Politecnico di Torino
Sede di Alessandria
Corso di informatica
Programmazione in c: introduzione

prof. Lorenzo Porcelli

e mail: genna18@iol.it

sito: users.iol.it/genna18

Risoluzione di un problema

Dato un problema, per arrivare ad un programma che descrive la soluzione dobbiamo:

- Definire con precisione il risultato: *output*;
- Individuare gli elementi a disposizione: *input*
- Descrivere la risoluzione attraverso:
  - un *algoritmo*,
  - le *strutture dati* su cui l'algoritmo opera.
- Codificare algoritmo e strutture dati in un linguaggio comprensibile alla macchina virtuale utilizzata (in questo corso useremo il linguaggio C)

output

Per poter definire il risultato è necessario stabilire l'obiettivo del problema.

Esempio: esame di informatica.

1. Qual è il voto finale.
2. Chi ha superato l'esame.
3. Quanti hanno superato l'esame.

Per ogni obiettivo cambia il risultato previsto.

input

Stabilito il risultato è possibile definire i dati necessari e soprattutto le restrizioni sui dati.

Esempio:

per definire il voto finale è necessario conoscere il voto dell'esercitazione, il voto della prova teorica e il voto di programmazione.

Ogni voto è compreso tra 1 e 30.

algoritmo

E’ la legge che lega i risultati ai dati.

Esempio 1:

votoFinale = (votoEs + votoTe + votoPr) / 2

Esempio 2:

Scorri l’elenco degli studenti che hanno sostenuto l’esame ed estrai tutti coloro che hanno votoFinale >= 18

algoritmo

I due esempi precedenti non sono sufficientemente precisi per essere trasformati in programma. Non viene indicato come procedere all'acquisizione dei dati, né come visualizzare il risultato.

Bisogna inoltre chiarire il significato di operazioni come ***scorri*** ed ***estrai***

Algoritmo: pseudo linguaggio

Per precisare meglio l'algoritmo si può utilizzare uno pseudo linguaggio mutuato dall'italiano e dal Pascal.

Esempio:

```
int votoEs, votoTe, votoPr, votoFi
leggi votoEs, votoTe, votoPr
votoFi = (votoEs + votoTe + votoPr) / 2
scrivi votoFi
```

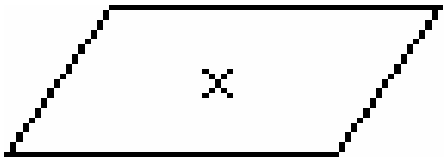
Algoritmo: flow chart

Permette di descrivere un algoritmo attraverso un linguaggio grafico.

Le istruzioni vengono rappresentate entro blocchi.

La sequenza di esecuzione delle istruzioni è rappresentata mediante l'utilizzo di frecce.

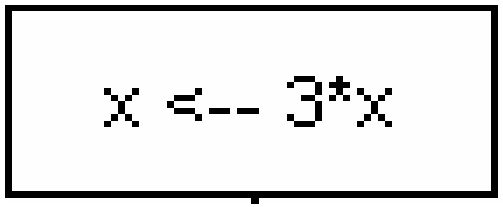
Flow chart: ingresso/uscita



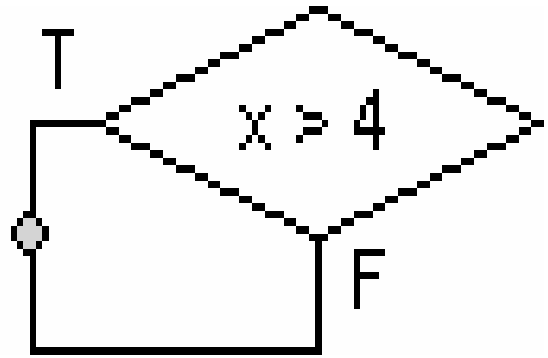
- viene richiesto un valore che viene inserito nella variabile x
- il valore contenuto della variabile x viene visualizzato

Flow chart: attività

- alla variabile x viene assegnato il valore 3x



Flow chart: scelta



.....

.....

