Dall'Algoritmo al Programma

Prof. Francesco Accarino
IIS Altiero Spinelli Sesto San Giovanni

IL PROGRAMMA

Gli algoritmi sono **modelli di descrizione astratti** e per controllarne il funzionamento devono essere **Implementati**

Ad esempio la descrizione di una ricetta di cucina è un algoritmo ma la preparazione della pietanza è la realizzazione della ricetta. E' solo allora che si vede se la ricetta "funziona"

In ambito informatico, un algoritmo viene **Implementato** tramite un **programma**, ossia un testo scritto in uno specifico linguaggio detto **linguaggio di programmazione** per essere eseguito da uno specifico calcolatore elettronico

L'algoritmo e il programma

Ogni linguaggio di programmazione ha una sua **sintassi** ed una propria **rappresentazioni dei dati**, e utilizza le proprie **operazioni di manipolazione dei dati**

Invece:

Le proprietà degli algoritmi sono **generali** e **indipendenti** dalle caratteristiche di specifici linguaggi di programmazione o di particolari calcolatori elettronici

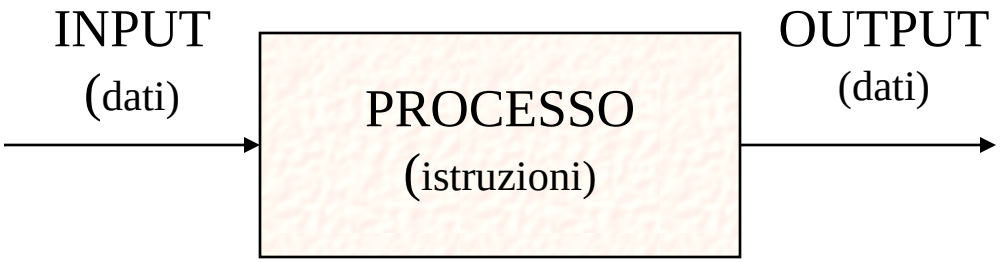
Quindi:

PER RISOLVERE UN PROBLEMA E' ASSOLUTAMENTE NECESSARIO TROVARE UN PROCEDIMENTO RISOLUTIVO DESCRIVIBILE CON UN ALGORITMO, E SOLO DOPO SI PUO' PASSARE A SCRIVERE IL PROGRAMMA

PROGRAMMA = ISTRUZIONI + DATI

Un programma può essere visto come un manipolatore di dati:

riceve dei dati in ingresso che descrivono il problema e produce in uscita come risultato la soluzione del problema



Le operazioni sui dati sono le **istruzioni**.
I dati sono gli **oggetti** su cui vengono eseguite le **istruzioni**.

Tutti i linguaggi di programmazione prevedono un insieme di
dati e istruzioni
che il programmatore può utilizzare nella realizzazione dei propri programmi.

I DATI

I dati possono essere distinti in base a :

- Utilizzo all'interno del processo di cacolo
  - **INPUT**
  - **OUTPUT**
  - **LAVORO**
- Formato
  - **BOOLEANO**
  - **NUMERICO**
    - **INTERO**
    - **DECIMALE**
  - **ALFANUMERICO**
- Valore assunto durante il processo
  - **VARIABILE**
  - **COSTANTE**

Questi aspetti dei dati devono essere definiti già nell'algoritmo, in modo da poterli poi "tradurre" con la sintassi propria del linguaggio di programmazione

Tipi di dichiarazione	Rappresentazione
Char	**Carattere (es. a)** **0<=n<=255 oppure -128<=n<=127**
Int	**Numero intero a 32 bit** $-2^{31} <= I <= 2^{31}-1$
Short	**Numero intero corto a 16 bit** $-2^{15} <= I <= 2^{15}-1$
Long	**Come intero**
Float	**Numero reale "corto" (es 14.4)** **A 32 bit**
Double	**Numero reale "lungo"** **A 64 bit**

In C **non esiste il tipo boolean**: Si usa la convenzione che lo zero rappresenta il valore **falso** e l'uno il valore **vero** (tutti i valori diversi da zero rappresentano il vero)

