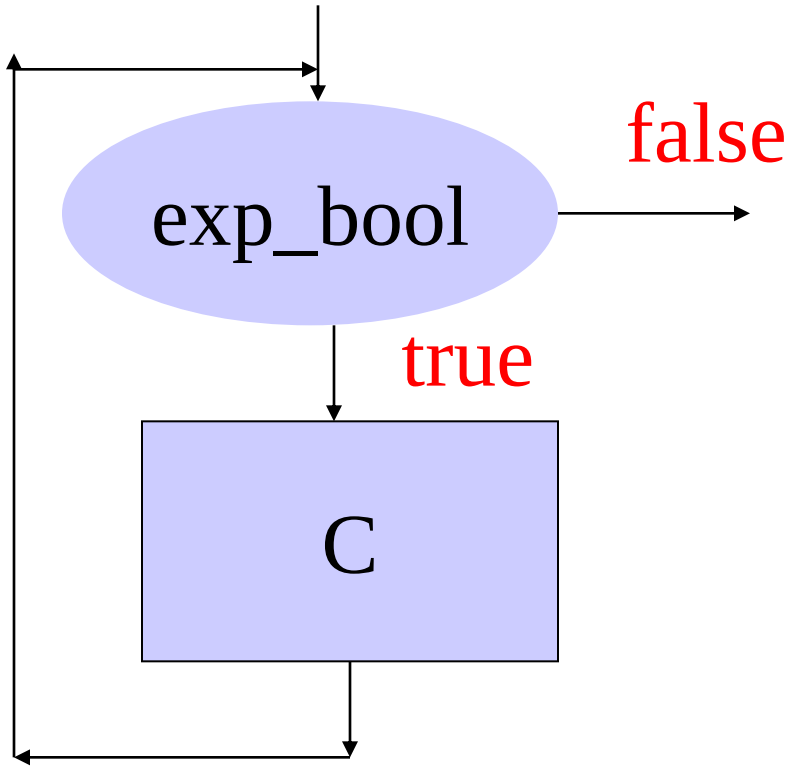
Sintassi: istruzioni del C++

- assegnazione: assegna un valore ad una variabile
- istruzione condizionale: esegue un confronto e a seconda del risultato fa cose diverse
- iterazione: ripete alcune istruzioni fino a quando una condizione è verificata

ciclo while



1 punto d’entrata ed 1 d’uscita

esempio di while:

```
int x=0;
while(x < 10)      {0<= x <=10} sempre vero
  {                 {0<= x <10}
     x=x+1;
  }                 {0<= x <=10} &&{ x >=10}
cout << x;  // quanto vale x?
```

```
#include<iostream>
using namespace std;
main()
{
  int somma=0, x, m=1;
  cin >> x;
  while (x!=-1)
  {
    somma=somma+x;
    m=m+1;
    cin >> x;
  }
}
```

