Corso di

Fondamenti di Programmazione

canale E-O

Tiziana Calamoneri

Il controllo del programma in C

DD Cap. 3, pp.91-130

Un esempio per iniziare…

```
/* calcolo di 8! */
#include <stdio.h>
#define C 8
int main()
{
   int i=1; int fatt=1;
   while (i<=C)
   {
      fatt*=i;
      i++;
   }
printf("8!=%d", fatt);
return 0;
}
```

inizializzazione

calcolo

conclusione

prima di proseguire, parliamo ancora della `printf()`...

La maggior parte degli algoritmi ha:

1. una parte di inizializzazione
2. una parte di calcolo
3. una parte conclusiva

Altri cenni su printf()

`printf("8!=%d", fatt);`

Questa `printf()` ha due argomenti, una stringa tra apici ed una lista di variabili.

- Nella stringa, `%d` indica che i dati da stampare sono interi decimali;
- `fatt` è la variabile il cui contenuto deve essere stampato esattamente dove si trova il `%d` nella stringa;

Alcuni caratteri di conversione

`%d` intero decimale	`%c` carattere
`%f` reale (notaz. decimale)	`%s` stringa
`%e` reale (notaz. esponenz.)	`%l` davanti a d o l

Esercizi 8. (printf)

- Scrivere un programma che prenda in input due interi che rappresentano le lunghezze di cateti di un triangolo rettangolo e restituisca in output la lunghezza dell'ipotenusa
- Scrivere un programma che stampi il proprio nome e cognome: a. su una riga; b. su due righe; c. in un riquadro.
- Scrivere un programma che stampi una tabella delle potenze graficamente ordinata:

Intero	Quadrato	Cubo	Quarta	Quinta
1	1	1	1	1
2	4	8	16	32
…	…	…		

- Scrivere un programma che stampi la seguente figura, dove le dimensioni (# di righe e # di asterischi per riga) siano date in input:

```
*****
 *****
*****
 *****
*****
```

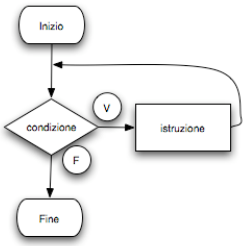
il ciclo while (1)

- Il ciclo `while`:

  viene testata la condizione per la ripetizione del ciclo; se essa è verificata il corpo del ciclo viene eseguito, altrimenti il controllo va all'istruzione successiva al ciclo while.

- Formato:

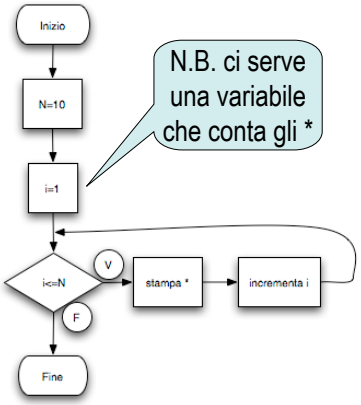
```
while(condizione)
  {
    istruzione;
  }
```



il ciclo while (2)

Problema: data una costante N, stampare N asterischi di fila

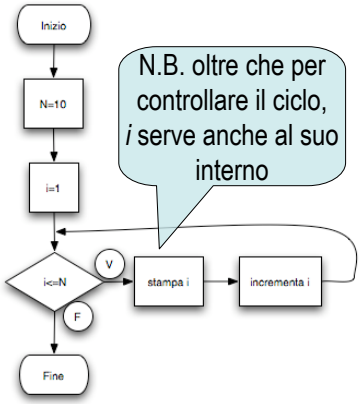
```
#include <stdio.h>
#define N 10
int main()
{
int i=1;
while(i<=N)
{
   printf("*");
         i++;
}
}
```



il ciclo while (3)

Problema: data una costante N, stampare tutti i numeri tra 1 ed N

```
#include <stdio.h>
#define N 10
int main()
{
int i=1;
while(i<=N)
   {
      printf("%d ", i);
      i++;
   }
}
```



Osserviamo…

… il ciclo while del seguente esempio:

```
i=1; fatt=1;
while (i<=C)
    {
      fatt*=i;
      i++;
    }
```

Alcuni programmatori amano compattare il codice:

```
i=0; fatt=1;
while (++i<=C)
fatt*=i;
```

