

Fundamentos de Programação

Saída de Dados em C: uma breve introdução

Dainf - UTFPR


Profa. Leyza Baldo Dorini
Prof. Bogdan Tomoyuki Nassu

Saída de dados

Saída de dados: impressão de mensagens de texto

A função `printf` (*print formatted*) é utilizada para impressão de mensagens na saída padrão. A sintaxe é a seguinte:

```
1 printf("Fundamentos de programação 1");
```

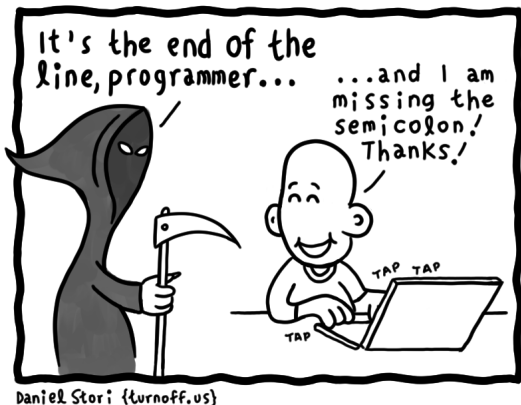


Sequência de caracteres (*string*)
(deve estar entre aspas)

Saída padrão é lugar para onde o programa envia suas saídas. Normalmente, um terminal de texto (a “janela preta”...).

Saída de dados: cuidado com a sintaxe

Cuidado com a sintaxe: é preciso abrir e fechar parênteses, a mensagem deve estar entre aspas e a declaração deve ser finalizada com ponto e vírgula!



Saída de dados: impressão de valores numéricos

Para imprimir constantes numéricas ou resultado de expressões aritméticas a sintaxe é um pouco diferente:

são dois argumentos,
separados por vírgula...

```
1 printf( "Mensagem %d", 5+3 );
```

Saída de dados: impressão de valores numéricos

Para imprimir constantes numéricas ou resultado de expressões aritméticas a sintaxe é um pouco diferente:

```
1 printf( "Mensagem %d", 5+3 );
```

após a vírgula, está a
constante numérica ou a
expressão aritmética cujo
resultado será impresso.

Saída de dados: impressão de valores numéricos

Para imprimir constantes numéricas ou resultado de expressões aritméticas a sintaxe é um pouco diferente:

```
1 printf( "Mensagem %d", 5+3 );
```



Na mensagem, no local em que você deseja que o resultado seja mostrado, é preciso incluir o **especificador de formato**, o qual deve ser compatível com o tipo do valor que terá seu conteúdo impresso:

%d para valores inteiros

%f para valores reais

Saída de dados: outro exemplo

Agora um exemplo com valores reais:

```
1 printf("5.5+3.2 é igual a %f", 5.5+3.2);
```

o especificador agora é %f

expressão aritmética cujo resultado é um número real


observe que, como esta expressão está entre aspas, será apenas impressa na tela (e não efetivamente calculada)

Saída de dados: impressão de múltiplos valores

É possível imprimir o resultados de quantos valores forem necessários! Para isso, basta inserir os especificadores de formato adequados nos locais desejados (ou seja, onde o resultado deve aparecer). Após a vírgula, colocar as expressões na ordem em que devem ser impressas. Exemplo:

expressões ou constantes numéricas
cujos valores serão impressos
(separados por vírgula).

```
1 printf( "A soma eh %d e uma constante eh %f", 5+2, 5.0 );
```



Saída de dados: impressão de múltiplos valores

É possível imprimir o resultados de quantos valores forem necessários!
Para isso, basta inserir os especificadores de formato adequados nos locais desejados (ou seja, onde o resultado deve aparecer). Após a vírgula, colocar as expressões na ordem em que devem ser impressas. Exemplo:

```
1 printf( "A soma eh %d e uma constante eh %f", 5+2, 5.0 );
```

↑
especificadores de formato, os quais devem
ser **compatíveis com o tipo (e a ordem)** dos resultados
das expressões que terão seu conteúdo impresso.

No caso do exemplo, como $5+2$
é um inteiro, o primeiro especificador é %d

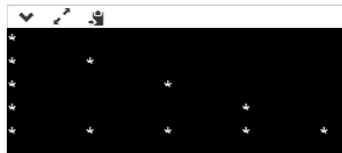
Caracteres de escape

A mensagem pode conter também caracteres de escape, tais como:

- `\n` - Nova linha
- `\t` - Tabulação horizontal
- `\a` - Sinal de alerta sonoro
- `\\` - Barra invertida


Exemplo:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("*\n");
6      printf("*\t*\n");
7      printf("*\t\t*\n");
8      printf("*\t\t\t*\n");
9      printf("*\t*\t*\t*\t*");
10     return 0;
11 }
```



Exemplo

a função `printf()` está declarada no arquivo `stdio.h`
(portanto, não se esqueça de incluir sua interface)



