
# Linguagem C

## Funções básicas, Strings e operadores

MsC. Douglas Santiago Kridi

Programação I - 2018.2

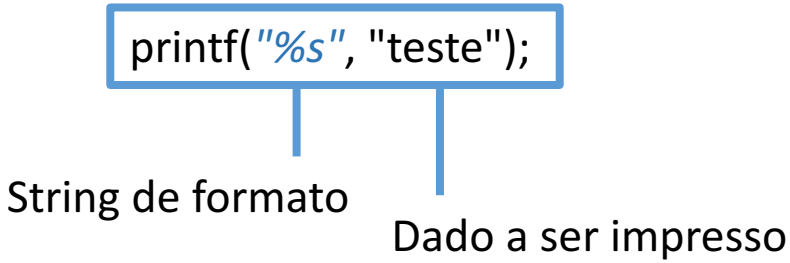
Bacharelado em Ciência da Computação

Universidade Estadual do Piauí

*douglaskridi@gmail.com*

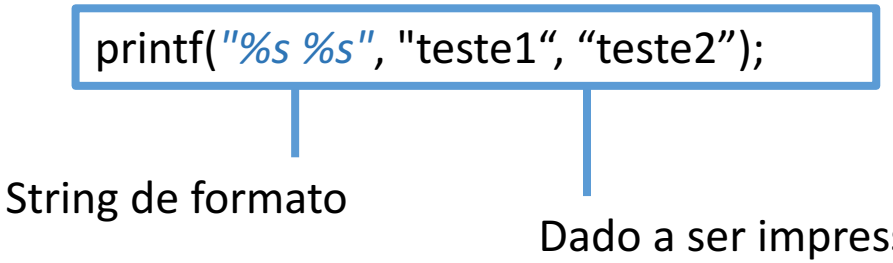
# Impressão: *printf*

- A função *printf* exibe um ou mais dados na tela. Para tanto ele deve receber pelo menos dois parâmetros, separados por vírgula:
  - Uma *string de formato* que define, através de caracteres especiais, os tipos dos dados a serem impressos e suas posições na linha de impressão;
  - Um dado a ser impresso. Este dado pode ser qualquer um dos dados vistos anteriormente.
- Por exemplo:



# Impressão: *printf*

- A string de formato define quais os tipos dos dados a serem impressos.
  - O símbolo *%s* será substituído pelo dado que vem após a vírgula.
  - Os dados definem quais os valores a serem impressos.
- Se for necessário, uma string de formato pode definir que *mais de um dado* será impresso.
  - Para tanto, dentro da string de formato deve haver *mais de um %*, um para cada dado a ser impresso.
  - Neste caso, os dados devem vir após a string de formato separados por vírgulas.




- Experimente colocar *\n* entre os *%s*.

# Impressão: *printf*

- Também é possível incluir um texto dentro da string de formato.
- Este texto irá aparecer exatamente como for digitado no código-fonte.

```
printf(“O aluno %s faltou", “Fulano“);
```

# Impressão: *printf*

- Para imprimir um ***inteiro*** com printf usa-se o símbolo ***%d***.

```
#include <stdio.h>

int main ()
{
  int Contador = 10;
  int NroDeFilhos = 3;

  printf("Valor da Variável: %d\n", Contador);
  // No momento da execução sinal %d vai  ser substituído pelo valor da variável Contador
  printf("Maria tem %d filhos", NroDeFilhos);
  // o inteiro pode ficar no meio da string
  return 0;
}
```

# Impressão: *printf*

- Impressão de Expressões aritméticas

```
#include <stdio.h>

int main ()
{
  int NroDeAndares = 7;
  int AlturaPorAndar = 3;

  printf("Altura Total do Prédio: %d metros", NroDeAndares*AlturaPorAndar);
            // No momento da execução o sinal %d vai ser substituído
            // pelo valor da multiplicação
}
```

# Impressão: *printf*

- Impressão de números reais

```
#include <stdio.h>

int main ()
{
  float NotaDaP1, NotaDaP2;
  float Media;
  NotaDaP1 = 6.6;
  NotaDaP2 = 8.2;
  Media = (NotaDaP1 + NotaDaP2) / 2.0;

  printf("Média Final : %f", Media);
            // No momento da execução o sinal %f vai ser substituído
            // pelo valor da variável Media com SEIS casas decimais
            // Média Final : 7.400000
  return 0;
}
```

# Impressão: *printf*

- Formato de Impressão dos Números Reais
  - No exemplo anterior o resultado da média (7.4) foi impresso com 6 casas decimais (7.400000).
  - Isto acontece pois o padrão da função *printf* é completar o número com zeros à direita, até que fique com seis casas decimais.
    - Podemos usar junto ao *%f* uma especificação de quantas casas decimais se deseja que o número tenha.
    - Especifica-se também o número total de caracteres do número a ser impresso.
  - Por exemplo:

    *%6.3f*

    - Especifica que se quer imprimir um float com *3* casas decimais e com um tamanho total de *6* caracteres.

