

MC-102 — Aula 02  
Variáveis, Atribuições, Escrita/Leitura e Estrutura  
Básica

Alexandre M. Ferreira

IC – Unicamp

2017

# Roteiro

- 1 Variáveis
- 2 Atribuição
- 3 Estrutura de um Programa em C
- 4 Saída de dados: printf

## Definição

Variáveis são locais onde armazenamos valores. Toda variável é caracterizada por um nome, que a identifica em um programa, e por um tipo, que determina o que pode ser armazenado naquela variável.

- Durante a execução do programa, um pedacinho da memória corresponde à variável.

# Declarando uma variável

Declara-se da seguinte forma: **Tipo\_Variável Nome\_Variável;**

Exemplos corretos:

- `int soma;`
- `float preco_abacaxi;`
- `char resposta;`

Exemplos incorretos:

- `soma int;`
- `float preco_abacaxi`

# Variáveis inteiras

Variáveis utilizadas para armazenar valores inteiros. Ex: 13 ou 1102 ou 24.

Abaixo temos os **tipos da linguagem C** que servem para armazenar inteiros:

- **int**: Inteiro cujo comprimento depende do processador. É o inteiro mais utilizado. Em processadores Intel comum, ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647.
- **unsigned int**: Inteiro cujo comprimento depende do processador e que armazena somente valores positivos. Em processadores Intel comum, ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295.

# Variáveis inteiras

- **long int:** Inteiro que ocupa 64 bits em computadores Intel de 64bits, e pode armazenar valores de aprox.  $-9 \times 10^{18}$  a aprox.  $9 \times 10^{18}$ .
- **unsigned long int:** Inteiro que ocupa 64 bits em computadores Intel de 64bits, e armazena valores de 0 até aprox.  $18 \times 10^{18}$ .
- **short int:** Inteiro que ocupa 16 bits e pode armazenar valores de -32.768 a 32.767.
- **unsigned short int:** Inteiro que ocupa 16 bits e pode armazenar valores de 0 a 65.535.

# Variáveis inteiras

Exemplos de declaração de variáveis inteiras:

- `int numVoltas;`
- `int ano;`
- `unsigned int quantidadeChapeus;`

Exemplos Inválidos:

- `int int numVoltas;`
- `unsgned int ano;`

# Variáveis inteiras

Você pode declarar várias variáveis de um mesmo tipo. Basta separar as variáveis por vírgula:

- `int numVoltas , ano;`
- `unsigned int a, b, c, d;`



# Variáveis de tipo caractere

Variáveis utilizadas para armazenar letras e outros símbolos existentes em textos. OBS: Guarda apenas um caractere.

Exemplos de declaração:

- `char umaLetra;`
- `char YorN;`

## Variáveis de tipo ponto flutuante

Armazenam valores reais. Mas possuem problemas de precisão pois há uma quantidade limitada de memória para armazenar um número real. Exemplos de números em ponto flutuante: 2.1345 ou 9098.123.

- **float:** Utiliza 32 bits, e na prática tem precisão de aproximadamente 6 casas decimais (depois do ponto). Pode armazenar valores de  $(+/-)10^{-38}$  a  $(+/-)10^{38}$
- **double:** Utiliza 64 bits, e na prática tem precisão de aproximadamente 15 casas decimais. Pode armazenar valores de  $(+/-)10^{-308}$  a  $(+/-)10^{308}$

# Variáveis de tipo ponto flutuante

Exemplos de declaração de variáveis de tipo ponto flutuante.

- float salario;
- float resultado, cotacaoDolar;
- double a, b, c;

# Regras para nomes de variáveis em C

- **Deve** começar com uma letra (maiuscula ou minuscula) ou subcrito(\_). **Nunca** pode começar com um número.
- Pode conter letras maiúsculas, minúsculas, números e subcrito.
- Não pode-se utilizar como parte do nome de uma variável:

{ ( + - \* / \ ; . , ?

- Letras maiúsculas e minúsculas são diferentes:

```
int c;
```

```
int C;
```

## Regras para nomes de variáveis em C

As seguintes palavras já tem um significado na linguagem C e por esse motivo não podem ser utilizadas como nome de variáveis:

auto	double	int	struct	break
enum	register	typedef	char	extern
return	union	const	float	short
unsigned	continue	for	signed	void
default	goto	sizeof	volatile	do
if	static	while		

# Comando de Atribuição

## Definição

O comando de atribuição serve para atribuir valores para variáveis.

- A sintaxe do uso do comando é:

**variável = valor ;**

- Exemplos:

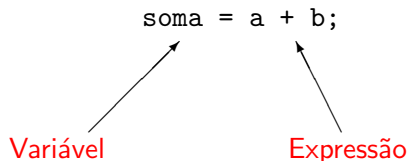
```
int a;  
float c;  
a = 5;  
c = 67.89505456;
```

# Comando de Atribuição

- O comando de atribuição pode conter expressões do lado direito:  
**variável = expressão ;**
- Atribuir um valor de uma expressão para uma variável significa calcular o valor daquela expressão e copiar aquele valor para a variável.

# Comando de Atribuição

No exemplo abaixo, a variável **soma** recebe o valor calculado da expressão **a + b**.





# Comando de Atribuição

- Exemplos:

```
int a;
```

```
float c;
```

```
a = 5 + 5 + 10;
```

```
c = 67.89505456 + 8 - 9;
```

# Atribuição

- O sinal de igual no comando de atribuição é chamado de **operador de atribuição**.
- Veremos outros operadores mais adiante.

