# Laboratorio di Informatica

Ingegneria Meccanica
(canale 1)

Lezione 3, 15 ottobre 2007



## Conversione fra basi

- Problema: dato un numero rappresentato da $N_1$ in base $B_1$, trovarne la rappresentazione $N_2$ in base $B_2$ (trasformazione di rappresentazione)
- $(N_1)_{B_1} \rightarrow (N_2)_{B_2}$
- Nel seguito, se chiaro dal contesto, N denota sia il numero che il numerale



## Conversione alla base B (I)

- N numero da convertire (N > 0, intero)
- Per ogni intero D ($1 \leq D \leq N$), $N = Q \cdot D + R$ dove Q ($Q \leq N$) e R ($0 \leq R < D$) sono quoziente e resto della divisione fra gli interi N e D
- Notazione
  $Q = N/D$
  $R = N \bmod D$
- Nella nuova base B, N sarà rappresentato da
  $d_{K-1}B^{K-1} + \ldots + d_1B + d_0 = (d_{K-1}B^{K-2} + \ldots + d_1)\cdot B + d_0$
  e quindi $d_0 = N \bmod B$ e $d_{K-1}B^{K-2} + \ldots + d_1 = N/B$
- Ora, $d_{K-1}B^{K-2} + \ldots + d_1 = (d_{K-1}B^{K-3} + \ldots + d_2)\cdot B + d_1$ e quindi $d_1 = N' \bmod B$ e $d_{K-1}B^{K-3} + \ldots + d_2 = N'/B$ con $N' = N/B$ (per le altre cifre si procede analogamente)



## Conversione alla base B (II)

- Le cifre $d_0, d_1, \ldots, d_{K-1}$ così ottenute sono rappresentate nella base in cui si sono eseguite le operazioni (la stessa base di partenza in cui N è rappresentato)
- Per ottenere la rappresentazione di N in base B, le cifre $d_0, d_1, \ldots, d_{K-1}$ devono essere rappresentate con i simboli previsti per la base B (si veda il seguente Esempio 2)

# Regola di conversione

N intero, B base

Poni h = 0
Finché N ≠ 0 esegui
    $d_h$ = N mod B
    N = N/B
    h = h + 1
fine

**Esempio 1: $(25)_{10}$ = $(??)_2$**

| N | N/2 | N mod 2 | Cifra |
|---|---|---|---|
| 25 | 12 | 1 | $d_0 = 1$ |
| 12 | 6 | 0 | $d_1 = 0$ |
| 6 | 3 | 0 | $d_2 = 0$ |
| 3 | 1 | 1 | $d_3 = 1$ |
| 1 | 0 | 1 | $d_4 = 1$ |

$(25)_{10} = (11001)_2$



# Esempi

**Esempio 2: $(30)_{10}$ = $(??)_{16}$**

| N | N/16 | N mod 16 | Cifra |
|---|---|---|---|
| 30 | 1 | 14 | $d_0$ = E |
| 1 | 0 | 1 | $d_1 = 1$ |

$(30)_{10} = (1E)_{16}$ =0x1E

**Esempio 3: $(30)_{10}$ = $(??)_2$**

| N | N/2 | N mod 2 | Cifra |
|---|---|---|---|
| 30 | 15 | 0 | $d_0 = 0$ |
| 15 | 7 | 1 | $d_1 = 1$ |
| 7 | 3 | 1 | $d_2 = 1$ |
| 3 | 1 | 1 | $d_3 = 1$ |
| 1 | 0 | 1 | $d_4 = 1$ |

$(30)_{10} = (11110)_2$



# Conversioni fra basi speciali (I)

- Dato un numero rappresentato da $(n)_b$, trovarne la rappresentazione $(N)_B$ in base $B = b^m$ ($m > 0$, intero)
- $(n)_b \rightarrow (N)_B$
- In questo caso, le cifre in base B si ottengono immediatamente da quelle in base b come segue

  $N = d_{K-1}b^{K-1} + \ldots + d_1 b + d_0 = D_{H-1}B^{H-1} + \ldots + D_1 B + D_0$

