Tipi in C

Tipi numerici nel C

Dimensione e range sono valori tipici!

Tipo	**Range** (tipico)	**Precisione**	**Dimensione**
int	$\pm 2.147.483.647$	–	4 bytes
unsigned	$[0, 4.294.967.295]$	–	4 bytes
long	$\pm(2^{63} - 1)$	–	8 bytes
unsigned long	$[0, 2^{64} - 1]$	–	8 bytes
short	± 32768	–	2 bytes
unsigned short	$[0, 65535]$	–	2 bytes
double	$\pm 10^{308}$	15 cifre	8 bytes
float	$\pm 10^{38}$	6 cifre	4 bytes

Tipi numerici nel C

1. Sarà importante **conoscere la dimensione** quando si debbono allocare **grandi porzioni di memoria**. ESEMPIO:
   - allocare un array di miliardi di elementi di tipo `int`
   - il fatto che un tipo int occupi 8 bytes piuttosto che 4 bytes sarà determinante. Nel primo caso la memoria potrebbe non bastare.
2. **Conoscere intervallo di valori rappresentabili** dei tipi in virgola mobile per non incorrere in overflow. ESEMPIO: per gli interi con ordine di grandezza 10^{10} useremo i long.
3. **Conoscere la precisione p** dei tipi in virgola mobile per non incorrere in underflow o approssimazioni non previsti.

Tipi numerici nel C

Numero di byte corrispondenti ad un determinato tipo

In C esiste **l'operatore** `sizeof()`, che si usa come una fuinzione con argomento il particolare tipo o anche una variabile di un certo tipo.

```
include <stdio.h>
int main()
{
printf("%lu", sizeof(char));
printf("%lu", sizeof(int));
printf("%lu", sizeof(float));
printf("%lu", sizeof(double));
return 0;
}
```

Intervalli numerici e precisione

Header `float.h` e `limits.h`

```
#include <limits> // header da includere
INT_MIN, INT_MAX // più piccolo/grande valore intero
LONG_MIN, LONG_MAX // più piccolo/grande valore long
```

```
#include <floats> //header da includere
// più piccolo/grande valore float positivo
FLT_MIN, FLOAT_MAX
// no.  di cifre significative rappresentabili
// (precisione!)
FLT_DIG, DBL_DIG
```

Altri tipi in C

Tipo	**Range** (tipico)	**Dimensione**
`char`	[-128,+127]	1 byte
`unsigned char`	[0,255]	1 byte
`bool`	{true,false}	1 byte
`void`	#	1 byte

Il tipo `void` sta ad indicare **nessun valore di ritorno**.

In C si usano spesso puntatori void (`void *`) per restituire indirizzi di memoria per i quali il "chiamante" fara un **casting** ad un tipo specifico di puntatore.

Operatori aritmetici, funzioni matematiche di base, conversioni

Operatori aritmetici

Operatore	Num. argomenti	Significato
*	2	Moltiplicazione
/	2	Divisione
+	2	Somma
-	2	Sottrazione
++	1	Incremento
--	1	Decremento
%	2	Modulo

Operatore modulo restituisce il **resto della divisione** tra due numeri **interi**.

Funzioni matematiche (libreria `math.h`)

```
#include <math.h>
```

Funzione	Argomenti	Significato
`pow()`	(x,y) in virgola mobile	x^y
`sqrt()`	(x,y) in virgola mobile	$\sqrt{x}$
`sin()`	(x) in virgola mobile	$\sin(x)$
`abs()`	(x) in virgola mobile	$\lvert x \rvert$
`log()`	(x) in virgola mobile	$\ln x$
`log10(), log2()`	(x) in virgola mobile	$log_{10}(x)$, $log_2(X)$

Tipi numerici: conversioni e promozioni

Attenzione ai passaggi da un tipo ad un altro che **contiene meno informazioni** (e.g. long $\leftarrow$ int, double $\leftarrow$ int, etc.)

Da floating point a intero

```
double d = 2.8965;
int a = d; // Perdita della frazione!
```

Arrotondare allo intero più vicino?

```
double d = 2.8965;
int a = d+0.5; // Arrotondamento OK
```

Tipi numerici: conversioni e promozioni

Divisione / moltiplicazione tra interi

```
float result = 5 / 2; // =2 perdita della frazione!
float result = 5.0 / 2; // =2.5 OK
```

Quale è la differenza?

Conversioni implicite

Per una espressione che contiene interi e numeri in virgola mobile, basta che uno dei membri della espressione sia **esplicitamente floating point**.

Il compilatore opererà una **conversione implicita** in virgola mobile degli altri membri per eseguire moltiplicazioni e divisioni in floating point, **senza alcuna perdita di informazione**.

Casting statico

Casting o conversione esplicita

Il **casting** é un'operazione con cui si indica al compilatore che **deve convertire** una variabile o il risultato di una espressione ad tipo ben definito.

Si rende necessario quando si vuole una **conversione differente** dalla conversione implicita.

(type)value

Casting statico

Esempio di casting

```
 double d = 10.73;
//arrotonda all'intero più vicino
double result = (int) (d + 0.5);
```

FINE