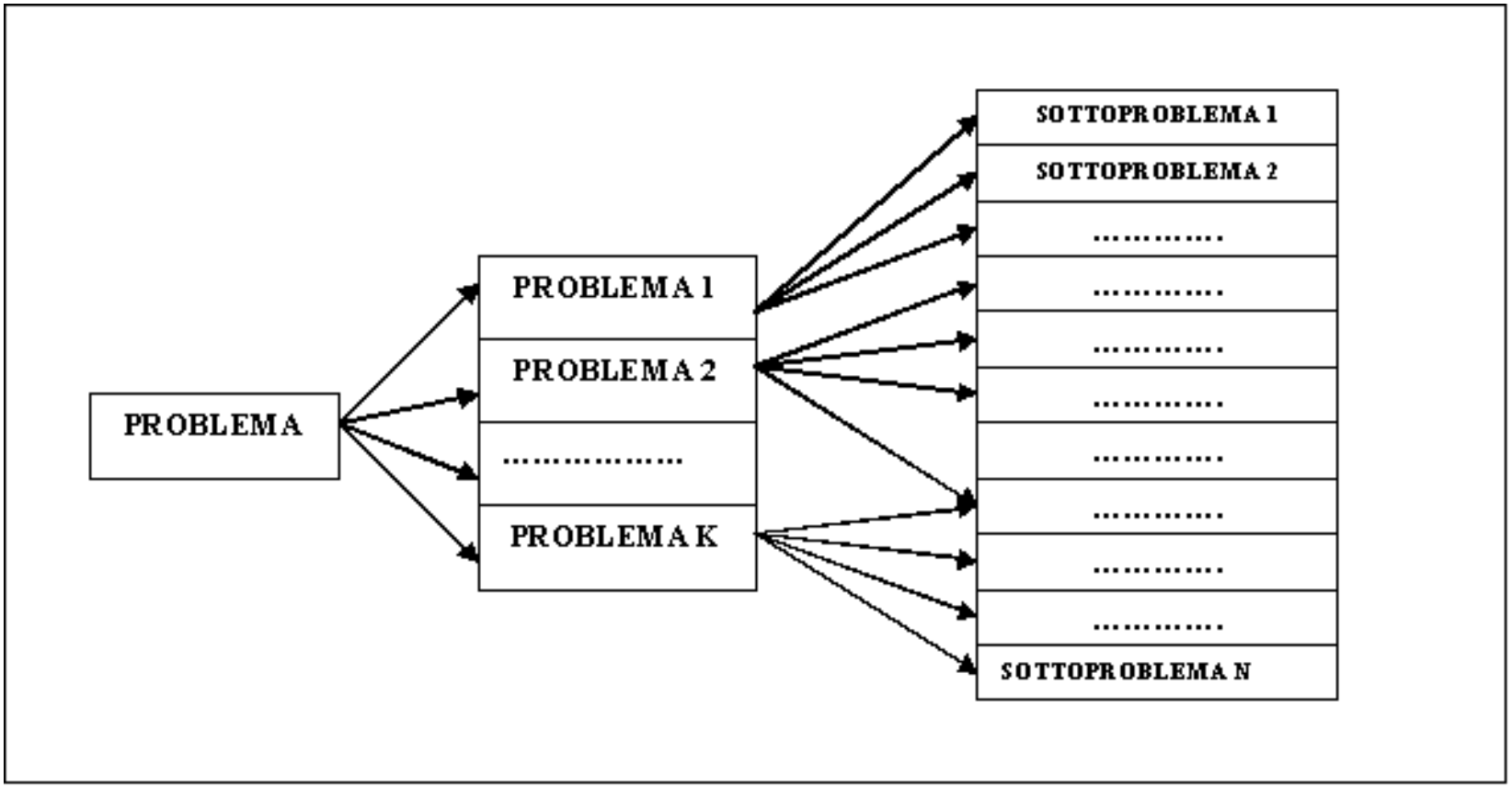
Progettazione Top Down e funzioni in C

Prof. Francesco Accarino
IIS Altiero Spinelli Sesto San Giovanni

Progettazione top-down

- Quando si ricerca la soluzione di un problema complesso può essere complicato trovare subito i passi elementari per la sua soluzione e soprattutto rappresentarli in un Flow Chart.
- In questo caso può essere utile concentrarsi sugli aspetti generali del problema individuando le parti che lo compongono e arrivare alla soluzione per affinamenti successivi

Progettazione top-down



Vantaggi della programmazione Top-down

- permette di concentrarsi in ogni momento del progetto sugli aspetti più significativi, rimandando a momenti successivi gli aspetti di maggior dettaglio
- consente di dare una descrizione dell'algoritmo risolutivo più leggibile
- essendo ciascun sottoproblema indipendente dagli altri, la sua risoluzione può essere modificata, se necessario, senza che si modifichi la struttura generale dell'algoritmo risolutivo (migliore manutenzione dei programmi)
- si può suddividere il compito di risolvere il problema tra più persone o gruppi di lavoro
- la risoluzione di sottoproblemi può essere riutilizzata in altri problemi