LE ISTRUZIONI

Ogni linguaggio di programmazione ha le proprie istruzioni, ma tutti i linguaggi devono possedere alcune istruzioni per compiere le azioni fondamentali

- **Acquisire** i dati su cui attivare il processo di elaborazione ⟶ **input**
- **Comunicare** all'esterno i risultati ottenuti ⟶ **output**
- **Azioni di tipo aritmetico** ⟶ **assegnazione**

  - utilizzare gli operatori aritmetici **(+, -, /, *, %)**

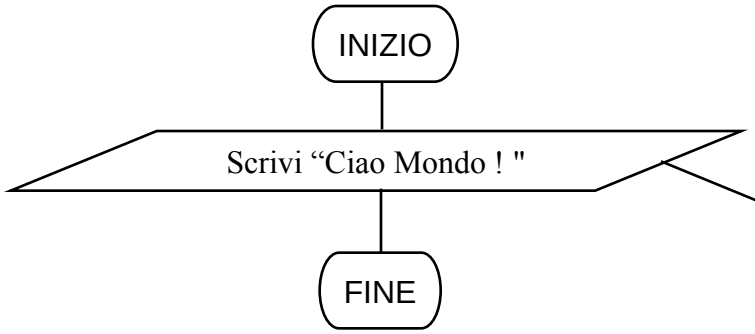
- **Azioni di tipo logico** ⟶ **confronto**

  - Utilizzare gli operatori di relazione **(=, <, >, <=, >=, !=)** per confrontare il contenuto di due variabili
  - Combinandoli con gli operatori logici **(and, or, not)**

ISTRUZIONI DI OUTPUT

Sono le istruzioni che visualizzano dati e messaggi a video

Esempio: scrivere un programma che manda un messaggio a video




```
#include <stdio.h>
#include <conio.h>
int main( )
{
  printf("\nCiao Mondo !");
    getche();
}
```

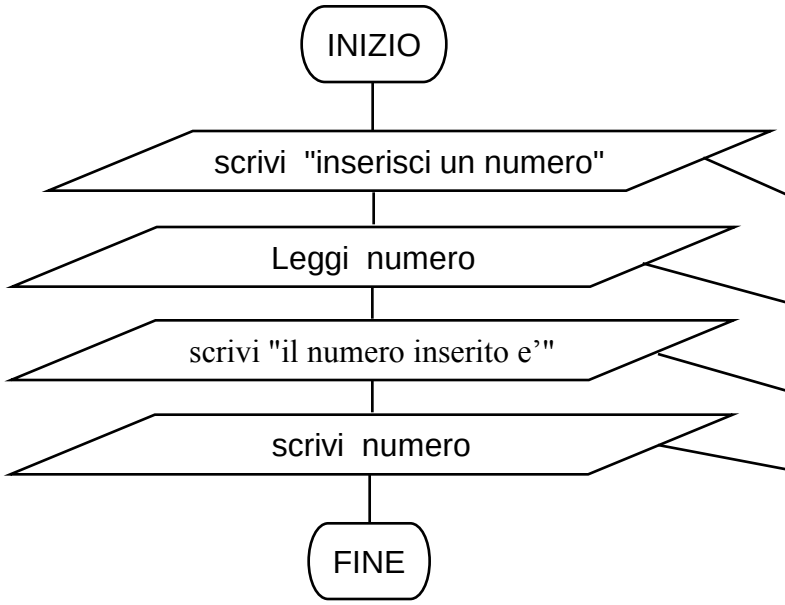
Nell’algoritmo il comando di output è *scrivi* seguito dal messaggio o dal dato che si deve far vedere a video

In linguaggio C l’istruzione è printf(" ");
Il messaggio deve essere inserito tra virgolette

ISTRUZIONI DI INPUT

Sono le istruzioni che acquisiscono dati digitati da tastiera

Esempio: scrivere un programma che legge un numero e le riscrive a video




```
#include <stdio.h>
#include <conio.h>
int main()
{   int numero;

   printf("inserisci un numero\n  ");
   scanf("%d",&numero);
   printf("\nil numero  inserito e' :");
   printf("%d",numero);
   getche();
}
```

Nell'algoritmo il comando di input è *leggi* seguito dal dato che si deve acquisire

