1. Operadores

1.1 – Operador de Atribuição (=)

sintaxe:
nome_variável = expressão ;
ex.:

```
y = 2 ;                 /*  atribui o valor 2 a y  */x = 4 * y + 3 ;          /*  atribui o valor
da expressão a x  */
```

Observação: **Conversão de Tipos em Atribuições**
Regra: O valor do lado direito de uma atribuição é convertido no **tipo** do lado esquerdo.

ex.:

```
int     x ;
char    c ;
float   f ;
```

c = x ; => Os bits mais significativos da variável inteira x são ignorados, deixando c com os bits menos significativos. Se x está entre 0 è 256, então c e x têm valores idênticos.

x = f ; => x recebe a parte inteira de f.

f = c ; => f converte o valor inteiro de 8 bits armazenado em c no mesmo valor em formato de ponto flutuante.

f = x ; => f converte um valor inteiro no formato de ponto flutuante.

Cuidado:

```
float   f ;int    x ;

x = f = 3.5 ;    /*  resulta em f = 3.5  e  x = 3  */
f = x = 3.5 ;    /*  resulta em f = 3.0  e  x = 3  */
```

1.2 - Operadores Aritméticos

1.2.1 - Operadores Unários - atuam sobre apenas um operando

-	**(menos unário)**	multiplica o operando por (-1)
++	**(incremento)**	incrementa o operando em uma unidade
--	**(decremento)**	decrementa o operando em uma unidade

ex.: x = 2 ; e y = 4*x + 3 ;

++x incrementa o valor de x antes de usá-lo (portanto usaremos x = 3 e teremos y = 15)**x++** incrementa o valor de x depois de usá-1o (portanto usaremos x = 2 e teremos y = 11)**--x** decrementa o valor de x antes de usá-lo (portanto usaremos x = 1 e teremos y = 7)**x--** decrementa o valor de .x depois de usá-lo (portanto usaremos x = 2 e teremos y = 11)

1.2.2 - Operadores Binários - atuam sobre dois operandos

\+ (adição)**-** (subtração)
***** (multiplicação)/ (divisão)**%** (mod) - fornece o resto da divisão de 2 nºs inteiros

ex.: 10 % 2 = **0** 11 % 2 = **1** 10 2

11 2

0 5 **1** 5

1.2.3 – Precedência

++ , --
* , / , %
+ , -

Para alterar a precedência basta colocar a expressão entre parênteses. Quando dois operandos têm o mesmo nível de precedência, eles são avaliados da esquerda para a direita.

ex.:

```
(x + y) / 2     /*  será feito 1º a soma e depois a divisão  */
 x / y * 2      /*  será feito 1º a divisão e depois a multiplicação  */
```

1.3 - Operadores de Atribuição Compostos

sintaxe:
expressão_1 operador = expressão_2 é equivalente a

expressão_1 = expressão_1 operador expressão_2

ex.:

x = x * 5	**x * = 5**		
a = a + 1	**a + = 1**	ou	a ++
x = x / b	**x / = b**		
y = y – 1	**y - = 1**	ou	--y

1.4 - Operadores Relacionais

São usados para **comparar** expressões. Resultam em falso ou verdadeiro.

= =	(igual – comparação) - compara se 2 valores são iguais
>	(maior que)
<	(menor que)
>=	(maior ou igual)
<=	(menor ou igual)
!=	(diferente)

ex.:

```
4 = = 3          /*  resulta em falso  */
3 > 2            /* resulta em verdadeiro  */
```

1.4.1 - Precedência

< , <= , > , >=
! = , = =

1.5 - Operadores Lógicos

Permitem **relacionar** duas ou mais expressões.

&& (e) - resulta em verdadeiro se ambas expressões forem verdadeiras
| | (ou) - resulta em verdadeiro se pelo menos uma expressão for verdadeira
! (não) - resulta em verdadeiro se a expressão for falsa

ex.:

```
(5 > 2)  &&  (3 != 2)        /*  resulta em verdadeiro – ambos verdadeiros  */
(5 < 2)  &&  (3 != 2)        /*  resulta em falso – apenas 1 verdadeiro  */
(5 < 2)  &&  (3 = = 2)       /*  resulta em falso – ambos falsos  */

(3 >= 2)  ||  (4 != 2)       /*  resulta em verdadeiro – ambos verdadeiros */
(3 >= 2)  ||  (4 = = 2)      /*  resulta em verdadeiro – pelo menos 1 verdadeiro */
(3 <= 2)  ||  (4 = = 2)      /*  resulta em falso – ambos falsos  */

!(4 = = 2)          /*  resulta em verdadeiro – pois a expressão é falsa  */
!(4 != 2)           /*  resulta em falso – pois a expressão é verdadeira  */
```

1.6- Operadores bit a bit

Permite a manipulação direta com bits de variáveis inteiras. Os operadores bit a bit só podem operar sobre variáveis do tipo **int** e **char**.

1.6.1 - Operadores de deslocamento

Desloca os bits de **n** posições dentro de uma variável inteira

sintaxe:

- Deslocamento à esquerda

variável << número de posições de bits

- Deslocamento à direita

variável >> número de posições de bits

ex.: x = 00001100 /* 00001100 é igual a $12_{(10)}$ */

x >> 2 /* resulta em x = 00000011 é igual a $3_{(10)}$ */

x << 2 /* resulta em x = 00110000 é igual a $48_{(10)}$ */

Observação:

Notar que as operações de deslocamento podem ser usadas para multiplicar e dividir inteiros:

x << y é equivalente a $x * 2^y$

x >> y é equivalente a $x / 2^y$

1.6.2 - Operadores lógicos com bits

~ (complemento de 1) - inverte o estado de cada bit da variável.
& (e) - resulta em 1 somente se ambos operandos forem l, caso contrário, o bit é zerado.
| (ou inclusivo) - zera somente se ambos operandos forem 0, caso contrário, resulta em 1.
^ (ou exclusivo) - resulta em 1 se ambos operandos forem diferentes, caso contrário, o bit é zerado.

ex.:

```
  11110000        01010000        11110000
 &01010101       |11110000       ^01010101
 --------        --------        --------
  01010000        11110000        10100101
```

~ (11111101) = (00000010)