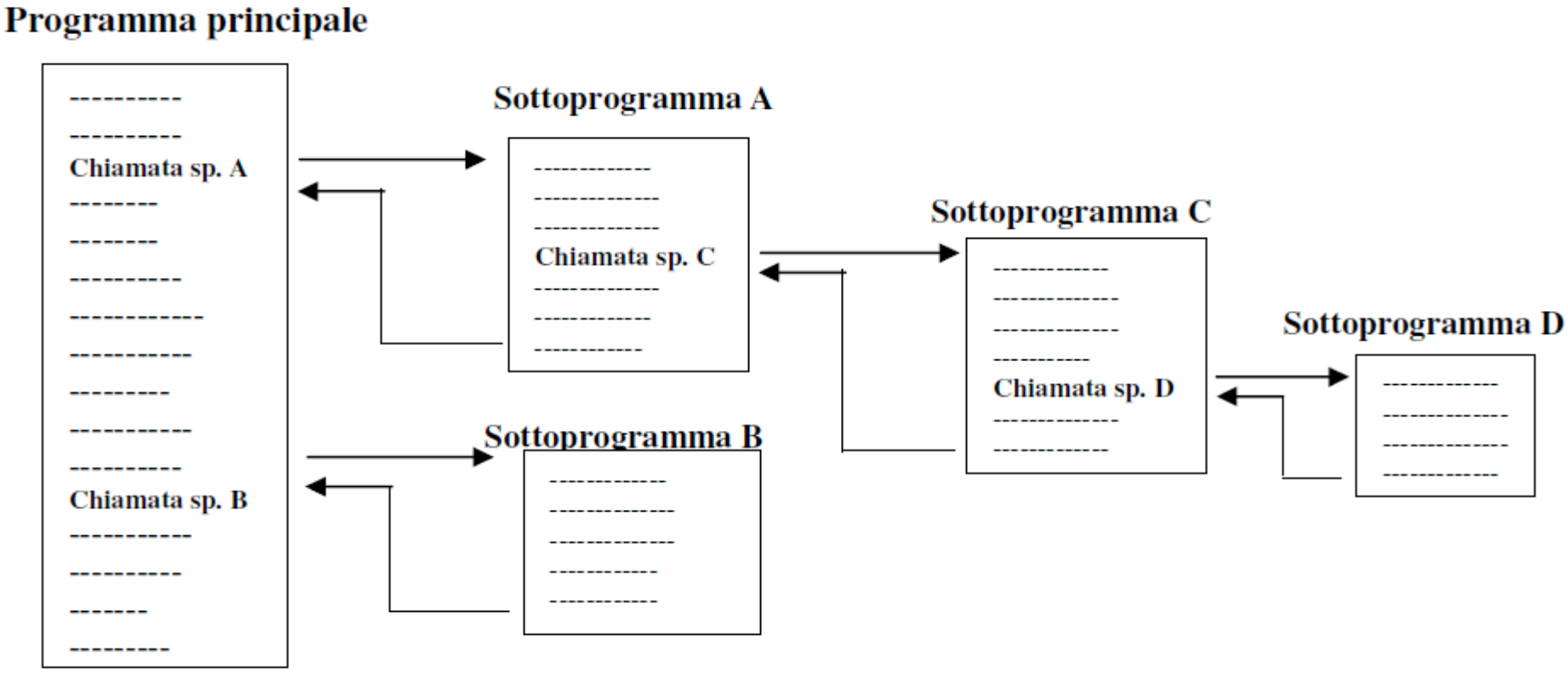
Progettazione top-down

- analisi del problema ovvero riconoscimento dei sotto-problemi in cui il problema può essere scomposto
- analogamente può avvenire per i sotto-problemi
- ci si ferma quando
  - non si ha un ulteriore beneficio dalla scomposizione
  - i sotto-problemi sono di immediata soluzione

Progettazione top-down

Il programma che risolve il problema P sarà quindi costituito da:

- **un programma principale (main), avente la funzione di risolvere globalmente il *problema P,***
- **uno o più sottoprogrammi la cui funzione è quella di risolvere i *sottoproblemi contenuti in P.***



Esempio: preparo colazione

- preparo colazione
  - prendo il latte dal frigorifero
  - faccio scaldare il latte
  - servo il latte in tavola

Esempio: preparo colazione

- prendo il latte dal frigorifero
  - apro il frigorifero
  - tolgo il latte dal frigorifero
  - verso il latte nel bricco
  - metto il restante latte nel frigorifero
  - chiudo il frigorifero

Esempio: preparo colazione

- faccio scaldare il latte
  - metto il bricco sul fornello
  - accendo il fuoco
  - aspetto quanto basta
  - spengo il fuoco
  - tolgo il bricco dal fornello

Esempio: preparo colazione

- Servo il latte in tavola
  - **Verso il latte nella tazza**
  - **Aggiungo zucchero e caffè**
  - **Mescolo**
  - **Porto in tavola**

Funzioni

Una funzione è un modulo indipendente di software che svolge una ben definita elaborazione su degli eventuali dati in input combinati con i dati interni per fornire un comportamento ed eventualmente restituire un risultato

Dalla definizione si intuisce che:

- I dati in input (passati alla funzione) possono essere più di uno o nessuno
- L'output è costituito sempre da un unico dato (restituito dalla funzione) e può anche non esserci

Implementare Funzioni

- Servono:
  - Il nome della funzione
  - I parametri della funzione (cosa riceve)
  - Che cosa restituisce
  - Il blocco di istruzioni che devono essere eseguite quando la funzione viene chiamata
- Il blocco di istruzioni è detto corpo della funzione

Esempio Funzione Saluto

```
# include < stdio.h >
//Prototipo della funzione

Void Saluto( Void);
/*
 * chiam a una funzione che
 * stampa un semplice saluto
 * /
Int main(void)
{
 saluto() ;
 return 0 ;
}

/*
 * stampa un semplice saluto
 * /
Void Saluto( Void)
{
Printf("Un saluto dal programma! \ n " );
}
```

Prototipo della funzione:
definisce il tipo di dato ritornato dalla funzione
Il nome della funzione
funzione
i tipi e il numero dei dati ricevuti
con void si intende nessun dato

Chiamata della funzione

Corpo della funzione:
Istruzioni che devono essere eseguite o eventuali chiamate ad altre funzioni

