




Tipi di dato avanzati

String, struct, enum, tipi per valore
e per riferimento



Cast

Operazione che **trasforma** una variabile cambiando momentaneamente il tipo di dato.

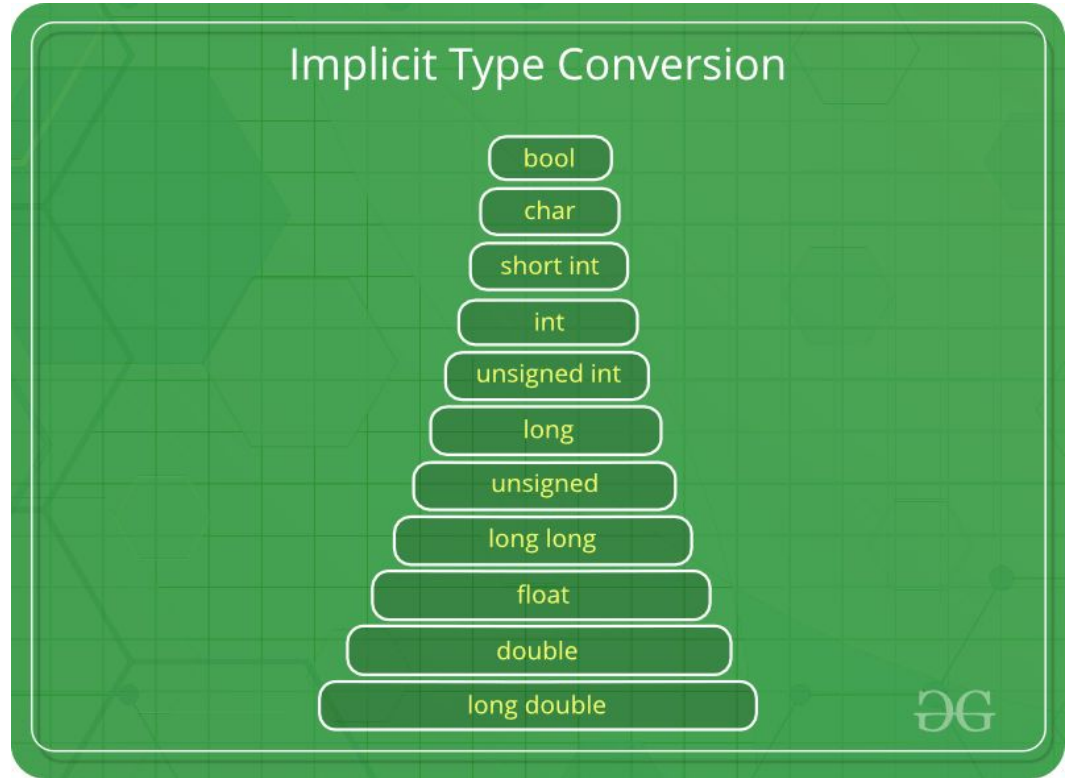
Il cast può essere:

- **implicito**: quando il cambio di dato non richiede una forzatura perché matematicamente o logicamente permesso e quindi viene applicato in automatico.
- **esplicito**: quando il cambio di dato automatico (cast implicito) non è considerato sicuro dal linguaggio e quindi il programmatore deve applicare la forzatura esplicitamente.

Gerarchia nel cast

Il cast **implicito** è possibile quando un tipo di dato "più piccolo" viene trasformato in un tipo di dato "più grande" (es. int -> long)

Per effettuare il contrario si deve ricorrere al cast **esplicito**, stando attenti a possibili **errori** a run time.



Considerazione e esempi sul cast

Quindi il **cast implicito**,
quando ammesso, è
sempre sicuro.

