



## Insegnamento di Elementi di informatica (6 CFU)

### Il linguaggio C - Parte 4

Nadia Ranaldo  
RCOST - Dipartimento di Ingegneria  
Università degli Studi del Sannio

## Istruzioni cicliche



- Consentono di realizzare cicli di elaborazione, ossia l'esecuzione ripetuta di una sequenza di istruzioni

il numero di volte per il quale viene ripetuta la esecuzione della sequenza è noto a priori

il numero di volte per il quale la sequenza viene ripetuta non è noto a priori, ma è condizionato dal verificarsi di una condizione

2

Elementi di Informatica - Linguaggio C - parte 4

## Esempi



- calcolare le paghe di tutti i dipendenti di una azienda
- noto il numero  $N$  dei dipendenti, bisogna ripetere  $N$  volte la sequenza di *calcolo della paga*
- superare l'esame di Elementi di Informatica
- la sequenza *sostenere l'esame* va eseguita e rieseguita fino a quando non si viene promossi ( $\text{voto} \geq 18$ )

3

Elementi di Informatica - Linguaggio C - parte 4

## Istruzioni cicliche



- ❖ **for**
- ❖ **while**
- ❖ **do-while**

4

Elementi di Informatica - Linguaggio C - parte 4

# Esempio

- Vogliamo visualizzare tre volte la scritta
- Ciao, mondo.

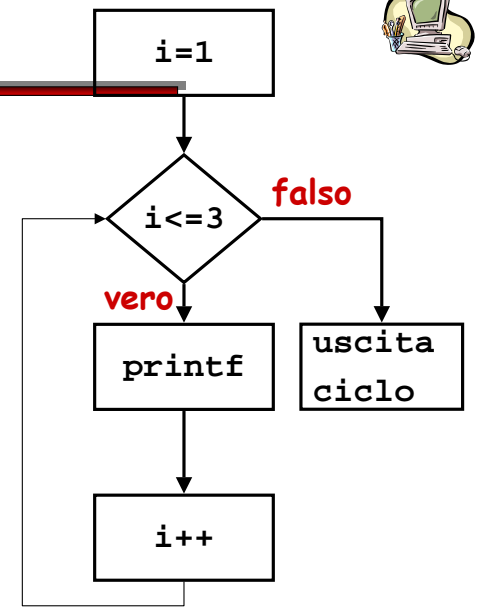
il numero di volte per il quale viene ripetuta la esecuzione della sequenza è noto a priori: 3

```
...  
printf("Ciao, mondo.\n");  
printf("Ciao, mondo.\n");  
printf("Ciao, mondo.\n");  
...
```

# Usando il for

```
#include<stdio.h>  
main()  
{  
  int i;  
  for (i=1; i<=3; i++)  
    printf("Ciao, mondo.\n");  
}
```

Variabile  
contatore



# Sintassi del for

```
for (inizializzazione; condizione; incremento)  
  istruzione
```

# Semantica

- for (**inizializzazione**; condizione; incremento) istruzione
- **inizializzazione**
  - Viene eseguita una volta sola
  - Serve per dare un valore iniziale alle variabili contatore
  - Può anche non essere presente, ed allora dopo la parentesi tonda aperta viene subito il punto e virgola



for (inizializzazione; **condizione**; incremento)  
istruzione

## • **condizione**

- Viene valutata ogni volta prima di eseguire le istruzioni del ciclo... (è la domanda!!!)
- Se **condizione** è vera si esegue ancora istruzione
- Se **condizione** è falsa si esce dal ciclo for passando all'istruzione successiva del programma
- Può anche non essere presente, ed allora lo spazio tra i due punti e virgola rimane vuoto. Il compilatore valuta vera la condizione assente, quindi continua ad eseguire istruzione → realizza un *loop infinito*

9



for (inizializzazione; condizione; **incremento**)  
istruzione

## • **incremento**

- Viene eseguita alla fine di ogni ciclo
- Modificano ad esempio le variabili **contatore**, ossia la variabile usata per contare (incremento o decremento)
- Può anche non essere presente, ed allora dopo il secondo punto e virgola viene subito la parentesi tonda chiusa