Parametri

- Sono le informazioni (cioè i dati) passati ad una funzione
  - I parametri Formali sono variabili dichiarate dalla funzione all'atto della dichiarazione del suo corpo funzionale per contenerei dati che gli vengono passati
  - I Parametri attuali sono i valori passati alla funzione quando viene chiamata

Esempio Stampa in ordine

```
#include ”stdio.h”
void Ordina(int , int);

int main(void){
Int x=3,y=5;
Ordina(10,8);
Ordina(y,x+4);
Return 0;
}

//stampa due numeri ordinati
Void Ordina(int a, int b)
{
If(a>b)
   printf("  %d,%d”,a,b);
else
    printf("  %d,%d”,b,a);
}
```

Prototipo La funzione si chiama Ordina non ritorna niente e riceve due interi

Chiamata con parametri attuali 10 e 8

Chiamata con parametri attuali y e x+4

Corpo della funzione con dichiarazione delle due variabili (i parametri formali) per ricevere i due dati passati

Output del programma:
8 10 5 7

Esempio Funzione che ritorna un dato

```
#include ”stdio.h”
int Max(int , int);

int main(void){
Int m;
m=Max(9,12);
Printf(”il massimo è%d ”,m);
return 0;
}

//ritorna il massimo tra due valori
int Max(int a, int b)
{
Int m;
If(a>b)
   m = a;
else
m = b;
return m;
}
```

Prototipo La funzione si chiama Max ritorna un intero e riceve due interi

Chiamata con parametri attuali 9 e 12

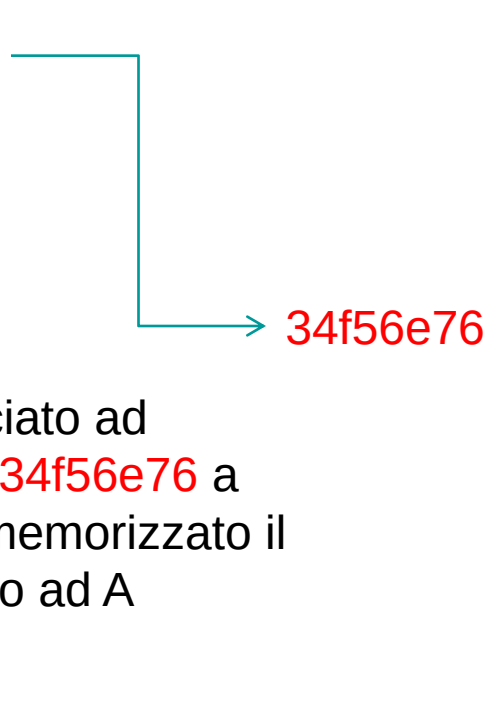
Corpo della funzione con dichiarazione delle due variabili (i parametri formali) per ricevere i due dati passati m variabile per uso locale

Ritorno del valore calcolato

Concetto di variabile

Quando in c si dichiara una variabile si dice al compilatore di associare al nome della variabile da noi dichiarata una locazione di memoria (cioè un certo numero di byte) idonea a contenere il tipo di dato dichiarato

Esempio: int A;



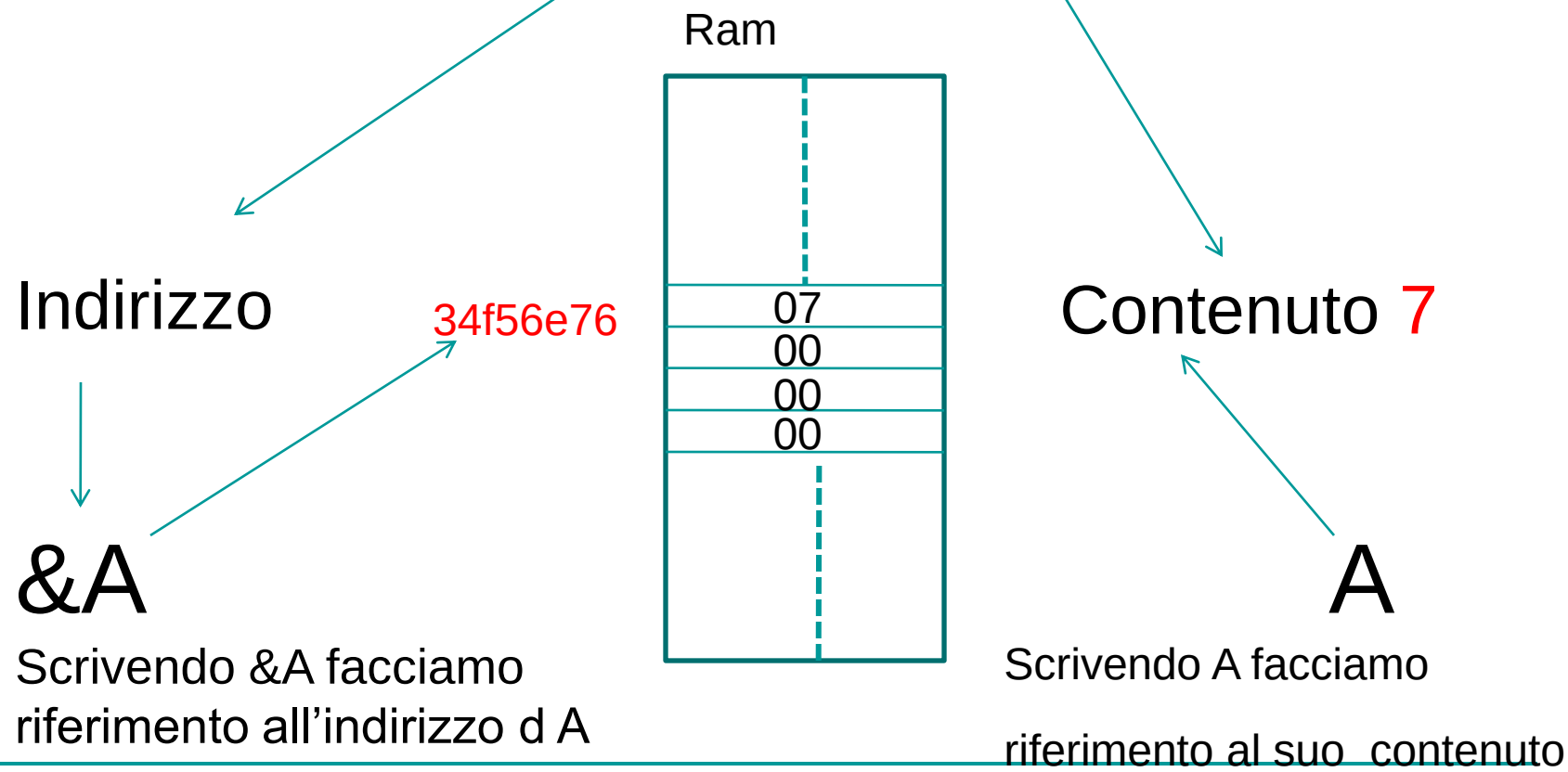
Quindi ad A è associato ad esempio l'indirizzo: 34f56e76 a partire dal quale è memorizzato il contenuto assegnato ad A

