



07 Espressioni

07.00 - Tipi di dato per rappresentare numeri naturali

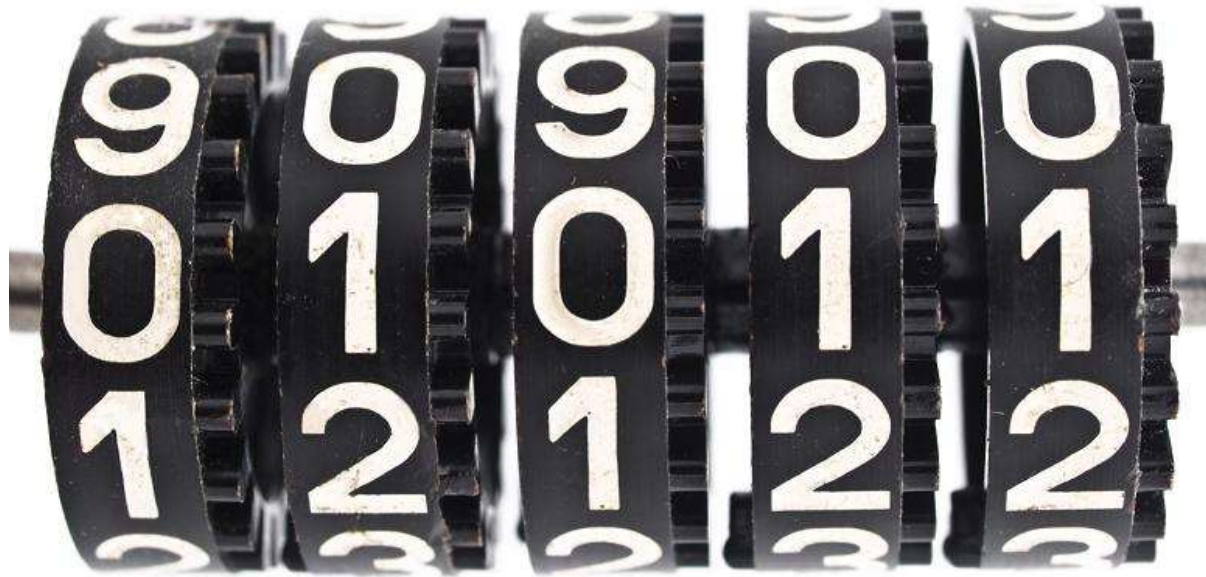
Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

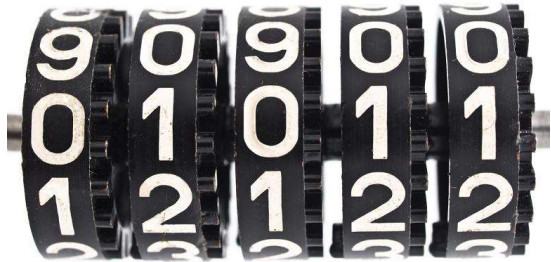
QUESTO MATERIALE DIDATTICO È PER USO PERSONALE DELLO STUDENTE ED È COPERTO DA COPYRIGHT. NE È SEVERAMENTE VIETATA LA RIPRODUZIONE O IL RIUTILIZZO ANCHE PARZIALE, AI SENSI E PER GLI EFFETTI DELLA LEGGE SUL DIRITTO D'AUTORE.

Tipi di dato: scalari



Rappresentazione numeri in base 10

In questo esempio:



- *Minimo numero rappresentabile:*
00000
- *Massimo numero rappresentabile:* $b^n - 1$
99999
- *Numero totale configurazioni:* b^n
 $99999 + 1 = 10^5$.
- *10 corrisponde al numero di simboli della base (utilizzabili come cifra)*
 $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$: b
- *5 corrisponde al numero di cifre:* n

Numeri Interi senza segno

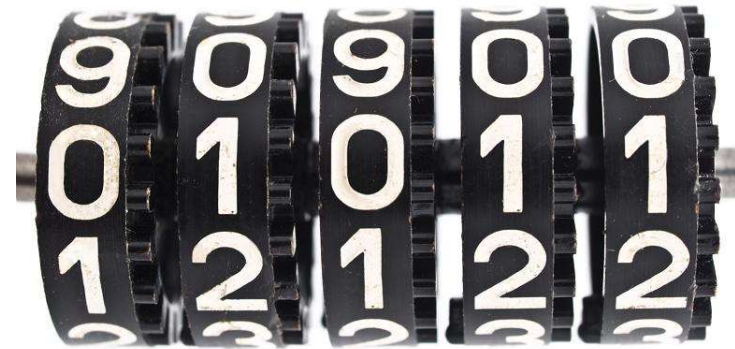
- Nei calcolatori attuali si usano due sole cifre: 0 e 1 $\rightarrow b=2$
- Se si decide di usare n cifre (bit):
 - il numero di configurazioni è 2^n
 - il massimo numero rappresentabile è 2^n-1

