

# Entrada/Saída e Operadores Aritméticos

Algoritmos e Programação de Computadores – ABI/LFI/TAI



**INSTITUTO  
FEDERAL**  
Brasília

Prof. Daniel Saad Nogueira  
Nunes

IFB – Instituto Federal de Brasília,  
Campus Taguatinga



# Sumário

---

- 1 Introdução
- 2 Saída
- 3 Entrada
- 4 Expressões Aritméticas



# Sumário

---

## 1 Introdução



# Introdução

---

- A linguagem C possui mecanismos de captura e envio de dados para dispositivos.
- Estamos nos referindo à entrada e saída.
- As funcionalidades são providas pelo cabeçalho `<stdio.h>` da biblioteca padrão do C.



# Introdução

---

## Entrada Padrão

- A entrada padrão da linguagem C é denominada `stdin` e geralmente é dada pelo teclado. Isto é, os dados, por padrão, são capturados pelo teclado.
- O usuário pode digitar valores, através do teclado, que são, por sua vez, armazenados nas variáveis, em memória.



# Introdução

---

## Saída Padrão

- A saída padrão da linguagem C é denominada `stdout` e geralmente é dada pela tela (monitor). Isto é, os dados por padrão são impressos na tela.
- É possível instruir o programa a imprimir o valor de uma variável na tela para visualização do usuário.



# Introdução

---

## Saída de Erros

- A saída de erros padrão da linguagem C é denominada `stderr` e também é dada pela tela. Isto é, as informações relativas à erros também são impressas em tela.



# Introdução

---

- Nesta aula examinaremos as funções `printf` e `scanf` que possibilitam a interação das variáveis com a entrada e saída padrão.
- Através destes mecanismos, será possível implementar programas que façam leitura e escrita de dados.
- Também veremos os operadores aritméticos da linguagem C.
- Com isso, poderemos construir programas simples que capturam dados através do teclado, realizam operações aritméticas e imprimem o resultado do processamento em tela.



# Sumário

---

## 2 Saída



# Saída

---

## printf

- A impressão de mensagens ou de valores das variáveis em tela pode ser feita através do comando `printf`.
- Para utilizar este comando, precisamos instruir o nosso programa a incorporar as definições presentes no arquivo de cabeçalho `stdio.h`.
- Adicionamos a linha `#include <stdio.h>` no início do programa.



# Saída

---

- Para imprimir mensagens, basta utilizar o comando `printf` especificando, entre aspas duplas, a mensagem a ser impressa.
- Após o comando, é necessária a presença do ponto-e-vírgula.
- Exemplo: `printf("Mensagem");`



# Saída

---

## Exemplo: Hello World

```
1  #include <stdio.h>
2
3  int main(void){
4      printf("Hello World!\n");
5      return 0;
6  }
```

- O programa irá imprimir a mensagem “Hello World!” e saltar uma linha.
- O símbolo `\n` representa um caractere de controle que instrui uma quebra de linha.



# Saída

---

- Além de imprimir mensagens, é possível imprimir o conteúdo de uma variável.
- Temos que indicar para o `printf` o formato no qual queremos imprimir aquela variável.
- Utilizamos os **especificadores de formato**.



# Sumário

---

## 2 Saída

- Impressão de inteiros
- Impressão de reais
- Impressão de caracteres
- Impressão de palavras
- Caracteres de Escape
- Imprimindo Múltiplos Valores



## Impressão de Inteiros

---

`%d`

- Para imprimir variáveis `int`, podemos utilizar o especificador de formato `%d`.
- Ao encontrar o `%d`, o comando `printf` irá substituir este especificador pelo valor da variável (ou expressão) correspondente.
- `%d` especifica que queremos imprimir um inteiro em decimal.
- Equivalentemente, podemos usar o `%i`.



# Saída

---

## Exemplo: Impressão de Inteiros

```
1  #include <stdio.h>
2
3  int main(void){
4      int numero = 42;
5      printf("O valor do número é %d\n",numero);
6      return 0;
7  }
```

- O programa irá imprimir a mensagem “O valor do número é 42” na tela e saltar uma linha.
- Observe que o nome da variável está separada da mensagem com especificador por uma vírgula.



