Aula 12 - Sub-rotinas

Revisão

Padrão de um programa em linguagem C:

```c
#include <"arquivo cabeçalho">
/* inclusão dos cabeçalhos necessários */

int main( void )
{
    /* declaração de variáveis */
    /* leitura de dados */
    /* cálculo de resultados */
    /* escrita de resultados */
    return 0;
}
```

Contexto

- Em todos os programas que vimos até então, todo o código executado encontrava-se dentro da função **main.**
- Dependendo da nossa aplicação, o código pode torna-se grande e consequentemente difícil de manter, principalmente se muitos trechos de código se repetem.
- Podemos então estruturar nosso programa utilizando as **sub-rotinas**.

Sub-rotinas

- As **sub-rotinas** são componentes do código que realizam uma função específica;
- As **sub-rotinas** se encontram em um **ponto específico** do programa;
- São relativamente independentes do resto do programa **(escopo local)**;
- Podem ser chamadas a partir de **qualquer ponto do programa principal**;
- Existem dois tipos de **sub-rotinas** na linguagem C: **função** e **procedimento**.

Sub-rotinas

```
#include <"arquivo cabeçalho">
/* inclusão dos cabeçalhos necessários */

int main( void )
{
    /* declaração de variáveis */
    /* leitura de dados */
    /* cálculo de resultados */
    /* escrita de resultados */
    return 0;
}
```

Interface das sub-rotinas

- Ela é necessária para definir como a subrotina deve ser chamada pelo programa;
- Uma interface de sub-rotina é composta de:
  - Tipo de retorno;
  - Nome;
  - Tipos dos parâmetros entre parâmetros e separados por vírgula

```
tipo_de_retorno nome( tipo_dos_parametros );
```

```
int max( int, int );
int min( int, int );
int media( int, int, int );
```

Função

Elementos de uma função

- Tipo de retorno
  - Tipo do dado retornado após a execução da função
- Nome
  - Identifica a função. O nome remete ao problema que a função se propõe a resolver (mesma regra dos nomes de variáveis)
- Corpo
  - Código que soluciona o determinado problema
- Parâmetros
  - Dados recebidos externamente para utilização da função
- Retorno
  - Dado retornado após a execução da função

```
tipo_de_retorno nome( parametros )
{
    corpo  /* comandos */
    return valor_de_retorno;
}
```

Chamadas das Funções

- Os parâmetros da função se comportam como variáveis e existem apenas durante a execução dela;
- Ou seja, para cada parâmetro:
  - Um compartimento na memória é alocado (variável);
  - O valor armazenado inicialmente é o valor da expressão que aparece como parâmetro efetivo na chamada da função;
  - A memória é desalocada quando a função terminar a execução;

EXEMPLO 1 - CALCULADORA

Operações

1 - Soma

2 - Subtração

3 - Multiplicação

4 - Divisão

Procedimento

Contexto

- Ao contrário da função, o procedimento é uma sub-rotina que não retorna valor;
- Por padrão, na linguagem C toda sub-rotina é definida como uma função;
- No entanto, temos como definir um procedimento utilizando o **void** (tipo) como tipo de retorno. O **void** também pode ser utilizado quando **não há parâmetros** na função;
- Quando o retorno for do tipo **void**, o comando return não é seguido de uma expressão (não precisa ser utilizado).

```c
int imprimir_linha( void )
{
    printf( "--------------------\n" );
    return 0;
}  /* função */
```

```c
void imprimir_linha( void )
{
    printf( "--------------------\n" );
    return;
}  /* procedimento */
```

EXEMPLO 2 - IMC

Condição	IMC em adultos
Abaixo do Peso	< 18,5
Peso Normal	18,5 <= IMC <= 25
Acima do Peso	25 < IMC <= 30
Obeso	> 30

IMC = peso/(altura)^2

Vantagens do uso de sub-rotinas

- Decomposição de tarefas complexas em tarefas menores
  - As tarefas se tornam menos complexas
  - Facilidade em manter o código
- Reutilização de código
  - Diminuição do tamanho do programa
  - Eliminação de códigos repetidos
- Reutilização de código em outras aplicações
  - Criação de bibliotecas, por exemplo.
- Divisão do trabalho entre uma equipe
  - Cada programador é responsável por determinada função do programa.