

Curso de C

Declaração de Variáveis

Declaração de Variáveis

Objetivos:

- Aprender como:
 - Criar variáveis
 - Atribuir um valor inicial à variável
 - Escolher um nome para a variável

Declaração de Variáveis

Roteiro:

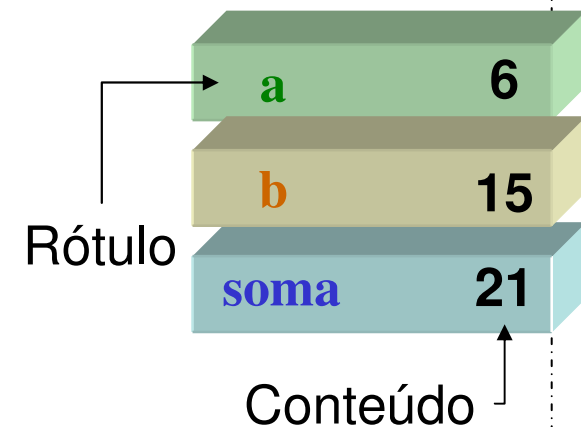
- Relembrando conceitos
- Tipos de Variáveis
- Declaração
- Identificadores

Conceitos

Variável: nome simbólico associado a um dado

Relembrando:

Variável: { Nome (rótulo)
Valor (conteúdo)



Conceitos

Variável em C:

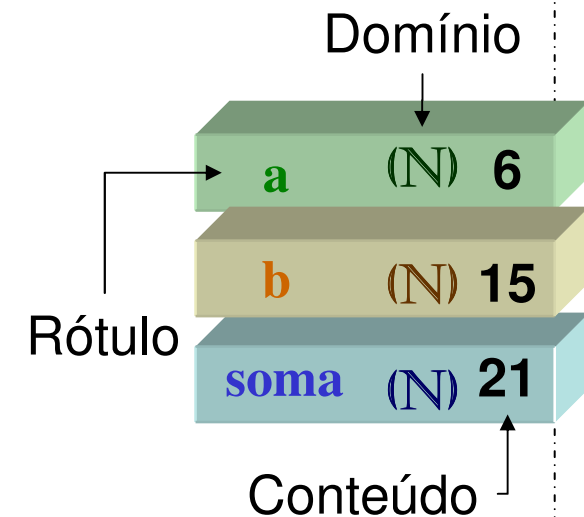
Variável:

Nome (rótulo)

Tipo (domínio)

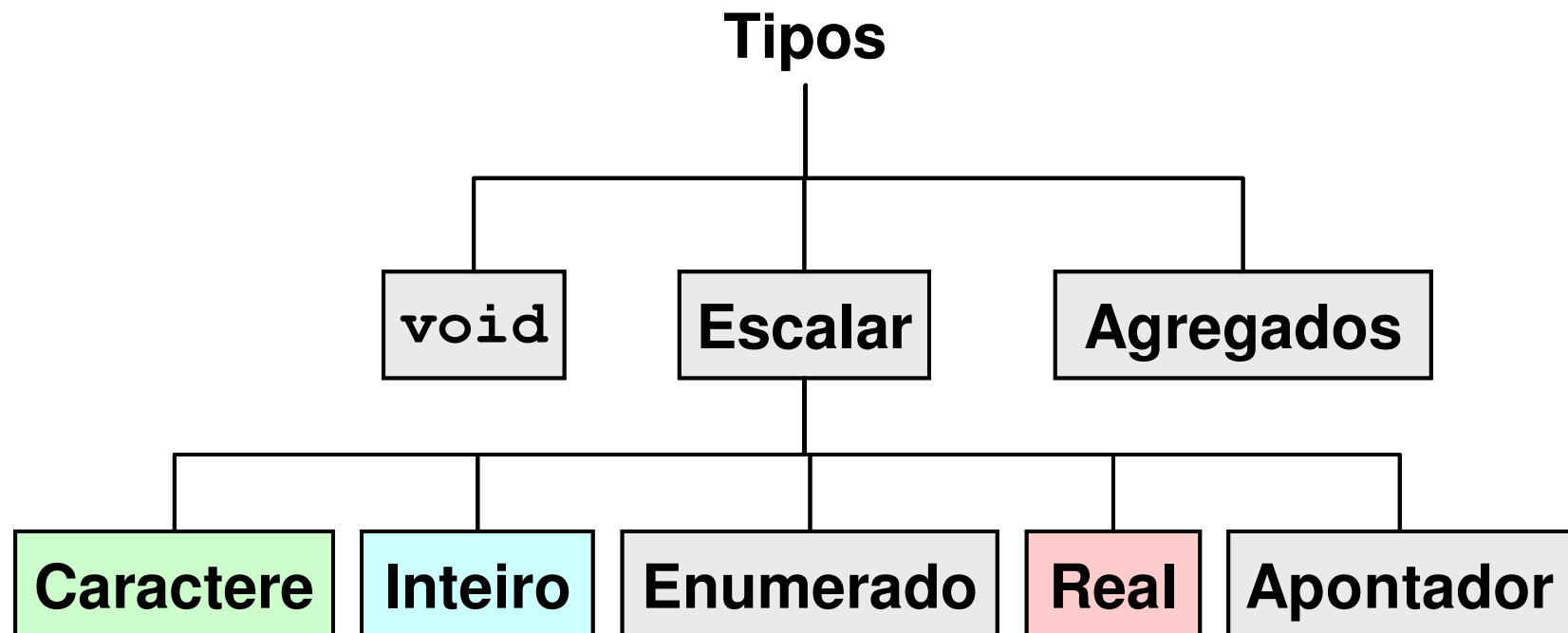
Valor (conteúdo)

