Istruzioni Iterative

Concetti chiave

- Strutture di controllo iterative
  - `while`
  - `do-while`
  - `for`
- Cicli annidati

Altre Espressioni

- **Operatori Aritmetici** +, -, *, /, si applicano a qualunque tipo
- **Operatore Aritmetico modulo** %, restituisce il resto della divisione intera. **Si applica solo ad interi e produce un intero**
  - Esempio: se x vale 5, 34 % x vale 4
- **Operatore di Assegnazione**:
  - x=x+3
  - x=y=z=5

Esempio:

```
#include<stdio.h>
float z=3.4, x;
int main(void){
        printf("\nEspressione = %f  ", x=z);
        printf("\nx = %f  ", x);
}
```

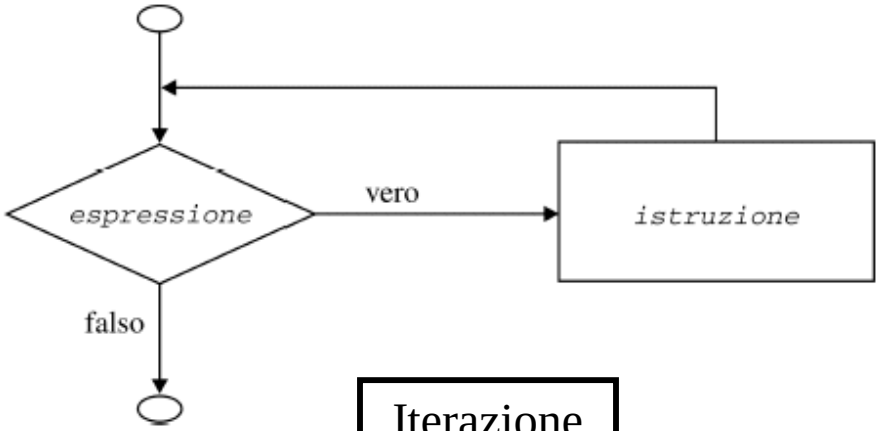
Altre Espressioni

- E' possibile combinare l'assegnazione con gli operatori aritmetici:
  - +=, -=, /=, *=, %=
  - x+=3 (x=x+3), x-=y (x=x-y), x*=2 (x=x*2)
  - x/=z (x=x/z), x%=y (x=x%y, **solo se x, y interi**)
  - x+=1 (x=x+1), x-=1 (x=x-1)

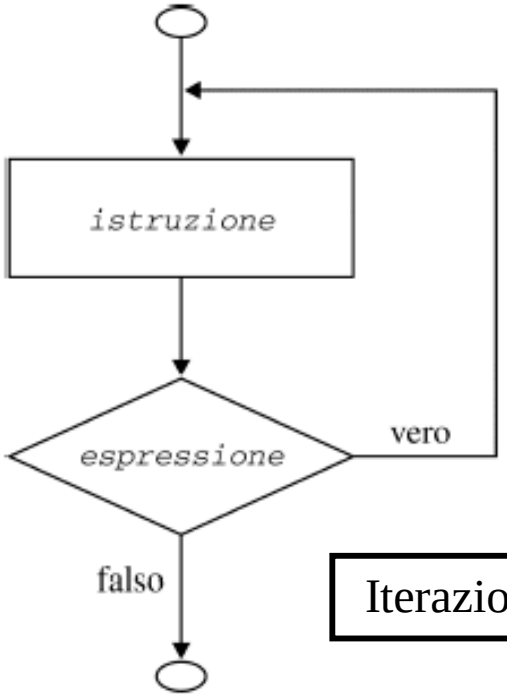
- Per gli incrementi e decrementi unitari esistono ++ e --
  - Forma prefissa ++x, --x
  - Forma postfissa x++, x --
  - Differenza: x=y++, x=++y. Esempio:

```
int x=4, y;
y=x++;      y=4 e x=5
y=++x;      y=5 e x=5
```

