

# Um valor pode ser

---

- **Constante**
  - Valor se manterá inalterado durante todo o programa
- **Variável**
  - Valor pode ser modificado durante a execução do programa

# Constante

---

- Posição de memória associada a um identificador
  - **identificador**: em geral, em letras maiúsculas
- Precisa ser declarada: 2 formas de declaração

global

```
1  #include <stdio.h>
2
3  # define PI 3.14159
4
5  int main(){
6      const int TAXA = 15;
7
8      printf("valor da primeira constante = %f\n", PI);
9      printf("valor da segunda constante = %d\n", TAXA);
10
11      return 0;
12  }
```

local

# Variável

---

- Posição de memória cujo conteúdo pode ser alterado
- Referência → identificador → regras
- Valor armazenado depende do tipo de dado
- Declaração obrigatória (reserva de memória): 2 formas

declara e  
inicializa

```
1  #include <stdio.h>
2
3  ▼ int main(){
4      int a;
5      float x = 5.0;
6
7      // restante do código
8
9      return 0;
10 }
```

somente  
declara

# Tipos de dados básicos

---

- Determinam o conjunto de valores e operações permitidas

char

int

float

double

- Cadeia de caracteres – “tipo” **string**
  - Tipo **bool** (biblioteca stdbool.h): define o tipo bool e seus valores TRUE e FALSE
  - Tipo **void** = vazio: usado como retorno/parâmetro de funções
-

# Declaração de variáveis

---

```
<tipo> <nome>;  
<tipo> <nome> = <valor>;
```

Exemplos:

```
int idade;  
float media;  
char resposta;  
int contador = 0;  
double salario = 12345.00;  
char cidade[15];  
char nome[10] = "Maria";
```

---

# Entrada e saída: printf() e scanf()

---

```
1  #include <stdio.h>
2
3  #define pi 3.14159
4
5  ▼ int main(){>>>    >>    >>
6    >>    float r, area;
7    >>    printf("\ndigite valor do raio: ");
8    >>    scanf("%f", &r);>>    >> // & - endereço>>
9    >>    area = pi * r * r;
10   >>    printf("\nvalor da area do circulo = %f\n\n", area);
11   >>    return 0;
12   }
13
```

**Operador =**  
usado para atribuição  
ex.: x = 0;  
a = a + 1;

# Comandos printf() e scanf()

---

- biblioteca `stdio.h`
- especificadores de formato (precedidos por %)
- *printf()*: escreve dados na saída padrão

```
printf("digite o valor do raio: ");  
printf("valor da area do circulo = %f\n", area);
```
- *scanf()*: lê dados da entrada padrão

```
scanf("%f", &r);           // & - endereço
```

# Especificadores de formato

---

## Formato

<code>%c</code>	<code>char</code>
<code>%d</code> ou <code>%i</code>	<code>int</code>
<code>%ld</code>	<code>long int</code>
<code>%f</code>	<code>float</code>
<code>%lf</code>	<code>double</code>
<code>%s</code>	<code>string</code>

## Códigos especiais

<code>\n</code>	<code>nova linha</code>
<code>\t</code>	<code>tabulação</code>
<code>\"</code>	<code>caractere “</code>
<code>\\</code>	<code>caractere \</code>
<code>%%</code>	<code>caractere %</code>

Para fixar o nro de casas decimais

`%.2f` → `exibe nº em ponto flutuante c/ 2 casas decimais`

---



# Exercícios: Faça um programa que...

---

- 1) Declare uma variável **x** como inteira. A seguir, atribua o valor 10 a **x**. Ao final, imprima **x**.
  - 2) Declare uma variável inteira, leia um valor para ela e a imprima.
  - 3) Declare uma variável inteira e uma variável real. A seguir, leia um número inteiro e um número real e os imprima. Utilize apenas 1 comando scanf (leitura das 2 variáveis) e 1 comando printf (escrita das 2 variáveis).
  - 4) Declare uma variável como string. A seguir, leia uma palavra e a imprima. Por último, imprima novamente a palavra, colocando-a entre aspas. (obs.: com string, o especificador é %s e o scanf dispensa o &)
  - 5) Leia seu primeiro nome e idade e os imprima, um em cada linha.
-

# Exercícios: Faça um programa que...

---

- 6) Declare 3 variáveis: uma inteira (x), uma real de precisão curta (y) e uma real de precisão longa (z). A seguir, leia valores para cada uma e as imprima, sendo y impressa com 3 casas decimais e z com 4.
- 7) A fim de observar como funciona a formatação, edite as seguintes linhas de código em um programa e observe o que acontece em cada printf:

```
printf("Os alunos são %d.\n", 350);  
printf("Os alunos são %2d.\n", 350);  
printf("Os alunos são %4d.\n", 350);  
printf("Os alunos são %6d.\n", 350);  
printf("Os alunos são %02d.\n", 350);  
printf("Os alunos são %04d.\n", 350);  
printf("Os alunos são %06d.\n", 350);  
printf("Os alunos são %10.02d.\n", 350);  
printf("Os alunos são %10.04d.\n", 350);  
printf("Os alunos são %10.06d.\n", 350);
```

---

# Exercícios:

---

8) Observe as instruções abaixo:

a) `printf("ab%dcd", 27);`

imprime o texto substituindo o `%d` pelo número contido após a vírgula, ou seja, imprime `ab27cd`

b) `printf("xy%3dwz", 65);`

faz o mesmo com o número ocupando 3 posições da linha, alinhado no lado direito, ou seja, imprime `xy 65wz`

c) `printf("*a%-4d*a", 58);`

faz o mesmo com o número ocupando 4 posições da linha, alinhado no lado esquerdo, ou seja, imprime `*a58 *a`

# Exercícios:

---

Cont. 8)

Com base nos exemplos anteriores, faça um programa que reproduza a figura abaixo, sem utilizar espaço em branco no printf:

```
*1111 1111*
*111   111*
*11    11*
*1     1*
*11    11*
*111   111*
*1111 1111*
```

# Exercícios: Faça um programa que...

---

- 9) Declare 2 constantes (PI e DOLAR) e atribua valores a elas. As declarações devem ser feitas de modos distintos. A seguir, imprima as constantes.
  
- 10) Declare uma constante denominada NUM, cujo valor deve ser 5. A seguir, imprima o valor do sucessor da constante e do antecessor da mesma, utilizando os operadores de soma e subtração.