Laço de Repetição

Prof. Thiago Felski Pereira, MSc.
Adaptado: Elisangela Maschio de Miranda

Definições

- Laços de repetição, também conhecidos como laços de iteração ou simplesmente loops, são comandos que permitem iteração de código, ou seja, que comandos presentes no bloco sejam repetidos diversas vezes.

Definições

- Imagine que fosse necessário solicitar 5 valores ao usuário, somá-los todos e exibir o resultado da soma.
- Se fosse utilizado algoritmo sequencial seria necessário criar 5 variáveis, fazer a leitura das mesmas e somá-las.

```
public class Program {
    public static void Main(string[] args) {
        int num1, num2, num3, num4, num5, soma;
        Console.WriteLine("Informe 5 números: ");

        num1 = int.Parse(Console.ReadLine());
        num2 = int.Parse(Console.ReadLine());
        num3 = int.Parse(Console.ReadLine());
        num4 = int.Parse(Console.ReadLine());
        num5 = int.Parse(Console.ReadLine());

        soma = num1 + num2 + num3 + num4 + num5;

        Console.WriteLine("A soma dos valores é "+soma);
    }
}
```

Definições

- Observando o código com atenção, nota-se que existem instruções que se repetem: a leitura e a soma dos valores.
- Sempre que existir repetição de código é utilizado **Laço de Repetição**.
- Portanto, Laço de Repetição é uma estrutura que permite repetir um determinado bloco de instruções até que uma (ou mais) condição(ões) seja(m) satisfeita(s).

```
6  public class Program {
7      public static void Main(string[] args) {
8          int num, soma, quantidade;
9          soma=0;
10         quantidade=0;
11         while (quantidade<5) {
12             Console.WriteLine("Informe o número: ");
13             num = int.Parse(Console.ReadLine());
14             //acumula os valores em soma
15             soma = soma + num;
16             //aumenta a quantidade em 1
17             quantidade = quantidade + 1;
18         }
19         Console.WriteLine("A soma dos valores é "+soma);
20     }
21 }
```

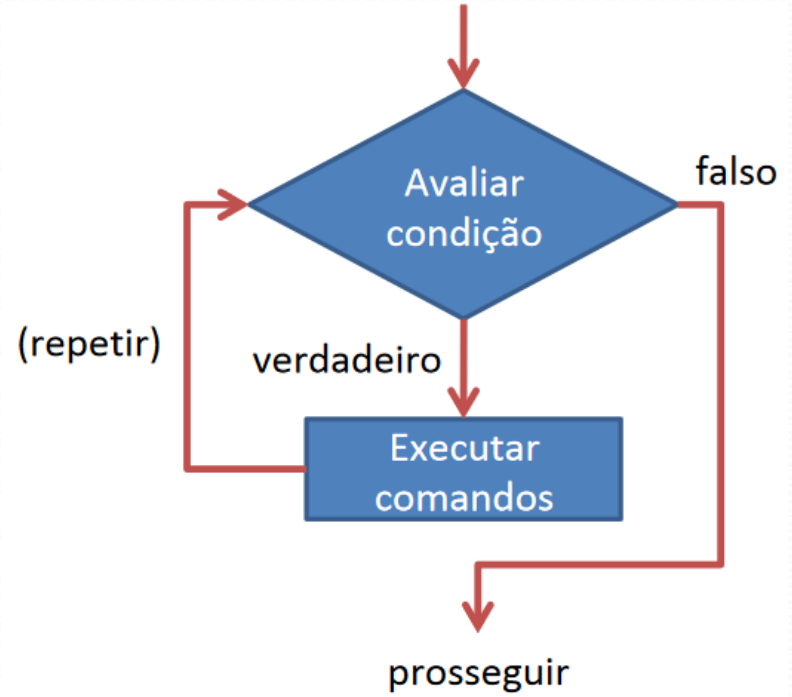
Definições

- Existem três tipos de laço de repetição
  - Laço de repetição com teste lógico no início.
  - Laço de repetição com teste lógico no final.
  - Laço de repetição com variável de controle.

while

- Neste laço a condição de término é definida desde o início.

- Ele testa a condição (ou condições) e caso seja(m) verdadeira(a) executa o bloco de instruções que seguem após a condição, entre as chaves. Caso seja(m) falsa(s) executam a primeira instrução fora do laço (depois do fechamento das aspas).



C#

```
while (condição/ões) {
      ... Bloco de Instruções ...
}
```


UNIVALI

while

- **EXEMPLO 1**: Elabore um algoritmo que exiba os valores de 1 a 20 na tela.