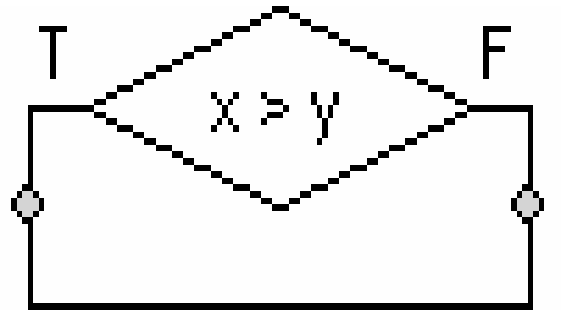
se x>4 allora

azione ramo true

if (x > 4)

azione-t

Flow chart: scelta



se x>y allora

azione ramo true

altrimenti

azione ramo false

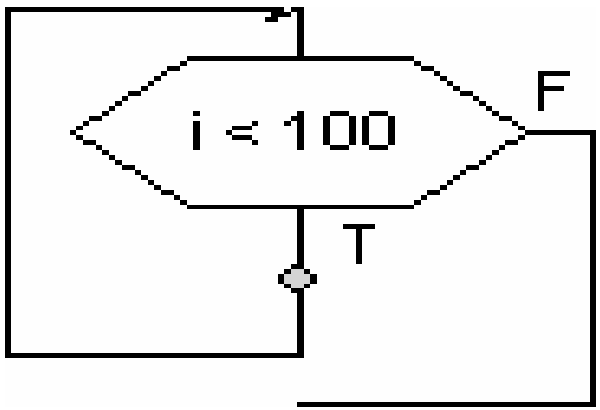
if(x > y)

azione-t

else

azione-f

Flow chart: ciclo while



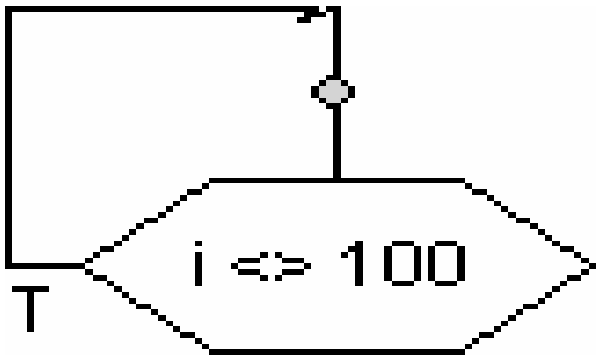
mentre i<100

esegui azione sul ramo true

while (i < 100)

azione-t

Flow chart: ciclo do-while



Ripeti

azione

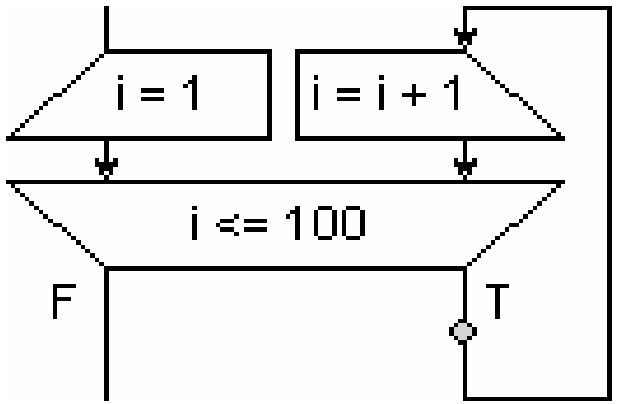
mentre i diverso 100

do {

azione

} while (i != 100);

