Cicli ed iterazioni in C

Struttura di un ciclo

1. **Inizializzazione**. Assegnazione del valore iniziale a tutte le variabili che vengono lette durante il ciclo (nella condizione o nel corpo).
2. **Condizione di ripetizione**. Condizione, di solito inizialmente vera, che al termine del ciclo diventerà falsa. Deve dipendere da variabili che saranno modificate all'interno del ciclo (nel corpo o nell'aggiornamento).
3. **Corpo del ciclo**. Le istruzioni che effettivamente occorre ripetere: sono lo scopo per cui il ciclo viene realizzato. Si possono usare e modificare le variabili inizializzate.
4. **Aggiornamento**. Modifica di una o più variabili in grado di aggiornare il valore della condizione di ripetizione (rendendola, prima o poi, falsa). Tengono "traccia" del progresso dell'iterazione.

Operatori di auto-incremento/decremento

Auto-incremento	`i++ ;` `++i ;`	equivale a `i = i + 1 ;`
Auto-decremento	`i-- ;` `--i ;`	equivale a `i = i - 1 ;`

Costrutti iterativi

Costrutto **while**	`while(condizione)`
	`{`
	`  corpo ;`
	`}`

Costrutto **do-while**	`do`
	`{`
	`  corpo ;`
	`}`
	`while(condizione) ;`

Costrutto **for**	`for( inizializzazione; condizione; incremento )`
	`{`
	`  corpo ;`
	`}`

Equivalenza `for-while`

```
for ( inizializz; condiz; aggiornamento )
{
    corpo ;
}
```

```
inizializz ;
while ( condiz )
{
    corpo ;
    aggiornamento ;
}
```

Ciclo infinito

```
for( ; ; )
{
    corpo ;
}
```

```
while( 1 )
{
    corpo ;
}
```

Numero di iterazioni noto a priori

Da 0 a $N-1$, crescente	`for( i=0 ; i<N ; i++) { corpo ; }`	`i = 0 ; while( i<N ) { corpo ; i++ ; }`
Da 1 a N, crescente	`for( i=1 ; i<=N ; i++) { corpo ; }`	`i = 1 ; while( i<=N ) { corpo ; i++ ; }`
Da $N-1$ a 0, decrescente	`for( i=N-1 ; i>=0 ; i--) { corpo ; }`	`i = N-1 ; while( i>=0 ) { corpo ; i-- ; }`
Da N a 1, decrescente	`for( i=N ; i>0 ; i--) { corpo ; }`	`i = N ; while( i>0 ) { corpo ; i-- ; }`

Numero di iterazioni non noto a priori

Finché l'utente non inserisce un dato speciale

```
scanf("%d", &dato) ;

while( dato != DATOSPECIALE )
{
    elabora_dato ;
    scanf("%d", &dato) ;
}
```

```
do
{
    scanf("%d", &dato) ;
    if( dato != DATOSPECIALE )
    {
        elabora_dato ;
    }
}
while( dato != DATOSPECIALE ) ;
```

Finché non si verifica una condizione particolare

```
fine = 0 ; /* inizializzazione "flag" */
while( fine == 0 )
{
    elabora1 ;

    if( condizione_particolare )
       fine = 1 ;

    elabora2 ;
}
```

Contatori

Conta le iterazioni

```
conta = 0 ;

while( condizione )
{
    istruzioni ;

    conta ++ ;
}
```

Conta quante volte si verifica una condizione particolare

```
conta = 0 ;

while( condizione )
{
    istruzioni ;

    if (condizione_particolare)
        conta ++ ;

    altre_istruzioni ;
}
```

Accumulatori

Somma valori

```
somma = 0 ;

for( i=0 ; i<N; i++ )
{
    istruzioni ; /* calcola "valore" */

    somma = somma + valore ;
}
```

Massimo

```
max = INT_MIN ;
/* inizializzato ad un valore minore dei
   numeri di cui si vuole calcolare
   il massimo */

for( i=0 ; i<N; i++ )
{
    istruzioni ; /* calcola "numero" */

    if( numero > max )
        max = numero ;
}
```

Minimo

```
min = INT_MAX ;
/* inizializzato ad un valore maggiore dei
   numeri di cui si vuole calcolare
   il massimo */

for( i=0 ; i<N; i++ )
{
    istruzioni ; /* calcola "numero" */

    if( numero < min )
        min = numero ;
}
```

Flag

```
trovato = 0 ; /* flag per la ricerca */
/* inizializzo a "NO" = falso */

for( i=0 ; i<N; i++ )
{
    istruzioni ;

    if(condizione_particolare)
        trovato = 1 ;

    altre_istruzioni ;
}

/* al termine del ciclo, verifico */
if( trovato == 1 )
{
    printf("SI") ;
}
else
{
    printf("NO") ;
}
```

Esistenza e Universalità

	Esistenza	**Universalità**

P è vero	Esiste almeno un caso in cui P sia vero
	`esiste = 0 ;`
	`while(condizione)`
	`{`
	`  if( P è vero )`
	`  esiste = 1 ;`
	`}`
	`if ( esiste==1 ) ...`
		In tutti i casi, P è vero
		`sempre = 1 ;`
		`while(condizione)`
		`{`
		`  if( P non è vero )`
		`  sempre = 0 ;`
		`}`
		`if ( sempre==1 ) ...`

P è falso	Esiste almeno un caso in cui P sia falso
	`esiste = 0 ;`
	`while(condizione)`
	`{`
	`  if( P non è vero )`
	`  esiste = 1 ;`
	`}`
	`if ( esiste==1 ) ...`
		In tutti i casi, P è falso
		`sempre = 1 ;`
		`while(condizione)`
		`{`
		`  if( P è vero )`
		`  sempre = 0 ;`
		`}`
		`if ( sempre==1 ) ...`

Cicli Annidati

```
for( i=0; i<10; i++ )
{
    for( j=0; j<10; j++ )
    {
        printf("i=%d - j=%d\n", i, j);
    }
}
```

```
i=0 - j=0
i=0 - j=1
i=0 - j=2
...
i=0 - j=8
i=0 - j=9
i=1 - j=0
i=1 - j=1
i=1 - j=2
...
i=1 - j=8
i=1 - j=9
i=2 - j=0
i=2 - j=1
i=2 - j=2
...
i=2 - j=8
i=2 - j=9
...
...
i=9 - j=0
i=9 - j=1
i=9 - j=2
...
i=9 - j=8
i=9 - j=9
```

Istruzioni `break` e `continue`

```
while ( C )
{
  B1 ;
  if ( U ) /* condizione uscita */
     break ;
  B2 ;
}
/* se U e' vera, salta
immediatamente qui,
ed interrompe il ciclo
anche se C e' ancora vera.
In tal caso, B2 non
viene eseguita. */
```

```
while ( C )
{
  B1 ;
  if ( U )
     continue ;
  B2 ;
  /* se U e' vera, salta
  immediatamente qui,
  poi riprende la prossima
  iterazione. In tal caso,
  B2 non viene eseguita. */
}
```