Scrivere:
A=7;
Significa :
memorizza nella locazione di memoria di indirizzo 34f56e76
Il valore 7

Concetto di variabile

Ad una variabile quindi sono associate due informazioni

Esempio: int A;



Passaggio di parametri per indirizzo

Avendo capito che ad una variabile sono associate due informazioni:

- L'indirizzo cioè il contenitore
- E un valore cioè il contenuto del contenitore

Si può comprendere che ad una funzione si può passare:

- l'indirizzo di una variabile quando la funzione ne deve modificare il suo contenuto
- Oppure il suo valore quando la funzione deve fare solo uso del suo contenuto ma senza modificarlo

Esempio di funzioni che già usiamo

Alla funzione printf passiamo il valore della variabile perché è appunto il suo valore che vogliamo che essa stampi sul video

printf(“%d”, A);

Passaggio per valore

Alla funzione scanf passiamo l’indirizzo di A perche vogliamo che la funzione metta in A il valore che l’utente ha inserito da tastiera cioè abbiamo bisogno dell’indirizzo, o in senso figurato del contenitore, per contenere ciò che l’utente fornisce in input;

scanf(“%d”, &A)

Passaggio per indirizzo

Esempio di passaggio per indirizzo

```
#include ”stdio.h”
void Max(int , int, int*);

int main(void){
Int m;
Max(9,12, &m);
Printf(”il massimo è%d ”,m);
return 0;
}

//ritorna il massimo tra due valori
void Max(int a, int b, int *c)
{
If(a>b)
   *c = a;
else
*c = b;
}
```

Nuovo tipo variabile che contiene l’indirizzo di un intero

Passo due valori e un indirizzo per contenere il massimo dei due valori

Dichiaro i parametri formali per contenere i dati passati due normali per contenere valori una di tipo puntatore per contenere un indirizzo

Cos'è un puntatore

- Un puntatore è una variabile speciale il cui contenuto è un indirizzo di memoria

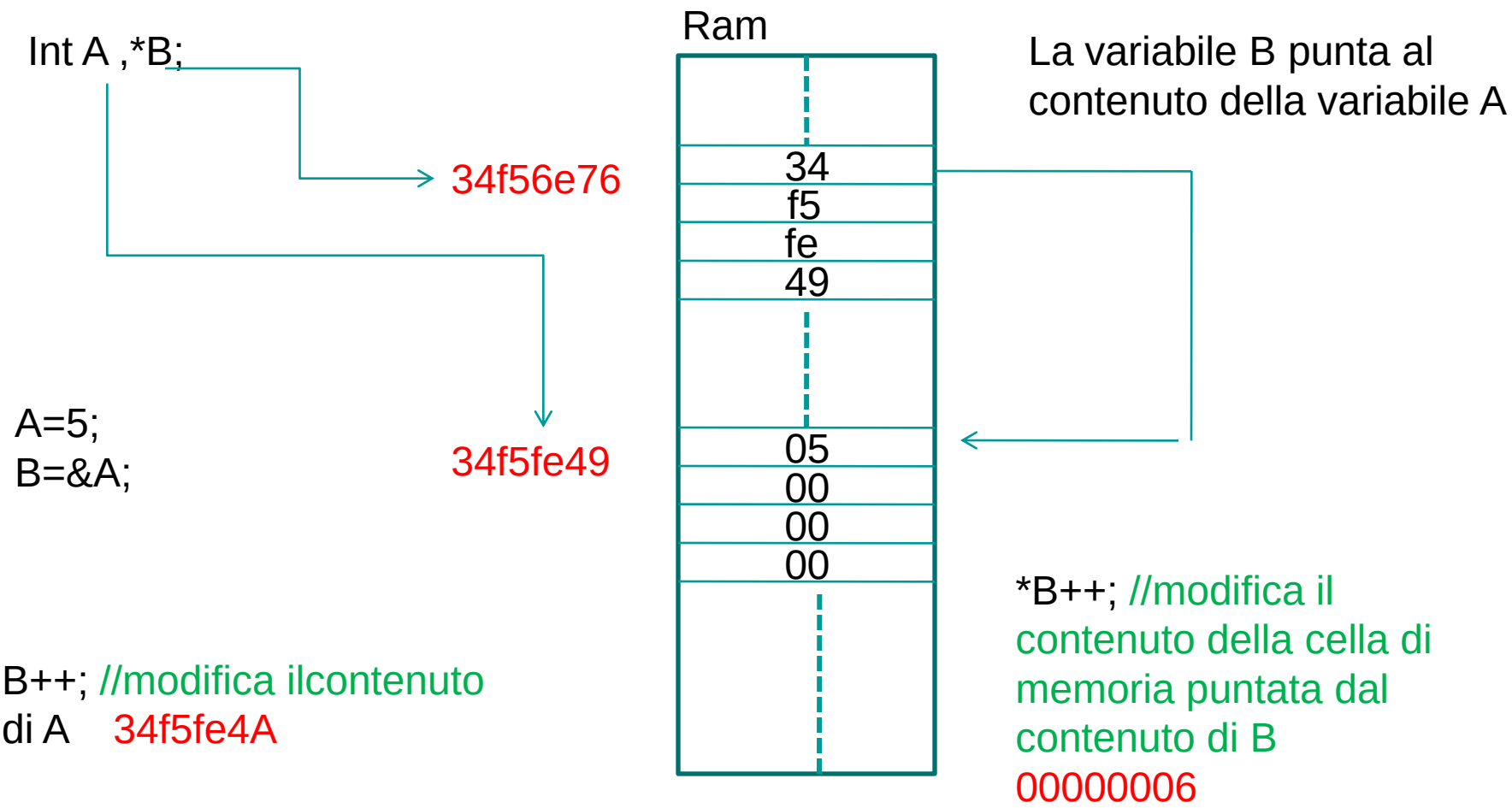
```
Int A ,*B;
A=5;
B=&A;
*B++;
Printf("%d",A); //stamperà 6
```