  $d_{m-1}b^{m-1} + \ldots + d_1 b + d_0 = D_0$

  $d_{2m-1}b^{2m-1} + \ldots + d_{m+1}b^{m+1} + d_m b^m = D_1 B$

  etc.
- Partendo da $d_{m-1} \ldots d_1\ d_0$, per il numero corrispondente a ogni singolo gruppo di m cifre in $(n)_b$ si determina la cifra richiesta per rappresentarlo in base B



# Conversioni fra basi speciali (II)

- Si deve partire dal gruppo di m cifre meno significativo $(d_{m-1} \ldots d_1\ d_0)$
- Se il numero K di cifre in $(n)_b$ è multiplo di m, si può partire anche dal gruppo più significativo

## Conversioni fra basi speciali (III)

- **Esempio:** $b = 2$, $B = 8 = 2^3$
  $(1100011000101010110)_2 \rightarrow (??)_8$
  $d_2b^2+d_1b+d_0 = D_0$
  $d_5b^5+d_4b^4+d_3b^3 = D_1B$ ...

*verso di conversione* ←

1 100 011 000 101 010 110
1 4 3 0 5 2 6

- $(1100011000101010110)_2 \rightarrow (1430526)_8$



## Conversioni fra basi speciali (IV)

- Quanto detto per la conversione da base $b$ a base $B = b^m$ ($m > 0$, intero) è utilizzabile per risolvere il problema inverso
- $(N)_B \rightarrow (n)_b$
  $N = D_{H-1}B^{H-1}+ \dots +D_1B+D_0 = d_{K-1}b^{K-1}+ \dots +d_1b+d_0$
  $D_0 = d_{m-1}b^{m-1}+ \dots +d_1b+d_0$
  $D_1B = d_{2m-1}b^{2m-1}+ \dots +d_{m+1}b^{m+1}+d_mb^m$
  etc.
- Per il numero corrispondente a ogni singola cifra di $(N)_B$ si determinano le $m$ cifre richieste per rappresentarlo in base $b$ (si può partire da $D_0$ o da $D_{H-1}$ )



## Conversioni fra basi speciali (V)

- **Esempio:** $b = 2$, $B = 16 = 2^4$
  $(E134A6)_{16} \rightarrow (??)_2$
  $D_0 = d_3b^3 + d_2b^2 + d_1b + d_0$
  $D_1B = d_7b^7 + d_6b^6 + d_5b^5 + d_4b^4$
  ...

*verso di conversione* ↔

E 1 3 4 A 6
1110 0001 0011 0100 1010 0110

- $(E134A6)_{16} \rightarrow (1110\ 0001\ 0011\ 0100\ 1010\ 0110)_2$



## Conversioni fra basi speciali (VI)

- La tecnica rapida può essere applicata solo per conversioni tra basi del tipo detto ma può risultare utile anche per il passaggio dalla base $b^m$ alla base $b^n$
- In questo caso si può operare in due tempi, p. es., passando prima dalla base $b^m$ alla base $b$ e poi da questa alla base $b^n$
- Un caso di interesse è la conversione da base 16 ($2^4$) a base 8 ($2^3$) (o viceversa)
- Esempio
  $(FB0A9)_{16} \leftrightarrow (1111\ 1011\ 0000\ 1010\ 1001)_2$
  $(11\ 111\ 011\ 000\ 010\ 101\ 001)_2 \leftrightarrow (3730251)_8$

# Rappresentazione di caratteri in ASCII (I)

- Codice **ASCII** (American Standard Code for Information Interchange) definisce una rappresentazione su **7 bit** di **128 caratteri**
- *Tipicamente*, i 7 bit che definiscono un carattere sono *contenuti* nei 7 bit meno significativi di un byte (8 bit) (l'uso del bit più significativo non è standardizzato)
- Esempi usando un byte
  A → ?1000001, a → ?1100001
- Il codice di un carattere secondo ASCII (7 bit) è spesso interpretato come un numero in base 2, rappresentato poi in base 8, 16 o 10 (carattere rappresentato da un valore ottale, esadecimale o decimale)



# Rappresentazione di caratteri in ASCII (II)

- E' possibile suddividere i 128 caratteri ASCII in 4 gruppi (suddivisione seguente basata sulla **rappresentazione decimale** di un carattere):
  - Caratteri di controllo non stampabili (0-31)
  - Punteggiatura, spaziatura, simboli, cifre (32-64) (91-96) (123-127)
  - Caratteri alfabetici maiuscoli (65-90)
  - Caratteri alfabetici minuscoli (97-122)