Flow chart: ciclo for



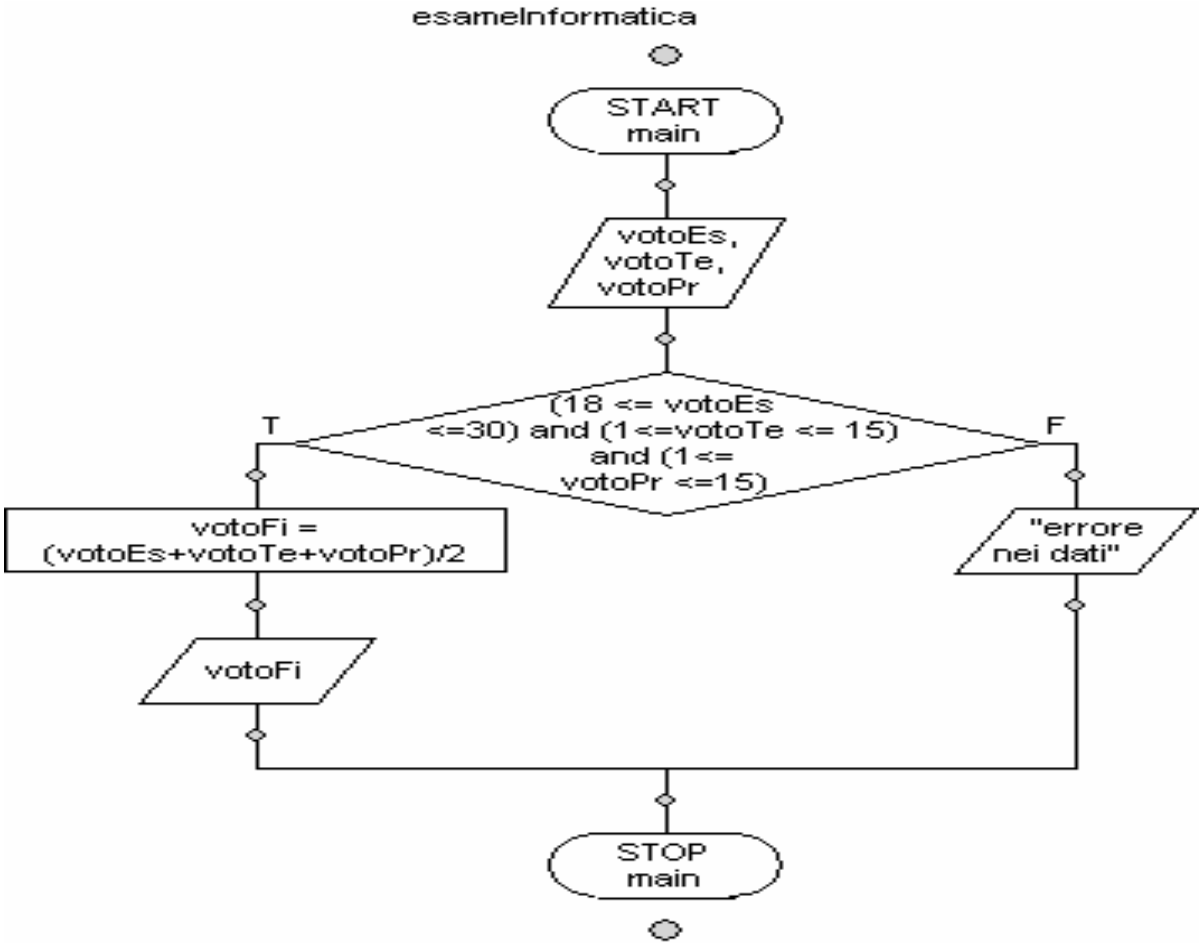
Per i che va da 1 a 100 fai

azione t

for(i=1; i<=100; ++i)

azione-t

flow chart: voto esame informatica



programma

"*I programmi sono formulazioni concrete di algoritmi astratti, che si basano sulle particolari rappresentazioni e strutture dei dati*". (N. Wirth)

"*Non si possono prendere decisioni sulla struttura dei dati senza conoscenza degli algoritmi che verranno applicati, e … la struttura e la scelta degli algoritmi dipendono fortemente dalla struttura dei dati sottostante*." (Hoare)

programma

Nel corso impareremo:

1. ad utilizzare le strutture dati elementari array e struct.
2. ad utilizzare il file sequenziale.
3. Ad utilizzare alcuni algoritmi fondamentali di ricerca e ordinamento.

Il tutto utilizzando il C come linguaggio di programmazione.

