
Linguagem C: Variáveis e Operadores

Sumário

- Variáveis:
 - Tipos de variáveis;
 - Nome da variável;
 - Palavras reservadas;
 - A função printf();
 - A função scanf();
 - Variáveis locais;
 - Variáveis globais;
- Constantes:
 - Constantes Hexadecimais e Octais;
 - Constantes de barra invertida;

Sumário

- Operadores:
 - Operador de atribuição;
 - Conversão de tipos em atribuição;
 - Operadores Aritméticos;
 - Operadores Relacionais;
 - Operadores Lógicos;
 - Operadores bit a bit.

Variáveis

- São espaços de **memória** reservados que guardam valores durante a execução de um programa;
- **Todas** as variáveis em C devem ser **declaradas**, antes de serem **usadas**;
- Uma declaração de variável em C consiste no nome de um **tipo**, seguido do **nome** da variável, seguido de **ponto-e-vírgula**.

Ex: tipo_da_variavel lista_de_variaveis;

int num;

Tipo da Variável

Nome da Variável

Tipos de Variáveis

- O tipo de uma variável informa a quantidade de **memória**, em bytes, que a variável ocupará e a forma como um valor deverá ser armazenado;
- Há **cinco** tipos básicos de dados em C:

TIPO	BIT	BYTES	ESCALAS
char	8	1	-128 a 127
int	16	2	-32768 a 32767
float	32	4	3.4E-38 a 3.4E+38
double	64	8	1.7E-308 a 1.7E+308
void	0	0	Nenhum valor

Tipos de Variáveis

- Exceto o **void**, os tipos de dados básicos podem ter vários **modificadores** precedendo-os;
- Um modificador é usado para **alterar** o significado de um tipo básico para adaptá-lo mais precisamente às **necessidades** de diversas situações, veja:

signed;

unsigned;

long;

short.

Tipo	Bits	Início	Fim
int	16	-32.768	32.767
unsigned int	16	0	65.535
signed int	16	-32.768	32.767
short int	16	-32.768	32.767
long int	32	-2.147.483.648	2.147.483.647

Nome da Variável

- O nome de uma variável pode ser de uma letra até palavras com no máximo 32 caracteres;
- Obrigatoriamente **deve** começar com uma letra ou underline (“_”). O restante pode ser letras de A a Z, maiúsculas, minúsculas, números e o underscore;
Ex: a; num; essa_e_uma_variavel; tambem_essa;
- **Cuidados:**
 - ❑ O nome de uma variável não pode ser igual a uma palavra reservada;
 - ❑ O nome de uma variável não pode ser igual a de uma função declarada pelo programador ou pelas bibliotecas do C.

Palavras reservadas

- Eis algumas palavras reservadas da linguagem C:

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
asm	pascal	far	huge
interrupt	near	_cs	_ds

Nome da Variável

- Em C, letras maiúsculas e minúsculas são tratadas diferentemente.

```
int variavel;
```

```
int Variavel;
```

```
int VaRiAVeL;
```

```
int VARIAVEL;
```

ou

```
int variavel, Variavel, VaRiAVeL, VARIAVEL;
```

Exemplos de Variáveis

```
#include <stdio.h>
```

```
/* Exemplo da variável Char */
```

```
int main()
```

```
{
```

```
    char Ch;
```

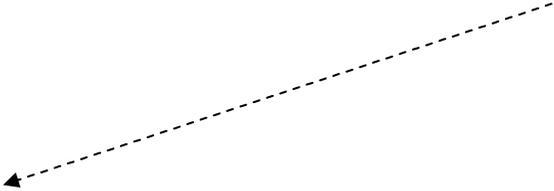
```
    Ch='D';
```

```
    printf("%c", Ch);
```

```
    return 0;
```

```
}
```

%c indica que printf() deve colocar um caracter na tela.