Interi senza segno (naturali)

unsigned char	0..255	8 bit
unsigned short	0 .. 65535	16 bit
unsigned int	???	16 o 32 bit
unsigned long	0 .. 4294967295	32 bit (a volte 64)
unsigned long long	0.. 18446744073709551615	64 bit

Overflow

- *Che cosa succede se dalla configurazione 99999 si aggiunge 1?*
- *E se dalla 00000 si toglie 1?*
- *Che cosa succede in linguaggio C in questi casi?*





07 Espressioni

07.01 - Tipi di dato per rappresentare numeri relativi

Corso di Laurea in Ingegneria Elettronica e Informatica

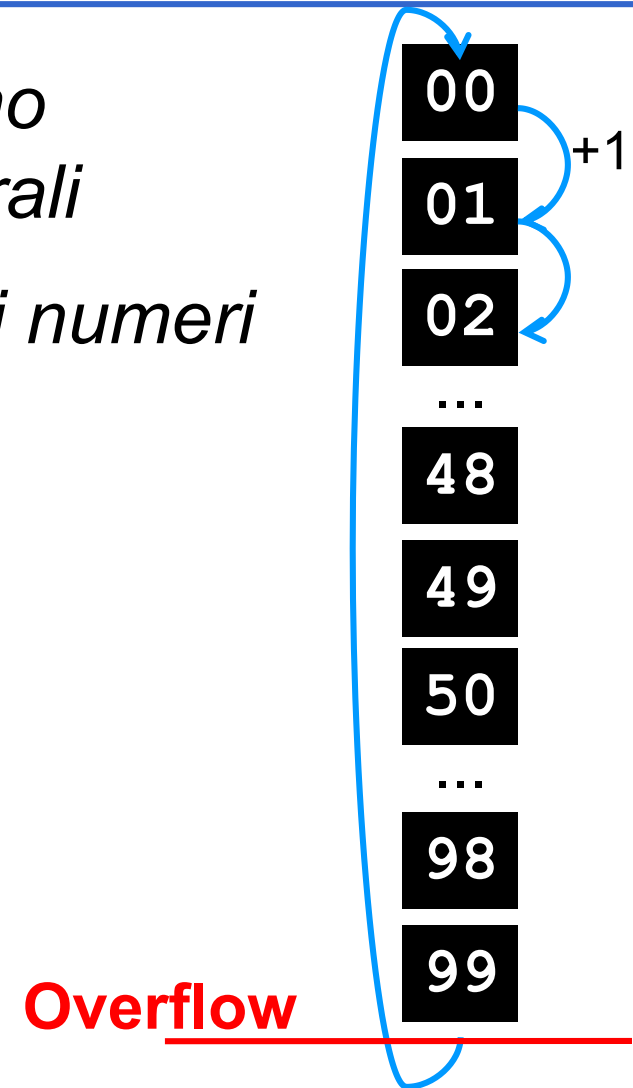
Anno accademico 2020/2021

Prof. MARCO GAVANELLI

QUESTO MATERIALE DIDATTICO È PER USO PERSONALE DELLO STUDENTE ED È COPERTO DA COPYRIGHT. NE È SEVERAMENTE VIETATA LA RIPRODUZIONE O IL RIUTILIZZO ANCHE PARZIALE, AI SENSI E PER GLI EFFETTI DELLA LEGGE SUL DIRITTO D'AUTORE.

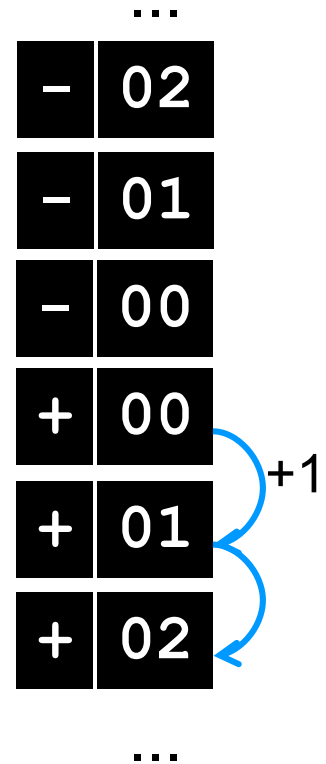
Numeri relativi

- *In questo modo, però, si possono rappresentare solo numeri naturali*
- *Come possiamo rappresentare i numeri negativi?*



