



Array mono- e bi-dimensionali

Cicli while e do-while



Array Monodimensionali

Nell'uso di array

- L'operazione sull'indice può essere un decremento. Per stampare gli elementi di un array di 15 elementi in ordine inverso

```
for(i=14; i >= 0; i--)  
    System.out.println(elementi[i]);
```

- Si possono avere anche più operazioni di inizializzazione e sugli indici separate da una , (virgola)

```
for(i=0, j=14; i < 15 ; i++, j--)  
    System.out.println(elementi[i]+" | " + elementi[j]);
```

visualizza i valori in ordine di posizione crescente e decrescente.

```
i < elementi.length
```



Array – Leggere e stampare i valori di un vettore

```
class dueindici
{
    public static void main(String args[])
    {
        int i, j, N;
        int vettore[];

        N = Console.readInt("Dimensione vettore:");
        vettore = new int[N];

        for(i=0; i < N; i++)
            vettore[i]= Console.readInt("Ins. Elem. vettore:");

        for(i=0, j=N-1; i < N ; i++, j--)
        { System.out.println(vettore[i]+" | " + vettore[j]);
          System.out.println("____");
        }
    }
}
```

5	-9
16	59
3	3
59	16
-9	5



Array – Assegnare e stampare i valori di un vettore

```
class matrice
{
    . . . . .
    public void AssegnaeStampaMatrice()
    { int i,j;
      double[][] matrice;

      matrice= new double[5][5];
      for(i=0; i<matrice.length;i++)
          for(j=0; j<matrice[i].length; j++)
              matrice[i][j]=i*j;

      for(i=0;i<matrice.length;i++)
      {
          for(j=0; j<matrice[i].length; j++)
              System.out.print(" " + matrice[i][j]);
          System.out.println();
      }
    }
}
```

0	0	0	0	0
0	1	2	3	4
0	2	4	6	8
0	3	6	9	12
0	4	8	12	16



Istruzione while

Ciclo while

- L'istruzione **while** serve per eseguire un ciclo: una o più operazioni vengono eseguite finché la condizione è vera:

```
while (condizione)
{
    istruzioni ;
}
```

- La condizione è definita come una espressione booleana.
- **Inizialmente si controlla la condizione**, se è vera si eseguono le istruzioni, alla fine si ricontra la condizione e si prosegue finché la condizione rimane vera (**true**).



Istruzione while

ESEMPI

```
int x = 0 ;
while (x < 50)
{
    x = x + 9 ;
}
```

```
num1 = num2;
while (num1 == num2)
{
    if (a == true)
        num1 = num2 + 1;
}
```

```
int vara = 0, varb =100;
while (varb > vara)
{
    vara = vara + 10;
    varb = varb - 10;
}
```

Quando non ci sono indici da gestire nei cicli conviene usare il ciclo while o do-while.



Istruzione while

```
int vet1[] = new int[20];  
int vet2[] = new int[20];  
  
for(int i=0; i< 20; i++)  
    vet1[i] = vet2[i];
```

```
int vet1[] = new int[20];  
int vet2[] = new int[20];  
  
int i=0;  
while(i< 20)  
{  
    vet1[i] = vet2[i];  
    i=i+1;  
}
```

Queste due operazioni sono equivalenti

```
for(iniz_ind; cond; op_ind)  
{ istruzioni; }  
↔  
iniz_ind ;  
while (cond)  
{ istruzioni;  
  op_ind ;  
}
```



Istruzione do-while

Ciclo do-while

- L'istruzione **do-while** serve per eseguire un ciclo controllando la condizione **dopo aver eseguito le istruzioni**.

```
do  
{  
    istruzioni ;  
}  
while (condizione);
```

- Si eseguono le istruzioni, alla fine si controlla la condizione, se è vera si riesegue il ciclo.
- **Si usa quando è necessario che le istruzioni vengano eseguite almeno una volta !**



Istruzione while

ESEMPI

```
int x = 0;
do
{
    x = x + 10;
    System.out.println(x);
}
while (x < 100);
```

```
int x;
do
{
    x = Console.readInt("Inserire numero positivo");
}
while (x <= 0);
```



do-while : esempio fattoriale

```
class fattoriale2
{
    public static void main(String args[])
    {
        int n;
        int fatt = 1;
        int i = 1;
        n = Console.readInt("Inserire n:");
        do
        {
            fatt = fatt * i;
            i = i + 1;
        } while (i <= n);
        System.out.println("Fattoriale di"+n + " = " + fatt);
    }
}
```



Istruzioni break e continue

Istruzioni break e continue

- Le istruzioni **break** e **continue** servono rispettivamente
 - per interrompere un ciclo (**break**)
 - per interrompere **una iterazione** di un ciclo (**continue**)controllando la condizione di uscita in un punto interno di un ciclo (non all'inizio ne alla fine).

```
while (cond)
{
    istruzione1;
    istruzione2;
    if (c==d)
        break;
    istruzione3;
    istruzione4;
}
Istruzione5;
```

```
while (cond)
{
    istruzione1;
    istruzione2;
    if (c==d)
        continue;
    istruzione3;
    istruzione4;
}
Istruzione5;
```



Array in Java – Massimo tra N numeri

```
class cercamassimo
{
    public static void main(String args[])
    {
        int seq[];
        int max, ind;

        seq= new int[10];

        ind = 0;
        while (ind < 10)
        {
            seq[ind] = Console.readInt("dammi un numero");
            ind = ind + 1;
        }
        max = seq[0];
        ind = 1;
        while(ind < 10)
        {
            if (seq[ind] > max)
                max = seq[ind];
            ind = ind+1;
        }
        System.out.println ("Massimo = " + max);
    }
}
```