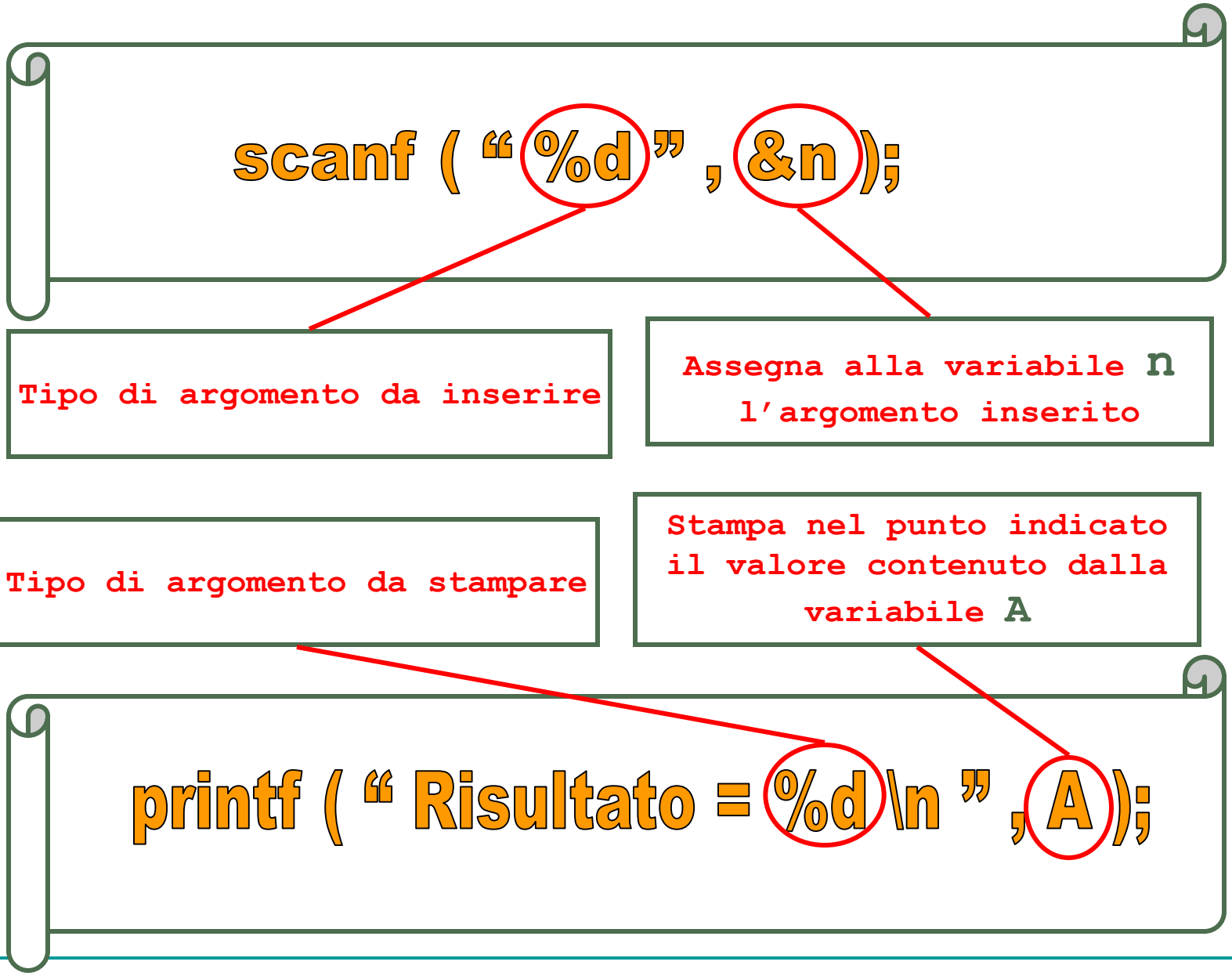
In linguaggio C l'istruzione è **scanf("%d",&numero);**

Il dato deve essere stato definito prima dell'acquisizione

Argomenti di printf e scanf

printf ("<stringa di Formattazione>" , <elenco argomenti>);

scanf ("<stringa di Formattazione>" , <elenco argomenti>);