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int contador;
6          contador = 1;
7          while (contador <= 20) {
8              Console.Write(contador+" ");
9              //equivalente a contador = contador+1
10             contador++;
11         }
12     }
13 }
```

Enquanto esta condição for verdadeira o código entre as chaves será executado.

while

- Problemas comuns.
  - Dentre os problemas que podem ocorrer, um deles é o conhecido como looping infinito.

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int contador;
6          contador = 1;
7          while (contador <= 20) {
8              Console.Write(contador+" ");
9          }
10     }
11 }
```

Essa condição será sempre verdadeira, pois o valor de contador não é alterado

while

- Problemas comuns.
  - Dentre os problemas que podem ocorrer, um deles é o conhecido como looping infinito.

- Neste caso a condição sempre será verdadeira, fazendo com que o laço execute sem parada. A variável contador vale somente 1.
- Isso ocorreu pois não existe nenhuma instrução que faça com que a variável contador seja 20 em algum momento.
- Seria necessário uma instrução chamada **contador**.
- O contador nada mais é que uma variável a qual somamos um outro valor numérico e atribuímos para a própria variável (para modificar a própria variável e na próxima iteração trabalhar com o novo valor).

while

- Problemas comuns.
  - Dentre os problemas que podem ocorrer, um deles é o conhecido como looping infinito.

- Ao se trabalhar com um contador é necessário atribuir um valor inicial a mesma, antes de executar a condição do laço de repetição.
- Isso ocorre pois, ao declararmos uma variável a mesma possui o que é chamado lixo de memória, ou seja, possui um valor qualquer nela.
- Então, no caso era necessário imprimir o valor a partir de 1, então, foi atribuído 1 a variável. Se o início fosse a partir de 30, a variável seria inicializada de 30 (contador = 30).
- Se fosse necessário gerar os números de 2 a 2 (1, 3, 5, 7, 9, 11, 13, 15, 17, 19) o contador seria modificado para
- **contador = contador + 2;**

while

- Existem as reduções que podem ser utilizadas. Seguem abaixo a lista.

Incrementos	Explicação

variavel = variavel + 1;
variavel += 1;
variavel++; (Redução)
++variavel; (Redução)
	Todas as 3 fazem a mesma função: incrementam 1 a variável.

variavel = variavel + 5; variavel += 5; (Redução)	As duas incrementam 5 a variável.
variavel = variavel % 7; variavel %= 7; (Redução)	É efetuado o resto da divisão inteira com o valor contido em variável por 7, e atribuído a variávell
variavel = variavel + numero; variavel += numero; (Redução)	As duas incrementam o valor que tiver na variável número.

while

- As reduções podem ser efetuadas com qualquer operador aritmético.
- Existem dois contadores que apesar de parecerem iguais possuem diferenças. Segue a explicação:
  - `cont++;`
  - `++cont;`

- Ambos fazem o mesmo processo, que é incrementar a variável cont de 1. Segue abaixo a explicação da diferença:
  - `cont++;` utiliza a variável e incrementa o valor depois. Por exemplo, o código `cout<<cont++;` Imagine que foi atribuído 6 à variável cont. Neste caso imprimiria 6 e depois incrementaria mais um, e a variável cont conteria 7.
  - `++cont;` incrementa o valor e depois utiliza a variável. Por exemplo, o código `cout<<++cont;`. Imagine que foi atribuído 6 à variável cont. Neste caso incrementaria mais um, e a variável cont conteria 7. Depois imprimiria 7.

while

- Na tabela de incrementos houve um código diferenciado
  - `variavel = variavel + numero;`
  - Este caso seria chamado de acumulador ou somador, pois a variavel soma os valores.

- O somador ou acumulador pode ser utilizado com qualquer operador aritmético.

while

- **EXEMPLO 2**: Faça um programa que solicite o valor de 10 salários e informe, ao final, a soma de todos.