Escopo (tempo de vida)



Tipos de Variáveis

Tipos da Linguagem C:



Declaração de Variável

Declaração:

- *Reservar espaço na memória*
- *Associar com identificador*

Sintaxe: *Valor inicial*

```
tipo nome = valor;
```

Domínio


Rótulo

Conteúdo

Sintaxe: *Sem valor inicial*

```
tipo nome;
```

Cuidado: valor inicial indefinido!




Declaração de Variável

Sintaxe: *Diversas variáveis, mesmo tipo*

```
tipo nome1, nome2, nome3;
```

Sintaxe: *Diversas variáveis, mesmo tipo*

```
tipo nome1 = valor, nome2;
```

Declaração de Variável

Exemplo:

```
float nota_prova_a = 8.0;  
float nota_prova_b = 6.0;  
float nota_laboratorio = 10.0;  
float media;
```

Identificadores

Nome de variável:

- Seqüência de:
 - Letras maiúsculas (A-Z)
 - Letras minúsculas (a-z)
 - Dígitos (0-9)
 - Sublinhado (_)

Não pode:

- Começar com dígito
- Ser uma palavra chave



Identificadores

Nome de variável:

Correto:

`contador``nota1``media``resto_divisao`

Errado:

`2lugares``média`



Identificadores

Nome de variável:

- Distinção maiúscula/minúscula
- Máximo 31 símbolos
- Palavras chaves (proibidas):

`auto, break, case, char, const, continue,
default, do, double, else, enum, extern,
float, for, goto, if, inline, int, long,
register, restrict, return, short,
signed, sizeof, static, struct, switch,
typedef, union, unsigned, void, volatile,
while`

Curso de C

Tipos Inteiros

Tipos Inteiros

Objetivos:

- Aprender a:
 - Declarar variáveis que armazenam números inteiros
 - Imprimir números inteiros na tela
 - Ler números inteiros digitados pelo usuário



Tipos Inteiros

Roteiro:

- O Tipo Inteiro
- Escrever texto na tela
- Escrever números na tela
- Ler números do teclado
- Exemplo
- Particularidades da leitura

Tipos Inteiros

A background image showing a close-up of a traditional abacus with black beads on wooden rods. Two hands are visible, with fingers positioned to move the beads. The image is slightly faded and serves as a backdrop for the text.