# Rappresentazione di caratteri in ASCII (III)

- Con un byte, si possono rappresentare fino a 256 caratteri: esistono varie *estensioni* (*non standard*) del codice ASCII nelle quali sono rappresentati più di 128 caratteri (il valore del bit più significativo del byte di codice permette di distinguere se i rimanenti 7 bit identificano uno dei 128 carattere ASCII o un carattere aggiuntivo (*non standard*))
- Le due tabelle seguenti riportano un esempio di *ASCII esteso*: se il bit più significativo del byte di codice vale 0, allora i 7 bit rimanenti individuano un carattere ASCII
- *Oltre all'ASCII, esistono vari codici, anche standard, per la rappresentazione di caratteri*



# Esempio di ASCII esteso: caratteri ASCII

| Byte | Cod. | Char |
|---|---|---|
| 00000000 | 0 | Null |
| 00000001 | 1 | Start of heading |
| 00000010 | 2 | Start of text |
| 00000011 | 3 | End of text |
| 00000100 | 4 | End of transmit |
| 00000101 | 5 | Enquiry |
| 00000110 | 6 | Acknowledge |
| 00000111 | 7 | Audible bell |
| 00001000 | 8 | Backspace |
| 00001001 | 9 | Horizontal tab |
| 00001010 | 10 | Line feed |
| 00001011 | 11 | Vertical tab |
| 00001100 | 12 | Form Feed |
| 00001101 | 13 | Carriage return |
| 00001110 | 14 | Shift out |
| 00001111 | 15 | Shift in |
| 00010000 | 16 | Data link escape |
| 00010001 | 17 | Device control 1 |
| 00010010 | 18 | Device control 2 |
| 00010011 | 19 | Device control 3 |
| 00010100 | 20 | Device control 4 |
| 00010101 | 21 | Neg. acknowledge |
| 00010110 | 22 | Synchronous idle |
| 00010111 | 23 | End trans. block |
| 00011000 | 24 | Cancel |
| 00011001 | 25 | End of medium |
| 00011010 | 26 | Substitution |
| 00011011 | 27 | Escape |
| 00011100 | 28 | File separator |
| 00011101 | 29 | Group separator |
| 00011110 | 30 | Record Separator |
| 00011111 | 31 | Unit separator |

| Byte | Cod. | Char |
|---|---|---|
| 00100000 | 32 | Spc |
| 00100001 | 33 | ! |
| 00100010 | 34 | " |
| 00100011 | 35 | # |
| 00100100 | 36 | $ |
| 00100101 | 37 | % |
| 00100110 | 38 | & |
| 00100111 | 39 | ' |
| 00101000 | 40 | ( |
| 00101001 | 41 | ) |
| 00101010 | 42 | * |
| 00101011 | 43 | + |
| 00101100 | 44 | , |
| 00101101 | 45 | - |
| 00101110 | 46 | . |
| 00101111 | 47 | / |
| 00110000 | 48 | 0 |
| 00110001 | 49 | 1 |
| 00110010 | 50 | 2 |
| 00110011 | 51 | 3 |
| 00110100 | 52 | 4 |
| 00110101 | 53 | 5 |
| 00110110 | 54 | 6 |
| 00110111 | 55 | 7 |
| 00111000 | 56 | 8 |
| 00111001 | 57 | 9 |
| 00111010 | 58 | : |
| 00111011 | 59 | ; |
| 00111100 | 60 | < |
| 00111101 | 61 | = |
| 00111110 | 62 | > |
| 00111111 | 63 | ? |

| Byte | Cod. | Char |
|---|---|---|
| 01000000 | 64 | @ |
| 01000001 | 65 | A |
| 01000010 | 66 | B |
| 01000011 | 67 | C |
| 01000100 | 68 | D |
| 01000101 | 69 | E |
| 01000110 | 70 | F |
| 01000111 | 71 | G |
| 01001000 | 72 | H |
| 01001001 | 73 | I |
| 01001010 | 74 | J |
| 01001011 | 75 | K |
| 01001100 | 76 | L |
| 01001101 | 77 | M |
| 01001110 | 78 | N |
| 01001111 | 79 | O |
| 01010000 | 80 | P |
| 01010001 | 81 | Q |
| 01010010 | 82 | R |
| 01010011 | 83 | S |
| 01010100 | 84 | T |
| 01010101 | 85 | U |
| 01010110 | 86 | V |
| 01010111 | 87 | W |
| 01011000 | 88 | X |
| 01011001 | 89 | Y |
| 01011010 | 90 | Z |
| 01011011 | 91 | [ |
| 01011100 | 92 | \ |
| 01011101 | 93 | ] |
| 01011110 | 94 | ^ |
| 01011111 | 95 | _ |

