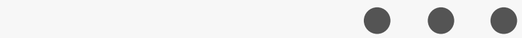
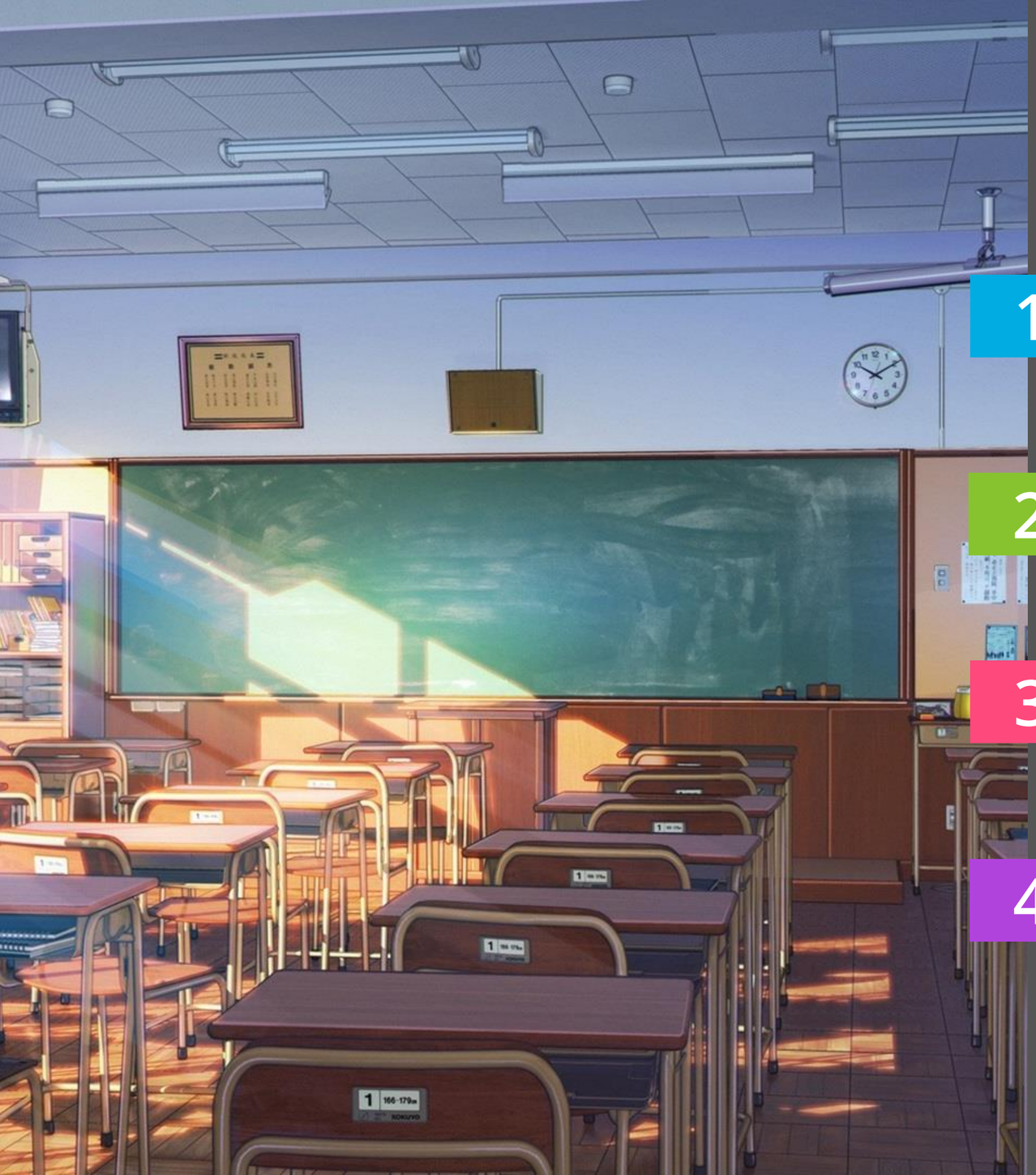
Condicionais e expressão lógica