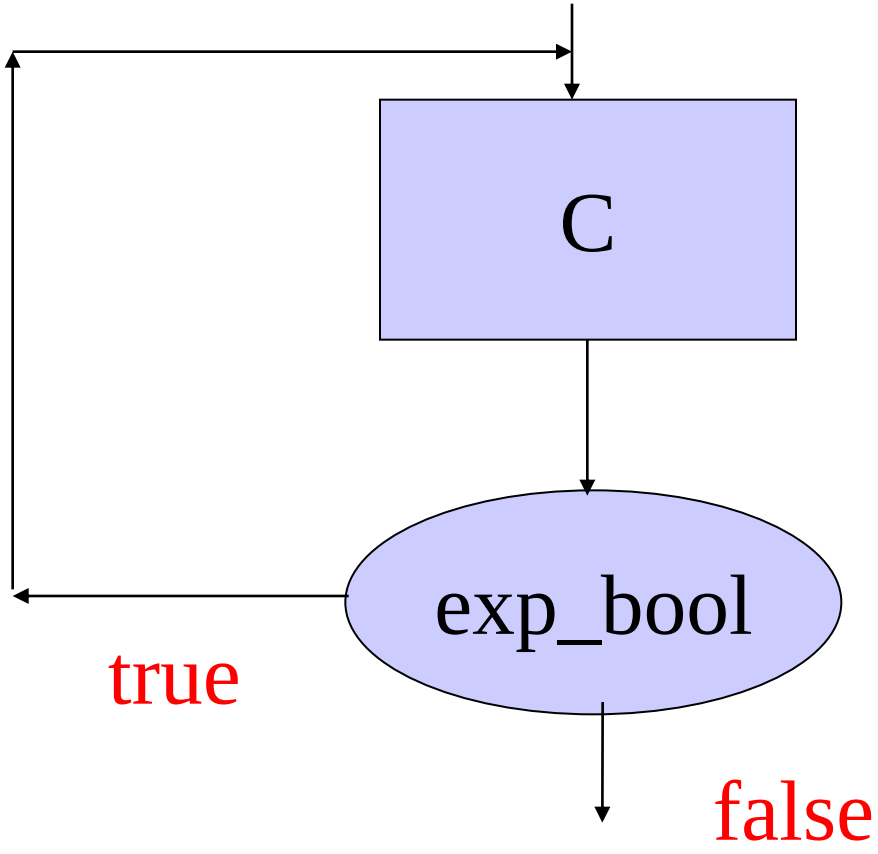
INVARIANTE

letti i primi m valori da cin, x contiene l'ultimo

i primi m-1 sono != -1 e somma è la somma di questi m-1 valori

nota che resta vero anche se il -1 non venisse mai letto

do-while: prima C e poi il test



1 punto d'entrata ed 1 d'uscita

esempio di do_while:

```
int x=0;
do      {0<= x < 10}
x=x+1;  {0< x <= 10}
while(x<10);  {0< x <= 10} && {x >=10}
cout << x;    {x = 10}
```

variabili, espressioni e conversioni

leggere il testo

Tutti i Linguaggi di Programmazione usano variabili che hanno un tipo e che assumono valori di quel tipo e che possono cambiare durante l'esecuzione del programma

le variabili sono sequenze di caratteri che rispettano certe regole

in C++ le variabili

iniziano con un carattere alfabetico (minu. o maiu.) e il resto può contenere ancora caratteri alfabetici, numeri, e '_' (sottolineatura)

- la variabile x è diversa dalla variabile X

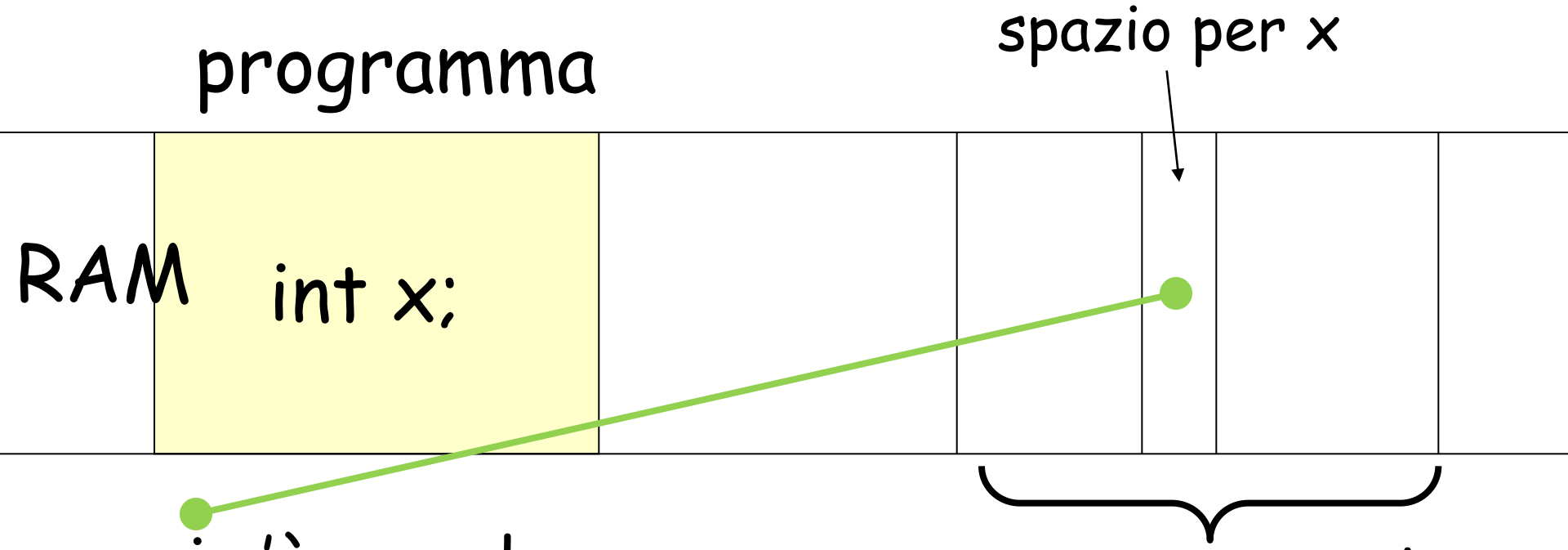
- le variabili devono essere distinte dalle parole chiave del C++