10



for (inizializzazione; condizione; incremento)  
istruzione

**inizializzazione:** eseguita una volta sola

**condizione:** valutata ogni volta prima di eseguire le istruzioni del ciclo

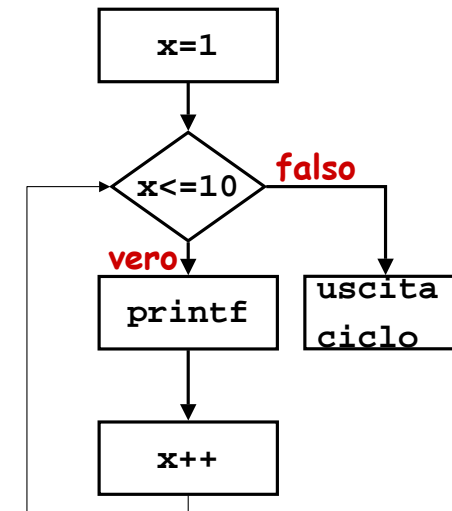
**incremento:** eseguita alla fine di ogni ciclo  
più in generale è un'espressione



11

```
{
int x;
for (x=1; x<=10; x++) printf("%d\n",x);
}
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10



x	x<=10?
1	si
2	si
3	si
4	si
5	si
6	si
7	si
8	si
9	si
10	si
11	no

12

```
{
int x;
for (x=10; x>=0; x-=2) printf("%d\n",x);
}
```

10  
8  
6  
4  
2  
0

x	x>=0
10	si
8	si
6	si
4	si
2	si
0	si
-2	no

## Esempio: loop infinito



```
{
int x;
for ( ; ; ) printf("loop");
}
{
int x;
for ( x=0; ; ) printf("loop");
}
{
int x;
for ( x=0; x==0; ) printf("loop");
}
```

Diagram showing three code snippets with arrows pointing to a vertical stack of "looplooplooplooplooploop" text, illustrating infinite loops.

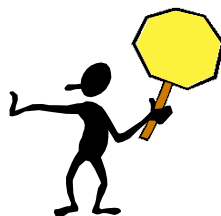
## Attenzione!!!



Iterazione infinita

```
for (i=5; i>=5; i++) ...
```

Compilatore non segnala errore



## Osservazioni



- Poiché la condizione viene valutata prima di ogni ciclo, il for permette anche di non eseguire nemmeno una volta il ciclo

### • Esempio

```
{
int x;
for (x=9; x<6; x++) printf("%d\n", x);
}
```

Non stampa nulla

## Osservazioni (cont.)



- Inizializzazione ed incremento, nella sintassi del `for`, possono contenere più istruzioni, che dovranno essere separate da virgola.

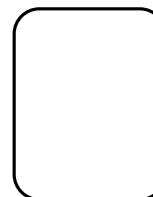
### • Esempio

```
{
int x, y;
for (x=0, y=0; x+y<10; x++, y+=3)
printf("%d\n", x+y);
}
```

17

```
{
int x, y;
for (x=0,y=0; x+y<10; x++,y+=3)
printf("%d\n", x+y);
}
```

Cosa stampa?



x	y	x+y<10
0	0	si
1	3	si
2	6	si
3	9	no

18

## Esempio di uso del `for`



Calcolare la somma di 5 numeri interi immessi dall'utente

```
#include <stdio.h>
main()
{
int i, somma, numero;
somma=0;
for (i=1; i<=5; i++)
{
printf ("Dammi il numero: ");
scanf("%d", &numero);
somma=somma + numero;
}
printf("Somma = %d\n", somma);
}
```

19

## Esempio (cont.)



Utile far apparire il numero d'ordine d'inserimento:

```
#include <stdio.h>
main()
{
int i, somma, numero;
somma=0;
for (i=1; i<=5; i++)
{
printf ("Dammi il numero %d: ", i);
scanf("%d", &numero);
somma=somma + numero;
}
printf("Somma = %d\n", somma);
}
```

20

## Esercizio



```
#include <stdio.h>
main()
{
  int i, n=3;
  for (i = 0; i < n; i++)
    printf("*");

  for (i = 0; i < n; i++)
    printf("!");

}
```

\*\*\*!!!

21

## Cosa stampa?



```
#include <stdio.h>
main()
{
  int x;
  for (x=1; x<=3; x++)
  {
    printf("%d\n", x);
    x=x+2;
  }
}
```



22

## Quando usare for?



- Per realizzare cicli basati su una variabile che si incrementa e diminuisce di una certa quantità ogni volta che il ciclo viene eseguito.
- Quando la variabile raggiunge un determinato valore, l'esecuzione del ciclo viene interrotta

23

## Esercizi



1. Scrivere un programma che richiede all'utente un numero naturale  $n$  e calcola la somma dei primi  $n$  numeri naturali.
2. Scrivere un programma che richiede all'utente un numero naturale  $n$  e calcola la somma di  $n$  numeri inseriti dall'utente.
3. Scrivere un programma che richiede all'utente un numero naturale  $n$  e calcola il massimo degli  $n$  numeri inseriti dall'utente.

24

## Esercizi (cont.)



4. Scrivere un programma che richiede all'utente un naturale  $n$  e calcola il fattoriale di  $n$ , indicato con  $n!$

$$0! = 1$$

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 2 \cdot 1 \quad n > 0$$

### Esempio:

$$4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$$

$$6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$$

25

## Somma dei primi $n$ numeri (Es. 1)



