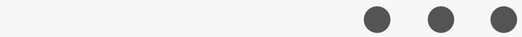
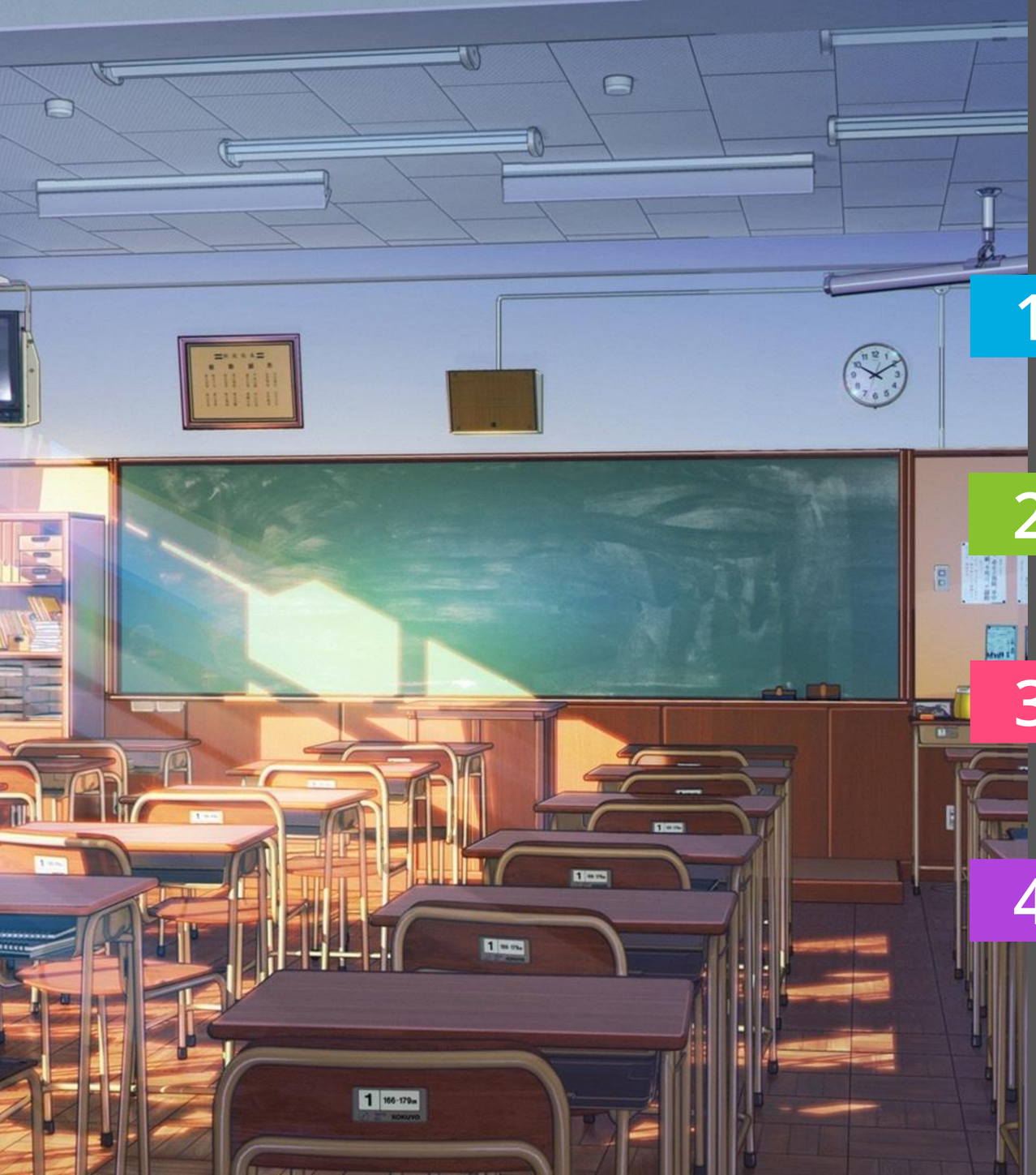
Estruturas de repetição