Programma: voto esame di informatica

```
#include <stdio.h>

int main() {
        int votoEs, votoTe, votoPr, votoFi;

        scanf("%d%d%d", &votoEs, &votoTe, &votoPr);

        if( ((18<=votoEs) && (votoEs<=30)) &&
           (( 1<=votoTe) && (votoTe<=15)) &&
           (( 1<=votoPr) && (votoPr<=15)) ) {
                votoFi = (votoEs + votoTe + votoPr) / 2;
                printf("Voto finale: %d\n", votoFi );
        } else
                printf("Errore nei dati.\n");

        return 0;
}
```

Esempio 2

Problema: determinare la prima potenza del 2 maggiore di x.

L'obiettivo è chiaramente espresso dal testo ma si deve decidere se con maggiore si intende strettamente maggiore o maggiore-uguale.

Il risultato può essere:

- L'esponente da assegnare al 2.
- Il valore della potenza.

Il dato è x, numero positivo. Intero?

Esempio 2

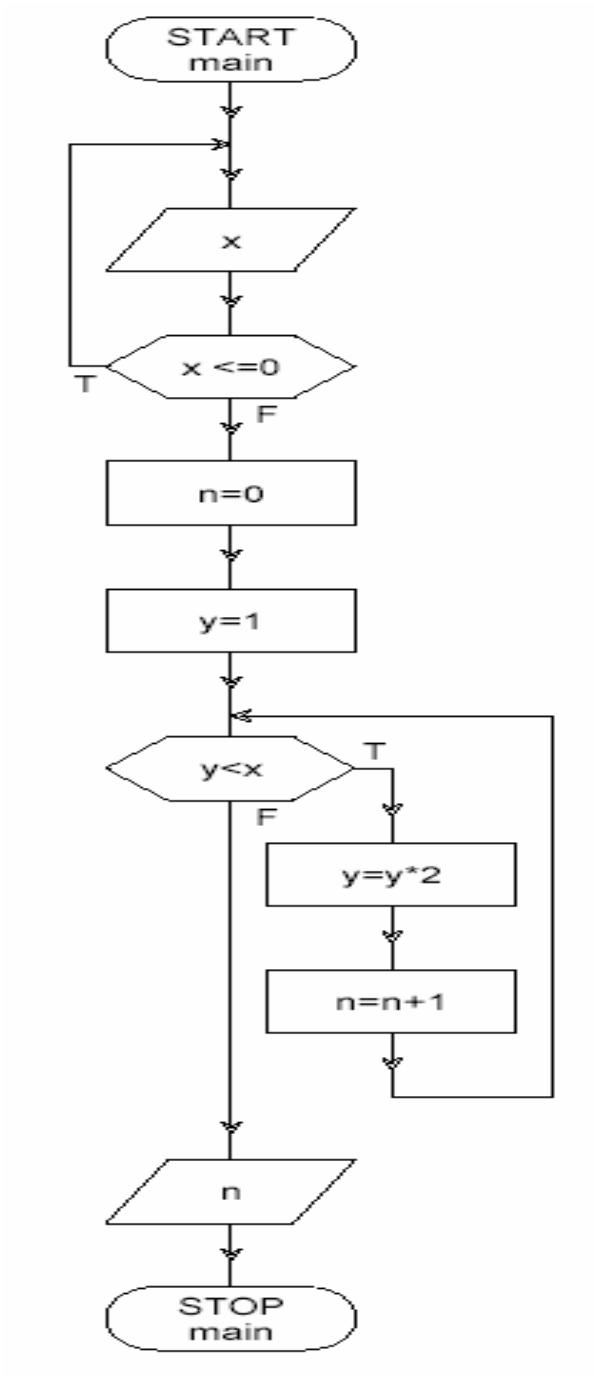
Riformuliamo il testo dell'esercizio.

Determinare l'esponente n più piccolo che bisogna assegnare a 2 in modo tale che $2^n >= x$

Con x numero intero positivo.

