Funções: o que são e por que usá-las

Quando queremos resolver um problema, em geral tentamos dividi-lo em subproblemas mais simples e relativamente independentes, e resolvemos os problemas mais simples um a um.

A linguagem C dispõe de construções (abstrações) que auxiliam o projeto de programas de maneira *top-down*. Uma função cria uma maneira conveniente de encapsular alguns detalhes de processamento, ou seja, como algum resultado é obtido.

Quando esta computação é necessária, a função é *chamada*, ou *invocada*. Desta forma, quando uma função é chamada o usuário não precisa se preocupar como a computação é realizada.

É importante saber o que a função faz (qual o resultado da execução de uma função) e também como se usa a função. Criando funções, um programa C pode ser estruturado em partes relativamente independentes que correspondem às subdivisões do problema.

Você já viu algumas funções: printf(), scanf(). Elas são funções de uma biblioteca padrão (do C). Você não sabe como elas foram escritas, mas já viu como utilizá-las. Ou seja, você sabe o *nome* das funções e quais informações específicas você deve fornecer a elas (valores que devem ser *passados* para as funções) para que a função produza os resultados esperados.

Definindo funções

Um programa em C consiste de uma ou mais definições de funções (e variáveis).

- Há sempre uma função chamada main.
- Outras funções também podem ser definidas.
- Cada uma pode ser definida separadamente, mas nenhuma função pode ser definida dentro de outra função.
- Todas as funções devem ser declaradas antes de serem usadas.
- As funções da biblioteca padrão, tais como printf() e scanf(), são pré-definidas, mas mesmo assim devem ser declaradas (deve ser anunciado ao compilador que elas existem). É por isso que incluímos a linha #include <stdio.h> no início do código fonte.
- O formato geral da definição de uma função é:
    - *tipo-do-resultado nome-da função* (*lista-de-argumentos*)

Exemplo de declaração de função simples sem nenhum argumento:

```
void nome-da-função()

{

declarações e senteças (corpo da função)

}
```

Abaixo, mostramos um exemplo simples de um programa que consiste de duas funções: main() e msg(). Quando executado, este programa imprimirá a mensagem **Olá mundo!** dez vezes.

```
#include <stdio.h>

/* declaracao (protótipo) da funcao msg() */
void msg();

/* definicao da funcao main() */
int main()
{
  int i;

  i = 0;
  while ( i < 10)
    {
      msg();
      i ++;
    }
  return 0;
}

/* implementacao da funcao msg() */
void msg()
{
   printf("olá mundo!\n");
}
```

Argumentos

No próximo exemplo, vamos incluir o índice com argumento na função msg para ser impresso.

```
#include <stdio.h>

/* declaracao (protótipo) da funcao msg() */
void msg(int a);

/* definicao da funcao main() */
int main()
{
  int i=0;

  i = 0;
  while (i < 10)
    {
      msg(i);
      i ++;
    }
return 0;
}

/* implementacao da funcao msg() */
void msg(int a1)
{
   printf("[%d] olá mundo!\n", (a1+1));
}
```

Funções que retornam um valor

Uma função que retorna um valor tem no cabeçalho o nome do tipo do resultado.

O valor retornado pode ser de qualquer tipo, incluindo int, float e char (é claro que uma vez definida, a função só é de um tipo específico).

Uma função que retorna um **tipo** diferente de void executa alguns cálculos, e retorna o resultado (que é um único valor) para quem a chamou.

A função invocadora pode então usar o resultado. Para retornar um valor para a função invocadora, a função usa a sentença return.

O formato da sentença return é a seguinte:

```
tipo nome_da_funcao(argumento)
{
        corpo da função
    return variável tipo;
}
```

Considere o seguinte exemplo. O programa consiste de duas funções: main() e soma. O programa pede que o usuário digite 2 números inteiros e verifique e imprima a soma de um pelo outro.

```
#include <stdio.h>

int soma(int a, int b);
int subtracao(int a, int b);

/* definicao da funcao main() */
int main()
{
  int i=0;
  int v1, v2, resultSoma, resultSub;
  printf("Digite o valor para v1:");
  scanf("%d", &v1);
  printf("Digite o valor para v2:");
  scanf("%d", &v2);
  resultSoma= soma(v1, v2);
  resultSub= subtracao(v1, v2);
  printf("A soma de v1 por v2 eh:%d\n",resultSoma);
  printf("A subtracao de v1 por v2 eh:%d\n",resultSub);

return 0;
}

/* implementacao da funcao soma */
 int soma(int a, int b)
{
   int result= a+ b;
   return result;
}

/* implementacao da funcao subtracao */
 int subtracao(int a, int b)
{
   int result= a- b;
   return result;
}
```

Exercício: Implemente a função multiplicação e divisão tendo 2 argumentos, conforme explicado na função soma. Atente-se a função divisão, não se divide por 0, precisa fazer uma validação. Se o valor b for zero, o programa deve exibir uma informação ao usuário: “Não é possível dividir por zero!!”.

Biblioteca Math.h

Fornece um conjunto de funções para operações matemáticas, tais como funções trigonométricas, hiperbólicas, logarítmicas, potência e arredondamentos.

As funções da biblioteca math.h apresentadas a seguir retornam um valor do tipo double.

Nome	**Descrição**
floor( )	arredonda para baixo
ceil()	arredonda para cima
sqrt()	calcula raiz quadrada
pow(variavel, expoente)	potenciação
sin()	seno
cos()	cosseno
tan()	Tangente
fabs()	valor absoluto

```
#include <stdio.h>
#include <math.h>//necessária para usar as funções matemáticas

int main ()
{
  double x = 9.75;
  double arredonda_pbaixo = 0.0;
  double arredonda_pcima = 0.0;
  double raiz_quadrada = 0.0;
  double potencia = 0;
  double seno = 0;
  double cosseno = 0;
  double tangente = 0;

  printf("\n********* Utilizando a biblioteca math.h ***********\n\n");

  printf("\nFuncoes de arredondamento\n\n");
```

```
  printf("Valor original de x = %f\n",x);

  arredonda_pbaixo = floor(x);
  printf("Valor aproximado para baixo %f \n", arredonda_pbaixo );

  arredonda_pcima = ceil(x);
  printf("Valor aproximado para cima %f \n", arredonda_pcima);

  printf("\n-------------------------------------------------\n\n");

  printf("\nFuncoes de raiz e potenciacao \n\n");
  printf("Valor original de x = %f\n",x);
  raiz_quadrada = sqrt(x);
  printf("Valor da raiz quadrada %f \n",raiz_quadrada);

  x = ceil(x); //arredondando o x para cima, x passa a valer 10

  potencia = pow(x,2); //elevando o valor de x ao quadrado
  printf("Valor de %.2lf ao quadrado %.2f \n",x,potencia);

  printf("\n-------------------------------------------------\n\n");

  printf("\nFuncoes trigonometricas\n\n");

  x = 0; //atribuindo zero em x para fazer os cálculos trigonométricos

  seno = sin(x);
  printf("Valor de seno de %.2f = %.2f \n",x,seno);

  cosseno = cos(x);
  printf("Valor de cosseno de %.2f = %.2f \n",x,cosseno);

  tangente = tan(x);
  printf("Valor de tangente de %.2f = %.2f \n\n",x,tangente);
  printf("\n-------------------------------------------------\n\n");

  return 0;
}
```