Aula 03 - Programação orientada à objetos



Professor Daniel Marques

INSTITUTO FEDERAL
Paraíba
Campus Campina Grande



Na aula de hoje

1. Problemas com repetição
2. Estruturas de repetição
3. Comando FOR
4. Comando WHILE e Do WHILE

INSTITUTO FEDERAL
Paraíba
Campus Campina Grande

1

Problemas com repetição

Problema

Antigamente, o terror das crianças na escola era uma atividade onde pedia-se para escrever os número do 0 até 100, ou de 100 a 200 e assim por diante

Era algo cansativo e repetitivo. Imagine como fazer um programa de computador para contar de 1 até 100.

```python
print("1")
print("2")
print("3")
print("4")
print("5")
print("6")
print("7")
print("8")
print("9")
print("10")
print("11")
print("12")
print("13")
print("14")
print("15")
print("16")
print("17")
print("18")
print("19")
print("20")
print("CANSEI")
```

Em python já cansa, imagina em C++

Solução do problema

Fazer uma estrutura que em poucas linhas de código, faça o trabalho de 100, 200 ou 500 linhas de código

Solução: Estrutura de repetição

Repetições

Imagine ter que testar milhões e milhões de possibilidades de uma máquina de criptografia

Allan Turing passou por isso na construção da máquina Bombe, que foi utilizada para derrotar a máquina Enigma utilizada pelos alemães na 2ª guerra mundial.

2

Estrutura de um comando de repetição

Estruturas de repetição

As estruturas de repetição são conhecidas como laços ou loops

Como são chamadas?

São usadas para executar, repetidamente, uma instrução, ou bloco de instrução enquanto determinada condição estiver sendo satisfeita

Para que servem?

Composição de uma estrutura de repetição

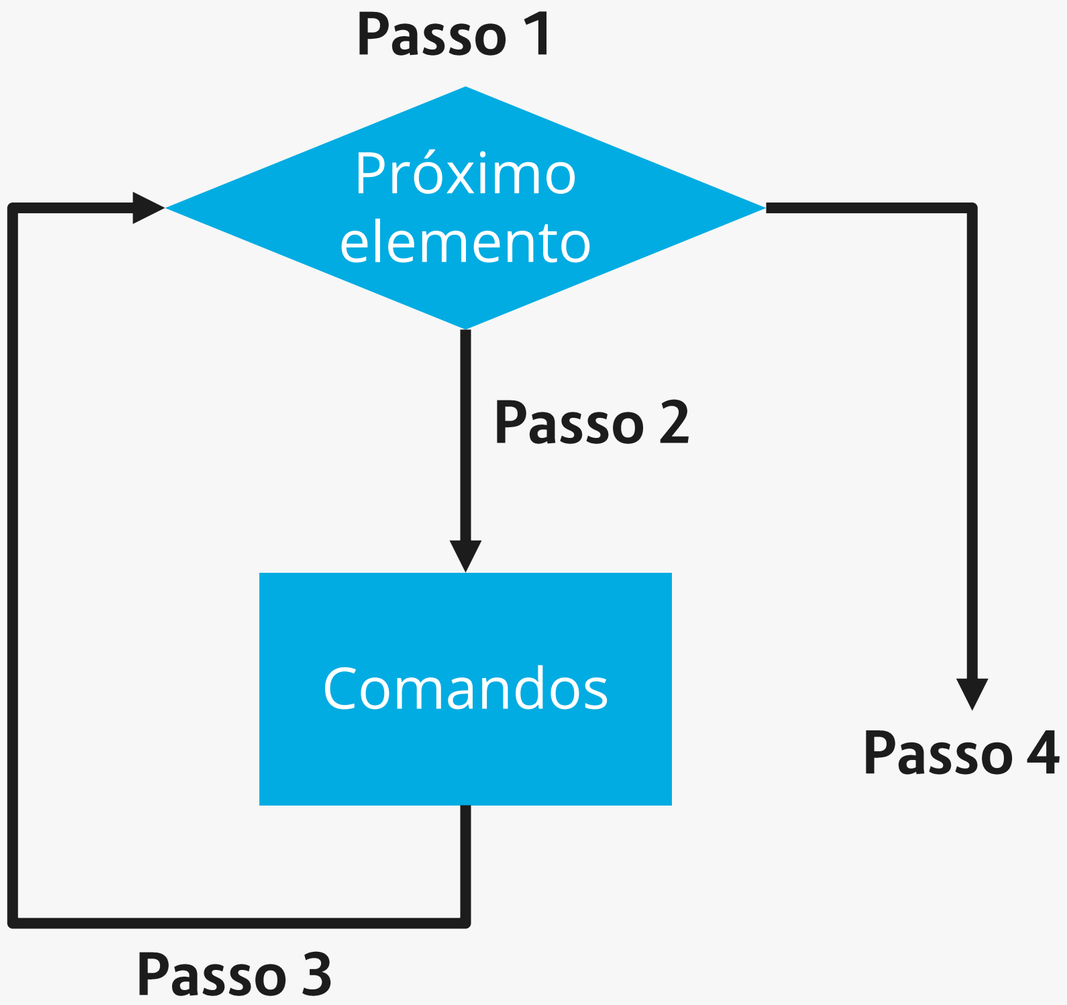
Inicialização — Todo o código que determina a condição inicial

