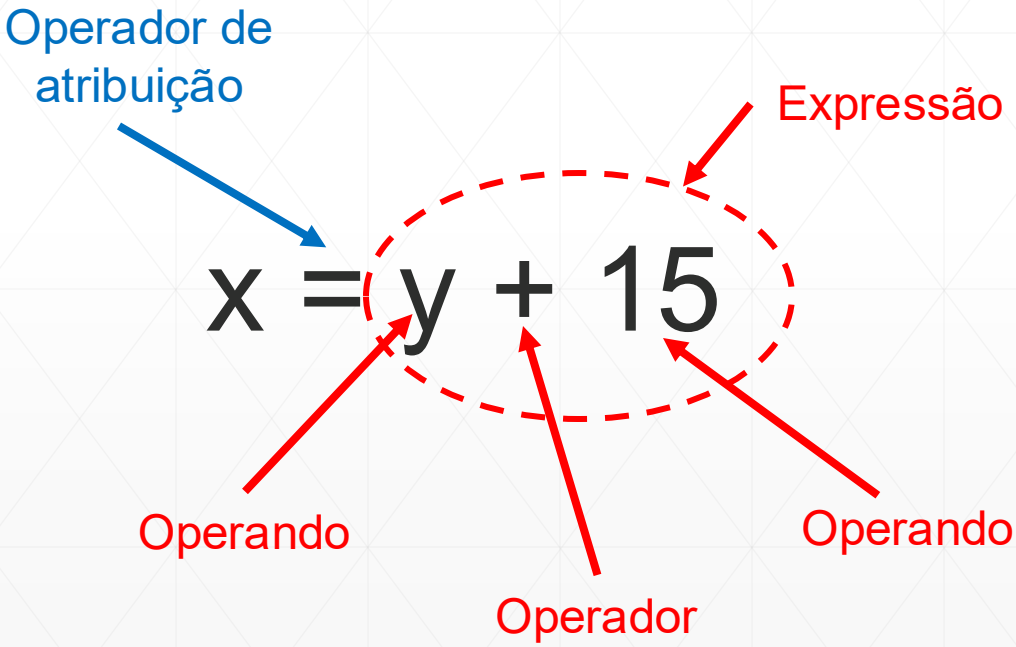
Técnicas de Programação

Prof. Protásio

Laboratório de Microengenharia/DEE/CEAR/UFPB

Expressões

- Expressão
  - É uma sequência de **operandos** e **operadores** que especifica uma operação.



Operadores

- O significado de um **operador** depende dos **tipos de seus operandos**

$$m + n$$

- O operador + pode ser:
  - Soma de inteiros
  - Soma de pontos flutuantes
  - Concatenação de *strings*
  - Soma de ponteiros

Sobrecarga de Operador

Operadores

- Classificação:
  - Unário
    - Atuam sobre um operando
      - Exemplo: &P
  - Binário
    - Atuam sobre dois operandos
      - Exemplo: x * y
  - Ternário
    - Atuam sobre três operandos
      - Exemplo: a?b:c

Expressões com vários operandos

- É importante saber:
  - **Precedência**
    - Indica a prioridade do operador em relação a outros.
    - Exemplo:

20*5+24

  - **Associatividade**
    - Indica a ordem de avaliação de operandos do mesmo tipo em uma expressão
    - Exemplo:

20+5+24 equivale à (20+5)+24 : associatividade L-R

x + 32760 + y + 5 equivale a (((x + 32760) + y) + 5) : associatividade L-R

m=n=p equivale à m = (n = p) : associatividade R-L

Operadores

Prioridade	Grupo de precedência	Operador	Descrição	Associatividade
1	Escopo	::	Qualificador de escopo	LR
2	Sufixo (unário)	++ - -	Incremento e decremento pós fixado	LR
		( )	Parêntese	
		[ ]	Índice	
		. ->	Acesso a membros de objetos	
3	Prefixo (unário)	++ - -	Incremento e decremento prefixado	RL
		~ !	NOT bitwise / NOT lógico	
		+ -	Prefixo unário	
		& *	Referência / Derreferência	
		new delete	Alocação / desalocação	
		sizeof	Comprimento de parâmetro	
		(*type*)	Casting (conversão explícita)	
4	Ponteiro para membros	.* ->*	Acesso de membros de objetos via ponteiro	LR
5	Aritmético	* / %	Multiplicação, divisão, módulo	LR
6	Aritmético	+ -	Adição, subtração	LR

