



Professor  
Maromo

# LINGUAGEM DE PROGRAMAÇÃO

Material 002



GitHub  
maromo71

# C





# Agenda



## Estrutura de um programa

- Tipos de dados,
- Identificadores, declaração de variáveis,
- Palavras reservadas, operações básicas, comandos de entrada e saída

**Material: LP\_002**

# Tipos de Dados Básicos em C

- Há 05 tipos de dados básicos em C:
  - **char,**
  - **int,**
  - **float,**
  - **double e**
  - **void.**
- Outros tipos de dados são baseados nesses tipos.





# Tipo de dados em C



Tipo de Dado	Tamanho (bits)	Faixa (Range)
char	8	-128 a 127
unsigned char	8	0 a 255
short	16	-32,768 a 32,767
unsigned short	16	0 a 65,535
int	32	-2,147,483,648 a 2,147,483,647
unsigned int	32	0 a 4,294,967,295
long	32 ou 64	-2,147,483,648 a 2,147,483,647 ou -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807
unsigned long	32 ou 64	0 a 4,294,967,295 ou 0 a 18,446,744,073,709,551,615
float	32	1.2E-38 a 3.4E+38
double	64	2.3E-308 a 1.7E+308
long double	80 ou 128	3.4E-4932 a 1.1E+4932

# Modificando os tipos de dados

Segundo **Herbert(1997)** “um modificador é usado para alterar o significado de um tipo básico para adaptá-lo mais precisamente às necessidades de diversas situações”.





# Lista de modificadores de tipo

**Modificadores:** signed, short, long e unsigned.

Os modificadores podem ser aplicados aos tipos básicos caractere e inteiro.

Contudo long também pode ser aplicado a double.



# Identificadores

Em C, os identificadores são usados para nomear variáveis, funções, rótulos e outros objetos definidos pelo usuário. Eles devem começar com uma letra (a-z, A-Z) ou um sublinhado (\_) seguido por letras, dígitos (0-9) ou sublinhados. Aqui estão alguns exemplos de identificadores válidos:

**int contador;** – Aqui, "contador" é um identificador usado para nomear uma variável.

**double salarioMensal;** – "salarioMensal" é um identificador para uma variável de tipo double.

**void imprimirMensagem() { ... }** – "imprimirMensagem" é um identificador para uma função.

**\_temp** – Um identificador que começa com um sublinhado.

**MAX\_TAMANHO** – Geralmente, identificadores em letras maiúsculas são usados para constantes.



# Variáveis

Em C, uma variável é um nome dado a uma área de armazenamento que nosso programa pode manipular. Cada variável em C tem um tipo específico, que determina o tamanho e o layout do armazenamento da variável, a faixa de valores que podem ser armazenados nela e o conjunto de operações que podem ser aplicadas a ela.

## Exemplos de Definição de Variáveis:

### **Inteiro (int)**

```
int idade;
```

### **Ponto flutuante (float)**

```
float salario;
```

### **Caractere (char)**

```
char primeiraLetra;
```

### **Double (para precisão dupla)**

```
double saldoBancario;
```





# Onde declaramos as variáveis



**1. Dentro de funções**



**2. Definição de parâmetros das funções**



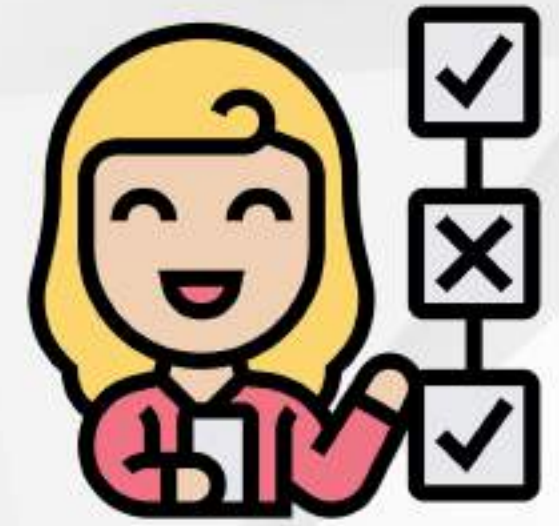
**3. Fora de todas as funções**

**A declaração de uma variável deve ser feita antes de sua utilização e antes de qualquer instrução.**





# Nome de variáveis (Regras)



Conjunto de regras para definição de nomes de variáveis:

- O nome de uma variável deve ser constituído por letras do alfabeto (maiúsculas e minúsculas).
- Maiúsculas e minúsculas representam caracteres diferentes, logo variáveis distintas.
- O primeiro caracteres não pode ser um dígito. Pode ser uma letra, ou o caractere underscore.
- Uma variável não pode ter por nome uma palavra reservada da linguagem C.





# Exemplos de uso



```
int idade;          /* Correto */
int Num_Cliente;    /* Correto */
float alB2c3;       /* Correto */
float 7a2b3c;       /* INCORRETO: primeiro caractere é um dígito */
char float;         /* INCORRETO: utilizou-se uma palavra reservada */
double vinte%;      /* INCORRETO: utilizou-se caractere inadmissível */
char sim?não;       /* INCORRETO: utilizou-se caractere inadmissível */
int _alfa;          /* Correto, mas não aconselhável */
int _123;           /* Correto, mas não aconselhável */
                   /* Notar que o primeiro caractere não é um dígito */
                   /* mas sim o underscore */
char Num, NUM;      /* Correto, pois o C é case sensitive. */
                   /* Será aconselhável ??? */
```



# Variáveis - continuação

- Sempre que uma variável é declarada, estamos solicitando ao compilador para reservar um espaço em memória para armazená-la.
- Esse espaço passará a ser referenciado por esse nome da variável.

**Nota: Quando uma variável é declarada fica sempre com um valor, o qual é o resultado do estado aleatório dos bits que a constituem.**



# Variáveis - continuação



- O "**Nome da Variável**" é o que usamos no código para se referir à variável.
- O "**Endereço de Memória**" é o local físico na memória RAM do computador onde o valor da variável é armazenado.
- O "**Valor da Variável**" é o dado que armazenamos na variável, e ele reside no endereço de memória.

Nome	Conteúdo	Endereço
a	20	1000
b	10	1004

# Operações com números inteiros

Operação	Descrição	Exemplo	Resultado
+	Adiciona dois operandos.	$A + B$	25
-	Subtrai o segundo operando do primeiro.	$A - B$	17
*	Multiplica ambos os operandos.	$A * B$	84
/	Divide o numerador pelo denominador.	$A / B$	5
%	Operador de módulo e resto após uma divisão inteira.	$A \% B$	1
++	Operador de incremento aumenta o valor inteiro em um.	$A++$	25
--	Operador de decremento diminui o valor inteiro em um.	$A--$	24

**Nota: No exemplo acima, assumimos que a variável A contém o valor 21 e a variável B contém o valor 4.**

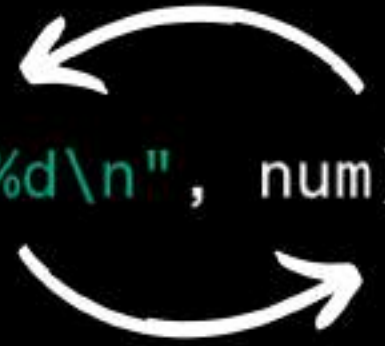


## Formato de escrita de um número inteiro:

A função printf é usada para imprimir valores formatados na tela. Para imprimir números inteiros, usamos o especificador de formato %d. Aqui está uma explicação detalhada:

```
#include <stdio.h>

int main() {
    int num = 12345;
    printf("O valor de num é: %d\n", num);
    return 0;
}
```

A diagram consisting of two white curved arrows on a black background. One arrow starts from the variable 'num' in the line 'int num = 12345;' and points to the '%d' format specifier in the line 'printf("O valor de num é: %d\n", num);'. The second arrow starts from the '%d' and points back to the variable 'num', indicating the substitution of the variable's value into the format string.