Il **cast esplicito** è sempre
possibile ma passibile di
errori o comportamenti
indesiderati.

```
uint x = 5;  
int y = x; // Cannot implicitly convert type 'uint' to 'int'
```

```
int x = 5;  
uint y = x; // Cannot implicitly convert type 'int' to 'uint'
```

```
double d = 6.626e34;  
int i = d; // Cannot implicitly convert type 'double' to 'int'
```

```
int x = 8;  
double d = x;
```

```
double d = 6.626e34;  
int i = (int)d; // cast senza errore ma i=-2147483648 cioè int.MinValue
```

```
double d = 25;  
int i = (int)d; // 25 cast senza errore
```

Tipi di dati primitivi

Nelle slide precedenti abbiamo visto i tipi di dato **numerici**, sia interi che con parte decimale, e quelli booleani. (es. int, double, bool, ecc...)

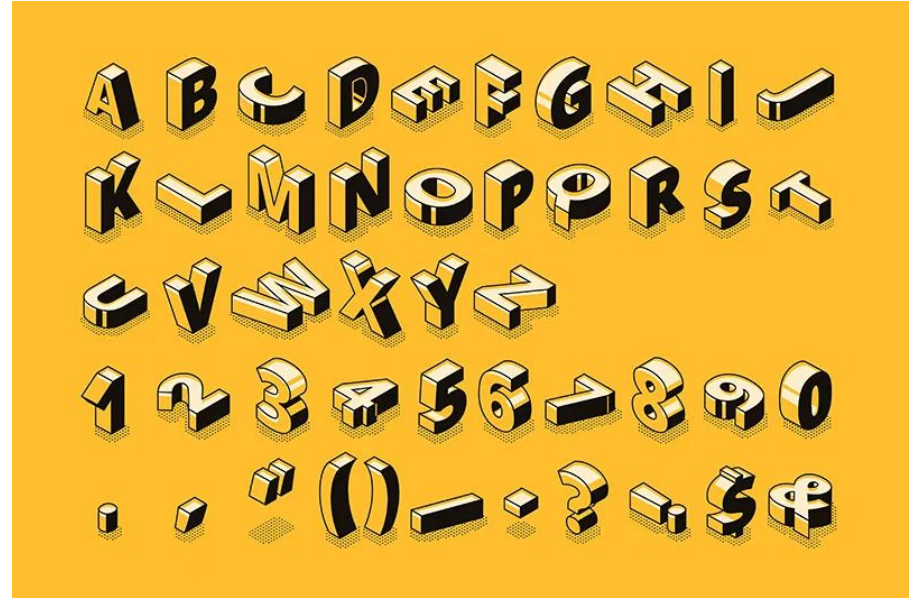
Questi tipi di dato sono detti **primitivi** perché sono dati semplici, cioè **non scomponibili** in ulteriori parti.

Dobbiamo ancora vedere un altro tipo di dato appartenente a questa famiglia: il **char**.

Testo

Il testo è una sequenza di **caratteri**.

In informatica, dove le variabili sono rappresentate in memoria come sequenze di 0 e 1, come possiamo rappresentare i caratteri?



ASCII table

ASCII Code

Char.	ASCII	Char.	ASCII	Char.	ASCII
@	64	U	85	j	106
A	65	V	86	k	107
B	66	W	87	l	108
C	67	X	88	m	109
D	68	Y	89	n	110
E	69	Z	90	o	111
F	70	[91	p	112
G	71	\	92	q	113
H	72]	93	r	114
I	73	^	94	s	115
J	74	_	95	t	116
K	75	`	96	u	117
L	76	a	97	v	118
M	77	b	98	w	119
N	78	c	99	x	120
O	79	d	100	y	121
P	80	e	101	z	122
Q	81	f	102	{	123
R	82	g	103		124
S	83	h	104	}	125
T	84	i	105	~	126

B → 1000010

L → 1101100

U → 1110101

e → 1100101

Originariamente creata come standard usato dai telegrafi.

Tabella dove le lettere dell'alfabeto, ma anche i simboli usati nella lingua scritta, vengono associati a **numeri** corrispondenti.