| Byte | Cod. | Char |
|---|---|---|
| 01100000 | 96 | ` |
| 01100001 | 97 | a |
| 01100010 | 98 | b |
| 01100011 | 99 | c |
| 01100100 | 100 | d |
| 01100101 | 101 | e |
| 01100110 | 102 | f |
| 01100111 | 103 | g |
| 01101000 | 104 | h |
| 01101001 | 105 | i |
| 01101010 | 106 | j |
| 01101011 | 107 | k |
| 01101100 | 108 | l |
| 01101101 | 109 | m |
| 01101110 | 110 | n |
| 01101111 | 111 | o |
| 01110000 | 112 | p |
| 01110001 | 113 | q |
| 01110010 | 114 | r |
| 01110011 | 115 | s |
| 01110100 | 116 | t |
| 01110101 | 117 | u |
| 01110110 | 118 | v |
| 01110111 | 119 | w |
| 01111000 | 120 | x |
| 01111001 | 121 | y |
| 01111010 | 122 | z |
| 01111011 | 123 | { |
| 01111100 | 124 | \| |
| 01111101 | 125 | } |
| 01111110 | 126 | ~ |
| 01111111 | 127 | Del |

# Esempio di ASCII esteso: caratteri aggiuntivi

| Byte | Cod. | Char |
|---|---|---|
| 10000000 | 128 | Ç |
| 10000001 | 129 | ü |
| 10000010 | 130 | é |
| 10000011 | 131 | â |
| 10000100 | 132 | ä |
| 10000101 | 133 | à |
| 10000110 | 134 | å |
| 10000111 | 135 | ç |
| 10001000 | 136 | ê |
| 10001001 | 137 | ë |
| 10001010 | 138 | è |
| 10001011 | 139 | ï |
| 10001100 | 140 | î |
| 10001101 | 141 | ì |
| 10001110 | 142 | Ä |
| 10001111 | 143 | Å |
| 10010000 | 144 | É |
| 10010001 | 145 | æ |
| 10010010 | 146 | Æ |
| 10010011 | 147 | ô |
| 10010100 | 148 | ö |
| 10010101 | 149 | ò |
| 10010110 | 150 | û |
| 10010111 | 151 | ù |
| 10011000 | 152 | ÿ |
| 10011001 | 153 | Ö |
| 10011010 | 154 | Ü |
| 10011011 | 155 | ø |
| 10011100 | 156 | £ |
| 10011101 | 157 | Ø |
| 10011110 | 158 | × |
| 10011111 | 159 | ƒ |

| Byte | Cod. | Char |
|---|---|---|
| 10100000 | 160 | á |
| 10100001 | 161 | í |
| 10100010 | 162 | ó |
| 10100011 | 163 | ú |
| 10100100 | 164 | ñ |
| 10100101 | 165 | Ñ |
| 10100110 | 166 | ª |
| 10100111 | 167 | º |
| 10101000 | 168 | ¿ |
| 10101001 | 169 | ® |
| 10101010 | 170 | ¬ |
| 10101011 | 171 | ½ |
| 10101100 | 172 | ¼ |
| 10101101 | 173 | ¡ |
| 10101110 | 174 | « |
| 10101111 | 175 | » |
| 10110000 | 176 | ▒ |
| 10110001 | 177 | ▒ |
| 10110010 | 178 | ▒ |
| 10110011 | 179 | ▒ |
| 10110100 | 180 | ┤ |
| 10110101 | 181 | Á |
| 10110110 | 182 | Â |
| 10110111 | 183 | À |
| 10111000 | 184 | © |
| 10111001 | 185 | ╣ |
| 10111010 | 186 | ║ |
| 10111011 | 187 | ╗ |
| 10111100 | 188 | ╝ |
| 10111101 | 189 | ¢ |
| 10111110 | 190 | ¥ |
| 10111111 | 191 | ┐ |

