



# Mini Apostila de Programação em C

Prof. Manassés Ribeiro  
manasses.ribeiro@ifc.edu.br



## Declaração de variáveis e tipos de dados em C

A declaração de variáveis em C segue o padrão

**<tipo\_de\_dados identificador;>**.

Os tipos dados primitivos são os três primeiros, sendo:

Palavra-chave	Tipo (descrição)
<b>char</b>	Caracter
<b>int</b>	Inteiro e lógico
<b>float</b>	Ponto flutuante (real) de precisão simples
<b>double</b>	Ponto flutuante (real) de precisão dupla
<b>void</b>	vazio (sem valor)



## Declaração de variáveis e tipos de dados em C


- Com exceção de **void**, os outros tipos de dados primitivos possuem **modificadores que alteram o tamanho do tipo de dado** ou sua forma de representação.
- Os modificadores fazem com que seja **possível adequar o tipo de dados às necessidades de armazenamento** em determinados casos. São eles:
  - signed, unsigned, long e short.
- Para ver as formas de utilização consultar documentação mais abrangente.



## Declaração de variáveis

- As regras para declaração de variáveis e formação de identificadores são as mesmas já vistas nos algoritmos.
- Não pode conter **caracteres especiais** (Ex: \$%#@), **espaços** e nem começar com **números**.

```
int a, b, c;  
float salario, _peso, altura23;  
char tipo, nome[50];
```



# Estrutura mínima de um programa em C

Todo programa em C é composto no mínimo de duas partes que são:

- as **diretivas de pré-processamento**; e
- a **função principal (main)** que é o ponto de partida acionado pelo sistema operacional quando da execução.

```
// Diretivas de pré-processamento
```

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

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

Inclusão de bibliotecas auxiliares

```
// Função principal
```

```
int main(){
```

```
    comando 1;
```

```
    comando 2;
```

```
    ...
```

```
    comando n;
```

```
}
```

Função principal que é acionada pelo S.O



# Primeiro Programa em C

```
#include <stdio.h> // Biblioteca de entrada e saída de dados

int main(){
    printf("Hello world\n");
    return 0;
}
```



# Entrada e saída de dados

Comando leia: `scanf`

O comando **`scanf`** é composto por dois parâmetros, sendo o primeiro o **tipo de dados** e o segundo a **variável** para qual o valor será lido.

Sintaxe:

```
scanf("%d", &variavel);
```



## Entrada e saída de dados

onde:

“%d”: indica o formato (tipo) do dado que será recebido pelo fluxo de entrada. Neste caso (utilizando “%d”) será dado entrada de um valor do tipo **double**. **OBS:** Para maiores detalhes consultar outras opções para formatação de fluxo de entrada e saída de dados;

&: indica que será utilizado passagem de parâmetro por referência (endereço do identificador na memória)

**variável:** identificador da variável.





# Entrada e saída de dados

Comando escreva: **printf**

O comando **printf** é composto por dois parâmetros, sendo o **primeiro o tipo de dados** e o segundo a **variável** que será escrita.

Sintaxe:

```
printf("%d", variavel);
```



## Entrada e saída de dados

onde:

“%d”: indica o formato (tipo) do dado que será escrito pelo fluxo de saída. Neste caso (utilizando “%d”) será escrito um valor do tipo double. **OBS:** Para maiores detalhes consultar outras opções para formatação de fluxo de entrada e saída de dados;

**variável:** identificador da variável.



## Formatação de fluxo de entrada e saída de dados

Tipo de Dados	Formatação
char	“%c”
int	“%i”
float	“%f” ou “%.2f” para formatar casas decimais
char[]	“%s”



## Operadores lógicos, relacionais e aritméticos

Operador	Tipo
&&	<b>E</b> lógico
	<b>Ou</b> lógico
!	Não
==	Igual
!=	Diferente
>=	Maior ou igual a




## Operadores lógicos, relacionais e aritméticos

Operador	Tipo
<code>&lt;=</code>	Menor ou igual a
<code>/</code>	Divisão
<code>%</code>	Módulo - Resto da divisão inteira
<code>abs(a)</code>	Retorna o valor absoluto de um valor real. Pode ser usado para encontrar o quociente da divisão inteira
<code>pow(a,b)</code>	Potência de $a^b$ . É necessário incluir a biblioteca <b>math.h</b>
<code>sqrt(a)</code>	Raiz quadrada de $a$ . É necessário incluir a biblioteca <b>math.h</b>



## Estrutura condicional se-senão (if-else)

```
if ( <condição> ) {  
    comando 1;  
    comando 2;  
    ...  
    comando n;  
} else {  
    comando 1;  
    comando 2;  
    ...  
    comando n;  
}
```



## Estrutura condicional se-senão-se (if-else-if)

```
if ( <condição> ) {  
    comando 1;  
    comando 2;  
  
    ...  
    comando n;  
} else {  
    if ( <condição> ) {  
        comando 1;  
        comando 2;  
  
        ...  
        comando n;  
    } else {  
        comando 1;  
        comando 2;  
  
        ...  
        comando n;  
    }  
}
```



## Estrutura condicional escolha (switch-case)

```
switch (variável) {  
    case 1 :  
        comando 1;  
        comando n;  
    break;  
    case 2 ... 3 :  
        comando 1;  
        comando n;  
    break;  
    default :  
        comando 1;  
        comando n;  
}
```





## Estrutura de repetição enquanto (while)

Sintaxe:

```
while (<condição>) {  
    comando 1;  
    comando 2;  
    ...  
    comando n;  
}
```

Exemplo:

```
int cont = 0;  
while (cont < 10) {  
    printf("O valor do contador eh: %i\n", cont);  
    cont ++;  
}
```



## Estrutura de repetição repita (do-while)

Sintaxe:

```
do {  
    comando 1;  
    comando 2;  
    ...  
    comando n;  
} while (<condição>);
```

Exemplo:

```
int cont = 0;  
do {  
    printf("O valor do contador eh: %i\n", cont);  
    cont ++;  
} while (cont < 10);
```



## Estrutura de repetição para-faça (for)

Sintaxe:

```
for (valor_inicial; <condição>; passo) {  
    comando 1;  
    comando 2;  
    ...  
    comando n;  
}
```

Exemplo:

```
for (int cont=0; cont < 10; cont++) {  
    printf("O valor do contador eh: %i\n", cont);  
}
```



## Exemplo de programa em C

### Mínimo divisor comum (MDC.c)

O programa em C ao lado encontra o mínimo divisor comum entre dois números inteiros informados pelo usuário.

```
#include <stdio.h>
int main(){
    int n1, n2, aux, cont=1, mdc=1;
    printf("Digite n1: ");
    scanf ("%i", &n1);
    printf("Digite n2: ");
    scanf ("%i", &n2);
    if (n1 > n2){
        aux = n1;
        n1 = n2;
        n2 = aux;
    }
    while (cont <= n1){
        if (n1 % cont == 0 && n2 % cont == 0)
            mdc = cont;
        cont++;
    }
    printf("MDC de %d e %d é: %d\n", n1, n2, mdc);
    return 0;
}
```