Il programmatore che c'e' in noi – Lezione 15– Gira che ti rigira …

A) Il mese scorso avevo ordinato un oggettino dagli stati uniti; sin qui nulla di speciale, dopo circa una settimana mi comunicavano via mail che il materiale era stato spedito ed era arrivato alla dogana Italiana (… non facciamo commenti …).
Da quel giorno, appena rientravo a casa dal lavoro, eseguivo le medesime operazioni:

- controllavo se nella cassetta della posta c'era un pacco o una lettera che mi avvisava dell'arrivo
- chiedevo in casa se era passato un corriere ed aveva lasciato un pacco per me
- controllavo su internet se per caso la dogana o le poste mi avessero inviato una mail per avvisarmi dell'imminente arrivo del pacco

Bene, tutte queste operazioni, io le eseguivo esattamente nello stesso modo, nella stessa sequenza, ogni giorno.

Ovviamente avrei aspettato in eterno, se non mi fossi ad un certo punto (diciamo una ventina di giorni dopo), appeso al telefono e sollecitato la consegna del pacco.

B) Ho preso un bel raffreddore e sono quindi andato dal Dottore, risultato: prendere un sciroppo per la tosse per 5 giorni consecutivi.

Con questi due esempi, introduco quello che in informatica e' chiamata ITERAZIONE o CICLI o LOOP.

Un ciclo e' quindi il RIPETERSI di una o piu' sequenze di operazioni (istruzioni) per un certo numero di volte o, addirittura, per sempre (ciclo infinito o loop infinito) a seconda che si verifichi o no una determinata condizione.

Le istruzioni in linguaggio C / C++ e ObjectiveC per eseguire dei cicli sono

for che consente di impostare preventivamente il numero di volte (numero di cicli) nelle quali eseguire un gruppo di istruzioni (il caso B esposto in precedenza e' un tipico esempi di un for , prendere lo sciroppo per 5 volte).

while che consente di effettuare un loop (=un ciclo) sulle varie istruzioni sino a quando non si verifica una determinata condizione che ne provoca il termine (si dice 'uscire dal loop' o 'uscire dal ciclo').
L'esempio A del pacco, e' un esempio di un ciclo while, in quanto le operazioni descritte erano eseguite sino a quando non e' arrivato il pacco, che ne ha quindi provocato il termine.

Nota: Esisteva anche la possibilita' di 'uscire dal loop' rinunciando a ritirare il pacco, in questo caso sarei uscito dal ciclo non perche' la condizione era stata soddisfatta (il pacco e' arrivato ?) ma perche' mi ero stufato di aspettare, tale evenienza esiste anche in ambito informatico, quindi si puo' uscire da un ciclo (sia un while sia for) utilizzando un'altro statement che e' break.

In alcuni casi e' anche possibile ritornare a rifare il loop, senza aver eseguito tutte le istruzioni previsto all'interno del loop stesso, utilizzando l'istruzione continue.

Per chiarire il concetto, riprendo l'esempio A, modificando le operazioni che eseguivo ad ogni ciclo (ogni giorno)

1. controllo la cassetta della posta

2. chiedo in casa se e' passato un corriere
3. se ho il collegamento internet
   controllo se ho messaggi inerenti il mio pacco
   altrimenti
   riesegue il ciclo, inutile eseguire il punto 4

4. se ho il messaggio rispondo e preparo la documentazione per il ritiro

SINTASSI

Istruzione FOR

```
for( inizializza_lista_varabili ; condizione_test ; incremento_valore_variabili){
  prima istruzione;
  seconda istruzione;
  …
}
```

Se le istruzioni da eseguire si riducono ad una sola si puo' non mettere l'inizio e fine blocco {} quindi

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

Con il termine inizializza_lista_varabili si intende la possibilita' di dichiarare ed inizializzare una o piu' variabili da utilizzare nel nostro for.

Ad esempio

```
for( int nIndex=0 ; …  //Dichiaro ed inizializzo una sola variabile di tipo intero

for(int nIndex=0, int nAltezza=67; … //Dichiaro ed inizializzo una lista di variabili intere
                                     //( sono separate dalla virgola)
```

Per ragioni di chiarezza di scrittura di codice, io preferisco sempre dichiarare le variabili prima del loro utilizzo, quindi

```
int nIndex=0; //Indice dell'array
…
…
for(nIndex=0; … // utilizzo la variabile nIndex nel for
```

Il termine condizione_test consente di specificare la condizione per poter continuare ad eseguire il ciclo, sino a quando il risultato della condizione e' VERO il ciclo sara' eseguito.
Esempi

```
for(nIndex=0; nIndex < 5; … // significa che il ciclo sara' eseguito sino a quando nIndex avra' un
                            //valore inferiore a 5.
```

La condizione di test puo' essere una qualsiasi espressione valida, ad esempio

for(nIndex=0,nAltezza=67; (nIndex < = nMaxIndex && nAltezza < 200); …

significa che il ciclo sara' eseguito sino a quando nIndex e' inferiore o uguale a nMaxIndex e nAltezza e' inferiore a 200.

L'utima parte della sintassi del for incremento_valore_variabili consente di specificare l'eventuale incremento delle variabili, ad ogni giro del ciclo. (posso usare qualsiasi operatore di incremento/decremento, quali ++, --, +=, -= ecc.)