| Byte | Cod. | Char |
|---|---|---|
| 11000000 | 192 | └ |
| 11000001 | 193 | ┴ |
| 11000010 | 194 | ┬ |
| 11000011 | 195 | ├ |
| 11000100 | 196 | ─ |
| 11000101 | 197 | ┼ |
| 11000110 | 198 | ã |
| 11000111 | 199 | Ã |
| 11001000 | 200 | ╚ |
| 11001001 | 201 | ╔ |
| 11001010 | 202 | ╩ |
| 11001011 | 203 | ╦ |
| 11001100 | 204 | ╠ |
| 11001101 | 205 | ═ |
| 11001110 | 206 | ╬ |
| 11001111 | 207 | ¤ |
| 11010000 | 208 | ð |
| 11010001 | 209 | Ð |
| 11010010 | 210 | Ê |
| 11010011 | 211 | Ë |
| 11010100 | 212 | È |
| 11010101 | 213 | ı |
| 11010110 | 214 | Í |
| 11010111 | 215 | Î |
| 11011000 | 216 | Ï |
| 11011001 | 217 | ┘ |
| 11011010 | 218 | ┌ |
| 11011011 | 219 | █ |
| 11011100 | 220 | ▄ |
| 11011101 | 221 | ¦ |
| 11011110 | 222 | Ì |
| 11011111 | 223 | ▀ |

| Byte | Cod. | Char |
|---|---|---|
| 11100000 | 224 | Ó |
| 11100001 | 225 | ß |
| 11100010 | 226 | Ô |
| 11100011 | 227 | Ò |
| 11100100 | 228 | õ |
| 11100101 | 229 | Õ |
| 11100110 | 230 | µ |
| 11100111 | 231 | þ |
| 11101000 | 232 | Þ |
| 11101001 | 233 | Ú |
| 11101010 | 234 | Û |
| 11101011 | 235 | Ù |
| 11101100 | 236 | ý |
| 11101101 | 237 | Ý |
| 11101110 | 238 | ¯ |
| 11101111 | 239 | ´ |
| 11110000 | 240 |  |
| 11110001 | 241 | ± |
| 11110010 | 242 | ‗ |
| 11110011 | 243 | ¾ |
| 11110100 | 244 | ¶ |
| 11110101 | 245 | § |
| 11110110 | 246 | ÷ |
| 11110111 | 247 | ¸ |
| 11111000 | 248 | ° |
| 11111001 | 249 | ¨ |
| 11111010 | 250 | · |
| 11111011 | 251 | ¹ |
| 11111100 | 252 | ³ |
| 11111101 | 253 | ² |
| 11111110 | 254 | ■ |
| 11111111 | 255 | |



# Tipo char in C (I)

- L'insieme dei valori che una variabile di tipo **char** può assumere è un intervallo **I** di interi (gli estremi dipendono dall'*implementazione* e sono definiti dalle costanti **CHAR_MIN** e **CHAR_MAX** nel file **limits.h**)
- L'intervallo **I** contiene gli interi corrispondenti al codice utilizzato per la codifica dei caratteri (*tipicamente*, **ASCII**)
- Un valore di tipo **char** è *tipicamente* memorizzato in **1 byte**, con **I ≡ [-128, 127]** o **I ≡ [0, 255]**
- Un valore di tipo **char** (**un singolo carattere**) può essere indicato *esplicitamente* (**tra singoli apici**) o con l'*intero corrispondente al suo codice* (p. es., se è usato codice ASCII, **'a'** e **97** rappresentano **a**)



# Tipo char in C (II)

- Esempio **char z = 'b' ;**
  dichiara **z** come **char** assegnandogli il codice della lettera **b** (se è usato il codice ASCII il valore assegnato a **z** sarà **98**)
- Esempio **char beta = 99 ;**
  dichiara **beta** come **char** assegnandogli il valore **99** (**beta** è associato al carattere **c** solo se è usato il codice ASCII (attenzione alla *portabilità*))
- Esempio **char cc = '\n' ;**
  dichiara **cc** come **char** assegnandogli il codice di **newline** (**'\n'** è un singolo carattere)
- **Specifica di conversione**: **%c** in **scanf** e in **printf**
- Operazioni: un valore di tipo **char** può essere usato in espressioni aritmetiche



