

Aula 02 - Funções

Funções

1. Assinatura de uma função e retorno
2. Declaração de funções em arquivos

Assinatura de uma função

Considerando que C é uma linguagem estruturada, as funções são a base para a divisão do algoritmo em partes que podem ser reutilizadas.

Assinatura de uma função

```
tipo_de_retorno nome_da_funcao (parametro1, parametro2, ...) {  
    instrucoes;  
    return valor; // ou return;  
}
```

Assinatura de uma função

- tipo de retorno: tipos fundamentais do C ou `void` . Define o tipo de uma função;
- nome da função: não pode ser uma palavra reservada. Define como essa função será chamada durante o resto do código;
- parametros: os tipos devem ser declarados e podemos passar quantos parâmetros forem necessários;
- return: caso o tipo da função não seja `void` , precisamos retornar algo. A variável retornada deve ser do mesmo tipo que a função.

Assinatura de uma função

```
int soma_int(int a, int b){  
    int soma = a + b;  
    return soma;  
}
```

```
float soma_float(float a, float b){  
    float soma = a + b;  
    return soma;  
}
```

Assinatura de uma função

```
int fibonacci(int pos){  
    int a = 0;  
    int b = 1;  
  
    int i;  
    for(i = 3; i <= pos; i++){  
        int aux = b;  
        b = a + b;  
        a = aux;  
    }  
  
    return b;  
}
```

Declaração de funções

Em C:

- Uma função **deve** ser declarada antes de ser usada;
- Podemos declarar a assinatura da função antes de ser usada e fazer a implementação depois;
- Só podemos declarar uma função fora de outra função

Declaração de funções

```
#include <stdio.h>

int fibonacci(int pos){
    // implementação
}

int main() {
    int n = 10;
    int r = fibonacci(n);
    printf("\no numero na posicao %d eh %d\n", n, r);
}
```

Declaração de funções

```
#include <stdio.h>
int fibonacci(int); // somente o tipo é obrigatório

int main() {
    int n = 10;
    int r = fibonacci(n);
    printf("\no numero na posicao %d eh %d\n", n, r);
}

int fibonacci(int pos){
    // implementação
}
```

Arquivos de cabeçalho e implementação

Com o tempo, as funções são usadas sem sabermos exatamente o que está implementado nelas.

Por exemplo a função `printf` definida em `stdio.h` : não vemos a implementação dela, mas usamos para exibir valores na tela.

Arquivos de cabeçalho e implementação

São os arquivos `.h` que incluímos no início do código, como o `stdio.h`.

Eles apresentam a assinatura das funções, mas não necessariamente a sua implementação.

Facilitam a reutilização e organização do código.

Arquivos de cabeçalho e implementação

Para sabermos como uma função funciona, olhamos apenas a sua assinatura. Por exemplo:

```
int soma(int, int);  
int fatorial(int);  
float valor_percentual(float total, float percentual);  
int idade(int ano, int mes, int dia);  
int printf(const char *format, ...);  
int main(void);  
int main(int argc, char* argv[]);
```

Arquivos de cabeçalho e implementação

1. Descrição das assinaturas no arquivo `.h`
2. Implementação das funções no arquivo `.c` de mesmo nome
3. Inclusão no arquivos que possui a função `main` :
 - entre `""` se os arquivos estiverem na mesma pasta
 - entre `<>` se os arquivos estiverem no path

Arquivos de cabeçalho e implementação

```
// funcoes.h  
int soma_int(int, int);
```

```
// funcoes.c  
int soma_int(int a, int b){  
    return a + b;  
}
```

```
// main.c  
#include <stdio.h>  
#include "funcoes.h"  
  
int main(){  
    int a = 1, b = 2;  
    int r = soma_int(a, b);  
    printf("%d\n", r);  
}
```

Preparação para o laboratório

Para ficar mais fácil no laboratório, vamos criar:

- Um arquivo biblioteca.h com o cabeçalho das funções que vamos implementar
- Um arquivo biblioteca.c com as implementações das funções
- Um arquivo main.c com os testes das funções