

## ANSI C – Costrutti Condizionali

Eccoci finalmente a parlare di costrutti del linguaggio ANSI C, da questo momento in avanti cercherò di fornirvi tutti gli strumenti necessari per costruire un programma, strumenti utili per ripetere istruzioni, per effettuare scelte, per stampare su output, per ottenere dati in ingresso, etc...

Nel tutorial precedente abbiamo visto il funzionamento della logica booleana e abbiamo dato uno sguardo approfondito agli operatori di confronto, ora vediamo come possiamo utilizzare tutto questo a nostro favore nel codice, quindi andremo ad inserire un nuovo elemento, ovvero il costrutto condizionale.

Che cosa sono i costrutti condizionali? Il termine “condizionali” ci aiuta ad intuire di che cosa si tratta, spesso quando si scrive un applicativo, dobbiamo fare in modo che il software scelga una strada invece che un'altra in base a una condizione. Ricordate lo stupido esempio dei cesti di natale, il nostro ipotetico programmino doveva decidere in base al sesso dell'amico se scegliere un cesto rosa o un cesto azzurro. Ebbene questa decisione può essere presa con un costrutto condizionale, vediamo cosa dobbiamo fare in pseudo-codice (lo pseudo-codice non è altro che una descrizione a parole delle operazioni che il nostro codice deve fare, una via di mezzo tra la lingua parlata e il codice sorgente)

```
/*
 * frammento di pseudo-codice per decidere se assegnare
 * un cesto azzurro o rosa in base al sesso.
 */
se amico è uguale a maschio
allora
    assegna cesto azzurro
altrimenti
    assegna cesto rosa
```

Il frammento di pseudo-codice è molto semplice e chiaro, proviamo a vedere cosa conosciamo o ci sembra di conoscere. Abbiamo un controllo su amico, vogliamo vedere se è un maschio o una femmina, quindi possiamo vedere “amico” come una variabile. Supponiamo che sia una variabile, ora troviamo “è uguale a”, se prendete il tutorial precedente noterete che abbiamo tradotto “è uguale a” con ==, infatti come abbiamo detto nello scorso tutorial si tratta di un operatore di confronto, quindi tradotto diventa:

```
se amico == maschio
allora
    assegna cesto azzurro
altrimenti
    assegna cesto rosa
```

Mettiamo alcune parentesi in modo tale da poter vedere meglio quello che stiamo facendo. Se ricordate in uno dei primi tutorial abbiamo detto che usiamo le parentesi graffe per racchiudere blocchi di codice, in questo caso possiamo racchiudere tra parentesi graffe le due assegnazioni “assegna cesto azzurro” e “assegna cesto rosa”, perché effettivamente sono dei blocchi di codice a se stanti, quindi utilizzando le parentesi:

```
se (amico == maschio) allora
{
    assegna cesto azzurro
}
altrimenti
{
    assegna cesto rosa
}
```

Perfetto, in questo il momento il codice sembra molto più chiaro e leggibile rispetto a prima. Abbiamo davanti un frammento di pseudo-codice molto vicino al codice sorgente C, infatti proviamo a sostituire le parole in **grassetto** con i termini corrispettivi in inglese otteniamo

```
if (amico == maschio) then
{
    assegna cesto azzurro
}
else
{
    assegna cesto rosa
}
```

In realtà non siamo ancora giunti al costrutto ANSI C che ci interessa, perché come già ho ripetuto più volte i programmatori tendono a risparmiare tempo nello scrivere, quindi la parola “then” è omessa, il costrutto si riduce a:

```
if (amico == maschio)
{
    assegna cesto azzurro
}
else
{
    assegna cesto rosa
}
```

Questo è il costrutto C “if-else”, che ci permette di fare un controllo su una condizione messa tra parentesi e se il risultato del controllo (in informatiche si chiama guardia dell’if) è TRUE eseguo il contenuto del primo blocco, altrimenti eseguo il contenuto del secondo blocco:

```

if (espressione)
{
    blocco di istruzioni1
}
else
{
    blocco di istruzioni2
}

```

Ovviamente questa è la base, quindi vediamo i vari casi che possiamo trovarci davanti in modo tale da non rimanere spiazzati in caso vediate una scrittura diversa dell'IF.

Abbiamo parlato di blocco di istruzioni, in realtà è possibile che dopo il controllo della guardia dell'IF ci sia solo una istruzione e dopo else ci sia solo una istruzione (proprio come nell'esempio visto prima dove assegnavo solamente cesto rosa o azzurro), ebbene dato che i programmatori tendono a risparmiare energie nello scrivere il codice, quando c'è una sola istruzione possiamo omettere le parentesi graffe, quindi non vi spaventate se vi trovate davanti del codice scritto così:

```

if (espressione)
    istruzione1
else
    istruzione2

```

Ovviamente se in uno dei due blocchi c'è più di una istruzione le parentesi sono obbligatorie, quindi un piccolo consiglio, scrivete sempre le parentesi anche se avete una sola istruzione, oltre evitare possibili errori, avrete un codice più chiaro e leggibile.

