BCSCHOOL

Empowering Digital Skills For The Jobs Of The Future

by

BC Soft
consulenza sistemi informativi

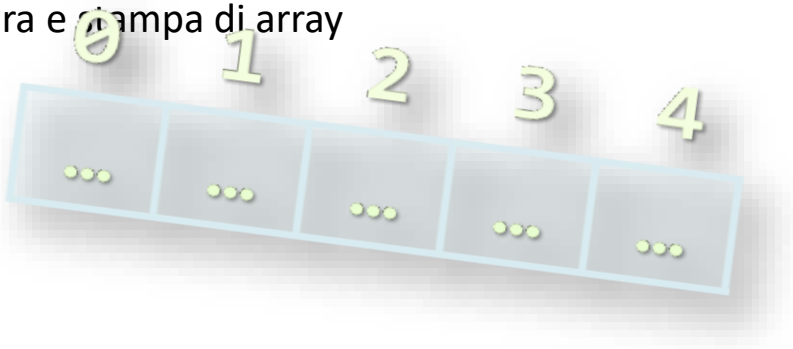
AiRED
Artificial Intelligence Ricerca e Sviluppo

Academy .NET

Gli Array

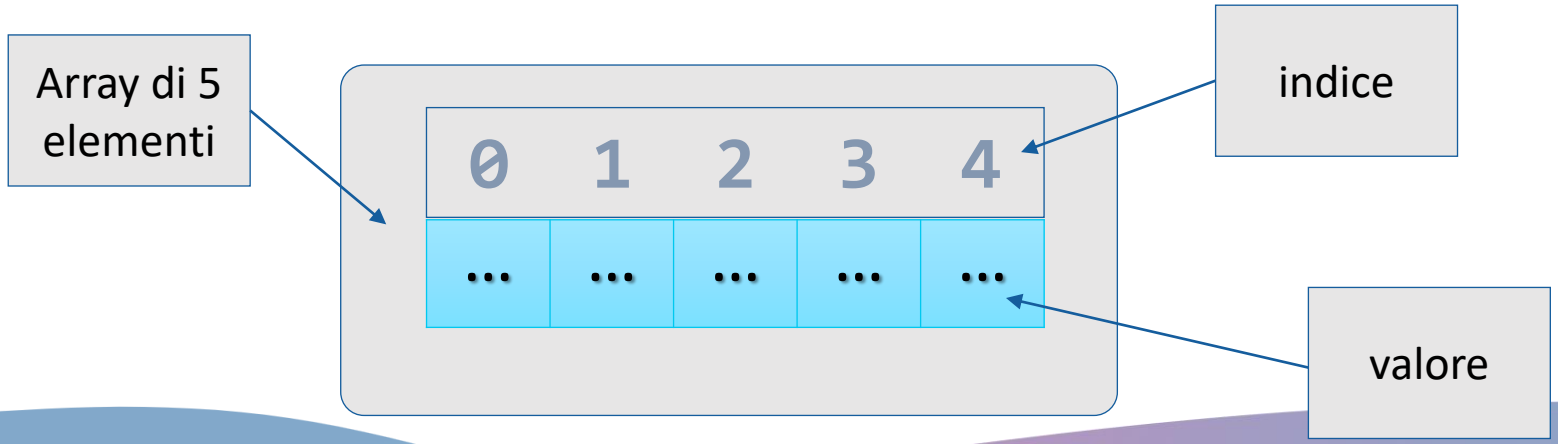
Sommario

- Operazioni sugli array
- Definizione di array
- Inizializzazione di array
- Elaborazione di array
- Tipo valore vs tipo riferimento
- Lettura e stampa di array



Cosa sono gli array?

