Il programmatore che c'e' in noi – Lezione 12– Statements

Mi sveglio, vado in bagno, doccia, colazione.
Per colazione, preparo il caffe', accendo il gas, se e' acceso metto la caffettiera sopra il fornello ed aspetto. Aspetto sino a quando non sento la caffettiera brontolare, mi alzo, se il caffe' e' salito spengo il fornello del gas.
Bevo il caffe'.
Cerco le chiavi della macchina sino a quando non le trovo. Scendo, guido sino all'ufficio, incontro un semaforo, se e' verde, passo, se e' rosso mi fermo (…ehm… una volta mi sono distratto e sono passato con il rosso, mi e' costata una multa salata e 6 punti sulla patente…).
Arrivo nel piazzale dell'ufficio, ma e' tutto pieno, allora giro, giro, giro sino a quando non si libera un posto. Parcheggio. Lavoro, poi rientro a casa, se e' pronta la cena, mangio, altrimenti improvviso qualche piatto veloce. Un po' di TV (o di Mac ((vedi stasera…))) poi vado a dormire.

Un programma, come nella vita reale, non e' quasi mai sequenziale.

Ha un **inizio**, uno **svolgimento** ed un **termine** e ciascuna di queste parti si evolve in modo diverso in base a come si presentano determinate situazioni o eventi.

Gli statements, non sono altro che istruzioni che interessano lo svolgimento di alcune parti del programma, potremmo paragonarli a delle azioni che possono essere compiute e che influenzano lo svolgimento del programma stesso.

Gli statements possono essere raggruppati in

- decisionali - if, if else if, ?, switch
- iterazioni - for, while,do while
- salti - return,goto,break,continue
- etichette - nomeetichetta:
- espressioni - una espressione terminata con;
- blocchi - { }

Un programma ha sempre un inizio, e nel nostro caso (objectivec & cocoa) e' rappresentato da queste poche righe inserire in modo automatico da xcode, nel file main.m

```
#import <Cocoa/Cocoa.h>

int main(int argc, char *argv[])
{
    return NSApplicationMain(argc,  (const char **) argv);
}
```

Per fare il paragone di quanto scritto all'inzio,
"Mi sveglio" equivale a "utente manda in esecuzione il programma",quindi il programma si sveglia, ossia esegue il codice sopra specificato o per essere piu' precisi esegue la funzione
`int main(int argc, char *argv[])`.

Cosa significa, praticamente viene creato (istanziato) un primo oggetto di tipo (NSApplication). Tale oggetto rappresenta il punto di risveglio del programma, o meglio, il punto di INGRESSO nel programma, tale oggetto e di conseguenza il programma restera' attivo siano a quando l'utente non decidera' di CHIUDERE l'applicazione.

La chiusura dell'applicazione corrisponde alla parte finale della storiella iniziale (…poi vado a dormire).

Per verificare che, quando eseguo un programma, appena esso e' eseguito il codice e' proprio quello indicato sopra, facciamo un esperimento.

Usiamo l'istruzione `NSLog(@"Programma partito");` per far scrivere nel file di Log dell'applicazione un messaggio quando il programma e' eseguito.

Quindi, individuiamo il file main.m e modifichiamo il codice in questo modo

```
#import <Cocoa/Cocoa.h>

int main(int argc, char *argv[])
{
    NSLog(@"Programma partito");

    return NSApplicationMain(argc,  (const char **) argv);

}
```

Compiliamo il programma (Build e Run) e guardiamo cosa succede:
Compare la classica finestra vuota e nel file di run log
[Session started at 2006-07-20 22:34:47 +0200.]
2006-07-20 22:34:47.983 statements[367] Programma partito

Bene, e quando termina ? Bella domanda.

L'istruzione `return NSApplicationMain(argc,  (const char **) argv);`
dobbiamo immaginarla composta da queste parti

```
//1: Variabile di tipo int
int valoreRestituito=0;

//2: esegue la funzione NSApplicationMain ed assegna il valore restituito dalla
funzione alla variabile valoreRestituito;
valoreRestituito=NSApplicationMain(argc,  (const char **) argv)

//3: esce dal blocco corrente (un blocco e' una parte di codice compresa tra
aperta e chiusa parentesi graffa) restituendo il valore della variabile
valoreRestituito
return valoreRestituito;
```

dato che il blocco di programma da cui si esce e' il main(){ } il programma termina la propria esecuzione restituendo al sistema operativo, il codice di ritorno (valoreRestituito).

