Funções

Sub-rotinas (Módulos)

As sub-rotinas, também chamadas de módulos ou sub-programas são trechos de instruções que realizam tarefas específicas e que podem ser chamadas (executadas) diversas vezes.

Ao criarmos sub-programas deixamos o programa como um todo mais simples de entender. Desta forma é possível focalizar a atenção em um problema de cada vez e fazer os programas ficarem mais organizados e menores.

As linguagens sem uso de sub-rotinas executam uma linha de código, ou uma instrução por vez, até o final do programa, não sendo possível fazer desvios de execução.

A linguagem C possibilita o uso de sub-rotinas por meio de funções. Existe a função main(), que é sempre a primeira rotina a ser executada, e a partir dela, podemos executar outras funções dentro de nosso programa.

A linguagem C é formada por funções, como printf(), scanf(), system(), e outras, que são incluídas através da diretiva #include.

Como declarar as funções

```
    Tipo de retorno  nome da função  (tipo de parâmetro  nome do
parâmetro, ....)
    {
          Declaração das variáveis locais;
          Corpo da função (instruções);
          Retorno;
    }
```

Tipo de retorno -> especifica o tipo de valor (int, float) que a função irá retornar através do comando return. Caso a função não retorne nenhum valor, ou seja, ela é usada para executar comandos e não devolve nada, então é usado o tipo de retorno void (não retorna nada).

Nome da Função -> deve ter um nome significativo para identificar o que a função executa. Exemplo: Se vamos apenas somar 2 números na função, ela poderá ser chamada de Soma ou Adição.

Após o nome da função temos uma ***lista de parâmetros***. São valores que a função precisa receber para realizar seu processamento. Cada parâmetro é separado por vírgula. O tipo de parâmetro é o tipo da variável que será passada para a função. **Uma função pode não ter parâmetros.**

Variáveis locais -> Declaradas dentro da função, e portanto, só serão conhecidas (visíveis) dentro da função. Uma variável local é criada quando ocorre a entrada da função e é destruída ao sair da função.

Corpo da Função -> Tudo que será executado dentro da função (comandos). Ela deve estar entre chaves. { marca início da função e } marca o fim da função.

Exemplo:

```
float media_notas(float nota1, float nota2)
{
	float media;
	media=(nota1+nota2)/2;
	return media;
}
```

Protótipo das funções

Caso a função tenha sido implementada após a função main(), devemos declará-la antes da função main(). Isto deve ser feito para que o compilador saiba da existência de uma função dentro do programa, podendo esta estar no meio ou final.

Exemplo:

```
float media_notas(float nota1, float nota2);

void main()
{
  ……

}
```

Chamada da Função (como usar a função no programa)

A chamada da função existe quando há uma referência de uma função dentro do programa. Durante o acionamento da função, o fluxo de execução é desviado para ele, e logo após o término de sua execução, o fluxo de execução do programa retorna ao comando de origem (que chamou a função).

Na chamada da função deve haver o envio ou não de argumentos para outra função. Esses argumentos são valores de variáveis que podem ser usadas na função. Neste momento não precisa ser definido o tipo de dados, pois estes já foram definidos no início da função.

Exemplo:

```
float media_notas(float nota1,float nota2);

main()
{
	float n1, n2,media;

	media = media_notas(n1,n2)
```

n1 e n2 são argumentos (enviam)

```
        ...
}

float media_notas(float nota1, float nota2)
{
    ..........
}
```

nota1 e nota2 são parâmetros (recebem)

Os argumentos enviados para a função devem ser compatíveis com os parâmetros, ou seja, se forem enviados dois valores como argumentos do tipo float, na função seus parâmetros devem ser do tipo float.

Exemplo Completo:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

float media_notas(float nota1,float nota2);

void main()
{
 float n1, n2, media;

 printf("Cálculo de media de duas notas\n");

    printf("Informe a primeira nota");
 scanf("%f",&n1);

    printf("\nInforme a segunda nota");
    scanf("%f",&n2);

 media=media_notas(n1,n2);

 printf(" A media das notas: %.2f ",media);

 getch();
}

float media_notas(float nota1, float nota2)
{
      float media;
      media=(nota1+nota2)/2;
      return media;
}
```

Obs.: A função media_notas é do tipo float porque seu retorno será uma variável do tipo float.

Os parâmetros nota1 e nota2 são do tipo float porque seus respectivos argumentos (n1 e n2) são do tipo float.

Return

Podemos usar o comando return de duas maneiras:
- Para provoca a saída imediata da função que o contém, retornando ao código chamador.
- Para devolver um valor.

Void

Todas as funções que não devolvem valor devem ser declaradas como tipo void.

Quando uma função não tem parâmetros, deve ser colocado o void entre parênteses (para alguns compiladores).

Exemplo:

```
void entrada(void);

main()
{
 entrada();

 ...


}

void entrada(void)
{
 printf("<1> Adição \n");
 printf("<2> Subtração \n");
 printf("<3> Multiplicação \n");
 printf("<4> Divisão \n");}
```

Exemplos de passagem de parâmetro

Exemplo:

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

```
#include <conio.h>

float quadrado(float num);

void main()
{
  float num, quad;

      printf("Entre com o número");
      scanf("%f",&num);
     printf("\nNa funcao main a variavel num vale %.2f \n",num);

      quad = quadrado(num); // chamando a função

       printf("O quadrado do %.2f eh igual a %.2f \n",num,quad);
      getch();

}


float quadrado(float num)
{
     num = num * num;
     printf("Na função QUADRADO a variavel num vale %.2f \n",num);
      return num;
}
```

No exemplo vemos que o valor mostrado da variável *num* a primeira vez (na função main()) é diferente do valor mostrado na função quadrado(), pois, são duas variáveis diferentes.

Este tipo de passagem de parâmetros é chamada de passagem de parâmetros por valor. Isto ocorre porque são passados para a função apenas os valores dos parâmetros e não os próprios parâmetros.

Obs.: A função quadrado() é do tipo float porque o retorno dela será numa vairável do tipo float (num).

O parâmetro (num) é do tipo float porque o argumento (num) é do tipo float.

Variáveis Globais

Quando declaramos uma variável dentro de uma função, ela é visível apenas dentro da função. Para criarmos variáveis que sejam visíveis pelo programa todo, criamos variáveis globais. Elas são declaradas fora da função main() e fora de qualquer outra função.

Exemplo:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

float n1, n2, media;

void media_notas();

void main()
{
 printf("Cálculo de media de duas notas\n");

    printf("\nInforme a primeira nota");
 scanf("%f",&n1);

    printf("\nInforme a segunda nota");
    scanf("%f",&n2);

    media_notas();

 printf(" \nA media das notas: %.2f ",media);

 getch();

}

void media_notas()
{
        media=(n1+n2)/2;

}
```

Variáveis Globais