

# **Instrução de atribuição**

## **#**

# **Instruções de entrada/saída**

## Instrução de atribuição

### Definição

- Uma **instrução de atribuição** é uma forma de “colocar” um valor numa variável (pedaço de memória com um nome)
  - o valor é de qualquer tipo de dados
  - a variável é do mesmo tipo de dados do valor
- Uma instrução de atribuição consiste em guardar um valor de qualquer tipo na memória do computador, na zona reservada para a variável envolvida

## Sintaxe

- Linguagem C

**variável = expressão ;**

em que,

- a **variável** recebe o valor da *expressão*
  - a **variável** e a **expressão** têm que ser do mesmo tipo
    - exceção: se a variável é do tipo real, a expressão pode ser do tipo inteiro
  - colocar sempre um ponto e vírgula (;) no final da instrução
- Fluxograma

**variável ← expressão**

## Exemplo

- Inteiros

```
int a, b;  
a = 6/4;           // a = 1  
b = a + 3;         // b = 4  
a = b % 3 + a;     // a = 2
```

## Exemplo

- Reais

```
float x, y;  
x = 2.2 + 3 * 1.0;    // x = 5.2  
y = x / 2 + 5;        // y = 7.6  
x = y - x * 2;        // x = -2.8
```

## Exemplo

### - Carateres

```
char ch;  
ch = 'A';           // ch = A  
ch = 65;            // ch = A (ch recebe o carácter cujo código ASCII é 65)
```

- na linguagem C os carateres são valores inteiros guardados na memória em apenas 1 byte (8 bites)

## Observações

- O espaço de memória necessário para guardar um número (inteiro ou real) usando o tipo de dados "int" e "float" é normalmente 4 bytes (32 bits)
- Se for necessário pode-se utilizar tipos de dados que ocupam mais bytes (por exemplo, 8 bytes), para permitir manipular números
  - maiores (com um maior número de dígitos)
  - com mais precisão (no caso dos reais)

## Instruções de entrada

### Definição

- Entrada de dados para o programa (memória do computador)
  - é feita do dispositivo de entrada padrão (teclado) para as variáveis
  - a instrução que faz entrada de dados é a **função** predefinida **scanf()**, que se encontra definida na biblioteca "stdio.h" (*standard input/output*)
  - para a entrada apenas de dados do tipo carácter, pode-se também usar a **função** predefinida **getchar()**, que se encontra também definida na biblioteca "stdio.h"
- Uma outra forma de atribuir um valor a uma variável (leitura de valores)
  - a outra é através de uma instrução de atribuição

## Sintaxe

- Linguagem C

```
scanf ("formato_1 ... formato_N", &var_1, ..., &var_N) ;
```

em que,

**formato\_1** corresponde ao formato de leitura de dados do tipo da variável **var\_1**

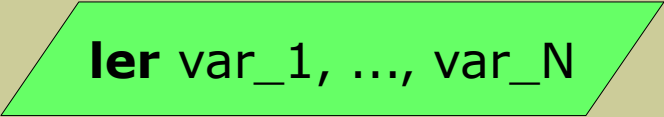
...

**formato\_N** corresponde ao formato de leitura de dados do tipo da variável **var\_N**

- as variáveis **var\_1, ..., var\_N**
  - são todas do tipo simples (inteiro, real e caráter)
  - podem ser de tipos diferentes
- todos os **formatos** devem ser especificados (não colocar reticências)
- todas as **variáveis** devem ser especificadas (não colocar reticências)

## Sintaxe

- Fluxograma



**ler** var\_1, ..., var\_N

em que var\_1, ..., var\_N

- são variáveis
- devem ser todas especificadas (não colocar reticências)

## Formato de leitura de valores inteiros

- Formato de leitura (para variáveis)

```
%d
```

- Exemplo

```
scanf("%d%d", &num1, &num2);
```

- são lidos (por esta ordem):
  - um valor inteiro para a variável do tipo inteiro **num1**
  - um valor inteiro para a variável do tipo inteiro **num2**
- note-se a necessidade de colocar **&** antes de cada variável (& = endereço de ...)
- as variáveis **num1** e **num2** devem estar (é obrigatório) previamente
  - declaradas como do tipo inteiro (int)
  - podem ou não já ter valores atribuídos

