# Laboratorio di informatica

## Ingegneria meccanica

Lezione 2 – 8 ottobre 2007



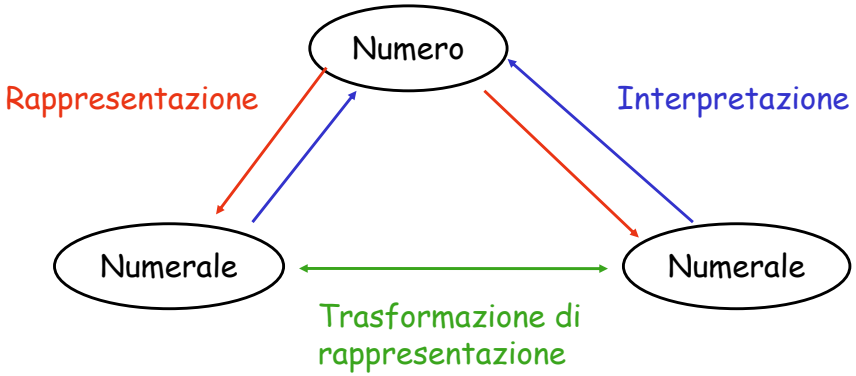
## Rappresentazione ed interpretazione


- Rappresentare un insieme di valori con sequenze di bit significa associare ad ogni valore una sequenza distinta
- Con sequenze di K bit si possono rappresentare $2^K$ valori. Per rappresentare M valori, è necessario usare $k >= \lceil \log_2 M \rceil$
- Interpretare una sequenza di bit significa ricostruire il valore che essa rappresenta



## Valori numerici: numeri e numerali

Numero: concetto astratto
Numerale: rappresentazione del numero (Esempi: "dodici", 12, XII, 0xC)






## Sistema di numerazione posizionale (1)

- Le cifre hanno un valore diverso a seconda della posizione che occupano nel numerale
- E' definito da una coppia (B, A) dove:
  - B è un intero > 1 detto base
  - A è un insieme di B simboli distinti detti cifre
- sistema decimale, B = 10, A = {0,1,2,3,4,5,6,7,8,9}
- binario, B = 2, A = {0,1}
- ottale, B = 8, A = {0,1,2,3,4,5,6,7}
- esadecimale, B = 16, A={0,1,…,9,A,B,…,F}
- Una singola cifra rappresenta univocamente un numero compreso fra 0 e B - 1

## Sistema di numerazione posizionale (2)

- In una sequenza di cifre, il valore rappresentato dalla cifra $d_K$ dipende dalla posizione K → valore($d_k$) = $d_k * B^k$
- In 100 (base 10), 1 rappresenta $1*10^2 = 100$
- Per numeri frazionari si usa anche una sequenza di cifre con peso negativo, il cui inizio è segnalato da un punto "."
- In 0.001 (base 10), 1 rappresenta $1*10^{-3} = 10^{-3}$
- Valore rappresentato da $d_n \ldots d_2 d_1 d_0 \, . \, d_{-1} d_{-2} \ldots d_{-m}$ è la somma di tutti i valori rappresentati dalle cifre $d_n B^n + d_{n-1} B^{n-1} + \ldots + d_0 + d_{-1} B^{-1} + \ldots + d_{-m} B^{-m}$



## Notazione

- La base B utilizzata in una rappresentazione R si indica tipicamente con $(R)_B$, con B in base 10
- Esempio: $(1011)_2$, $d_3 = 1$, $d_2 = 0$, $d_1 = 1$, $d_0 = 1$
  Valore = $2^3 + 2^1 + 2^0 = 8 + 2 + 1 = (11)_{10}$
- La cifra più a sinistra è detta cifra più significativa (most significant digit) e quella più a destra è detta cifra meno significativa (least significant digit)
- Se B = 2, si usano gli acronimi msb (most significant bit) ed lsb (least significant bit)
- Se B = 16, si usa il prefisso 0x per indicare la base
  Esempio: $0x721 = (721)_{16}$



## Esempi

| Numero | Base | Simboli | Rappresentazione |
|---|---|---|---|
| 15 | 2 | {0,1} | 1111 |
| 15 | 8 | {0,1,2,…,7} | 17 |
| 15 | 10 | {0,1,2,…9} | 15 |
| 15 | 16 | {0,…9,A,…,F} | F |



## Variabili in C

- Una variabile è (parzialmente) caratterizzata da:
  - tipo: specifica i *valori* che la variabile può assumere e le operazioni possibili su questi (consente di eseguire in fase di compilazione alcuni controlli sull'uso della variabile)
  - nome: individua la variabile (*fisicamente*, permette l'accesso al dato memorizzato che rappresenta il *valore* della variabile)
  - valore
- Prima di essere *usata* una variabile deve essere dichiarata, specificando tipo e nome (opzionalmente, si può specificare anche un valore iniziale)
- E' possibile impostare o modificare il valore di una variabile con un assegnamento