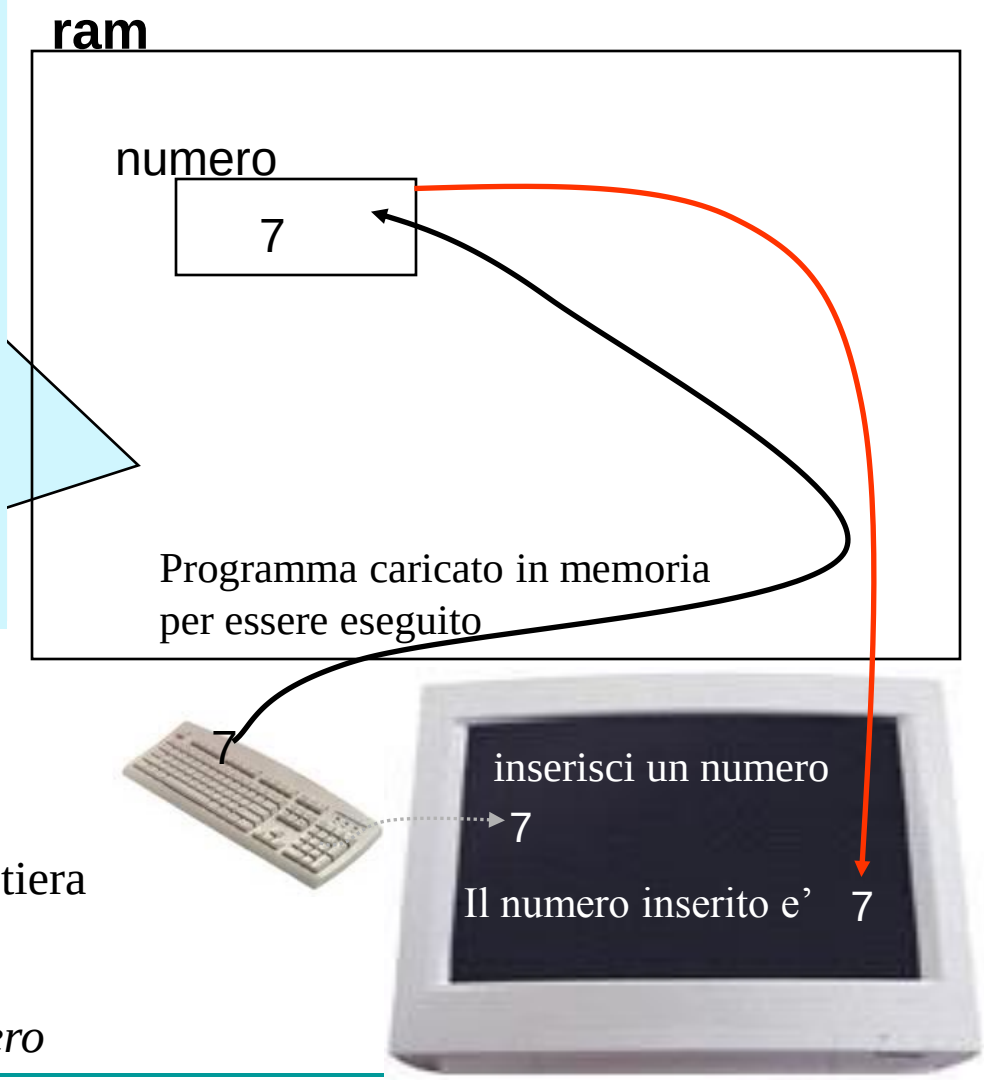

scanf (“ %d ” , &n);
Tipo di argomento da inserire
Assegna alla variabile n l’argomento inserito
Tipo di argomento da stampare
Stampa nel punto indicato il valore contenuto dalla variabile A
printf (“ Risultato = %d \n ” , A);

Argomenti di printf e scanf

Sintassi da utilizzare	Descrizione
%d	Dati di tipo int
%lf	Dati di tipo double
%l	Dati di tipo long
%f	Dati di tipo float
%c	Dati di tipo char
%s	Dati di tipo stringa

