Il programmatore che c'e' in noi – Lezione 5 – Introduzione ai Puntatori

Le variabili sono estremamente importanti nella programmazione, indipendentemente dal linguaggio di programmazione che si intende utilizzare.

La loro comprensione e' di conseguenza di fondamentale importanza, quindi oggi approfondiamo il discorso relativo alle variabili iniziato in precedenza.

Quando noi dichiariamo una variabile di un certo tipo di dato, non facciamo altro che "dire" (tramite l'istruzione di programma) al compilatore, "mi riservi una certa quantita' di memoria RAM del computer"

La quantita' di memoria che ci viene "riservata" dipende dal tipo di dato che la variabile stessa dovra' contenere e di conseguenza dal suo tipo dato specificato nella sua dichiarazione.

Tanto per capirci, se io dichiaro una variabile di tipo int il compilatore mi assegnera' 4 bytes di memoria Ram (questo lo possiamo verificare con l'utilizzo dell'operatore sizeof() come avevamo fatto in una delle prime lezioni).

Esempio

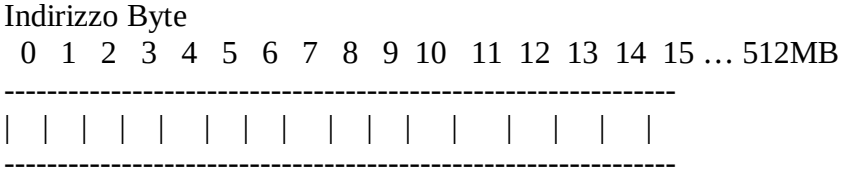
```
int nNumeroRivista=0; //Dichiaro la variabile di tipo int e la inizializzo a 0
int nBytesOccupati=0;

//Calcolo lo spazio di memoria occupato dalla variabile nNumeroRivista
nBytesOccupati = sizeof(nNumeroRivista);  // 4 bytes
```

Ma dove sono dislocati questi quattro bytes nella nostra memoria RAM ?
O per dirlo in termine informatico, quale e' l'indirizzo nella nostra memoria RAM della variabile ?

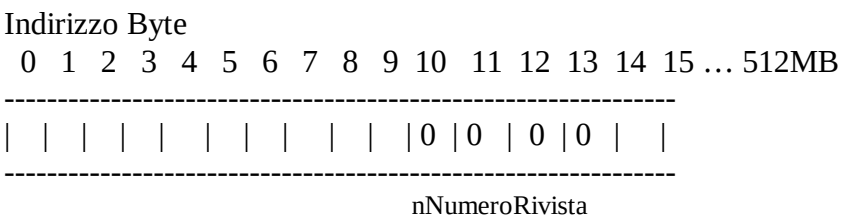
Il programma, quando sara' eseguito riservera' lo spazio di memoria richiesto, come indicato dalla nostra istruzione ma lo riservera' in una zona di memoria RAM libera e a sua completa discrezione, di conseguenza solo quando il programma sara' in esecuzione saremo in grado si conoscere la locazione di memoria (indirizzo) dove iniziano i BYTES che ci sono stati riservati.

Inoltre ad ogni riavvio del programma le locazioni dei nostri BYTES saranno quasi sicuramente cambiate. La cosa certa e' che i BYTES saranno CONSECUTIVI.

Prima di rispondere a tale domanda, realizziamo un semplice (spero) schema per riepilogare la situazione.
Supponiamo che la nostra memoria RAM si presenti tutta libera ed in questa forma…



Dopo l'esecuzione del programma, quando viene eseguita la nostra istruzione int nNumeroRivista=0 la situazione cambia e supponiamo sia diventata questa



come detto in precedenza, non siamo in grado di prevedere dove si troveranno le locazioni di memoria occupate dai nostri 4 BYTES

Quindi, in questo caso, la risposta alla domanda precedente e': la variabile nNumeroRivista e' stata posizionata (=allocata) alla locazione di memoria con indirizzo di partenza pari a 10.

Quindi i nostri 4 BYTES saranno rispettivamente agli indirizzi 10,11,12,13.

Bene, in C/C++ e ObjectiveC esiste un OPERATORE che ci consente di conoscere la locazione di partenza di una qualsiasi variabile, ossia di conoscerne il suo INDIRIZZO in MEMORIA RAM.

Tale operatore e' & (and)

Quindi per sapere realmente la locazione di memoria a cui e stata "posizionata" la nostra variabile devo scrivere una istruzione di questo tipo

indirizzoVariabile = &nomeVariabile e quindi nel nostro esempio

indirizzoVariabile = &nNumeroRivista; //Ricavo l'indirizzo (di partenza) della variabile

Spero che sino a qui, tutto sia sufficientemente chiaro… Se volete fare una pausa, e prendere un caffe', questo e' il momento… J

Le cose, si complicano…

1) Complicazione

Tutti gli indirizzi di memoria del computer sono indicati con un valore ESADECIMALE (HEX) Noi normalmente siamo abituati a usare i numeri con base DECIMALE, ossia, contiamo da 0 a 9 e quando arriviamo a 10, incrementiamo le decine e ripartiamo con le unita' 0 a 9 ecc. si dice che la base e' 10 (decimale appunto).