```
#include <stdio.h>
main()
{
    int n, somma, i;
    printf("Inserisci n: ");
    scanf("%d", &n);
    somma=0;
    for (i=1; i<=n; i++)
        somma=somma+i;
    printf("La somma dei primi n num = %d", somma);
}
```

26

## Somma dei primi $n$ numeri



**Gauss**  $n(n+1)/2$

```
#include <stdio.h>
main()
{
    int n;
    printf("Inserisci n: ");
    scanf("%d", &n);
    printf("Somma = %d", n*(n+1)/2);
}
```

27

## Somma di $n$ numeri (Es. 2)



```
#include <stdio.h>
main()
{ int n, somma=0, num, i;
  printf("Inserisci n: ");
  scanf("%d", &n);
  printf("Ora inizia l'inserimento di %d numeri\n", n);
  for (i=1; i<=n; i++)
      { printf("Inserisci il numero %d: ", i);
        scanf("%d", &num);
        somma=somma+num;
      }
  printf("La somma = %d", somma);
}
```

28

## Massimo di n numeri (Es. 3)



Se uso int

Minimo =  $-2^{31}$  = -2147483648

Massimo =  $2^{31} - 1$  = 2147483647

```
#include <limits.h>
```

```
INT_MIN
```

```
INT_MAX
```

29

## Massimo di n numeri



```
#include <stdio.h>
#include <limits.h>
main()
{ int n, max=INT_MIN, num, i;
  printf("Inserisci n: ");
  scanf("%d", &n);
  printf("Ora inizia l'inserimento di %d numeri\n", n);
  for (i=1; i<=n; i++)
    { printf("Inserisci il numero %d: ", i);
      scanf("%d", &num);
      if (max < num)
        max=num;
    }
  printf("Il massimo= %d", max);
}
```

30

## Osservazione



- Soluzione alternativa ad inizializzare `max` al minimo numero negativo accettato da una variabile intera:
- inizializzare `max` al primo numero letto dall'esterno, all'inizio del ciclo, se  $n > 0$

31

## Fattoriale di n (Es. 4)



```
#include <stdio.h>
main()
{
  int n, fatt, i;
  printf("Inserisci n: ");
  scanf("%d", &n);
  fatt=1;
  for (i=n; i>=1; i--)
    fatt=fatt*i;
  printf("Il fattoriale di n = %d", fatt);
}
```

32

# while



- Permette di ottenere la ripetizione ciclica di una istruzione sotto il controllo di una condizione di terminazione

33

# while



- Sintassi

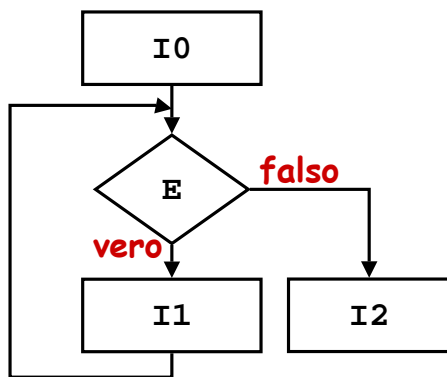
**while** (espressione) istruzione

- Semantica

- **espressione** deve essere valutata ogni volta prima di eseguire istruzione
- se la valutazione di **espressione** è vera allora **istruzione** viene eseguita, altrimenti no, e si esce dal ciclo

34

# while: diagramma di flusso



```

I0;
while (E) I1;
I2;
  
```

35

# Analisi



- Poiché la valutazione dell'espressione è effettuata all'inizio, il ciclo stesso può essere eseguito:
  - ❖ zero volte
  - ❖ un numero finito di volte
  - ❖ infinite volte

36

# Esempio

```
int x=12;
int y= 1;
while(y<x)
    y=y*2;
```

- Quanto vale *y* alla fine dell'esecuzione



Y=

# Soluzione

```
int x=12;
int y=1;
while(y<x)
    y=y*2;
```

y	x
<del>1</del>	12
<del>2</del>	
<del>4</del>	
<del>8</del>	
16	

# Errori frequenti: ripetizioni infinite

## Esempi

```
...
while (x > 0)
a = b + c;
```

poiché il valore di *x* non viene in alcun caso modificato.....