Esempio di esecuzione di input e di output

```
#include <stdio.h>
#include <conio.h>
int main()
1 {   int numero;

2   printf("inserisci un numero\n  ");
3   scanf("%d",&numero);
4   printf("\nil numero  inserito e' :");
5   printf("%d",numero);

    getche();
}
```



1 Allocazione della variabile *numero*

2 Output a video di un messaggio

3 Acquisizione del valore digitato da tastiera
con echo a video

4 Output a video di un messaggio

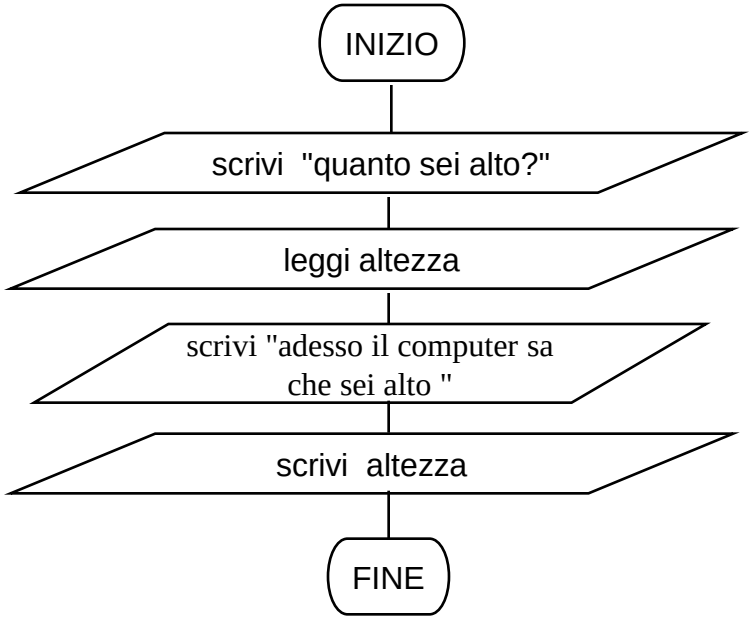
5 Output del valore della variabile *numero*

OUTPUT DI DATI

Nell'esempio precedente abbiamo visto l'output di un dato numerico:

```
printf("%d",numero);
```

Facciamo un altro esempio: Programma che chiede all'utente la sua altezza e la stampa avideo



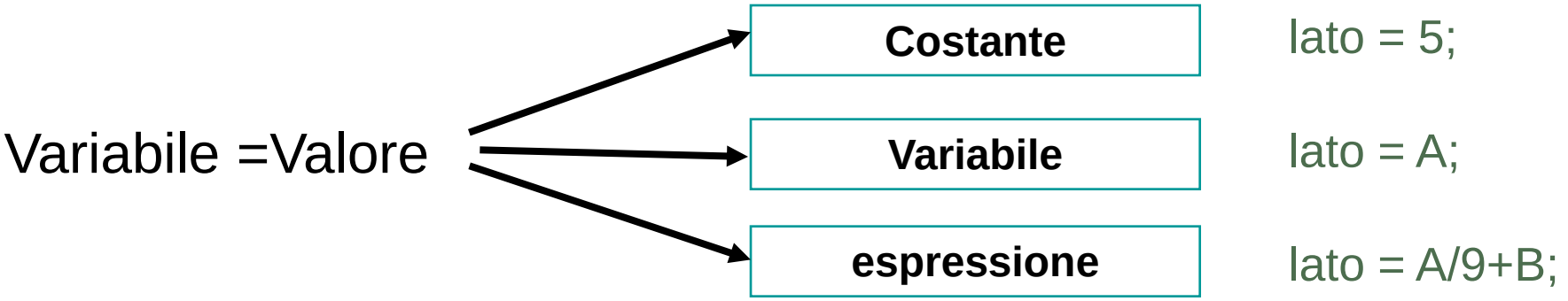

```
#include <stdio.h>
#include <conio.h>
int main()
{
   int altezza;
   printf("\nquanto sei alto?");
   scanf("%d",&altezza);
   printf("\nadesso il computer sa che sei alto%d ",altezza);
   getche();
}
```