O Tipo Inteiro

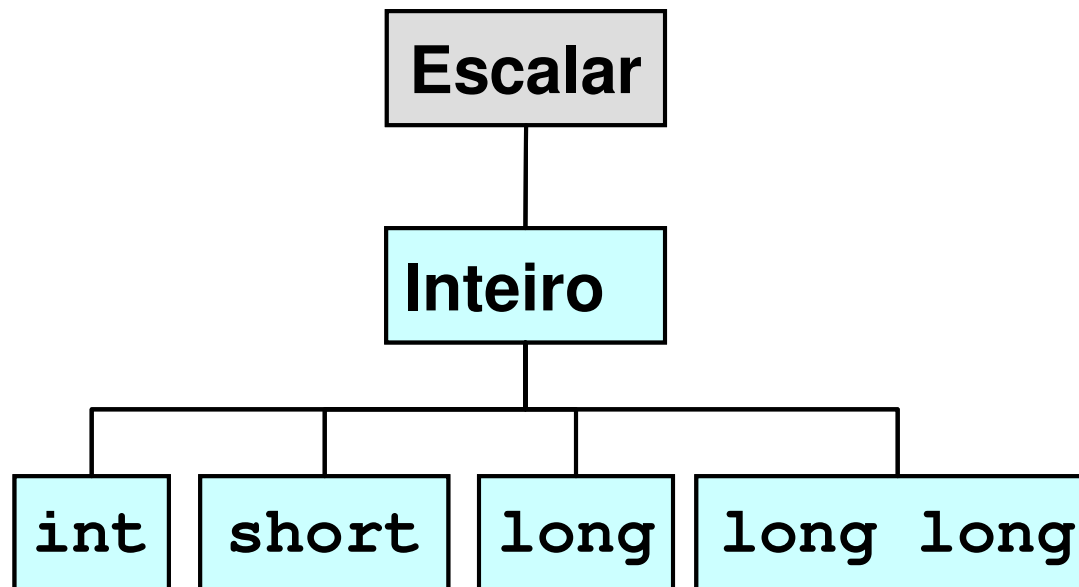
O Tipo Inteiro

Tipos Inteiros:

- Representação de números inteiros
Positivos e negativos
- Limitação de valor mínimo e máximo
Intervalo válido para números inteiros
- Compromisso:
Memória x Amplitude

O Tipo Inteiro

Tipos Inteiros: *Hierarquia*

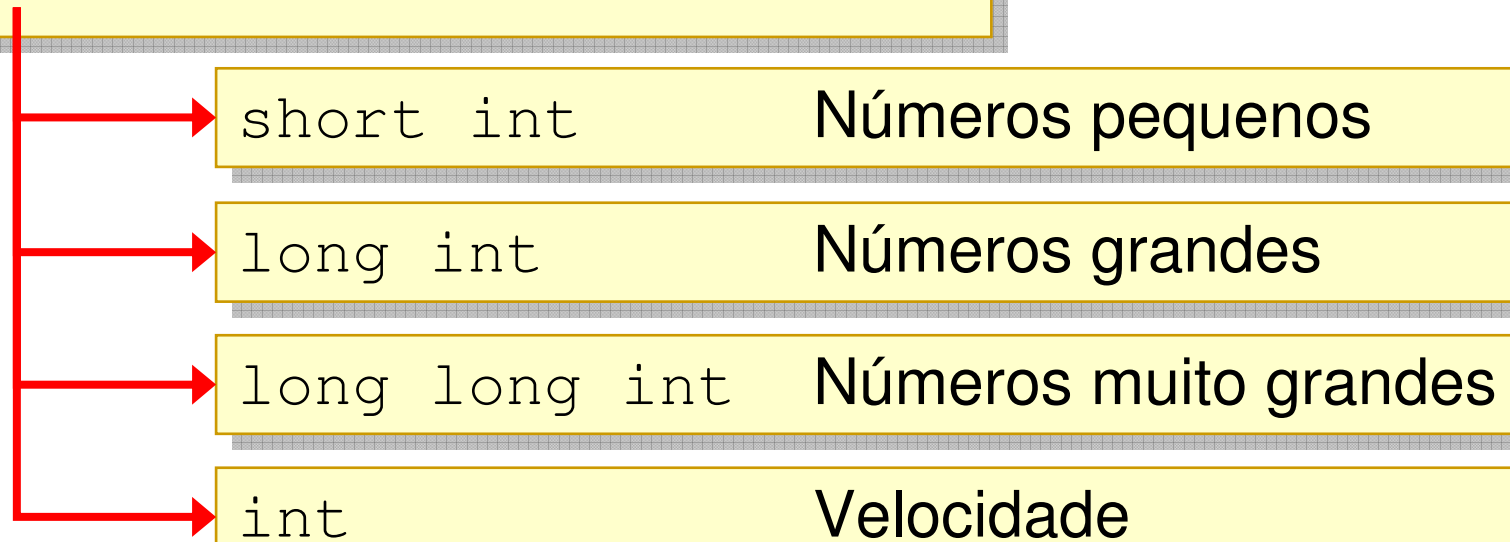



O Tipo Inteiro

Opções de Tipos Inteiros:

Declaração

```
tipo nome = valor;
```





O Tipo Inteiro


Opções de Tipos Inteiros:

Exemplos de declaração:

```
int contador;  
int limite_tentativas = 100;
```

```
short int numero_pequeno;  
short int contador = 4;
```

```
long int quantidade_pecas;  
long int numero_repeticoes = 5000000;
```



O Tipo Inteiro

Tipo	Descrição	Memória*	Intervalo*
<i>int</i>	Tamanho padrão	4 bytes	- 2.147.483.648 até 2.147.483.647
<i>short int</i>	Números pequenos	2 bytes	-32.768 até 32.767
<i>long int</i>	Números grandes	4 bytes	- 2.147.483.648 até 2.147.483.647
<i>long long int</i>	Números muito grandes	8 bytes	- $9,223 \cdot 10^{15}$ até $9,223 \cdot 10^{15}$

O Tipo Inteiro

Confuso?

