Formabase
Aprenda passo a passo

Java
Ciclos com arrays

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. Ciclos com Arrays
2. Quebras de Ciclos
   2.1. Break
   2.2. Continue

1. Ciclos com Arrays

Nos módulos anteriores concluímos que os ciclos são bastante úteis para repetir uma ação inúmeras vezes. Isto é especialmente vantajoso quando se trabalha com arrays. Desta forma consegue-se fazer operações simples sobre arrays bastante grandes, como por exemplo, incrementar todos os valores existentes num array de dimensão 300.

› **Crie** a classe `CiclosArrays.Java`

› **Insira** agora o **código**:

```java
public class CiclosArrays {
    public static void main(String[] args) {
        int[] arraynumeros = new int[300];

        //preencher array
        for(int contador = 0; contador < 300; contador++){
            arraynumeros[contador] = 1;
        }

        //mostrar array
        for(int contador = 0; contador < 300; contador++){
            System.out.println(arraynumeros[contador]);
        }
    }
}
```

› **Compile** e **teste** a classe

O código acima exemplifica como através de poucas linhas de código se cria e preenche um array de 300 posições.

Para se poder aceder a cada posição no ciclo utiliza-se como índice uma variável que é incrementada no ciclo:

```
arraynumeros[contador] = 1;
```

Em cada uma das trezentas vezes que o código dentro do ciclo é executado, é preenchida uma posição diferente no array. O exemplo acima mencionado preenche todo o array com o número 1.

Pode até ser utilizada a própria variável que incrementa no ciclo como valor para ser guardado no array, gerando assim números diferentes para cada posição do array.

- **Altere** a instrução que se encontra dentro do **primeiro ciclo** `for` para a seguinte:

```
arraynumeros[contador] = contador;
```

- **Visualize** o efeito desta alteração

Agora todas as posições do array contêm um número diferente. Utilizando a mesma técnica podemos aplicar uma operação a todos os elementos do array, como por exemplo: transformá-los no dobro.

- **Insira** o seguinte código **antes do final do** `main`:

```
//transformar o array no seu dobro
for(int contador = 0;contador < 300;contador++){
    arraynumeros[contador] = arraynumeros[contador]*2;
}

System.out.println("O array no seu dobro:");

//mostrar array
for(int contador = 0;contador < 300;contador++){
```

```
    System.out.println(arraynumeros[contador]);
}
```

- **Confirme** o resultado desta alteração

Passemos a um exemplo diferente sobre a utilização de ciclos com arrays. O programa que se segue inverte a ordem dos números escritos pelo utilizador.

- **Crie** a classe `CiclosArraysInversao.Java` e **insira** o respetivo código:

```
import java.util.Scanner;

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

        Scanner teclado = new Scanner(System.in);

        System.out.println("Quantos numeros quer inserir ?");
        int quantidadenumeros = teclado.nextInt();

        int[] arraynumeros = new int[quantidadenumeros];

        for(int contador = 0; contador < quantidadenumeros; contador++){
            System.out.println("Insira o " + (contador+1) + " numero");
            arraynumeros[contador] = teclado.nextInt();
            //leitura do numero introduzido directamente para o array
            //na posicao contador
        }

        System.out.println("Os numeros de forma inversa:");

        for(int contador = quantidadenumeros - 1; contador >= 0; contador--){
```

```
            System.out.println(arraynumeros[contador]);
        }
    }
}
```

- **Compile** a classe e **teste** o seu funcionamento

- **Confirme** que os números aparecem na consola de forma inversa

```
arraynumeros[contador] = teclado.nextInt();
```

Nesta linha de código é feita a leitura de cada valor introduzido pelo utilizador para a posição correta no array (posição de índice contador).

```
for(int contador = quantidadenumeros - 1; contador >= 0; contador--){
```

A instrução acima faz o ciclo iniciar no valor mais alto do array, e terminar no primeiro (índice 0). Desta forma, o código percorre o array de forma inversa, escrevendo para o ecrã cada posição.

O início deste último ciclo foi definido para o valor `quantidadenumeros - 1` porque, num array de tamanho `N`, o último índice válido é `N-1`. Por exemplo: num array de **10** posições o último índice válido é o **9**, pois a gama de índices vai de **0** a **9**.

> A última posição indexável de um array será a correspondente ao seu tamanho máximo menos um.

2. Quebras de Ciclos

Como vimos anteriormente, as condições definem o ponto de paragem de um ciclo. No entanto, existe maneira de controlar a sua execução e paragem dentro do próprio ciclo. Para isso são usadas as palavras `break` e `continue`.

2.1. Break

Comecemos pela palavra break, que permite sair diretamente do ciclo em que está inserido.

- **Crie** a classe `QuebraCiclos.Java` e **insira** o seguinte código:

```
import java.util.Scanner;

public class QuebraCiclos {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);

        int contador = 0;

        while(contador < 10){
            System.out.println("Insira um numero");
            int numero = teclado.nextInt();

            if(numero > 100){
                break;
            }

            contador++;
        }

        System.out.println(“Fim do programa”);
```

```
    }
}
```

- **Compile** e **teste** a classe com valores acima e abaixo de 100

Através da condição que definimos do ciclo:

```
if(numero > 100){
    break;
}
```

O programa executa o código break quando um número está acima de 100, fazendo com que o ciclo pare e o programa continue a executar o código seguinte.

2.2. Continue

A instrução `continue` faz com que o ciclo passe imediatamente para a sua próxima execução.

- **Adicione** antes do final do `main` o seguinte código:

```
System.out.println("Instrução continue:");
contador = 0;

while(contador < 10){

    System.out.println("Insira um numero");
    int numero = teclado.nextInt();
```

```
    if(numero > 100){
        continue;
    }

    contador++;
    System.out.println("O numero inserido foi " + numero);
}
```

- **Teste** esta alteração também com valores abaixo e acima de 100

> Observe que quando são introduzidos valores acima de 100 não é mostrada a mensagem no ecrã do número que inseriu.

Quando o número é maior que 100 é executado o `continue` e o Java passa por cima das instruções que restam até ao fim daquela iteração do ciclo seguindo diretamente para a próxima iteração.