# Impressão de Inteiros

---

## Especificadores: Inteiros com Sinal

- Para imprimir variáveis `short int`, usa-se o especificador `%hd`.
- Para imprimir variáveis `int`, usa-se o especificador `%d`.
- Para imprimir variáveis `long int`, usa-se o especificador `%ld`.
- Para imprimir variáveis `long long int`, usa-se o especificador `%lld`.



# Saída

---

## Exemplo: Impressão de Inteiros

```
1  #include <stdio.h>
2
3  int main(void){
4      short int numero_short = -42;
5      int numero_int = -70000;
6      long int numero_long = 10000000000;
7      long long int numero_long_long = -9123456789123;
8      printf("O valor de numero_short é %hd\n",numero_short);
9      printf("O valor de numero_int é %d\n",numero_int);
10     printf("O valor de numero_long é %ld\n",numero_long);
11     printf("O valor de numero_long_long é %lld\n",numero_long_long);
12     return 0;
13 }
```



# Impressão de Inteiros

---

## Especificadores: Inteiros sem Sinal

- Para imprimir variáveis `unsigned short int`, usa-se o especificador `%hu`.
- Para imprimir variáveis `unsigned int`, usa-se o especificador `%u`.
- Para imprimir variáveis `unsigned long int`, usa-se o especificador `%lu`.
- Para imprimir variáveis `unsigned long long int`, usa-se o especificador `%llu`.



# Saída

---

## Exemplo: Impressão de Inteiros

```
1  #include <stdio.h>
2
3  int main(void){
4      unsigned short int numero_short = 42;
5      unsigned int numero_int = 70000;
6      unsigned long int numero_long = 10000000000;
7      unsigned long long int numero_long_long = 9123456789123;
8      printf("O valor de numero_short é %hu\n",numero_short);
9      printf("O valor de numero_int é %u\n",numero_int);
10     printf("O valor de numero_long é %lu\n",numero_long);
11     printf("O valor de numero_long_long é %llu\n",numero_long_long);
12     return 0;
13 }
```



# Saída

---

## Impressão de Inteiros em Diferentes Bases

- Também é possível imprimir números em octal e hexadecimal.
- Usamos os especificadores %o para octal e %x para hexadecimal.
- Independente do número ser positivo ou negativo, imprime-se o padrão de bits que compõe aquele número.



# Saída

---

## Exemplo: Impressão de Inteiros em Octal

```
1  #include <stdio.h>
2
3  int main(void){
4      short int numero_short = 42;
5      int numero_int = -70000;
6      long int numero_long = -10000000000;
7      unsigned long long int numero_long_long = 9123456789123;
8      printf("O valor de numero_short em octal é %ho\n",numero_short);
9      printf("O valor de numero_int em octal é %o\n",numero_int);
10     printf("O valor de numero_long em octal é %lo\n",numero_long);
11     printf("O valor de numero_long_long em octal é %llo\n",numero_long_long);
12     return 0;
13 }
```



# Saída

---

## Exemplo: Impressão de Inteiros em Hexadecimal

```
1  #include <stdio.h>
2
3  int main(void){
4      short int numero_short = 42;
5      int numero_int = -70000;
6      long int numero_long = -10000000000;
7      unsigned long long int numero_long_long = 9123456789123;
8      printf("O valor de numero_short em hexa é %hx\n",numero_short);
9      printf("O valor de numero_int em hexa é %x\n",numero_int);
10     printf("O valor de numero_long em hexa é %lx\n",numero_long);
11     printf("O valor de numero_long_long em hexa é %llx\n",numero_long_long);
12     return 0;
13 }
```



# Sumário

---

## 2 Saída

- Impressão de inteiros
- **Impressão de reais**
- Impressão de caracteres
- Impressão de palavras
- Caracteres de Escape
- Imprimindo Múltiplos Valores



## Impressão de Reais

---

`%f`

- Tanto a impressão de números `float` quanto `double` pode ser feita através do especificador `%f`.
- Motivo: números `float` são convertidos para `double` na hora da impressão.