## Nome di una variabile in C

- Deve iniziare con una lettera [ _, a-z, A-Z ] e può seguitare con lettere e cifre (0-9) (c'è distinzione tra minuscole e maiuscole (case sensitiveness))
- Non può coincidere con una parola chiave (keyword)
- *Attenzione alla lunghezza*

| | | | |
|---|---|---|---|
| auto | enum | restrict | unsigned |
| break | extern | return | void |
| case | float | short | volatile |
| char | for | signed | while |
| const | goto | sizeof | _Bool |
| continue | if | static | _Complex |
| default | inline | struct | _Imaginary |
| do | int | switch | |
| double | long | typedef | |
| else | register | union | |



## Alcuni tipi in C

- char (caratteri)
- int (interi)
- float (reali, virgola mobile singola precisione)
- double (reali, virgola mobile doppia precisione)
- void ("nessun valore")
- E' possibile modificare alcune caratteristiche di alcuni tipi mediante i *qualificatori* short, long, signed e unsigned
- Esempio:
  long int (interi, intervallo rappresentato include quello rappresentato con int)



## Dichiarazione di variabili in C

Alcune forme

tipo nome ;

tipo nome = valore ; (inizializzazione)

tipo nome1 , nome2 , ... , nomeN ;

Esempi:

```
char   prefix ;
float  alpha , beta , gamma ;
int    cont ;
int    cont1 = 0 , cont2 = -2 , cont3 ;
```

*Normalmente,* non ci sono garanzie sul valore di una variabile dopo una sua dichiarazione senza specifica di un valore iniziale



## Assegnamento in C

- Una forma di assegnamento in C è
  nome = espressione ;
- nome specifica dove memorizzare il risultato della valutazione dell'espressione
- Un'espressione può essere:
  - semplice
    - una costante (p.es., contatore = -3 ; )
    - il valore di una variabile (p.es., inizio = fine ; )
    - il valore restituito da una funzione
  - composta
    - composizione di espressioni con operatori
    - Esempi: x = y + 6 ; x = y = 6

# C: operatori aritmetici

| Operazione | Operatore C | Esempio |
|---|---|---|
| Inversione segno | - | -15 |
| Somma | + | 12 + c |
| Differenza | - | 12 - 4<br>14 – 41 |
| Divisione | / | 7 / 2<br>2 / 7 |
| Moltiplicazione | * | a * ( - b ) |
| Modulo (resto) | % | 7 % 3 |



# C: operatori relazionali

- Permettono di *confrontare* due valori:
  - **>** **(maggiore)**
  - **<** **(minore)**
  - **>=** **(maggiore o uguale--non minore)**
  - **<=** **(minore o uguale--non maggiore)**
  - **==** **(uguale)**
  - **!=** **(diverso--non uguale)**
- Possono essere usati per formare espressioni:
- **x > y , x >= 2*y - 33 , x == y , s == 'g'**
- Un'espressione relazionale ha valore intero: **0**, se falsa, e **1**, se vera. E' possibile comporre espressioni. Es. **x == ( y <= 1 )**



# Input/Output

- Le operazioni di input fanno implicitamente riferimento ad un dispositivo *logico* detto stdin (standard input). *Fisicamente*, stdin corrisponde--di norma--alla tastiera
- Le operazioni di output fanno implicitamente riferimento ad un dispositivo *logico* detto stdout (standard output). *Fisicamente*, stdout corrisponde--di norma--al video




# Input con scanf

- scanf provoca l'acquisizione da stdin di una *sequenza di caratteri*, la loro interpretazione secondo il formato specificato, e la memorizzazione del valore corrispondente--opportunamente codificato--all'indirizzo specificato
- Esempio: scanf( "%d" , &varInt ) ;
  - "%d" *stringa di formato* che indica--tramite la *specifica di conversione* %d--che la sequenza di caratteri deve essere interpretata come un valore intero
  - &varInt fornisce l'indirizzo della variabile varInt tramite l'*operatore di indirizzo* &
- Sono possibili input *multipli*
- Esempio : scanf( "%d%f" , &vInt , &vFloat ) ;

## Output con printf (1)

- printf provoca la conversione del valore specificato--secondo il formato specificato--in una sequenza di caratteri che viene trasferita a stdout
- Esempio: printf( "%d" , varInt ) ;
  - "%d" *stringa di formato* che indica--tramite la *specifica di conversione* %d--che la sequenza di caratteri deve rappresentare un valore intero
  - variabile varInt fornisce il valore da convertire
- La stringa di formato può anche contenere *testo utile*
- Esempio: printf( "valore = %d" , varInt ) ;
- E' possibile produrre output *di solo testo utile*
- Esempio : printf( "Errore di tipo 6" ) ;



## Output con printf (2)

- Sono possibili output *multipli*
- Esempio : printf( "a = %d\nb = %f" , vInt , vFloat ) ;
- \n *newline* (cursore inizio riga successiva) è un esempio di *sequenza di escape*

| | |
|---|---|
| \a | *alert* (segnalazione visiva o sonora) |
| \b | *backspace* (cursore indietro di una posizione) |
| \r | *carriage return* (cursore inizio riga corrente) |
| \t | tabulazione orizzontale |
| \v | tabulazione verticale |
| \\ | *backslash* |
| \? | punto interrogativo |
| \' | singolo apice |
| \" | doppio apice |