## Formato de leitura de valores reais

- Formato de leitura (para variáveis)

**%f**

- Exemplo

```
scanf("%f%f", &altura, &peso);
```

- são lidos (por esta ordem):
  - um valor real para a variável do tipo real **altura**
  - um valor real para a variável do tipo real **peso**
- note-se a necessidade de colocar **&** antes de cada variável (& = endereço de ...)
- as variáveis **altura** e **peso** devem estar (é obrigatório) previamente
  - declaradas como do tipo real (float)
  - podem ou não já ter valores atribuídos

## Formato de leitura de caracteres

- Formato de leitura (para variáveis)

```
%c
```

- Exemplo

```
scanf("%c%c", &resposta, &marca);
```

- são lidos (por esta ordem):
  - um carácter para a variável do tipo carácter **resposta**
  - um carácter para a variável do tipo carácter **marca**
- note-se a necessidade de colocar **&** antes de cada variável (& = endereço de ...)
- as variáveis **resposta** e **marca** devem estar (é obrigatório) previamente
  - declaradas como do tipo carácter (char)
  - podem ou não já ter caracteres atribuídos

## Exemplo geral

- Ler 4 valores de vários tipos

```
scanf("%c%d%f%d", &ch, &num1, &x, &num2);
```

- considere-se que os valores introduzidos são os seguintes:

```
A 12345 12.675 54
```

- estes valores ao serem lidos (por esta ordem), seriam atribuídos às variáveis da seguinte forma:
  - à variável do tipo carácter **ch** é atribuído o carácter '**A**'
  - à variável do tipo inteiro **num1** é atribuído o valor inteiro **12345**
  - à variável do tipo real **x** é atribuído o valor real **12.675**
  - à variável do tipo inteiro **num2** é atribuído o valor inteiro **54**
- as variáveis devem estar previamente declaradas como dos respetivos tipos:
  - **ch** do tipo carácter (char)
  - **num1** e **num2** do tipo inteiro (int)
  - **x** do tipo real (float)

## Instruções de Saída

### Definição

- Saída de dados do programa (memória do computador)
  - que é feito da memória para o dispositivo de saída padrão (monitor)
  - que podem ser de dois tipos:
    - valores (constantes e conteúdos de variáveis)
    - mensagens (textos)
  - a instrução que faz a saída de dados é a função predefinida **printf**
- A saída de dados pode ser de 3 de formas
  - só valores (constantes e conteúdos de variáveis)
  - só mensagens
  - mistura de mensagens e de valores (constantes e conteúdos de variáveis)

## Sintaxe - só valores (constantes e conteúdos de variáveis)

- Linguagem C

```
printf ("formato_1 ... formato_N", valor_1, ..., valor_N) ;
```

em que,

**formato\_1** corresponde ao formato de escrita de dados do tipo do **valor\_1**

...

**formato\_N** corresponde ao formato de escrita de dados do tipo do **valor\_N**

- os valores **valor\_1, ..., valor\_N**
  - correspondem a valores constantes ou a variáveis com valores
  - são todos do tipo simples (inteiro, real ou carácter)
  - podem ser de tipos diferentes entre si
- os **formatos** devem ser todos especificados (não colocar reticências)
- os **valores** devem ser todos especificados (não colocar reticências)

## Sintaxe - só valores (constantes e conteúdos de variáveis)

- Fluxograma

**escrever** valor\_1, ..., valor\_N

em que o **valor\_1, ..., valor\_N**

- correspondem a valores constantes ou a variáveis com valores
- devem ser todas especificadas (não colocar reticências)

## Sintaxe - só mensagens

- Linguagem C

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

- Fluxograma

```
escrever "mensagem"
```

## Sintaxe - mensagens e valores

- Linguagem C

```
printf ("... formato_1 ... formato_N ...", valor_1, ..., valor_N) ;
```

em que,

**formato\_1** corresponde ao formato de escrita de dados do tipo do **valor\_1**

...