# Saída

---

## Exemplo: Impressão de Reais

```
1  #include <stdio.h>
2
3  int main(void){
4      float pi = 3.141592;
5      double e = 2.718281828459045;
6      printf("O valor de pi é %f\n",pi);
7      printf("O valor de e é %f\n",e);
8      return 0;
9  }
```



## Impressão de Reais

---

### `%.Nf`

- Por padrão, a precisão da impressão de números reais é 6 casas decimais depois da vírgula.
- É possível especificar a quantidade de dígitos depois da vírgula através do especificador `%.Nf`, em que `N` representa esta quantidade.
- Claro que isto está limitado pela precisão do tipo em questão.



## Impressão de Reais

---

`%e`

- Os especificadores `%e` e `%E` podem ser utilizados para imprimir os números em notação científica.
- Também podem ser acompanhados do especificador de precisão.



# Saída

---

## Exemplo: Impressão de Reais

```
1  #include <stdio.h>
2
3  int main(void){
4      float pi = 3.141592;
5      double e = 2.718281828459045;
6      printf("O valor de pi é %.6e\n",pi);
7      printf("O valor de e é %.10E\n",e);
8      return 0;
9  }
```



# Sumário

---

## 2 Saída

- Impressão de inteiros
- Impressão de reais
- **Impressão de caracteres**
- Impressão de palavras
- Caracteres de Escape
- Imprimindo Múltiplos Valores



## Impressão de Caracteres

---

`%c`

- Para imprimir um caractere, ou simplesmente o caractere associado a um inteiro pela tabela ASCII, usamos o modificador `%c`.



# Saída

---

## Exemplo: Impressão de Caracteres

```
1  #include <stdio.h>
2
3  int main(void){
4      char letra = 'A';
5      char numero = 65;
6      printf("O valor de letra é %c\n",letra);
7      printf("O caractere associado ao número 65 é %c\n",numero);
8      return 0;
9  }
```



# Impressão de Caracteres

---

## %hhd e hhu

- Para imprimir o inteiro de uma variável tipo `char`, utilizamos o especificador `%hhd`.
- No caso de variáveis do tipo `unsigned char`, usa-se o especificador `%hhu`.



# Saída

---

## Exemplo: Impressão de Caracteres

```
1  #include <stdio.h>
2
3  int main(void){
4      char num_1 = -40;
5      unsigned char num_2 = 156;
6      printf("O valor de num_1 é %hhd\n", num_1);
7      printf("O valor de num_2 é %hhu\n", num_2);
8      return 0;
9  }
```



# Sumário

---

## 2 Saída

- Impressão de inteiros
- Impressão de reais
- Impressão de caracteres
- **Impressão de palavras**
- Caracteres de Escape
- Imprimindo Múltiplos Valores



# Impressão de Palavras

---

- Através do especificador `%s` é possível instruir o `printf` a imprimir palavras.



# Saída

---

## Exemplo: Impressão de Palavras

```
1  #include <stdio.h>
2
3  int main(void){
4      printf("Ola turma de APC, meu nome é %s\n","Daniel Saad");
5      return 0;
6  }
```



# Sumário

---

## 2 Saída

- Impressão de inteiros
- Impressão de reais
- Impressão de caracteres
- Impressão de palavras
- **Caracteres de Escape**
- Imprimindo Múltiplos Valores



## Caracteres de Escape

---

- Alguns símbolos especiais como `\`, `%` e `"` são utilizados no `printf`.
- E se quisermos imprimir algum destes símbolos na tela, como fazer?
- Utilizamos uma sequência de escape!



# Saída

---

## Exemplo: Caracteres de Escape

```
1  #include <stdio.h>
2
3  int main(void){
4      printf("Imprimindo o símbolo de contrabarra: \\n");
5      printf("Imprimindo o símbolo de porcentagem: %%n");
6      printf("Imprimindo o símbolo de aspas duplas: \"n");
7      return 0;
8  }
```



# Sumário

---

## 2 Saída

- Impressão de inteiros
- Impressão de reais
- Impressão de caracteres
- Impressão de palavras
- Caracteres de Escape
- **Imprimindo Múltiplos Valores**



## Imprimindo Múltiplos Valores

---

- O comando `printf` aceita múltiplos argumentos, isto é, com um único comando é possível imprimir diversos valores.
- Basta utilizar os especificadores corretos e separar as variáveis por vírgula.