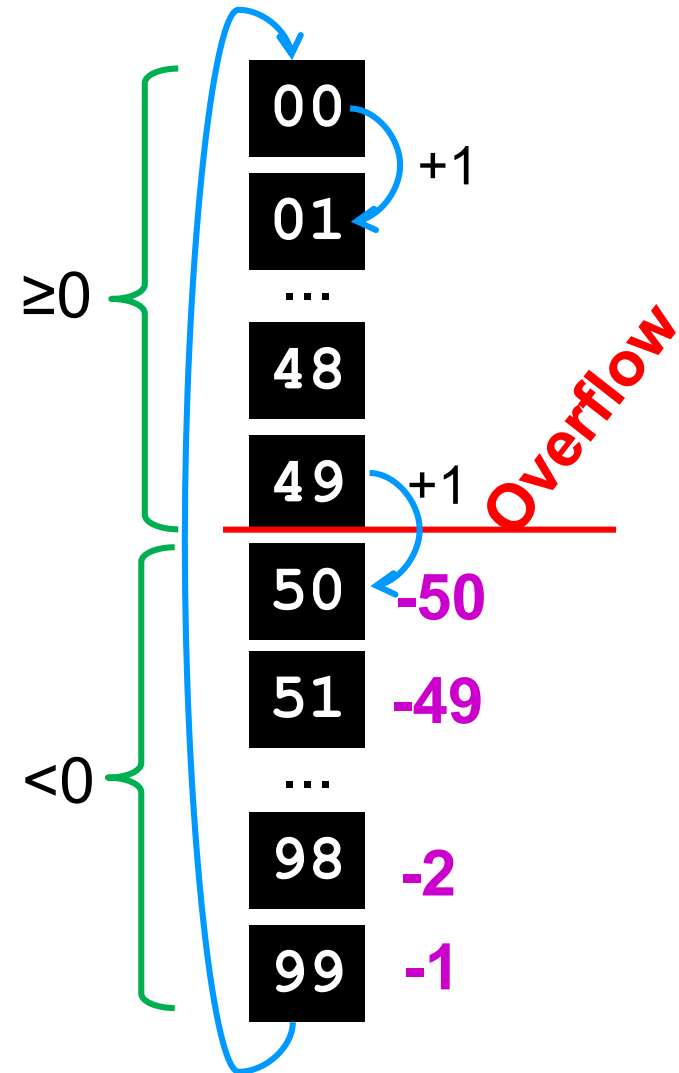
Numeri relativi: modulo e segno

- *La prima cosa che viene in mente è aggiungere un'ulteriore cifra che rappresenta il segno*
 - *Però così avrei due configurazioni che rappresentano il numero 0: +00 e -00*
 - *Non posso riutilizzare i circuiti elettronici che ho già creato per i numeri naturali*
- *Per questi motivi, questa **NON È** la rappresentazione usata in pratica*



Numeri relativi: complemento a base

- Riutilizzare i circuiti, ma cambiare l'interpretazione che diamo dei numeri
- Metà delle configurazioni vengono usate per rappresentare i numeri negativi, e metà per i naturali
- Si ha overflow se si sommano due numeri dello stesso segno e si supera la separazione fra naturali e negativi
- Se voglio usare gli stessi circuiti, allora il numero -1 dovrà essere quel numero a cui, se si somma 1, si ottiene 0



Numeri Interi

Interi con segno

<code>char</code>	-128 .. 127	8 bit
<code>short (int)</code>	-32768 .. 32767	16 bit
<code>int</code>	???	16 o 32 bit
<code>long (int)</code>	-2147483648 .. 2147483647	32 bit (a volte 64)
<code>long long (int)</code>	-9223372036854775808 .. 9223372036854775807	64 bit

Interi senza segno (naturali)

<code>unsigned char</code>	0..255	8 bit
<code>unsigned short</code>	0 .. 65535	16 bit
<code>unsigned int</code>	???	16 o 32 bit
<code>unsigned long</code>	0 .. 4294967295	32 bit (a volte 64)
<code>unsigned long long</code>	0.. 18446744073709551615	64 bit

Numeri Interi

- *La dimensione di alcuni tipi dipende dal compilatore.*
- *Comunque, dato un compilatore, le dimensioni (in bit) sono sempre ordinate come segue:*

`short ≤ int ≤ long ≤ long long`



07 Espressioni

07.02 - Tipi di dato per rappresentare numeri in virgola mobile

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

QUESTO MATERIALE DIDATTICO È PER USO PERSONALE DELLO STUDENTE ED È COPERTO DA COPYRIGHT. NE È SEVERAMENTE VIETATA LA RIPRODUZIONE O IL RIUTILIZZO ANCHE PARZIALE, AI SENSI E PER GLI EFFETTI DELLA LEGGE SUL DIRITTO D'AUTORE.