A função scanf é uma das funções mais usadas em C para ler dados da entrada padrão (geralmente o teclado). Ela pertence à biblioteca padrão stdio.h e é usada para ler dados formatados.



## Função para leitura [scanf]

A função scanf é usada em C para ler valores formatados do teclado (entrada padrão). Para ler números inteiros, usamos o especificador de formato %d. Aqui está uma explicação detalhada:

```
#include <stdio.h>

int main() {
    int num;
    printf("Digite um número inteiro: ");
    scanf("%d", &num);
    printf("Você digitou: %d\n", num);
    return 0;
}
```

O especificador de formato **%d** é usado para imprimir valores de números inteiros. Quando você usa **%d** dentro da string de formato de printf, a função espera que você forneça um valor inteiro correspondente.



## Comando sizeof

O comando **sizeof** é um operador em C que retorna o tamanho **(em bytes)** de um tipo de dado ou variável. É frequentemente utilizado para determinar o tamanho de tipos de dados ou de estruturas, o que pode ser útil em várias situações, como a alocação dinâmica de memória.

```
size_t tamanho = sizeof(int);
```

**No livro "Linguagem C" de Luís Damas, o sizeof é abordado como uma ferramenta fundamental para entender o tamanho e a alocação de memória dos tipos de dados em C.**





## Prefixo para inteiros

```
#include <stdio.h>
int main() {
    // Inteiros sem sinal
    unsigned int u_val = 123U;
    // Inteiros longos
    long int l_val = 123456789L;
    // Inteiros longos longos
    long long int ll_val = 12345678901234567LL;
    // Octal (base 8)
    int oct_val = 0123; // Representa o valor 83 em decimal

    // Hexadecimal (base 16)
    int hex_val = 0x1A3; // Representa o valor 419 em decimal

    printf("Unsigned int: %u\n", u_val);
    printf("Long int: %ld\n", l_val);
    printf("Long long int: %lld\n", ll_val);
    printf("Octal (como decimal): %d\n", oct_val);
    printf("Hexadecimal (como decimal): %d\n", hex_val);

    return 0;
}
```






## Exemplo

Escreva um programa que calcule o perímetro e a área de uma circunferência.

$$\text{Área} = \text{PI} * \text{Raio} * \text{Raio}$$

$$\text{Perímetro} = 2 * \text{PI} * \text{Raio}$$



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

```
// Definindo a constante PI
```

```
#define PI 3.14159265358979323846
```

```
int main() {
```

```
    double raio, area, perimetro;
```

```
    // Solicitando ao usuário o raio da circunferência
```

```
    printf("Digite o raio da circunferência: ");
```

```
    scanf("%lf", &raio);
```

```
    // Calculando área e perímetro
```

```
    area = PI * raio * raio;
```

```
    perimetro = 2 * PI * raio;
```

```
    // Exibindo os resultados
```

```
    printf("Área da circunferência: %.2lf\n", area);
```

```
    printf("Perímetro da circunferência: %.2lf\n", perimetro);
```

```
    return 0;
```

```
}
```



## Operações com números Reais

As operações básicas que você pode realizar com números reais em C incluem:

- Adição:  $x + y$
- Subtração:  $x - y$
- Multiplicação:  $x * y$
- Divisão:  $x / y$

## Operações com números Reais

Além dessas operações básicas, a biblioteca de matemática em C (`<math.h>`) fornece uma variedade de funções matemáticas para trabalhar com números reais, como:

- `sqrt()`: Calcula a raiz quadrada.
- `pow(x, y)`: Eleva  $x$  à potência de  $y$ .
- `sin()`, `cos()`, `tan()`: Funções trigonométricas.
- `log()`, `exp()`: Funções logarítmicas e exponenciais.





## Tipo char

- **Definição Básica:**

- O tipo char é usado para armazenar um único caractere, seja uma letra, um dígito, um símbolo de pontuação ou qualquer outro símbolo representável.

