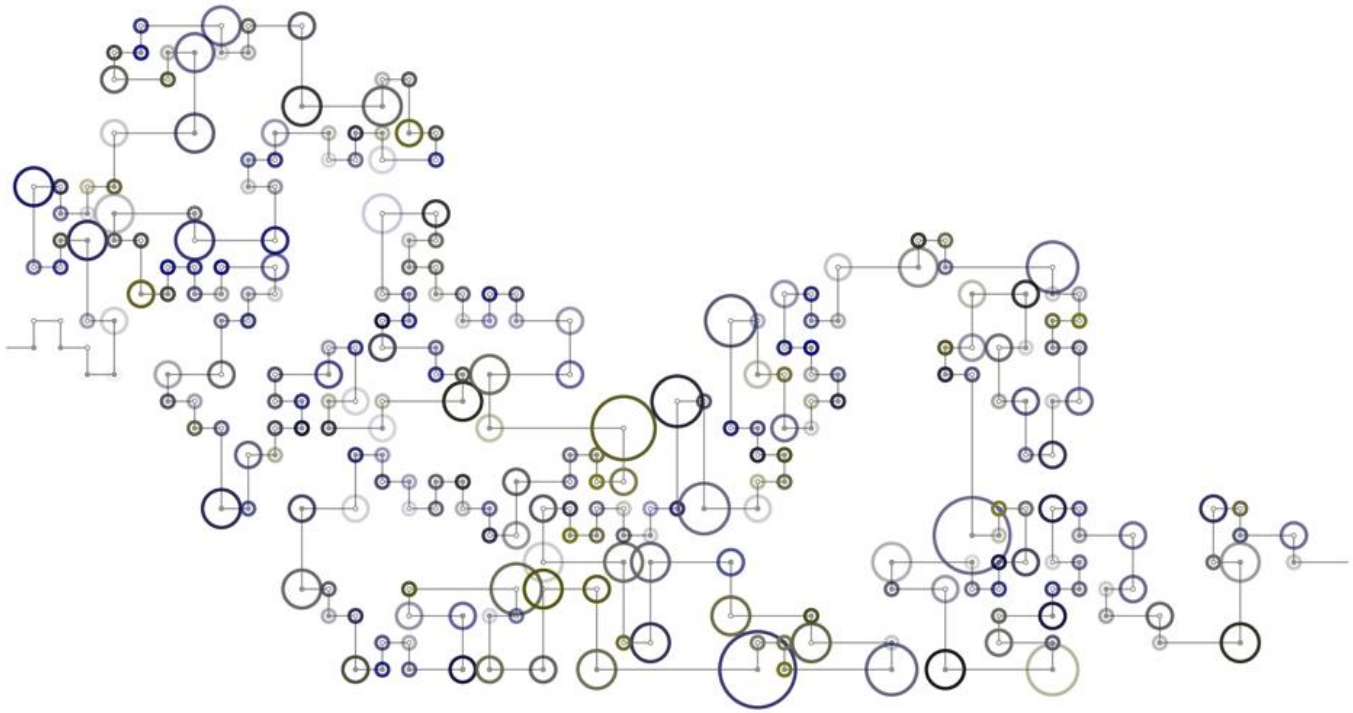
Aula 6
Laços de repetição



Prof. Me. Humberto Zanetti

Algoritmos

O que vamos ver?

Laços de repetição

- **`enquanto`**
- **`faça - enquanto`**
- **`para`**

Nesta aula, daremos maior ênfase na estrutura `para`

Laços de repetição

Em algumas situações precisamos que um ou mais comandos se repitam.

Um algoritmo é uma solução que deve ser automatizada e preparada para qualquer situação.

- **Uma verificação que se repete até estar correta;**
- **A repetição de ações até que alguma condição se satisfaça;**
- **A repetição acontece até que um evento ocorra, que a faça terminar.**

Laços de repetição

Laços de repetição são estruturas que delimitam um grupo de comandos que poderão repetir até uma condição torna-se falsa ou um números limitado de vezes.

Portanto, há 2 tipos:

- **Com um número determinado**: determinamos uma determinada quantidade de vezes que a repetição ocorre.
- **Orientados a uma "condição"**: enquanto a condição estiver Verdadeira, haverá repetição. Quando a condição se tornar Falsa, a repetição se encerrará.

Orientados a uma condição

Entre os orientados à uma condição, temos 2 maneiras de verificar essa condição:

- Verificação no início: a condição é declarada ao início do bloco e é verificada ANTES de um ciclo ocorrer, para definir se será executado ou não;

  estrutura `enquanto()`

- Verificação no final: a condição é verificada NO FINAL do bloco, definir se a PRÓXIMA repetição ocorrerá ou não.

  estrutura `faça - enquanto()`

Será tema da próxima aula!