Condição — Expressão booleana avaliada após cada leitura do corpo e que determina se uma nova leitura deve ser feita ou se a estrutura de repetição deve ser encerrada

Corpo ou bloco de código — Todas as instruções que são executadas repetidamente

Iteração — Incremento matemática da repetição

Estrutura de repetição



Verifica se percorreu toda a lista:

1
- Se não percorreu, atribui-se o próximo elemento da lista para a variável
- Se percorreu, vai para o Passo 4

2 Executa os comandos

3 Volta para o Passo 1

4 Sai do laço / loop

Comandos repetitivos

Até agora vimos como escrever programas capazes de executar comandos de forma linear, e, se necessário, tomar decisões com relação a executar ou não um bloco de comandos.

Entretanto, eventualmente faz-se necessário executar um bloco de comandos várias vezes para obter o resultado esperado.

3

Comando FOR

Comando FOR

É a estrutura de repetição mais utilizada quando conhecemos a quantidade **determinada** de repetições

Para cada elemento da lista, em ordem de ocorrência, é atribuído este elemento à variável e então executado o(s) comando(s)

Sintaxe do comando FOR

A sintaxe do comando FOR em C++ é similar a de C e a de Java

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

int main() {

    for (int i = 0; i < 3; i++) {
        std::cout << i << std::endl;
    }

}
```

```
for ( <inicialização> ; <condição> ; <iteração ou passo> ) {
        // bloco de código que será repetido
}
```

Incremento

Quando queremos incrementar, isto é, somar a iteração, usamos i++

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

int main() {

    for (int i = 0; i < 10; i = i + 2) {
        std::cout << i << std::endl;
    }

}
```

- i++ quer dizer que o próximo valor será i = i + 1;
- Você também pode incrementar valores diferentes como i = i + 2

