que `A` e `B` não possuem elementos repetidos.

```
int Uniao (int a[], int n, int b[], int m, int c[]) {
   int i, j, nc=0;
   /* move a para c *
   for (i = 0; i < n; i++) c[i] = a[i];
   nc = n;
   /* verifica os que estão em b e não em a e os acrescenta a c */
   for (i = 0; i < m; i++) {
       /* procura b[i] em c[0] até c[nc-1] */
       for (j = 0; j < nc; j++)
           if (b[i] == c[j]) break; /* este b[i] já está */
       /* verifica se não encontrou b[i] em c
          se não encontrou acrescenta b[i] a c e incrementa nc */
       if (j == nc) {c[nc] = b[i]; nc++;}
   }
   return nc;
}
```

O que será impresso no programa abaixo?

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int a[] = {0,1,2,3,4,5,6,7,8,9},
       b[] = {5,6,7,8,9,10,11,12},
       c[20];
   int nc, i;

   nc = uniao (a, 10, b, 8, c);
   printf("\n*******uniao de a com b tem %5d elementos **********", nc);
   for (i = 0; i < nc; i++)
        printf("\nc[%2d] = %5d", i, c[i]);
   system("pause");return 0;
}
```

```
*******uniao de a com b tem    13 elementos **********
c[ 0] =     0
c[ 1] =     1
c[ 2] =     2
c[ 3] =     3
c[ 4] =     4
c[ 5] =     5
c[ 6] =     6
c[ 7] =     7
c[ 8] =     8
c[ 9] =     9
c[10] =    10
c[11] =    11
c[12] =    12
```

Outra solução para a União, pode ser feita usando a função SemRepetição acima. Basta jogar em C os dois conjuntos A e B, um em seguida ao outra e chamar a SemRepetição para eliminar os repetidos.

P77) Idem `int Interseccao(int A[],int N,int B[],int M, int C[]).`

P78) Idem `int Diferenca(int A[],int N,int B[],int M,int C[]).`

Um exemplo de biblioteca de funções - Conjuntos

Quando reunimos funções que manipulam um determinado objeto, dizemos que construímos uma biblioteca de funções.

Acima definimos algumas funções que manipulam conjuntos. Podemos reuni-las e constituir uma biblioteca de funções:

`1) int SemRepeticao(int A[],int N)`

Elimina as repetições do vetor A de N elementos. Devolve o número de elementos distintos. Ou seja, transforma o conteúdo de A em um conjunto.

```
2) int Pertence(int A[],int N, int X)
```

Verifica se X pertence ao conjunto A devolvendo 0 ou 1.

```
3) int Uniao(int A[],int N,int B[],int M,int C[])
```

Calcula a união do conjunto A com o conjunto B em C. Devolve a quantidade de elementos de C.

```
4) int Interseccao(int A[],int N,int B[],int M,int C[])
```

Idem intersecção.

```
5) int Subtracao(int A[],int N,int B[],int M,int C[])
```

Idem subtração.

```
6) int Contem(int A[],int N,int B[],int M)
```

Devolve 1 se A contém B ou 0 senão.

```
7) int Contido(int A[],int N,int B[],int M)
```

Devolve 1 se A está contido em B ou 0 senão.

Etc.

Podemos agora reunir todas estas funções num arquivo. Um programa que precisa manipular conjuntos pode dar **`#include`** deste arquivo e usar as funções já definidas.

Outro exemplo de biblioteca de funções – Números Complexos

Um número complexo é um par de números reais.

```
1)void SomaComplexo(double A,double B,double C,double
D,double *X,double *Y)
```

Devolve em X e Y a parte real e parte imaginária de A+B.

```
2)void SubComplexo(double A, double B, double C, double D,
                   double *X, double *Y)
```

Devolve em X e Y a parte real e parte imaginária de A-B.

```
3)void MultComplexo(double A, double B, double C,
                    double D, double *X, double *Y)
```

Devolve em X e Y a parte real e parte imaginária de A*B.

```
4)void DivComplexo(double A, double B, double C, double D,
                   double *X, double *Y)
```

Devolve em X e Y a parte real e parte imaginária de A/B.

```
5)double ModComplexo(double A, double B)
```

Devolve o valor do módulo do complexo.

Etc.

Mais um exemplo de biblioteca de funções – Polinômios

Um vetor **A** com **N** elementos pode conter um polinômio. **A[i]** é o coeficiente de $\mathbf{x^i}$. Ou seja, o polinômio seria:

$$\mathbf{A[0]x^0 + A[1]x^1 + A[2]x^2 + ... + A[N]x^N}$$

Considere agora as seguintes funções:

```
1) int SomaPol(double A[],int N,double B[],int M,double C[])
```

Calcular C=A+B. Soma os polinômios A e B de graus N e M respectivamente em C. Devolve o grau da soma dos polinômios.

```
2) int SubPol(double A[],int N,double B[],int M,double C[])
```

Idem C= A-B.

```
3) int MultPol(double A[],int N,double B[],int M,double C[])
```

Idem C=A*B.

```
4) int DivPol(double A[],int N,double B[],int M,double C[])
```

Idem C=A/B.

```
5) int DerivPol(double A[],int N,double B[])
```

Idem B=A', ou seja, calcula o polinômio derivada de A.

```
6) int IntegPol(double A[],int N,double B[])
```

Idem B=integral de A, ou seja, calcula o polinômio integral de A.

7) `double ValPol(int A[],int N,double X)`

Devolve o valor do polinômio no ponto X.

Etc.