```
1 #include <stdio.h>
2 int main ()
3 {
4     printf ("%d %d\n", 2*3+4, 4+2*3);
5     printf ("%d + %d = %d\n", 22, 10, 22-10);
6     printf ("%d / %d = %d \n", 5, 2, 5/2);
7     printf ("%f / %f = %f\n", 5.0, 2.0, 5.0/2.0);
8     printf ("Valor = %f \n", 60/(1.6*1.2));
9     return 0;
10 }
```

Exemplo

as operações de * e / tem maior precedência que + e -. Portanto, neste exemplo, o resultado de ambas as expressões será o mesmo

```
1  #include <stdio.h>
2  int main ()
3  {
4      printf ("%d %d\n", 2*3+4, 4+2*3);
5      printf ("%d + %d = %d\n", 22, 10, 22-10);
6      printf ("%d / %d = %d \n", 5, 2, 5/2);
7      printf ("%f / %f = %f\n", 5.0, 2.0, 5.0/2.0);
8      printf ("Valor = %f \n", 60/(1.6*1.2));
9      return 0;
10 }
```

operações com mesma precedência são avaliadas da esquerda para a direita! Para alterar a ordem de avaliação das operações, use parênteses

Exemplo

```
1  #include <stdio.h>
2  int main ()
3  {
4      printf ("%d %d\n", 2*3+4, 4+2*3);
5      printf ("%d + %d = %d\n", 22, 10, 22-10);
6      printf ("%d / %d = %d \n", 5, 2, 5/2);
7      printf ("%f / %f = %f\n", 5.0, 2.0, 5.0/2.0);
8      printf ("Valor = %f \n", 60/(1.6*1 2));
9      return 0;
10 }
```

a operação de divisão tem resultados diferentes
ao operar inteiros e reais. Veja a discussão a seguir!

Atenção ao tipo dos valores!

Divisão (/)


Atenção!! A operação de divisão (/) possui dois modos de operação de acordo com os seus argumentos: inteira ou real (também denominada *de ponto flutuante*).

- Se os **dois argumentos forem inteiros**, acontece a divisão inteira. A expressão $(10 / 3)$ tem como resultado 3.
- Se **ao menos um** dos dois argumentos for **um valor real**, acontece a divisão de ponto flutuante. Portanto, as expressões $(10 / 3.0)$, $(10.0 / 3)$ e $(10.0 / 3.0)$ têm como resultado 3.333333.

Divisão inteira

Em outras palavras, quando os dois argumentos são inteiros, o resultado é a parte inteira da divisão (ou seja, o número inteiro obtido após descartar-se os dígitos depois do ponto decimal).

$$\begin{array}{r} 5 \\ 4 \\ \hline 1 \end{array} \quad \begin{array}{r} 2 \\ \hline \end{array}$$

2  $5 / 2$

Mas e se eu precisar do resultado da divisão?

Neste caso, você precisa que ao menos um dos argumentos seja real! Em C, basta adicionar `.0` à constante numérica.

Exemplo

```
1  #include <stdio.h>
2  int main ()
3  {
4      printf ("%d / %d = %d \n", 5, 2, 5/2);
5
6      printf ("%f / %f = %f \n", 5.0, 2.0, 5.0/2.0);
7
8      printf ("%f / %d = %f \n", 5.0, 2.0, 5.0/2);
9
10     return 0;
11 }
```

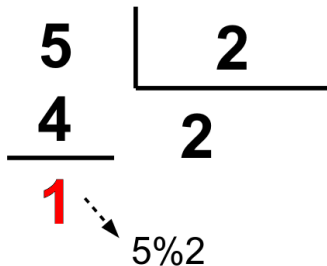
aqui, como a operação está sendo realizada entre argumentos inteiros, o resultado será a **parte inteira** da divisão

nos outros casos, como pelo menos um dos números não é inteiro, o valor resultante será real!

Atenção: cuidado com os especificadores de formato! Eles **precisam** ser compatíveis com os valores que serão impressos!!

Resto da divisão

Essa discussão nos leva a uma outra operação muito importante: o resto da divisão (ou módulo)! Em C, o operador para tal é %. Mas atenção: **essa operação só pode ser realizada entre inteiros!**



A handwritten calculation showing the division of 5 by 2. The number 5 is above a horizontal line, and 4 is below it. A vertical line to the right of the horizontal line separates the dividend from the divisor. The number 2 is to the right of the vertical line. Below the horizontal line, the number 1 is written in red. A dashed arrow points from the red 1 to the expression 5%2.

$$\begin{array}{r|l} 5 & 2 \\ \hline 4 & \\ \hline 1 & \end{array} \quad \text{5\%2}$$

Desafio

Você consegue pensar em um problema onde o módulo (ou resto da divisão) faça parte da solução?

Exemplo

Faça um programa que informe com quais cédulas podemos representar R\$ 218. Suponha que estejam disponíveis apenas notas de R\$ 50, R\$5 e R\$1.

