



Linguagem de Programação I

Neilor Tonin

2015

Char x int

- No C e C++, a associação, manipulação e conversão `int<->char<->int` é transparente:

```
for (int i='a'; i<='j'; i++) {  
    cout << (char)i << ' ';  
}
```

a b c d e f g h i j

- Pode ser feito utilizando char, da mesma forma:

```
for (char i='f'; i<='p'; i++) {  
    cout << i << ' ';  
}
```

f g h i j k l m n o p

- Existem muitas outras operações interessantes...

Char x int

- O que acontece ao diminuir 2 caracteres?

- Exemplo: `'f' - 'a'`

- Como somar dois números que estão no formato de caracter?

```
String digitos= "853"
```

```
char a = digitos[0] //8
```

```
char b = digitos[1] //5
```

- Como mostrar toda a tabela ASC dos valores acima do espaço (32)?

- Para ficar mais interessante mostre 10 elementos por linha

- Apresente de forma organizada, dentro de um mesmo espaçamento:

```
32=   33=!  34="  35=#  36=$  37=%  38=&  39='  40=(  41=)  42=*  
43=+  44=,  45=-  46=.  47=/  48=0  49=1  50=2  51=3  52=4  53=5
```

...

Flags

- Alguns flags interessantes (Pesquise na internet C++ I/O flags):
 - setw
 - right, left
 - dec, hex, oct
 - uppercase

```
#include <iostream>
#include <iomanip> /// para usar setw
/** Pesquise na internet C++ I/O flags */

using namespace std;

int main() {
    cout << setw(6) << right << 123 << endl;
    cout << dec << setw(6) << left << 123 << endl << endl;
    int valor;
    cout << "digite um valor: ";
    cin >> valor;
    cout << dec << valor << endl;
    cout << hex << valor << endl;
    cout << uppercase << hex << valor << endl;
    cout << oct << valor << endl;
    return 0;
}
```

```
      123
123

digite um valor: 13
13
d
D
15
```

Conversões

- De decimal para hexa e octal através de flags:

- hex , oct

- De hexa para decimal:

```
int number = 0x3FF;
```

```
cout << "Decimal: " << number << endl;
```

```
char str[20];
```

```
cout << "Valor hexadecimal (0xc, C ou c por exemplo): ";
```

```
cin >> str;
```

```
cout << dec << strtol(str, NULL, 16) << endl;
```

- De binário para decimal:


```
cout << "Digite valor em binario (0101 por exemplo): ";
```

```
cin >> str;
```

```
cout << strtol(str, NULL, 2) << endl;
```

```
cout << strtol("1011", NULL, 2) << endl;
```

Conversão simples de Base

Por Shahriar Manzoor  Bangladesh

Timelimit: 2

1199

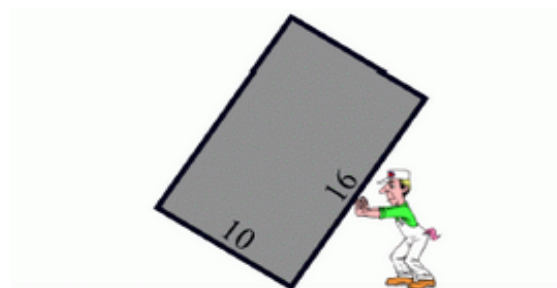


Descrição
Tela Cheia
Submeter
Ranking
Forum
Toolkit

RESOLVIDO



Rank: 8º
Run: 131883
Time: 0.024



Neste problema você é solicitado a escrever um simples programa de conversão de base. A entrada será um valor hexadecimal ou decimal. Você deverá converter cada valor da entrada. Se o valor for hexadecimal, você deve convertê-lo para decimal e vice-versa. O valor hexadecimal inicia sempre com "0x" ou também, é aquele valor cuja segunda casa contém a letra 'x'.

Entrada

A entrada contém vários casos de teste. Cada linha de entrada, com exceção da última, contém um número não-negativo, decimal ou hexa. O valor decimal será menor ou igual a 2^{31} . A última linha contém um número negativo que não deve ser processado, indicando o encerramento do programa.

Saída

Para cada linha de entrada (exceto a última) deve ser produzido uma linha de saída. Todo número hexadecimal deve ser precedido na saída por '0x' (zero xis).

Exemplo de Entrada	Exemplo de Saída
4	0x4
7	0x7
44	0x2C
0x80685	525957
-1	

Entrada e saída por Neilor.

Tipos de Dados e Operadores

- Recapitulando:

Name	Description	Size*	Range*
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	Wide character.	2 or 4 bytes	1 wide character

```
int main() {  
    unsigned long long a=12345678901234567890;  
    bool f = true;  
    double g = 1234567890.12345;  
    double h = 12345678901234567890;  
  
    cout << a << endl << f << endl;  
    cout << g << endl << h << endl;  
    cout << setprecision(15) << endl;  
    cout << g << endl;  
    cout << setprecision(20) << h << endl;  
    return 0;  
}
```

```
123456789012345678901  
1.23457e+009  
1.23457e+019
```

```
1234567890.12345  
12345678901234567168
```

Usar long double

Tipos de Dados e Operadores

- Operadores Aritméticos:

<i>Operador</i>	<i>Ação</i>
+	Soma (inteira e ponto flutuante)
-	Subtração ou Troca de sinal (inteira e ponto flutuante)
*	Multiplicação (inteira e ponto flutuante)
/	Divisão (inteira e ponto flutuante)
%	Resto de divisão (de inteiros)
++	Incremento (inteiro e ponto flutuante)
--	Decremento (inteiro e ponto flutuante)

- Muito simples... Será?

```
int a, b=3, c=4;  
a = b++;  
c = --a - ++b - --c;
```

- Quanto valem:

a: ____

b: ____

c: ____

Exercícios

Tipos de Dados e Operadores

- Simplificações:

Expressão Original	Equivalente	Expressão Original	Equivalente
$x = x + k;$	$x += k;$	$x = x >> k;$	$x >>= k;$
$x = x - k;$	$x -= k;$	$x = x << k;$	$x <<= k;$
$x = x * k;$	$x *= k;$	$x = x ^ k;$	$x ^ k;$
$x = x / k;$	$x /= k;$	etc...	

- Operadores relacionais e lógicos

Operador	Ação	Operador	Ação
>	maior que	<=	menor ou igual
<	menor que	==	igual
>=	maior ou igual	!=	diferente

Operador	Ação
&&	e
	ou
!	não

Exercícios

Tipos de Dados e Operadores

- Operador condicional (?):

Exemplo	Pode ser substituído por
<pre>if (x>3) k = k + 1; else k = s - 5;</pre>	<pre>k=(x>3)? k+1: s-5;</pre>

- Operadores bit-a-bit

Operador	OPERAÇÃO	Operador	OPERAÇÃO
&	e	^	ou exclusivo
	ou	>>	shift para direita (divisão por 2)
~	não	<<	shift para esquerda (multiplicação por 2)

char a=1, b=3, c;

c = a & b	<pre>00000001 & 00000011 00000001</pre>	c = b >> 1	<pre>00000011 00000001</pre>	(C) c = a ^ b	<pre>00000001 ^ 00000011 00000010</pre>
c = ~a	<pre>~ 00000001 11111110</pre>	c = a b	<pre>00000001 00000011 00000011</pre>	c = b << 1	<pre>00000011 00000110</pre>

Exercícios 1 a 11 (folha)