Dipartimento di Informatica e Sistemistica
Antonio Ruberti

“Sapienza” Università di Roma

Istruzioni iterative (o cicliche)

Corso di Fondamenti di Informatica

Laurea in Ingegneria Informatica

(Canale di Ingegneria delle Reti e dei Sistemi Informatici - Polo di Rieti)

Anno Accademico 2007/2008

Prof. Paolo Romano

Si ringrazia il Prof. Alberto Finzi per aver reso
disponibile il materiale didattico sul quale si basano queste slides

Istruzioni iterative

Esempio: leggi 5 interi, calcola la somma e stampala

Prima possibilità: 5 variabili, 5 letture, 5 somme etc.

```
int i1, i2, i3, i4, i5;
scanf("%d",&i1);
…
scanf("%d",&i5);
printf("%d", i1 + i2 + i3 + i4 + i5);
```

non è accettabile!

Seconda possibilità: 2 variabili, 5 letture, 5 somme etc.

```
int somma, i;
somma = 0;          /* inizializzato */
scanf("%d",&i);
somma = somma + i;
…                   /* per 5 volte */
scanf("%d",&i);
somma = somma + i;
printf("%d", somma);
```

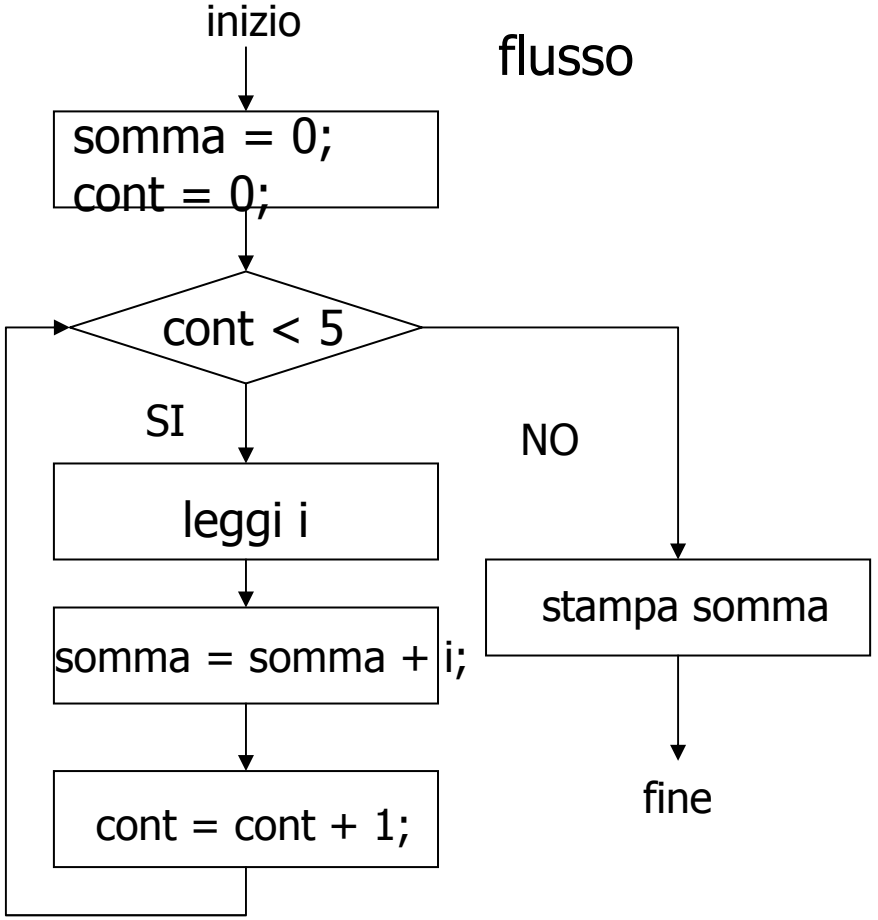
Occorre usare un'iterazione!