In questo modo la lettera B può essere memorizzata in RAM come il numero 66, cioè 1000010 in codice binario.

	1	2	3	4	5
A	+	-	-	-	-
B	+	+	+	+	-
C	+	+	+	+	-
D	+	+	+	+	-
E	+	+	+	+	-
F	+	+	+	+	-
G	-	+	+	+	-
H	+	+	+	+	-
I	+	+	+	+	-
J	+	+	+	+	-
K	+	+	+	+	-
L	+	+	+	+	-
M	+	+	+	+	-
N	-	+	+	+	-
O	+	+	+	+	-
P	+	+	+	+	-
Q	+	+	+	+	-
R	+	+	+	+	-
S	+	+	+	+	-
T	+	+	+	+	-
U	+	+	+	+	-
V	+	+	+	+	-
W	+	+	+	+	-
X	+	+	+	+	-
Y	+	+	+	+	-
Z	+	+	+	+	-
[-	-	-	+	+
\	-	-	-	+	+
]	-	-	-	+	+
^	-	-	-	+	+
_	-	-	-	+	+
`	-	-	-	+	+
a	-	-	-	+	+
b	-	-	-	+	+
c	-	-	-	+	+
d	-	-	-	+	+
e	-	-	-	+	+
f	-	-	-	+	+
g	-	-	-	+	+
h	-	-	-	+	+
i	-	-	-	+	+
j	-	-	-	+	+
k	-	-	-	+	+
l	-	-	-	+	+
m	-	-	-	+	+
n	-	-	-	+	+
o	-	-	-	+	+
p	-	-	-	+	+
q	-	-	-	+	+
r	-	-	-	+	+
s	-	-	-	+	+
t	-	-	-	+	+
u	-	-	-	+	+
v	-	-	-	+	+
w	-	-	-	+	+
x	-	-	-	+	+
y	-	-	-	+	+
z	-	-	-	+	+
{	-	-	-	+	+
	-	-	-	+	+
}	-	-	-	+	+
~	-	-	-	+	+

ASCII table: alfabeto

Dec	Hex	Char
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9

Dec	Hex	Char
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z

Dec	Hex	Char
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Char

Tipo di **dato primitivo** che rappresenta il **singolo carattere**.

Per natura, questo tipo di dato è interpretabile sia come lettera sia come il numero intero associato nella tabella ASCII.

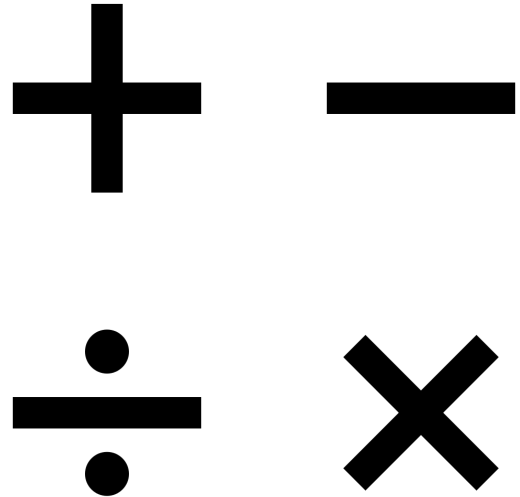
Possiamo passare da carattere a codice ASCII e viceversa molto agilmente, quando vogliamo e, soprattutto, in modo automatico.

```
char c = 'h';  
int code = c;  
Console.WriteLine("Il carattere " + c + " è il codice ASCII " + code);  
char c2 = (char)code;  
Console.WriteLine("Il codice ASCII " + code + " è il carattere " + c2);
```

Operazioni tra char

di default viene trattato come un numero e quindi tutti gli operatori matematici possono essere applicati ai char.

```
Console.WriteLine('a' / 'z'); // 0
Console.WriteLine('a' + 'b'); // 195
Console.WriteLine('z' - 'a'); // 25
Console.WriteLine('a' * 2); // 194
Console.WriteLine('a' > 'z'); // false
```