A variabile normale B variabile puntatore

Usando i nomi delle variabili faccio riferimento al loro contenuto un intero per A un indirizzo di un intero per B

Per riferirsi all'indirizzo di una variabile normale si usa la &.
Per far riferimento al valore puntato dall'indirizzo di una variabile puntatore si usa *

Cos'è un puntatore



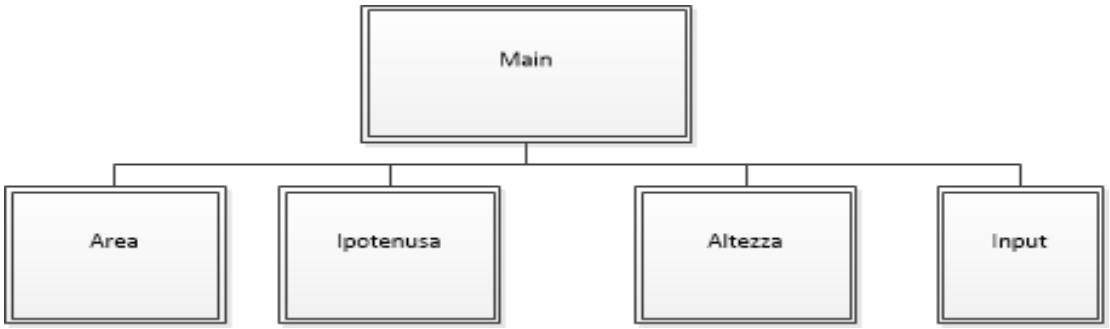
Progettazione top-down

Cosa Cambia nell'analisi del problema e nella ricerca del metodo Risolitivo?

Vediamo un esempio: Supponiamo di avere il seguente testo:

Date da tastiera le misure dei cateti di un triangolo rettangolo visualizza i valori dell'ipotenusa, dell'area e dell'altezza relativa all'ipotenusa

La Prima cosa da fare è individuare i moduli nei quali è scomponibile il problema complessivo. Nel nostro caso potremmo immaginare i moduli seguenti



Progettazione top-down

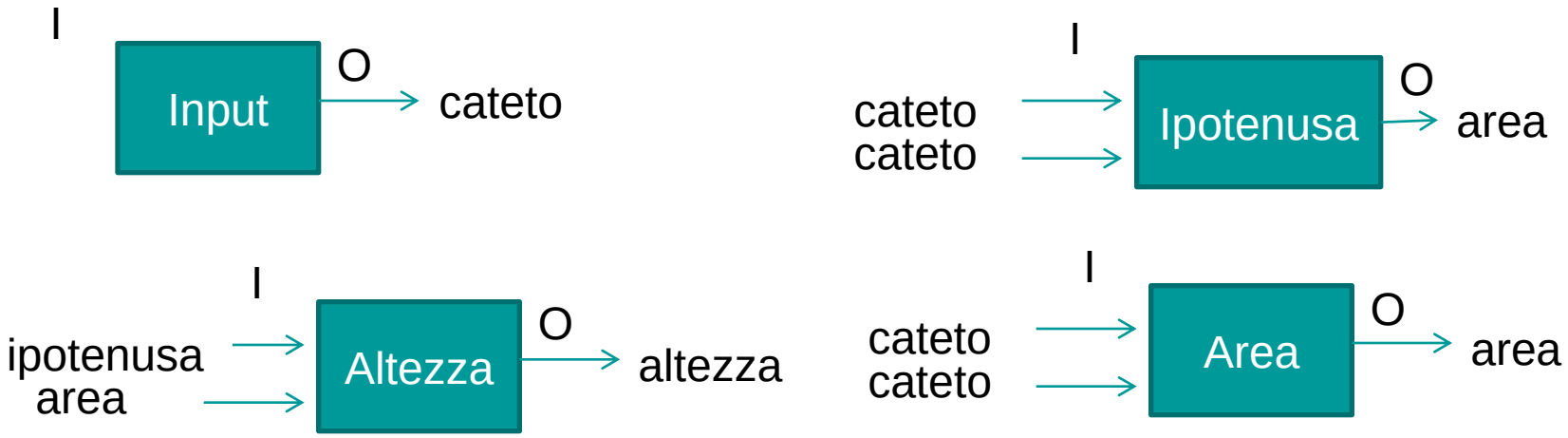
Il passo successivo è analizzare i dati contenuti nel testo e individuare le variabili di input e di output:

Nel nostro caso abbiamo:

input i : due cateti (vincoli entrambi maggiori di 0)

output : L'area, L'ipotenusa, e L'altezza.

