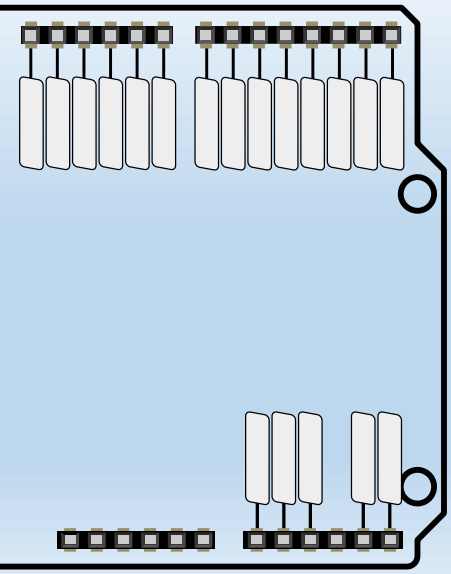
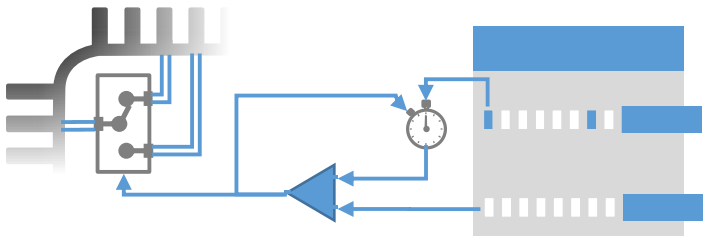
FUNÇÕES - I



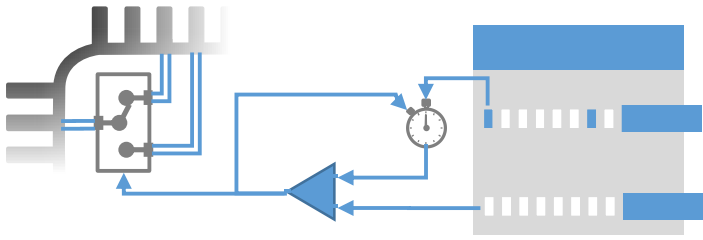
Funções em C

- A modularização é um recurso muito importante apresentado nas linguagens de programação, onde um programa pode ser particionado em sub-rotinas específicas.
- A linguagem C possibilita a modularização por meio das funções.
- Um programa escrito em C, possui no mínimo uma função chamada main, por onde a execução do programa começa.
- Existem muitas outras funções pré-definidas em C, por exemplo: strcmp, strcpy, entre outras.
  - Essas funções são inseridas no programa pela diretiva #include.
- O usuário também pode criar quantas funções quiser, dependendo do problema que está sendo resolvido pelo programa.



Funções em C

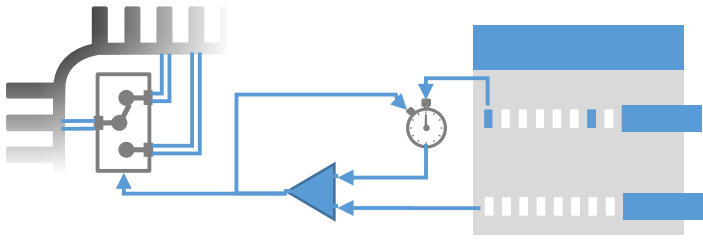
- Uma função é um conjunto de sentenças que podem ser chamadas de qualquer parte de um programa.
- As funções não podem ser aninhadas, ou seja, uma função não pode ser declarada dentro de outra função.
  - A razão para isso é permitir um acesso eficiente aos dados.
- Cada função realiza determinada tarefa, e quando o comando return é executado dentro da função
  - Retorna-se ao ponto em que a função foi chamada pelo programa principal (main) ou por uma função principal.



