

Programação de Computadores

OPERADORES LÓGICOS

Introdução

- A **instrução if** permite a um programa **tomar decisões**
 - **Se** (a nota for maior que 7) **o aluno** está aprovado
 - **Se** (o sensor detectar movimento) **dispare** o alarme
 - **Se** (os objetos colidirem) **emita** o som de explosão
 - **Se** (o nível do rio exceder o limite) **abra** a comporta
- Ele faz isso através de **desvios** em sua execução
 - Dependem do resultado de um (**teste condicional**)

Introdução

- Frequentemente os testes possuem mais de uma condição:
 - Para um caractere ser minúsculo ele deve ser
<maior ou igual a 'a'> E <menor ou igual a 'z'>
 - Um aluno está reprovado direto se ele possuir
<média final inferior a 3.5> OU <mais de 16 faltas>
- Por isso C++ oferece três operadores lógicos:
 - && (E), || (OU) e ! (NEGAÇÃO)

Operador Lógico OU

- A expressão com **OU lógico** é verdadeira quando **uma ou ambas as condições são verdadeiras**, de acordo com a tabela verdade abaixo:

| A | B | A OU B |
|---|---|--------|
| F | F | F |
| V | F | V |
| F | V | V |
| V | V | V |

Operador Lógico OU

- O operador C++ equivalente ao OU lógico é o **operador ||**

```
5 > 3 || 5 > 10 // true  || false = true
5 > 8 || 5 < 10 // false || true  = true
5 < 8 || 5 > 2  // true  || true  = true
5 > 8 || 5 < 2  // false || false = false
```

- O operador || tem uma **precedência menor que os operadores relacionais**, por isso os parênteses não são necessários no exemplo acima

Operador Lógico OU

- O operador `||` é um ponto de sequência, ou seja, qualquer mudança feita do lado esquerdo ocorre antes da avaliação da expressão do lado direito
 - Se a variável `i` tem inicialmente o valor `10`, ela terá o valor `11` no momento da comparação com `j`

```
i++ < 6 || i == j
```

- Além disso, C++ não avaliará a expressão a direita se a expressão a esquerda é verdadeira

Operador Lógico OU

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

int main()
{
    cout << "Este programa pode formatar seu disco rígido\n"
           "e destruir todos os seus dados.\n"
           "Você deseja continuar? <s/n> ";

    char ch;
    cin >> ch;

    if (ch == 's' || ch == 'S')
        cout << "Você foi avisado!\a\a\n";
    else
        if (ch == 'n' || ch == 'N')
            cout << "Uma escolha sábia... Tchau!\n";
        else
            cout << "Você não respondeu corretamente, eu deveria apagar tudo!\n";
}
```

Operador Lógico OU

- Saída do Programa:

```
Este programa pode formatar seu disco rígido  
e destruir todos os seus dados.  
Você deseja continuar? <s/n> N  
Uma escolha sábia...Tchau!
```

- O programa **lê apenas um caractere**
 - Ele funciona mesmo que o usuário responda "não"
 - Apenas o 'n' será lido pelo programa

Operador Lógico E

- A expressão com **E lógico** é verdadeira apenas quando **ambas as condições são verdadeiras**, de acordo com a tabela verdade abaixo:

| A | B | A E B |
|---|---|-------|
| F | F | F |
| V | F | F |
| F | V | F |
| V | V | V |

Operador Lógico E

- O operador C++ equivalente ao E lógico é o **operador &&**

```
5 == 3 && 4 == 4 // false && true = false
5 > 3 && 5 > 10 // true && false = false
5 < 8 && 5 < 10 // true && true = true
5 > 8 && 5 < 2 // false && false = false
```

- O operador && tem uma **precedência menor que os operadores relacionais**, por isso os parênteses não são necessários no exemplo acima

Operador Lógico E

- O operador `&&` é um ponto de sequência, ou seja, qualquer mudança feita do lado esquerdo ocorre antes da avaliação da expressão do lado direito
 - Se a variável `i` tem inicialmente o valor `10`, ela terá o valor `11` no momento da comparação com `j`

```
i++ < 6 && i == j
```

- Além disso, C++ não avaliará a expressão a direita se a expressão a esquerda é falsa

Operador Lógico E

```
#include <iostream>
using namespace std;
const int VetTam = 6;
int main()
{
    int idades[VetTam];
    cout << "Entre com a idade dos seus vizinhos.\nO programa termina quando você tiver\n"
           << VetTam << " entradas ou digitar um valor negativo.\n";

    int num;
    int i = 0;
    bool tem_vaga = true;

    cout << "Idade: ";
    while (tem_vaga && cin >> num && num >= 0)
    {
        idades[i++] = num;
        if (i < VetTam)
            cout << "Idade: ";
        else
            tem_vaga = false;
    }
}
```

Operador Lógico E