E poi analizzare i dati di input e di output dei singoli moduli

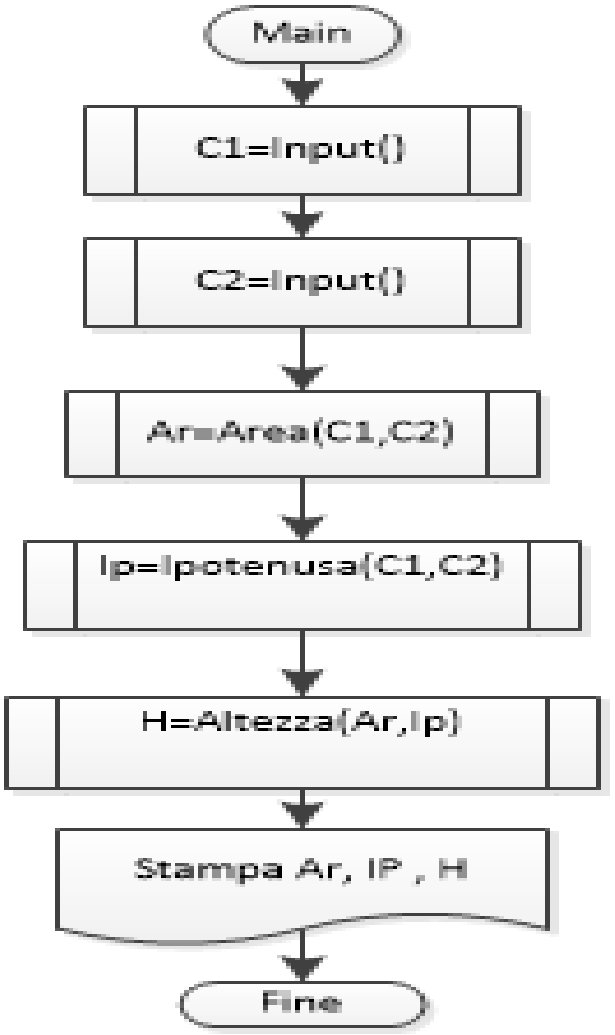


Progettazione top-down

- A questo punto siamo in grado di individuare le Variabili:
- Globali: Sono quelle variabili visibili a tutti i moduli e utilizzate da tutti i moduli
- Locali: Sono quelle variabili utilizzati solo da alcuni moduli e quindi visibili solo a loro
- Nel nostro problema non ci sono dati utilizzati da tutti e quindi i dati che abbiamo individuato nel testo saranno dichiarati come variabili locali del main e passati ai moduli.

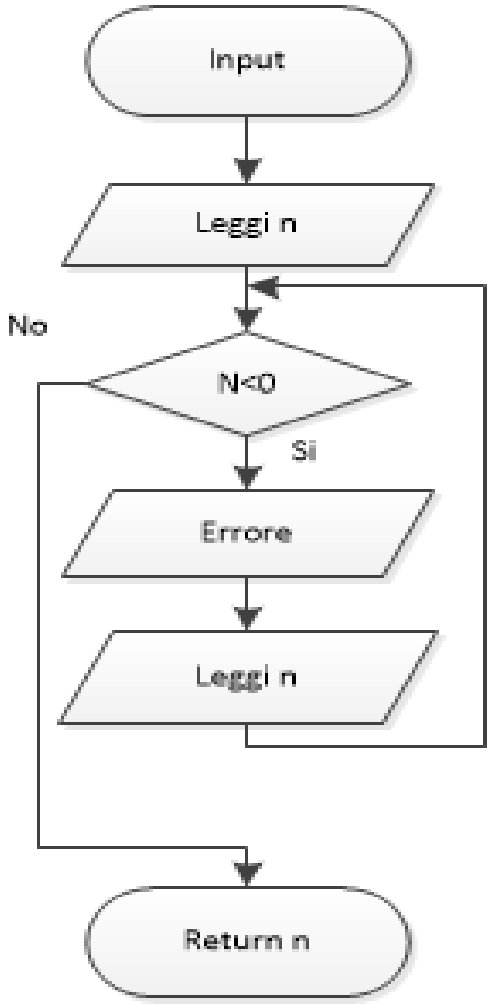
Progettazione top-down

- Flow Chart di Livello 0



Progettazione top-down

- Flow Chart Livello 1:

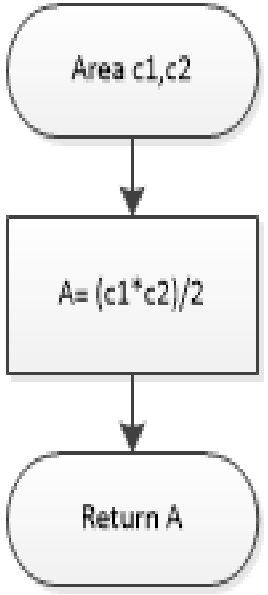


La funzione Input non riceve niente legge un numero con la virgola controllando che sia maggiore di 0 e restituisce il numero letto
Usa la variabile locale n