À esquerda do operador de atribuição deve existir somente o nome de uma **variável**.

=

À direita, deve haver uma **expressão** cujo valor será calculado e armazenado na variável.

# Variáveis e Constantes

Constantes são valores previamente determinados e que por algum motivo, devem aparecer dentro de um programa.

- Assim como as variáveis, as constantes também possuem um tipo. Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo `string`, que corresponde a uma sequência de caracteres.
- Exemplos de constantes:

`85, 0.10, 'c', "Hello, world!"`

# Variáveis e Constantes

- Uma **constante inteira** é um número inteiro como escrito normalmente  
Ex: 10, 145, 1000000
- Uma **constante ponto flutuante** é um número real, onde a parte fracionária vem depois de um ponto  
Ex: 2.3456, 32132131.5, 5.0
- Uma **constante do tipo caractere** é sempre representada por um caractere (letra, dígito, pontuação, etc.) entre aspas simples.  
Ex: 'A', '!', '4', '('
- Uma **constante do tipo string** é um texto entre aspas duplas  
Ex: "Hello, world!"

# Expressões Simples

Uma constante é uma expressão e como tal, pode ser atribuída a uma variável (ou ser usada em qualquer outro lugar onde uma expressão seja válida).

- Ex1:

```
int a;  
a = 10;
```

- Ex2:

```
char b;  
b = 'F';
```

- Ex3:

```
double c;  
c = 3.141592;
```

# Expressões Simples

Uma variável também é uma expressão e pode ser atribuída a outra variável.

Ex:

```
int a, b;  
a = 5;  
b = a;
```

## Exemplos de atribuição

- **OBS:** A declaração de uma variável sempre deve ocorrer antes de seu uso.

```
int a,b;  
float f;  
char h;
```

```
a = 10;  
b = -15;  
f = 10.0;  
h = 'A';
```

```
a = b;  
f = a;  
a = (b+a);
```

Qual o valor final na variável **a**?

## Exemplos errados de atribuição

```
int a,b;  
float f,g;  
char h;
```

```
a b = 10; //Errado! Por quê?  
b = -15  
d = 90; //Errado! Por quê?
```



# Estrutura Básica de um Programa em C

A estrutura básica é a seguinte:

Declaração de bibliotecas Usadas

Declaração de variáveis

```
int main(){
```

Declaração de variáveis

Comandos

.  
. .  
. .

Comandos

}

# Estrutura Básica de um Programa em C

Exemplo:

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

```
int main(){
```

```
    int a;
```

```
    int b,c;
```

```
    a = 7+9;
```

```
    b = a+10;
```

```
    c = b-a;
```

```
}
```

## Escrevendo na tela

- Para imprimir um texto, utilizamos o comando **printf**. O texto pode ser uma constante do tipo string.

### Exemplo

```
printf("Olá Pessoal!");
```

Saída: Olá Pessoal!

- No meio da constante string pode-se incluir caracteres de formatação especiais. O símbolo especial `\n` é responsável por pular uma linha na saída.

### Exemplo

```
printf("Olá Pessoal! \n Olá Pessoal");
```

Saída: Olá Pessoal!

Olá Pessoal

## Escrevendo o conteúdo de uma variável na tela

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando **printf**. Para isso utilizamos símbolos especiais no texto, para representar que aquele trecho deve ser substituído por uma variável ou constante, e no final, passamos uma lista de variáveis ou constantes, separadas por vírgula.

### Exemplo

```
int a=10;
printf("A variável %s contém o valor %d", "a", a);
```

Saída: A variável a contém o valor 10

- Nesse caso, `%s` deve ser substituído por uma variável ou constante do tipo **string**, enquanto `%d` deve ser substituído por uma variável ou constante do tipo **int**.

# Formatos inteiros

`%d` — Escreve um inteiro na tela.

## Exemplo

```
printf ("%d", 10);
```

Saída: 10

## Exemplo

```
int a=12;
```

```
printf ("O valor e %d", a);
```

Saída: O valor e 12

# Formatos inteiros

- A letra **d** pode ser substituída pelas letras **u** e **ld**, quando desejamos escrever variáveis do tipo `unsigned` ou `long`, respectivamente.

## Exemplo

```
printf ("%d", 4000000000);
```

Saída: -294967296.

Enquanto que

```
printf ("%ld", 4000000000);
```

Saída: 4000000000.

# Formatos ponto flutuante

`%f` — Escreve um ponto flutuante na tela.

## Exemplo

```
printf ("%f", 10.0);  
Saída: 10.000000
```

# Formatos ponto flutuante

`%.Nf` — Escreve um ponto flutuante na tela, com  $N$  casas decimais.

## Exemplo

```
printf (".2f", 10.1111);
```

Saída: 10.11



# Formatos ponto flutuante

- O formato `%f` pode ser substituído por `%lf`, para escrever um **double** ao invés de um **float**.

## Exemplo

```
printf (".2lf", 10.0);
```

Saída: 10.00

## Formato caracter

`%c` — Escreve um caracter.

### Exemplo

```
printf ("%c", 'A');
```

Saída: A

Note que **`printf ("%c", 65)`** também imprime a letra A. Por quê?

# Formato **string**

`%s` — Escreve uma string

## Exemplo

```
printf ("%s", "Meu primeiro programa");
```

Saída: Meu primeiro programa