Operadores

Prioridade	Grupo de precedência	Operador	Descrição	Associatividade
7	Deslocamento *bitwise*	<< >>	Deslocamento à esquerda, deslocamento à direita	LR
8	Relacional	< > <= >=	Operadores de comparação	LR
9	Igualdade	== !=	igualdade / desigualdade	LR
10	AND	&	AND *bitwise*	LR
11	XOR	^	XOR *bitwise*	LR
12	OR	|	OR *bitwise*	LR
13	AND lógico	&&	AND lógico	LR
14	OR lógico	||	OR lógico	LR
15	Atribuição	= *= /= %= += -= >>= <<= &= ^= |=	Atribuição / Atribuição composta	RL
		?:	Operador condicional	RL
16	Sequência	,	Operador vírgula	LR

É boa prática utilizar parênteses para melhorar a inteligibilidade do código.

Precedência

6 + 3 * 4 / 2 - 2

* e / têm maior precedência que + e -
* e / têm mesma precedência, então se usa regra da associatividade LR

6 + 12 / 2 - 2

/ têm maior precedência que + e -

6 + 6 - 2

+ e - têm mesma precedência, então se usa regra da associatividade LR

12 - 2

10

Operadores de atribuição

Símbolo	Uso	Operação
=	x = 6;	Atribui 6 a x
*=	x *= 6;	x = x * 6;
/=	x /= 6;	x = x / 6;
%=	x %= 6;	x = x % 6;
+=	x += 6;	x = x + 6;
-=	x -= 6;	x = x - 6;
<<=	x <<= 6;	x = x << 6;
>>=	x >>= 6;	x = x >> 6;
&=	x &= 6;	x = x & 6;
^=	x ^= 6;	x = x ^ 6;
|=	x |= 6;	x = x | 6;

Operadores de atribuição

- Exemplo

```
#include <iostream>
using namespace std;
int main () {
        int a, b=3;
        a = b;
        cout << a << endl;
        a+=2; // equivalente a a=a+2
        cout << a;
        a<<=1; // equivalente a a=a<<1
        cout << a;
}
```

Situação-problema:

- Faça um programa que receba do usuário um valor inteiro e multiplique por 2 usando o operador de deslocamento <<.
- É possível dividir 2 usando operador de deslocamento?

Operadores aritméticos

Operador	Descrição	Exemplo
+	soma	4 + 5
-	subtração	7 – 3
*	produto	4 * 5
/	Divisão	8 / 5
%	Resto	8 % 5

```
#include <iostream>
using namespace std;
int main () {
        cout << 10/3 << endl;
        cout << 10/3.0 << endl;
        cout << 10%3 << endl;
}
```

int / int = int

int / float = float

Operadores de incremento e decremento

Operador	Equivalência
++x;	x+=1; x = x +1;
--x;	x-=1; x = x -1;

OBS

```
#include <iostream>
using namespace std;
int main () {
        int y, x = 3;
        y = ++x;      // x será 4, y será 4
        cout << x << endl;
        cout << y << endl;
}
```

```
#include <iostream>
using namespace std;
int main () {
        int y, x = 3;
        y = x++;      // x será 4, y será 3
        cout << x << endl;
        cout << y << endl;
}
```

Operadores relacionais

Operador	Descrição	Exemplo
==	Igual a	x == y
!=	Diferente de	x != y
>	Maior que	x > y
<	Menor que	x < y
>=	Maior ou igual que	x >= y
<=	Menor ou igual que	x <= y

- O resultado de uma operação relacional é:
  - 0 : false
  - 1 : true

Operadores relacionais

- Exemplo
  - z = 1 < 5;
    - Resultado z = 1
  - z = 1 > 5;
    - Resultado z = 0
- Em geral, os operadores relacionais são usados em sentenças de seleção (*if*) ou de iteração (*while*, *for*)
  - Exemplo:

```
#include <iostream>
using namespace std;
int main () {
    int y;
    cout << "Digite a media do aluno entre 0 e 10" << endl;
    cin  >> y;
    if (y < 7)
        cout << "Reprovado" << endl;
    else
        cout << "Aprovado" << endl;
    return 0;
}
```

Situação-problema 1:

- Modifique o código abaixo para que se o aluno for **reprovado**, também mostre na tela a expressão "**Tente novamente, você consegue**."