- Nella programmazione, un **array** è una sequenza di elementi
- Gli elementi sono numerati da **0** a **Length – 1**
- Gli elementi nell'array sono dello stesso tipo (es. **int**)
- Gli array **hanno dimensioni fisse** (**Array.Length**), ne consegue che non possano essere ridimensionati



Array

- Allocazione di un array di 10 numeri interi:

```
int[] numbers = new int[10];
```

Gli elementi iniziano a indice == 0

- Assegnazione di valori agli elementi dell'array:

```
for (int i = 0; i < numbers.Length; i++)
   numbers[i] = 1;
```

Length rappresenta la lunghezza dell'array

- Accesso agli elementi dell'array per indice:

```
numbers[5] = numbers[2] + numbers[7];
numbers[10] = 1; // IndexOutOfRangeException
```

L'operatore **[]** permette l'accesso all'indice

Giorni della settimana - Esempio

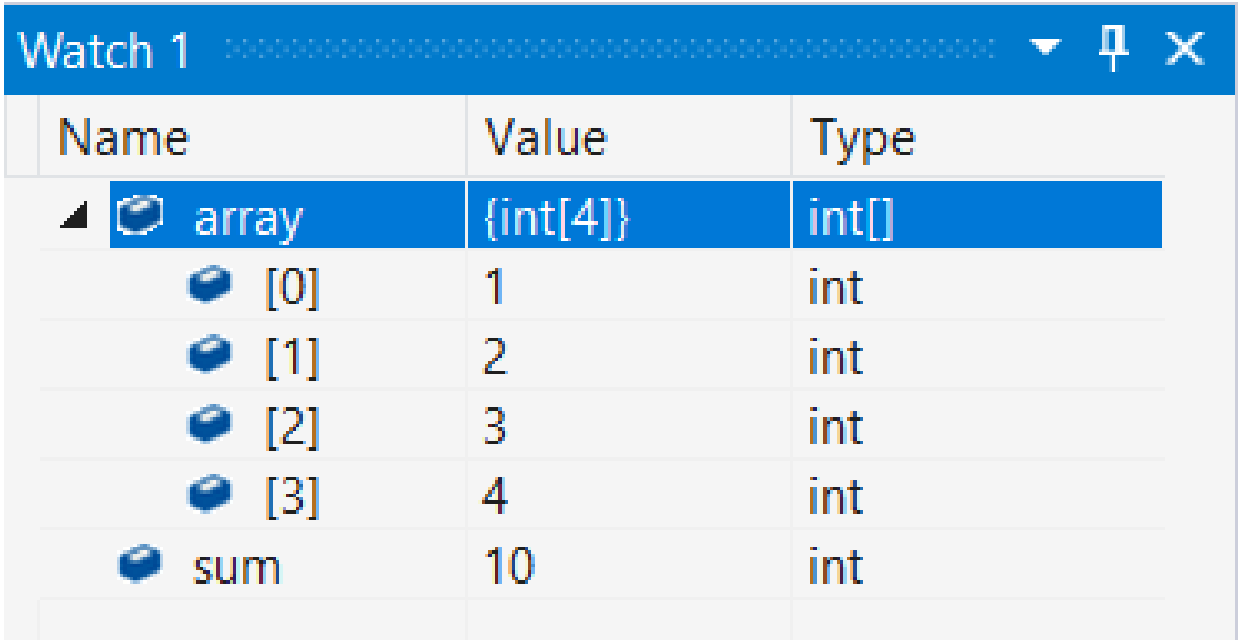
I giorni della settimana possono essere memorizzati in un array di stringhe:

```
string[] days = {
   "Monday",
   "Tuesday",
   "Wednesday",
   "Thursday",
   "Friday",
   "Saturday",
   "Sunday"
};
```

Expression	Value
days[0]	Monday
days[1]	Tuesday
days[2]	Wednesday
days[3]	Thursday
days[4]	Friday
days[5]	Saturday
days[6]	Sunday

Problema - somma degli elementi dell'array

Leggi un array di numeri interi e calcola la loro somma:



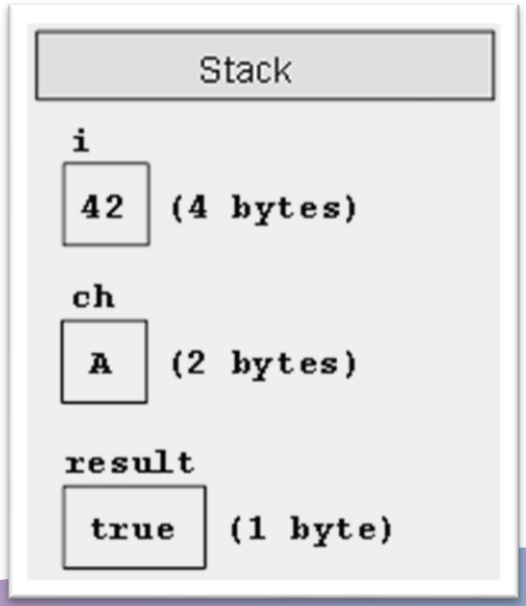
Soluzione - somma degli elementi dell'array

```
var numberOfElements = int.Parse(Console.ReadLine());
var array = new int[numberOfElements];

for (int i = 0; i < array.Length; i++)
  array[i] = int.Parse(Console.ReadLine());

var sum = 0;
for (int i = 0; i < array.Length; i++)
  sum += array[i];

Console.WriteLine(sum);
```

Value type vs Reference Type

- Le variabili di **tipo valore** contengono direttamente i propri dati
  int, **float**, **double**, **bool**, **char**, **DateTime**, **BigInteger**, ...
  Value types - C# reference | Microsoft Docs
- Ogni variabile ha la propria copia dei dati

```
int i = 42;
char ch = 'A';
bool result = true;
```



Tipo reference

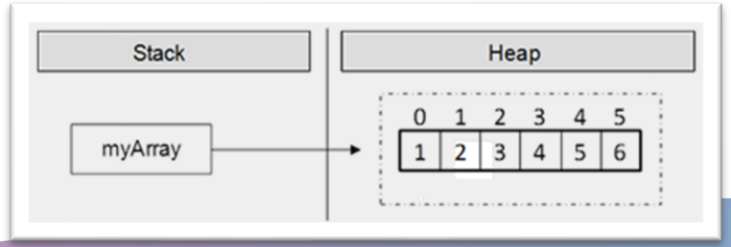
- Le variabili **di tipo riferimento** contengono il riferimento (puntatore / indirizzo di memoria) dei dati stessi

  string, int[], char[], string[], Random, istanze di classi, interfacce, delegate

- Due variabili di **tipo riferimento** possono far riferimento allo stesso oggetto:

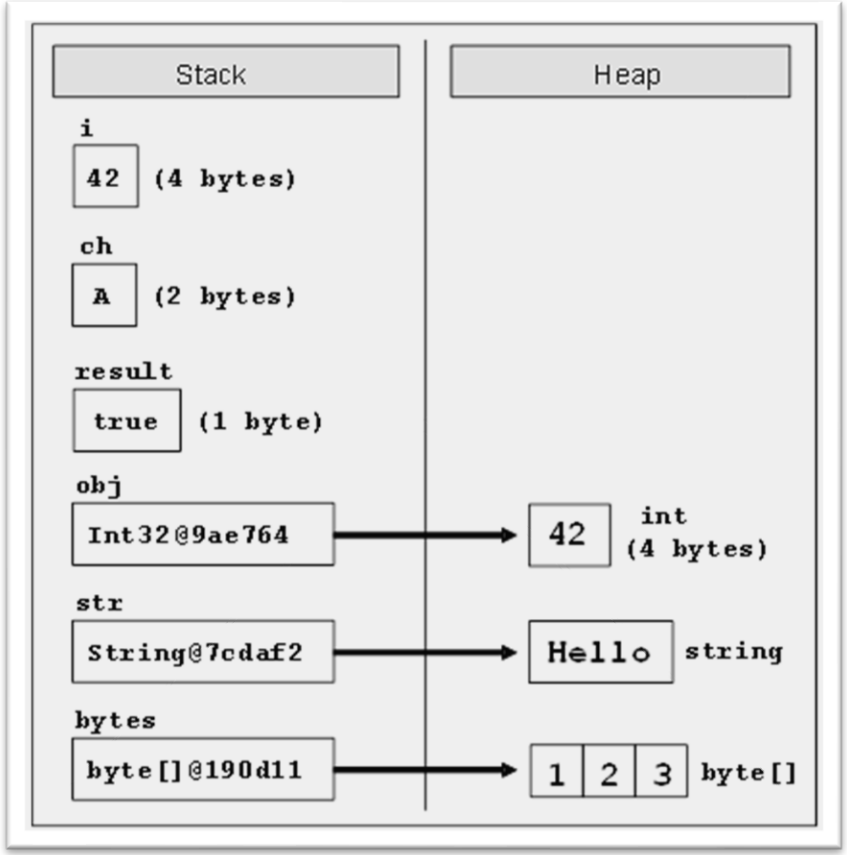
  Le operazioni su entrambe le variabili accedono / modificano gli stessi dati

```
var arr = new int[] { 1, 2, 3, 4, 5, 6 };
```