Rappresentazione floating point

- Un'altra rappresentazione dei numeri è quella scientifica

mantissa 5.87 x 10 -32 **esponente**

- Il numero di cifre riservate alla **mantissa** definisce la **precisione** del numero
- Il numero di cifre riservate all'**esponente** definisce la **dimensione massima** del numero (possibilità di rappresentare numeri con valore assoluto molto grande o molto piccolo)
- Viene chiamata rappresentazione **floating point** (o in **virgola mobile**), perché la posizione della virgola decimale dipende anche dall'esponente
 - $5.470000 \times 10^4 = 54700.00$
 - $5.470000 \times 10^1 = 54.70000$

Numeri reali

- **float** *singola precisione (32 bit)*
- **double** *doppia precisione (64 bit)*
- **long double** *lungo doppia precisione (80 bit)*
- *I numeri reali possono avere diverse sintassi*

24.0

2.4E1

240.0E-1

- *Nelle stringhe formato (**printf**, **scanf**) si usa
 %f (singola precisione)
 o **%lf** (double e long double)*



07 Espressioni

07.03 - Il tipo di dato carattere

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

QUESTO MATERIALE DIDATTICO È PER USO PERSONALE DELLO STUDENTE ED È
COPERTO DA COPYRIGHT. NE È SEVERAMENTE VIETATA LA RIPRODUZIONE O IL
RIUTILIZZO ANCHE PARZIALE, AI SENSI E PER GLI EFFETTI DELLA LEGGE SUL
DIRITTO D'AUTORE.

Caratteri

- *Per rappresentare caratteri in linguaggio C si usa il tipo di dato **char**.*
- *Es: **char c;***
- *Una variabile di tipo **char** contiene esattamente un carattere*

Interi con segno		
char	-128 .. 127	8 bit
short (int)	-32768 .. 32767	16 bit
int	???	16 o 32 bit
long (int)	-2147483648 .. 2147483647	32 bit (a volte 64)
long long (int)	-9223372036854775808 .. 9223372036854775807	64 bit

Caratteri

char (caratteri)

- Sono rappresentati *internamente* con 1 byte
 - con segno (*char*)
 - senza segno (*unsigned char*)
- Vi si possono applicare gli stessi operatori per gli interi (+, -, *, ..., <, <=, ...)
- *Esternamente* possono essere interpretati
 - come *numeri interi* (-128..127 o 0..255), usando il codice *%d* nelle stringhe formato
 - come *caratteri* (codice ASCII), usando il codice *%c* nelle stringhe formato

ASCII

American Standard Code for Information Interchange

Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex
(nul)	0	0x00	(sub)	26	0x1a	4	52	0x34	N	78	0x4e	h	104	0x68
(soh)	1	0x01	(esc)	27	0x1b	5	53	0x35	O	79	0x4f	i	105	0x69
(stx)	2	0x02	(fs)	28	0x1c	6	54	0x36	P	80	0x50	j	106	0x6a
(etx)	3	0x03	(gs)	29	0x1d	7	55	0x37	Q	81	0x51	k	107	0x6b
(eot)	4	0x04	(rs)	30	0x1e	8	56	0x38	R	82	0x52	l	108	0x6c
(enq)	5	0x05	(us)	31	0x1f	9	57	0x39	S	83	0x53	m	109	0x6d
(ack)	6	0x06	(sp)	32	0x20	:	58	0x3a	T	84	0x54	n	110	0x6e
(bel)	7	0x07	!	33	0x21	;	59	0x3b	U	85	0x55	o	111	0x6f
(bs)	8	0x08	"	34	0x22	<	60	0x3c	V	86	0x56	p	112	0x70
(ht)	9	0x09	#	35	0x23	=	61	0x3d	W	87	0x57	q	113	0x71
(nl)	10	0x0a	\$	36	0x24	>	62	0x3e	X	88	0x58	r	114	0x72
(vt)	11	0x0b	%	37	0x25	?	63	0x3f	Y	89	0x59	s	115	0x73
(np)	12	0x0c	&	38	0x26	@	64	0x40	Z	90	0x5a	t	116	0x74
(cr)	13	0x0d	'	39	0x27	A	65	0x41	[91	0x5b	u	117	0x75
(so)	14	0x0e	(40	0x28	B	66	0x42	\	92	0x5c	v	118	0x76
(si)	15	0x0f)	41	0x29	C	67	0x43]	93	0x5d	w	119	0x77
(dle)	16	0x10	*	42	0x2a	D	68	0x44	^	94	0x5e	x	120	0x78
(dc1)	17	0x11	+	43	0x2b	E	69	0x45	_	95	0x5f	y	121	0x79
(dc2)	18	0x12	,	44	0x2c	F	70	0x46	`	96	0x60	z	122	0x7a
(dc3)	19	0x13	-	45	0x2d	G	71	0x47	a	97	0x61	{	123	0x7b
(dc4)	20	0x14	.	46	0x2e	H	72	0x48	b	98	0x62		124	0x7c
(nak)	21	0x15	/	47	0x2f	I	73	0x49	c	99	0x63	}	125	0x7d
(syn)	22	0x16	0	48	0x30	J	74	0x4a	d	100	0x64	~	126	0x7e
(etb)	23	0x17	1	49	0x31	K	75	0x4b	e	101	0x65	(del)	127	0x7f
(can)	24	0x18	2	50	0x32	L	76	0x4c	f	102	0x66			
(em)	25	0x19	3	51	0x33	M	77	0x4d	g	103	0x67			