Istruzioni iterative

I costrutti iterativi sono necessari per definire cicli di istruzioni ripetute

Algoritmo per esempio precedente:

1. Assegna somma = 0;
2. Ripeti 5 volte le istruzioni:
   (a) leggi un intero i;
   (b) somma = somma + i;
3. Stampa il valore di somma



La 2. è un'istruzione iterativa

Nota due elementi:

1. Istruzione (semplice o complessa) da iterare
2. Condizione di uscita dal ciclo

Elementi fondamentali di un Ciclo

1. Test che verifica una condizione: condizione di entrata/uscita dal ciclo (condizione del ciclo)
2. Istruzioni da iterare: gruppo di istruzioni che vengono eseguite ad ogni iterazione (corpo del ciclo).
3. Tra le istruzioni da iterare occorre una **istruzione di modifica** che possa modificare l'esito del test: per permettere di uscire dal ciclo.
4. Inizializzazione delle variabili nella condizione di test: la prima volta che si esegue il test la condizione deve avere un valore sensato.

Due tipi di iterazioni

Iterazione definite: ripete l'istruzione per un numero definito di volte

Esempio:

```
Per 10 volte
fai una giravolta
```

istruzione `for`

Iterazione indefinite: ripete l'istruzione finchè una condizione è vera

Esempio:

```
Finchè non cadi
fai una giravolta
```

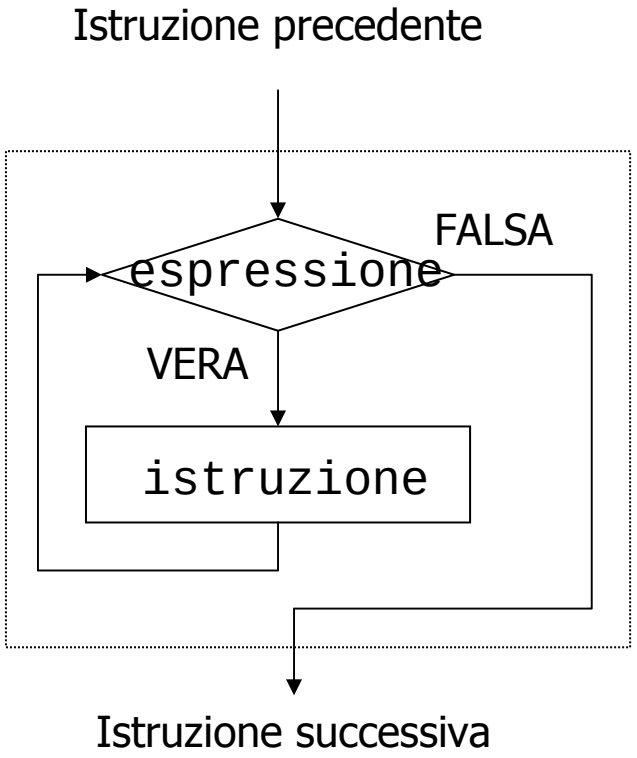
istruzione `while`

Istruzione while

Sintassi:

```
while(espressione)
    istruzione
```

- `espressione` è detta condizione del ciclo
- `istruzione` è detta corpo del ciclo



Semantica

a. viene valutata `espressione`

b. se `espressione` è vera
   viene eseguita `istruzione` quindi si torna al punto a.;
   se `espressione` è falsa si passa all'istruzione successiva

Nota: se l'espressione è subito falsa si passa subito all'istruzione successiva

Esempio istruzione while

Esempio iterazione definita:
stampa 100 asterischi

Algoritmo

1. inizializza il contatore a 0
2. Finche' il contatore è minore di 100 stampa "*" ed incrementa di uno il contatore

```
#include <stdio.h>

int main()

{ int c;
  c = 0;

  while (c < 100) {
    printf("*\n");
    c = c+1;
  }
}
```

Esempio istruzione while

Algoritmo per esempio iniziale:

Algoritmo

1. Assegna somma = 0;
2. Ripeti 5 volte le istruzioni:
   (a) leggi un intero i;
   (b) somma = somma + i;
3. Stampa il valore di somma

```
#include<stdio.h>

int main() {
   int somma, i, cont;
   somma = 0;  /* inizializzate variabili */
   cont = 0;

   while(cont < 5) {
      scanf("%d",&i);
      somma = somma + i;
      cont = cont + 1;
   }

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

`cont` è una variabile contatore:
iterazione controllata da un contatore

Esempio istruzione while

Leggere n interi positivi e stampane il massimo:

```
#include<stdio.h>
int main() {
   int n, i, max;

   printf("digita numero di interi?");
   scanf("%d",&n);
   max = 0;  /* iniz. var. */

   while(n > 0) {
      scanf("%d",&i);
      if (max < i) max = i;
      n = n – 1;
   }
   printf("il massimo è %d", max);
}
```

Algoritmo

1. Leggi il numero di interi n;
2. Assegna massimo = 0;
3. Ripeti n volte le istruzioni:
   (a) leggi un intero i;
   (b) Se (i > massimo) allora
       massimo = 1;
4. Stampa il valore di massimo

Esercizio: massimo di n interi qualunque (positivi e negativi)

Esempio istruzione while

Iterazione Indefinita: il numero di cicli non e' stabilito a priori

Esempio: stampa i valori della funzione –x*x + 10*x + 10 fino al primo negativo.

```
#include <stdio.h>
int main() {
   int x;
   x = 0;

   while (-x*x +10 *x + 10 >= 0){
      printf("%d \n", -x*x +10 *x + 10);
      x = x + 1;
   }
}
```

Esempio istruzione while

Esempio: dato un numero decimale stamparne le cifre binarie

Algoritmo

1. Leggi il numero n
2. Inizializza il quoziente ad n
3. Finche' quoziente non è 0
   c_i = resto di quoziente/2
   quoziente = quoziente/2

```
int n, /* numero in ingresso */
    q, /* quoziente */
    r; /* resto */

printf("digita il numero da trasformare: \n");

scanf("%d",&n);
q = n; /* inizializzazione variabile quoziente */

c = 0;
while(q != 0) {
   r = q%2; /* resto nella divisione per 2 */
   printf("%d \n",r);
   q = q/2;
}
```

Operatori di incremento e decremento

Operazioni come:

```
i = i + 1;
i = i – 1;
```

sono molto usate … forme abbreviate

Operatore di incremento: `++`
Operatore di decremento: `--`

In realtà con ++ (`--`) si hanno due operatori:
postincremento (predecremento)
- il valore dell'espressione è pari ad `i`
- il side-effect della valutazione: `i = i + 1`

preincremento (postdecremento): `++`
- il valore dell'espressione è pari ad `i + 1`
- il side-effect della valutazione è `i = i + 1`

Side-effect: modifica del contenuto di una locazione di memoria

Operazioni di assegnamento

Torniamo all'assegnazione ...

`a = b` è un'espressione!

- valore? valore di b (b è un'espressione)
- side effect? assegna valore di b alla variabile a

Operatore "=" associativo a destra

Esempio:

1. Associatività: `x = ( y = 4)`
2. Valore `(y = 4)` è `4` con side effect su y
3. Valore `x = (y = 4)` è pari a quello di `x = 4`: `4` con side effect su x

Nota: l'espressione `i = i + 1` (valore `i + 1` con side effect su `i`) equivale a `++i` ma non a `i++` !
Le istruzioni `i = i + 1; i++; ++i;` sono equivalenti!

Ancora incremento e decremento

Esempio:

	istruzione	x	y	z
1	`int x,y,z;`	?	?	?
2	`x = 4;`	4	?	?
3	`y = 2;`	4	2	?
4a	`z = (x + 1) + y;`	4	2	7
4b	`z = (x++) + y;`	4	2	6
4c	`z = (++x) + y;`	5	2	7

Attenzione! Non usare così!!

Per `z = (x++) + y;` scrivere:

```
z = x + y;
x++;
```

Per `z = (++x) + y;` scrivere:

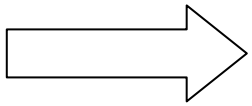
```
x++;
z = x + y;
```

Ricapitolazione Precedenze

!, - (segno), ++, --	**Massima precedenza**
*, /, %	
+, - (sottrazione)	
>, >= , < , <=	
== , !=	
&& (AND)	
|| (OR)	
= (assegnazione)	**Minima precedenza**

Forme abbreviate dell'assegnamento

```
a = a + b
a = a – b
a = a * b
a = a / b
a = a % b
```



```
a += b
a -= b
a *= b
a /= b
a %= b
```

Inizializzazione di variabili

L'assegnamento di un valore iniziale ad una variabile puo' essere fatto contestualmente alla sua dichiarazione.

