Code kata

Vettori, liste, matrici

Indice generale

1 Generare una sequenza di interi progressivi

Problema

Generare una sequenza di numeri interi progressivi che cadano all'interno di un intervallo. Restituire la sequenza in un vettore. Se l'estremo sinistro e destro dell'intervallo sono uguali deve essere restituito un vettore vuoto (di lunghezza zero).

Esempio:

```
int da = 5;  // estremo sinistro intervallo
int a = 10;  // estremo destro intervallo: a >= da
//-> 5,6,7,8,9,10
```

Variazioni

Fornire delle versioni con ciclo `for`, `while`, `do while`.

2 Modificare gli elementi di un vettore

Problema

Dato un vettore di `double`, aumentare tutti gli elementi del 10%.

Esempio:

```
double[] dati = { 1000, 1200, 1500 };
//-> 1100, 1320, 1650
```

Variazioni

Generalizzare il procedimento per l'aumento di un valore percentuale qualsiasi.

```
double[] dati = { 1000, 1200, 1500 };
double perc = 20;  // 20%
//-> 1200, 1440, 1800
```

Fornire delle versioni con ciclo `for`, `while`.

Non modificare il vettore originale, ma restituire un nuovo vettore contenente i valori originali modificati.

3 Ricerca di un elemento in un vettore

Problema

Dato un vettore di interi e dato un numero, restituire la posizione del numero nel vettore. Restituire -1 se il numero non è presente nel vettore.

Esempio:

```
int[] dati = { 10, 30, 45, -30 };
int numero = 30;
// -> posizione: 1

int numero2 = 75;
// -> posizione: -1
```

Variazioni

Fornire versioni del procedimento con il ciclo `while`, `for` e `foreach`.

Restituire `true` se esiste, `false` se non esiste.

```
int[] dati = { 10, 30, 45, -30 };
int numero = 30;   // -> true

int numero2 = 75; // -> false
```

4 Invertire la posizione degli elementi

Problema

Dato un vettore invertire la posizione dei suoi elementi. (Modificare il vettore)

Esempio:

```
int[] dati = { 10, 30, 45, -30 };
// -> -30, 45, 30, 10
```

Variazioni

Fornire versioni del procedimento con i cicli `while` e `for`.

Non modificare il vettore ma restituire un nuovo vettore contenente gli stessi elementi dell'originale ma in posizione invertita.

5 Generare una sequenza di numeri interi casuali

Problema

Generare una sequenza di interi casuali; ogni numero casuale deve cadere in un intervallo specificato. La sequenza deve essere restituita in un vettore. Se la lunghezza della sequenza è zero deve essere restituito un vettore vuoto.

Esempio:

```
int da = 1;  // estremo sinistro intervallo numero casuale
int a = 20;  // estremo destro intervallo numero casuale: a > da
int n = 10;  // lunghezza sequenza  n >= 0
//-> 5, 1, 18, 12, 10, 5, 8, 19, 12, 8
```

Variazioni

Fornire versioni con ciclo `for`, `while` e `do while`.

Requisiti (necessari / utili) per la soluzione

Generare un numero casuale; oggetto `Random`, metodo `Next()`:

```
Random rnd = new Random();
int casuale = rnd.Next(1, 10);  //-> casuale 1 - 9 (compresi)
```

6 Generare una sequenza di interi casuali non ripetuti

Problema

Generare una sequenza di interi casuali; ogni numero casuale deve cadere in un intervallo specificato. La sequenza non deve contenere elementi ripetuti. Se la lunghezza della sequenza è zero deve essere restituito un vettore vuoto.

Esempio:

```
int da = 1;  // estremo sinistro intervallo numero casuale
int a = 20;  // estremo destro intervallo numero casuale: a > da
int n = 10;  // lunghezza sequenza: n <= a – da +1
//-> 5, 1, 18, 12, 10, 7, 8, 19, 13, 9
```

Variazioni

Fornire versioni con ciclo `for`, **while** e **do while**.

7 Filtrare un vettore e restituire una lista

Dato un vettore di interi, creare una nuova lista contenente soltanto i valori negativi.

Esempio:

```
int[] dati = { -10, 30, 45, -30 };
// -> -10, -30
```

Variazioni

Fornire versioni con i cicli `foreach` e `for`.

Memorizzare gli elementi in una lista globale.

Memorizzare gli elementi in una lista passata come argomento al metodo.

8 Caricare i dati da una lista a un vettore

Data una lista di stringhe, creare un nuovo vettore che contenga una copia dei dati.

Esempio:

```
List<string> dati = new List<string>() {"10", "20", "30", "40"},
// -> "10", "20", "30", "40"
```

Variazioni

Fornire versioni con i cicli `foreach` e `for`.

Convertire gli elementi e memorizzarli in un vettore di interi.

9 Calcolare i totali di riga di una matrice

Problema

Sia data una matrice `double` Nx4 che contiene dei dati nelle prima 3 colonne. Calcolare il totale di ogni riga, memorizzandolo nell'ultima colonna

Esempio:

```
double[,] dati =
{
    {10, 20, 30, 0},
    {100, 200, 300, 0},
    {1000, 2000, 3000, 0},
    {10000, 20000, 30000, 0},
};
//->
//    {10, 20, 30, 60},
//    {100, 200, 300, 600},
//    {1000, 2000, 3000, 6000},
//    {10000, 20000, 30000, 60000},
```

Variazioni

Supporre che la matrice contenga dati in tutte le colonne; memorizzare i totali di riga in un vettore (la lunghezza del vettore corrisponde al numero di righe della matrice).

10 Calcolare i totali di colonna di una matrice

Problema

Sia data una matrice `double` Nx4 che contiene dei dati nelle prima N-1 righe. Calcolare il totale di ogni colonna, memorizzandolo nell'ultima riga

Esempio:

```
double[,] dati =
{
    {10, 20, 30, 40},
    {100, 200, 300, 400},
    {1000, 2000, 3000, 4000},
    {0, 0, 0, 0},
};
//->
//    {10, 20, 30, 40},
//    {100, 200, 300, 400},
//    {1000, 2000, 3000, 4000},
//    {1110, 2220, 3330, 4440},
```

Variazioni

Supporre che la matrice contenga dati in tutte le righe; memorizzare i totali di colonna in un vettore (la lunghezza del vettore corrisponde al numero di colonne della matrice).

Supporre che la matrice, pur contenendo dei numeri, sia di tipo `string`. (Memorizzare i totali di colonna in un vettore `double`.)

11 Calcolare la funzione di 2° grado

Problema

Sia data una matrice `double` Nx2 contenente, nella prima colonna, le X della funzione di 2° grado:

$$y = 2\ x^2 + x$$

Calcolare le Y e memorizzarle nella seconda colonna.

```
double[,] dati =
{
    {1, 0},
    {2, 0},
    {3, 0},
    {5, 0},
};
//->
//    {1, 3},
//    {2, 10},
//    {3, 21},
//    {5, 55},
```