Introduzione alla programmazione in C

DD Cap.2 pp.23-48
Cap.3 pp. 74-90
Cap. 4 pp. 112-116
Cap. 13 pp. 505-506

Un semplice programma C p.23

```
/* un primo programma C */
#include <stdio.h>
int main()
{
    printf("Primo programma in C");
    return 0;
}
```

commento
direttiva del preprocessore
funzione principale
istruzione di stampa
terminatore
indica che il programma è terminato con successo
rientro

E ora concentriamoci sul comando printf…

Cenni su printf()

- printf() è usata per l'output formattato
- non appartiene al C ma è presente nella libreria standard stdio.h
- la stringa da stampare è tra apici " "
- la stringa può contenere caratteri speciali:
  - \n invio
  - \t tabulazione
  - \" doppi apici
  - \\ backslash; eccetera
- la stringa può contenere anche caratteri di controllo per stampare le variabili (più avanti)

Cenni sul preprocessore (1) p. 505

Abbiamo detto che #include <stdio.h> è una direttiva del preprocessore.

Il preprocessore interviene prima della compilazione e alcune delle sue possibili azioni sono:

- l'inclusione di altri files in quello da compilare (#include)
- la definizione di costanti simboliche (#define).

Cenni sul preprocessore (2)

#include nome_file

indica al preprocessore di includere nel programma il contenuto del file nome_file.

nome_file può essere racchiuso tra <> (cerca nella directory dei files di intestazione della libreria standard) o tra "" (cerca nella directory corrente, se il file è stato scritto dal programmatore)

Alcuni files di intestazione della libreria standard:

<stdio.h> input/output

<math.h> funzioni matematiche

<string.h> funzioni su stringhe

<time.h> manipolazione data e ora

Cenni sul preprocessore (3)

#define identificatore testo_da_sostituire

Dal momento in cui compare questa direttiva tutte le occorrenze di identificatore vengono sostituite fisicamente con testo_da_sostituire.

Permette di creare le costanti simboliche, cioè di attribuire un nome a una costante ed usarlo in tutto il programma.

Qualora una costante debba essere modificata, si può cambiare solo testo_da_sostituire invece che tutte le sue occorrenze.

Esempio:

```
#define PI 3.14159
#define IPOTENUSA 20
#define CATETO 17
```

Attenzione: le costanti NON sono variabili, quindi non si può fare ad es.: IPOTENUSA=7;

Cenni sulla sintassi del C

- Alla fine di ogni istruzione va messo ;
- se l'istruzione è composta, cioè è costituita da più azioni (istruzioni), queste vanno racchiuse tra {}
- dopo la } non va il ;
- per aumentare la leggibilità tutti i programmi dovrebbero avere:
  - una indentazione chiara e coerente
  - dei commenti chiari ed esplicativi

Un altro semplice programma p.28

```
/* somma di due interi */
   int main()
   {
      int a=3;
      int b=4;
      int somma;

      somma=a+b;
      return 0;
   }
```

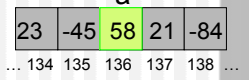
variabili intere

Ma che cos'è una variabile?

Variabili (1) p.29

In un programma C possiamo usare variabili per salvare dei valori che vogliamo usare.

in memoria:



Una variabile è un oggetto che ha:
- un nome (a e b sono nomi di variabili)
- un tipo (int rappresenta il tipo intero)
- un valore (3 è il valore di a e 4 di b)
- un indirizzo (locazione di memoria assegnata)

Prima di usare una variabile, cioè prima di assegnarle un valore, dobbiamo dichiararla, vale a dire dobbiamo fare in modo che il programma sappia che quella variabile esiste e che le assegni una locazione di memoria. Per dichiararla, dobbiamo specificarne il nome ed il tipo.

Esempio: `int a;` permette di dichiarare una variabile intera

Variabili (2)

int (intero), float (reale), char (carattere), ecc. sono parole chiave, cioè parole riservate dal C che non possono essere utilizzate altrimenti (ad esempio per chiamare le variabili).

La dichiarazione di tipo serve:
- per informare il compilatore su quanto spazio in memoria debba essere riservato per i valori associati alle variabili
- per "leggere" correttamente la codifica binaria della variabile
- per permettere al compilatore di scegliere le istruzioni macchina appropriate (ad es. somma di interi e di reali diversa)

Variabili (3)

Il nome della variabile viene detto identificatore.

Esso consiste di lettere o numeri.

Esempio: a, var1, num_tel, ecc.

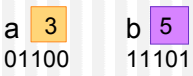
Attenzione:
- un identificatore non può iniziare con un numero o con _
- in un identificatore le maiuscole vengono distinte dalle minuscole, quindi a e A sono due variabili diverse
- un identificatore non può essere una parola chiave

Buona regola: assegnare alle variabili nomi significativi e non troppo lunghi (ad esempio, per una variabile che contiene la somma di altre, è meglio somma piuttosto che x, a, o pippo).

Variabili (4)

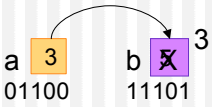
La lettura di una variabile NON comporta alcuna variazione in memoria.