Aula 02 - Programação orientada a objetos



Professor Daniel

INSTITUTO FEDERAL
Paraíba
Campus Campina Grande



Na aula de hoje

1. Expressões relacionais
2. Expressões lógicas
3. Comandos condicionais (IF e ELSE)
4. Exercícios

1

Expressões relacionais

Tipo booleano

Ou bool

Em C++ o tipo booleano especifica os valores booleanos False e True

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

int main() {

    bool escolha = false;
    std::cout << "Tipo: " << typeid(escolha).name() << std::endl;

    escolha = true;
    std::cout << "Tipo: " << typeid(escolha).name() << std::endl;

    int numero = 10;
    std::cout << "Tipo: " << typeid(numero).name() << std::endl;
    return 0;

}
```

Podemos criar variáveis associadas a booleanos, mas o uso mais comum é na verificação de resultados de expressões relacionais e lógicas.

Expressões

Já vimos que constantes e variáveis são expressões

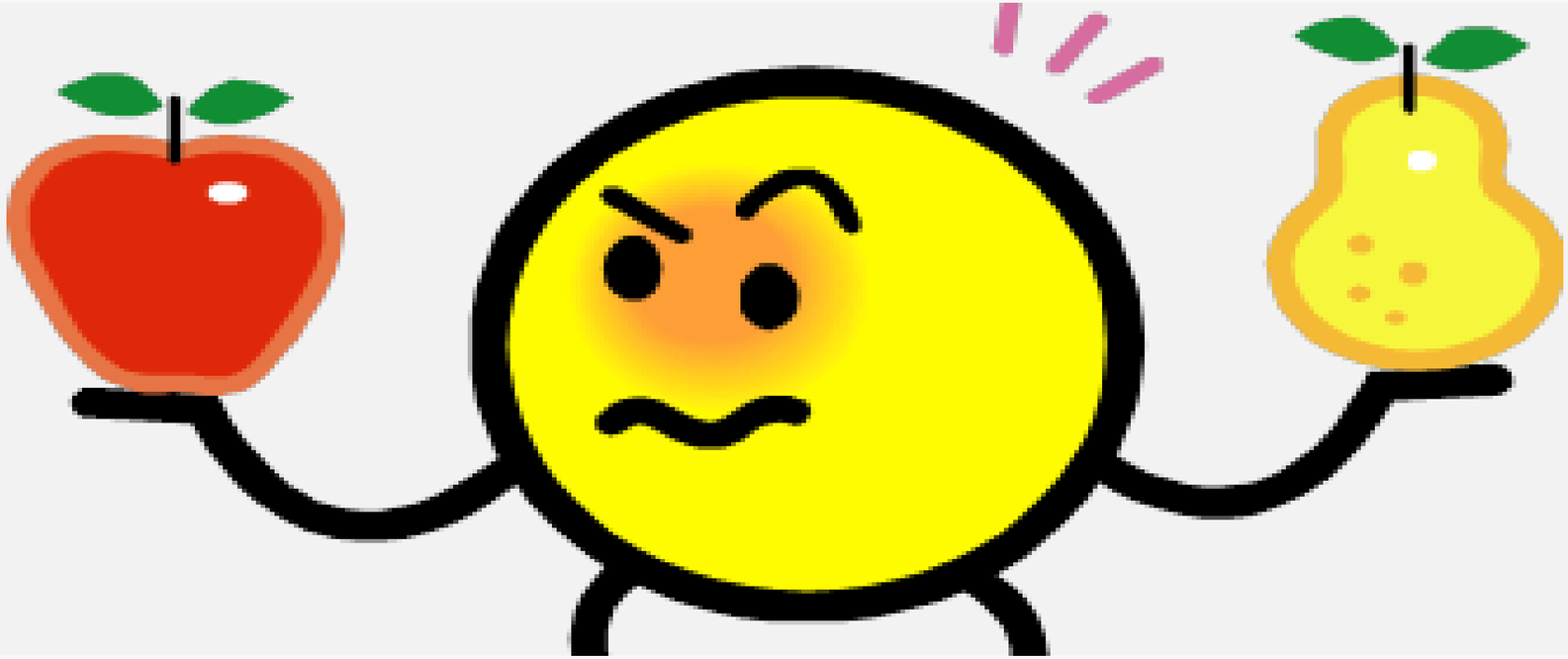
```
int a = 10;
int b = a;
```

Vimos também que operações aritméticas também são expressões.