ogni variabile deve avere un tipo, cioè deve essere dichiarata prima dell'uso

- `int pippo;` // senza inizializzazione
- `int pippo=127;`
- `char pluto='a';`
- `float y=3.14f;`
- `bool ok=true;`

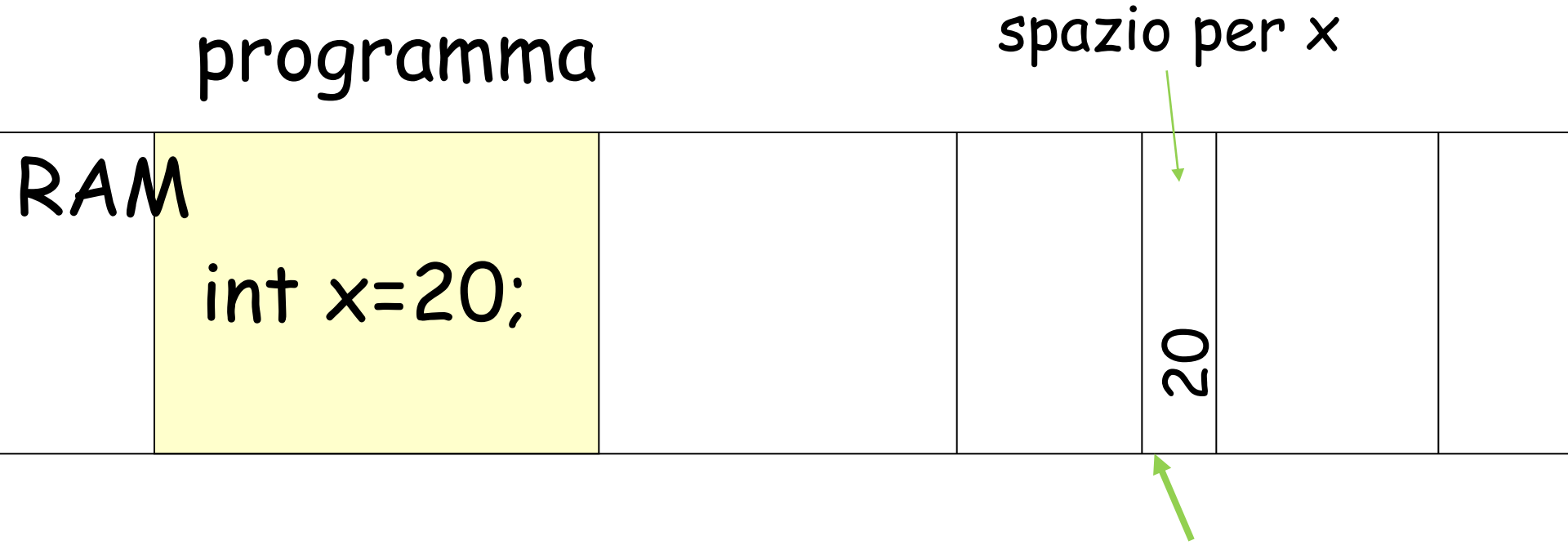
} con inizializzazione

quando viene eseguito un programma:



usare in qualsiasi modo un valore indefinito è un errore

assegnamo a x il valore 20

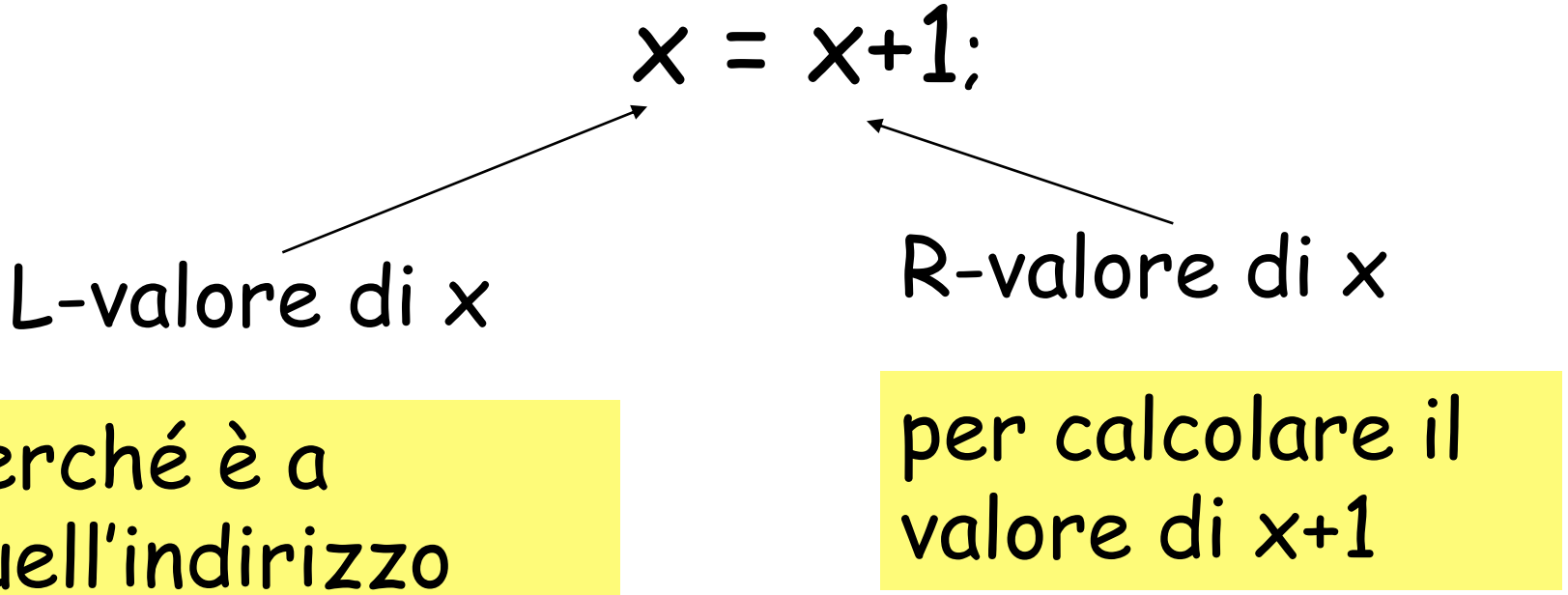


20 è l' R-valore di x mentre questo indirizzo RAM è il suo L-valore

ogni variabile ha:

-un R-valore che può essere definito se esso viene assegnato dal programma o altrimenti è indefinito

-un L-valore che è l'indirizzo RAM in cui si trova il suo R-valore


x = x+1;
L-valore di x
R-valore di x
perché è a quell'indirizzo che devo mettere il valore di x+1
per calcolare il valore di x+1

-non è possibile decidere l'L-valore delle nostre variabili, esso è stabilito dal SW che esegue il programma,

-però possiamo conoscerlo da programma: l'L-valore della variabile X è il valore dell'espressione &X