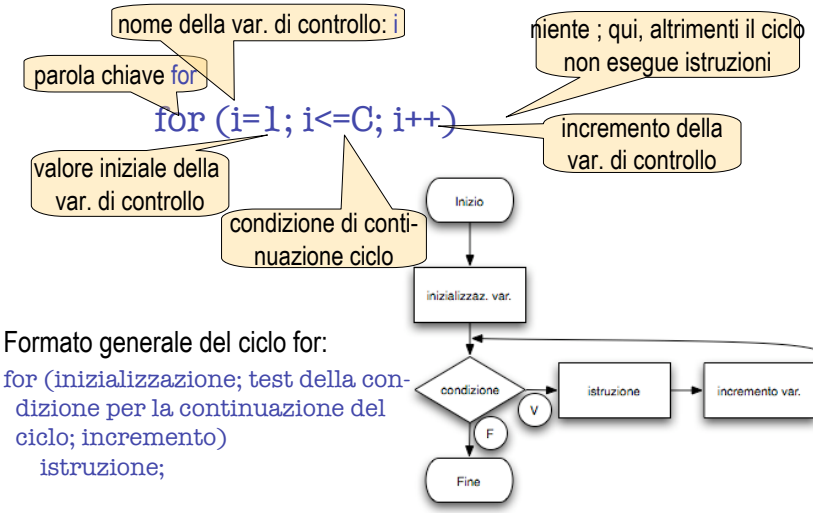
(attenzione: questa volta i parte da 0)

Altri costrutti per dire la stessa cosa:

```
fatt=1;
for (i=1; i<=C; i++)
fatt*=i;
```

```
i=1; fatt=1;
do
{
   fatt*=i;
   i++;
}while (i<=C);
```

Il ciclo for (1) p.94



Formato generale del ciclo for:

```
for (inizializzazione; test della con-
  dizione per la continuazione del
  ciclo; incremento)
    istruzione;
```

Il ciclo for (2)

- Note sul ciclo for:
  - L'inizializzazione, la condizione di continuazione del ciclo e l'incremento possono contenere espressioni aritmetiche.
    Esempio:
    ```
    x=2; y=10;
    for ( j = x; j <= 4 * x * y; j += y / x )
    ```
    è equivalente a:
    ```
    for ( j = 2; j <= 80; j += 5 )
    ```
  - L'"incremento" può essere negativo (decremento)
  - Se la cond. di continuazione del ciclo è inizialmente falsa il corpo del ciclo for non viene eseguito, e il controllo procede con l'istruzione seguente al ciclo for
  - La variabile di controllo è spesso usata all'interno del corpo, ma non è necessario
  - I cicli for e while sono assolutamente equivalenti

Un esempio

Problema: eseguire la moltiplicazione 8x6 simulandola con l'addizione (8+8+…+8).

```
somma=0
considera un contatore i
i=1
finchè (i non è > di 6)
   somma=somma+8
   i=i+1
stampa somma
```

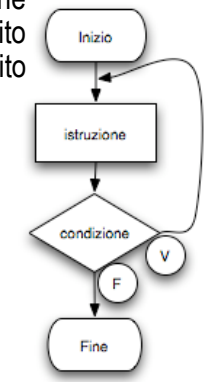
```
   /* calcolo di 8x6*/
#include <stdio.h>
#define C1 6
#define C2 8

int main()
{
   int somma=0; int i;
   for(i=1; i<=C1; i++)
      somma+=C2;
   printf("6x8=%d", somma);
   return 0;
}
```

Il ciclo do…while p.108

- Il ciclo do...while:
  - è simile al ciclo while
  - la condizione per la ripetizione del ciclo viene testata dopo che il corpo del ciclo è stato eseguito una prima volta, quindi tale corpo viene eseguito sempre almeno una volta.
- Formato:

```
do
  {
    istruzione;
  } while(condizione);
```



Un esempio

Problema: eseguire la moltiplicazione 8x6 simulandola con l'addizione (8+8+…+8).

```
somma=0
considera un contatore i
i=1
finchè (i non è > di 6)
   somma=somma+8
   i=i+1
stampa somma
```

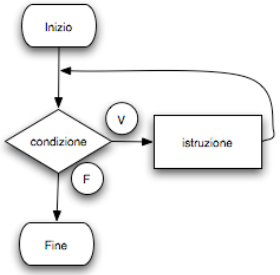
```
  /* calcolo di 8x6*/
#include <stdio.h>
#define C1 6
#define C2 8

int main()
{
   int somma=0; int i=1;
   do
   {
      somma+=C2;
      i++;
   }
   while (i<=C1);
   printf("6x8=%d", somma);
   return 0;
}
```

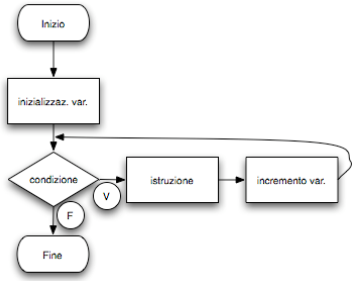
Attenzione: se C2=0 mentre C1 no, questo programma non funziona correttamente!