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int cont=0;
6          float salario, soma=0;
7
8          while (cont < 10) {
9              Console.WriteLine("Entre com o salário: ");
10             salario = float.Parse(Console.ReadLine());
11             soma = soma + salario;
12             cont++;
13         }
14         Console.WriteLine("Soma dos salários: "+soma);
15     }
16 }
```

A variável soma está acumulando os valores lidos na variável salario. Por este motivo foi inicializada com 0.

while

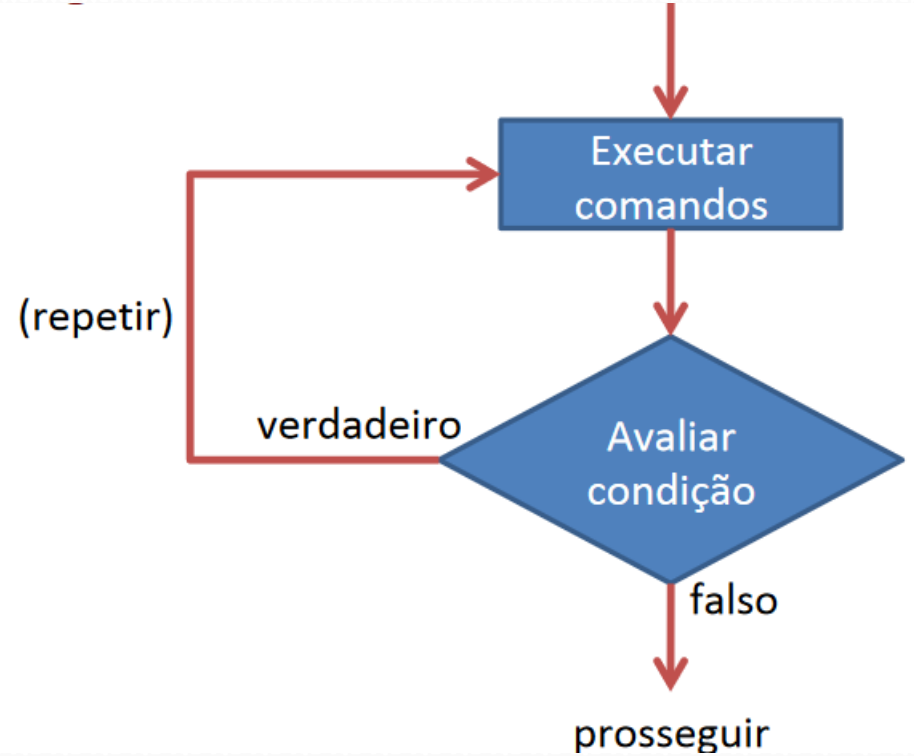
- **EXEMPLO 3**: Faça um programa que gere a tabuada de 1 a 10.

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int op1, op2, calculo;
6
7          op1=1;
8          while (op1 <= 10) {
9              op2=1;
10             while (op2 <= 10) {
11                 calculo = op1 * op2;
12                 Console.WriteLine(op1+" * "+op2+" = "+calculo);
13                 op2++;
14             }
15             op1++;
16         }
17     }
18 }
```

Quando se tem um laço dentro do outro, ele executa uma vez o laço mais externo (passa a valer 1) e depois todo o laço interno (varia de 1 a 10). Isto será feito até que o laço externo passe a valer 11. A toda vez que o laço externo rode uma vez, o interno roda 10 vezes (neste caso).

do~while

- A diferença entre o laço de repetição com teste lógico no final e o `while(){ … }` é que ele executa ao menos uma vez, pois o teste lógico é executado ao final.
- Todas as regras do `while` valem para o `do ~ while`.
- Uma das situações mais utilizadas é para validação de informação. Por exemplo, garantir que um valor digitado seja maior que 0.



C#

```
do {
      ... Bloco de Instruções ...
} while (condição/ões);
```


UNIVALI

do ~ while

- **EXEMPLO 4**: Faça um algoritmo que leia um valor e garanta que o mesmo seja positivo.

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int valor;
6
7          do {
8              Console.WriteLine("Informe um valor positivo: ");
9              valor = int.Parse(Console.ReadLine());
10             if (valor <= 0) {
11                 Console.WriteLine("Valor Inválido!");
12             }
13         } while (valor <= 0);
14         Console.WriteLine("O valor positivo registrado foi "+valor);
15     }
16 }
```

do ~ while

- **EXEMPLO 5**: Elabore um algoritmo que exiba os valores de 1 a 20 na tela.
- **EXEMPLO 6**: Faça um programa que gere a tabuada de 1 a 10.

do ~ while

- **EXEMPLO 5**: Elabore um algoritmo que exiba os valores de 1 a 20 na tela.
- **EXEMPLO 6**: Faça um programa que gere a tabuada de 1 a 10.

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int cont;
6          cont=1;
7          do  {
8              Console.Write(cont+" ");
9              cont++;
10         } while (cont <= 20);
11     }
12 }
```

