Funções em Linguagem C

Funções em Linguagem C

V1.0.0 (NOV/2023)

Sumário

1 Funções **3**

1.1 Declaração de Função 3

1.2 Definição de Função 3

1.3 Chamada de Função 4

1.4 Parâmetros 4

2 Change log **6**

2.1 Version 1.0.0 6

1 Funções

Em linguagem C, as funções são blocos de código que realizam tarefas específicas e podem ser chamadas em um programa principal. As funções desempenham um papel fundamental na organização e modularização do código, tornando-o mais legível e fácil de manter.

1.1 Declaração de Função

A declaração de uma função em C envolve informar ao compilador sobre a existência da função, especificando seu tipo de retorno, nome e os tipos de seus parâmetros. Isso permite que o compilador faça verificações de tipo durante a compilação e torne possível chamar a função antes de sua definição no código.

Para declarar uma função, você precisa fornecer: Nome da função. Tipo de retorno (int, float, void, etc.). Parâmetros que a função aceita.

Sintaxe da Declaração de função:

```
1 tipo_de_retorno nome_da_funcao(tipo_do_parametro1 parametro1,
     tipo_do_parametro2 parametro2, ...);
```

É comum colocar as declarações de função no início de um programa ou em um arquivo de cabeçalho (.h) para que elas sejam conhecidas pelo compilador antes de serem usadas. A declaração da função é fundamental para a modularização e reutilização do código em C.

1.2 Definição de Função

A definição de uma função em linguagem C envolve a implementação do corpo da função, onde a lógica específica da tarefa que a função realiza é detalhada.

Sintaxe da definição da Função:

```
1 tipo_de_retorno nome_da_funcao(tipo_do_parametro1 parametro1,
     tipo_do_parametro2 parametro2, ...) {
2 // Corpo da função - lógica específica aqui
3 return valor_de_retorno; // Opcional, dependendo do tipo de retorno
4 }
```

A definição de função é crucial para modularizar e organizar o código em C, permitindo a reutilização de lógica específica e facilitando a manutenção do programa.

1.3 Chamada de Função

A chamada de função é essencial para executar o código dentro do corpo da função. Isso permite a execução de tarefas específicas, retorno de resultados e interação dinâmica entre diferentes partes do programa.

Sintaxe da chamada de Função:

```
1 tipo_de_retorno nome_da_funcao(tipo_do_parametro1 parametro1,
     tipo_do_parametro2 parametro2, ...);
```

A chamada de função é essencial para executar o código dentro do corpo da função. Isso permite a execução de tarefas específicas, retorno de resultados e interação dinâmica entre diferentes partes do programa.

1.4 Parâmetros

Os parâmetros de função são variáveis que permitem a passagem de dados para uma função quando ela é chamada. Eles desempenham um papel vital na comunicação entre a função e o código que a invoca. Ao declarar uma função, você lista os parâmetros entre parênteses. Cada parâmetro inclui o tipo de dado e um nome que será usado internamente na função. Uma função pode ter zero ou mais parâmetros. O número e os tipos de parâmetros devem ser consistentes entre a declaração e a definição da função

Sintaxe dos Parâmetros:

```
1 tipo_de_retorno nome_da_funcao(tipo_do_parametro1 parametro1,
     tipo_do_parametro2 parametro2, ...){
2 // Corpo da função
3 }
```

Os parâmetros de função são peças-chave na construção de código modular e facilitam a criação de funções genéricas e flexíveis. Compreender como usá-los corretamente contribui significativamente para a eficácia e legibilidade do seu código em C.

Parâmetros podem ser passados de duas maneiras por **Valor** e por **Referência**.

- **Passagem por Valor:**

Em C, os parâmetros são passados por valor por padrão. Isso significa que uma cópia dos valores é passada para a função, e qualquer modificação dentro da função não afeta as variáveis originais fora dela.

Exemplo: void incrementa(int x) { x++; }

- **Passagem por Referêcia(Usando ponteiros):**

Passar por referência significa que a função recebe um ponteiro que aponta para a localização de memória da variável original. Qualquer alteração feita através do ponteiro dentro da função afetará a variável original fora da função. Para permitir que uma função modifique diretamente as variáveis originais, você pode usar ponteiros como parâmetros

2 Change log

2.1 Version 1.0.0

- Documento sobre Funções em linguagem C