Costanti di tipo char

- **Sintassi delle costanti**

- *singolo carattere racchiuso fra apici 'A' , '?' , '6'*
- *caratteri speciali:*

`'\n'` `'\t'` `'\''` `'\\'` `'\"'`

- **Si possono usare**
costanti numeriche
usate per gli interi

```
char c;  
c = 82;  
if (c < 65)  
    ...
```

- **Oppure costanti di**
tipo carattere:

```
char c;  
c = 'R';  
if (c < 'A')  
    ...
```

Esempio

```
#include <stdio.h>

main()

{ char C;

  scanf ("%c", &C) ;

  printf ("Il codice ASCII di %c ", C) ;

  printf ("e` %d\n", C) ;

}
```

Esercizio

- *Si legga da tastiera un carattere c*
- *Se c è una lettera minuscola, si visualizzi “minuscola”,
se è maiuscola si visualizzi “maiuscola”,
se è un numero si visualizzi “numero”*

Esercizio

- *Quanti errori ci sono in questo programma?*

```
main()  
{ char a='c';  
  char b;  
  b=a;  
  b='ciao';  
  b='\n';  
  a=64;  
  printf("%d %c",a,b);  
}
```



07 Espressioni

07.04 - Espressioni eterogenee

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

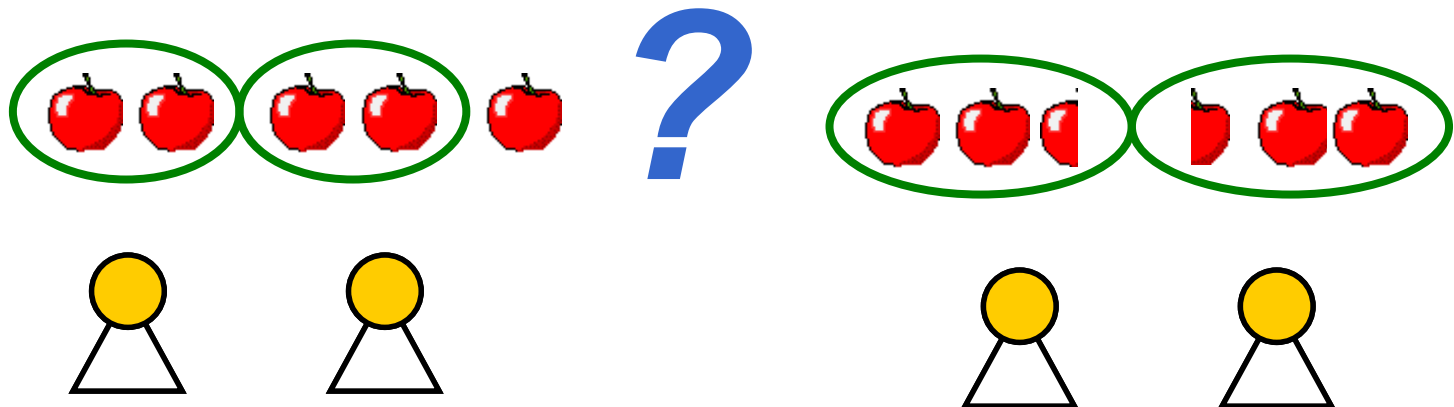
QUESTO MATERIALE DIDATTICO È PER USO PERSONALE DELLO STUDENTE ED È
COPERTO DA COPYRIGHT. NE È SEVERAMENTE VIETATA LA RIPRODUZIONE O IL
RIUTILIZZO ANCHE PARZIALE, AI SENSI E PER GLI EFFETTI DELLA LEGGE SUL
DIRITTO D'AUTORE.

Media di due valori

```
main()  
{ int a, b;  
  float media;  
  a=1;  
  b=2;  
  media = (a+b)/2;  
  printf("media=%f", media);  
}
```


OPERATORI: OVERLOADING

- *In C (come in Pascal, Fortran e molti altri linguaggi) operazioni primitive associate a tipi diversi possono essere denotate con lo stesso simbolo (ad esempio, le operazioni aritmetiche su reali o interi).*
- *In realtà l'operazione è diversa e può produrre risultati diversi. Ad esempio, lo stesso simbolo **/** viene usato per la divisione fra interi e quella fra reali*



OPERATORI: OVERLOADING

- Il C stabilisce quale delle operazioni effettuare *in base al tipo degli operandi*.
- Quindi, nel caso della divisione,
 - se gli operandi sono interi, effettua la divisione intera
 - se gli operandi sono reali, effettua la divisione fra reali.

```
int X,Y;  
se X = 10 e Y = 4;  
X/Y vale...
```

```
float X,Y;  
se X = 10.0 e Y = 4.0;  
X/Y vale...
```

- e se sono di tipo diverso?

```
int X; float Y;  
se X = 10 e Y = 4.0;  
X/Y vale...
```




CONVERSIONI DI TIPO

- *In C è possibile combinare tra di loro operandi di tipo diverso:*
 - espressioni **omogenee**: tutti gli operandi sono dello stesso tipo
 - espressioni **eterogenee**: gli operandi sono di tipi diversi.
- **Regola adottata in C:**
 - sono eseguibili le espressioni eterogenee in cui tutti i tipi referenziati risultano **compatibili** (cioè: dopo l'applicazione della regola automatica di conversione implicita di tipo del C risultano omogenei).