Situação-problema 2:

- Modifique o código para que se o aluno for **aprovado com nota maior que 9**, mostre na tela a expressão "**Aprovado e Parabéns**.", mantendo as condições anteriores

Operadores lógicos (booleanos)

Operador	Descrição	Exemplo
!	NOT	!(x >=y)
&&	AND	(m < n) && (i > j)
||	OR	(m == 5) || (i > j)

- Exemplo

```
#include <iostream>
using namespace std;
int main () {
    char c;
    cout << "Digite uma letra:" << endl;
    cin  >> c;
    if (c >= 'a' && c <='z' || c >= 'A' && c <='Z')
        cout << "Eh letra!" << endl;
    else
        cout << "Nao eh letra" << endl;
    return 0;
}
```

Situação-problema 1:
- Modifique o código para identificar se o caractere digitado é um número.

Situação-problema 2:
- Modifique o código para identificar se o caractere digitado é um '.' ou se é '-'.

Operadores de manipulação bit-a-bit (*bitwise*)

Operador	Descrição	Exemplo
&	AND bit-a-bit	a & b
|	OR bit-a-bit	a | b
^	XOR bit-a-bit	a ^ b
~	INVERSOR bit-a-bit	~a
<<	Deslocamento de bits à esquerda	a << 3
>>	Deslocamento de bits à direita	a >> 2

Situação-problema:
- Faça um código que receba um char e mostre na tela se seu 1º bit (LSB) é 1 ou 0.

a = 1	00000001	a = 1	00000001	a = 3	00000011
b = 2	00000010	b = 2	00000010	b = 2	00000010
a & b	00000000	a | b	00000011	a ^ b	00000001

a = 3	00000011	a = 4	00000100	a = 32	00100000
~a	11111100	a<< 3	00100000	a>> 2	00001000

OBS: operadores **bitwise** aplicam-se somente à **char**, **int** e **long**

Operadores de manipulação bit-a-bit (*bitwise*)

- Teste de bit

```
#include <iostream>
using namespace std;

int main () {
        char x, temp;
        cout << "digite valor de x: " << endl;
        cin >> x;
        cout << "O valor de x digitado eh: " << x << endl;

        temp = x & 0b00000001;

        if (temp >= 1)
                cout << "O 1o bit eh 1" << endl;
        else
                cout << "O 1o bit eh 0" << endl;

        temp = x & 0b00000010;

        if (temp >= 1)
                cout << "O 2o bit eh 1" << endl;
        else
                cout << "O 2o bit eh 0" << endl;

        temp = x & 0b00000100;

        if (temp >= 1)
                cout << "O 3o bit eh 1" << endl;
        else
                cout << "O 3o bit eh 0" << endl;

        return 0;
}
```

Resultado

```
protasio@DELL:~/CODEs$ ./teste
digite valor de x:
a
O valor de x digitado eh: a
O 1o bit eh 1
O 2o bit eh 0
O 3o bit eh 0
protasio@DELL:~/CODEs$ ./teste
digite valor de x:
b
O valor de x digitado eh: b
O 1o bit eh 0
O 2o bit eh 1
O 3o bit eh 0
```

Operadores de manipulação bit-a-bit (*bitwise*)

- Teste de bit

Resultado

```
protasio@DELL:~/CODEs$ g++ -o teste aula3_2.cpp
protasio@DELL:~/CODEs$ ./teste
digite valor de x:
b
    x= 01100010
  pos= 00000001
 temp= 00000000
0
    x= 01100010
  pos= 00000010
 temp= 00000010
1
    x= 01100010
  pos= 00000100
 temp= 00000000
0
    x= 01100010
  pos= 00001000
 temp= 00000000
0
    x= 01100010
  pos= 00010000
 temp= 00000000
0
    x= 01100010
  pos= 00100000
 temp= 00100000
1
    x= 01100010
  pos= 01000000
 temp= 01000000
1
    x= 01100010
  pos= 10000000
 temp= 00000000
0
```

```
#include <iostream>
#include <bitset>
using namespace std;

int main () {
        char x, temp, pos = 0b00000001;

        bitset<8> B;

        cout << "digite valor de x:" << endl;
        cin >> x;
        for (int i = 0; i < 8; i++){
                temp = x & pos;
                B = x;
                cout << " x= " << B << endl;
                B = pos;
                cout << " pos= " << B << endl;
                B = temp;
                cout << " temp= " << B << endl;

                if (temp >= 1)
                   cout << "1 ";
                else
                   cout << "0 ";

                pos = pos << 1;

                cout << endl;
        }
        return 0;
}
```