Exemplos de Variáveis

```
#include <stdio.h>
```

```
/* Exemplo da variável Inteiro */
```

```
int main()
```

```
{
```

```
    int num;
```

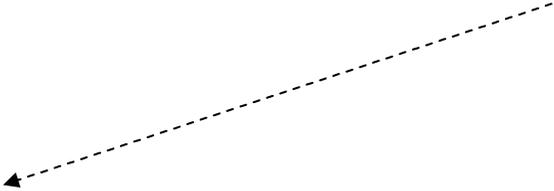
```
    num=10;
```

```
    printf("%d", num);
```

```
    return 0;
```

```
}
```

%d indica que printf() deve colocar um inteiro na tela.



A função printf()

- A função **printf()** tem a seguinte forma geral:
`printf(string_de_controle, lista_de_argumentos)`
- Teremos, na string de controle, uma descrição de tudo que a função vai colocar na tela.
- Isto é feito usando-se os códigos de controle, veja alguns exemplos:

Código	Significado
%d	Inteiro
%f	Float
%c	Caractere
%s	String
%%	Coloca um % na tela

A função scanf()

- O formato geral da função scanf() é:
scanf(string_de_controle, lista_de_argumentos);

```
#include <stdio.h>
/* Exemplo da função scanf() */
int main()
{
    int num;
    printf("Digite um número: ");
    scanf("%d", &num);
    printf("%d", num);
    return 0;
}
```

Exemplos de Variáveis

```
#include <stdio.h>
```

```
/* Exemplo da variável String */
```

```
int main()
```

```
{
```

```
    char nome[20];
```

Função para leitura
de String (Char)

```
    printf("Digite seu nome: ");
```

```
    gets(nome);
```

```
    printf("\n\nSeu nome é: %s", nome);
```

```
    return 0;
```

```
}
```

Variáveis Locais

- São variáveis declaradas **dentro** de uma função;
- **Só** podem ser **referenciadas** por comandos que estão dentro do bloco no qual as variáveis foram declaradas;

```
#include <stdio.h>
```

```
int main()  
{  
    int x;  
    int y;  
    x = 10;  
    y = 20;  
}
```

Início do bloco

Fim do bloco

Variáveis Locais

- As variáveis também podem ser declaradas dentro de qualquer outro bloco de código, veja:

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x;
```

```
    scanf("%d", &x);
```

```
    if (x == 1)
```

```
    {
```

```
        char s[30];
```

```
        printf("Entre com o nome");
```

```
        gets(s);
```

```
        /* Faz alguma coisa*/
```

```
    }
```

```
}
```

Criação da variável

Morte da variável

Variáveis Globais

- São variáveis **reconhecidas** pelo programa inteiro e podem ser usadas por **qualquer** bloco de código;
- As variáveis globais são criadas declarando-as **fora** de qualquer função. Veja:

```
#include <stdio.h>
```

```
int num;
```

```
void func1();
```

```
void func2();
```

Declaração de uma
variável global

Utilização da variável

```
int main()
```

```
{
```

```
    num = 100;
```

```
    func1();
```

```
}
```

```
void func1()
```

```
{
```

```
    num = num + 1;
```

```
    func2();
```

```
}
```

```
void func2()
```

```
{
```

```
    num = num + 1;
```

```
    printf("%d", num);
```

```
}
```

Variáveis Globais

- As variáveis globais encontram-se armazenadas em uma **região fixa da memória**, separada para esse propósito pelo compilador C;
- Variáveis globais são úteis quando o mesmo dado é usado em **muitas funções** em seu programa;
- **Alerta:** Variáveis globais ocupam memória durante todo o tempo em que seu programa estiver executando, portanto, **evite** usar variáveis globais **desnecessárias**.

Constantes

- Variáveis com o **modificador const** não podem ser modificadas por seu programa;

```
#include <stdio.h>
```

```
int main()  
{  
    const int num = 100;  
}
```

← Cria uma variável inteira chamada a, com um valor inicial 10, que seu programa não pode modificar.

Constantes Hexadecimais e Octais

- Muitas vezes precisamos inserir constantes hexadecimais (base dezesseis) ou octais (base oito) no nosso programa.
- Em C as constantes hexadecimais começam com **0x** e as constantes octais começam com **0**.