ISTRUZIONI DI ASSEGNAZIONE

L'istruzione di assegnazione è quella che **assegna un valore ad una variabile**

Esempio: definire una variabile detta *lato* → int lato;
ed assegnare a tale variabile il valore 9 → lato = 9;

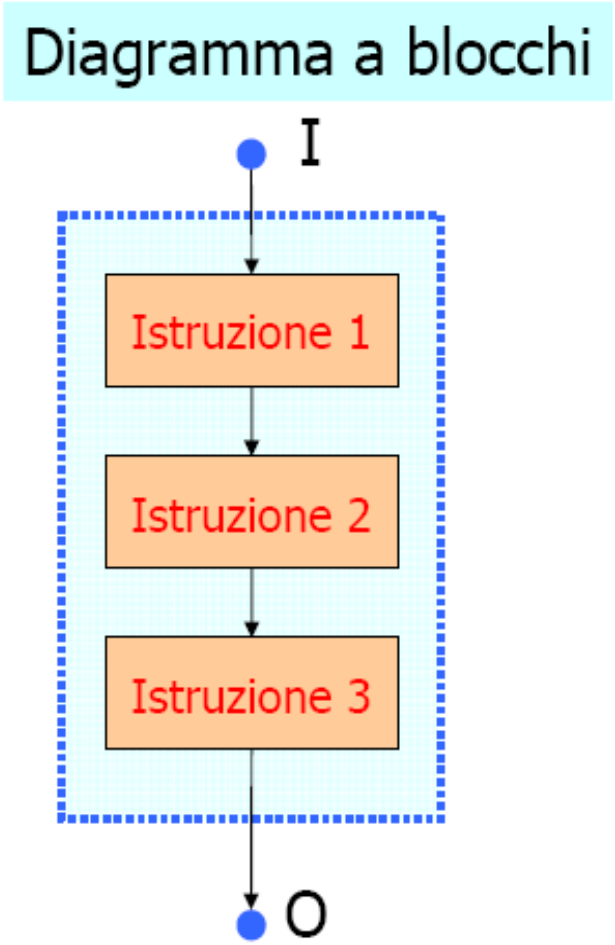
L'assegnazione del valore può essere fatto anche contestualmente alla definizione → int lato=9;



Nel caso di valori **costanti**, l'assegnazione avviene necessariamente nel momento della definizione e ovviamente **non può essere modificato** #define costante 9; oppure const int costante=9;

LA SEQUENZA

E' un insieme di azioni da svolgere secondo un **ordine prefissato**. L'ordine di esecuzione è dalla prima all'ultima istruzione seguendo un percorso unico.