# Saída

---

## Exemplo: Impressão de Múltiplos Valores

```
1 #include <stdio.h>
2
3 int main(void) {
4     int a = 10, b = 20, c = 30;
5     printf("O valor de a é %d, o valor de b é %d e o valor de c é %d\n",a,b,c);
6     return 0;
7 }
```



# Sumário

---

## 3 Entrada



# Entrada

---

## scanf

- Para conseguir ler dados e armazená-los às variáveis, utilizamos o comando `scanf`.
- Funciona de maneira muito parecida ao `printf`.
- Através dos especificadores, os mesmos utilizados no `printf`, indicamos o tipo da variável que está sendo lida.
- Observação: para o tipo `double`, devemos utilizar o especificador `%lf`.



# Entrada

---

## scanf

- Assim com o `printf`, é possível realizar várias leituras com um único comando `scanf`.
- **Importante:** o nome das variáveis deve ser precedido do operador `&`, pois na verdade o `scanf` deve receber o **endereço** da variável, para que ele possa modificar o valor da mesma.



# Entrada

---

## Exemplo: scanf

```
1  #include <stdio.h>
2
3  int main(void){
4      char c;
5      int num;
6      double num_real;
7      printf("Digite um caractere: ");
8      scanf("%c",&c);
9      printf("Digite um número inteiro: ");
10     scanf("%d",&num);
11     printf("Digite um número real: ");
12     scanf("%lf",&num_real);
13     printf("Os valores digitados foram: %c %d %f\n",c,num,num_real);
14     return 0;
15 }
```



# Entrada

---

## Exemplo: scanf

```
1  #include <stdio.h>
2
3  int main(void){
4      char c;
5      int num;
6      double num_real;
7      scanf("%c %d %lf",&c,&num,&num_real);
8      printf("Os valores digitados foram: %c %d %f\n",c,num,num_real);
9      return 0;
10 }
```



# Sumário

---

## 4 Expressões Aritméticas



# Expressões Aritméticas

---

## Expressões Aritméticas

- Uma variável ou constante são consideradas expressões aritméticas.
- Uma expressão aritmética também pode envolver, além de variáveis, operadores aritméticos, tais como:
  - ▶ Adição;
  - ▶ Subtração;
  - ▶ Multiplicação;
  - ▶ Divisão;
  - ▶ Resto;
- Através das expressões aritméticas podemos realizar cálculos.



# Expressões Aritméticas

---

<b>Operador</b>	<b>Significado</b>	<b>Exemplo</b>
+	Adição	$a+b$
-	Subtração	$a-b$
*	Multiplicação	$a*b$
/	Divisão	$a/b$
%	Resto	$a\%b$



# Expressões Aritméticas

---

## Exemplo: Adição

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2, b= 5;
5      int c = a + b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ?



# Expressões Aritméticas

---

## Exemplo: Adição

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2, b= 5;
5      int c = a + b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ? **7**



# Expressões Aritméticas

---

## Exemplo: Subtração

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2, b= 5;
5      int c = a - b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ?



# Expressões Aritméticas

---

## Exemplo: Subtração

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2, b= 5;
5      int c = a - b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ? -3



# Expressões Aritméticas

---

## Exemplo: Multiplicação

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2, b= 5;
5      int c = a * b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ?



# Expressões Aritméticas

---

## Exemplo: Multiplicação

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2, b= 5;
5      int c = a * b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ? **10**



# Expressões Aritméticas

---

## Exemplo: Divisão

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2, b= 5;
5      int c = a / b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ?



# Expressões Aritméticas

---

## Exemplo: Divisão

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2, b= 5;
5      int c = a / b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ? **0**



# Expressões Aritméticas

---

## Divisão Inteira

- Quando os dois parâmetros da divisão são inteiros, realiza-se a divisão inteira.
- Apenas a parte inteira é computada, a parte fracionária é desprezada.
- Se pelo menos um dos parâmetros é ponto flutuante, adota-se a divisão fracionária.
- Note que, se o retorno de uma divisão fracionária for para uma variável inteira, a parte fracionária será desprezada de qualquer forma, então, o retorno de uma divisão fracionária deve ser atribuída a uma variável do tipo `float` ou `double`.