“Utilize sempre o tipo **int** para declarar variáveis cujo conteúdo será um número inteiro.”

Tipos Inteiros

Escrever Texto



Escrever texto

Comando `printf()`

Sintaxe: *Mesma linha*

```
printf ( "mensagem" ) ;
```

Sintaxe: *Avançar para próxima linha*

```
printf ( "mensagem\n" ) ;
```




Escrever texto

Exemplo:

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


int main(int argc, char* argv[]) {
    printf("Primeira linha\n");
    printf("    Segunda linha\n");
    printf("Terceira linha");
    printf("continua terceira linha");

    return 0;
}
```

```
Primeira linha
    Segunda linha
Terceira linhacontinua terceira linha
```

Tipos Inteiros

*Escrever Números Inteiros
na Tela*



Escrever números inteiros

Indicador de escrita: %d

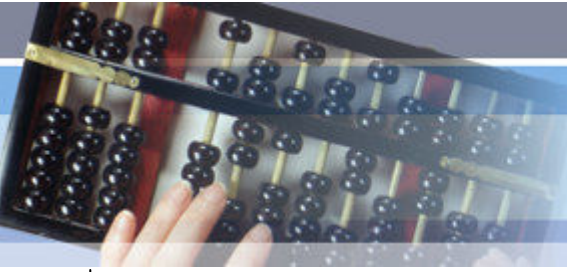
Sintaxe: *Uma variável*

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

Exemplo:

```
int q = 10;  
printf("Quantidade: %d itens", q);
```

```
Quantidade: 10 itens
```



Escrever números inteiros

Indicador de escrita: %d

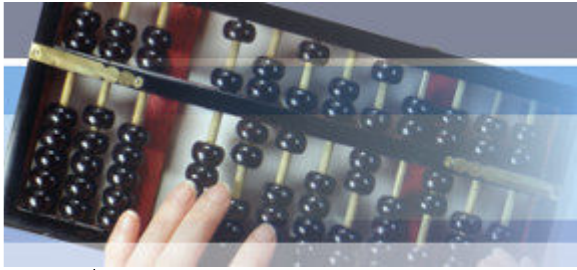
Sintaxe: *Mais variáveis*

```
printf("mensagem com varios %d", v1, v2 ...);
```

Exemplo:

```
int nota1 = 7;  
int nota2 = 8;  
printf("Primera nota: %d; segunda: %d.",  
      nota1, nota2);
```

Primeira nota: 7; segunda: 8.



Ler números inteiros

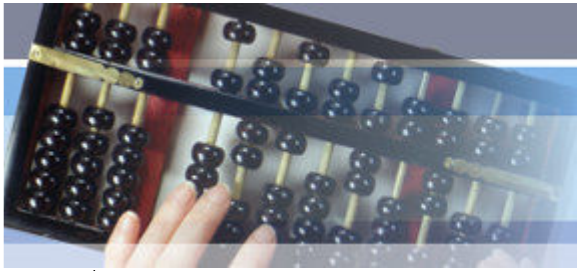
Comando `scanf()` com **%d**

Sintaxe: *Um número por comando*

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

Exemplo:

```
int quantidade;  
printf("Digite a quantidade: ");  
scanf("%d", &quantidade);
```



Ler números inteiros

Comando `scanf()` com `%d`

Sintaxe: *Vários números por comando*

```
scanf("formato com %d", &v1, &v1, ...);
```

Exemplo:

```
int nota1, nota2;  
printf("Digite as duas notas: ");  
scanf("%d %d", &nota1, &nota2);
```



Particularidades da leitura

Comando `scanf()` com **%d**

- Programa bloqueia até o usuário:
 - escrever todos os valores pendentes
 - pressionar ENTER.