BLOCCO

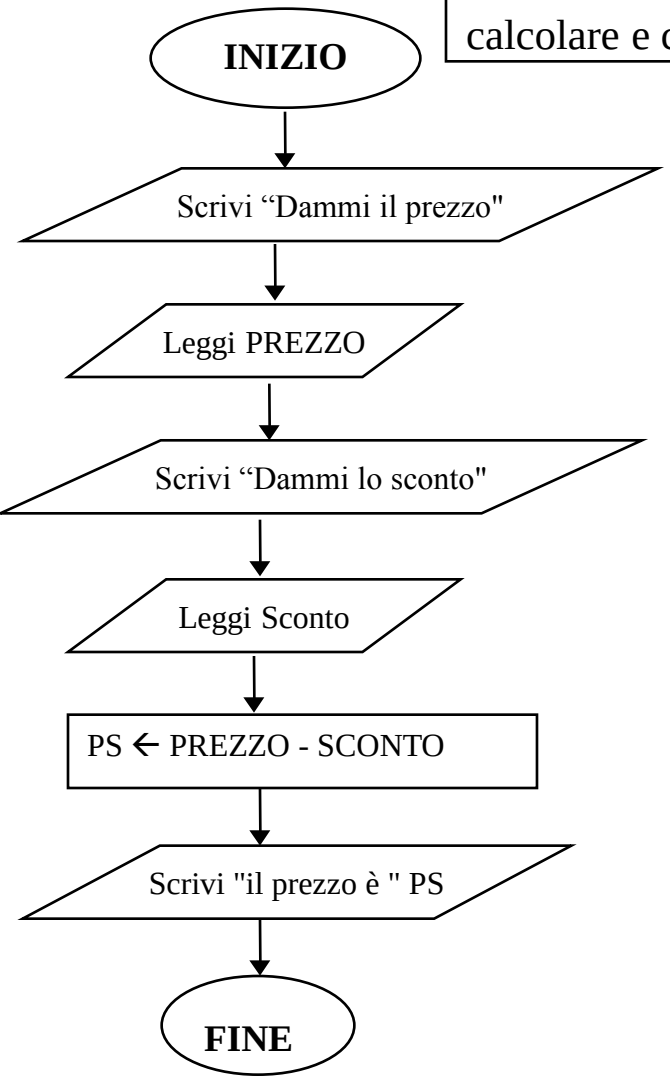
In C si utilizza

{ per indicare l'**inizio della equenza di istruzioni**

} per indicare la **fine della sequenza**

Esempio di istruzioni in sequenza

PROBLEMA: dato il prezzo di un prodotto e lo sconto effettuato, calcolare e comunicare il prezzo scontato



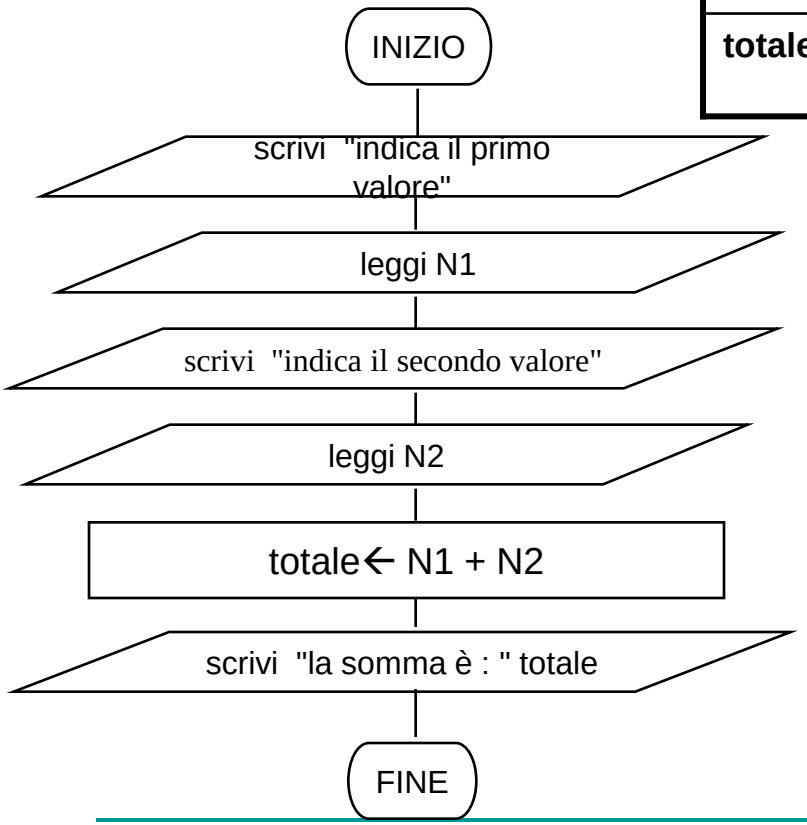

```
#include <stdio.h>
#include <conio.h>

int main()

{
   float prezzo, sconto, PS;

   printf("\nDammi il prezzo : ");

   scanf("%f",&prezzo);

   printf("\nDammi lo sconto : ");

   scanf("%f",&sconto);

   PS = prezzo - sconto;

   printf("\nil prezzo scontato e' :  %f",PS);

   getche();

}
```

scrizione
del prodotto
effettuato
scontato

Esempio di operazioni di calcolo

Acquisire due numeri e mostrare il risultato della loro somma

nome	I/O/L	V/C	N/A/B	descrizione
N1	I	V	N intero	Primo addendo
N2				
totale				




```
#include <stdio.h>
#include <conio.h>

int main()
   {   int N1,  N2, totale ;

    printf("\nindica il primo valore ");
    scanf("%d",&N1);
    printf("\nindica il secondo valore ");
    scanf("%d",&N2);
    totale= N1 +N2;
    printf(\nla somma è : %d",totale);
    getche();
   }
```