# Impressão: *printf*

- Formato de Impressão dos Números Reais

```
float NotaDaP1, NotaDaP2;
  float Media;
  NotaDaP1 = 6.6;
  NotaDaP2 = 8.2;
  Media = (NotaDaP1 + NotaDaP2) / 2.0;

  printf("Média Final : %6.3f", Media);
            // No momento da execução sinal %6.3f vai ser substituído
            // pelo valor da variável Media
            // Média Final :  7.400
```

# Impressão: *printf*

- Formato de Impressão dos Números Reais
- Regras para impressão de um número real.
  - o número de casas decimais é sempre respeitado. Se for preciso, zeros serão acrescentados à direita do número.
  - o tamanho total significa o número de caracteres do número incluindo o ponto decimal e um eventual sinal de menos (-), se for o caso;
  - Se a soma do número total de caracteres ainda for menor do que o tamanho total especificado, então, espaços em branco serão acrescentados à *esquerda* da parte real do número.
  - Se a soma do número total de caracteres for maior do que o tamanho total especificado, então, apenas o número de casas decimais é respeitado.

# Impressão: *printf*

- Formato de Impressão dos Números Reais

```
printf("\n");

  int abc = 555;
  float def = 12.3456;

  printf("|%5d| \n", abc);
  //|  555|
  printf("|%05d| \n", abc);
  //|00555|
  printf("|%07.3f| \n", def);
  //|012.346|
  printf("|%07.2f| \n", def);
  //|0012.35|
```

# Impressão: *printf*

- Especificadores de conversão

| | |
|---|---|
| *%d* | Número decimal inteiro. Também pode ser usado *%i* como equivalente. |
| *%f* | Número decimal de ponto flutuante. |
| *%e* | Número em notação científica. |
| *%E* | Número em notação científica com o "e" maiúsculo. |
| *%c* | caractere. |
| *%s* | Sequência de caracteres (string, em inglês). |
| *%%* | Imprime um % |

  - Quando se usa scanf, temos %f (float) e %lf (double).

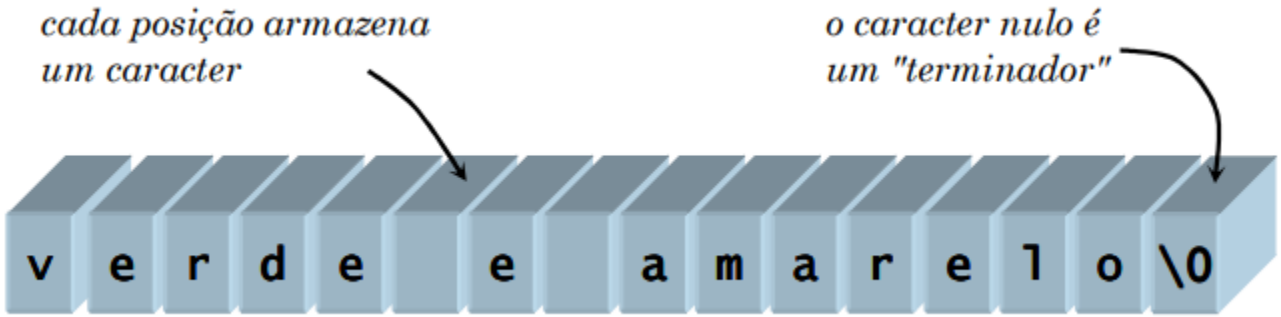
# Impressão: *printf*

- Sequências de escape
  - são combinações de caracteres que têm significado especial, e são sempre iniciadas por uma barra invertida (\).

| | |
|---|---|
| \n | Quebra de linha |
| \t | Tabulação horizontal |
| \a | Emite um sinal sonoro |
| \" | Aspa dupla |
| \' | Aspa simples |
| \\ | Barra invertida |
| \0 | Caractere nulo ( usado como finalizador de strings) |

# Strings

- Em C, uma *String* é uma série de caracteres terminada com um *caractere* nulo, representado por *\0*.
- Como um vetor é um tipo de dados capaz de armazenar uma série de elementos do mesmo tipo e a *String* é uma série de caracteres, é bastante natural que ela possa ser representada por um vetor de caracteres.