Regole di conversione implicita

- *Data una espressione x op y.*
 1. *Ogni operando di tipo **char** o **short** viene convertito nel tipo **int**;*
 2. *Se dopo l'esecuzione del passo 1 l'espressione è ancora eterogenea, rispetto alla seguente gerarchia*
int < long < float < double < long double
*si converte temporaneamente l'operando di tipo inferiore al tipo superiore (**promotion**);*
 3. *A questo punto l'espressione è **omogenea** e viene eseguita l'operazione specificata. Il risultato è di tipo uguale a quello prodotto dall'operatore effettivamente eseguito. (In caso di **overloading**, quello più alto gerarchicamente).*

CONVERSIONI DI TIPO

`int i` 
`char c` 
`double r` 

Valutare l'espressione
 $(i+c) / r$

Passo 1: $(i+c)$

`int c'` 

–Viene creata una nuova variabile `c'` di tipo `int` e le viene assegnato il valore di `c` convertito in `int`

`int tmp` 

–viene applicata la somma tra interi
–risultato intero `tmp`

Passo 2 (tmp / r)

`double tmp'` 

–Viene creata una nuova variabile `tmp'` di tipo `double` e le viene assegnato il valore di `tmp` convertito in `double`

`double tmp2` 

–viene applicata la divisione tra `double`
–risultato `double`



07 Espressioni

07.05 - Assegnamenti eterogenei

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

QUESTO MATERIALE DIDATTICO È PER USO PERSONALE DELLO STUDENTE ED È COPERTO DA COPYRIGHT. NE È SEVERAMENTE VIETATA LA RIPRODUZIONE O IL RIUTILIZZO ANCHE PARZIALE, AI SENSI E PER GLI EFFETTI DELLA LEGGE SUL DIRITTO D'AUTORE.

CONVERSIONI NEGLI ASSEGNAIMENTI

- *In un assegnamento, la variabile e l'espressione devono essere dello stesso tipo.*
- *Nel caso di tipi diversi:*
 - *se i tipi sono incompatibili → errore a tempo di compilazione*
 - *se possibile si effettua la conversione implicita*
 - *in alcuni casi, l'assegnamento può generare perdita di informazione → warning*

```
int i;  
char c;  
double r;  
i = c;      /* char -> int */  
i = c+i;  
r = c;      /* char -> int -> double */  
i = r;      /* troncamento */
```

Esempio

```
main()  
{  
    int a = 5;  
    float f = 6.6;  
    char c = 4;  
    c--;  
    f = a+f;  
    a = a % c;  
    a = f * c;  
}
```

a	5
f	6.6
c	4

Esempio

```
main()  
{ int a = 5;  
  float f = 6.6;  
  char c = 4;  
  c--;  
  f = a+f;  
  a = a % c;  
  a = f * c;  
}
```

a	5
f	6.6
c	3

Esempio

```
main()  
{ int a = 5;  
  float f = 6.6;  
  char c = 4;  
  c--;  
  f = a+f;  
  a = a % c;  
  a = f * c;  
}
```

a	5
f	11.6
c	3

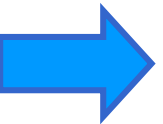
- si rende omogenea l'espressione:
- a viene promosso a float: 5.0
- si calcola a+f: 5.0+6.6=11.6
- questo valore viene assegnato a f

int < long < float < double < long double

Esempio

```
main()  
{ int a = 5;  
  float f = 6.6;  
  char c = 4;  
  
  c--;  
  
  f = a+f;  
  
  a = a % c;  
  a = f * c;  
  
}
```

a	2
f	11.6
c	3



int < long < float < double < long double

Esempio

```
main()  
{ int a = 5;  
  float f = 6.6;  
  char c = 4;  
  c--;  
  f = a+f;  
  a = a % c;  
  a = f * c;  
}
```

a	34
f	11.6
c	3

- si rende omogenea l'espressione:
- c viene promosso a float: 3.0
- si calcola $c * f$: $3 * 11.6 = 34.8$
- questo valore viene trasformato in int (troncato)
- assegnato ad a