Output concatenato

Dall'esempio precedente abbiamo visto che è possibile **concatenare più output con una stessa istruzione**

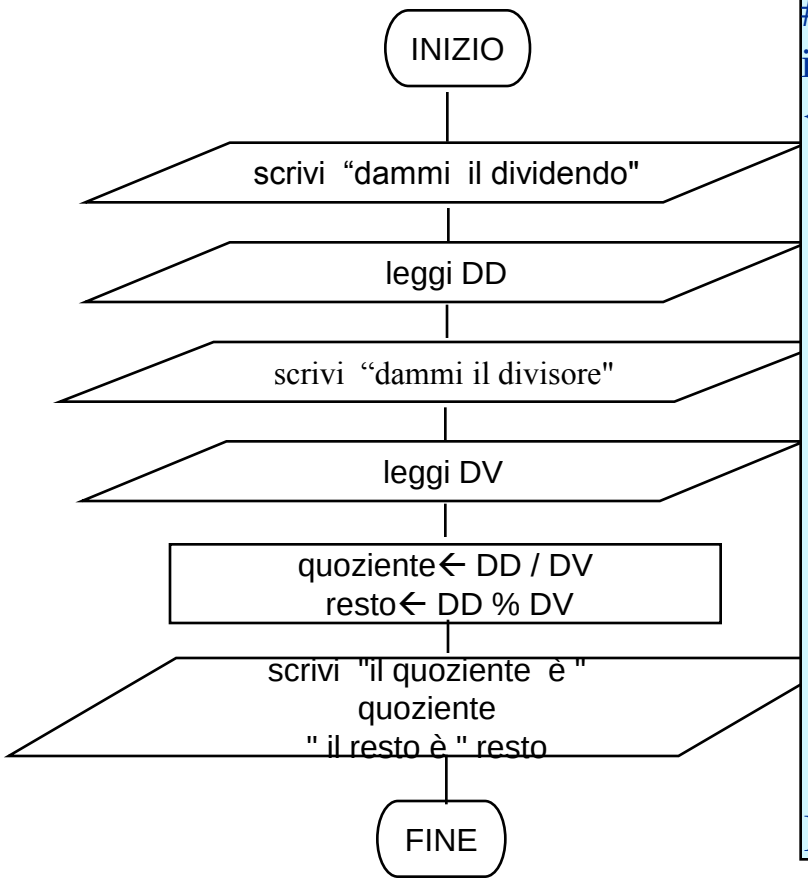
```
Printf("\nla somma è : %d",totale);
```

(carattere per *andata a capo)* (messaggio di testo) tipo dato

Abbiamo visto anche che è possibile definire più variabili dello stesso tipo **con una stessa istruzione:** `int N1, N2, totale ;`

Esempio di utilizzo di operatore modulo

Dati due numeri interi, calcolare quoziente e resto




```
#include <stdio>
#include <conio>
int main()
{
   int DD, DV, resto;
   float quoziente;

   printf("\niserisci il primo valore   :");
   scanf(%d",&DD);

   printf("\niserisci il secondo valore   :");
   scanf(%d",&DV);

   quoziente = DD/DV;
   resto = DD% DV;
   printf("\nil quoziente e': %f e il resto e' %d",quoziente,resto);
     getche();
}
```

ASSEGNAZIONE TRAMITE OPERATORI UNARI

Gli **operatori aritmetici unari** sono quelli che richiedono un solo operando.

Nel linguaggio C sono:

++	Per incrementare di una unità l'operando
- -	Per decrementare di una unità l'operando

l'istruzione numero++ equivale all'istruzione numero = numero + 1

l'istruzione numero- - equivale all'istruzione numero = numero - 1

Struttura di Un Programma C

```
#include….. //direttive di inclusione delle librerie
#include…..
#define……//definizioni di costanti
int main()    //inizio del programma
{
   int …..;   //dichiarazione delle variabili
   float…;
   char…;
     ……..
    Istruzioni //sequenza di istruzioni
     ……
getche();     //pausa per verificare il risultato
}             //fine
```