parte2 _ Operadores e Expressões

Marcos Silvano / DACOM

BCC32A-Algoritmos 1

Operadores

Operadores Aritméticos

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiplies both operands	A * B will give 200
/	Divides numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	**Increment operator** , increases integer value by one	A++ will give 11
--	**Decrement operator** , decreases integer value by one	A-- will give 9

Figure 1:

Operadores Aritméticos: Exemplo

```c
#include <stdio.h>
main() {
   int a = 21;   int b = 10;   int c ;

   c = a + b;
   printf("Line 1 - Value of c is %d\n", c );

   c = a - b;
   printf("Line 2 - Value of c is %d\n", c );

   c = a * b;
   printf("Line 3 - Value of c is %d\n", c );

   c = a / b;
   printf("Line 4 - Value of c is %d\n", c );

   c = a % b;
   printf("Line 5 - Value of c is %d\n", c );

   c = a++;
   printf("Line 6 - Value of c is %d\n", c );

   c = a--;
   printf("Line 7 - Value of c is %d\n", c );
}
```

Operadores Aritméticos: Expressões

```c
#include <stdio.h>

main() {
    printf("Entre com X: ");
    float x;
    scanf("%f", &x);

    printf("Entre com Y: ");
    float y;
    scanf("%f", &y);

    // precedência de operadores * / -> + -
    // usamos parênteses para forçar a ordem de avaliação da expressão
    // símbolos comuns em expressões aritméticas, [ ] e { }, possuem outros signific

    float res = (x+y)*2 + 15/x;
    printf("Resultado: %f", res);

    return 0;
}
```

Operadores Aritméticos: Expressões

- Qual o valor de a em `float a = 5/2`?

```
#include <stdio.h>

main() {
    printf("Polinômio: 5X + 5/2\n");
    printf("-----------------------------------\n");
    printf("Entre com o valor de X: \n");

    float x;
    scanf("%f", x);

    // divisão em C/C++ retorna o tipo dos operandos
    // operandos int -> retorna int
    // ao menos um operando float -> retorna float
    float y = 5*x + 5/2.0f;

    printf("Resultado: %.2f", y);

    return 0;
}
```

Operadores Relacionais

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Figure 2:

Operadores Relacionais: Exemplo

```
#include <stdio.h>
main() {
   int a = 21;   int b = 10;
   if( a == b ) {
      printf("Line 1 - a is equal to b\n" );
   } else {
      printf("Line 1 - a is not equal to b\n" );
   }
   if ( a < b ) {
      printf("Line 2 - a is less than b\n" );
   } else {
      printf("Line 2 - a is not less than b\n" );
   }
   if ( a > b ) {
      printf("Line 3 - a is greater than b\n" );
   } else {
      printf("Line 3 - a is not greater than b\n" );
   }
   a = 5;   b = 20;
   if ( a <= b ) {
      printf("Line 4 - a is either less than or equal to  b\n" );
   }
   if ( b >= a ) {
      printf("Line 5 - b is either greater than  or equal to b\n" );
   }
```

Operadores Lógicos

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false.	!(A && B) is true.

Figure 3:

Operadores Lógicos: Questões

- Há expressões lógicas verdadeiras abaixo?

```
#include <stdio.h>

main() {
    int a = 2;
    int b = 2;
    int c = 2;

    printf("%d\n", (a == b == c) );
    printf("%d\n", ((a == b) == c) );
    printf("%d\n", (a == (b == c)) );
}
```

Operadores Lógicos: Exemplo

```c
#include <stdio.h>
main() {
   int a = 5;
   int b = 20;
   int c ;

   if ( a && b ) {
      printf("Line 1 - Condition is true\n" );
   }
   if ( a || b ) {
      printf("Line 2 - Condition is true\n" );
   }

   a = 0;
   b = 10;
   if ( a && b ) {
      printf("Line 3 - Condition is true\n" );
   } else {
      printf("Line 3 - Condition is not true\n" );
   }
   if ( !(a && b) ) {
      printf("Line 4 - Condition is true\n" );
   }
}
```

Operadores Lógicos: Precedência

```c
#include <stdio.h>

main() {
   int a = 2;
   int b = 5;
   int c = 'm';

   if( (a == 2 || b > 10) && c == 'm') {
       printf("condition 1");
   }

   b = 20;
   c = 'k';
   if( (a == 2 && b > 10) || c == 'm' ) {
       printf("condition 2");
   }

   return 0;
}
```