COSTANTI

```
const int pippo=10;
const float pi_greco= 3.14f;
```

- vanno inizializzate SEMPRE al momento della dichiarazione

- non esistono: il compilatore le sostituisce immediatamente col valore di inizializzazione

espressioni

nei programmi usiamo espressioni come

a + 12 * bib

a e bib sono variabili e 12 è un valore intero

- un'espressione va valutata per ottenere un VALORE

- l'espressione è sensata solo se a e bib sono variabili di tipi compatibili (attenzione alla confusione tra tipi del C++)
- altrimenti la valutazione fallirebbe

controllo di sensatezza di un'espressione

il compilatore deve stabilire se un'espressione come a + 12 * bib è SENSATA o no

COME FA ??? usa i tipi di a e di bib (quello di 12 è int)

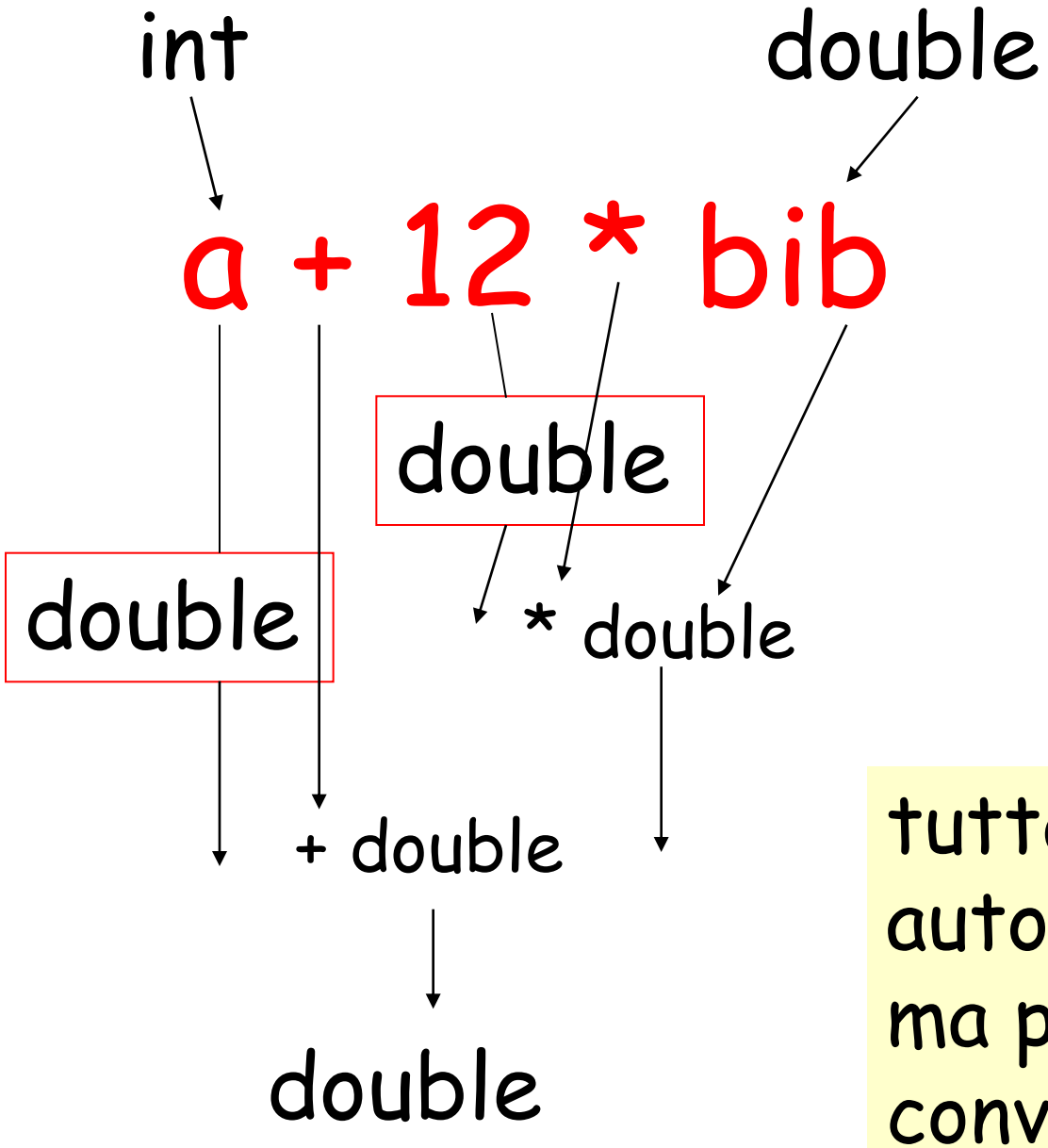
consideriamo alcuni casi possibili

a + 12 * bib

- a è int e bib è di tipo stringa : il compilatore dà errore di tipo

- a e bib sono entrambi int: è sensata e si usano + e * interi

- a è int e bib è double: il compilatore trasforma 12 in double, usa * per i double per (12*bib), poi trasforma l'R-valore di a in double e applica la + dei double per (a+...)



tutto avviene automaticamente, ma perché queste conversioni ??

consideriamo 12 * bib con bib: double

1) uno dei 2 operandi (12 e l'R-valore di bib) deve assumere il tipo dell'altro : esiste solo * tra interi o tra double o tra float, ma non tra un int e un double !!

2) quindi si potrebbe trasformare

R-valore di bib ➔ int oppure
12 ➔ double
e si sceglie questa seconda conversione

PERCHE' ??

principio: si operano automaticamente conversioni che non fanno perdere informazioni

int (4 byte) ➔ double (8 byte)

char ➔ int

bool ➔ int

int ➔ float (!!)

se l'espressione è sensata allora il compilatore produce codice che eseguito calcola il valore dell'espressione
a+12*bib con a:int e bib:double in

```