Ricapitoliamo: i tre cicli

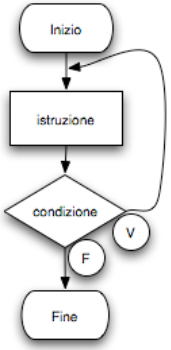
ciclo while



ciclo for

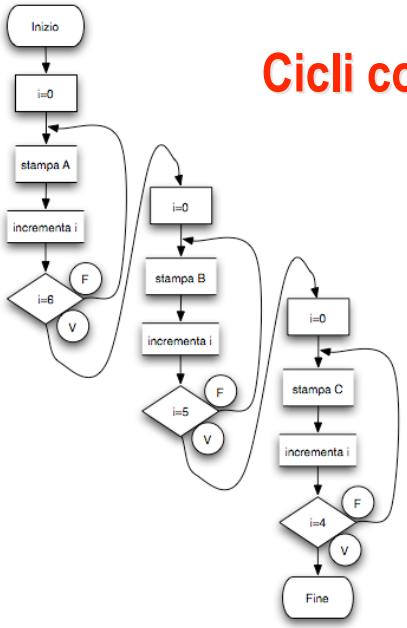


ciclo do…while



N.B. similitudini e differenze tra i tre cicli…

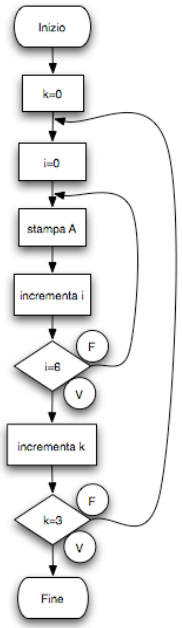
Cicli concatenati



Problema:
stampare l'oggetto seguente:
AAAAAA
BBBBB
CCCC

i cicli sono uno di fila all'altro

Cicli nidificati



Problema:
stampare l'oggetto seguente:
AAAAAA
AAAAAA
AAAAAA

i cicli sono uno dentro l'altro

Esercizi 9. (cicli)

Diagramma a blocchi, pseudocodice e traduzione in 3 programmi C, uno per ogni ciclo:

- visualizzare gli interi pari da 2 a 100
- visualizzare gli interi da 50 a 10
- stampare un rettangolo di 10 righe e 20 colonne di *
- stampare l'uno sotto l'altro i seguenti disegni:

```
*          ******     ******          *
**         *****      *****          **
***        ****       ****          ***
****       ***        ***          ****
*****      **         **          *****
******     *          *          ******
```

Lettura dell'input (1)

- Fin ora abbiamo sempre usato numeri fissi (costanti).
- Che succede se vogliamo inserire un valore da tastiera?
- istruzione scanf()
  - come printf() ha due argomenti, una stringa tra apici ed una lista di variabili
  - nella stringa compaiono i caratteri di controllo che specificano il tipo di input
  - nella lista compaiono le variabili in cui memorizziamo i valori
  - ogni nome di variabile è preceduto da & (operatore di indirizzo) che, al momento, non ha molto significato (ma lo avrà in seguito!). Per ora ricordiamo di metterlo sempre prima delle variabili da stampare.

Lettura dell'input (2)

Problema: Sommare due interi dati in input

leggi il primo intero, a
leggi il secondo intero, b
esegui a+b
stampa la somma

```
/* somma di 2 interi*/
#include <stdio.h>
int main()
{
   int a; int b; int s;
   scanf("%d",&a);
   scanf("%d",&b);
   s=a+b;
   printf("%d+%d=%d", a, b, s);
}
```

Meglio:
printf("%d+%d=%d", a, b, a+b);
e si evita la variabile s

Lettura dell'input (3)

Problema: Calcolare l'area di un cerchio di raggio dato in input

leggi il raggio, r
calcola r*r*π
stampa l'area trovata

```
/* area del cerchio*/
#include <stdio.h>
#define PI 3.14159
int main()
{
   float r; float area;
   printf("raggio?");
   scanf("%f",&r);
   area=r*r*PI;
   printf("area=%f", area);
}
```