```
int a, b, c;  
scanf("%d %d %d", &a, &b, &c);
```

O usuário poderá escrever:

3 4 6 (enter)

3 (enter)
4 6 (enter)

3 (enter)
4 (enter)
6 (enter)



Particularidades da leitura

Comando `scanf()` com **%d**

- Números digitados em excesso:
 - Ficam em uma fila para próximos `scanf`

```
int a, b, c, d, e;  
scanf("%d %d %d", &a, &b, &c);  
scanf("%d %d", &d, &e);
```

O usuário poderá escrever:

```
3 4 6 (enter)  
7 8 (enter)
```

```
3 4 6 7 8(enter)
```


Curso de C

Introdução aos Operadores Aritméticos

Introdução aos Operadores

Objetivos:

- Aprender como:
 - Realizar operações matemáticas com números e variáveis
 - Salvar novamente o resultado das operações em variáveis

Introdução aos Operadores

Roteiro:

- Atribuição
- Matemática
- Exemplo

Atribuição

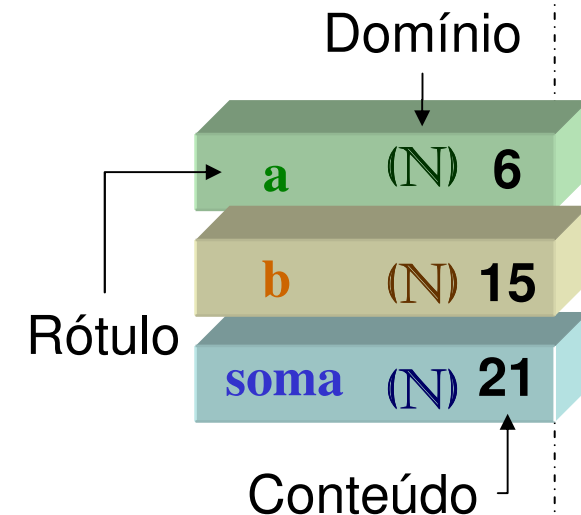
Atribuição: *Substitui o valor da variável*

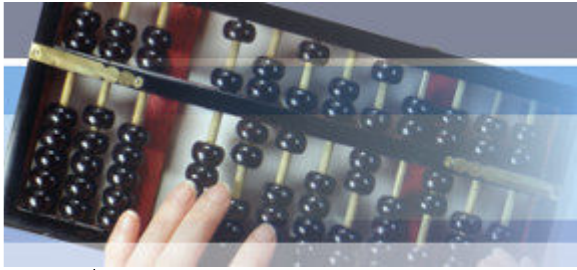
Sintaxe:

```
variavel = valor;
```

Sintaxe:

```
variavel = expressão;
```





Atribuição

Atribuição: *Substitui o valor da variável*

Atribuir um novo valor:

```
quantidade = 10;
```

Armazenar resultado de uma conta:

```
soma = valor_a + valor_b;
```

Atualizar um contador:

```
contador = contador + 1;
```

Matemática

Matemática:

- Operadores:
 - Soma
 - Subtração
 - Multiplicação
 - Divisão
 - Módulo (resto)
- Expressões



Matemática

Soma:

```
int parcela1 = 10, parcela2 = 16;  
int soma;  
soma = parcela1 + parcela2;  
printf("Soma: %d mais %d é %d",  
        parcela1, parcela2, soma);
```

Soma: 10 mais 16 é 26



Matemática

Subtração:

```
int parcela1 = 10, parcela2 = 16;  
int subtracao;  
subtracao = parcela1 - parcela2;  
printf("Subtração: %d menos %d é %d",  
       parcela1, parcela2, subtracao);
```

Subtração: 10 menos 16 é -6



Matemática

Multiplicação:

```
int fator_a = 4, fator_b = 6;  
int produto;  
produto = fator_a * fator_b;  
printf("Multiplicação: %d vezes %d é %d",  
      fator_a, fator_b, produto);
```

Multiplicação: 4 vezes 6 é 24



Matemática

Divisão inteira:

```
int dividendo = 46, divisor = 6;  
int quociente;
```

Divisão Inteira!
Sem parte fracionária

```
quociente = dividendo / divisor;  
printf("Divisão: %d por %d é %d",  
       dividendo, divisor, quociente);
```

Divisão: 46 por 6 é 7




Matemática

Resto:

```
int dividendo = 46, divisor = 6;  
int quociente, resto;  
quociente = dividendo / divisor;  
resto = dividendo % divisor;  
printf("Divisão: %d por %d é %d, resto %d",  
       dividendo, divisor, quociente, resto);
```

Divisão: 46 por 6 é 7, resto 4



Exemplo

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[]) {

    int horas, minutos, segundos;
    int total_segundos;

    printf("Digite o intervalo de tempo (segundos): ");
    scanf("%d", &total_segundos);

    horas      = (total_segundos / 60) / 60;
    minutos    = (total_segundos / 60) % 60;
    segundos   = total_segundos % 60;

    printf("\n");
    printf("Total de segundos: %d \n", total_segundos);
    printf("Tempo: %d:%d:%d\n", horas, minutos, segundos);

    return 0;
}
```