Risultato: n

Dato: x>0



Esempio 2

```
void main () {
    int x, y, n;
    do {
          scanf("%d", &x);
     } while (x<0);
    n=0;
    y=1;
    while (y<x) {
          y=y*2;
          n=n+1;
    }
    printf("%d minore o uguale2 ^ %d\n", x, n);
    return 0;
}
```

Esempio 3

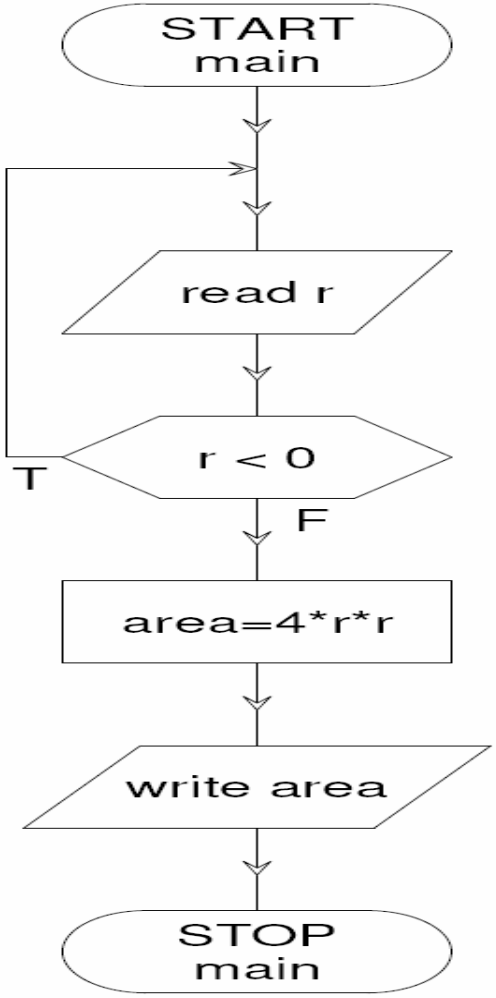
Calcolare l'area di un quadrato circoscritto ad una circonferenza.

Obiettivo: chiaramente espresso. Calcolare l'area.

Risultato: area

Dato: la circonferenza, identificata dal raggio. Il raggio è positivo.

Esempio 3




```
int main ( void )
{
        float r, area;
        do {
                scanf("%f", &r);
        } while (r < 0);
        area = 4 * r * r;
        printf("%f", area);

        return 0;
}
```

Esempio 4

Classificare un triangolo in base ai lati.

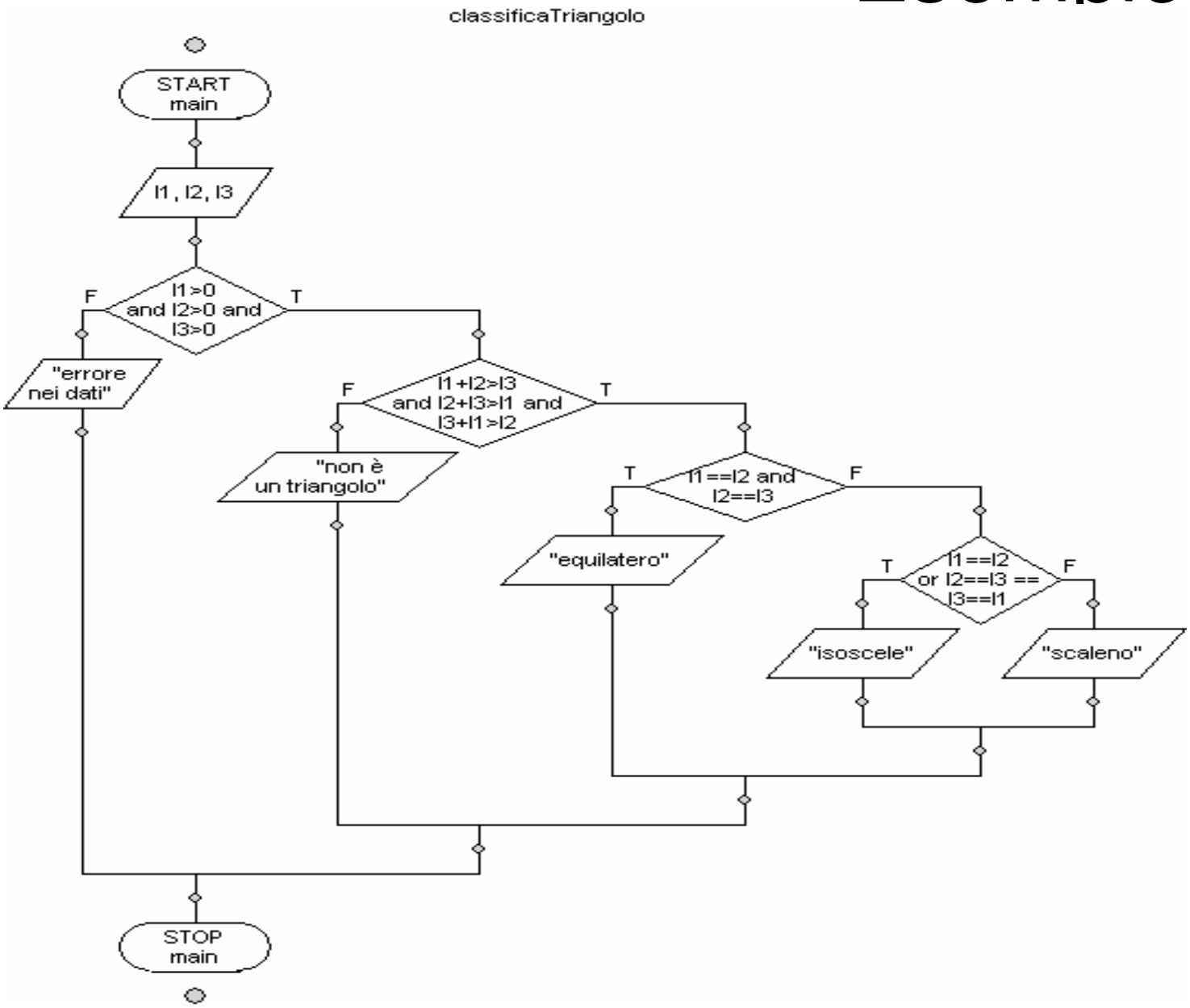
Obiettivo: determinare il tipo di triangolo