Funções em C

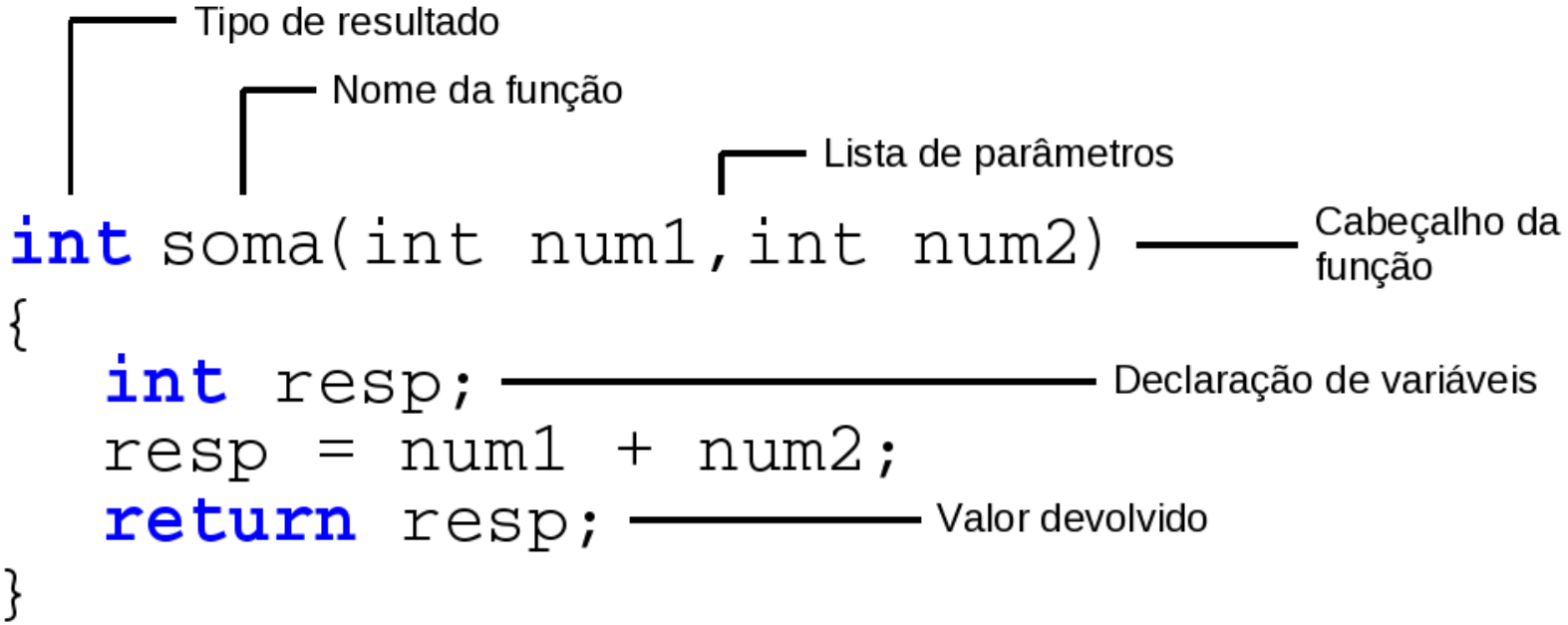
- Definições
  - tipoRetorno: tipo de valor devolvido pela função.
  - nomeDaFunção: identificador ou o nome dado à função.
  - listaDeParametros: são as variáveis passadas para a função. Quando a função recebe mais de um parâmetro, esses são separados por vírgula.
  - expressao: indica o valor que a função vai devolver para o programa.

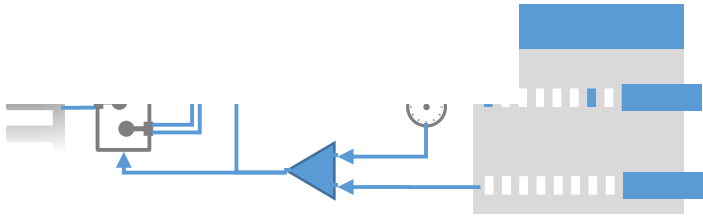
```
tipoRetorno nomeDaFuncao(listaDeParametros){
        //corpo da funcao
        return expressao;
}
```