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int op1, op2, calculo;
6
7          op1=1;
8          do  {
9              op2=1;
10             do {
11                 calculo = op1 * op2;
12                 Console.WriteLine(op1+" * "+op2+" = "+calculo);
13                 op2++;
14             } while (op2 <= 10);
15             op1++;
16         } while (op1 <= 10);
17     }
18 }
```

do ~ while

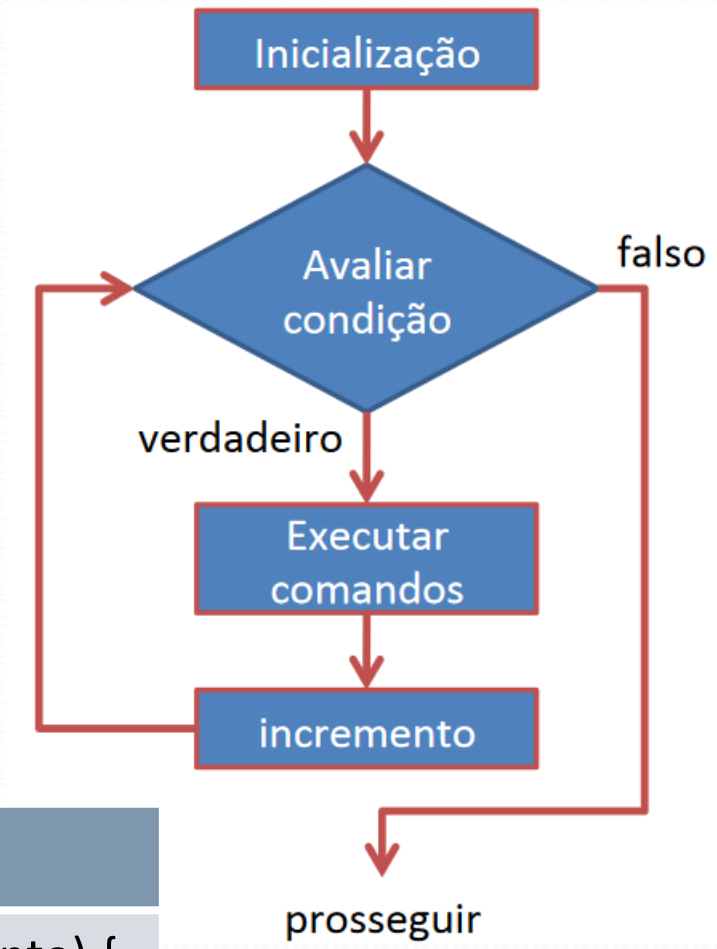
- Sempre existe confusão entre o `while` e o `do … while`. O principal é o que o `while` realiza o teste lógico no início, ou seja, pode não executar. Já o `do … while` sempre executará uma vez, pois o teste será somente ao final do laço.

- O `do … while` é utilizado quando se precisa executar o laço uma vez ao menos, ou quando não é preciso testar a primeira vez (em muitos casos que se pode usar o `while()` pode-se utilizar também do `do … while`.

- O laço será executado enquanto a condição for verdadeira. A partir do momento que for falsa ele executa a primeira instrução fora do laço.

for

- O laço for é um laço de repetição com variável de controle.
- O laço de repetição for funciona como os outros dois.
- Enquanto a condição é verdadeira o laço é executado e só para a execução quando a condição se torna falsa.



C#

