



Java

Controlo de Fluxo II - Ciclos

METODOLOGIA

- › Interprete o documento calmamente e com atenção.
- › Acompanhe a execução do exercício no seu computador.
- › Não hesite em consultar o formador para o esclarecimento de qualquer questão.
- › Não prossiga para o ponto seguinte sem ter compreendido totalmente o ponto anterior.
- › Caso seja necessário, execute várias vezes o exercício até ter compreendido totalmente o processo.

CONTEÚDO PROGRAMÁTICO

1. [Introdução](#)

1. Introdução

A par dos mecanismos de controlo de fluxo, também os ciclos são uma ferramenta essencial na programação. Os ciclos permitem automatizar a execução de instruções para um elevado número de vezes.

Tal como nas estruturas de controlo, também nos ciclos existem vários formatos:

- `while` – executa enquanto uma condição se mantiver verdadeira.

- `do ... while` – executa enquanto uma condição se mantiver verdadeira. A diferença para o formato anterior é que neste caso, executa sempre pelo menos uma vez.
- `for` – ideal para percorrer um conjunto fixo de registos ou executar uma operação um número predefinido de vezes.

Assim como foi visto anteriormente quando alteramos valores em variáveis, também é possível de afetar o valor de uma variável com base nela própria. A isto chama-se de incrementar ou decrementar o valor de uma variável.

Imagine uma variável saldo do tipo inteiro:

```
int saldo = 500;
```

É-nos possível incrementar o valor desta variável, somando por exemplo 300 ao que já tem, seguindo a seguinte sintaxe:

```
saldo = saldo + 300;
```

Desta forma estamos a alterar a variável saldo com valor da mesma mais 300, fazendo assim uma soma com 300. Isto é obviamente válido para todas as outras operações aritméticas que conhecemos. Este tipo de lógica permite-nos modificar o valor de uma variável sem ter a mínima perceção de que valor tem este no momento.

Vamos fazer alguns exemplos relativo aos ciclos:

- › **Inicie** uma nova classe com o nome `Ciclos.java`

- › Introduza o seguinte código:

```
class Ciclos {  
    public static void main (String[] args) {  
  
        int contador = 1;  
  
        do {  
            System.out.println("Valor atual " + contador);  
            contador = contador + 1;  
        }  
        while(contador <= 10);  
    }  
}
```

- › Compile a classe e execute

- › Analise o resultado

- › Altere agora a inicialização da variável `contador` para o valor 11

- › Compile e execute a classe

- › Analise o resultado

Apesar de o contador à partida ser logo maior que 10, ainda é escrita uma linha no ecrã, isto porque no caso do ciclo `do..while` a condição do ciclo apenas é avaliada depois de ser executado o corpo do ciclo.

› **Altere** agora o **código** para:

```
class Ciclos {  
    public static void main (String[] args) {  
  
        int contador = 11;  
  
        while(contador <= 10) {  
            System.out.println("Valor atual " + contador);  
            contador = contador + 1;  
        }  
    }  
}
```

› **Compile** e **execute** a classe

› **Analise** o resultado

📄 Repare como desta vez nenhuma linha é escrita no ecrã.

› **Altere** agora a inicialização da variável `contador` para o valor 0

- › **Execute** a página e **analise** o resultado

A instrução `contador = contador + 1;` pode ser substituída pela instrução `contador++;`. Este segundo formato é uma alternativa ao primeiro para a execução de operações aritméticas de variáveis. A única diferença é a forma mais compacta de escrever, o que faz com que seja mais utilizada pelos programadores.

- › **Altere** a instrução

```
contador = contador + 1;
```

Para

```
contador++;
```

- › **Compile** e **execute** a classe

- › **Analise** o resultado

- › **Altere** agora a instrução

```
contador++;
```

Para

```
contador = contador + 2;
```

- › **Execute** a página e **analise** o resultado

O último exemplo de ciclo que vamos abordar é o `for`. Este é sem dúvida o ciclo mais adequado para percorrer arrays. Isto deve-se ao facto de este ciclo já ter incorporado o bloco de iniciação e respetivos incrementos.