# Strings

- Devido à necessidade do *\0*, os vetores que armazenam *strings* devem ter sempre uma posição a mais do que o número de caracteres a serem armazenados.
- Quando a *string* é uma constante, o espaço adicional para o caractere *\0* é alocado automaticamente pelo compilador.
- No caso de *strings* variáveis, é responsabilidade do programador reservar esse espaço adicional.

# Strings

- Como qualquer outro vetor, *strings* também podem ser inicializadas quando são declaradas.
- Podemos usar a sintaxe própria para *strings*, na qual os caracteres são fornecidos entre aspas.
- O *caractere* nulo é incluído automaticamente.
- Exemplo:

```
char x[] = "um";
printf("%s ", x);
```

```
char n[21];
 printf("Qual o seu nome? ");
 gets(n);
 printf("Olá, %s!",n);
```

# Scanf()

- A função scanf() lê dados da entrada padrão (teclado) e os guarda em variáveis do programa.
- A sintaxe do scanf() é esta:

```
scanf ("string de formatação", &arg1, &arg2, ...);
```

- O *E* comercial *(&)* retorna o endereço de uma variável, sem ele, passamos apenas o valor de uma variável para a função.
- Isso é necessário pois a função scanf() deve modificar as variáveis.

# Scanf()

- Um exemplo básico da utilização de scanf():

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

  - Os dados só serão processados quando o usuário apertar Enter.
  - Os caracteres digitados pelo usuário serão *convertidos* para um valor inteiro e esse inteiro será guardado no endereço que corresponde à variável a.

- Pode receber vários valores, bastando usar vários especificadores de conversão.

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

```
char texto[30];
scanf(" %[^\n]", texto);
```

  - Lendo com espaços

# gets() e getchar()

- *gets()* e *getchar()*, assim como scanf(), lêem da entrada padrão.
- Não suportam formatação.
  - Como o nome sugere, getchar() lê apenas um caractere.
  - gets() lê uma string até o final da linha ou até que não haja mais dados para ler, e adiciona o terminador de string "\0".
- *A sintaxe das funções é:*

```
gets(ponteiro_para_string);
```

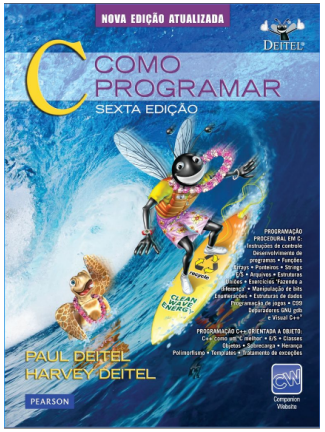
```
char c;
c = getchar();
```

- *Problema: estouro de buffer*

# Atividade

1. Efetuar a leitura de um número inteiro e apresentar o dobro desse número.
2. Ler 5 números inteiros e calcular o somatório e a média dos 5 números.
3. Implemente um programa para calcular a área de um trapézio.
   - *Fórmula: area = (altura * (base_menor + base_maior)) / 2*
4. Elaborar um programa que calcule e apresente o volume de uma caixa retangular.
   - *Fórmula: volume = comprimento * largura * altura*
5. Escreva um programa que solicite o número de segundos e, em seguida, indique quantas horas, minutos e segundos esse valor representa.
6. Faça um programa que calcule o ano de nascimento de uma pessoa a partir de sua idade e do ano atual.

# Referências Bibliográficas


**C: COMO PROGRAMAR**

DEITEL, Harvey M.; DEITEL, Paul J. Editora Pearson - 6ª ed. 2011


**Fundamentos da Programação de Computadores**

Ascencio, Ana F. G., Campos, Edilene A. V. de, - Editora Pearson 2012


**Lógica de Programação e Estrutura de Dados**

Puga, Sandra. Risseti, Gerson. – Ed. Pearson - 2016


**Lógica de Programação Algorítmica (Apostila)**

Guedes, Sergio. - Editora Pearson/Ser - 2014