FUNÇÕES

Funções são usadas para criar pequenos “pedaços” de códigos separados do programa principal. Em C, tudo, na verdade, é uma função. `int main (void)` é uma função, por exemplo.

Exceto a função `main`, todas as outras funções são secundárias, o que significa que elas podem existir ou não.

Funções são importantes porque elas retornam valores, ajudam a fragmentar o código em partes menores - mais fáceis de lidar - e ainda por cima podem ser utilizadas mais de uma vez no mesmo programa, poupando preciosos minutos de programação e inúmeras linhas de código.

Desta maneira, os principais motivos para a utilização de funções são:

- Evitar codificação: trocar certos trechos de programas que se repetem por chamadas de apenas uma rotina, que será codificada apenas uma vez.

- Modularizar programas: dividindo–os em rotinas (módulos) logicamente coerentes. Isto facilita a organização do programa, bem como seu entendimento.

Uma função deve ser definida com a seguinte sintaxe:

```
tipo nome_da_função (lista_de_parâmetros) {
    /* declaração de variáveis locais */
    /* instruções */
}
```

em que

- o conjunto `tipo nome_da_função (lista_de_parâmetros)` é chamado `protótipo` da função;
- `tipo` é o tipo da função e, pode ser `int`, `char`, `float`,`...` quando a função retorna um valor específico ou, `void` se a função realizar uma ação genérica sem retornar um valor definido;
- `lista_de_parâmetros` é um conjunto de variáveis utilizadas para a função receber os valores para os quais a função deve ser executada (também chamados de `argumentos`);
- na `declaração de variáveis` são declaradas as variáveis que as instruções da função manipularão internamente; estas variáveis (e os parâmetros da função) só são acessíveis pelas instruções da função e são chamadas de `variáveis locais`.

Se a função deve retornar um valor, uma de suas instruções deve ter a seguinte sintaxe:

```
return (expressão);
```

sendo os parâmetros facultativos. A execução desta instrução interrompe a execução da função e o processamento retorna à função que ativou a função.

FUNÇÃO SEM ARGUMENTOS

Exemplos:

1)

```
#include <stdio.h>
void mensagem (void) {
  printf ("\nSou uma funcao!\n");
}
/*-------------------------------------------------*/
void mensagem1 () {
  printf ("\nSou uma outra funcao!\n");
}
/*-------------------------------------------------*/
void tracos () {
  int i;
  printf ("\n");
  for (i = 1; i <= 40; i++)
    printf ("-");
  printf ("\n");
}
/*-------------------------------------------------*/
int main() {
  mensagem ();
  tracos ();
  mensagem1 ();
 }
```

2)

```
#include <stdio.h>
float soma () {
  float a, b;
  printf ("\nDigite um numero inteiro: ");
  scanf ("%f",&a);
  printf ("\nDigite um outro numero inteiro: ");
  scanf ("%f",&b);
  return a+b;
}
/*-------------------------------------------------*/
int main() {
  float total;
  total = soma();
  printf ("\nTotal: %f",total);
}
```

3)

```
#include <stdio.h>
float soma () {
  float a, b;
  printf ("\nDigite um numero inteiro: ");
  scanf ("%f",&a);
  printf ("\nDigite um outro numero inteiro: ");
  scanf ("%f",&b);
  return a+b;
}
/*-------------------------------------------------*/
int main() {
  printf ("\nTotal: %f",soma());
}
```

PASSAGEM DE PARÂMETROS

Como estratégia de programação, é desejável que cada função não faça uso de dados externos, pelo menos de forma direta, e toda comunicação seja feita através de parâmetros, ou se for o caso, como retorno da função.

Quando escrever funções que requeiram parâmetros, as informações sobre eles são incluídos no cabeçalho. Elas devem ser declaradas dentro dos parâmetros que seguem o identificador. As declarações de parâmetros são bem parecidas com declarações de variáveis. Os parâmetros declarados no cabeçalho são conhecidos como parâmetros formais. Os identificadores de parâmetros formais assumem os valores passados pela função de chamada e então são usados no bloco de comandos da função.

Exemplos:

1)

```
#include <stdio.h>
float soma (float x, float y) {
  return x+y;
}
int main() {
  float a, b;
  printf ("\nDigite um numero inteiro: ");
  scanf ("%f",&a);
  printf ("\nDigite um outro numero inteiro: ");
  scanf ("%f",&b);
  printf ("\n%.2f + %.2f = %.2f",a,b,soma(a,b));
}
```

2)

```
#include <stdio.h>
int sqr(int x);
int main() {
  int i = 10;
  printf("%d = %d^2",sqr(i),i);
}
int sqr(int x) {
  return (x*x);
}
```

PASSAGEM DE PARÂMETROS POR VALOR OU REFERÊNCIA

Existem dois modos usados para passagem de parâmetros nas funções: passagem de parâmetros por valor ou passagem de parâmetro por referência.