# Stampa di un carattere e del suo codice

```
#include <stdio.h>
int main( )
{
  char   car ;
  printf( "Inserire un carattere\n" ) ;
  scanf( "%c" , &car ) ;
  printf( "Il codice di %c e\' %d\n" , car , car ) ;
  return 0 ;
}
```

# Operatori relazionali per interi in C (I)

Operatori binari (due operandi) per confronto ordinato: maggioranza (>), minoranza (<), maggioranza o uguaglianza (non minoranza) (>=), minoranza o uguaglianza (non maggioranza) (<=)

Operatori binari (due operandi) per confronto non ordinato: uguaglianza (==) (attenzione!), diversità (non uguaglianza) (!=)

Sono usati per formulare condizioni (*espressioni condizionali* o *logiche*) e producono un risultato binario (due possibili valori) intepretato come *valore logico*: 0(1) indica che, con gli operandi specificati, la condizione è falsa(vera)

Possono apparire in espressioni in cui appaiono operatori di altro tipo



# Operatori relazionali per interi in C (II)

**Esempi (assumendo `x , y , w , z` di tipo `int` e `x = 3 , y = -7 , w = 11` al tempo di valutazione delle espressioni seguenti)**

`z = (x == 3) ;` (1 assegnato a `z` poiché `x` vale 3)

`z = ((x + 3) <= (y + w)) ;` (0 assegnato a `z` poiché `x + 3` vale 6 e `y + w` vale 4)

`y != 5` (vale 1)

`x == (y +10)` (vale 1)

`(y + 10) == x` (vale 1 (confronto non ordinato!))

`x < w` (vale 1)

`w < x` (vale 0 (confronto ordinato!))



# Operatori logici (I)

- Sono usati per combinare costanti, variabili, espressioni logiche, aventi valore FALSO o VERO, e producono nuovi valori FALSO o VERO
- AND, OR: operatori binari (due operandi);
- NOT: operatore unario (un operando)
- X AND Y: produce valore VERO solo se X e Y hanno entrambi valore VERO
- X OR Y: produce valore FALSO solo se X e Y hanno entrambi valore FALSO
- NOT X: inverte valore di X
- Operatori e valori di questo tipo sono detti booleani (da Boole)



# Operatori logici (II)

| A | B | A AND B | A OR B | NOT A |
|---|---|---|---|---|
| FALSO | FALSO | FALSO | FALSO | VERO |
| FALSO | VERO | FALSO | VERO | |
| VERO | FALSO | FALSO | VERO | FALSO |
| VERO | VERO | VERO | VERO | |

- Proprietà di *associatività*:
- (X AND Y) AND Z = X AND (Y AND Z) = X AND Y AND Z
- (X OR Y) OR Z = X OR (Y OR Z) = X OR Y OR Z

# Operatori logici in C (I)

- Per le operazioni logiche di AND, OR e NOT sono disponibili, rispettivamente, gli operatori &&, || e !
- Gli operandi di un operatore logico vengono resi logici in questo modo: il valore 0 viene intepretato come FALSO, mentre un valore diverso da 0 (qui indicato con ≠0) viene interpretato come VERO
- &&, || e ! forniscono come risultato 0 (inteso come FALSO) o 1 (inteso come VERO)

| A | B | A && B | A \|\| B | !A |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | ≠0 | 0 | 1 | |
| ≠0 | 0 | 0 | 1 | 0 |
| ≠0 | ≠0 | 1 | 1 | |



# Operatori logici in C (II)

| A | B | A && B | A \|\| B | !A |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | ≠0 | 0 | 1 | |
| ≠0 | 0 | 0 | 1 | 0 |
| ≠0 | ≠0 | 1 | 1 | |

- La valutazione delle espressioni contenenti && e || obbedisce alle regole seguenti
- espressione1 && espressione2: Se espressione1 vale 0 (FALSO), espressione2 non viene valutata e l'espressione complessiva vale 0 (FALSO)
- espressione1 || espressione2: Se espressione1 vale ≠0 (VERO), espressione2 non viene valutata e l'espressione complessiva vale 1 (VERO)



