Prática 5
(Estruturas de Decisão – WHILE / DO WHILE / FOR + Estruturas Homogêneas)

Data de Entrega: 30 de abril de 2023

Leia antes de começar:

1. A prática deve ser feita individualmente, pode conversar com o colega, mas cada um faz o seu.

2. A prática deve ser postada no Ulife. O arquivo final deve ter todos os arquivos de código-fonte dos exercícios compactados em formato **ZIP** com os nomes do(a) aluno(a) no seguinte formato.

   Exemplo: **Pratica5-JoaoDeDeus.zip**

3. Trabalhos idênticos serão considerados cópias, logo, suas avaliações serão anuladas.

4. Todos os arquivos devem ter um comentário de cabeçalho contendo:

```
/* Nome do Aluno:  Fulano de Tal
 * RA:
 * Nome do Programa:
 * Data:
 */
```

Estruturas Homogêneas: Vetores e Matrizes

Para declarar uma variável qualquer temos a seguinte estrutura: coloca-se o tipo da variável seguida do nome da variável e um ponto-e-vírgula no final:

```
int variavel;
```

Uma estrutura de dados é dita homogênea quando ela agrupa e comporta dentro dela variáveis de um mesmo tipo.

Os **vetores são estruturas homogêneas unidimensionais**, isso significa que podemos declarar uma única variável e solicitar que ela armazene, na memória, espaço para uma quantidade maior de valores.

Para declarar um vetor precisamos de duas figuras adicionais: o colchete ao lado do tipo e o predicado "new tipo[quantidade]" que solicita a reserva de um espaço na memória para uma quantidade finita de posições:

```
int[] v10 = new int[10];
int[] v20 = new int[20];
```

Se desejamos acessar uma posição dentro dessa estrutura basta indicar um índice:

```
v10[4] = 10;
```

Esse índice indica a quantidade de saltos necessários para se atingir a posição desejada. Por isso, o índice começa de 0.

Já as matrizes **são estruturas homogêneas multidimensionais**. Isso significa que ela pode possuir mais de um índice para indicar a posição. Ao se declarar uma matriz precisamos:

Java: acrescentar um colchete para cada dimensão adicional

```
int[][] matriz = new int[3][3];
```

C#: acrescentar uma virgula para cada dimensão adicional

```
string[,] matriz=new string[2,3];
```

Para acessar as posições da matriz precisamos:

Java: acrescentar um colchete e informar seu índice

```
matriz[2][2] = 91;
```

C#: acrescentar uma virgula e informar seu índice

```
matriz[0,0] = "Longitud de la dimension ";
```

LIDANDO COM REPETIÇÕES

Muitas vezes precisamos repetir a mesma operação várias vezes dentro do programa. Para isso precisamos das estruturas de programação que criam "laços" de repetição.

Temos, à nossa disposição, 3 opções de laços de repetição:

for
while
do / while

Em todos estes laços de repetição precisamos utilizar uma variável para controlar quantas vezes repetiremos a mesma operação. A **variável de controle** determina os 3 parâmetros do laço:

1) A condição inicial
2) A condição de parada
3) O passo de iteração

Se quisermos, por exemplo, imprimir todos os números entre 1 e 100 podemos dizer que repetiremos a operação de escrita na tela 100 vezes começando de 1 indo até 100 e variando de 1 em 1.

```
int i;
for (i = 1; i <= 100; i++) {
     //print(i);
}
```

Usando o **for**, os 3 parâmetros são colocados na mesma linha: i começa em 1 **(i=1)** repete enquanto for menor ou igual a 100 **(i<=100)** e varia de 1 em 1 **(i++)**

Já no while e no do / while os três parâmetros são descritos em linhas diferentes.

A diferença entre ambos é que o do/while realiza as operações pelo menos uma vez mesmo que a condição seja **FALSA**.

```
int i = 1;
while (i <= 100) {
     //print(i);
     i++;
}
```

```
int i = 1;
do {
     //print(i);
     i++;
} while (i <= 100);
```

As vezes precisamos repetir uma operação de forma indefinida (sem saber quantas vezes) como por exemplo:

> *"Faça um programa que leia número que o usuário digita até que o usuário digite 0"*

Nesse caso a variável de controle não assume um valor sequencial e, sendo assim, não é necessário controlar o passo. Nesse caso, a variável de controle é alterada recebendo o que foi digitado pelo usuário.

```
int i = 1;
while (i != 0) {
      //read(i);
}
```

É possível também, com este tipo de estrutura, criar laços dentro de laços. Como por exemplo:

> *"Faça um programa que imprima a tabuada de todos os números de 1 a 10"*

```
int i, j;
for (i = 1; i <= 100; i++) {
   for (j = 1; j <= 100; j++) {
      //print(i * j);

   }
}
```

A relação entre loops, vetores e matrizes

É muito comum que loops sejam utilizados como forma de preencher e acessar vetores e matrizes. Por terem suas posições preenchidas

O cuidado necessário e para que, em Java, ou outras linguagens, a primeira posição do vetor inicia em 0. Logo, qualquer loop que percorra um vetor (visite posição a posição do inicio ao fim) começa de 0 e vai até o tamanho do vetor – 1;

"Faça um programa que crie um vetor de 10 posições e preencha cada célula com o valor correspondente a sua posição no vetor"

```
int[] vetor = new int[10];
int i;

for (i = 1; i < 10; i++) {
     vetor[i] = i;
}
```

O mesmo princípio vale para matrizes.

"Faça um programa que preencha uma matriz com os resultados da tabuada de todos os números de 1 a 10"

```
int[][] matriz = new int[10][10];
int i, j;

for (i = 0; i < 10; i++) {
   for (j = 0; j < 10; j++) {
      matriz[i][j] = (i+1) * (j+1)
   }
}
```

Problemas

1. Faça um programa que imprima todos os números pares de 1 a 100
2. Faça um programa que imprima todos os números pares de 1 a 100 na ordem inversa
3. Faça um programa que some os números de 1 a 100 e imprima somente o valor total da soma
4. Faça um programa que solicite ao usuário que digite um número até que ele digite um número menor que 0 **(utilize while)**
5. Faça um programa que determine o fatorial de um número. Para este problema, tem-se como entrada o valor do número do qual se deseja calcular o fatorial. O fatorial de 0 é igual a 1. O fatorial de um número N (N!) é definido conforme a seguir **(utilize for)**:
   i. N! = 1 * 2 * 3 * 4 * ... * (N-1) * N
6. Um determinado gás duplica seu volume a cada segundo. Dada um volume inicial, em centímetros cúbicos, digitado pelo usuário faça um programa que determine o tempo necessário para que esse volume se torne maior que 1000 centímetros cúbicos. **(utilize while)**
7. Escreva um programa que, dada a carga máxima de um elevador e a quantidade máxima de pessoas digitadas pelo usuário, leia o peso de cada pessoa, também digitada pelo usuário, que entra no elevador até que a carga máxima seja atingida ou o número máximo de pessoas seja atingido **(utilize do / while)**.
8. Faça um programa que preencha com zeros todas as posições de um vetor de tamanho 50
9. Faça um programa que preencha com zeros todas as posições de uma matriz com 10 linha e 10 colunas