Esempi

```
for(nIndex=0; nIndex < 5; nIndex ++){
 NSLog("Prendere lo sciroppo\n");
}
```

Scrivera' 5 volte di prendere lo sciroppo.

Al primo giro del ciclo, nIndex vale 0
nIndex e' minore di 5 ? SI (VERO)
scrivi "prendere lo sciroppo"
nIndex++ (incrementa di 1, quindi da 0 passa a 1)

al secondo giro nIndex vale 1
nIndex e' minore di 5 = SI (VERO)
scrivi "prendere lo sciroppo"
nIndex++ (incrementa di 1, quindi da 1 passa a 2)

ecc.
al sesto giro nIndex vale 5
nIndex e' minore di 5 ? NO (FALSO)
fine del ciclo for

NOTA: All'uscita del ciclo nIndex vale 5.

```
for(nIndex=0,nAltezza=67; (nIndex < = nMaxIndex && nAltezza < 200); nIndex++, nAltezza+=50){
 NSLog("Eseguo istruzioni\n");
}
```

Quanto vale nIndex al termine del for? E nAltezza ?

VARIAZIONI SUL TEMA

Nella sintassi appena descritta nessuna parte del for e' obbligatoria, infatti posso anche scrivere un istruzione del genere

```
for(;;){
```

```
  istruzioni…;
}
```

Si tratta in questo caso di un loop infinito, in quanto non ho specificato una condizione di uscita dal for.

Per uscire da un ciclo loop infinito, posso utilizzare l'istruzione break che mi consente di interrompere immediatamente l'esecuzione del ciclo, e mi fa proseguire con la prima istruzione presente dopo la chiusura del for stesso.

Esempio

```
for(;;){
  ch=GetInputSceltaUtente(); //Restituisce il carattere premuto dall'utente

  if(ch == 'F')
   break;

}

NSLog("L'utente ha premuto F\n");
```

IMPORTANTE
Se si scrive for(; ;) ; //Si esegue un ciclo infinito da cui non e' piu' possibile uscire, quindi e' necessario forzare la chiusura dell'applicazione.

Posso anche indicare solo alcune parti del for (solo la condizione, la condizione e l'incremento, l'inizializzazione e la condizione), ma sono solo varianti della sintassi completa e spesso e' meglio (per motivi di leggibilita' del codice) utilizzare la forma completa.

Vediamo invece l'utilizzo all'interno di un ciclo dell'istruzione continue e break.

```
int nIndex=0;
char strTesto[101];
int nParole=0;
strcpy(strTesto,"Questo e' un testo di prova ");

for(nIndex=0,nParole=0; nIndex < strlen(strTesto); nIndex ++){

  if(strTesto[nIndex] != ' ') //Controllo se il carattere a cui sono posizionato e' uno spazio
     continue; //Se non lo e', allora continuo con il loop e rivaluto la condizione

 nParole++;

  if(nParole > 3) //se ho gia' almeno quattro parole, esco dal loop, anche se non ho finito
   break; //di analizzare tutto il testo
}

NSLog("Il numero di parole presenti nel testo e' %d",nParole);
```

SINTASSI

Istruzione WHILE

```
while(condizione){
  istruzioni eseguite se condizione = VERO;
  …
}
```

Esempio

```
while ( nIndex < 5){
  NSLog(“Eseguo istruzioni\n”);
}
```

equivalente a

```
for( ; nIndex < 5; ){

}
```

```
char ch=’Z’;

while( ch != ‘F’){

  ch = GetSceltaUtente();

}
```

simile a

```
for(;;){
 ch=GetSceltaUtente();
 if(ch==’F’)
  break;
}
```

```
while(1){ //Loop infinito …
  if(valore == 100)
    break;
}
```

simile a

```
for(;;){
 if(valore == 100)
```

```
    break;

}
```

Come visibile dagli esempi, e sempre possibile ricondurre un ciclo for ad un ciclo while e viceversa, ovviamente, a seconda delle circostanze e' piu' indicata una certa forma rispetto ad un'altra, sia per efficienza della scrittura del codice sia per migliorarne la leggibilita'.

Termino con una variante al ciclo WHILE che e' l'istruzione DO WHILE

SINTASSI

```
do{
  istruzioni da eseguire almeno una volta;
  …
}while(condizione);
```

La differenza sostanziale risiede nel fatto che la condizione viene valutata dopo aver eseguito le istruzioni presenti nel blocco {}; quindi tali istruzioni saranno eseguite almeno una volta.

Esempio

USO DEL WHILE

```
//Se nIndex vale 10, l'istruzione NSLog("nIndex = %d",nIndex); non sara' MAI eseguita.
while( nIndex != 10){
  NSLog("nIndex = %d",nIndex);  //Se nIndex e' diverso da 10, ottengo questa scritta,
  if(nIndex == 0)
   break;
}
```

USO DEL DO WHILE

```
//Se nIndex vale 10, l'istruzione NSLog("nIndex = %d",nIndex); sara' comunque eseguita 1 volta.
do{
NSLog("nIndex = %d",nIndex);  //Se nIndex e' diverso da 10, ottengo questa scritta,
  if(nIndex == 0)
   break;
}while(nIndex != 10);
```

Saluti.