# Esempi di espressioni logiche in C

Esempi (assumendo x , y , w , z di tipo int e x = 3 , y = -7 , w = 11 al tempo di valutazione delle espressioni seguenti)

```
z = ((x == 3) && (y == -7)) ;   (1 assegnato a z)
z = ((x != 2) || ((y + w) == 5)); (1 assegnato a z)
!w   (vale 0 poiché w non vale 0)
x == ((!y) +10)   (vale 0)
(y > 1) && z   (vale 0 (valore di z ininfluente))
(y < 1) || z   (vale 1 (valore di z ininfluente))
```



# Uso degli operatori logici

- Formulazione di condizioni
  Esempio: Le condizioni affinchè un triangolo di lati a, b, c risulti equilatero, isoscele, scaleno possono essere formulate come segue

equilatero: ( a = b ) AND ( a = c )

isoscele: (( a = b ) AND ( a ≠ c ))
OR (( a = c ) AND ( a ≠ b ))
OR (( b = c ) AND ( a ≠ c ))

scaleno: (a ≠ b) AND (a ≠ c) AND (b ≠ c)

## Uso degli operatori logici in C

- In C, le precedenti formulazioni assumono le forme

  equilatero: `( a == b ) && ( a == c )`

  isoscele:
  ```
  (( a == b ) && ( a != c ))
  || (( a == c ) && ( a != b ))
  || (( b == c ) && ( a != c ))
  ```

  scaleno: `(a != b) && (a != c) && (b != c)`



## Strutture if else nidificate

E' possibile ottenere ramificazioni complesse inserendo strutture if else nei blocchi istruzioni di altre strutture if else

Una scrittura *indentata* può facilitare la lettura del programma (semplifica *al programmatore* l'associazione di un else con il proprio if)

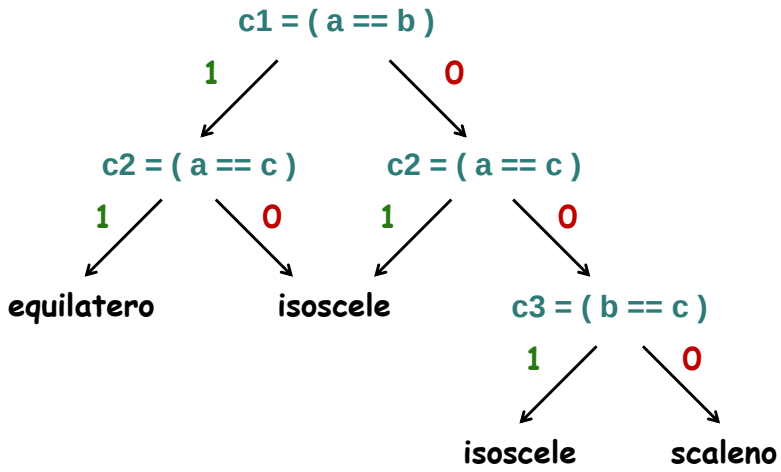


## Determinare il tipo di un triangolo (I)

```
...
c1 = ( a == b ) ;
c2 = ( a == c ) ;
c3 = ( b == c ) ;
if ( c1 )
{
  if ( c2 ) printf( "equilatero\n" ) ; /* c1 && c2 */
  else printf( "isoscele\n" ) ; /* c1 && !c2 */
}
else  /* !c1 */
{
  if ( c2 ) printf( "isoscele\n" ) ; /* !c1 && c2 */
  else if ( c3 ) printf( "isoscele\n" ) ; /* !c1 && !c2 && c3 */
       else printf( "scaleno\n" ) ; /* !c1 && !c2 && !c3 */
}
...
```



## Determinare il tipo di un triangolo (II)



# Determinare il tipo di un triangolo (III)

```
...
c1 = ( a == b ) ;
c2 = ( a == c ) ;
c3 = ( b == c ) ;
/* !c1 && !c2 && c3 equivale a !c2 && c3 */
if ( c1 && c2 )
     printf( "equilatero\n" ) ;
if (( c1 && !c2 ) || ( c2 && !c1 ) || ( c3 && !c2 ))
      printf( "isoscele\n" ) ;
If ( !c1 && !c2 && !c3 )
     printf( "scaleno\n" ) ;
...
```