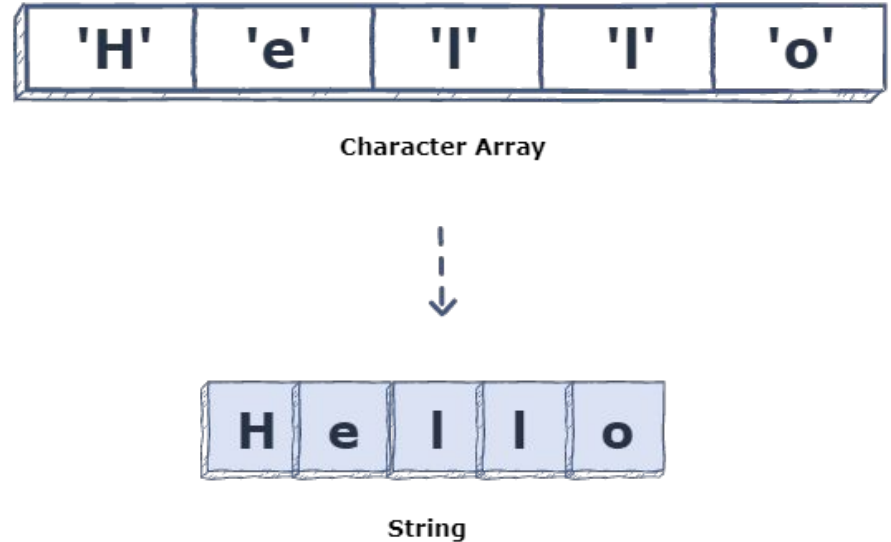


Stringhe

La stringa è un tipo di dato che rappresenta il **testo**.

Internamente il testo viene archiviato come una raccolta di **sola lettura** sequenziale di elementi char (caratteri).

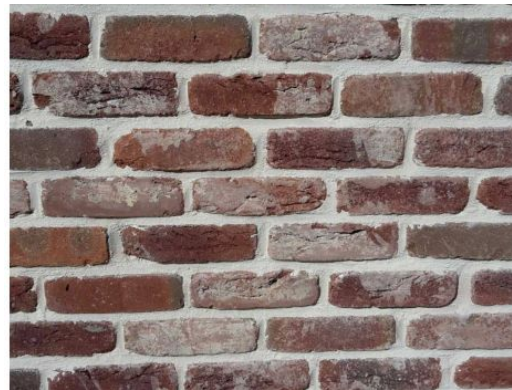
Quindi una stringa può essere vista anche come **array** di caratteri ma non modificabile.



String e char

Diversi modi per creare una stringa.

```
char[] chars = new char[] { 'w', 'o', 'r', 'd' };  
string s1 = new string(chars);  
string s2 = "word";  
Console.WriteLine(s1 == s2); // true
```



Unire variabili testuali

Per unire due stringhe tra loro si usa l'operatore +

```
string s1 = "tavolo";  
string s2 = "bicchiere";  
string s3 = s1 + s2; // tavolobicchiere
```

Allo stesso modo si può unire una stringa con una variabile di altra natura: quello che succede è che la variabile non stringa viene convertita nella sua forma testuale.

```
int x = 4;  
string s4 = s3 + x; // tavolobicchiere4
```

Alcuni metodi delle stringhe in .NET

- Length: <https://docs.microsoft.com/it-it/dotnet/api/system.string.length>
- IndexOf: <https://docs.microsoft.com/it-it/dotnet/api/system.string.indexof>
- Contains: <https://docs.microsoft.com/it-it/dotnet/api/system.string.contains>
- Substring: <https://docs.microsoft.com/it-it/dotnet/api/system.string.substring>
- ToUpper: <https://docs.microsoft.com/it-it/dotnet/api/system.string.toupper>
- Replace: <https://docs.microsoft.com/it-it/dotnet/api/system.string.replace>
- Insert: <https://docs.microsoft.com/it-it/dotnet/api/system.string.insert>