Istruzioni Iterative



Iterazione



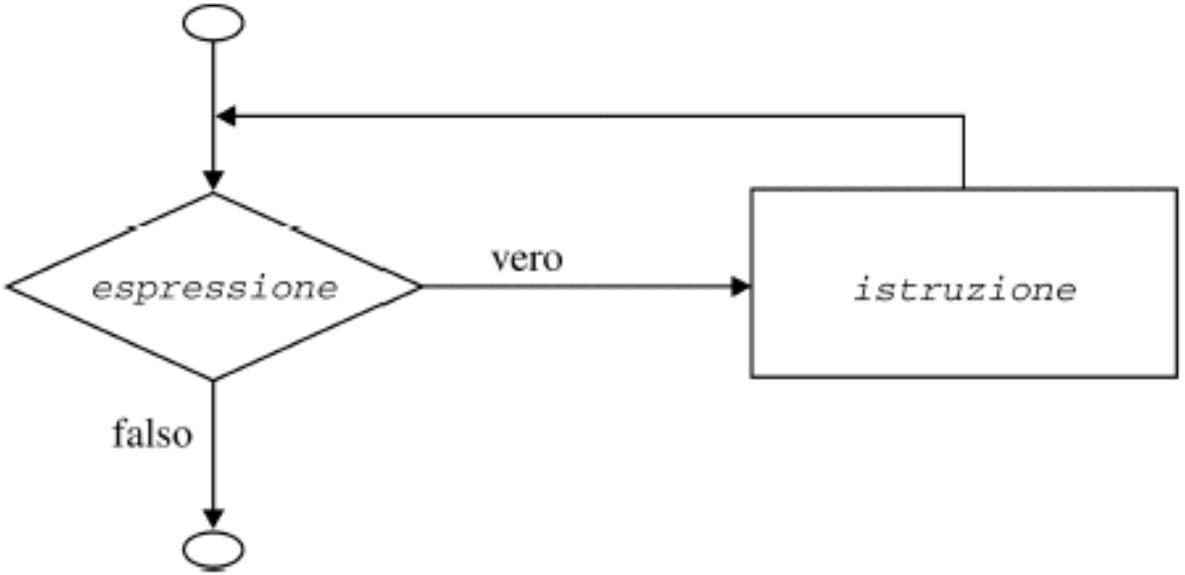
Iterazione

Istruzioni Iterative while

Sintassi:

`while(`*`espressione`*`)` *`istruzione`*

viene verificato che *espressione* sia vera, nel qual caso è eseguita *istruzione*. Il ciclo si ripete fintantoché *espressione* risulta essere vera.



Istruzioni Iterative while

```
somma=i=0;
while(i<5) {
        printf("Inserisci un numero intero: ");
        scanf("%d", &numero);
        somma += numero;
        i++;
}
```

Se non ci fosse: loop infinito !

- Si poteva inserire l'incremento `i++` all'interno della condizione logica tra parentesi tonde: **`while(i++<5)`**
- È obbligatorio posporre l'operatore alla variabile se si desidera che l'incremento avvenga dopo il confronto tra `i` e 5. Provare **while (++i<5)**

Istruzioni Iterative while

Uso di break

```
i = 0;
while(1) {
 if (i==5) break;
 printf("\nInserisci intero ");
 scanf("%d", &numero);
 somma += numero;
 i++;
}
```

```
i = 0;
while(i++<5) {
 printf("\nInserisci intero ");
 scanf("%d", &numero);
 somma += numero;
}
```

Istruzioni Iterative while

Calcolo della potenza

```
#include<stdio.h>

unsigned long esponente;
double base, potenza;

int main(void)
{
        printf("\nInserisci la base ");
        scanf("%lf",&base);
        printf("\nInserisci l'esponente ");
        scanf("%lu",&esponente);
        potenza=1.0;

        while (esponente-->0)
                potenza*=base;
        printf("\nPotenza = %e",potenza);
}
```

Istruzioni Iterative while

Calcolo Fattoriale

Fattoriale $n!$, di un intero n :

$n! = n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdot \ldots \cdot 2 \cdot 1$

1! e 0! sono per definizione uguali a 1.

$4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$
$6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$

Servono due variabili: n e fattoriale
fattoriale= $n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdot \ldots \cdot 2 \cdot 1$

- Problema: Scelta del tipo di dati
- unsigned long long n;
- unsigned long long fattoriale; ?????? Giusto ???????

Istruzioni Iterative while

- Max unsigned long long = 18.446.744.073.709.551.615~18E+18
- n=20, Fattoriale~2,43E+18

n	fattoriale	Tipo di dato necessario
2	2	unsigned long
…….		
11	39.916.800	
12	479.001.600	
13	6.227.020.800	double
……		
170	7.3E+306	
171	1.24E+309	long double
……		
1754	6.42E+4930	

unsigned long long

Istruzioni Iterative while

Calcolo Fattoriale

```
#include<stdio.h>

long double fattoriale;
unsigned long n;

int main(void)
{
    printf("\nInserisci un numero intero > 1 ");
    scanf("%lu",&n);

    fattoriale=1.0;

    while (n>1) fattoriale*=n--;

    printf("\nIl fattoriale e' %Le ",fattoriale);
}
```