# Expressões Aritméticas

---

## Exemplo: Divisão entre Reais

```
1  #include <stdio.h>
2
3  int main(void){
4      double a = 2, b= 5;
5      double c = a / b;
6      printf("O valor de c é: %.2f\n",c);
7      return 0;
8  }
```

- Qual o valor de  $c$  ?



# Expressões Aritméticas

---

## Exemplo: Divisão entre Reais

```
1  #include <stdio.h>
2
3  int main(void){
4      double a = 2, b= 5;
5      double c = a / b;
6      printf("O valor de c é: %.2f\n",c);
7      return 0;
8  }
```

- Qual o valor de c ? **0.40**



# Expressões Aritméticas

---

## Exemplo: Resto

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 23, b= 5;
5      int c = a % b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ?



# Expressões Aritméticas

---

## Exemplo: Resto

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 23, b= 5;
5      int c = a % b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ? 3



## Expressões Aritméticas: Resto

---

- Apenas aplicável quando os argumentos são inteiros.
- O que acontece se um deles for negativo?



# Expressões Aritméticas

---

## Exemplo: Resto

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = -23, b= 5;
5      int c = a % b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de `c` ?



# Expressões Aritméticas

---

## Exemplo: Resto

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = -23, b= 5;
5      int c = a % b;
6      printf("O valor de c é: %d\n",c);
7      return 0;
8  }
```

- Qual o valor de c ? -3



# Sumário

---

- 4 Expressões Aritméticas
  - Precedência
  - Atribuição simplificada
  - Incremento e Decremento
  - Conversão de tipos (cast)



# Precedência

---

- Assim como na aritmética convencional, alguns operadores da C, possuem precedência sobre outros.
- Considerando o sentido da esquerda para a direita, a seguinte ordem de precedência é adotada:
  - 1 \* e /.
  - 2 %.
  - 3 + e -.



# Precedência

---

- Qual o resultado da expressão: `5*3+2 ?`
- Qual o resultado da expressão: `5*6/3-3 ?`
- Qual o resultado da expressão: `5 + 4 % 3 ?`
- Qual o resultado da expressão: `5 * 4 % 3 + 2 ?`



# Precedência

---

- Qual o resultado da expressão: `5*3+2` ? **17**
- Qual o resultado da expressão: `5*6/3-3` ?
- Qual o resultado da expressão: `5 + 4 % 3` ?
- Qual o resultado da expressão: `5 * 4 % 3 + 2` ?



# Precedência

---

- Qual o resultado da expressão: `5*3+2` ? **17**
- Qual o resultado da expressão: `5*6/3-3` ? **7**
- Qual o resultado da expressão: `5 + 4 % 3` ?
- Qual o resultado da expressão: `5 * 4 % 3 + 2` ?



# Precedência

---

- Qual o resultado da expressão:  $5 * 3 + 2$  ? **17**
- Qual o resultado da expressão:  $5 * 6 / 3 - 3$  ? **7**
- Qual o resultado da expressão:  $5 + 4 \% 3$  ? **6**
- Qual o resultado da expressão:  $5 * 4 \% 3 + 2$  ?



# Precedência

---

- Qual o resultado da expressão:  $5 * 3 + 2$  ? **17**
- Qual o resultado da expressão:  $5 * 6 / 3 - 3$  ? **7**
- Qual o resultado da expressão:  $5 + 4 \% 3$  ? **6**
- Qual o resultado da expressão:  $5 * 4 \% 3 + 2$  ? **4**



## Precedência Parênteses

---

- Assim como na aritmética convencional, podemos usar os parênteses para especificar a ordem na qual as expressões devem ser avaliadas.
- Além deste ponto principal, o uso de parênteses também deixa o código mais **legível**.
- Podemos elaborar expressões mais complicadas pensando mais naturalmente.



# Precedência

---

- Qual o resultado da expressão:  $((5+3)/2)*((13 \% 4)+1)$  ?



# Precedência

---

- Qual o resultado da expressão:  $((5+3)/2)*((13 \% 4)+1)$  ? **8**



# Sumário

---

- 4 Expressões Aritméticas
  - Precedência
  - **Atribuição simplificada**
  - Incremento e Decremento
  - Conversão de tipos (cast)



## Atribuição Simplificada

---

- Frequentemente nos deparamos com expressões que **reescrevem** valores de uma variável quando ela depende do valor antigo.
- Exemplo: `a = a * 2` .
- Se o valor de `a` é 3, o novo valor de `a` passará a ser 6 após a execução desta linha.
- Como este tipo de operação é muito comum, podemos escrever de maneira abreviada: `a *= 2` .



# Atribuição Simplificada

---

●  $a=a+b \equiv a+=b .$

●  $a=a-b \equiv a-=b .$

●  $a=a*b \equiv a*=b .$

●  $a=a/b \equiv a/=b .$

●  $a=a\%b \equiv a\%=b .$



# Sumário

---

- 4 Expressões Aritméticas
  - Precedência
  - Atribuição simplificada
  - **Incremento e Decremento**
  - Conversão de tipos (cast)



# Incremento e Decremento

---

- Duas outras operações muito comuns são as operações de incremento e decremento: isto é, aumentar 1 ou diminuir 1 de um valor inteiro.
- Usamos os operadores ++ e --.
- `a=a+1`  $\equiv$  `a++`  $\equiv$  `++a` .
- `a=a-1`  $\equiv$  `a--`  $\equiv$  `--a` .



## Incremento e Decremento

---

- Podem haver diferenças caso os operadores sejam usados de forma prefixada ou pós-fixada em expressões mais complexas.
- Pré-fixada: primeiro incrementa-se o valor da variável e depois avalia-se a expressão.
- Pós-fixada: primeiro avaliamos a expressão e depois incrementamos o valor da variável.



# Incremento e Decremento

---

## Pré-fixada

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2;
5      int b = ++a + 2;
6      printf("O valor de a é %d e o valor de b é %d\n",a,b);
7      return 0;
8  }
```