```
1  #include<stdio.h>
2
3  int main()
4  {
5      printf("Notas de R$50: %d\n", 218 / 50);
6      printf("Notas de R$5: %d\n", 218 % 50 / 5);
7      printf("Notas de R$1: %d\n", 218 % 50 % 5);
8
9      return 0;
10 }
```

Exemplo

Faça um programa que informe com quais cédulas podemos representar R\$ 218. Suponha que estejam disponíveis apenas notas de R\$ 50, R\$5 e R\$1.

```
1  #include<stdio.h>
2
3  int main()
4  {
5      printf("Notas de R$50: %d\n", 218 / 50);
6      printf("Notas de R$5: %d\n", 218 % 50 / 5);
7      printf("Notas de R$1: %d\n", 218 % 50 % 5);
8
9      return 0;
10 }
```

Exercícios de fixação

Exercício de fixação 1

O programa abaixo deveria calcular o IMC (definido como o peso dividido por altura²). Entretanto, o resultado é inconsistente. Por quê?

```
1  #include<stdio.h>
2  int main(){
3
4      printf("Calculo do IMC \n");
5      printf("Altura: %f \n", 1.64);
6      printf("Peso: %f \n", 62.5);
7
8      printf("IMC: %f ", 62.5 / 1.64* 1.64);
9
10     return 0;
11 }
```

Exercício de fixação 1

O programa abaixo deveria calcular o IMC (definido como o peso dividido por altura²). Entretanto, o resultado é inconsistente. Por quê?

Os operadores / e * tem a mesma precedência. Como não usamos parênteses, eles são avaliados da esquerda para a direita. Com isso, o cálculo realizado é equivalente a (peso / altura) * altura (e não peso / (altura*altura), como esperado)

```
1  #include<stdio.h>
2  int main(){
3
4      printf("Calculo do IMC \n");
5      printf("Altura: %f \n", 1.64);
6      printf("Peso: %f \n", 62.5);
7
8      printf("IMC: %f ", 62.5 / 1.64* 1.64);
9
10     return 0;
11 }
```


Exercício de fixação 2

Quais das alternativas abaixo convertem corretamente 70 graus Fahrenheit (F) para graus Celsius (C)? Considere que:

$$C = \frac{5}{9} (F - 32) \quad (1)$$

- ① `printf("%f ", 5/9*(70-32));`
- ② `printf("%d ", 5/9*(70-32));`
- ③ `printf("%f ", (70-32.0)*5/9);`
- ④ `printf("%f ", 5.0/9*70-32);`
- ⑤ `printf("%d ", 5.0/9*(70-32));`
- ⑥ `printf("%f ", 5.0/9*(70-32));`

Exercício de fixação 2

Quais das alternativas abaixo convertem corretamente 70 graus Fahrenheit (F) para graus Celsius (C)? Considere que:

$$C = \frac{5}{9} (F - 32) \quad (1)$$

- ① `printf("%f ", 5/9*(70-32));`
- ② `printf("%d ", 5/9*(70-32));`
- ③ `printf("%f ", (70-32.0)*5/9);`
- ④ `printf("%f ", 5.0/9*70-32);`
- ⑤ `printf("%d ", 5.0/9*(70-32));`
- ⑥ `printf("%f ", 5.0/9*(70-32));`

Alternativas 3 e 6!

Exercício de fixação 3 (desafio)

Faça um programa que, para qualquer valor inteiro utilizado como exemplo, imprima na tela seu último dígito. Por exemplo, ao considerar a constante numérica 12235 o programa deve imprimir 5!

```
1  #include<stdio.h>
2
3  int main()
4  {
5      printf("%d ", 12235 % 10);
6
7      return 0;
8  }
```

Exercício de fixação 3 (desafio)

Faça um programa que, para qualquer valor inteiro utilizado como exemplo, imprima na tela seu último dígito. Por exemplo, ao considerar a constante numérica 12235 o programa deve imprimir 5!

```
1  #include<stdio.h>
2
3  int main()
4  {
5      printf("%d ", 12235 % 10);
6
7      return 0;
8  }
```

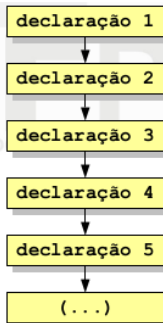
Estruturas sequenciais

Você já percebeu que nos programas que fizemos até agora as instruções são executadas na ordem em que aparecem?

Estruturas sequenciais

As linhas de um programa (o que está entre {} na função main) são executadas de **forma sequencial**. Dizemos que o programa segue um fluxo.

```
int main ()  
{  
    declaração 1;  
    declaração 2; declaração 3;  
    declaração 4;  
    declaração 5;  
    (...)  
}
```



Isto é um *fluxograma*. Fluxogramas eram muito usados antigamente para descrever algoritmos. Hoje em dia, são mais usados para descrever ideias gerais ou partes de algoritmos.

E se eu quisesse fazer um programa que considere valores fornecidos pelo usuário (e não apenas constantes numéricas)?



@peanutsspecials

Próximo material: variáveis

Os valores fornecidos pelo usuário (via teclado, por exemplo) são armazenados na memória principal em **variáveis**!