Quando um parâmetro é passado por valor, uma cópia do valor do parâmetro vigente é passada à função. A função não tem acesso a variável que contém o parâmetro e, portanto não pode modificá-la. A função pode mudar o valor de parâmetro passado por valor, mas no momento de voltar ao programa, o valor original de parâmetro vigente não será modificado.

Quando o parâmetro é passado por referência, o parâmetro que será passado na chamada da função deve necessariamente ser uma variável. Isso porque não é o valor da variável que está sendo passado, mas sim a sua referência. Qualquer alteração de valor no parâmetro correspondente refletirá em mudanças na variável correspondente, externa à função. Desta forma, na passagem de parâmetro por referência é necessário a passagem do endereço do argumento para a função.

Exemplos:

1)

```
#include <stdio.h>
// *a e *b: ponteiros que recebem os endereços de x e y
void troca(int *a, int *b);
int main() {
  int x=10, y=20;
  printf("x = %d y = %d\n",x,y);
  troca(&x,&y);
  printf("x = %d y = %d\n",x,y);
}
void troca(int *a, int *b) {
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
```

2)

```
#include <stdio.h>
int LeIntPositivo ();
void LeRealPositivo (float *x);
int main() {
  int a;
  float num;
  a = LeIntPositivo();
  LeRealPositivo (&num);
  printf("\ninteiro = %d\nreal = %4.2f",a,num);
}
int LeIntPositivo () {
  int n;
  do {
     printf ("\nDigite um inteiro positivo: ");
     scanf ("%d",&n);
  } while (n <= 0);
  return n;
}
void LeRealPositivo (float *x) {
  do {
     printf ("\nDigite um numero real positivo: ");
     scanf ("%f",x);
  } while (*x <= 0);
}
```

Variáveis Locais

As variáveis que são declaradas dentro de uma função são chamadas de locais. Na realidade todas as variáveis declaradas dentro de um bloco `{ }` podem ser referenciadas apenas dentro deste bloco. Elas existem apenas durante a execução do bloco de código no qual estão declaradas. O armazenamento de variáveis locais por default é feito na pilha, assim sendo uma região dinâmica.

Exemplo:

```
#include <stdio.h>
void linha (int x);
int main( ) {
  int tamanho;
  printf ("Digite o tamanho: ");
  scanf ("%d", &tamanho);
  linha (tamanho);
}
void linha(int x) {
  int i;
  for( i = 0; i <= x; i++)
```

```
    printf ("%c",95);
  /* A variavel i na funcao linha
     nao é reconhecida pela função main.*/
}
```

Variáveis Globais

São conhecidas por todo programa e podem ser usadas em qualquer parte do código. Permanecem com seu valor durante toda execução do programa. Deve ser declarada fora de qualquer função e até mesmo antes da declaração da função `main`. Fica numa região fixa da memória própria para esse fim.

Exemplo:

```
#include <stdio.h>
void func1( ), func2( );
int cont; // variavel global
int main( ) {
  cont = 100;
  func1( );
}
void func1( ) {
  int temp;
  temp = cont;
  func2( );
  printf ("\ntemp = %d", temp);
  printf ("\ncont = %d", cont);
}
void func2( ) {
  int cont;
  for(cont =1; cont<10; cont++)
    printf(".");
}
```

Exercícios

1 Faça uma função que retorne 1 se o número digitado for positivo e 0 se for nulo e -1 se for negativo.

2 Faça uma função que receba dois números inteiros e retorne a soma dos números inteiros existentes entre eles. Por exemplo: Para os valores de entrada 2 e 8, a soma será 3+4+5+6+7=25.

3 Faça uma função que receba três números inteiros: a, b e c, onde a>1; e retorne a soma de todos os números inteiros de b até c que sejam divisíveis por a. Exemplo: Para os valores de entrada 2 (para a), 5 (para b) e 10 (para c), a soma será 6+8+10= 24.

4 Faça uma função que receba três notas de um aluno e uma letra como parâmetros. Se a letra for A, a função deve calcular e retornar a média aritmética das notas do aluno (p1+p2+p3)/3, e, se for P deve calcular e retornar a média ponderada com pesos 1, 2 e 3 ((1*p1+2*p2+3*p3)/(1+2+3)).

5 Faça uma função que receba três notas de um aluno e os pesos de cada uma das provas. A, a função deve calcular e retornar a média ponderada das notas do aluno.

6 Faça um programa que leia cinco valores inteiros e imprima o maior e o menor valor. Utilize o conceito de função.