Quando invece si conta in esadecimale (indicato con Hex, h o 0X o 0x) la base e' 16 e quindi si conta da 0 a F (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) .

Le seguenti due istruzioni sono uguali
nNumeroRivista=223; //Assegno il valore DECIMALE
nNumeroRivista=0xDF; //Assegno il valore Hex

Per la conversione dei valori Dec/Hex e viceversa, vi consiglio di usare l'applicazione Calcolatrice del vostro Mac. Per chi vuole approfondire il discorso, consiglio di fare delle ricerche in Internet o se qualcuno ha tempo/voglia potrebbe approfondire meglio il discorso qui su Tevac.

2) Complicazione
Il valore che viene calcolato dall'operatore &, e' di un TIPO DI DATO particolare, molto particolare, e' un PUNTATORE.

Quindi sembra un valore int ma non lo e'.

Di conseguenza NON posso fare una operazione di questo tipo

int nNumeroRivista=0; //Dichiaro la variabile di tipo int e la inizializzo a 0

int indirizzoVariabile=0;

indirizzoVariabile=& nNumeroRivista; //Calcolo l'indirizzo della variabile

Questa riga viene compilata con un Warning del compilatore che recita pressappoco cosi' "warning: assignment makes integer from pointer without a cast"

Per memorizzare un INDIRIZZO devo avere una variabile di TIPO PUNTATORE ad TIPO DI DATO DELLA VARIABILE DELLA QUALE MI INTERESSA L'INDIRIZZO.

Come faccio a dichiarare la variabile puntatore che mi serve ?

1) penso ad un nome coerente con la variabile a cui si riferisce l'indirizzo
esempio, la variabile della quale voglio conoscere l'indirizzo si chiama nNumeroRivista, il nome della mia variabile di tipo puntatore che dovra' contenere l'indirizzo della variabile nNumeroRivista la chiamo pNumeroRivista (la p che ho usato come prefisso mi "ricorda" che e' un puntatore

2) di che tipo e' la variabile della quale voglio calcolare l'indirizzo ?
nNumeroRivista e' di tipo int, di conseguenza il mio puntatore e' un puntatore ad un tipo int

3) come indico nel programma che la variabile e' una variabile puntatore ?
Si usa un ulteriore OPERATORE del linguaggio di programmazione, che e' * (asterisco) da anteporre al nome della variabile (*nomevariabile) o da attaccare all'identificatore di tipo (int*) il risultato e' lo stesso. La scelta di un modo piuttosto che l'altro e' una questione stilistica e (reputo) personale.

NOTA
Quando io dichiaro una variabile di tipo int, ad esempio, la inizializzo al valore 0 (zero).
Quando la variabile e' di tipo puntatore a quale indirizzo la inizializzo quando la dichiaro ?
La risposta e' a NULLA, a NIENTE, a nessun indirizzo.
Per indicare questa condizione, in C e C++ si usa la parola chiave **null** mentre in ObjectiveC si usa **nil**.

Bene, ci siamo… riprendiamo il nostro esempio

int nNumeroRivista=0; //Dichiaro la variabile di tipo int e la inizializzo a 0
int* pNumeroRivista=nil; //Dichiaro la variabile di tipo puntatore ad un int e la inizializzo a nil

//Calcolo l'indirizzo della variabile

pNumeroRivista=& nNumeroRivista; //calcolo il valore dell'indirizzo e lo memorizzo nella variabile pNumeroRivista. Il valore di indirizzo potrebbe essere, ad esempio, 0xbffff614

Per vedere il valore di indirizzo del puntatore, se usiamo la NSLog() gia' incontrata in altre occasioni, dovremo scrivere in questo modo

```
NSLog(@"Indirizzo di nNumeroRivista: %p",pNumeroRivista);
```

Da notare l'uso del simbolo **%p** per convertire il dato da puntatore (indirizzo) a stringa (caratteri).

In allegato trovate un programma di esempio con quanto trattato in questa parte e qualcosetta in piu' che sara' trattato prossimamente.

Questo e' il Run log prodotto dal programma …

LeVariabiliApprofondimento[1085] Valore contenuto in nNumeroRivista: 223
LeVariabiliApprofondimento[1085] Dimensione Occupata da nNumeroRivista: 4
LeVariabiliApprofondimento[1085] Indirizzo di nNumeroRivista: 0xbffff628
LeVariabiliApprofondimento[1085] All' Indirizzo 0xbffff628 c'e' il valore 223
LeVariabiliApprofondimento[1085] Ora all' Indirizzo 0xbffff628 c'e' il valore 548
LeVariabiliApprofondimento[1085] ed ovviamnete anche nNumeroRivista vale 548

Per i piu' intraprendenti… nella finestra dell'applicazione e' contemplata anche la conversione del valore della variabile in valore binario, ma non e' ancora implemenata … chi vuole puo' farlo e postare la propria soluzione.

Saluti e buone feste.