Funções em C

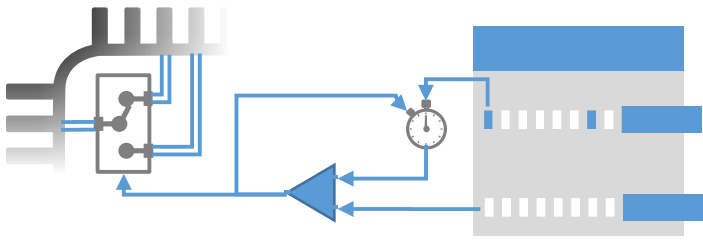
- Tipo de resultado: o tipo de dado retornado (devolvido) pela função. O tipo sempre aparece antes do nome da função.
- Lista de parâmetros: a lista de parâmetros com tipos com o seguinte formato:
  - (tipo1 parametro1, tipo2 parametro2, ...)





Funções em C

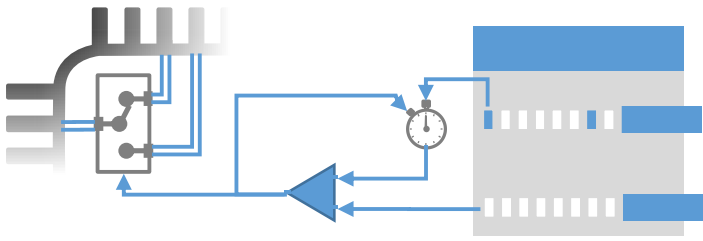
- Corpo da função: tem que estar definido entre o abre chaves "{" e o fecha chaves "}". {Não há ponto-e-vírgula depois da chave de fechamento.
- As constantes, tipos de dados e variáveis declaradas dentro da função são locais à função e não podem ser lidas ou acessadas fora da função.
- Valor devolvido pela função: mediante a palavra return, pode-se devolver (retornar) o valor da função.
- Uma chamada de função produz:
  - A execução das instruções o corpo da função.
  - Um retorno para a unidade de programa que fez a chamada depois que a execução da função terminou (ocorre quando a senteça return é encontrada).



Nome da Função

- Um nome de função começa com uma letra ou um (_).
- O nome pode conter tantas letras, números ou (_), quanto deseje o programador. Alguns compiladores ignora a partir de uma certa quantidade de caracteres.
- Letras maiúsculas e minúsculas são distintas para efeito de nome de função.

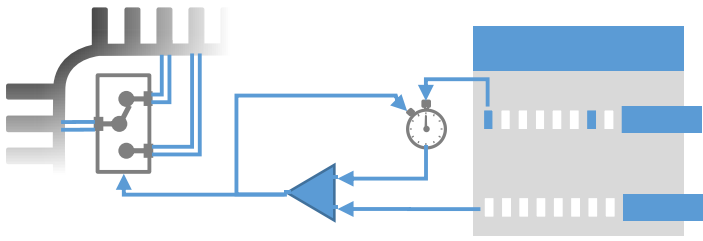
```
//nome da função: max
int max(int x, int y)
//nome da função: media
double media(double x1, double x2)
```



Funções em C

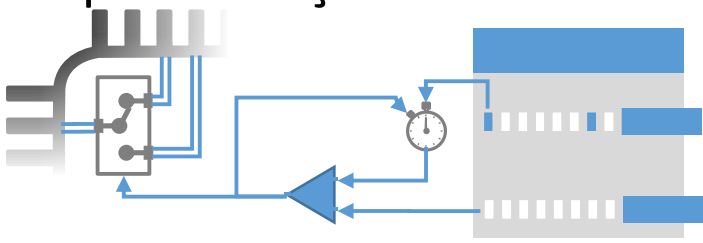
- Tipo de dado de retorno
- O usuário deve especificar qual é o tipo de dado que a função vai retornar.
- O tipo de dado deve ser um dos tipos de C, por exemplo:
  - int, double, float ou char; ou então um tipo definido pelo usuário.
- O tipo void serve para indicar que a função não retorna nenhum valor.

```
//retorna um valor inteiro
int max(int x, int y)
//retorna um double
double media(double x1, double x2)
//retorna um float
float soma(int numElem)
//não retorna nada
void tela(void)
```



