Il programmatore che c'e' in noi – Lezione 7 – Arrays

Quante volte avete aperto e chiuso un cassetto di una cassettiera ? Molte, credo.
Bene, una cassettiera altro non e' che un array di cassetti.

Un array altro non e' che un INSIEME DI ELEMENTI DELLO STESSO TIPO INDIVIDUATI DA UNO STESSO NOME. Per accedere ad un determinato elemento dell'array usiamo un indice.

Nel esempio della nostra cassettiera, diro' apri il primo cassetto, il secondo ecc.

[1° cassetto]
[2° cassetto]

Gli arrays possono avere piu' di una dimensione, di conseguenza, per accedere ad un determinato elemento si usano indici per ciascuna dimensione.

Se la nostra cassettiera e' formata da tre colonne di cassetti, disposti su due righe, in tutto avremo 3 x 2 cassetti = 6 cassetti.

[1° cassetto] [2° cassetto] [3° cassetto] (prima riga)
[1° cassetto] [2° cassetto] [3° cassetto] (seconda riga)

Per accedere ad un determinato cassetto, dovro' indicare il numero del cassetto ed in numero della riga, esempio apri il primo cassetto della prima riga, apri il secondo cassetto della seconda riga.

Quindi possiamo avere arrays monodimensionali (una sola colonna o una sola riga), bidimensionali (una riga ed una colonna) e multidimensionali.

Nella maggior parte delle applicazioni, ci troviamo a lavorare con arrays monodimensionali e bidimensionali

Nel linguaggio di programmazione C/C++ e ObjectiveC l'indice del primo elemento e' ZERO e tutti gli elementi dell'arrays occupano LOCAZIONI DI MEMORIA CONSECUTIVE nella Ram del nostro Mac. Il primo elemento corrisponde all'indirizzo di memoria piu' basso, l'ultimo a quello piu' alto.

Per dichiarare una array di variabili si usa la seguente sintassi

Singola dimensione
TYPO_VARIABILE NOME_VARIABILE [NUMERO_ELEMENTI];

Due dimensioni
TYPO_VARIABILE NOME_VARIABILE [NUMERO_RIGHE] [NUMERO_COLONNE];

Operazioni sugli arrays

int nEtaPersone[50]; //Array di 50 elementi (accessibili usando indice tra 0 e 40 compresi)
// di tipo INT
char strCodice[12]; //Array di 12 char (caratteri) accessibili con indice 0-11
BOOL bBattagliaNavale[3][4];//Array bidimensionale di tre righe e quattro colonne di tipo BOOL

Per assegnare il valore ad uno specifico elemento

nEtaPersone[5]=37; //Assegna il valore 37 al 6° elemento dell'array … il primo elemento e' 0…

//Assegno dei caratteri all'array strCodice che e' di tipo char
strCodice=[0]='C';
strCodice=[1]='i';
strCodice=[2]='a';
strCodice=[3]='o';

bBattagliaNavale[0][0]=TRUE; //Assegno valore del primo elemento
bBattagliaNavale[2][3]=TRUE; //Assegno valore dell'ultimo

Per leggere il valore contenuto nell'array

int nEta=0;

nEta = nEtaPersone[5]; //nEta, vale 37 legge il valore del 6° elemento dell'array e lo assegna a nEta

MOLTO IMPORTANTE

1) Prestate molta attenzione al valore degli indici degli array, il compilatore non e' in grado di segnalare un uso errato del indice. Quindi se io ho un array di 50 caratteri, ed uso un indice che supera il valore di 49, il compilatore non mi segnala il problema, ma quando l'applicazione sara' in esecuzione, potrebbe provocare dei malfunzionamenti all'applicazione stessa e farla chiudere in maniera inaspettata.

2) La sola dichiarazione delle variabili di tipo arrays non comporta la loro inizializzazione, quindi il valore in esso contenuto e' sconosciuto. E' sempre opportuno INIZIALIZZARE i valori degli arrary, semplicemente assegnando un valore ad ogni singolo elemento dell'array.

L'inizializzazione puo' essere fatta nella dichiarazione stessa o mediante uso dell'operatore di assegnazione (=) .