Esempio con le stringhe

```
string ungaretti = "M'illumino d'immenso";  
int len = ungaretti.Length;  
Console.WriteLine("La poesia è lunga " + len + " caratteri.");  
int oPosition = ungaretti.IndexOf('o');  
Console.WriteLine("La prima lettera o si trova in posizione " + oPosition);  
bool minoExists = ungaretti.Contains("mino");  
Console.WriteLine("La poesia contiene la parola mino? " + minoExists);  
string word = ungaretti.Substring(2, 8);  
string wordMaiusc = word.ToUpper();  
string newWord = wordMaiusc.Replace('O', 'A');  
string sentence = newWord.Insert(0, "La stanza si ");  
Console.WriteLine(sentence);  
Console.WriteLine("Il primo carattere della frase è " + sentence[0]);
```


Esercizi (NON usare i metodi forniti da .NET)

1. Fornita una parola in input, dire se è palindroma.
2. Fornita una frase in input, contare quante vocali e quante consonanti ci sono.
(N.B. la frase può contenere lettere maiuscole o minuscole e anche simboli come le virgole e i punti)
3. Generare una frase casuale (non deve per forza avere un senso logico)
Es: Fsdads dsfedds errots? Gsdfzsf, fgahhsd!
4. Fornita una frase in input, eliminare le vocali e stampare la frase risultante.
5. Chiedere in input una frase e un numero intero, cifrare la frase utilizzando il cifrario di Cesare e stampare la frase risultante.
([https://it.wikipedia.org/wiki/Cifrario di Cesare](https://it.wikipedia.org/wiki/Cifrario_di_Cesare))
6. Chiedere in input la frase cifrata con il cifrario di Cesare e la chiave, stampare la frase decifrata.

DateTime

Rappresenta un istante di **tempo**, in genere espresso come **data e ora** del giorno.

Si può inizializzare ad un momento che si desidera.

Si può inizializzare chiedendo al linguaggio.

Namespace System



```
var date1 = new DateTime(2008, 5, 1, 8, 30, 52);  
Console.WriteLine(date1);
```

```
DateTime date1 = DateTime.Now;  
DateTime date2 = DateTime.UtcNow;  
DateTime date3 = DateTime.Today;  
  
// date1 12/13/2020 21:56:53  
// date2 12/13/2020 20:56:53  
// date3 12/13/2020 00:00:00
```

Da testo a DateTime

DateTime.Parse traduce una data in forma **testuale** nell'oggetto **DateTime**.

Datetime.Parse funziona similmente come int.Parse che abbiamo già visto.

```
string s = "30/03/2019 15:56:11";  
DateTime date1 = DateTime.Parse(s);
```

```
string s = Console.ReadLine();  
DateTime date1 = DateTime.Parse(s);
```