```
int varHex = 0x80; ← 128 em decimal  
int varOct = 012; ← 10 em decimal
```

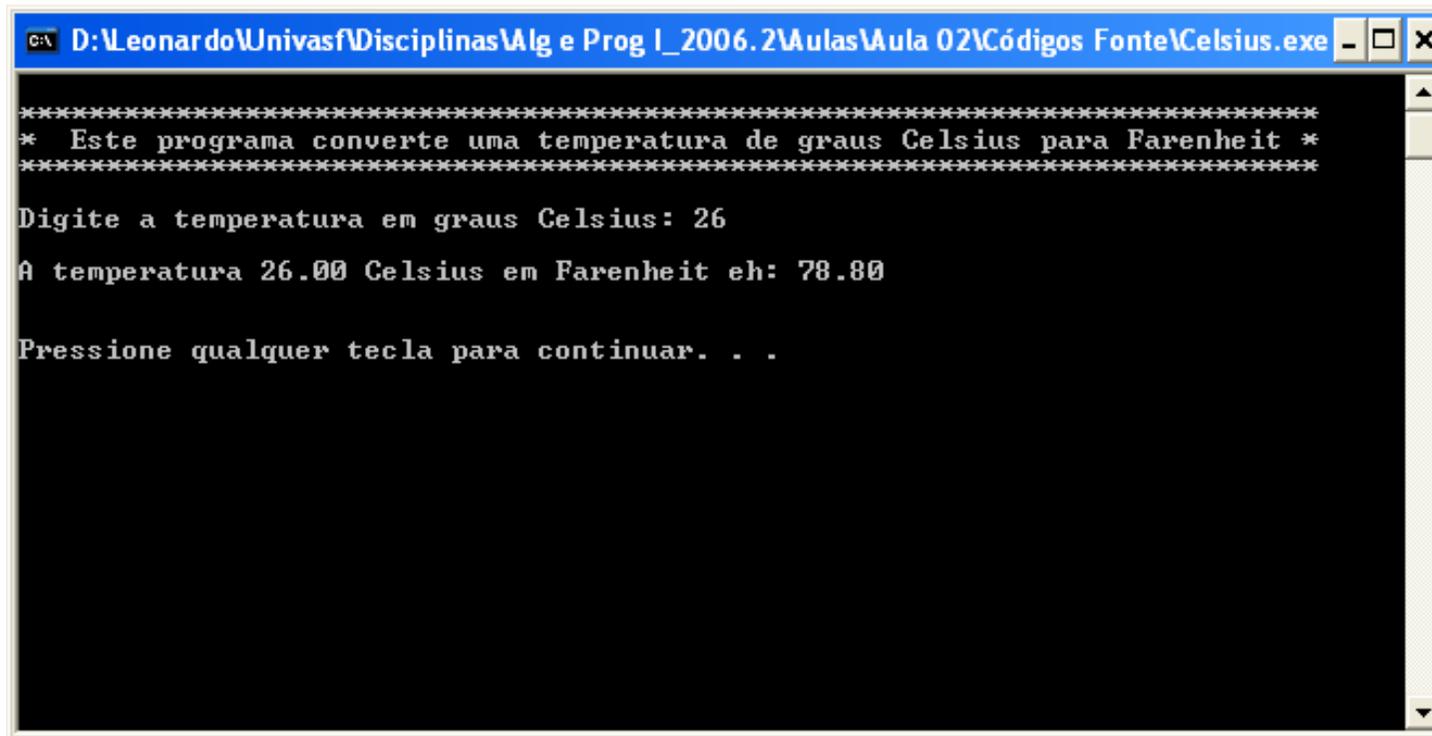
Constantes de Barra Invertida

Código	Significado
<code>\b</code>	Retrocesso
<code>\f</code>	Alimentação de formulário
<code>\n</code>	Nova linha
<code>\r</code>	Retorno de carro
<code>\t</code>	Tabulação horizontal
<code>\"</code>	Aspas duplas
<code>\'</code>	Aspas simples
<code>\0</code>	Nulo
<code>\\</code>	Barra invertida
<code>\v</code>	Tabulação vertical
<code>\a</code>	Alerta
<code>\xN</code>	Constante Hexa (onde N é a constante Hexa)

Exercício da Lista

- Faça um programa em C para ler uma temperatura em graus Celsius e transformá-la em Farenheit. Utilize a seguinte fórmula:

$$F = ((9 * C) / 5) + 32$$