O ciclo for tem a seguinte sintaxe:

```
for (expressão de inicio; condição ; incremento/decremento){  
    bloco de instruções;  
}
```

- › **Crie** uma nova classe chamada `CiclosFor.java`

- › **Insira** o seguinte **código**:

```
public class CiclosFor {  
    public static void main (String[] args) {  
        int contador;  
  
        for (contador = 1; contador <= 10; contador++){  
            System.out.println("Valor atual " + contador);  
        }  
    }  
}
```

```
}  
}  
}
```

› **Compile e execute** a classe

› **Analise** o resultado

Analisemos agora as respectivas características do ciclo for que realizámos:

```
for (contador = 1; contador <= 10; contador++){
```

- A parte `contador = 1` refere à instrução de início do ciclo e corresponde à inicialização da variável. Podiam ter sido inicializadas várias variáveis neste bloco.
- `contador <= 10` é a parte mais semelhante ao `while`, e define qual a condição sobre a qual o ciclo continuará a ser executado. Neste caso em concreto, será executado enquanto a variável contador for menor que 10.
- `contador++` refere o bloco de incrementos do for, ou seja, de que forma as variáveis do ciclo aumentam ou diminuem. No caso apresentado, a variável contador aumenta de um em um a cada iteração do ciclo.

📄 A declaração da variável foi feita antes do ciclo for mas poderia ter sido feita no mesmo bloco de inicialização de variáveis.

› **Altere o código** da classe para o seguinte código:

```
public class CiclosFor {  
    public static void main (String[] args) {
```

```
    for (int contador = 1; contador <= 10; contador++){  
        System.out.println("Valor atual " + contador);  
    }  
}
```

- › **Confirme** que o resultado da execução do programa é precisamente o mesmo

De forma a rever conceitos abordados anteriormente, vamos alterar a classe `CiclosFor` para que peça ao utilizador o número de vezes que vai executar.

- › **Reescreva** o código da classe de acordo com o código abaixo:

```
import java.util.Scanner;  
  
public class CiclosFor {  
    public static void main (String[] args) {  
  
        Scanner teclado = new Scanner(System.in);  
  
        System.out.println("Insira o numero de vezes que quer executar");  
        int maximo = teclado.nextInt();  
  
        for (int contador = 0; contador < maximo; contador++){  
            System.out.println("Valor atual " + contador);  
        }  
    }  
}
```


› Compile e teste a classe

📄 Repare como a condição do ciclo for se alterou de forma a contemplar o valor que o utilizador introduziu como máximo.

📄 Apesar de o último número a ser escrito no ecrã ser inferior ao que inseriu no programa, este executou as vezes que especificou, uma vez que o zero também conta.

Agora segue-se uma alteração à mesma classe com o intuito de realizar a soma dos números introduzidos pelo utilizador. A quantidade de números a somar será definida pelo mesmo.

› Volte a **alterar** o **código** da classe para o seguinte:

```
import java.util.Scanner;

public class CiclosFor {
    public static void main (String[] args) {

        Scanner teclado = new Scanner(System.in);
        System.out.println("Quantos numeros deseja somar?");
        int quantidade_numeros = teclado.nextInt();
        int soma = 0;

        for (int contador = 0; contador < quantidade_numeros; contador++){
            System.out.println("Insira o " + (contador + 1) + "º numero para somar");
            int numero_lido = teclado.nextInt();
            soma = soma + numero_lido;
        }

        System.out.println("A soma de todos os numeros que introduziu é " + soma);
    }
}
```

› **Compile, execute** a classe e **faça testes** com diversos valores

📄 Em caso de dúvida, não hesite em solicitar a ajuda do formador.

📄 Repare que para cada passagem do ciclo é solicitado um valor ao utilizador e é esse mesmo valor que é adicionado à variável soma, compondo assim a soma total.

📄 Na generalidade dos casos os ciclos for são bastante úteis para manusear arrays. Com estes podemos efetuar todo o tipo de operações sobre arrays independentemente do seu tamanho.