Da DateTime a testo

Identificatore di formato	Descrizione	"m"	Minuti, da 0 a 59. Altre informazioni: identificatore di formato personalizzato "m" .	"y"	Anno, da 0 a 99. Altre informazioni: Identificatore di formato personalizzato "y" .	"h"	Ora, usando un orario in formato 12 ore da 1 a 12. Altre informazioni: identificatore di formato personalizzato "h" .
"d"	Giorno del mese, da 1 a 31. Altre informazioni: Identificatore di formato personalizzato "d" .	"mm"	Minuti, da 00 a 59. Altre informazioni: Identificatore di formato personalizzato "mm" .	"yy"	Anno, da 00 a 99. Altre informazioni: Identificatore di formato personalizzato "yy" .	"hh"	Ora, usando un orario in formato 12 ore da 01 a 12. Ulteriori informazioni: identificatore di formato personalizzato "HH" .
"dd"	Giorno del mese, da 01 a 31. Altre informazioni: Identificatore di formato personalizzato "dd" .	"M"	Mese, da 1 a 12. Altre informazioni: Identificatore di formato personalizzato "M" .	"yyy"	Anno, con un minimo di tre cifre. Altre informazioni: Identificatore di formato personalizzato "yyy" .	"H"	Ora, usando un orario in formato 24 ore da 0 a 23. Altre informazioni: Identificatore di formato personalizzato "H" .
"ddd"	Nome abbreviato del giorno della settimana. Altre informazioni: Identificatore di formato personalizzato "ddd" .	"MM"	Mese, da 01 a 12. Altre informazioni: identificatore di formato personalizzato "mm" .	"yyyy"	Anno, come numero a quattro cifre.	"HH"	Ora, usando un orario in formato 24 ore da 00 a 23. Altre informazioni: Identificatore di formato personalizzato "HH" .
"dddd"	Nome completo del giorno della settimana. Altre informazioni: Identificatore di formato personalizzato "dddd" .	"MMM"	Nome abbreviato del mese. Altre informazioni: Identificatore di formato personalizzato "MMM" .				

Il metodo ToString trasforma il DateTime nella forma testuale così come espresso dagli identificatori di formati forniti.

```
string oggi = DateTime.Now.ToString("dd/MMM/yyyy HH:mm:ss");  
Console.WriteLine(oggi); // 13/Dec/2020 21:12:21
```

Operazioni tra DateTime

Due DateTime possono essere comparati utilizzando gli operatori già visti per gli altri tipi di dato.

Confronto di uguaglianza

`dt1 == dt2`

Confronto di minore o maggiore

`dt1 > dt2`

`dt1 < dt2`



Enumerati

Tipo di **dato primitivo** creato dal programmatore e che consiste in un determinato **elenco** di valori **costanti**.

I valori definiti sono in realtà numeri interi di tipo **int**.

Il passaggio **da enum a int** e viceversa è possibile applicando il **cast**.

Perché usarli? Per fissare tutti i possibili valori che una variabile di quel tipo potrà avere.

Per aumentare la leggibilità del codice.

```
enum PrimoPiatto
{
    Spaghetti = 0,
    Maccheroni = 1,
    Fusilli = 2,
    Penne = 3
}
```

```
PrimoPiatto p1 = PrimoPiatto.Spaghetti;
Console.WriteLine(p1); // Spaghetti
int ip1 = (int)p1;
Console.WriteLine(ip1); // 0
int ip2 = 3;
PrimoPiatto p2 = (PrimoPiatto)ip2;
Console.WriteLine(p2); // Penne
```

Accorpamento di informazioni: struct

La **struct** (record, struttura) è un tipo di dato **strutturato** che comprende diversi **elementi** (detti campi o membri) di tipo **eterogeneo**.

A differenza degli array, nelle struct il numero di campi è usualmente stabilito nella definizione del tipo e i campi hanno un nome.

Molto utile per **organizzare** le informazioni in contenitori logici complessi.

Inoltre permette di definire una sola volta le caratteristiche delle entità logiche usate.

```
struct Autore
{
    public string Nome, Cognome;
    public int AnnoNascita;
}

struct Libro
{
    public string Titolo, Editore;
    public Autore Autore;
    public int AnnoPubblicazione;
    public uint NumeroPagine;
    public double Costo;
}

Autore dante = new Autore();
dante.Nome = "Dante";
dante.Cognome = "Alighieri";
dante.AnnoNascita = 1265;

Libro divina = new Libro();
divina.Titolo = "Divina commedia - Inferno";
divina.Editore = "Sconosciuto";
divina.AnnoPubblicazione = 2009;
divina.Autore = dante;
divina.Costo = 25.50;
divina.NumeroPagine = 389;
```