```
double a = 2 * 2;
double a = 10 / 3;
a = a + 1;
```

Expressões relacionais

Expressões relacionais são aquelas que realizam uma comparação entre duas expressões e retornam:



- **False**, se o resultado é falso.
- **True**, se o resultado é verdadeiro.

Operadores relacionais em C++

Estes operadores serão utilizados nas expressões lógicas para comparações

Símbolo	Operação	Exemplo
>	Maior que	10 > 2
>=	Maior ou igual a	4 >= b
<	Menor quer	3 < a
<=	Menor ou igual a	b <= 2
==	Igual a	a == 5
!=	Diferente de	5 != 7

Expressões relacionais

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

int main() {
  int numero = 9;
  bool comparacao1 = 9 == numero;
  bool comparacao2 = 8 == numero;

  std::cout << "9 == 9: " << comparacao1 << std::endl;
  std::cout << "8 == 9: " << comparacao2 << std::endl;

  return 0;
}
```

expressão == expressão

Retorna verdadeiro quando os valores resultantes da avaliação das expressões forem iguais.

expressão != expressão

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

int main() {
  int numero = 9;
  bool comparacao1 = 9 != numero;
  bool comparacao2 = 8 != numero;

  std::cout << "9 != 9: " << comparacao1 << std::endl;
  std::cout << "8 != 9: " << comparacao2 << std::endl;

  return 0;
}
```

Retorna verdadeiro quando os valores resultantes da avaliação das expressões forem diferentes

Expressões relacionais

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

int main() {
  int numero = 9;
  bool comparacao1 = 9 > numero;
  bool comparacao2 = 8 > numero;

  std::cout << "9 > 9: " << comparacao1 << std::endl;
  std::cout << "8 > 9: " << comparacao2 << std::endl;

  return 0;
}
```

expressão > expressão

Retorna verdadeiro quando o valor da esquerda for maior que o valor da direita.

expressão < expressão

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

int main() {
  int numero = 9;
  bool comparacao1 = 9 < numero;
  bool comparacao2 = 8 < numero;

  std::cout << "9 < 9: " << comparacao1 << std::endl;
  std::cout << "8 < 9: " << comparacao2 << std::endl;

  return 0;
}
```

Retorna verdadeiro quando o valor da esquerda for menor que o valor da direita.

Expressões relacionais

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

int main() {
  int numero = 9;
  bool comparacao1 = 9 >= numero;
  bool comparacao2 = 8 >= numero;

  std::cout << "9 >= 9: " << comparacao1 << std::endl;
  std::cout << "8 >= 9: " << comparacao2 << std::endl;

  return 0;
}
```

expressão >= expressão

Retorna verdadeiro quando o valor da esquerda for maior ou igual que o valor da direita.

expressão <= expressão

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

int main() {
  int numero = 9;
  bool comparacao1 = 9 <= numero;
  bool comparacao2 = 8 <= numero;

  std::cout << "9 <= 9: " << comparacao1 << std::endl;
  std::cout << "8 <= 9: " << comparacao2 << std::endl;

  return 0;
}
```

Retorna verdadeiro quando o valor da esquerda for menor ou igual que o valor da direita.

2

Operações lógicas

Operadores lógicos

Expressões lógicas são aquelas que realizam uma operação lógica (ou, e, não, etc...)

Operador	Expressão
!	Negação (NÃO)
&&	Conjunção Lógica (E)
||	Disjunção Lógica (OU)

retornam True ou False, como as expressões relacionais.

Expressões lógicas

Expressão AND expressão

Op1	Op2	Op1 and Op2
V	V	V
V	F	F
F	V	F
F	F	F

Descrição

- Retorna verdadeiro quando ambas as expressões são verdadeiras

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

int main() {

  int numero1 = 100;
  int numero2 = 200;
  bool expressao = (numero1 == 100) && (numero2 == 200);

  std::cout << "Resultado da expressão: " << expressao << std::endl;

  return 0;
}
```

Qual o resultado da expressão lógica acima?