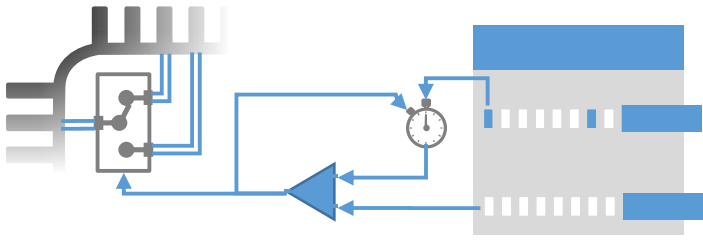
Resultados de uma função

- A função pode devolver um único valor, e o resultado é mostrado como uma sentença return.
- O valor retornado por uma função deve seguir as mesmas regras que são aplicadas a um operador de atribuição.
  - Por exemplo, o valor int não pode ser retornado se o tipo de retorno da função for um char.
- Uma função pode ter qualquer número de sentenças return.
  - Sempre que o programa encontra uma instrução return, retorna para a instrução que originou a chamada para a função.
  - A execução de uma chamada à função termina se não encontrar nenhuma instrução return; e nesse caso, a execução continua até a chave final do corpo da função



Chamando uma função

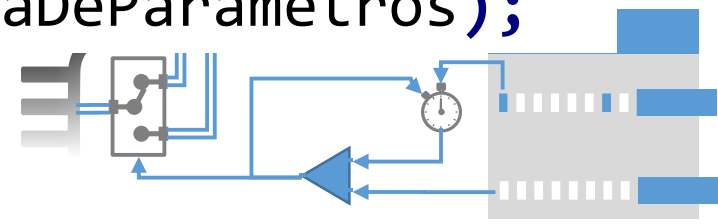
- Para que uma função seja executada, é necessário que a função seja chamada.
- Qualquer expressão pode conter uma chamada a uma função, a qual redirecionará o controle do programa para a função chamada.
- Normalmente, a chamada a uma função é realizada pela função main, mas a chamada pode ser feita através de outra função.
- Quando a função termina sua execução, o controle do programa volta para a função main() ou para a função de chamada se esta não for o main.



Protótipo de Funções

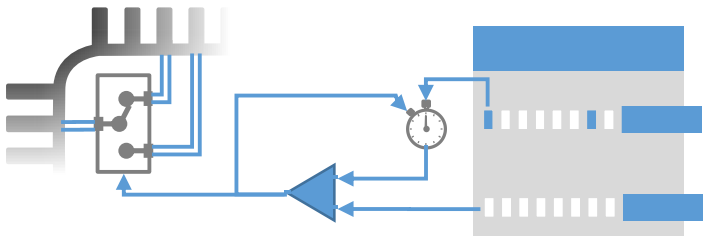
- Em C é necessário que uma função seja declarada ou definida antes do seu uso.
- A declaração de uma função sem sua implementação antes do main é chamada de protótipo da função.
- O protótipo de uma função possui o mesmo cabeçalho da função, com a diferença que os protótipos terminam com ponto-e-vírgula.
- Um protótipo é composto de: tipo, nome da função, parâmetros (que devem estar entre parênteses e é opcional) e um ponto-e-vírgula no final

```
tipoRetorno nomeDaFuncao(listaDeParametros);
```



Protótipo de Funções

- Um protótipo declara uma função e passa ao compilador informação suficiente para verificar se a função está sendo chamada corretamente em relação ao número e tipo de parâmetros e ao tipo de retorno da função.
- Os protótipos são definidos sempre no início do programa, antes da definição do main().
- O compilador utiliza os protótipos para validar que o número e os tipos de dados do argumentos na chamada da função, sejam os mesmos que aparecem na declaração formal da função chamada.
- Caso encontre alguma inconsistência, uma mensagem de erro é visualizada.



Protótipo de Funções

- Uma vez que os protótipos foram processados:
  - o compilador conhece quais são os tipos de argumentos que ocorreram.
- Quando é feita uma chamada para a função, o compilador confirma se o tipo de argumento na chamada da função é o mesmo definido no protótipo.
  - Se não são os mesmos, o compilador gera uma mensagem de erro.

```
int quadrado(int num); //protótipo
int main(int argc, char *argv[]){
        //...
}
int quadrado(int num){
        //...
}
```