Operador condicional

- Operador ternário
- Sintaxe: (condição) ? valor1 : valor2
- Exemplo

```
y = (x < 20) ? 50 : 70; // y recebe 50 se x for menor que 20, senão 70.
```

- Equivalente

```
if (x < 20) // Se x for menor que 20
    y = 50; // então y recebe 50
else
    y = 70; // senão y recebe 70
```

Operador condicional ternário

- Exemplo: sensor de temperatura

```
#include <iostream>
#include <cstring>
using namespace std;
int main () {
    int temp;
    string y = “oi string“;
    cout << "Digite a temperatura" << endl;
    cin  >> temp;
    y =  (temp > 212) ? “fogo” : “okay”;
    cout << “Status“<< y  << endl;
    return 0;
}
```

Conversão de tipos

- Conversões implícitas
  - São aquelas executadas automaticamente e tem como regra a **promoção de tipos:**

Tipo de precisão mais baixa → Tipo de precisa mais alta

  - Exemplos:
    - float pi = 3.14 + 3;

int

↓

float

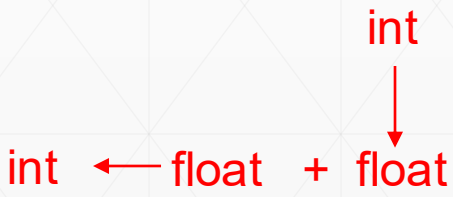
Resultado pi = 6.14

Conversão de tipos

- Conversões implícitas
  - São aquelas executadas automaticamente e tem como regra a **promoção de tipos:**

Tipo de precisão mais baixa → Tipo de precisa mais alta

  - Exemplos:
    - `int pi = 3.14 + 3;`



Resultado pi = 6

Conversão de tipos

Situação-problema:

- Faça um código em que o usuário digite um valor em ponto flutuante e imprima o valor inteiro usando **casting**.
- Refaça o código **sem casting** e veja o resultado. Explique!

- Conversão Explícita (*cast*)
  - Sintaxe:
    - tipo_de_dado (expressão)
    - (tipo_de_dado) expressão

2 opções

  - Exemplo

```
double a = 3.27;
int b;
b = int (a);          // casting de double para int
b = (int) a;          // casting de double para int
```

Conversão de tipos

- Conversão Explícita (*cast*)
  - Exemplos:

```
int  i  = 7, j = 2;
double m = i/j;                    // truncamento m = 3.0
double m = (double)(i/j);          // m = 3.0
double m = (double)i/j;            // m = 3.5, pois i é convertido
double m = i/(double)j;            // m = 3.5, pois j é convertido
double m = (double)i/(double)j;    // m = 3.5
```

int = int / int

float = float / int = int / float = float / float

Lista de exercícios

- Elabore um código em C++ que receba do usuário dois valores, um valor char ***a*** e um inteiro ***n*** e:
  1. Mostre na tela se o caractere **a** é letra ou número (0 à 9);
  2. Mostre na tela o valor em binário do ***n***-ésimo bit de ***a***, dado que n esteja entre 1 e 8;
  3. Mostre na tela se **n** é par ou ímpar;
  4. Mostre na tela valor de $\mathbf{f(n)=n^2+5n+2}$;
  5. Mostre na tela uma semi-pirâmide, como mostrada abaixo, em que sua altura seja de acordo como o valor de **n** dado e o caractere da pirâmide seja o valor de **a**. No exemplo abaixo, a pirâmide é para n = 6 e a = 'x'.

```
x
xx
xxx
xxxx
xxxxx
xxxxxx
```