```
int a, b = 10, c;
float pi = 3.14152, d;
```

`b, pi` sono inizializzate, non lo sono `a,c,d`

Istruzione `for`

Elementi comuni ai cicli controllati da contatore:
- variabile di controllo (contatore)
- inizializzazione della variabile di controllo
- incremento (decremento) della variabile di controllo
- verifica del valore finale

```
int i;
i = 0;
while (i <= 10) {
   printf("%d",i);
   i++;
}
```

l'istruzione for permette di gestire questo direttamente:

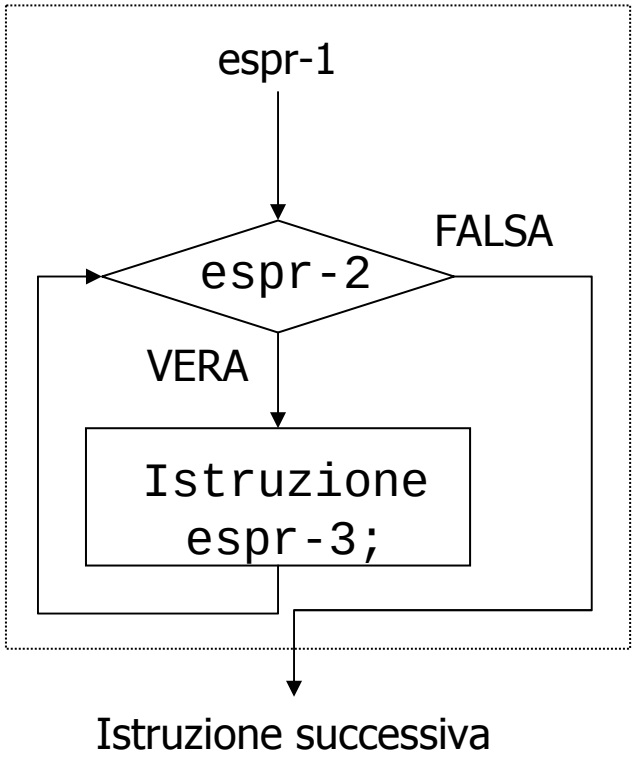
```
int i;
for (i=0;i <= 10;i++)
   printf("%d",i);
```

Istruzione `for`

Sintassi

```
for (espr-1;espr-2;espr-3)
   istruzione
```

- `espr-1`: inizializza la var di controllo
- `espr-2`: condizione di terminazione
- `espr-3`: modifica della variabile di controllo
- `istruzione`: corpo del ciclo



Semantica

```
espr-1;
while (espr-2) {
   istruzione
   espr-3;
}
```

Esempi istruzione `for`

Esempio: stampa i valori di x*x + 10*x -2 per x intero tra 10 e 20.

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

   for(x=10; x <=20; x++)
      printf("%d\n", x*x + 10*x  -2 );
   return 0;

}
```

Esercizio: stampa i valori di x*x + 10*x -2 per x pari da 20 a 10.

Esempio istruzione `for`

Algoritmo

1. Assegna somma = 0;
2. Ripeti 5 volte le istruzioni:
   (a) leggi un intero i;
   (b) somma = somma + i;
3. Stampa il valore di somma

```
#include<stdio.h>

int main() {
   int somma, i, cont;
   somma = 0;  /* inizializzate variabili */

   for(cont = 0; cont < 5; cont++) {
      scanf("%d",&i);
      somma += i;
   }

   printf("%d", somma);
   return 0;
}
```

Esempi istruzione `for`

```
for (i=1 ; i<=10 ; i++)
for (i=10; i>=1 ; i--)
for (i=-4; i<=4 ; i+=2)
for (i=0 ; i>=-10; i-=3)
```

```
i=1,2,3,…,10
i=10,9,8,…,1
i=-4,-2,0,2,4
i=0,-3,-6,-9
```

La sintassi `for` ammette `espr-i` qualsiasi:

e' comunque opportuno:
- utilizzare le `espr-i` per definire i contatori;
- non modificare i contatori nel ciclo.