Horario01/Horario02

Curso de C

Tipo Caractere

Tipo Caractere

Objetivos:

- Aprender a:
 - Declarar variáveis que armazenam um caractere (letra ou símbolo)
 - Imprimir um caractere na tela
 - Ler um caractere digitado pelo usuário

Tipo Caractere

Roteiro:

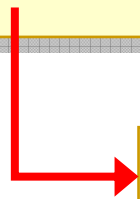
- O Tipo caractere
- Escrever caracteres na tela
- Ler caracteres do teclado

O Tipo Caractere

Única opção de Tipo Caractere:

Declaração

```
tipo nome = valor;
```



char

Caractere/Letra

O Tipo Caractere


Caractere vs Código ASCII:

Exemplos de declaração:

```
char letra = 'A';
```

```
char letra = 65;
```

Tabela ASCII
'A' equivale a 65



Escrever caracteres

Indicador de escrita: `%c`

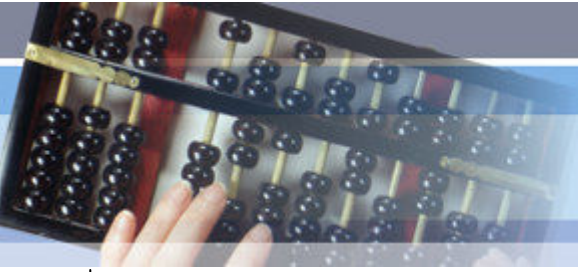
Sintaxe: *Uma variável*

```
printf("mensagem", variavel);
```

Exemplo:

```
char l = 'A';  
printf("Letra: %c", l);
```

```
Letra: A
```



Ler caracteres

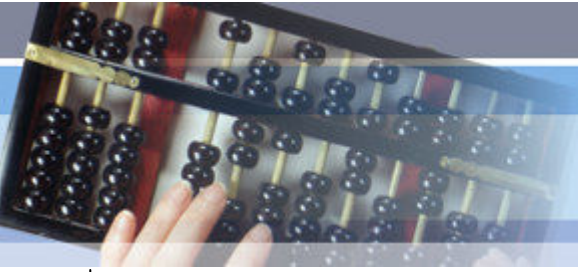
Comando `scanf()` com **`%c`**

Sintaxe: *Um número por comando*

```
scanf ("formato", &variavel);
```

Exemplo:

```
char letra;  
printf("Digite a letra: ");  
scanf ("%c", &letra);
```



Ler caracteres

Dica: Ler próxima letra

```
int numero;  
char letra;  
printf("Digite um número e uma letra: ");  
scanf("%d %c", &numero, &letra);
```

Ou:

```
scanf("%d", &numero);  
scanf(" %c", &letra);
```



Espaço!

Curso de C

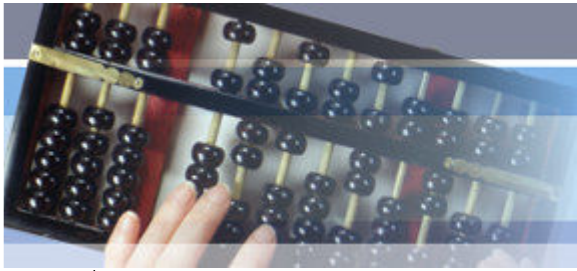
Outros Tipos Inteiros



Outros Tipos Inteiros

Objetivos:

- Aprender a:
 - Declarar variáveis que armazenam outros tipos de números inteiros
 - Imprimir estes números inteiros na tela
 - Ler estes números inteiros digitados pelo usuário



Outros Tipos Inteiros

Roteiro:

- Tipos com Sinal
- Tipos sem Sinal
- Escrever Inteiros sem Sinal
- Ler Inteiros sem Sinal

Outros Tipos Inteiros

Tipos Modificados:

Declaração:

```
int variavel;
```

→ Tamanho padrão
→ 4 bytes
→ - 2.147.483.648 até 2.147.483.647

Positivo e
negativo

Intervalo simétrico
de números



Outros Tipos Inteiros

Tipos com sinal:

Tipos inteiros conhecidos: (**com sinal**)

char

signed char

int

signed int

short int

signed short int

long int

signed long int

long long int

signed long long int

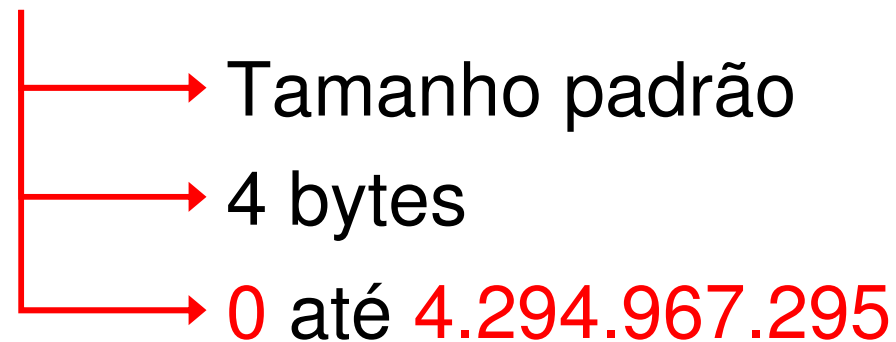
Declaração equivalente

Outros Tipos Inteiros

Tipos sem sinal:

Declaração:

```
unsigned int variavel;
```



Apenas
Positivo

Intervalo não
simétrico



Outros Tipos Inteiros

Tipos sem sinal:

Novos Tipos Inteiros: (**sem sinal**)

unsigned char

unsigned int

unsigned short int

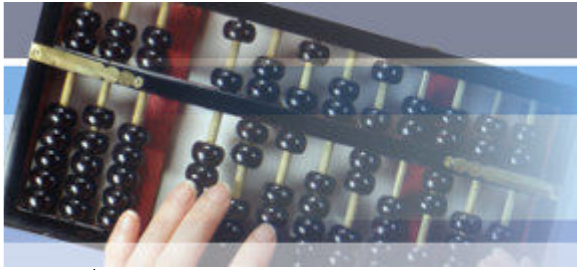
unsigned long int

unsigned long long int



Outros Tipos Inteiros

Tipo	Tamanho	Domínio
<i>(signed)</i> char	1 byte	- 128 até 127
<i>unsigned</i> char	1 byte	0 até 255
<i>(signed)</i> int	4 bytes	- 2.147.483.648 até 2.147.483.647
<i>unsigned</i> int	4 bytes	0 até 4.294.967.296
<i>(signed)</i> short int	2 bytes	- 32.768 até 32.767
<i>unsigned</i> short int	2 bytes	0 até 65.536
<i>(signed)</i> long int	4 bytes	- 2.147.483.648 até 2.147.483.647
<i>unsigned</i> long int	4 bytes	0 até 4.294.967.296
<i>(signed)</i> long long int	8 bytes	- $9,223 \cdot 10^{15}$ até $9,223 \cdot 10^{15}$
<i>unsigned</i> long long int	8 bytes	0 até $18,446 \cdot 10^{15}$




Outros Tipos Inteiros

Confuso?

“Utilize sempre o tipo `int` para declarar variáveis de número inteiro.”

Será adequado para 99% dos casos.



Escrever Inteiros sem Sinal

Indicador de escrita: %u

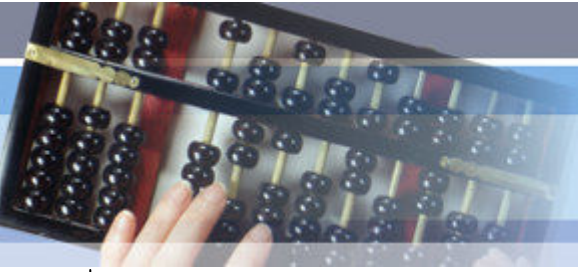
Sintaxe: *Uma variável*

```
printf("mensagem com %u", variavel);
```

Exemplo:

```
unsigned int n = 5000;  
printf("Quantidade: %u itens", n);
```

```
Quantidade: 5000 itens
```

Ler Inteiros sem Sinal

Comando `scanf()` **com** `%u`

Sintaxe: *Um número por comando*

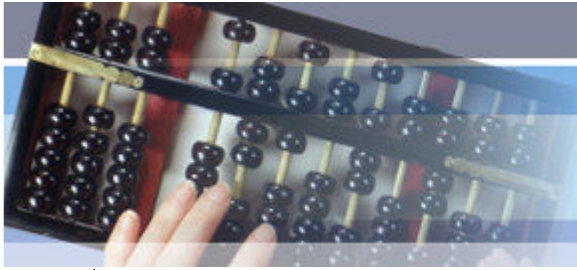
```
scanf("formato com %u", &variavel);
```

Exemplo:

```
unsigned int repeticoes;  
printf("Número de repetições: ");  
scanf("%u", &repeticoes);
```

