Linguagem de Programação C
Aula 4

Professor: Rodrigo Rocha

Conteúdo - Operadores e Expressões

- Operadores aritméticos
- Operadores aritméticos de atribuição
- Operadores binários x unários
- Operadores relacionais
- Operadores lógicos
- Operadores lógicos bit a bit
- Expressões
- Conversão de tipos
- Encadeando expressões
- Precedência

Operadores aritméticos

Operador	Ação	Tipo de dado
+	Soma	Inteiro Ponto flutuante
-	Subtração Troca de sinal	Inteiro Ponto flutuante
*	Multiplicação	Inteiro Ponto flutuante
/	Divisão	Inteiro Ponto flutuante
%	Resto da divisão	Inteiro

Operadores aritméticos (cont.)

- Exemplo:

```
int i=1, j=1, k;
float x;

i=i+1;  // 2
j=i+j;  // 3
k=j-i;  // 1
k=i*j;  // 6
k=j/1;  // 1 (divisão inteira)
k=j%1;  // 0 (resto da divisão)

x=(float)j/i;  // 1,5 (divisão de flutuante)
```

Operadores aritméticos de atribuição

- Aplicado a uma variável, alteram o valor da mesma
- Podem ser pré ou pós fixado

Operador	Ação	Tipo de dado
++	Incremento	Inteiro Ponto flutuante
--	Decremento	Inteiro Ponto flutuante

Operadores aritméticos de atribuição (Cont.)

- Exemplo:

```
int i=1, j=1, k;

i++;          // i=i+1 → i=2
++j;          // j=j+1 → j=2
k=i++ + 2;    // k=4, i=3
k=++j + 2;    // k=5, j=3
```

Operadores Binários x Unários

- Operadores binários são aplicados a duas variáveis e retornam um terceiro valor
  - Ex.: +, - (subtração), /, *, %
- Operadores unários são aplicados a uma variável e alteram o valor da variável
  - Ex.: - (Troca de sinal), ++, --

Operadores relacionais

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

Operadores lógicos

Operador	Ação
&&	E
||	Ou
!	Não

Tabela verdade

- Resultado das operações aplicadas sobre operadores lógicos

x	y	x && y	x || y	! x
V	V	V	V	F
V	F	F	V	-
F	F	F	F	V
F	V	F	V	-

Operadores lógicos bit a bit

- Operadores binários são aplicados a duas variáveis e retornam um terceiro valor.
- Aplicados apenas a char, int ou longint

Operador	Ação
&	E
|	Ou
^	Ou exclusivo (XOR)
~	Não
>>	Deslocamento de bits a direita
<<	Deslocamento de bits a esquerda

Operadores lógicos bit a bit

```
char i=2, j; // Em binário i=00000010

j=i|1;    // 00000010 | 00000001 → j=3
j=i&j;    // 00000010 & 00000011 → j=2
j=i^j;     // 00000010 → j= 00000000, j=2
i=i>>1;   // 00000010 → i=00000001, i=1
i=i<<2;   // 00000001 → i=00000100, i=4
```

Expressões

- Expressões são combinações de variáveis, constantes e operadores

- Quando montamos expressões temos que levar em consideração a ordem com que os operadores são executados, conforme a tabela de precedência da linguagem C

Conversão de tipos em expressões

- Quando o C avalia expressões onde temos variáveis de tipos diferentes o compilador verifica se as conversões são possíveis. Se não são, ele não compilará o programa, dando uma mensagem de erro.

- Se as conversões forem possíveis ele as faz, seguindo as regras abaixo:
  - Todos os **char**s e **short int**s são convertidos para **int**s
  - Todos os **float**s são convertidos para **double**s.
  - Para pares de operandos de tipos diferentes: se um deles é **long double** o outro é convertido para **long double**; se um deles é **double** o outro é convertido para **double**; se um é **long** o outro é convertido para **long**; se um é **unsigned** o outro é convertido para **unsigned**

Modeladores (casts)

- Um modelador é aplicado a uma expressão
- Ele força a mesma a ser de um tipo especificado. Sua forma geral é:

```
(tipo)expressão
```

- Exemplo:

```
int num;
float f;
num=10;
f=(float)num/7;
```

Expressões abreviadas

Expressão Original	Expressão 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...	

Encadeando expressões: o operador ,

- O operador **,** determina uma lista de expressões que devem ser executadas sequencialmente.
- A vírgula diz ao compilador: execute as duas expressões separadas pela vírgula, em sequência.
- O valor retornado por uma expressão com o operador , é sempre dado pela expressão mais à direita.
- Pode-se encadear quantos operadores , forem necessários

- Exemplo:

```
x=(y=2, y+3); // x=5
```

o valor 2 vai ser atribuído a **y**, se somará 3 a **y** e o retorno (5) será atribuído à variável **x** .

Tabela de precedências do C

() [] ->
! ~ ++ -- . -(unário) (cast) *(unário) &(unário) sizeof
* / %
+ -
<< >>
<<= >>=
== !=
&
^
|
&&
||
?
= += -= *= /=
,

Tabela de precedências do C (cont.)

- Caso dois operadores de mesmo nível de precedência estejam em uma expressão sem parênteses, será executado da esquerda para direita
- Em caso de dúvida deve-se utilizar parênteses para explicitar a ordem de execução
- Exemplo:

```
i = j + 3 - k; // Executa primeiro a soma depois a subtração

i = j + (3 - k); // Primeiro a subtração depois a soma
```

Exercícios

- Nas expressões abaixo qual o valor de x, y, z1 e z2 ?

```
int a = 17, b = 3;
int x, y;
float z = 17. , z1, z2;
x = a / b;
y = a % b;
z1 = z / b;
z2 = a/b;
```

- Nas expressões abaixo qual o valor de x, y e z ?

```
x=23;
y=x++;
z=++x;
x=x+y-(z--);
```

- Qual o valor final de j ?

```
int i = 5, j =7;
if ( (i > 3) && ( j <= 7)  && ( i != j) )  j++;
```

Exercício - Programa

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

void main() {
  int a = 17, b = 3;
  int x, y;
  float z = 17. , z1, z2;
  x = a / b;
  y = a % b;
  z1 = z / b;
  z2 = a/b;
  printf("\nx=%d, y=%d, z1=%0.3f, z2=%0.3f", x, y, z1, z2);

  x=23;
  y=x++; // y=23, x=24
  z=++x; // z=25, x=25
  x=x+y-(z--); // x=23, z=24
  printf("\nx=%d, y=%d, z=%0.3f", x, y, z);

  int i=5, j=7;
  if ((i > 3) && (j <= 7) && (i != j))
    j++;
  printf("\nj=%d", j);
  getch();
}
```

Exercício - Resultado

```
x=5, y=2, z1=5.667, z2=5.000

x=23, y=23, z=24.000

j=8
```