Vediamo adesso un'altra possibile differenza, potrebbe succedere di trovare del codice dove il "ramo" else del costrutto non serve, se per esempio vogliamo eseguire un blocco se e solo se la valutazione della guardia è TRUE, altrimenti non vogliamo fare nulla di particolare ma continuare l'esecuzione del programma, allora potreste trovare questo:

```

if (espressione)
{
    blocco di istruzioni
}

```

Anche in questo caso vale il discorso delle parentesi detto prima, ora vediamo l'ultimo caso, ovvero se abbiamo più di due possibili scelte, se per esempio vogliamo vedere in base a range di età chi può votare alla camera e chi al senato, quindi avremo uno pseudo-codice di questo tipo:

```

se (anniVotante < 18)
{
    non può votare
}

```

```

altrimenti se ((anniVotante >= 18) && (anniVotante < 25))
{
    vota per la camera
}
altrimenti
{
    vota per la camera e il senato
}

```

Ottimo esempio, ci consente di vedere innanzitutto l'utilizzo degli operatori visti nel tutorial precedente, quindi guardiamo il ramo “altrimenti se” vediamo un'altra guardia, questa è composta, diciamo che a noi interessa andare avanti se l'espressione è TRUE, come detto prima, ma quando l'espressione è TRUE?

Se riprendete il tutorial precedente vedrete che l'operatore di AND (&&) restituisce TRUE solo quando entrambi gli operandi sono TRUE, quindi se anniVotante è maggiore o uguale di 18 e allo stesso tempo anniVotante è minore di 25, allora il risultato è TRUE (24 va bene, 26 no).

Trasformiamolo in lingua inglese:

```

if (anniVotante < 18)
{
    non può votare
}
else if ((anniVotante >= 18) && (anniVotante < 25))
{
    vota per la camera
}
else
{
    vota per la camera e il senato
}

```

Otteniamo così il seguente costrutto:

```

if (espressione1)
{
    blocco di istruzioni1
}
else if (espressione2)
{
    blocco di istruzioni2
}
else

```

```
{
    blocco di istruzioni3
}
```

Anche qui vale lo stesso discorso per le parentesi fatto prima e naturalmente anche in questo caso l'ultimo ramo, quello dell'else può essere omesso. Attraverso questo costrutto possiamo fare tutti i controlli che vogliamo aggiungendo “a cascata” tutti gli “else if” che ci servono, ad esempio:

```
if (anni == 1)
{
    ...
}
else if (anni == 2)
{
    ...
}
else if (anni == 3)
{
    ...
}
else if (anni == 4)
{
    ...
}
else if (anni == 5)
{
    ...
}
etc...
etc...
else
{
    ...
}
```

Come potete vedere scrivere tutti questi if non è proprio comodo, soprattutto nel caso sopra dove abbiamo una serie di controlli su un valore che cambia di 1.

Esiste quindi un altro costrutto condizionale molto utile proprio nel caso in cui abbiamo da testare una variabile e in base al suo valore vogliamo eseguire una particolare istruzione(o blocco di istruzioni). Il costrutto è lo SWITCH, vediamo subito un esempio perché sicuramente è il modo

più semplice per capirne il funzionamento:

```
switch (anni)
{
    case 1:
        istruzione1;
        break;
    case 2:
        istruzione2;
        break;
    case 3:
        istruzione3;
        break;
    case 4:
        istruzione4;
        break;
    default:
        istruzioneDefault;
        break;
}
```

Vediamo ora come funziona, ma è davvero molto semplice, per prima cosa il comando “switch” testa il valore della variabile, e in base al suo valore entra nel ramo “case” giusto, se per esempio anni ha il valore di 3, viene eseguita l'istruzione3 (ma possono essere anche un blocco di istruzioni classiche) e poi viene eseguito “break” che significa? Che il codice esce dallo switch saltando tutto il resto del codice e riprendendo l'esecuzione del programma normalmente dopo la graffa chiusa.

Ultimissima cosa, se il valore non è nessuno di quelli elencati viene eseguito il ramo “default”. Se non metto il break, le istruzioni vengono eseguite a cascata, ad esempio:

```
switch (anni)
{
    case 1:
        istruzione1;
        break;
    case 2:
    case 3:
    case 4:
        istruzione2;
        break;
    default:
        istruzioneDefault;
        break;
}
```

Se anni vale 2, 3, oppure 4 viene comunque eseguita l'istruzione2.

Benissimo siamo giunti alla fine di questo tutorial, spero di essere stato abbastanza chiaro, sicuramente più avanti vedremo l'utilizzo di questi costrutti in un programma completo e funzionale. Per ora studiate bene questa parte perché veramente importante per ogni programmatore. Il prossimo tutorial sarà dedicato alla ripetizione del codice un numero variabile di

volte, quindi parleremo di cicli. Alla prossima

Manuel (aka Jekrom)

per segnalazioni : [manuel.montini@gmail.com](mailto:manuel.montini@gmail.com)

<http://iphonecodes.wordpress.com/>

#### LICENZA:

**Questo tutorial è rilasciato sotto licenza Creative Commons : “Attribuzione-Non commerciale-Non opere derivate 2.5 Italia”,**

<http://creativecommons.org/licenses/by-nc-nd/2.5/it/>