Com um número determinado

Para definirmos um número definido de repetições usamos a estrutura de repetição `para()`.

Com ela, determinamos um quantidade finita de vezes que o bloco de comandos irá se repetir.

Vamos definir três termos:

- início da contagem;
- limite da contagem;
- incremento ou decremento da contagem.

Estrutura para

```
para ( início do contador ; limite da contagem ; iteração) {
        comandos
}
```

início do contador: determinamos qual será o início da contagem, ou seja qual será o valor inicial que a variável "contador" irá iniciar. Essa variável deverá ser do tipo inteiro; e não importa seu nome, desde que seja inteiro.
limite da contagem: condição que definimos qual será o limite da contagem (até qual valor o contador irá chegar)
iteração: o valor que a variável "contador" irá aumentar ou diminuir, durante a contagem

Exemplo

Faça um programa que repita a frase “Olá Mundo” 5 vezes na tela.

Quais são os comando que deverão repetir?

```
escreva(“Olá Mundo!\n”)
```

Qual é o valor do início da contagem?

começaremos em 1

Qual será a iteração da contagem?

de 1 em 1

Até quando o laço “irá contar”?

até chegar a 5

Exemplo

Faça um programa que imprima todos os números de 0 a 10.

```
inteiro contador
para( contador = 0; contador <=10; contador++){
      escreva(contador, "\n")
}
```

A variável contador se inicializa com 0, e vai incrementando de um em um (++), até o valor chegar a 10!

contador++ ➔ contador = contador + 1

contador- - ➔ contador = contador - 1

Saída:

```
0
1
2
3
4
5
6
7
8
9
10
```

Teste de mesa

Método de simulação, para verificarmos a valor que variáveis podem assumir durante a execução do algoritmo.

```
inteiro contador
para( contador = 0; contador <=10; contador++){
      escreva(contador, "\n")
}
```

ciclo	contador	contador++	repete?
1ª	0	1	sim
2ª	1	2	sim
3º	2	3	sim
4ª	3	4	sim
5ª	4	5	sim
6ª	5	6	sim
7ª	6	7	sim
8ª	7	8	sim
9ª	8	9	sim
10ª	9	10	sim
11ª	10	11	não

Exemplo

Faça um programa que imprima todos os números de 10 a 0 (ordem decrescente).

```
inteiro contador

para (contador = 10; contador >= 0; contador--){
     escreva(contador, "\n")
}
```

Saída:

10
9
8
7
6
5
4
3
2
1
0

Exercício

Faça uma programa que receba uma número inteiro e mostre a tabuada deste números. Exemplo:

```
Digite um número: 2


2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
...
2 x 10 = 20
```

Contadores e acumuladores

Como vimos até então, há variáveis as quais possuem funções específicas de contagem, no contexto de um laço de repetição.

Podemos também criar variáveis que podem incrementar (ou decrementar valores), não apenas de 1 em 1, mas de qualquer valor.

- Isso chamamos de **acumulação de valor**
- Ação muito comum em programação
  - Exemplo: fazer um somatória de valores

Exemplo

Faça um programa que receba os valores de 3 itens comprados pela usuário.

```
funcao inicio()
{
      real valor, total = 0.0
      inteiro cont

      para(cont = 1; cont <= 3; cont++){
            escreva("Digite o valor:")
            leia(valor)
            total = total + valor
      }

      escreva("Total da compra: ", total)
}
```

inicializar uma variável acumuladora é sempre uma boa prática

aqui a variável `total` recebe o valor que ela já contém adicionado a mais um valor, armazenando **um novo valor** (o acumulado)

Operadores para acumulação

Podemos usar uma forma abreviada para acumular valores, além do ++ e - - já mostrados.

operador	função
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y

```
real valor, total = 0.0
inteiro cont

para(cont = 1; cont <= 3; cont++){
      escreva("Digite o valor:")
      leia(valor)
      total += valor
}

escreva("Total da compra: ", total)
```

Exercício

1. Faça um programa que receba a nota de 4 alunos de uma sala de aula e ao final mostre qual foi a média dessa sala.
2. Faça um programa que receba 10 valores inteiros digitados pelo usuário, e some todos aqueles que forem pares. Mostre a soma.
3. Faça um programa que mostre todos os números pares entre 0 e algum número positivo digitado pelo usuário.

DÚVIDAS!?