```
...
while(i > n) {
...
i = i + 1; }
```

*i* varia, ma aumenta .....

# Esempio

```
#include<stdio.h>
main()
{
int x=0;
while (x==0)
    printf("loop");
}
```

loop infinito

looplooplooplooplooplooploop  
looplooplooplooplooplooploop  
looplooplooplooplooplooploop  
looplooplooplooplooplooploop  
looplooplooplooplooplooploop  
looplooplooplooplooplooploop  
looplooplooplooplooplooploop

# Convergenza del ciclo



- assicurarsi, in prima istanza, che nel corpo del `while` venga alterato il valore di almeno una delle informazioni coinvolte in `E`
- assicurarsi, in ogni caso, che in un numero finito di iterazioni, `E` diventi falsa
- assicurarsi che a tutte le informazioni usate nell'espressione `E` sia stato assegnato un valore prima del `while`

41

# Esempio



- Somma numeri interi forniti in ingresso dalla tastiera finché non viene dato il valore zero

42

# Soluzione



```
#include<stdio.h>
main()
{int num=1;
 int somma=0;
 while (num !=0)
 {
 printf("Dammi un numero: ");
 scanf("%d", &num);
 somma=somma+num;
 }
 printf("La somma = %d%", somma);
}
```

Alla variabile `num` si è assegnato il valore 1 per far in modo che il ciclo venga eseguito almeno una volta, ovviamente qualunque numero diverso da zero va bene

43

# Soluzione alternativa



```
#include<stdio.h>
main()
{int num;
 int somma=0;
 printf("Dammi un numero: ");
 scanf("%d", &num);
 while (num !=0)
 {
 somma=somma+num;
 printf("Dammi un numero: ");
 scanf("%d", &num);
 }
 printf("La somma = %d", somma);
}
```

Alla variabile `num` si è assegnato il primo numero inserito dall'utente fuori dal ciclo

44

## Relazione tra for e while



```
for (iniz; cond; inc)
{
    istruzioni;
}
```

.... è equivalente a:

```
iniz;
while (cond)
{
    istruzioni;
    inc;
}
```

45

## Calcolare il massimo comun divisore



Algoritmo di Euclide:

Siano dati due numeri  $x$  e  $y > 0$

Se  $x==y \rightarrow \text{MCD}(x,y)=x$

Se  $x>y \rightarrow \text{MCD}(x-y,y)$

Se  $x<y \rightarrow \text{MCD}(x,y-x)$

• Esempio

15 9 restituisce 3

46

## Soluzione



```
#include<stdio.h>
main()
{
    int x, y, mcd;
    printf("Digita due numeri:");
    scanf("%d%d", &x, &y);
    while (x!=y)
        if (x>y) x=x-y;
        else y=y-x;
    mcd = x;
    printf("il mdc =%d", mcd);
}
```

47

## Esercizio



• Siano  $i$  e  $x$  variabili intere. Data l'istruzione:

```
for(i=1; i<=x; i++)
    printf("%d\n", i*x);
```

• scrivere un programma equivalente usando il while

48



```

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

```

49



- **for** e **while** controllano la condizione di terminazione all'inizio del ciclo
- **do-while** controlla la condizione al termine di ogni iterazione
- consente di eseguire un ciclo condizionale da 1 ad infinite volte

50



## • Sintassi

```

do
    istruzione
while (espressione);

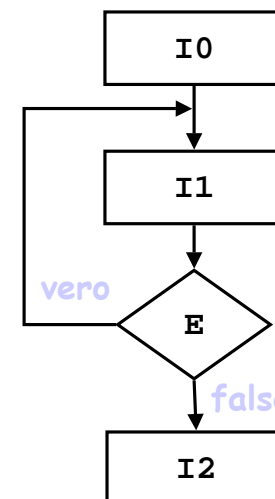
```

## • Semantica

- **espressione** deve essere valutata ogni volta dopo aver eseguito istruzione

- se la valutazione di **espressione** è vera allora il ciclo viene ripetuto, cioè si riesegue istruzione, altrimenti si esce dal ciclo

51



```

I0;
do {
    I1;
} while (E);
I2

```

52

## Cosa stampa?



```
#include<stdio.h>
main()
{
  int n=12, k=5;
  do{
    n=n/2;
    k=k-1;}
  while(n>=k);

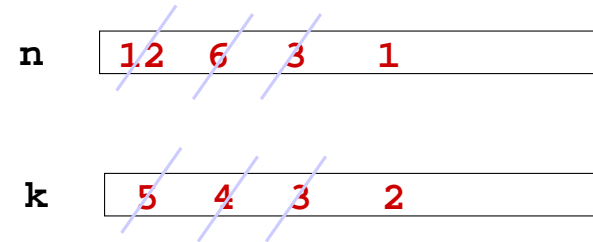
  printf("%d %d\n", n, k);
}
```

1 2

53

## Perché?

