UNIDADE 6: FUNÇÕES

Funções / Procedimentos

- Dividir uma tarefa complexa em tarefas menores, permitindo esconder detalhes de implementação

- Evita-se a repetição de um mesmo código

```
Tipo   Nome (lista de parâmetros)
{
    corpo
}
```

Procedimentos

- “Funções” que não retornam valores
- Tipo: void

```
#include <stdio.h>
#include <locale.h>

void desenha();
void   desenha( )
{
   int i;
   for (i = 0;  i < = 10;  i++)
            printf (“-”);
}
main ( )
{
    setlocale(LC_ALL, "Portuguese");
    desenha ( );
    printf (“ usando funções”);
              desenha ( );
}
```

Funções

Retornam valores

```
#include <stdio.h>
int   fatorial (int);
int   fatorial (int  n)
{
   int  i, resultado = 1;
   for ( i = 1;  i <= n;  i ++)
            resultado *=  i;
   return  resultado;
}
main ( )
{
   printf (“ o fatorial de 4 = %d”,
fatorial(4) );
   printf (“ o fatorial de 3 = %d”,
fatorial(3) );
}
```

Variáveis locais

- Variáveis declaradas dentro de uma função são denominadas locais e somente podem ser usadas dentro do próprio bloco

- São criadas apenas na entrada do bloco e destruídas na saída (automáticas)

Variáveis Locais

```
void desenha ( )
   {
      int  i, j;
         . . .
   }
main ( )
   {
         int a;
         desenha();
         a = i; ← erro
         . . .
   }
```

```
void desenha ( )
{
   int i, j;
    . . .
    . . .
}
void calcula ( )
{
    int i, j;
    . . .
    . . .
}
```

i, j em desenha são variáveis diferentes de i, j em calcula.

Variáveis Globais

- Variável que é declarada externamente podendo ser acessada por qualquer função

```
#include <stdio.h>
main ( )
{
   int i;
  .........
 .........
desenha ( );
calcula ( );
}
```

```
void desenha ( )
{
     int j;
     i = 0;
     . . .
}
 void calcula ( )
{
      int m;
      i = 5;
      . . .
}
```

Comando return

- Causa a atribuição da expressão a função

forçando o retorno imediato ao ponto de chamada da função.

```
#include <stdio.h>
main ( )
{
   char letra;

    printf (“Digite uma letra em
              minusculo”);

   letra = minusculo ( );
    if   (letra  == ‘a’)
            printf (“ok”);
}
```

```
char  minúsculo ( )
{
   char ch;

   scanf(“%c”, &ch);
   if  ( (ch >= ‘A’)  &&   (ch <= ‘Z’))
       return (ch + ‘a’ - ‘A’);
   else
       return (ch);
}
```

- Note pelo exemplo anterior que a função minúsculo lê um valor internamente convertendo-o para minúsculo.

Como usar esta função se já temos uma letra e desejamos convertê-la para minúsculo?

Passando dados para função

- Passagem de parâmetro por valor - uma cópia do argumento é passada para a função

- O parâmetro se comporta como uma variável local

Passando dados para função

```
main ( )
{
   printf (“ %c”, minúsculo (‘A’) );
```

parâmetro real

```
}
char minúsculo (char ch)
```

parâmetro formal

```
{
  if (( ch >= ‘A’)&&  (ch <= ‘Z’))
        return (ch + ‘a’-, ‘A’);
  else
        return (ch);
}
```

Passando dados para função - Exemplo

```
#include <stdio.h>
main ( )
   {
      int  num, b;
      printf (“ entre com um número  > o”);
      scanf (“ %d”,  &num );
      b = abs (num);
      printf (“ Valor absoluto de num = %d”, abs(num) );
    }

   int  abs (int x)
   {
      return ( ( x < 0 ) ? -x : x );
   }
```

Passando vários argumentos

Ex 1:

```
float área_retângulo
    (float   largura, float
    altura)
{
    return (largura *
    altura);
}
```

Ex 2:

```
float potência (float  base,  int  expoente)
{
   int  i;    float  resultado = 1;
   if (expoente == 0)
     return 1;
   for (i = 1; i <= expoente; i++)
      resultado *= base
   return  resultado;
}
```

Usando várias funções: calcular a seguinte seqüência

$$S(x, n) = x/1! + x^2/2! + x^3/3! + \ldots + x^n/ n!$$

```
#include <stdio.h>
float  serie (float ,   int );
float   potencia (float ,  int)
int  fat (int);
main( )
{
        float x;
        int  termos;

        printf("entre com o numero de termos: ");
        scanf("%d", &termos);
        printf("entre com o valor de X: ");
        scanf("%f", &x);
        printf("O valor de série = %f ", serie(x, termos));
}
```

```
float  serie (float x,   int n)
{
    int i;      float  resultado = 0;
    for ( i = 1;  i <= n; i++)
       resultado  +=  potência( x, i )  /  fat( i );
    return resultado;
}
```

```
int  fat (int n)
{
   int i, resultado = 1;

   for ( i = 1;  i <= n;  i ++)
            resultado *= i;
   return resultado;
}
```

```
float   potencia (float  base,  int expoente)
{
   int i;   float  resultado = 1;
   if (expoente == 0)
      return 1;
   for (i = 1;  i <= expoente;  i++)
        resultado *= base;
   return resultado;
}
```