Operadores Bit a bit (Bitwise)

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

Figure 4:

Operadores Bit a bit: Exemplo

```c
#include <stdio.h>

main() {
   unsigned int a = 60; /* 60 = 0011 1100 */
   unsigned int b = 13; /* 13 = 0000 1101 */
   int c = 0;

   c = a & b;        /* 12 = 0000 1100 */
   printf("Line 1 - Value of c is %d\n", c );

   c = a | b;        /* 61 = 0011 1101 */
   printf("Line 2 - Value of c is %d\n", c );

   c = a ^ b;        /* 49 = 0011 0001 */
   printf("Line 3 - Value of c is %d\n", c );

   c = ~a;           /*-61 = 1100 0011 */
   printf("Line 4 - Value of c is %d\n", c );

   c = a << 2;     /* 240 = 1111 0000 */
   printf("Line 5 - Value of c is %d\n", c );

   c = a >> 2;     /* 15 = 0000 1111 */
   printf("Line 6 - Value of c is %d\n", c );
}
```

Operadores de atribuição

- Operador de atribuição: $=$
  - C/C++ e linguagens derivadas: Java, C#, JavaScript, PHP, etc.
- Igualdade é verificada pelo operador: $==$
  - C/C++ e linguagens derivadas
- C/C++ e derivadas permitem contrações de outros operadores com a atribuição

```
// As contrações agilizam a escrita, mas são opcionais
a += 2;  // a = a + 2
a %= 2;  // a = a % 2
```

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand.	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator.	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator.	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator.	C &= 2 is same as C = C & 2
^=	Bitwise exclusive OR and assignment operator.	C ^= 2 is same as C = C ^ 2
|=	Bitwise inclusive OR and assignment operator.	C |= 2 is same as C = C | 2

Operadores de Atribuição: Exemplos

```c
#include <stdio.h>

main() {
   int a = 21;
   int c ;

   c =  a;
   printf("Line 1 - =  Operator Example, Value of c = %d\n", c );

   c +=  a;
   printf("Line 2 - += Operator Example, Value of c = %d\n", c );

   c -=  a;
   printf("Line 3 - -= Operator Example, Value of c = %d\n", c );

   c *=  a;
   printf("Line 4 - *= Operator Example, Value of c = %d\n", c );

   c /=  a;
   printf("Line 5 - /= Operator Example, Value of c = %d\n", c );
}
```

Operadores de Atribuição: Exemplos

```c
#include <stdio.h>

main() {
   int a = 21;
   int c ;

   c  = 200;
   c %=  a;
   printf("Line 6 - %= Operator Example, Value of c = %d\n", c );

   c <<=  2;
   printf("Line 7 - <<= Operator Example, Value of c = %d\n", c );

   c >>=  2;
   printf("Line 8 - >>= Operator Example, Value of c = %d\n", c );

   c &=  2;
   printf("Line 9 - &= Operator Example, Value of c = %d\n", c );

   c ^=  2;
   printf("Line 10 - ^= Operator Example, Value of c = %d\n", c );

   c |=  2;
   printf("Line 11 - |= Operator Example, Value of c = %d\n", c );
}
```

Operador Ternário

- Permite uma simplificação de atribuição condicional
- Também pode ser usado como condicional simples

```c
#include <stdio.h>

int main () {
    int x1, x2, y = 10;

    x1 = (y < 10) ? 30 : 40;

    // o código acima faz o mesmo que o condicional abaixo
    if(y < 10) {
        x2 = 30;
    } else {
        x2 = 40;
    }

    printf("value of x1: %d", x1);
    printf("value of x2: %d", x2);

    return 0;
}
```

Precedência de Operadores

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Precedência de Operadores: Exemplo

```c
#include <stdio.h>

main() {
   int a = 20;
   int b = 10;
   int c = 15;
   int d = 5;
   int e;

   e = (a + b) * c / d;       // ( 30 * 15 ) / 5
   printf("Value of (a + b) * c / d is : %d\n",  e );

   e = ((a + b) * c) / d;     // (30 * 15 ) / 5
   printf("Value of ((a + b) * c) / d is  : %d\n" ,  e );

   e = (a + b) * (c / d);    // (30) * (15/5)
   printf("Value of (a + b) * (c / d) is  : %d\n",  e );

   e = a + (b * c) / d;      //  20 + (150/5)
   printf("Value of a + (b * c) / d is  : %d\n" ,  e );

   return 0;
}
```