## Strutture di controllo

- La formulazione di molti algoritmi richiede che il linguaggio di programmazione in uso offra la possibilità di modificare il flusso di esecuzione delle istruzioni in base al valore di un'espressione (modifiche condizionate)
- Strutture di selezione: ramificazione condizionata del flusso
- Strutture di iterazione: ripetizione condizionata di un blocco di istruzioni



## Strutture di controllo in C

- **Le istruzioni if, if else, switch permettono la scelta tra azioni alternative**
- **Le istruzioni while e for forniscono meccanismi per l'iterazione condizionata**
- **Blocco di istruzioni: sequenza di istruzioni delimitata da parentesi graffe**
  **{ istr1; istr2; …; istrN; }**
- **In caso di blocco di singola istruzione, le parentesi graffe possono essere omesse**
  **{ istr; } equivale a istr ;**

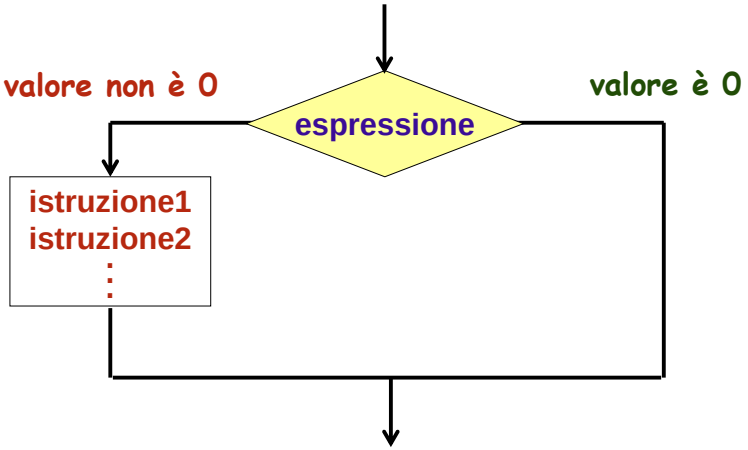
## Strutture di selezione in C

- **Selezione singola if**: viene eseguita o meno una singola azione
- **Selezione doppia if else**: viene eseguita una tra due azioni *logicamente* distinte (caso particolare: una delle due azioni è nulla → selezione singola)
- **Selezione multipla switch**: viene eseguita una tra un certo numero di azioni *logicamente* distinte
- Un'azione è formulata con un blocco di istruzioni
- La selezione è sempre in base al valore di un'espressione, distinguendo solo due casi: **zero** e **diverso da zero**



## C: Istruzione if

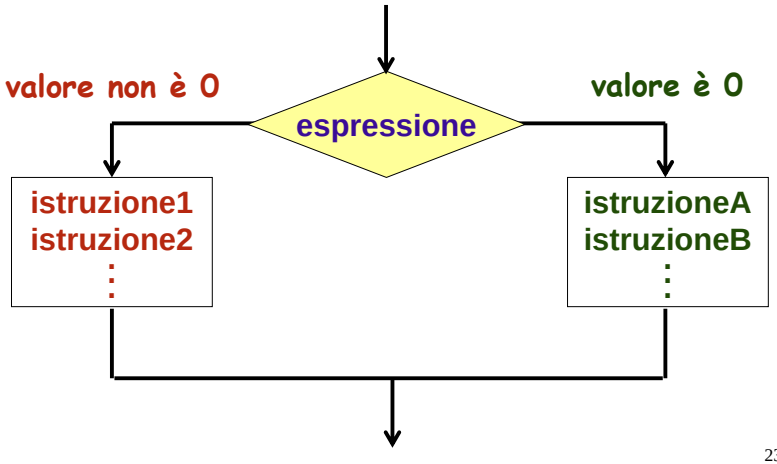
**if ( espressione ) { istruzione1 ; istruzione2 ; ... ; }**






## C: Istruzione if else

**if ( espressione ) { istruzione1 ; istruzione2 ; ... ; }**
**else { istruzioneA ; istruzioneB ; ... ; }**






## Esempio

Scrivere un programma che legga da stdin un numero intero e visualizzi su stdout un messaggio che dice se il numero letto è pari o dispari

Soluzione in *pseudocodice*

**leggi numero**
***se*** **numero è divisibile per 2**
**stampa "il numero è pari"**
***altrimenti***
**stampa "Il numero è dispari"**

## Esempio

```
#include <stdio.h>

int main ()
{
      int numero;
      printf("Inserisci un numero: ");
      scanf("%d", &numero);
      if (numero % 2 == 0)
           printf("%d e' un numero pari\n", numero);
      else
           printf("%d e' un numero dispari\n", numero);

      return 0;
}
```