Progettazione top-down

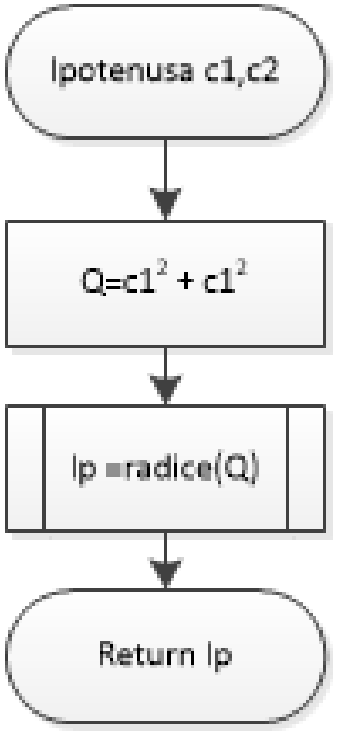
- Flow Chart Livello 1:

La funzione Area riceve i due cateti c1, c2
Calcola l'area e restituisce l'area calcolata
Usa la variabile locale A



Progettazione top-down

- Flow Chart Livello 1:

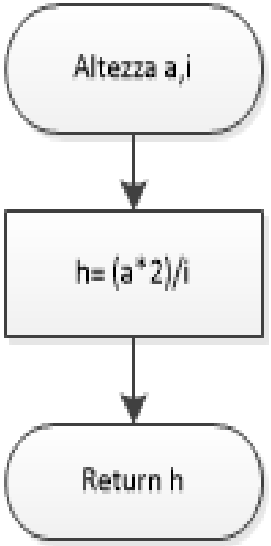


La funzione Ipotenusa riceve i due cateti c1, c2
Calcola l'ipotenusa e restituisce l'ipotenusa calcolata
Usa la variabile locale Q

Progettazione top-down

- Flow Chart Livello 1:

La funzione Altezza riceve L'area e L'ipotenusa
Calcola l'altezza relativa all'ipotenusa e restituisce l'altezza calcolata
Usa la variabile locale h



Progettazione top-down Considerazioni

- Tutti i dati individuati nel testo del problema devono essere rappresentati come variabili locali del main.
- Se ci sono dati utilizzati da tutti i moduli questi saranno rappresentati come variabili globali dichiarate al di fuori del main
- Tutti i dati di lavoro dei moduli vanno dichiarati come variabili locali dei singoli moduli ed eventualmente passati ai loro sottomoduli

Implementazione in C del problema

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

//Prototipi
float Ipotenusa (float , float);
float Area (float , float);
float Altezza (float, float);
float Leggi(void);

void main() // Programma principale
{float c1, c2, ar, ip, alt;
  c1 = Input();
  c2 = Input();
  ar = Area(c1,c2);
  ip = Ipotenusa(c1,c2);
  alt= Altezza(ar,ip);
 printf("Larea del triangolo e' %f\n la sua altezza e' %f e l'ipotenusa e' %f", ar,ip,alt);
 getche();
}
```

Prototipi delle funzioni

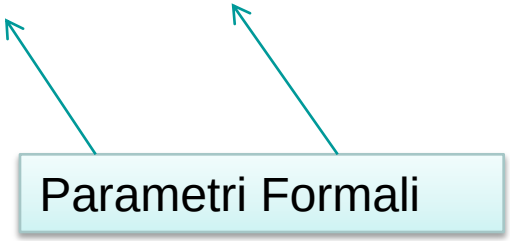
Variabili locali del main e quindi visibili solo nel main

Parametri attuali

Chiamate alle funzioni

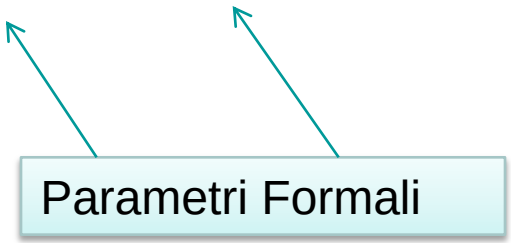
Implementazione in C del problema

```
/* Definizione di funzione Ipotenusa
   * Restituisce il valore dell'ipotenusa
   *dati i valori dei cateti del  triangolo
*/
float Ipotenusa (float a, float b )
{
      float q, ipo;
      q =a*a + b*b;
      ip =sqrt(q);
      return ipo;
}
```



Implementazione in C del problema

```
/* Definizione di funzione Area
   * Restituisce il valore dell'area
   *dati i valori dei cateti del  triangolo
*/
float Area(float c1, float c2 )
{
      float  a;
      a=(c1*c2)/2;
      return a;
}
```

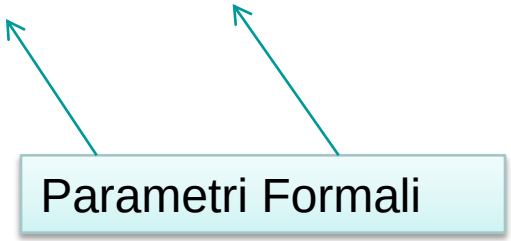


Implementazione in C del problema

