Artigo Sobre Funções do Usuário

Vitor Amadeu Souza (Vitor@cerne-tec.com.br)

Introdução

O C possui uma série de funções que permitem com que o programador utilize destas para o desenvolvimento de seus projetos. Posso citar funções como para cálculo de seno, cosseno, arco seno, impressão de dados e etc que já estão prontos no C e basta utilizá-las adequadamente para que tenhamos o resultado esperado para um determinado processo. Porém na maioria dos projetos, precisamos criar funções específicas, que atendam uma necessidade de um determinado projeto. A estas funções chamamos de *funções de usuário* e elas tem a seguinte sintaxe:

```
tipo_de_dado_de_saída nome_função (tipo_de_dado_de_entrada)
{



}
```

Antes de continuarmos, irei citar o caso da função vista no capítulo passado chamada *abs().* Esta função tem um parâmetro de entrada, que é do tipo *inteiro* e ela retorna desta função um dado em forma de valor absoluto, também *inteiro.* Se fossemos analisar o arquivo de cabeçalho desta função, iríamos encontrar algo do seguinte tipo:

Int abs(int);

Ou seja, a função está declarada para *receber* e *retornar* um dado do tipo int. Os tipos de dados que uma função pode receber ou retornar são os mesmos que foram apresentados na seção de variáveis, porém há um novo que no momento não foi apresentado. É o tipo de dado *void (nulo)* onde neste caso, a função pode não receber nenhum valor assim como não retornar.

Vejamos alguns exemplos práticos afim de elucidar o conteúdo apresentado.

1 . Faça um programa onde haja uma função de usuário chamada teste. Nesta função, será declarada uma variável e a mesma será iniciada com o valor 10. Esta função será chamada de dentro do bloco main.

```
#include <mega16.h>    //Arquivo de cabeçalho definindo o
ATMEGA16
```

```
void teste (void);

main()                          //Bloco principal de programa
{
      teste();                  //Chama a função teste

}

void teste (void)               //Função teste
{
      char dado=10;             //Declara e inicializa a
                                //variável dado em 10
}
```

Vamos analisar agora passo a passo este pequeno programa:

1. `#include <mega16.h>` : Aqui está sendo definido o AVR que será utilizado pelo projeto.

2. `void teste (void);` : Este ponto é chamado de *prototipagem* da função. Sempre que usarmos funções de usuário, é necessário fazer a *prototipagem.* A *prototipagem* deve ficar sempre antes do bloco main e serve para informar ao compilador que uma determinada função existe no corpo do programa.

3. `main()`
   `{` : Aqui inicia o bloco principal de programa.

4. `teste();` : Neste ponto, a função teste é chamada. Neste caso o fluxo do programa é alterado já que neste momento a função teste iniciará o seu tratamento.

5. `void teste (void)`
   `{`
   `char dado=10;`
   `}` : Este aqui é o tratamento da função. Veja que esta função não recebe nenhum dado como parâmetro (void) e também não retorna nenhum (void). Aqui fica mais fácil visualizar o conceito de visibilidade de variáveis. Note que a variável dado somente é *visível* dentro da função teste e fora desta função nenhum outro ponto do programa tem acesso a esta variável. Este pensamento é válido para todas as outras funções do sistema, inclusive a *main().* Para que uma variável seja visível a todo o corpo de programa, ela teria que ser declarada fora de todas as funções, ou seja, ser global.

2. Faça um programa onde haja uma função de usuário chamada produto. Esta função receberá 1 *parâmetro* de entrada do tipo int, e irá dentro da própria função multiplicar o conteúdo deste parâmetro por ele mesmo.

```
#include <mega16.h>    //Arquivo de cabeçalho definindo o
ATMEGA16
```

```
void produto (int x);

main()                          //Bloco principal de programa
{
      produto(10);              //Chama a função produto

}

void produto (int x)            //Função produto
{
      int  y;
      y=x*x;                    //A variável y recebe x
                                //multiplicado por x
}
```

Note que neste exemplo, a função se chama agora produto e ela tem 1 parâmetro de entrada, chamado x. Ao entrar na função, uma nova variável é declarada, chamada y. A variável y recebe o conteúdo de x multiplicado por ele mesmo.

3. Faça um programa onde haja uma função de usuário chamada soma. Esta função receberá 2 *parâmetros* de entrada do tipo int, e irá dentro da própria função somar o conteúdo destes dois parâmetros.

```
#include <mega16.h>    //Arquivo de cabeçalho definindo o
ATMEGA16

void soma (int x, int y);

main()                          //Bloco principal de programa
{
      soma(100,1000);           //Chama a função de soma de
variáveis

}

void soma (int x, int y)        //Função soma
{
      int dado;                 //Declara a variável dado
      dado=x + y;               //A variável dado recebe a soma de
x+y
}
```

Observe que uma função não necessariamente precisa ter 1 parâmetro de entrada. É possível ter vários parâmetros e no exemplo passado está apresentado esta função com dois parâmetros de entrada. É importante ressaltar também que não necessariamente os parâmetros de entrada tenham que ser do mesmo tipo, pois podemos colocar vários tipos como parâmetros de entrada para as funções de usuário no C.

4. Faça um programa onde haja uma função de usuário chamada soma_numeros. Esta função receberá 2 *parâmetros* de entrada do tipo int, e irá retornar o resultado da soma para quem chamar esta função.

```
#include <mega16.h>    //Arquivo de cabeçalho definindo o
ATMEGA16
int soma_numeros (int x, int y);

main()                         //Bloco principal de programa
{
      int x;
      x=soma_numeros(100,1000);
                               //Chama a função de soma de
variáveis
                               //O retorno da função será
                               //armazenado na variável x

}

int soma_numeros (int x, int y)     //Função soma
{
      return (x + y);               //Retorna a soma de x + y
}
```

Neste exemplo, o parâmetro de retorno não é mais *void* e sim *int.* Ao entrar na função, encontramos o comando `return (x + y)` que retorna da função com o resultado da soma dos dois parâmetros da função. No caso do parâmetro de retorno, diferente do parâmetro de entrada, somente pode retornar 1 valor.