```
for (inicialização da(s) variável(is); condição(ões); incremento/decremento) {
                                ... Bloco de Instruções ...
}
```


UNIVALI

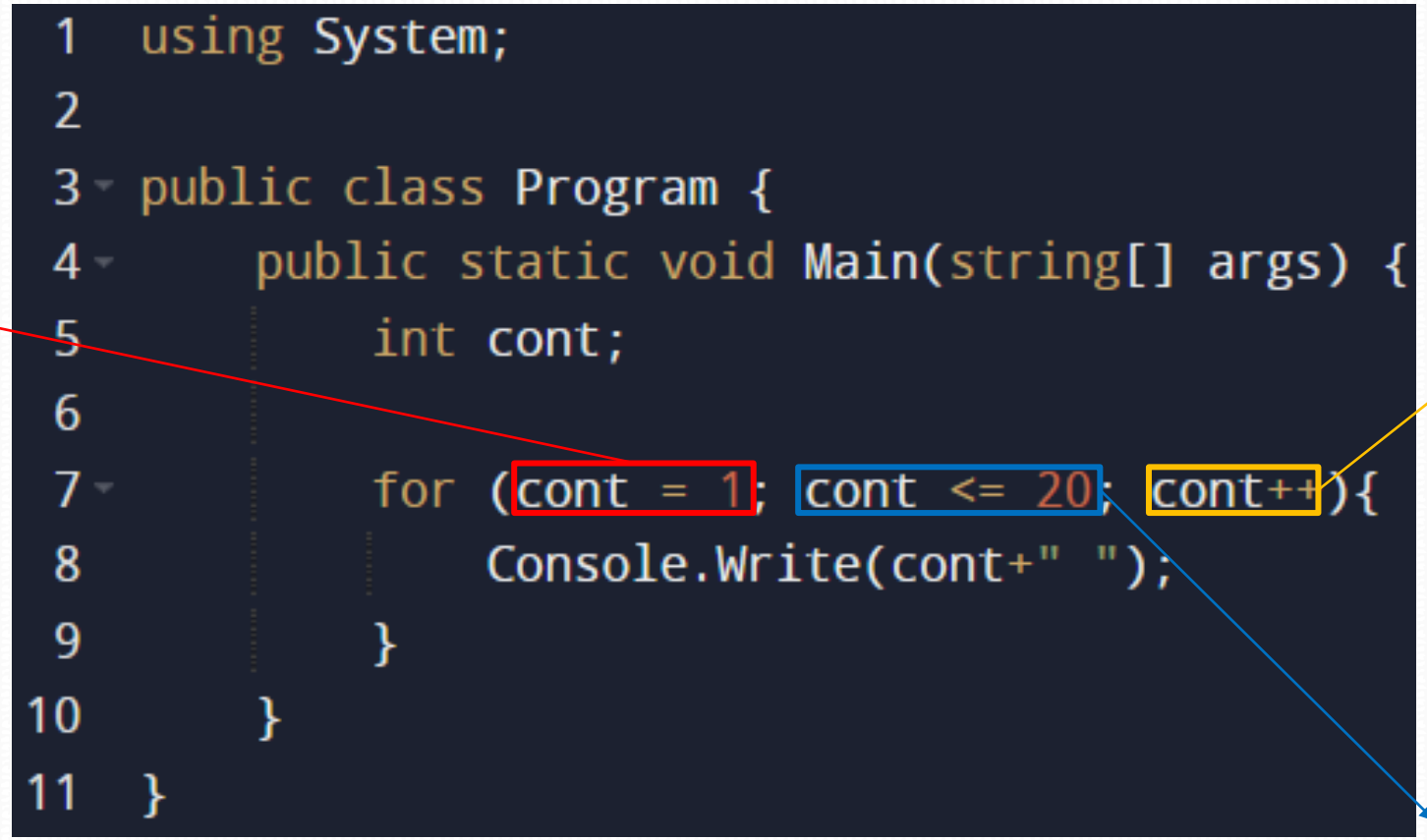
for

- O laço de repetição for é formado pelos itens abaixo:
  - Inicialização: atribui os valores de início das variáveis.
  - Condição: condição ou condições para que o laço execute.
  - Incremento/Decremento: incremento/decremento de variáveis.

- O que separa cada um dos itens é o ponto e vírgula (;).
- O laço for pode conter os três itens ou não.

for

- **EXEMPLO 7**: Elabore um algoritmo que exiba os valores de 1 a 20 na tela.



Inicializou a variável cont em 1.

Toda vez que o laço repete incrementa 1 a variável cont.

Enquanto a variável cont não chegar a 21, o laço continuará repetindo.

UNIVALI

for

- **EXEMPLO 8**: Elabore um algoritmo que exiba os valores de 10 a 100 na tela, pulando de 5 em 5.

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int cont;
6
7          for (cont = 10; cont <= 100; cont=cont+5){
8              Console.Write(cont+" ");
9          }
10     }
11 }
```

Inicializou a variável cont em 10.

Toda vez que o laço repete incrementa 5 a variável cont.

Enquanto a variável cont não chegar a 105, o laço continuará repetindo.

for

- **EXEMPLO 9**: Faça um programa que gere a tabuada de 1 a 10.

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int calculo;
6          for (int op1 = 1; op1 <= 10; op1++){
7              for (int op2 = 1; op2 <= 10; op2++){
8                  calculo = op1 * op2;
9                  Console.WriteLine(op1+" * "+op2+" = "+calculo);
10             }
11         }
12     }
13 }
```

- Note que op1 e op2 foram criados dentro do laço.

for

- Em relação aos itens que constituem o for, pode-se ter os 3, bem como algum deles pode ser excluído.
- Também pode-se ter mais de um item, que deve ser separado por vírgula (,).

- **EXEMPLO 10**:

```
1  using System;
2
3  public class Program {
4      public static void Main(string[] args) {
5          int n1, n2;
6          for (n1 = 1, n2 = 0; n2 <= 100; n1++, n2 += 5){
7              Console.WriteLine(n1+"\t"+n2);
8          }
9      }
10 }
```

Obrigado pela atenção

contato: felski@univali.br