Istruzioni Iterative while

Calcolo Numeri Primi

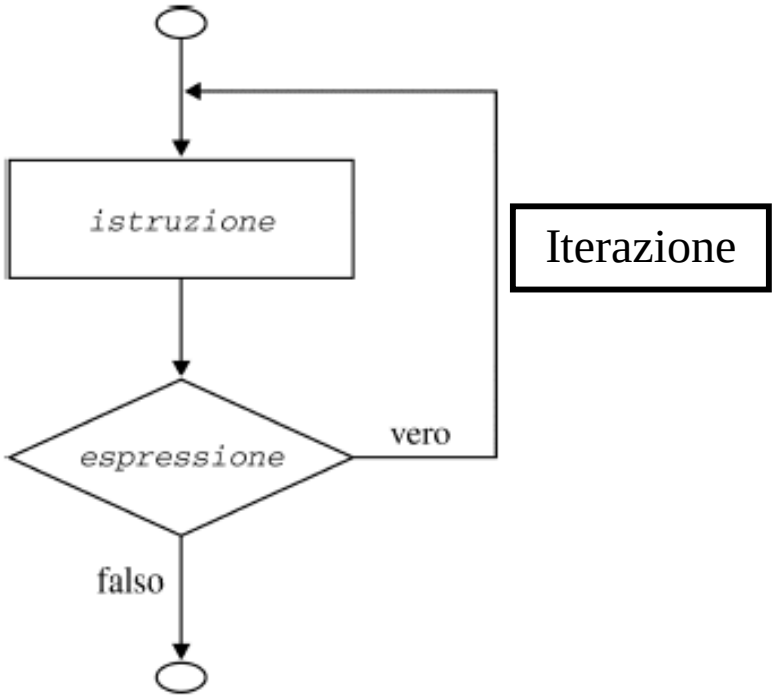
```
#include<stdio.h>

unsigned long n,d, count;

int main(void)
{
   printf("\nInserisci un numero intero positivo > 2 ");
   scanf("%lu",&n);
   d=2;
   while (n>1) {
         while (n%d) d++;
         count=0;
         while (n%d==0) {   /*oppure (!(n%d))*/
             n/=d;
             count++;
         }
         printf("\nUn numero primo e' %lu con potenza %lu ",d,count);
   }
}
```

Istruzioni Iterative do-while

Sintassi:

```
do
    istruzione
while(espressione)
```

Viene eseguita *istruzione* e successivamente controllato se *espressione* è vera, nel qual caso il ciclo è ripetuto.



Istruzioni Iterative do-while

```
x=3;
do
   printf("\nCiao");
while (x<5);
```

Loop infinito

```
x=3;
do {
   printf("\nCiao");
   x++;
} while (x<5);
```

```
x=3;
do
   printf("\nCiao");
while (++x<5);
```

```
x=3;
do {
   printf("\nCiao");
   if (++x>=5) break;
} while (1);
```

3 codici equivalenti

Istruzioni Iterative do-while

```
do {
   printf("\nInserisci un numero compreso tra 4 e 8 estremi inclusi");
   scanf("%d",&x);
} while (x<4 || x>8);
```

controllo correttezza dati in ingresso

controllo uscita da programma

```
do {

   istruzioni;

   printf("\nvuoi continuare (s/n) ? ");
   scanf("%c",&x);
} while (x!='n' && x!='N');
```

Istruzioni Iterative do-while

controllo uscita da programma con menu

```
do {
        printf("\n\t\t\tMenu'\n");
        printf("\n1) Operazione 1");
        printf("\n2) Operazione 2");
        printf("\n3) Operazione 3");
        printf("\n4) Fine Programma ");
        printf("\nInserisci la tua scelta: ");
        scanf("%u",&scelta);

        switch(scelta)  {
                case 1:
                    istruzione1;
                    break;
                case 2:
                   istruzione2;
                   break;
                case 3:
                   istruzione3;
                   break;
         }
} while (scelta<4);
```

Istruzioni Iterative do-while

Determina la media

```
#include<stdio.h>

double media, n;
unsigned int count;
char s;

int main(void)
{
   count = 0;
   do {
     printf("\nInserisci un numero  ");
     scanf("%lf", &n);
     media += n;
     count++;
     printf("\nContinui ? (s/n) ");
     scanf("%*c%c", &s);
   } while (s != 'n' && s != 'N');
   printf("\nLa media e' %f ", media / count);

}
```

Istruzioni Iterative
cicli annidati

- All'interno del blocco istruzioni di un ciclo while o do-while è possibile inserire anche istruzioni cicliche
- In questo modo si realizzano i cicli annidati
- Ad esempio servono per ripetere una determinata istruzione $N \cdot M$ volte:

```
i=0;
do {
  j=0;
  do
    istruzione;
  while (++j<M);
} while (++i<N);
```

```
i=0;
do {
  j=0;
  while (j++<M) istruzione;
} while (++i<N);
```