LOAD bib R0;
LOAD 12 R1
CONV_INT_DOUBLE  R1
MULT_DOUBLE  R0 R1
……
```

esempio:

2345 + 'a'

converte 'a' in int

si passa da 1 a 4 byte : è sicuro

il viceversa da 4 a 1 byte,
porterebbe 2345 % 256 = 41 = ')'

con evidente perdita di informazioni

(2345 + 'a') * 23.56

prima 'a' si converte in int

2345+97=2442

poi 2442 si converte in double

double usa 8 byte, int solo 4 : OK

risultato è di tipo double

MA che vuol dire convertire ? Dipende

char → int

'a' = | 97 | intero 97 in un byte

viene convertito nel valore intero:

0	0	0	97

che occupa quattro byte

la cosa si complica se il codice ASCII del carattere è un intero negativo:

per esempio 'è' = -24

come char è rappresentato dal byte

232

che in complemento a 2 rappresenta -24

convertirlo a int significa trasformarlo in -24 in complemento a 2 con 4 byte:

xxx	xxx	xxx	xxx

il valore contenuto nei 4 byte è = 4294927272 = 2^{32} -24

convertire int → float

int = 102101 → segno = 1 bit, e = 8, m = 23 bit

segno=0

e = 132 (-127)

m = 02101 (1. è implicito)

$2^5 * 1.02101$

int → double simile con il doppio dei bit

RIFLESSIONI

una visione a basso livello dei programmi

un programma è una sequenza di parole separate da spazi o simboli come ; ( ) [ ] + - / * = == eccetera

lessico

ogni parola appartiene ad una di queste categorie :

-parola chiave: main, while, for, int, true, …

-variabile: bib, pippo,…

-costante numerica:

- in base 10 : 12, 13.5, 1.4f
- in base 8: 012
- in base 16: 0x12

-costante carattere: 'a', 'w',…

-costante stringa alla C: "eccomi qui",…

una visione a medio livello dei programmi

sono fatti di istruzioni che hanno una forma sintattica ben precisa

iniziamo con le istruzioni di base

poi amplieremo pian piano il menù

sintassi

visione ad alto livello dei programmi

un programma calcola qualche funzione o compie un certo compito

semantica