- **Tamanho e Representação:**

- Em C, um char tem um tamanho fixo de 1 byte, o que significa que pode representar 256 valores distintos.
- O intervalo desses valores pode ser de -128 a 127 (para char com sinal) ou de 0 a 255 (para char sem sinal, ou unsigned char).
- A representação exata e a faixa de valores dependem da plataforma e da implementação do compilador. A maioria das plataformas modernas usa o padrão ASCII para a representação de caracteres, mas outras codificações, como UTF-8, também são possíveis.



# Tipo char

- Exemplo:

```
char letra = 'A';  
printf("O caractere é: %c e seu valor ASCII é: %d\n", letra, letra);
```

# getchar() scanf()



## getchar

- **Função:** Esta função é usada para ler um único caractere de entrada.
- **Retorno:** Retorna o caractere lido como um valor inteiro. Se não houver mais caracteres disponíveis para leitura, retorna EOF (End Of File).
- **Uso:** Para ler um caracteres usando getchar(), você usa char variavel = getchar();



## scanf

- **Função:** Esta função é usada para ler a entrada formatada. Ela pode ler vários tipos de dados, incluindo, char, strings, números e mais.
- **Uso:** Para ler um caractere usando scanf(), você usaria o especificador de formato %c.





# Programa

Crie um programa em C que deve solicitar, através da função scanf, um caractere ao usuário e, em seguida, peça outro.

Depois de introduzirmos ambos os caracteres, o programa deve mostrar os dois caracteres lidos entre aspas simples.

# Programa

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

```
int main() {
```

```
    char char1, char2;
```

```
    // Solicitar o primeiro caractere
```

```
    printf("Digite o primeiro caractere: ");
```

```
    scanf(" %c", &char1); // Espaço antes do %c para ignorar espaços em branco ou novas linhas
```

```
    // Solicitar o segundo caractere
```

```
    printf("Digite o segundo caractere: ");
```

```
    scanf(" %c", &char2); // Espaço antes do %c para ignorar espaços em branco ou novas linhas
```

```
    // Mostrar os caracteres lidos entre aspas simples
```

```
    printf("Você digitou os caracteres '%c' e '%c'.\n", char1, char2);
```

```
    return 0;
```

```
}
```



## Cast

Em C, o termo "casting" refere-se à conversão explícita de um tipo de dado para outro. Um **cast** é usado quando você quer manualmente converter um tipo de variável em outro tipo, seja porque você sabe que a conversão é segura, ou porque você precisa satisfazer a tipagem de uma função ou operação específica.

```
float numero = 3.14;
```

```
int inteiro = (int) numero; // inteiro receberá o valor 3
```





## Exercícios - Aula 02

### 1. Tamanho de Tipos de Dados:

- Escreva um programa em C que imprima o tamanho (em bytes) dos tipos de dados básicos: **char**, **int**, **float**, **double** e **long double** usando a função **sizeof**.

### 2. Cálculo de Área e Perímetro de um Círculo:

- Desenvolva um programa que solicite ao usuário o raio de um círculo e calcule a área e o perímetro do círculo. Use as fórmulas discutidas anteriormente e exiba os resultados.

### 3. Leitura de Caracteres:

- Crie um programa que leia dois caracteres do usuário usando a função **scanf()**. Em seguida, exiba os caracteres lidos entre aspas simples, assim como no exemplo fornecido anteriormente.

### 4. Conversões de Tipo (Casting):

- Implemente um programa que:
  - Solicite ao usuário um número inteiro.
  - Converta esse número inteiro em um **float**.
  - Multiplique o **float** resultante por **0.5**.
  - Converta o resultado de volta para um inteiro (truncando a parte decimal).
  - Exiba o valor final ao usuário.



# Referências

---

**DAMAS, L. M. D. Linguagem C. LTC, 2007.**

---

**HERBERT, S. C completo e total. 3a. ed. Pearson, 1997.**

---