- O que será impresso?



# Incremento e Decremento

---

## Pré-fixada

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2;
5      int b = ++a + 2;
6      printf("O valor de a é %d e o valor de b é %d\n",a,b);
7      return 0;
8  }
```

- O que será impresso? **3 e 5**



# Incremento e Decremento

---

## Pós-fixada

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2;
5      int b = a++ + 2;
6      printf("O valor de a é %d e o valor de b é %d\n",a,b);
7      return 0;
8  }
```

- O que será impresso? .



# Incremento e Decremento

---

## Pós-fixada

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 2;
5      int b = a++ + 2;
6      printf("O valor de a é %d e o valor de b é %d\n",a,b);
7      return 0;
8  }
```

- O que será impresso? **3 e 4.**



# Sumário

---

- 4 Expressões Aritméticas
  - Precedência
  - Atribuição simplificada
  - Incremento e Decremento
  - Conversão de tipos (cast)



# Conversão de Tipos

---

- Na linguagem C a conversão de tipos pode ser implícita ou explícita.
- Na conversão implícita, caso atribua-se um valor para uma variável de tipo diferente, ela pode ser convertida sem prejuízo de informação ou com alguma perda.
  - ▶ Por exemplo: caso queiramos atribuir um número fracionário a uma variável `int`, haverá uma perda, pois haverá truncamento do número inteiro.
  - ▶ O contrário pode ocorrer sem problemas, já que o tipo `double` suporta todos os inteiros.



# Conversão de Tipos

## Exemplo: Conversão Implícita

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 5;
5      double b = a;
6
7      printf("%d %lf\n",a,b);
8
9      b= 9.74;
10     a = b;
11
12     printf("%d %lf\n",a,b);
13     return 0;
14 }
```



## Conversão de Tipos

---

- É possível também indicar como uma determinada expressão deve ser avaliada explicitamente.
- Útil para realizar divisões fracionárias em vez de divisões inteiras.



# Conversão de Tipos

---

## Exemplo: Conversão Explícita

```
1  #include <stdio.h>
2
3  int main(void){
4      int a = 5, b = 2;
5      printf("O valor da divisão 5/2 = %.2f\n", (double) a/b);
6      return 0;
7  }
```