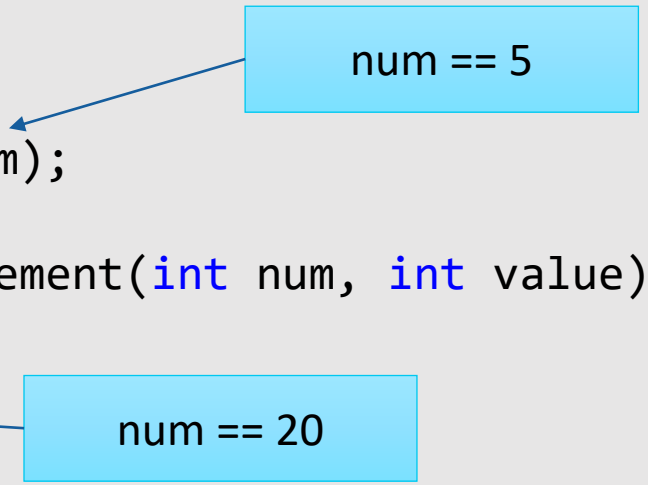
Value type vs Reference Type

```
int i = 42;
char ch = 'A';
bool result = true;
object obj = 42;
string str =
  "Hello";
byte[] bytes =
  { 1, 2, 3 };
```



Esempio: value type

```
public static void Main()
{
    int num = 5;
    Increment(num, 15);
    Console.WriteLine(num);
}
private static void Increment(int num, int value)
{
    num += value;
}
```



Esempio: reference type

```
public static void Main()
{
    int[] nums = { 5 };
    Increment(nums, 15);
    Console.WriteLine(nums[0]);
}

private static void Increment(int[] nums, int value)
{
    nums[0] += value;
}
```

num == 20

num == 20

Leggere array dalla console

- Lettura dalla console della lunghezza dell'array:

```
int n = int.Parse(Console.ReadLine());
```

- Creazione array di una data dimensione **n** e lettura dei suoi elementi:

```
int[] arr = new int[n];

for (int i = 0; i < n; i++)
{
    arr[i] = int.Parse(Console.ReadLine());
}
```

Leggere valori array da singola linea

- Gli array possono essere letti da una singola riga di valori separati da spazi:

```
string values = Console.ReadLine();

string[] items = values.Split(' ');

int[] arr = new int[items.Length];

for (int i = 0; i < items.Length; i++)
    arr[i] = int.Parse(items[i]);
```

.Split(‘ ’) suddivide la stringa di input in sottostringhe in base a uno o più delimitatori

Lettura di array da singola riga

- Leggere un array di numeri interi utilizzando la programmazione funzionale:

```
using System.Linq;
…
var inputLine = Console.ReadLine();
string[] items = inputLine.Split(' ');
int[] arr = items.Select(int.Parse).ToArray();
```

```
2 8 30 25 40 72 - 2 44 56
```

- più breve:

```
int[] arr = Console.ReadLine().
Split(' ').Select(int.Parse).ToArray();
```

Stampare array in console

- Per stampare tutti gli elementi dell'array, è possibile utilizzare un ciclo for:

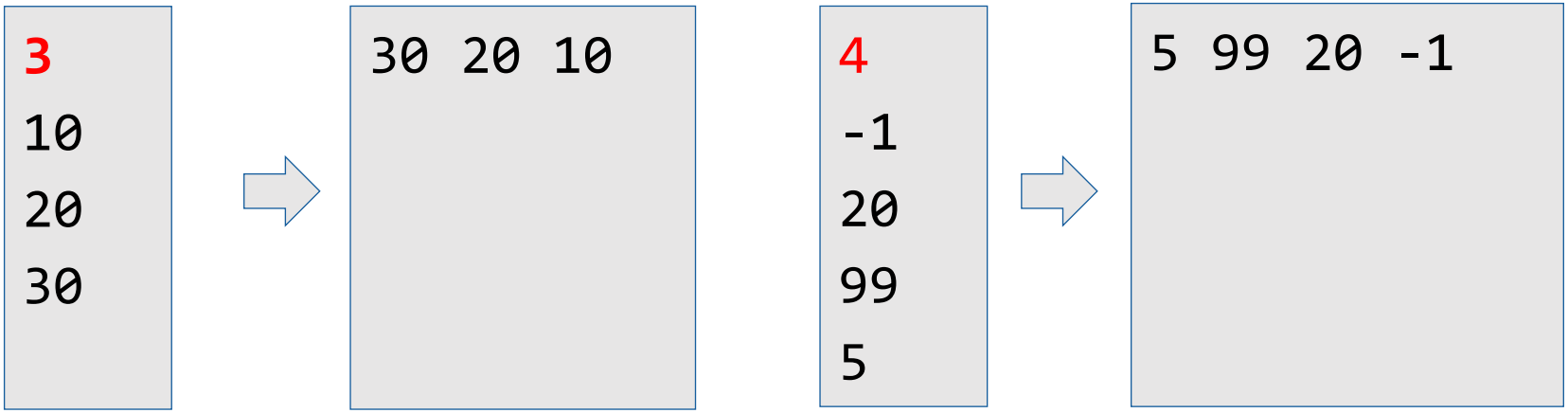
  Separare gli elementi con uno spazio bianco o una nuova riga

- Esempio:

```
string[] arr = {"one", "two", "three", "four", "five"};
// Process all array elements
for (int index = 0; index < arr.Length; index++)
{
  // Print each element on a separate line
  Console.WriteLine("arr[{0}] = {1}", index, arr[index]);
}
```

Problema: invertire array di numeri interi

- Leggi un array di **numeri interi** e **stampa** all'inverso i suoi elementi (su una singola riga, separati da spazi):



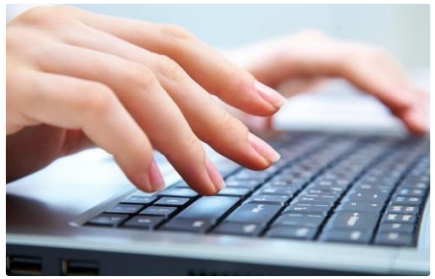
Soluzione: invertire array di numeri interi

```
// Read the array (a number n + n lines of integers)
var n = int.Parse(Console.ReadLine());
var arr = new int[n];
for (int i = 0; i < n; i++)
  arr[i] = int.Parse(Console.ReadLine());

// Print the elements from the last to the first
for (int i = n-1; i >= 0; i--)
  Console.Write(arr[i] + " ");
Console.WriteLine();
```

Stampare Array con foreach / String.Join(…)

- Utilizzo di **foreach-loop**:

```
int[] arr = { 10, 20, 30, 40, 50};
foreach (var element in arr)
  Console.WriteLine(element)
```



- Utilizzo di **string.Join**(separator, array):

```
int[] arr = { 1, 2, 3 };
Console.WriteLine(string.Join(", ", arr)); // 1, 2, 3
string[] strings = { "one", "two", "three", "four" };
Console.WriteLine(string.Join(" - ", strings));
// one - two - three - four
```

Domande e approfondimenti

Academy .NET