Expressões lógicas

Expressão OR expressão

Op1	Op2	Op1 or Op2
V	V	V
V	F	V
F	V	V
F	F	F

Descrição

Retorna verdadeiro quando **pelo menos uma** das expressões são verdadeiras

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6    int numero1 = 100;
7    int numero2 = 200;
8    bool expressao = (numero1 == 200) || (numero2 == 200);
9
10   std::cout << "Resultado da expressão: " << expressao << std::endl;
11
12   return 0;
13 }
```

Qual o resultado da expressão lógica acima?

Expressões lógicas

Op1	not Op1
V	F
F	V

Descrição

Retorna verdadeiro quando a expressão é falsa e vice-versa

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

int main() {

  int numero1 = 100;
  int numero2 = 200;
  bool expressao = !((numero1 == 200) || (numero2 == 200));

  std::cout << "Resultado da expressão: " << expressao << std::endl;

  return 0;
}
```

Qual o resultado da expressão lógica acima?

3

Comandos condicionais

Comandos condicionais

Um comando condicional é aquele que permite decidir se um determinado bloco de comandos deve ou não ser executado, a partir do resultado de uma expressão relacional ou lógica.

Blocos de comandos

É um conjunto de instruções agrupadas.

Os comandos agrupados do bloco devem estar indentados

dentro de um comando anterior seguido de dois pontos

A identação é feita em geral com 2 ou 4 espaços em branco

Comandos condicionais

O principal comando condicional é o if, cuja sintaxe é:

```
if (expressão relacional ou lógica){
        comandos executados se a expressão for verdadeira
}
```

Os comandos são executados somente se a expressão relacional/lógica for verdadeira.

Comandos condicionais: if

O programa determina se um valor é par.

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

int main() {

    int numero = 100;

    if((numero % 2) == 0) {
        std::cout << "O número é par" << std::endl;
    }

}
```

Comandos condicionais

Uma variação do comando if, é o if/else, cuja sintaxe é:

```
if (expressão relacional ou lógica){
        comandos executados se a expressão for verdadeira
} else {
        comandos executados se a expressão for falsa
}
```

Comandos condicionais: if/else

O programa determina se um valor é par ou ímpar.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int numero = 100;
7
8      if((numero % 2) == 0) {
9          std::cout << "O número é par" << std::endl;
10     } else {
11         std::cout << "O número é ímpar" << std::endl;
12     }
13
14 }
```

Estrutura de seleção múltipla

Similar a linguagem de programação python

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

int main() {

    if (valor == valor1) {
    //comando1
    } else if (valor == valor2) {
        //comando2
    } else if (valor == valor3) {
        //comando3
    } else {
        //...
    }

}
```

Comando condicionais

Note que o **if** é um comando, e como tal pode aparecer dentro do bloco de comandos de outro **if**.

Exemplo: Usando apenas operadores relacionais e aritméticos, vamos escrever um programa que lê um número e verifica em qual dos seguintes casos o número se enquadra:

- Par e menor que 100
- Par e maior ou igual a 100
- Ímpar e menor que 100
- Ímpar e maior ou igual a 100

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

int main() {

    int numero = 57;

    if((numero % 2) == 0) {

        if (numero < 100){
            std::cout << "O número é par e menor que 100" << std::endl;
        } else {
            std::cout << "O número é par e maior ou igual a 100" << std::endl;
        }

    } else {

       if (numero < 100){
            std::cout << "O número é ímpar e menor que 100" << std::endl;
        } else {
            std::cout << "O número é ímpar e maior ou igual a 100" << std::endl;
        }
    }

}
```

Estrutura de seleção múltipla

SWITCH - CASE

Conhecida por Switch-Case, em C++, o valor a ser testado deve ser um inteiro ou um caracter

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      switch (valor) {
7          case valor1 : //comandos + break;
8          case valor2 : //comandos + break;
9          case valor3 : //comandos + break;
10         ...
11         case valorN : //comandos + break;
12         default : //comandos + break;
13     }
14
15 }
```

Dúvidas???

Referências

1. DA COSTA, Anderson Fabiano F. **Fundamentos de C++**. Instituto Federal da Paraíba. 2022.

2. OLIVEIRA, Victor A. P. **Fundamentos de C++**. Instituto Federal da Paraíba. 2022.

3. HORSTMANN, C. **Conceitos de Computação com o Essencial de C++**, 3ª edição, Bookman, 2005.

4.

5.