```
if (i == 0)
    cout << "Sem dados, tchau!\n";
else
{
    cout << "Entre com sua idade: ";
    int voce;
    cin >> voce;

    int cont = 0;
    for (int j = 0; j < i; ++j)
    {
        if (idades[j] > voce)
            cont++;
    }

    cout << cont;
    cout << " dos seus vizinhos são mais velhos que você.\n";
}

return 0;
}
```

Operador Lógico E

- Saída do Programa:

Entre com a idade dos seus vizinhos.
O programa termina quando você tiver
6 entradas ou digitar um valor negativo.

Idade: 39

Idade: 45

Idade: 19

Idade: 28

Idade: -1

Entre com sua idade: 20

3 dos seus vizinho são mais velhos que você.

- O programa usa um contador para guardar quantos valores obedecem a uma condição

Operador Lógico E

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

const char * qualifica[4] = // vetor de ponteiros para char
{ "jovem.\n", "adulto.\n", "velho.\n", "bebê.\n" };

int main()
{
    int idade, indice;
    cout << "Entre com sua idade em anos: ";
    cin >> idade;

    if (idade > 17 && idade < 35)
        indice = 0;
    else if (idade >= 35 && idade < 50)
        indice = 1;
    else if (idade >= 50 && idade < 65)
        indice = 2;
    else
        indice = 3;

    cout << "Você se qualifica na categoria " << qualifica[indice];
}
```

Operador Lógico E

- Saída do Programa:

Entre com sua idade em anos: 3
Você se qualifica na categoria bebê.

- Observe que para o teste da faixa de valores o operador && deve unir duas expressões relacionais

```
if (idade > 17 && idade < 35)    // ok

if (17 < idade < 35)             // não faça isso
if ((17 < idade) < 35)           // sempre verdadeiro
```


Operador Lógico NÃO

- O **operador lógico de negação** inverte o valor verdade da expressão, **tornando-a falsa se ela é verdadeira**, e verdadeira se ela é falsa

| A | NÃO A |
|---|-------|
| F | V |
| V | F |

- O operador C++ equivalente a negação é o **operador !**

Operador Lógico NÃO

- Uma relação pode ser **expressa de forma mais clara** sem o operador de negação

```
if (!(x > 5))    // a mesma coisa que if (x <= 5)
```

- Porém ele é útil com **funções que retornam zero** em caso de sucesso

- A função `strcmp(s1, s2)` retorna zero se as strings `s1` e `s2` são iguais

```
// se as strings são iguais...  
if (!strcmp(s1,s2))
```

Operador Lógico NÃO

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

bool is_int(double);

int main()
{
    cout << "Entre com um valor inteiro: ";
    double num;
    cin >> num;
    while (!is_int(num)) // repete enquanto não é um inteiro
    {
        cout << "Fora da faixa, tente novamente: ";
        cin >> num;
    }
    int val = int (num); // type cast
    cout << "Você digitou o inteiro " << val << "\nTchau!\n";
}
```

Operador Lógico NÃO

```
bool is_int(double x)
{
    if (x >= INT_MIN && x <= INT_MAX)
        return true;
    else
        return false;
}
```

■ Saída do Programa:

Entre com um valor inteiro: **6234128679**
Fora da faixa, tente novamente: **-8000222333**
Fora da faixa, tente novamente: **99999**
Você digitou o inteiro **99999**
Tchau!

Operadores Lógicos

- Os operadores lógicos E e OU tem uma precedência menor que os operadores relacionais e aritméticos

```
// equivalente a (x > 5) && (x < 10)  
x > 5 && x < 10
```

- O operador lógico NÃO tem uma precedência maior que os operadores relacionais e aritméticos

```
!(x > 5)    // falso se x maior que 5  
!x > 5      // sempre falso, !x é convertido para 0 ou 1
```

Operadores Lógicos

- O **operador lógico E** tem uma precedência maior que o **operador lógico OU**

```
// equivalente a x > 300 || (y > 30 && y < 45)  
x > 300 || y > 30 && y < 45
```

- **Recomenda-se o uso de parênteses** para aumentar a legibilidade e clareza do código

```
(idade > 50) || ((peso > 80) && (donativo > 1000))
```

Operadores Lógicos

| Tabela de Precedências | | |
|------------------------|--------------------|-----------------|
| Símbolo | Significado | Associatividade |
| Aritméticos | ! - | Direita |
| | * / % | Esquerda |
| | + - | Esquerda |
| | < <= > >= | Esquerda |
| | == != | Esquerda |
| Relacionais | && | Esquerda |
| | | Esquerda |

Quanto mais alto na tabela maior a precedência

Resumo

- Os operadores lógicos podem ser usados para construir condições complexas para os testes de desvio
 - Operador && (AND)
 - Operador || (OR)
 - Operador ! (NOT)
- Existem versões binárias destes operadores
 - & (AND), | (OR), ~ (NOT), ^ (XOR), << (esq), >> (dir)