`int < long < float < double < long double`

CONVERSIONE IMPLICITA

- *In generale, negli **assegnamenti** sono automatiche le conversioni di tipo.*
- *Espressioni che possono provocare perdita di informazioni non sono illegali (generano solo un warning a tempo di compilazione)*

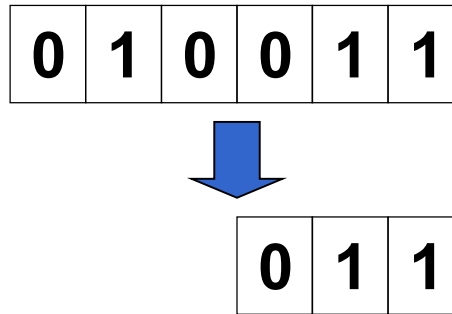
CONVERSIONE IMPLICITA

- *Conversioni che non provocano perdita di informazione*
`short -> int, int -> long,`
`float -> double, double -> long double`
- *Conversioni che possono portare a perdita di informazione*
 - ***A causa del fatto che gli estremi del tipo da convertire sono esterni a quelli del nuovo tipo***
 - ***A causa del fatto che il numero di cifre decimali rappresentabili nel tipo da convertire sono maggiori di quelle nuovo tipo***

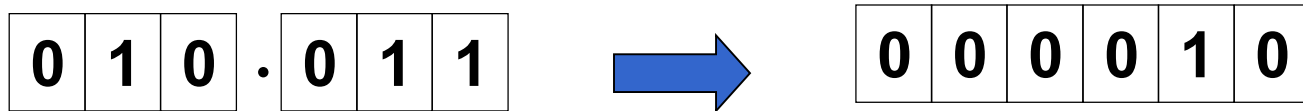
CONVERSIONE IMPLICITA

- A causa del fatto che gli estremi del tipo da convertire sono esterni a quelli del nuovo tipo:

- intero con n bit → intero con meno di n bit



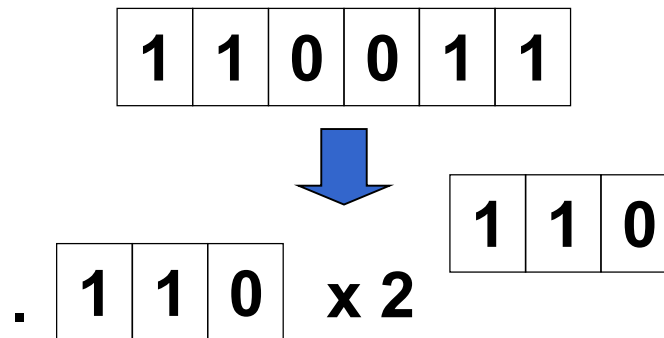
- float con n bit → float con meno di n bit
- float → intero.



- Se il numero da convertire non sta nella dimensione del nuovo tipo il risultato è imprevedibile.

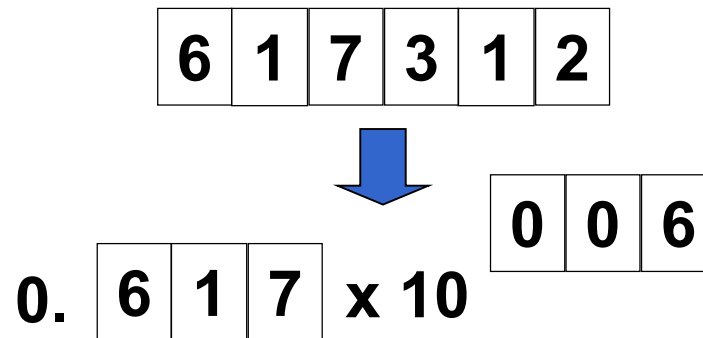
CONVERSIONE IMPLICITA

- A causa del fatto che il numero di cifre decimali rappresentabili nel tipo da convertire sono maggiori di quelle nuovo tipo: conversioni da tipo intero a tipo float se il numero di bit dell'intero sono maggiori di quelli riservati alla mantissa nel tipo float. Ad esempio, se int è 32 bit, int->float può causare perdita di informazioni. In questo caso vengono perse le cifre meno significative.***



CONVERSIONE IMPLICITA

- A causa del fatto che il numero di cifre decimali rappresentabili nel tipo da convertire sono maggiori di quelle nuovo tipo: conversioni da tipo intero a tipo float se il numero di bit dell'intero sono maggiori di quelli riservati alla mantissa nel tipo float. Ad esempio, se int è 32 bit, int->float può causare perdita di informazioni. In questo caso vengono perse le cifre meno significative.***



CONVERSIONE IMPLICITA

- *ESEMPI DI POSSIBILE PERDITA DI INFORMAZIONE*

```
int i;
```

```
float f,f2;
```

```
double d;
```

```
f = i;
```

```
f2= f + i;
```

```
f = d;
```

```
/* int -> float  
possibile perdita  
di cifre significative */
```

```
/* double -> float  
possibilita' di  
risultati imprevedibili */
```



07 Espressioni

07.06 - Type casting

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

QUESTO MATERIALE DIDATTICO È PER USO PERSONALE DELLO STUDENTE ED È
COPERTO DA COPYRIGHT. NE È SEVERAMENTE VIETATA LA RIPRODUZIONE O IL
RIUTILIZZO ANCHE PARZIALE, AI SENSI E PER GLI EFFETTI DELLA LEGGE SUL
DIRITTO D'AUTORE.