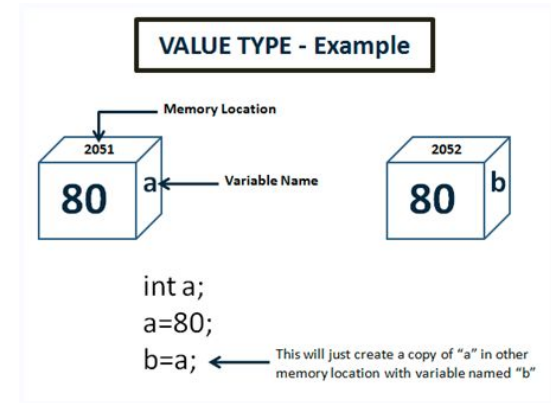
Tipo di valore

Un tipo di valore contiene un **valore** di dati all'interno del proprio spazio di memoria.

Due tipologie:

- tipi predefiniti: int, char, long, double, bool, string, ecc...
- tipi definiti dal programmatore: un enum, una struct

Quando si assegna una variabile di questo tipo ad un'altra variabile, allora viene fatta una **copia** del dato in memoria.



Operazioni tra tipi di valore

I tipi di valore vengono considerati come la "somma" di tutte le loro variabili interne. Questo significa che un qualsiasi tipo di valore, per quanto complesso e articolato sia internamente, viene sempre valutato come l'insieme di tutte le informazioni contenute dentro di sé.

Ad esempio, due struct Persona che contengono gli stessi identici dati, allora verranno considerate come uguali.

Non ci stupisce questo comportamento; è lo stesso comportamento che abbiamo tra due variabili intere che rappresentano lo stesso numero.

Tipo di riferimento

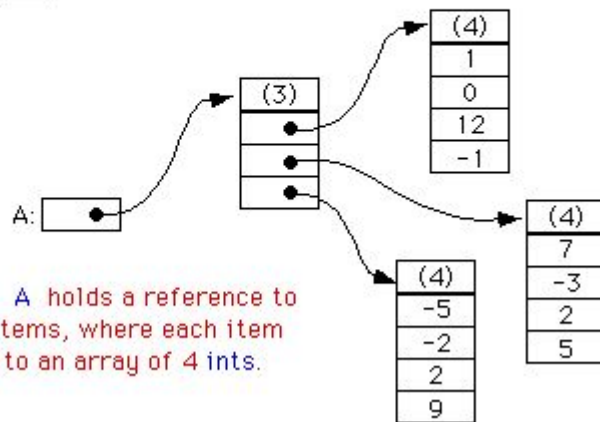
Un tipo di riferimento mantiene un **puntatore** a un'altra posizione di memoria che contiene i dati.

Ad esempio, una matrice è un tipo di riferimento. Immaginiamo di definire la matrice `A[3,4]`. La variabile `A` viene memorizzata in una cella di memoria. Quella cella di memoria non può contenere tutti i numeri contenuti nella matrice. Quindi, i numeri della matrice devono essere memorizzati ognuno nella propria cella.

A:

1	0	12	-1
7	-3	2	5
-5	-2	2	9

If you create an array `A = new int[3][4]`, you should think of it as a "matrix" with 3 rows and 4 columns.



But in reality, `A` holds a reference to an array of 3 items, where each item is a reference to an array of 4 ints.

Esercizio (si possono usare i metodi di .NET)

1. Chiedere in input razza, anno di nascita, sesso, nome proprietario e codice microchip di N cani. Memorizzarli come struct "Cane" in una struttura dati a scelta.
Stampare separatamente i cani femmina e maschi. Stampare la lista di cani più vecchi di 10 anni.
2. Chiedere in input il nome del proprietario, cercare e stampare i dati del/dei cane/i.
3. Ordinare la lista dei cani per anno di nascita, dal più giovane al più vecchio. Stampare la lista così ordinata.