Risultato: messaggio opportuno

Dati: le misure dei tre lati.

Limiti: numeri positivi.

Esempio 4


classificaTriangolo
START
main
l1 , l2, l3
l1>0
and l2>0 and
l3>0
F
T
"errore
nei dati"
l1+l2>l3
and l2+l3>l1 and
l3+l1>l2
F
T
"non è
un triangolo"
l1==l2 and
l2==l3
T
F
"equilatero"
l1==l2
or l2==l3 ==
l3==l1
T
F
"isoscele"
"scaleno"
STOP
main

Esempio 4

```
#include <stdio.h>

int main() {
        float n1, n2, n3;

        printf("ok. ");
        scanf("%f%f%f", &n1, &n2, &n3);
        if((n1 > 0) && (n2 > 0) && (n3 > 0))
                if ((n1+n2>n3) && (n2+n3>n1) && (n3+n1>n2))
                        if((l1 == l2) && (l2 == l3)) printf("Triangolo equilatero \n");
                        else if((l1 == l2) || (l2 == l3) || (l3 == l1)) printf("Triangolo isoscele \n");
                        else printf("Triangolo scaleno \n");
                esle printf("Non è un triangolo! \n");
        } else printf("Non è un triangolo! \n");

        return 0;
}
```

Cosa fa questo programma?

```
int main() {
    int counter, grade,  total;
    int average;
    total = 0; counter = 1;
    while ( counter <= 10 ) {
          printf( "Enter grade: " );
          scanf( "%d", &grade );
          total = total + grade;
          counter = counter + 1;
    }
    average = total / 10;
    printf( "Class average is %d\n", average );
    return 0;
}
```

Cosa fa questo programma?

```
int main() {
        int counter, grade, total;
        float average;
        total = 0; counter = 0;
        printf( "Enter grade, -1 to end: " );
        scanf( "%d", &grade );
        while ( grade != -1 ) {
                total = total + grade;
                counter = counter + 1;
                printf( "Enter grade, -1 to end: " );
                scanf("%d", &grade);
        }
        if ( counter != 0 ) {
                average = ( float ) total / counter;
                printf( "Class average is %.2f\n", average );
         } else
                printf( "No grades were entered\n" );
        return 0;
}
```