CONVERSIONE ESPLICITA DI TIPO: L' OPERATORE DI CAST

- In qualunque espressione è possibile forzare una particolare conversione utilizzando l'operatore di cast:*

(<tipo>) <espressione>

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

```
...
```

```
int i;
```

```
float f;
```

```
f = 10.3/2;
```

```
i = (int) f+2;
```

```
i = (int) sqrt(384) ;
```

L'operatore di cast
evita i warning

CONVERSIONE ESPLICITA DI TIPO: L' OPERATORE DI CAST

- *In qualunque espressione è possibile forzare una particolare conversione utilizzando l'operatore di cast:*

(<tipo>) <espressione>

```
int i;
```

```
float f;
```

```
i = (int) f % 2;
```

CONVERSIONE ESPLICITA DI TIPO: L' OPERATORE DI CAST

- In qualunque espressione è possibile forzare una particolare conversione utilizzando l'operatore di cast:*

(<tipo>) <espressione>

```
int    i;
```

```
long L;
```

```
i = 65536;    // 2^16
```

```
L = i*i;      // 2^32, troppo grande per int a 32 bit
```

```
L = ((long)i) * i; // risultato corretto
```



07 Espressioni

07.08 - Espressioni concatenate Assegnamento

Corso di Laurea in Ingegneria Elettronica e Informatica

Anno accademico 2020/2021

Prof. MARCO GAVANELLI

QUESTO MATERIALE DIDATTICO È PER USO PERSONALE DELLO STUDENTE ED È
COPERTO DA COPYRIGHT. NE È SEVERAMENTE VIETATA LA RIPRODUZIONE O IL
RIUTILIZZO ANCHE PARZIALE, AI SENSI E PER GLI EFFETTI DELLA LEGGE SUL
DIRITTO D'AUTORE.

ESPRESSIONI CONCATENATE

Una espressione concatenata è introdotta dall'*operatore di concatenazione (la virgola)*

espr1, espr2, ..., esprN

- *tutte le espressioni vengono valutate (da sinistra a destra)*
- *l'espressione esprime il valore denotato da **esprN***
- *Supponiamo che **i=5** e **k=7**, allora l'espressione:*

i + 1, k - 4

*denota il valore denotato da **k-4**, cioè 3.*

Espressioni concatenate: trabocchetto

- *Attenzione a non mettere la virgola invece di altri simboli!*

- *Volevo scrivere* *ma ho scritto*

a = 2 (10+2.3) ; a = 2* (10+2,3) ;*

*per il compilatore va **benissimo**: non mi dà errore!*

- *Risultato: a=6*

Trabocchetto: assegnamento

- *Per il C anche l'assegnamento è un operatore, che può comparire in un'espressione.*

var = espressione

ha come risultato il valore dell'espressione. Questo è stato pensato per scrivere assegnamenti multipli:

x = y = 3;

- *Attenzione: a volte capita di sbagliarsi ed usare l'assegnamento = invece del confronto ==*
- *Volevo scrivere*

if (a==0) printf("zero");

invece ho scritto

if (a=0) printf("zero");

per il C è sintatticamente corretto, ma fa una cosa completamente diversa da quello che volevo io!

Riassunto operatori del C

Priorità	Operatore	Simbolo	Associatività
1 (max)	Chiamate a funzioni Selezioni	() [] . ->	Sinistra
2	Operatori unari <ul style="list-style-type: none">• negazione• aritmetici unari• incr. / decr.• indirezione / dereferenziazione• sizeof• type cast	! + - ++ -- * & sizeof() (type)	Destra
3	moltiplicativi	* / %	Sinistra
4	additivi	+ -	Sinistra

RIASSUNTO OPERATORI DEL C

Priorità	Operatore	Simbolo	Associatività
5	op. di shift	>> <<	a sinistra
6	op. relazionali	< <= > >=	a sinistra
7	op. uguaglianza	== !=	a sinistra
8	op. di AND bit a bit	&	a sinistra
9	op. di XOR bit a bit	^	a sinistra
10	op. di OR bit a bit		a sinistra
11	op. di AND logico	&&	a sinistra
12	op. di OR logico		a sinistra
13	op. condizionale	? . . . :	a destra
14	op. assegnamento e sue varianti	= += -= *= /= %= &= ^= = <<= >>=	a destra
15 (min)	op. concatenazione	,	a sinistra