Ciascuna delle `espr-i` può mancare (in questo caso usare `while`):
- si devono mantenere le ";"
- se manca `espr-2` questa viene assunta vera

es. `for (i=1 ; ; i++) printf("%d",i);`

Esempi istruzione `for`

Somma di tutti gli interi pari da 2 a 100:

```
int sum = 0, number;

for (number = 2 ; number <=100 ; number += 2)
   sum += number;
```

L'intestazione del `for` può contenere anche parte del corpo `for`:

```
for (number = 2 ; number <= 100 ; sum += number, number +=2);
```

L'inizializzazione (`sum = 0`) può essere unita all'inizializzazione del `for`:

```
for (sum = 0, number = 2 ; number <= 100 ; sum += number,
number +=2);
```

Esempi istruzione `for`

L'espressione:

```
sum += number, number +=2
```

Utilizza l'operatore virgola "`,`"

```
espressione1 , espressione2 , espressione3
```

- Espressione valutata da sinistra a destra;
- Valore dell'espressione è quello dell'ultima espressione

```
y = 0;
x = (5+7, y++, 4);
printf("valore espressione %d %d \n", x,y);
```

Valore espressione?

Istruzione `do-while`

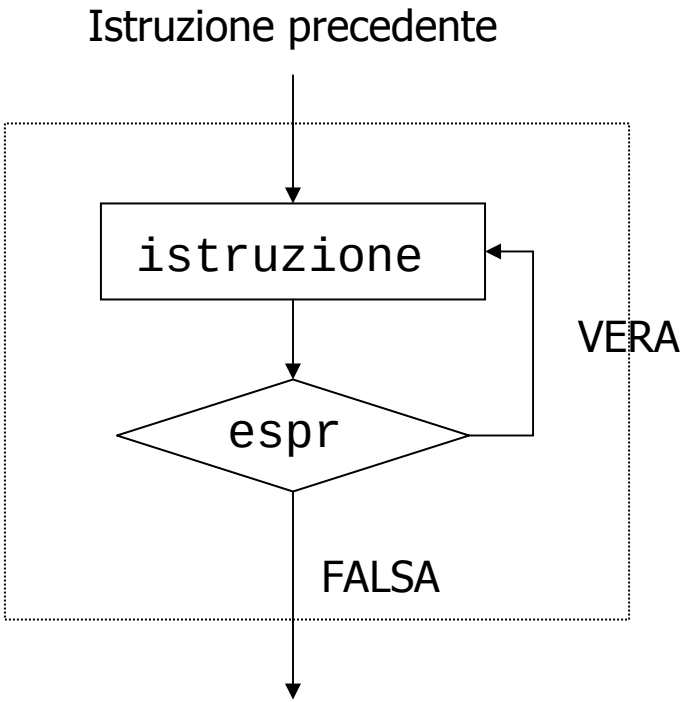
Istruzione simile a `while` con controllo alla fine di ogni iterazione

Sintassi

```
do
   istruzione
while (espr);
```

Semantica

```
istruzione
while (espr) {
   istruzione
}
```



Esempio istruzione `do-while`

```
main() {
   int counter = 1;
   do {
      printf(“%d”, counter);
   } while(++counter <= 0);
   return 0;
}
```

Esempio istruzione `do-while`

```
#include<stdio.h>
int main() {
   int n, i, max, c=0;

   printf("digita numero positivo di interi\n");
   scanf("%d",&n);


   do{  printf("digita intero");
        scanf("%d",&i);
        if (c==0) max = i;
        if (max < i) max = i;
      } while(n > ++c);

   printf("il massimo è %d", max);
   return 0;
}
```

1. Leggi primo intero e poni a max
2. Finchè n >c
   a. leggi intero;
   b. se intero è maggiore di max aggiorna max

Cicli Annidati

Il corpo di un ciclo può contenere un altro ciclo

Algoritmo:

per ogni riga da 1 a 10
    {per ogni colonna da 1 a 10
        stampa riga * colonna
     stampa a capo
    }

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

Istruzione `break`

L'istruzione break permette di uscire dai cicli (`while, for, do-while`) e da `switch`.

Esempio

```
int i,n;

for(i=1; i <10 ; i++) {
   scanf("%d",n);
   if (n > 0) printf("%d",n)
   else break;
}
```

`break` altera il flusso del programma: si perde la strutturazione, si guadagna in efficienza