**formato\_N** corresponde ao formato de escrita de dados do tipo do **valor\_N**

- os valores **valor\_1, ..., valor\_N**
  - correspondem a valores constantes ou a variáveis com valores
  - são todos do tipo simples (inteiro, real ou carácter)
  - podem ser de tipos diferentes entre si
- os **formatos** devem ser todos especificados (não colocar reticências)
- os **valores** devem ser todos especificados (não colocar reticências)
- as mensagens devem ser colocadas antes ou depois dos formatos de escrita dos valores

## Sintaxe - mensagens e valores

- Fluxograma

**escrever "... valor\_1 ... valor\_N ..."**

- Os valores **valor\_1**, ..., **valor\_N**
  - correspondem a valores constantes ou a variáveis com valores
  - são todos do tipo simples (inteiro, real ou caráter)
  - podem ser de tipos diferentes entre si
  - devem ser todos especificados (não colocar reticências)
- As mensagens devem ser colocadas antes ou depois da escrita dos valores

## Formato de escrita de valores inteiros

- Formato de escrita (números ou variáveis)

```
%d
```

- Exemplo 1 (só valores)

```
printf("%d %d %d", a, 8, f);
```

- são escritos (nesta ordem):
  - o valor da variável **a**
  - (espaço em branco)
  - o número **8**
  - (espaço em branco)
  - o valor da variável **f**
- as variáveis **a** e **f** devem estar (é obrigatório) previamente
  - declaradas como do tipo inteiro (int) e
  - com valores atribuídos
- notar a necessidade de colocar "espaço em branco" entre a escrita dos valores

## Formato de escrita de valores inteiros

- Exemplo 2 (mensagens e valores)

```
printf("a = %d %d f = %d\n", a, 8, f);
```

- são escritos (nesta ordem):
  - `a =` (mensagem)
  - o valor da variável `a`
  - (espaço em branco)
  - o número `8`
  - `f =` (mensagem)
  - o valor da variável `f`
  - mudar de linha (`\n`)
- as variáveis `a` e `f` devem estar (é obrigatório) previamente
  - declaradas como do tipo inteiro (`int`) e
  - com valores atribuídos

## Formato de escrita de reais

- Formato de escrita (números e variáveis)

**%f**

- Exemplo 1 (só valores)

```
printf("%f %f %f", x, peso, 4.2);
```

- são escritos (nesta ordem):
  - o valor da variável **x**
  - (espaço em branco)
  - o valor da variável **peso**
  - (espaço em branco)
  - o número **4.2**
- as variáveis **x** e **peso** devem estar (é obrigatório) previamente
  - declaradas como do tipo real (float) e
  - com valores atribuídos
- notar a necessidade de colocar "espaço em branco" entre a escrita dos valores

## Formato de escrita de reais

- Exemplo 2 (mensagens e valores)

```
printf("x = %f peso = %f %f\n", x, peso, 4.2);
```

- são escritos (nesta ordem):
  - `x =`
  - o valor da variável **x**
  - `peso =`
  - o valor da variável **peso**
  - (espaço em branco)
  - o número **4.2**
  - muda de linha (`\n`)
- as variáveis **x** e **peso** devem estar (é obrigatório) previamente
  - declaradas como do tipo real (float) e
  - com valores atribuídos

## Formato de escrita de caracteres

- Formato de escrita (símbolos e variáveis)

**%c**

- Exemplo

```
printf("%c %c %c\n", a, opcao, 'E');
```

- são escritos (nesta ordem):
  - o valor da variável **a**
  - (espaço em branco)
  - o valor da variável **opcao**
  - (espaço em branco)
  - o carácter/símbolo **E**
  - muda de linha (**\n**)
- as variáveis **a** e **opcao** devem estar (é obrigatório) previamente
  - declaradas como do tipo carácter (char) e
  - com caracteres atribuídos

## Exemplo geral

- Escrever 4 valores de vários tipos com mensagem

```
printf("CC: %d # Género: %c # Idade: %d # Altura: %f m", numCC, gen, idade, alt);
```

- considere-se que as variáveis envolvidas têm os seguintes valores:
  - a variável do tipo inteiro **numCC** tem o valor **1523427**
  - a variável do tipo carácter **gen** tem o valor **'F'**
  - a variável do tipo inteiro **idade** tem o valor **45**
  - a variável do tipo real **alt** tem o valor **1.65**
- a instrução em cima iria escrever no monitor o seguinte:

```
CC: 1523427 # Género: F # Idade: 45 # Altura: 1.65 m
```