```
C:\ D:\LeonardoUnivasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 02\Códigos Fonte\Celsius.exe - _ □ ×
*****
* Este programa converte uma temperatura de graus Celsius para Farenheit *
*****
Digite a temperatura em graus Celsius: 26
A temperatura 26.00 Celsius em Farenheit eh: 78.80
Pressione qualquer tecla para continuar. . .
```

Resposta sugerida para o exercício

```
#include <stdio.h>
/* Este programa converte uma temperatura dada em graus Celsius para Farenheit */

int main()
{
    float Cel, Far;
    printf("\n*****");
    printf("\n*   Este programa converte uma temperatura de graus Celsius para Farenheit   *");
    printf("\n*****");
    printf("\n\nDigite a temperatura em graus Celsius: ");
    scanf("%f", &Cel);
    Far = ((9*Cel)/5)+32;
    printf("\nA temperatura %.2f Celsius em Farenheit eh: %.2f\n\n\n", Cel, Far);
    system("pause");
    return 0;
}
```

Operadores

Operadores

- A linguagem C é **muito rica** em **operadores** internos. Ela define quatro classes de operadores:

□ Aritmético; ← - , + , * , / , % , -- , ++

□ Relacionais; ← > , < , >= , <= , == , !=

□ Lógicos; ← && , || , !

□ Bit a bit. ← & , | , ^ , ~ , >> , <<

- Além disso, C tem alguns **operadores especiais** para tarefas particulares.

Operador de Atribuição

- A forma geral do operador de atribuição é:
`nome_da_variavel = expressao`
- A expressão pode ser tão simples como uma única constante ou tão complexa quanto você necessite;
- O destino, ou parte esquerda da atribuição, deve ser uma variável ou um ponteiro, não uma função ou uma constante.

Conversão de tipos em atribuição

- Refere-se à situação em que variáveis de um tipo são convertidas em outro tipo;
- A regra é **muito simples**: o valor do lado direito (**expressão**) de uma atribuição é **convertida** no tipo do lado esquerdo (**variável destino**), veja:
`#include <stdio.h>`

```
int i;  
char ch;  
float f;
```

```
int main()
```

```
{  
    i = f;   
    f = i;   
    ch = i;   
    f = ch;   
}
```

← i recebe a parte inteira (16 bits) de f;

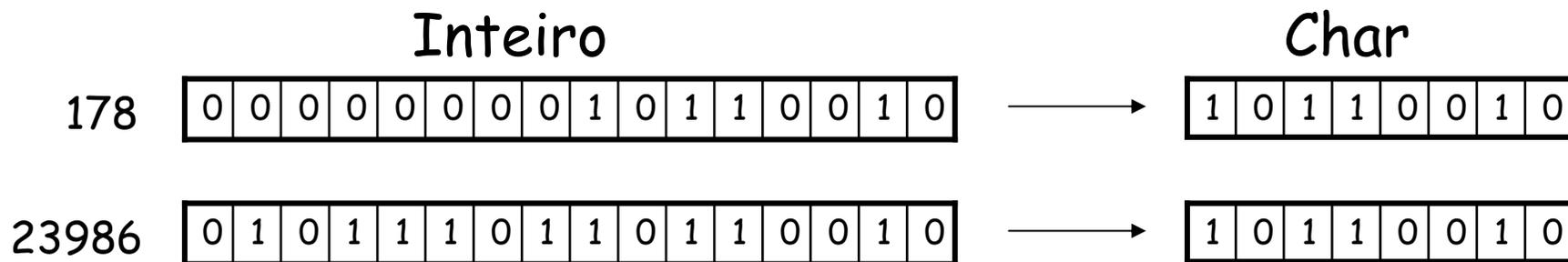
← f converte um inteiro de 16 bits em um ponto flutuante;

← coloca os bits menos significativos de i em ch;

← f converte o inteiro de 8 bits guardado em ch em um ponto flutuante;

Conversão de tipos em atribuição

- Quando se converte de:
 - inteiro para caracter;
 - inteiros longos para inteiros;
 - Inteiros para inteiros curtos;
- A regra básica é que a quantidade apropriada de bits **significativos** seja ignorada, veja um exemplo:



Conversão de tipos em atribuição

- A tabela abaixo reúne essas conversões de tipos.
- Lembre-se que a conversão de **int** em um **float** ou **float** em **double**, etc não aumenta a precisão ou exatidão, apenas mudam a forma de representar.

Tipo do destino	Tipo da expressão	Possível informação perdida
signed char	char	Se valor > 127, o destino é negativo
char	short int	Os 8 bits mais significativos
char	int	Os 8 bits mais significativos
char	long int	Os 24 bits mais significativos
int	long int	Os 16 bits mais significativos
int	float	A parte fracionária e/ou mais
float	double	Precisão, o resultado é arredondado
double	long double	Precisão, o resultado é arredondado

Atribuições múltiplas

- Em C é permitido que você atribua o mesmo valor a muitas variáveis usando atribuições múltiplas em um único comando, veja:

Exemplo 1:

```
int x, y, z;  
x = y = z = 0;
```

Exemplo 2:

```
int i;  
char ch;  
float f;  
f = ch = i = 72;
```

Operadores Aritméticos

- A tabela abaixo lista os operadores aritméticos de C:

Operador	Ação
-	Subtração, também menos unário
+	Adição
*	Multiplicação
/	Divisão
%	Módulo da divisão (resto)
--	Decremento
++	Incremento

Operadores Aritméticos

- C possui operadores unário, binário e ternário:
 - Os unários agem sobre uma variável apenas modificando ou não seu valor. Ex: o operador - (-1, -3000, etc);
 - Os binários usam duas variáveis e retornam um terceiro valor, sem alterar as variáveis originais. Ex: os operadores + e -
 - O ternário é usado para substituir certas sentenças de forma if-then-else. Ex: o operador ?

Operadores Aritméticos

- O operador / (divisão) quando aplicado a variáveis inteiras ou caracter, nos fornece o resultado da divisão inteira, ou seja, o resto é truncado;

```
int x = 5, y = 2;  
printf("%d", x/y);
```

Mostra na tela o número 2

- O operador % (módulo) quando aplicado a variáveis inteiras ou caracter, nos fornece o resto de uma divisão inteira;

```
int x = 5, y = 2;  
printf("%d", x%y);
```

Imprime na tela o resto da divisão, portanto, o número 1

Operadores Aritméticos

- O operador / (divisão) quando aplicado a variáveis em ponto flutuante nos fornece o resultado da divisão "real".

```
float x = 5, y = 2;
```

```
printf("%f", x/y);
```

Mostra na tela o número 2.500000



- **Alerta:** O operador % (módulo) não pode ser usado nos tipos em ponto flutuante (float e double).

Operadores Aritméticos

- Outros operadores aritméticos definidos em C são os operadores de:
 - Incremento: ++
 - Decremento: --
- O operador ++ soma 1 ao seu operando, e - subtrai 1.

Incremento

```
x = x + 1;
```

```
x++;
```

Decremento

```
x = x - 1;
```

```
x--;
```

- Ambos os operadores de incremento ou decremento podem ser utilizados como **prefixo** ou **sufixo** do operando.

```
x = x + 1; /* Equivale a x++ e a ++x */
```

Operadores Aritméticos

- Existe uma diferença quando os operadores de incremento e decremento são usados em uma expressão, veja:

Ex1:

```
x = 10;
```

```
y = ++x;
```

Executa o incremento antes de usar o valor do operando para atualizar y. Resultado: X = 11 e Y = 11

Ex2:

```
x = 10;
```

```
y = x++;
```

Usa o valor do operando para atualizar y antes de incrementar x. Resultado: X = 11 e Y = 10

Operadores Aritméticos

- A **precedência** dos operadores aritméticos é a seguinte:

Mais alta	++, --
	- (menos unário)
	*, /, %
Mais baixa	+, -

- Operadores do mesmo nível de precedência são avaliados pelo compilador da **esquerda para a direita**;
- Obviamente, **parênteses** podem ser usados para alterar a ordem de avaliação.

Operadores Aritméticos

- **Exercício:** Qual o resultado das variáveis **x**, **y** e **z** depois da seguinte seqüência de operações:

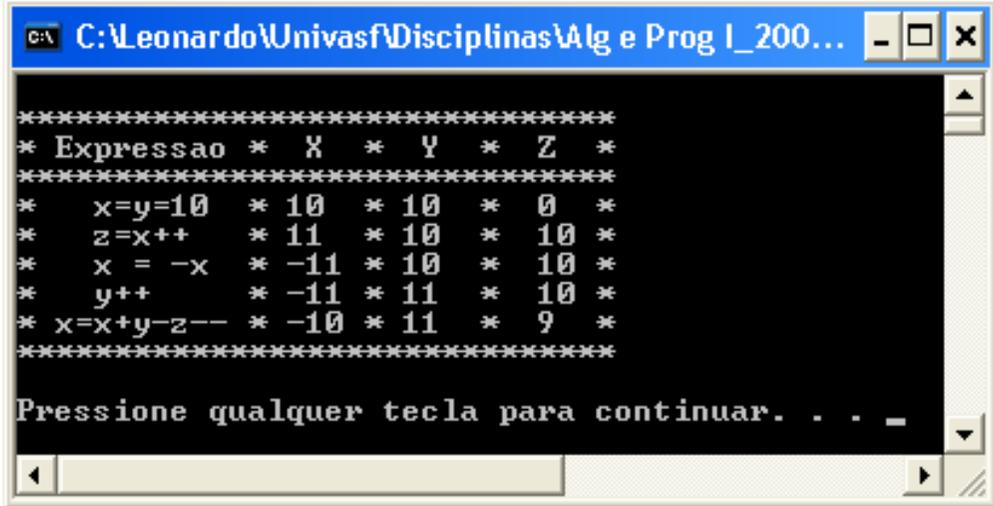
```
int x, y, z;  
x = y = 10;  
z = (x++);  
x = -x;  
y++;  
x = x + y - (z--);
```

x	y	z
10	10	-
11	10	10
-11	10	10
-11	11	10
-10	11	9

Exercício

- Qual o resultado das variáveis x , y e z depois da seguinte seqüência de operações:

```
int x, y, z;  
x = y = 10;  
z = x++;  
x = -x;  
y++;  
x = x + y - z--;
```



```
C:\Leonardo\Univasf\Disciplinas\Alg e Prog I_200...  
*****  
* Expressao * X * Y * Z *  
*****  
* x=y=10 * 10 * 10 * 0 *  
* z=x++ * 11 * 10 * 10 *  
* x = -x * -11 * 10 * 10 *  
* y++ * -11 * 11 * 10 *  
* x=x+y-z-- * -10 * 11 * 9 *  
*****  
Pressione qualquer tecla para continuar. . . -
```

Programa sugerido para o exercício

```
#include <stdio.h>
/* Este programa demonstra o funcionamento dos operadores aritméticos em C */

int main()
{
    int x, y, z=0;
    printf("\n*****");
    printf("\n* Expressao * X * Y * Z *");
    printf("\n*****");
    x=y=10;
    printf("\n*   x=y=10   * %d * %d * %d *", x, y, z);
    z = x++;
    printf("\n*   z=x++   * %d * %d * %d *", x, y, z);
    x = -x;
    printf("\n*   x = -x   * %d * %d * %d *", x, y, z);
    y++;
    printf("\n*   y++     * %d * %d * %d *", x, y, z);
    x = x+y-z--;
    printf("\n* x=x+y-z-- * %d * %d * %d *", x, y, z);
    printf("\n*****\n\n");
    system("pause");
    return 0;
}
```

Operadores Relacionais

- No termo *operadores relacionais*, relacional refere-se às **relações** que os valores podem ter uns com os outros;
- Os operadores relacionais do C realizam **comparações** entre variáveis. São eles:

Operador	Ação
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a
==	Igual a
!=	Diferente de

Operadores Relacionais

- Os operadores relacionais retornam **verdadeiro (1)** ou **falso (0)**.
- Em C, verdadeiro é qualquer valor diferente de zero.
- Os operadores relacionais tem precedência menor do que os operadores aritméticos, veja:

```
int a = 10, b = 1, c = 12;  
a > b + c;
```

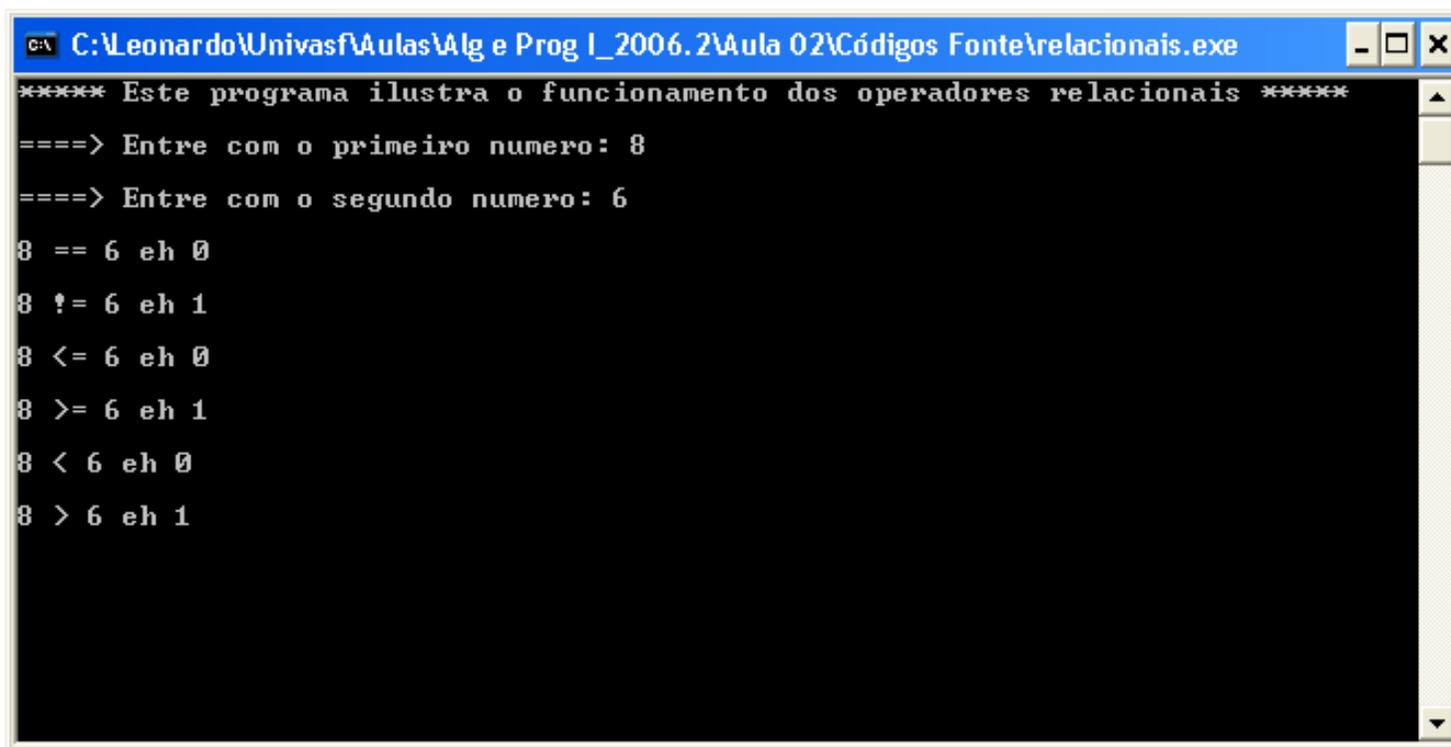
- O resultado da expressão acima é: **Falso**

Operadores Relacionais