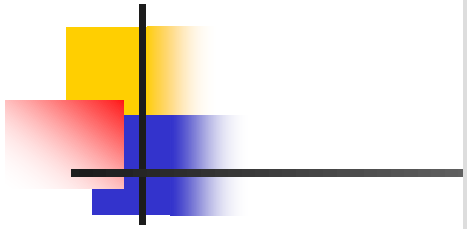
```
i=0;
while (i++<N) {
  j=0;
  while (j++<M) istruzione;
};
```

Istruzioni Iterative
cicli annidati

Stampa di una Matrice

```
#include<stdio.h>

#define N 10
#define M 5
unsigned int i,j;

int main(void){
    i=0;
    while (i++<N) {
        j=0;
        printf("\n");
        while (j++<M) printf("R%u,C%u\t",i,j);
  }
}
```

Istruzioni Iterative
cicli annidati



Calcolo Fattoriale

```
#include<stdio.h>

double fattoriale;
unsigned long n;
char s;

int main(void)
{
   do {
      fattoriale=1.0;
      do {
          printf("Inserisci un numero intero > 1 ");
          scanf("%lu",&n);
       } while (n<2);
       while (n>1) fattoriale*=n--;
       printf("Il fattoriale e' %le \n",fattoriale);
       printf("\nContinui ? (s/n) ");
       scanf("%*c%c",&s);
 } while (s!='n' && s!='N');
}
```

Istruzioni Iterative
for

La sintassi dell'istruzione `for` è

`for(esp1; esp2; esp3) istruzione`

esp1, esp2 e esp3 sono opzionali

L'ordine di esecuzione è:

1. E' eseguita *`esp1`*,
2. Poi si valuta *`esp2`*,
3. se *esp2* è vera, viene eseguita *`istruzione`* (altrimenti il processo ha termine).
4. E' eseguita *`esp3`*,
5. Si torna al punto 2. Il processo si ripete finché *`esp2`* non risulta essere falsa.

Istruzioni Iterative
for

Esempio:

```
somma = 0;
for(i=0; i<3; i++) {
   scanf("%d", &numero);
   somma += numero;
}
```

si ottiene la somma di tre numeri interi immessi dall'utente.

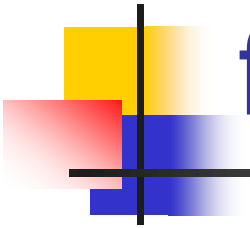
Istruzioni Iterative
for

- Il ciclo for è equivalente al ciclo while

for(*esp1*; *esp2*; *esp3*) *istruzione*

```
esp1;
while (esp2) {
    istruzione;
    esp3;
}
```

Istruzioni Iterative
for



Errori tipici:

- for (n=1; n<1;n++) printf("\nciao");
- for (x=0; x<5; x++); printf("\nciao");
- for (x=0; ;x++) printf("\nciao");

Istruzioni Iterative
for

Calcolo del fattoriale

```
#include<stdio.h>

double fattoriale;
unsigned long n;

int main(void){
    printf("\nInserisci un numero intero > 1 ");
    scanf("%lu",&n);

    for (fattoriale=1.0; n>1; n--) fattoriale*=n;

    printf("\nIl fattoriale e' %e ",fattoriale);

}
```

Istruzioni Iterative
for – cicli annidati

Esempio di Cicli Annidati

Stampa matrice Ai,j

```
#include<stdio.h>
#define RIGHE 15
#define COLONNE 2

unsigned int i,j;

int main(void)
{
  for (i=0; i<RIGHE ;i++) {
       printf("\n");
       for (j=0; j<COLONNE ;j++)
                    printf("A%u,%u\t",i,j);

  }
}
```

Istruzioni Iterative
for e l'operatore virgola

- L'operatore virgola permette di inserire più istruzioni all'interno delle espressioni.
- Un `for` può includere le inizializzazioni all'interno di *esp1*

```
for(numero=1, somma=0; numero!=0;)
```

- e istruzioni all'interno di *esp3*

```
for(i=1, j=5; i<10 && j<100; i++, j*=i)
```

- Nell'esempio, `i` viene inizializzato a 1 e `j` a 5. Il ciclo si ripete finché `i` è minore di 10 e contemporaneamente `j` è minore di 100. A ogni ciclo `i` viene incrementato di 1 e a `j` viene assegnato il prodotto di `i` per `j`.
- L'operatore virgola ha priorità più bassa di tutti gli altri.