Esempio di inizializzazione di una array nello stesso momento della sua dichiarazione

```
    //Dichiaro un array di caratteri di 12 elementi, accessibili con indice tra
    //0 e 11 e lo inizializzo con la scritta Tevac Corso

    char strCodice[12]={'T','e','v','a','c',' ','C','o','r','s','o','.'};
```

Come visibile dall'esempio, per inizializzare i valori, devo usare l'operatore di assegnazione (=) quindi aprire un blocco usando il simbolo { specificare i singoli valori per ogni indice del nostro array, separandoli da una virgola ed infine chiudere il blocco con }

Dato che si tratta di un array di char, i valori da usare per l'inizializzazione devono essere delle costanti di tipo char e si indicano con il valore di carattere racchiuso tra apici (non VIRGOLETTE, ma APICI, lo stesso simbolo dell'apostrofo)

'T' e' una costante di tipo char (=di tipo carattere)

int nCosti[3]={ 1200, 800, 899 }; //Inizializzo un array di dipo int

double nQta={ 123.45, 234.56, 89.00 } //Inizializzo un array di tipo double

Per inizializzare arrays bidimensionali, seguo la stessa regola, raggruppando ulteriormente ciascun gruppo di valori con un inizio e fine blocco ({ }) per ciascuna riga.

```
BOOL bBattagliaNavale[3][4]=
   {
     {0,0,0,0},  //inizializzo la prima riga
     {0,0,0,0},  //inizializzo la seconda riga
     {0,0,0,56}  //inizializzo la terza riga
    };
```

Esempio di inizializzazione di una array con l'operatore di assegnazione
E' necessario assegnare singolarmente i valori ai vari elementi che compongono l'array.

int nCosti[3]; //dichiaro l'array di interi ma non e' inizializzato

//inizializzo i valori dell'array
nCosti[0]=1200;
nCosti[1]=800;
nCosti[2]=899;

//Modico il valore di un elemento dell'array

```
char strCodice[12]={'T','e','v','a','c',' ','C','o','r','s','o','.'};

strCodice[5]='!'; //ora strCodice contiene Tevac!Corso ho sostituito il valore
//del sesto elemento dell'array (era uno spazio) con il !
```

Uso dei puntatori con gli array

Nelle lezioni precedendi abbiamo visto l'uso dei puntatori e della relativa aritmetica per accedere alle variabili. Dato che gli array di cui stiamo parlando sono collezioni di variabili dello stesso tipo, posso accedere ai dati di un array (quindi ai vari elementi) usando un puntatore invece che l'indice.

//Dichiaro una variabile puntatore ad interi e la inizializzo a NULL
int* pElemento=NULL;
int nCosti[3]={ 1200, 800, 899 }; //Dichiaro ed inizializzo un array di dipo int di TRE elementi.

//Calcolo l'indirizzo di memoria del primo elemento dell'array usando l'operatore &

pElemento = & nCosti[0]; //Calcola l'indirizzo della variabile relativa al primo elemento dell'array

(*pElemento)=45; //ASSEGNO IL VALORE 45 alla variabile puntata dal puntatore
                  //pElemento (in questo caso e' il primo elemento dell'array)

pElemento++; //Passo all'elemento successivo dell'array (indice 1)

(*pElemento)=78; //Ora pElemento PUNTA al secondo elemento dell'array (indice = 1)

E cosi via. Attenzione a non superare il numero di elementi precedentemente dichiarati nella'array.

Concludo con un esempio pratico dell'uso di un array.

Esempio:

Vogliamo memorizzare le nostre uscite finanziarie nei diversi mesi dell'anno.
Se non uso gli array, devo dichiarare 12 variabili per memorizzare il valore delle uscite per ciascun mese invece mi basta dichiarare un array di 12 elementi.

```
int nSpeseNelMese[12]; //Spese per ciascun mese
```

//Inizializzo l'array con le spese, mese per mese

```
nSpeseNelMese[0]=120;  //speso a Gennaio
nSpeseNelMese[1]=170;  //speso a Febbraio
nSpeseNelMese[2]=190;  //speso a Marzo
nSpeseNelMese[3]=10;   //speso a Aprile
nSpeseNelMese[4]=140;  //speso a Maggio
nSpeseNelMese[5]=180;  //speso a Giugno
nSpeseNelMese[6]=190;  //speso a Luglio
nSpeseNelMese[7]=320;  //speso a Agosto
nSpeseNelMese[8]=520;  //speso a Settembre
nSpeseNelMese[9]=220;  //speso a Ottobre
nSpeseNelMese[10]=1120;//speso a Novembre
nSpeseNelMese[11]=70;  //speso a Dicembre
```

Saluti.