Decremento

Também existe a possibilidade de decrementar, isto é, subtrair a iteração

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

int main() {

    for (int i = 10; i >= 0; i = i - 1) {
        std::cout << i << std::endl;
    }

}
```

FOR each

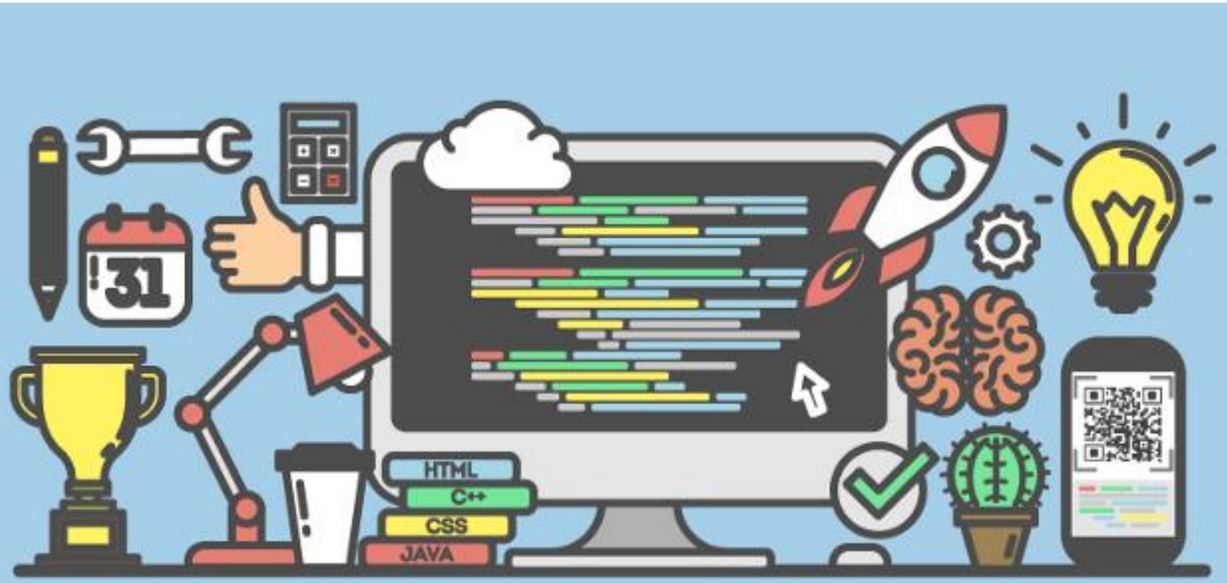
O For each é outra maneira de repetir alguns comandos, mas agora, em uma lista

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int vetor[10] = {1, 2, 3, 6, 9, 10, 123, 21};
7
8      for (int elemento : vetor) {
9          std::cout << elemento << std::endl;
10     }
11
12 }
```

```
for ( <variável> : <lista> ){
        <comandos>
}
```

Praticar

Faça um programa que imprima os números entre +10 e -10 e diga se o número é ou não é divisor de 3



4

Comando WHILE

Comando WHILE

Diferentemente do FOR, usamos WHILE quando a quantidade de repetições é indeterminada

Este comando sempre vai depender de uma condição indeterminada, é muito utilizado quando queremos utilizar alguma opção de escolha para o usuário

Sintaxe do comando WHILE

A sintaxe do comando WHILE em C++ é similar a de C e a de Java

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

int main() {

    int i = 0;