```
int n=12, k=5;
do{
  n=n/2;
  k=k-1;}
while(n>=k);
printf("%d %d\n", n, k);
```



54

## Esempio



```
int num;

do {
  scanf("%d", & num);
} while (num > 10);
```

Legge numeri finché  
non viene inserito un  
numero <=10

55

## Riprendiamo l'esempio



- Somma di numeri interi forniti in ingresso dalla tastiera finché non viene dato lo zero

56

## Soluzione con do-while



```
#include<stdio.h>
main()
{int num;
 int somma=0;
 do {
  printf("Dammi un numero: ");
  scanf("%d", &num);
  somma=somma+num;
 } while (num !=0);

 printf("La somma = %d", somma);
}
```

57

## Relazione tra do-while e while



```
do {
  istruzioni;
}
while (espressione);
```

.... è equivalente a:

```
istruzioni;
while (espressione)
{
  istruzioni;
}
```

58

## Esercizio



- Scrivere un programma che legge un carattere ed aspetta che l'utente digiti s oppure n. In questo caso stampa OK.

59

## Problema???



```
#include<stdio.h>
main()
{
  char car;
  do {
    printf("rispondi s oppure n: \n");
    scanf("%c", &car);
  } while ((car!='s') && (car!='n'));
  printf("OK");
}
```

60

## Immissione di più caratteri



Se il programma richiede l'immissione di più valori in tempi diversi, l'inserimento di un carattere potrebbe costituire un problema, dato che la digitazione del tasto di Invio da tastiera corrisponde a un carattere accettato da

```
scanf("%c", ..)
```

In tal caso verrà utilizzata un'opportuna ulteriore lettura di un carattere in una variabile ausiliaria tramite un'istruzione del tipo

```
scanf("%c", &pausa);
```

1

## Soluzione



```
#include<stdio.h>
main()
{
    char car, pausa;
    do {
        printf("rispondi s oppure n: \n");
        scanf("%c", &car);
        scanf("%c", &pausa);
    } while ((car!='s') && (car!='n'));
    printf("OK");
}
```

62

## Suggerimenti



- ❖ Evitare programmi prolissi
- ❖ Non usare variabili più del necessario
- ❖ Memorizzare i risultati di computazioni intermedie quando tali risultati sono riusati

63

## Esempio



- Programma che calcola il rapporto tra la somma e la differenza di due variabili e lo memorizza in  $r$

prolisso	semplice
<pre>somma = m+n;</pre>	
<pre>diff = m-n;</pre>	<pre>r = (m+n)/(m-n)</pre>
<pre>r = somma/diff</pre>	

può essere utile per maggiore comprensione

64



- ❖ Se un'espressione compare in più punti è meglio memorizzarla in una variabile piuttosto che ricalcolarla

```
s = (m+n)+5*(m+n)*(m+n);    sum = m+n;
t = (m+n)/(m-n);            s = sum+5*sum*sum;
...                          t = sum/(m-n);
u = 3*(m+n);                ...
                             u = 3*sum;
```



65



```
while ((i<m-n+1) && (ris<max))
    ris=ris*i;
```

- m e n non vengono alterati nel corpo del ciclo.



```
sup = m-n+1;
while ((i<sup) && (ris<max))
    ris=ris*i;
```

Evita di ricalcolare  
sempre l'espressione  
m-n+1

66

## Numero magico



- Generare un numero a caso e chiedere all'utente un numero fino a quando non è uguale a quello generato casualmente.

67

## Numeri casuali



- rand() definita in stdlib.h
- per la generazione di numeri pseudo-casuali nell'intervallo compreso tra 0 e RAND\_MAX (costante definita in stdlib.h)

68



```
#include <stdio.h>
#include <stdlib.h>
main()
{ int guess;
  int magic;
  magic = rand();
  do {
    printf("Indovina il numero magico: ");
    scanf("%d", &guess);
    if (guess==magic)
      printf("Bravo!!\n");
    else printf("Sbagliato. Riprova.\n");
  } while (magic != guess);
}
```

genera un  
numero casuale

69



```
#include <stdio.h>
#include <stdlib.h>
main()
{ int guess;
  int magic;
  magic = rand() % 10;
  do {
    printf("Indovina il numero magico: ");
    scanf("%d", &guess);
    if (guess==magic)
      printf("Bravo!!\n");
    else printf("Sbagliato. Riprova.\n");
  } while (magic != guess);
}
```

...così potrai  
indivinare più  
facilmente!

70