Esempio: `b=a;`



per copiare a su b, devo prima leggere a…

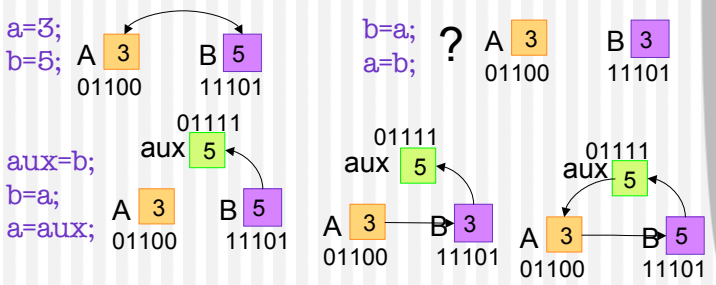
L'assegnamento di un valore ad una variabile implica la perdita dell'informazione contenuta precedentemente.



Variabili (4)

Esempio:

Supponiamo di avere 2 variabili e di doverne scambiare il contenuto:



L'aritmetica nel C (1) p.34

Spesso i programmi eseguono calcoli. Gli operatori aritmetici, tutti operatori binari, servono per eseguirli:

addizione +
sottrazione -
moltiplicazione *
divisione /
resto %

Attenzione a:

- La divisione tra interi restituisce un intero mentre la divisione tra reali restituisce un reale;
  Esempio: 10/4=2 ma 10/4.0=2.5
- la divisione per 0 non è definita!
- l'operatore resto può essere applicato solo ad operandi interi.

L'aritmetica nel C (2)

Si usano le parentesi per raggruppare i termini nelle espressioni (come nelle espr. algebriche, ma solo tonde!)

Il C valuta le espressioni seguendo le seguenti regole di priorità (come nelle espr. algebriche):

- Prima vengono moltiplicazione, divisione e resto, nell'ordine in cui si trovano
- Poi vengono addizione e sottrazione, nell'ordine in cui si trovano

Esempio: (a+b+c+d+e)/5
1 2

Esempio: a*b+c-d/e
1 3 4 2

Operatori di uguaglianza e relazionali p.37, p.115

- Operatori di uguaglianza:
  uguale ==
  diverso !=
- Operatori relazionali:
  maggiore >
  minore <
  maggiore o uguale >=
  minore o uguale <=

Attenzione a:

- non confondere = e == (if (voto==18) printf("pochino");)
- non interporre uno spazio quando l'operatore è formato da due caratteri

Operatori logici p.112

- && (AND logico)
  - Ritorna true se entrambe le condizioni sono vere
- || (OR logico)
  - Ritorna true se almeno una delle due condizioni è vera
- ! (NOT logico, negazione logica)
  - Inverte la verità/falsità della condizione
  - E' un operatore unario
- Sono utili nelle condizioni dei cicli

Esempio:

Espressione	Risultato
true && false	false
true || false	true
!false	true

Attenzione: se vi sono più condizioni in AND o in OR il C le valuta da sx a dx, interrompendo la valutazione se può già dedurre il risultato; quindi bisognerebbe anteporre le condizioni che variano di più e posporre quelle più dispendiose computazionalmente.

Esercizi (oper. logici)

- Scrivere un'espressione C che, data una variabile intera a, restituisca true solo se a contiene un numero positivo pari oppure un numero negativo dispari
- scrivere un'espressione C che, data una variabile reale b, restituisca true solo se b contiene un valore compreso in uno dei seguenti intervalli: [2,3], (5,6)
- scrivere un'espressione C che, date due variabili intere a e b, restituisca true solo se a e b contengono entrambe valori >2 o <0
- dati due punti (a,b) e (c,d) che rappresentano i vertici superiore sinistro ed inferiore destro di un rettangolo, scrivere un'espressione C che restituisca true solo se un terzo punto (x,y) si trova dentro il rettangolo
- dati 3 punti (a,b), (c,d) ed (e,f), scrivere un'espressione C che restituisca true solo se i tre punti giacciono sulla stessa retta.

Operatori di assegnamento (1) p.74

Abbiamo usato comandi come:

```
Somma=0
Somma=Somma+cont
Somma=Somma+1
```

Che si sicrivono in C come:

```
Somma=0;
Somma=Somma+cont; oppure
   Somma+=cont;
Somma=Somma+1; oppure
   Somma+=1; oppure
      Somma++;
```

Operatori di assegnamento (2)

- Ogni istruzione del tipo `variabile=espressione;` significa che alla variabile viene assegnato il valore dell'espressione.
  Quando un valore viene assegnato ad una variabile, tale valore rimpiazza (e quindi distrugge) il precedente
- Ogni istruzione del tipo `var=var operatore espressione;` può essere scritta come:
  `var operatore=espressione` dove `operatore` è un operatore binario (+, -, *, /)

Esempio: `c=c+3;` è equivalente a `c+=3;`

Operatori di assegnamento (3)

- Se nell'istruzione `var=var operatore espres;`
  - `operatore` è + o - e
  - `espres` è 1

  si parla di incremento o decremento.

Esempio: `c=c+1;`

Postincremento: `c++;` (prima usa c poi la incrementa)

Preincremento: `++c;` (prima incrementa c poi la usa)

Esempio:

```
c=3;          d=3 e c=4
d=c++;
```

```
c=3;          d=4 e c=4
d=++c;
```

Esercizi (assegnamento)

Valutare i valori delle variabili in gioco alla fine dei seguenti frammenti di codice:

- `x=3; somma=2;`
  `somma+=x++;`
- `x=3; prodotto=2;`
  `prodotto*=++x;`
- `x=3; y=2;`
  `++x += y++;`
- `x=3; y=2;`
  `x+=++x-y--`