    while (i < 3) {
        i++;
        std::cout << i << std::endl;
    }

}
```

```
while ( expressão ) {

        \\ bloco de comandos

}
```

Comando WHILE

Note que o WHILE tem fora de sua estrutura a variável de incremento

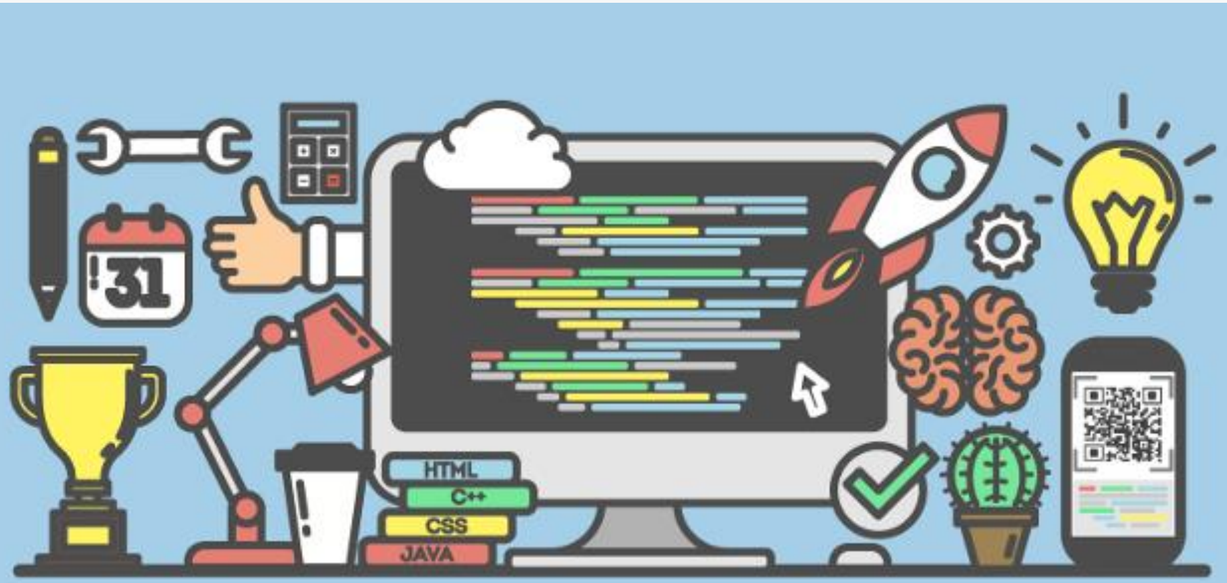
E dentro de sua estrutura, o incremento de i

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int i = 0;
7
8      while (i < 3) {
9          i++;
10         std::cout << i << std::endl;
11     }
12
13 }
```

Então, podemos concluir que com o WHILE, podemos fazer a mesma coisa que o FOR realiza. Mas não, o contrário

Praticar

Crie um programa que fique somando os números que o usuário digitar e só pare quando o usuário digitar o valor zero



5

Do WHILE

Sintaxe do comando Do While

A única diferença do WHILE para o DO WHILE, é que este último executa ao menos uma vez o bloco de comando e a condição é analisada após a primeira iteração

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6    int i = 0;
7
8      do {
9          i = i + 1;
10         std::cout << i << std::endl;
11
12   } while (i < 3);
13
14 }
```

```
do {
        //bloco de comandos
} while ( expressão ) ;
```

WHILE ou FOR?

Use um laço FOR, se você souber, antes de iniciar o laço, o número máximo de vezes que você precisará executar o corpo do laço.

Por exemplo, se você estiver percorrendo uma lista de elementos, você sabe que o número máximo de iterações do laço que você pode precisar é "todos os elementos da lista".

Use um laço WHILE se você precisa repetir alguma computação até que alguma condição seja atendida, e você não pode calcular antecipadamente quando isso acontecerá.

- **for : "iteração definida**
- **while : "iteração indefinida", não temos certeza de quantas iterações precisamos nem podemos estabelecer um limite superior"**

Dúvidas???

Referências

1. BORIN, Edson. **Algoritmos e Programação de Computadores: Variáveis, Objetos e Atribuição**. Unicamp, São Paulo. Disponível em: <https://www.ic.unicamp.br/en/~edson/disciplinas/mc102/2019-1s/ef/slides/MC102-Aula06.pdf>. Acesso em: 16 Mar. 2021.

2. FILGUEIRAS, Filipe. **JAVA – Estruturas de Repetição.** Disponível em: <https://tableless.com.br/java-estruturas-de-repeticao/>. Acesso em: 08 Mai 2019.

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

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

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

INSTITUTO FEDERAL
Paraíba
Campus Campina Grande