Curso de C

Tipos de Ponto Flutuante (Números Fracionários)



Tipos de Ponto Flutuante


Objetivos:

- Aprender a:
 - Declarar variáveis que armazenam valores fracionários
 - Imprimir números fracionários na tela
 - Ler números fracionários digitados pelo usuário

Tipos de Ponto Flutuante

Roteiro:

- O tipo ponto flutuante
- Escrever número em ponto flutuante
- Ler número em ponto flutuante
- Exemplo

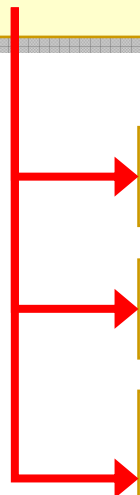


O Tipo Ponto Flutuante

Declaração de tipos ponto flutuante:

Declaração


```
tipo nome = valor;
```



float Pouca precisão, baixa magnitude

double Muita precisão, alta magnitude

long double
Precisão maior, altíssima magnitude



O Tipo Ponto Flutuante

Exemplo:

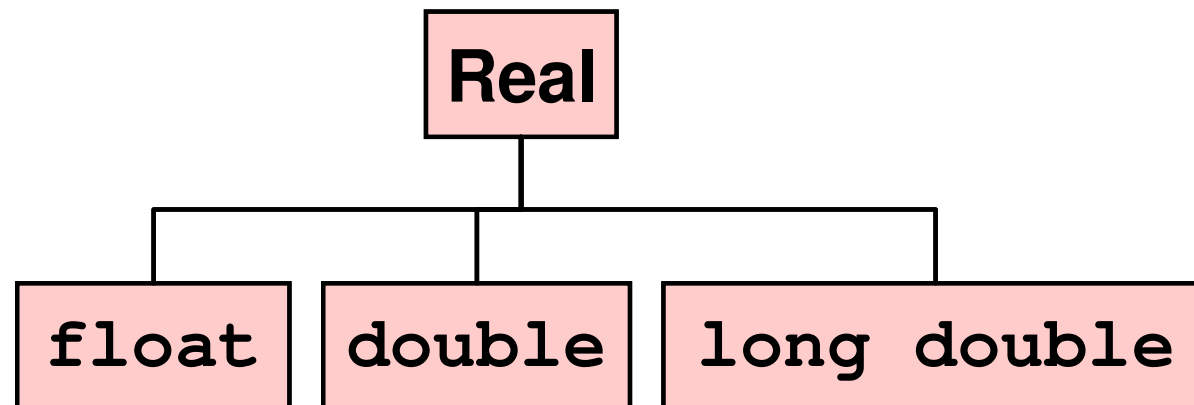
Exemplos de declaração:


```
float raio = 5.4;  
float area = 50040.22;
```

```
double velocidade = 5.333222567854;
```

O Tipo Ponto Flutuante

Tipos Ponto Flutuante: *Hierarquia*





O Tipo Ponto Flutuante

Tipo	Tamanho*	Precisão*	Intervalo*
<i>float</i>	4 bytes	7 dígitos	- $3,4 \cdot 10^{38}$ até $3,4 \cdot 10^{38}$
<i>double</i>	8 bytes	15 dígitos	- $1,7 \cdot 10^{308}$ até $1,7 \cdot 10^{308}$
<i>long double</i>	10 bytes	19 dígitos	- $1,2 \cdot 10^{4932}$ até $1,2 \cdot 10^{4932}$




O Tipo Ponto Flutuante

Confuso?

“Utilize sempre o tipo **double** para declarar variáveis de ponto flutuante.”

Será adequado para 99% dos casos.



Escrever números reais

Indicadores de substituição: %f

Sintaxe: *Uma variável*

```
printf("mensagem com %f", variavel);
```

Exemplo:

```
float v = 10.1;  
printf("Velocidade: %fkm/h", v);
```

```
Velocidade: 10.1km/h
```



Ler números reais


Comando `scanf()` com **%f**

Sintaxe: *Um número por comando*

```
scanf("formato com %f", &variavel);
```

Exemplo:

```
float nota;  
printf("Digite a nota da prova: ");  
scanf("%f", &nota);
```



Exemplo

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[]) {

    double pi = 3.141592;
    double raio, area, perimetro;

    printf("Digite o raio: ");
    scanf("%lf", &raio);

    area = pi * (raio * raio);
    perimetro = 2.0 * pi * raio;

    printf("\n");
    printf("Raio: %lf \n", raio);
    printf("Área: %lf \n", area);
    printf("Perímetro: %lf \n", perimetro);

    return 0;
}
```

Circulo01