Se, quindi, desiderassi visualizzare un messaggio "programma terminato" quando il programma e' prossimo al termine della sua esecuzione, usando `NSLog(@"Programma terminato");` probabilmente cercherei di scrive qualcosa del genere

```
#import <Cocoa/Cocoa.h>
```

```
int main(int argc, char *argv[])
{
    NSLog(@"Programma partito");

    return NSApplicationMain(argc,  (const char **) argv);

    NSLog(@"Programma terminato"); //NON FUNZIONA! Questa istruzione non sara'
                                   //mai eseguita, in quanto e' stata incontrata
                                   //una istruzione return che ha causato
                                   //l'uscita da questo blocco,
}
```

Provare per credere…

Bene, modifichiamo leggermente il codice in modo da poter intercettare il momento della terminazione dell'applicazione, per fare cio' e' necessario commentare la riga in cui viene effettuata la chiamata a NSApplicationMain() che crea ed esegue l'applicazione cocoa.
Nota: date un occhiata alla documentazione Apple relative al tale funzione.

```
#import <Cocoa/Cocoa.h>

int main(int argc, char *argv[])
{
    NSLog(@"Programma partito");

   // return NSApplicationMain(argc,  (const char **) argv);

    NSLog(@"Programma terminato");

    return 0;
}
```

Compiliamo ed eseguiamo, ora il risultato e' che NON compare nessuna finestra, il programma parte e termina istantaneamente; e' esattamente cio' che gli abbiamo chiesto di fare, ossia SVEGLIATI, scrivi PROGRAMMA PARTITO, scrivi PROGRAMMA TERMINATO,vai a dormire.
Il run log e' il seguente
[Session started at 2006-07-20 23:00:06 +0200.]
2006-07-20 23:00:06.903 statements[494] Programma partito
2006-07-20 23:00:06.903 statements[494] Programma terminato

statements has exited with status 0.

Notare la riga che dice che il programma statements e' uscito (e' terminato) con lo stato 0! Zero e' infatti il valore che noi abbiamo indicato nella nostra istruzione return. (se metto 23…)

```
#import <Cocoa/Cocoa.h>

int main(int argc, char *argv[])
{
    NSLog(@"Programma partito");
```

```
   // return NSApplicationMain(argc,  (const char **) argv);

    NSLog(@"Programma terminato");

    return 23;
}
```

[Session started at 2006-07-20 23:01:43 +0200.]
2006-07-20 23:01:43.453 statements[513] Programma partito
2006-07-20 23:01:43.454 statements[513] Programma terminato

statements has exited with status 23.

Il codice di uscita del programma puo' essere intercettato da altri programmi o dal sistema operativo per valutarne il valore ed effettuare o meno determinate attività, ma questo non ci interessa (...per il momento…)

Ritorniamo alla storiella iniziale; e proviamo a farne il programma…
Mi sveglio

```
int main(int argc, char *argv[]){
```

vado in bagno,

```
vaiInBagno();   //Eseguo la funzione
```

doccia,

```
faiDoccia();    //Eseguo la funzione
```

colazione

```
faiColazione();//Eseguo la funzione
```

cerco le chiavi della macchina, sino a quando non le trovo…

```
BOOL bHoLeChiaviDellaMacchina=FALSE;

while(bHoLeChiaviDellaMacchina!=TRUE){
  bHoLeChiaviDellaMacchina=cercaChiaviDellaMacchina();
}
```

incontro un semaforo, se e' verde, passo, se e' rosso mi fermo

```
BOOL bSemaforoVerde=FALSE;
if(bSemaforoVerde == FALSE){
 aspetto();
}else{
 parcheggio();
}
```

Un po' di TV (o di Mac ((vedi stasera…)))

```
NSLog(@"un po' di TV o MAC...");
```

poi vado a dormire

```
NSLog(@"Programma terminato (=poi vado a dormire)");
return 23;
```

```
}
```

Questo e' il run log del programma

```
[Session started at 2006-07-20 23:42:58 +0200.]
2006-07-20 23:42:58.071 statements[742] Programma partito
2006-07-20 23:42:58.072 statements[742] Sono in bagno!
2006-07-20 23:42:58.072 statements[742] Sono sotto la doccia!
2006-07-20 23:42:58.072 statements[742] Metto il caffe'
2006-07-20 23:42:58.072 statements[742] guido...
2006-07-20 23:42:58.072 statements[742] semaforo rosso...aspetto...5 secondi
2006-07-20 23:43:03.072 statements[742] semaforo Verde!
2006-07-20 23:43:03.072 statements[742] lavoro...
2006-07-20 23:43:03.072 statements[742] cena...
2006-07-20 23:43:03.072 statements[742] un po' di TV o MAC...
2006-07-20 23:43:03.072 statements[742] Programma terminato (=poi vado a dormire)

statements has exited with status 23.
```

Conclusione:
In questa puntata di introduzione agli statements ho cercato di spiegare con un esempio cosa sono e a cosa servono le varie istruzioni e parti del programma.
Nella prossima puntata, tratteremo i vari statements singolarmente, analizzandone la sintassi ed il modo di utilizzo in un programma.

Mi auguro di essere stato sufficientemente chiaro e vi saluto.

In allegato il progetto in Xcode 2.3 contenente l'esempio trattato.