Attenzione: la variabile area si può evitare:
printf("area=%f", r*r*PI);

Lettura dell'input (4)

Correttezza? Non è garantita se il raggio inserito è negat.

leggi il raggio r
se r<0
  allora stampa 0
  altrimenti
    calcola r*r*π
    stampa l'area trovata

```
/* area del cerchio con gestione errori*/
#include <stdio.h>
#define PI 3.14159
int main()
{
   float r;
   printf("raggio?");
   scanf("%f",&r);
   if (raggio<0) printf("area=0");
   else printf("area=%f", area=r*r*PI);
}
```

Lettura dell'input (5)

Problema: Calcolare l'area di più cerchi di raggi dati in input

leggi il numero di cerchi, n
ripeti n volte:
  leggi il raggio, r
  se r<0 stampa 0
  altrimenti stampa r*r*π

```
/* area di più cerchi*/
#include <stdio.h>
#define PI 3.14159
int main()
{
   float r; int n; int i;
   printf("quanti cerchi?");
   scanf("%d", &n);
   for (i=1; i<=n; i++)
   {   printf("raggio?");
       scanf("%f",&r);
       if (r<0)
          printf("area=0 \n");
       else printf("area=%f \n", r*r*PI);
   }
}
```

Esercizi 10. (cicli e scanf())

Diagramma a blocchi, pseudocodice e traduzione in 3 programmi C, uno per ogni ciclo:

- Prendere in input un intero *n* e stampare tutti i numeri da 1 a *n* intervallati da un * (es. 1*2*3*4*5)
- Prendere in input 2 interi *a* e *b* e calcolare axb simulando con la somma, *a+a+…+a* (*b* volte)
- Prendere in input un intero *n* e stampare un quadrato di "A" di dimensione *nxn*
- Prendere in input 20 interi e calcolarne la media (senza usare 20 variabili!). E se invece di 20 ci fosse una variabile?
- Prendere in input 20 interi compresi tra 1 e 3 e dire quanti 1, quanti 2 e quanti 3 ci sono nella sequenza (come sopra).

Risolviamo…

attenzione: l'ultimo valore va trattato a parte

Problema: prendere in input un intero n e stampare tutti i numeri da 1 a n intervallati da un *

leggi n;
se n<1 stampa: errore
altrimenti
  per ciascun valore i da 1 a n-1
    stampa i*
  stampa n

```
i=1;
while (i <n)
  {printf("%d*", i);
   i++;
  }
printf("%d", n);
```

```
i=1;
do
  {printf("%d*", i);
   i++;
  }
while (i <n);
printf("%d", n);
```

```
#include <stdio.h>
int main()
{
 int n;
 scanf("%d", &n);
 if (n==0) || (n==1)
   printf("errore");
 else
   {
   for (int i=1; i<n; i++)
      printf("%d*", i);
   printf("%d",n);
  }
```

Valore sentinella (1)

Problema: far inserire ad uno studente i propri voti e stamparne la media

- Dobbiamo eseguire un ciclo, ma l'utente non deve preventivamente inserire il numero di esami. Come facciamo a sapere quando ha finito?
- In un ciclo, se ci sono dei valori che la variabile contatore non assume mai, uno di tali valori può essere usato per segnalare che si è arrivati alla fine della serie di valori da inserire.
- Qui, i voti vanno da 18 a 30, quindi 0 può essere usato come valore sentinella

Attenzione: non si può sempre dire!

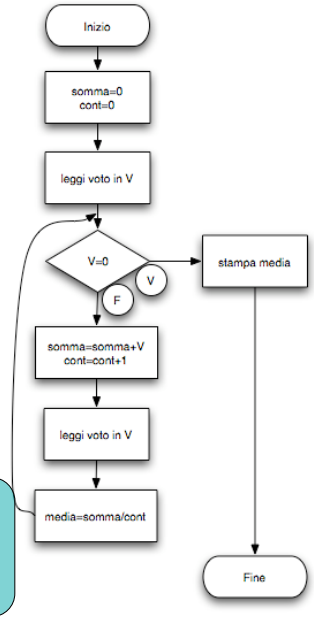
Input: 28 27 30 22 0

Valore sentinella (2)