```
/* Definizione di funzione Input
   * Restituisce un numero decimale
   *controllando che sia maggiore di 0
*/
float Input( )
{
      float  n;
      printf("Inserisci un valore decimale >0");
      scanf("%f",&n);
      while(n<=0){
      printf("\nErrore il numero deve essere >0");
      printf("\nInserisci un valore decimale >0");
      scanf("%f",&n);
      }
      return n;
}
```

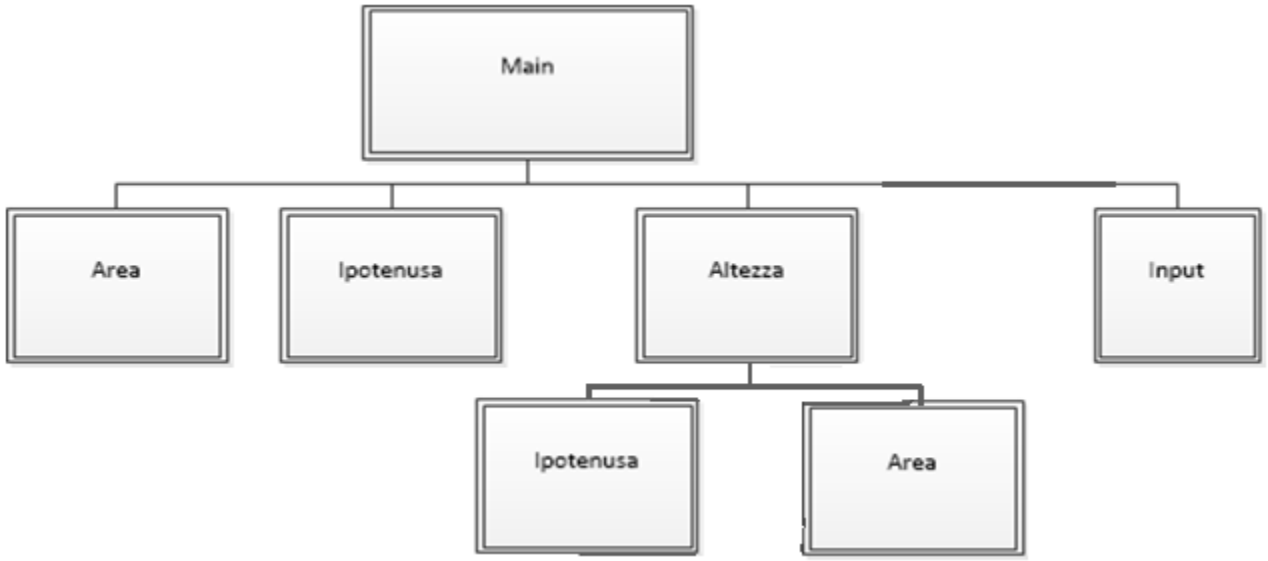
Implementazione in C del problema

```
/* Definizione di funzione Altezza
  *Restituisce l'altezza relativa all'ipotenusa
  *data l'area e l'ipotenusa
*/
float Altezza(float a, float i )
{
     float  h;
     h=(a*2)/i;
     return h;
}
```



Progettazione top-down

Supponiamo ora di aver progettato la soluzione del problema utilizzando i moduli come nello schema seguente:



In questo caso la funzione altezza utilizza anche lei i moduli Ipotenusa e area per calcolare i valori che nella soluzione precedente gli venivano passati come parametri. Possiamo quindi affermare che tutti i moduli che dovrebbero ricevere dei parametri usano gli stessi parametri e cioè c1 e c2

Implementazione in C del problema

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

//Prototipi
float Ipotenusa (void);
float Area (void);
float Altezza (void);
float Leggi(void);

float c1,c2;  //variabili globali sono viste da tutte le funzioni

void main(){ // Programma principale
float ar, ip, alt;
  c1 = Input();
  c2 = Input();
  ar = Area();
  ip = Ipotenusa();
  alt= Altezza();
 printf("Larea del triangolo e' %f\n la sua altezza e' %f e l'ipotenusa e' %f", ar,ip,alt);
 getche();
}
```

Prototipi delle funzioni

Variabili locali del main e quindi visibili solo nel main

Chiamate alle funzioni

Implementazione in C del problema

```
/* Definizione di funzione Ipotenusa
   * Restituisce il valore dell'ipotenusa
   *dati i valori dei cateti del  triangolo
*/
float Ipotenusa ( )
{
      float q, ipo;
      q =c1*c1 + c2*c2;
      ip =sqrt(q);
      return ipo;
}
```

uso delle variabili globali c1 e c2

Implementazione in C del problema

```
/* Definizione di funzione Area
   * Restituisce il valore dell'area
   *dati i valori dei cateti del  triangolo
*/
float Area( )
{
      float  a;
      a=(c1*c2)/2;
      return a;
}
```

uso delle variabili globali c1 e c2

Implementazione in C del problema

```
/* Definizione di funzione Altezza
  *Restituisce l'altezza relativa all'ipotenusa
  *data l'area e l'ipotenusa
*/
float Altezza( )
{
     float  a, i , h;
     a=Area();
     i=Ipotenusa();
     h=(a*2)/i;
     return h;
}
```

La funzione Altezza calcola L'area e L'ipotenusa utilizzando le funzioni

Implementazione in C del problema

```
/* Definizione di funzione Input
   * Restituisce un numero decimale
   *controllando che sia maggiore di 0
*/
float Input( )
{
      float  n;
      printf("Inserisci un valore decimale >0");
      scanf("%f",&n);
      while(n<=0){
      printf("\nErrore il numero deve essere >0");
      printf("\nInserisci un valore decimale >0");
      scanf("%f",&n);
      }
      return n;
}
```

La funzione Input resta invariata

Progettazione top-down Riepilogo

1. Analisi del testo del problema
   a. individuazione dei dati di input e di output
   b. individuazione di vincoli imposti dal testo più vincoli aggiuntivi

2. algoritmo a parole contenente non passi elementari ma macropassi
   a. Individuazione di variabili di lavoro
   b. Tabella dei dati
   c. schema modulare del problema
   d. per ogni modulo definire cosa riceve e cosa restituisce
   e. individuazione di variabili globali
3. flow chart dei singoli moduli
4. Traduzione dei flow chart nel linguaggio di programmazione

Al passo 2 praticamente sviluppiamo i passi del main.Se i moduli sono a loro volta scomponibili bisogna ripetere il passo 2 per ogni modulo