```
#include <stdio.h>

int main()
{
    int i, j;
    printf("***** Este programa ilustra o funcionamento dos operadores relacionais *****");
    printf("\n\n====> Entre com o primeiro numero: ");
    scanf("%d", &i);
    printf("\n====> Entre com o segundo numero: ");
    scanf("%d", &j);
    printf("\n%d == %d eh %d\n", i, j, i==j);
    printf("\n%d != %d eh %d\n", i, j, i!=j);
    printf("\n%d <= %d eh %d\n", i, j, i<=j);
    printf("\n%d >= %d eh %d\n", i, j, i>=j);
    printf("\n%d < %d eh %d\n", i, j, i<j);
    printf("\n%d > %d eh %d\n", i, j, i>j);
    scanf("%d", &i);
    return (0);
}
```

Operadores Relacionais



```
C:\Leonardo\Univasf\Aulas\Alg e Prog I_2006.2\Aula 02\Códigos Fonte\relacionais.exe
***** Este programa ilustra o funcionamento dos operadores relacionais *****
====> Entre com o primeiro numero: 8
====> Entre com o segundo numero: 6
8 == 6 eh 0
8 != 6 eh 1
8 <= 6 eh 0
8 >= 6 eh 1
8 < 6 eh 0
8 > 6 eh 1
```

Operadores Lógicos

- No termo *operador lógico*, lógico refere-se às maneiras como as relações podem ser conectadas;
- Os operadores lógicos definidos por C são os seguintes:

Operador	Ação
&&	And (E)
	Or (Ou)
!	Not (Não)

Operadores Lógicos

- A tabela da verdade dos operadores lógicos são mostrados a seguir, usando 1s e 0s:

p	q	p&&q	p q	!p
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Operadores Bit a bit

- Operação bit a bit refere-se a testar, atribuir ou deslocar os bits efetivos em um byte ou uma palavra, que correspondem aos tipos de dados **char** e **int** e variantes do padrão **C**.
- Operações bit não podem ser usadas em **float**, **double**, **long double**, **void** ou outros tipos mais complexos.

Operadores Bit a bit

- A tabela abaixo lista os operadores que se aplicam às operações bit a bit.

Operador	Ação
&	And
	Or
^	Or exclusivo (Xor)
~	Complemento de um
>>	Deslocamento à esquerda
<<	Deslocamento à direita

Operadores Bit a bit

- Os operadores bit a bit AND, OR e NOT (complemento de 1) são governadas pela mesma tabela da verdade anterior*, exceto por trabalharem bit a bit;
- O Or exclusivo (\wedge) tem a tabela da verdade mostrada a seguir, veja:

p	q	p&q	p q	!p	p \wedge q
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	0
1	1	1	1	0	1

*Slide 45

Operadores Bit a bit

- Operadores bit a bit encontram aplicações mais freqüentes em "drivers" de dispositivos - como em programas de **modem**, rotinas de **arquivos em disco** e **impressoras**;
- Operador de complemento de um, \sim , inverte o estado de cada bit da variável especificada.

0	0	1	0	1	1	0	0	← Byte Original
1	1	0	1	0	0	1	1	← Após complemento de 1

- todos os 1s são colocados em 0 e todos os 0s são colocados em 1

Operadores Bit a bit

- Os operadores de deslocamento, \gg e \ll **movem** todos os **bits** de uma variável para a direita ou para a esquerda, respectivamente.
- A forma geral de deslocamento à **direita** é:
variável \gg número de posições de bits
- A forma geral de deslocamento à **esquerda** é
variável \ll número de posições de bits
- Conforme os bits são deslocados para uma extremidade, zeros são colocados na outra;

Operadores Bit a bit

- Exemplo de deslocamento à direita e à esquerda:

Unsigned char x;	X a cada execução	Valor de x
$x = 7$	00000111	7
$x = x \ll 1$	00001110	14
$x = x \ll 3$	01110000	112
$x = x \ll 2$	11000000	192
$x = x \gg 1$	01100000	96
$x = x \gg 2$	00011000	24

- Note que cada deslocamento à esquerda multiplica por 2.

Bibliografia

- SCHILDT H. *"C Completo e Total"*, Makron Books. SP, 1997.
- MIZRAHI, V. V. *"Treinamento em Linguagem C++ Módulo 1"*, Makron Books, SP, 1995.
- FORBELLONE, A. L. V. *"Lógica de Programação: A construção de algoritmos e estruturas de dados"*, Prentice Hall, SP, 2005.