Problema: far inserire ad uno studente i propri voti e stamparne la media

```
somma=0; cont=0;
leggi il primo voto in V
finché (V è diverso da 0)
  somma=somma+V
  cont=cont+1
  leggi il prossimo voto in V
media=somma/cont
stampa media
```

Per esercizio: tradurre in C in almeno 2 modi diversi (cicli)



Valore sentinella (3)

Problema: calcolare le aree di un numero indefinito di cerchi di raggi dati in input

valore sentinella=0

```
leggi il primo raggio, r
ripeti finché r non diventa 0:
   se r<0 stampa 0
   altrimenti stampa r*r*π
   leggi il valore del prossimo raggio, r
```

Per esercizio: tradurre in C in almeno 2 modi diversi (cicli)

L'istruzione break p.110

- L'istruzione `break` causa l'uscita immediata da un ciclo (oppure da uno switch, che vedremo tra poco)
- L'esecuzione del programma continua con la prima istruzione dopo la struttura interrotta
- L'istruzione `break` nei cicli si può sempre evitare scrivendo opportunamente la condizione di uscita dal ciclo, e può confondere la comprensione, quindi…
  E' MEGLIO EVITARLA! (non fa parte della progr. strutturata)

Esempio:

```
for (i=1; i<=10; i++)
{
  if (i==5) break;
  printf("%d ", i);
}
```

Output: 1 2 3 4

L'istruzione continue (1) p.110

- L'istruzione `continue` fa in modo che vengano ignorate le istruzioni ad essa successive nel corpo di un ciclo, SOLO PER L'ITERAZIONE CORRENTE.
- L'esecuzione del programma continua così:
  - while e do…while
    viene immediatamente eseguito il test di entrata nel ciclo e, se verificato, si rientra nel ciclo
  - for
    viene eseguita l'espressione di incremento e poi eseguito il test di entrata; se verificato si rientra nel ciclo

L'istruzione continue (2)

- L'istruzione `continue` si può sempre evitare scrivendo oppor-tunamente la condizione di uscita dal ciclo, e può confondere la comprensione, quindi…
- E' MEGLIO EVITARLA! (non fa parte della progr. strutturata)
- Si usa solo quando le istruzioni del ciclo sono così complesse che cambiare la condizione di rientro complicherebbe troppo le cose.

Esempio:
```
for (i=1; i<=10; i++)
{
  if (i==5) continue;
  printf("%d ", i);
}
```

Output: 1 2 3 4 6 7 8 9 10

L'istruzione continue (3)

Un esempio di uso (ma qui le istruzioni nel ciclo non sono complesse…

Problema: prendere N interi in input e calcolare il quadrato dei soli elementi positivi

Input: 1 -2 3 4 -5

Output: 1 9 16

```
...
int x;
for (i=1; i<=N; i++)
{
        scanf("%d", &x);
        if (x<0) continue;
        x*=x;
        printf("%d ", x);
}
```

L'istruzione di selezione multipla switch (1)

p.102

L'istruzione `switch` è utile quando una variabile o un'espres-sione viene testata per tutti i valori che essa può assumere e, a seconda del valore, sono intraprese azioni diverse.
Il formato prevede una serie di `case` ed un caso di `default` opzionale:

Formato:
```
switch(valore)
{case '1':
   azione1;
 case '2':
   azione2;
 ...
 default:
   azione_default;
}
```

L'istruzione di selezione multipla switch (2)

```
scanf("%d", &scelta);
switch(scelta)
{
   case 1:
      printf("hai scelto 1");
      break;
   case 2:
      printf("hai scelto 2");
      break;
   case 3:
   case 4:
      printf("hai scelto 3 o 4");
      break;
   default:
      printf("hai scelto un valore >4");
      break;
}
```

non servono {} anche per istruzioni multiple

possibilità di case multipli

necessario, altrimenti esegue tutti i case dopo quello in cui entra

Esercizi 11. (switch)

Diagramma a blocchi, pseudocodice e traduzione in un programma C per risolvere ciascuno dei seguenti problemi:

- Prendere in input un intero e stampare una stringa che dica se è positivo o negativo, pari o dispari.
- Prendere in input un intero da 1 a 10 e stampare il suo valore a lettere (3: tre)
- Prendere in input un intero da 1 a 12 e stampare il mese corrispondente (3: marzo)
- Prendere in input 3 valori, che rappresentano i coefficienti di un'equazione di secondo grado, e stampare i risultati dell'equazione (attenzione a tutti i casi estremali!).
- Prendere in input un intero tra 1 e 99 e stamparlo come numero romano.