

ANSI C – Gli Operatori

Bentornati, nel tutorial passato abbiamo visto che cosa sono le variabili e come si definiscono nel linguaggio ANSI C, abbiamo visto le differenze tra i vari tipi di dati e abbiamo introdotto il concetto di costante, che può essere definita all'interno del codice attraverso la parola chiave `const`, oppure indicandola al compilatore tramite la direttiva `#DEFINE`.

In questo tutorial andremo a vedere come queste variabili possono essere utilizzate all'interno del nostro codice, partendo dal modo più semplice di farne uso, ovvero attraverso gli operatori, ed in particolare andremo ad analizzare gli operatori aritmetici e gli operatori logici.

Operatori Aritmetici

Quando abbiamo visto le variabili le abbiamo introdotte utilizzando esempi di vita vissuta da tutti noi, facendo un passo indietro nel tempo prima alle scuole elementari e successivamente alle medie, anche per gli operatori aritmetici la strada è la stessa, perché si parla esattamente di quella simbologia che ci permette di effettuare delle operazioni sui valori che abbiamo definito.

Parlare di operazioni aritmetiche significa scomodare concetti come l'addizione, la sottrazione, la moltiplicazione, la divisione e naturalmente l'assegnazione. Concetti che ci sono stati insegnati fin dal più tenera infanzia (ricordate gli insiemi? Quando andavamo ad inserire o a rimuovere oggetti da questi insiemi?).

Tutte queste operazioni erano fatte su numeri, noi ovviamente possiamo pensare di trasportarle anche sulle variabili in un linguaggio di programmazione. Guardiamo questo esempio:

```
x = 3 + 5           // Riga 1
x = a - b           // Riga 2
x = a * b / c       // Riga 3
x = ( a + b ) / 2   // Riga 4
```

Queste non sono delle righe di codice sorgente di un programma, innanzitutto perché non sappiamo se tutte quelle “lettere” (ovvero x, a, b, c) sono state dichiarate come si dichiarano delle variabili, non sappiamo se sono state inizializzate, non sappiamo il tipo di queste variabili e le singole righe non terminano con il punto e virgola, ma se prendiamo le singole righe fuori dal contesto della programmazione, queste ci vanno più che bene, sono delle comunissime operazioni matematiche che sappiamo risolvere senza troppi problemi (naturalmente avendo dei valori validi da assegnare alle singole variabili).

Immaginiamo ora di prendere le singole righe e di trasportarle in codice sorgente ANSI C, un codice che, una volta compilato, anche il nostro computer possa eseguire senza troppi sforzi. Prendiamo la Riga 1 e trasformiamola in un frammento di codice valido:

```
int x = 3 + 5;
```

Un codice banale, ma che se inserito all'interno di un programma in C, è in grado di darci un risultato ben definito, ovvero assegnare alla variabile `x` il valore 8. Benissimo ma andiamo ad esaminare la riga di codice in ogni sua parte. Iniziamo ovviamente con la dichiarazione della variabile come visto nel tutorial precedente, quindi creiamo una variabile di nome `x` che è di tipo `int`, subito dopo troviamo l'operatore di assegnamento `=`, in effetti l'uguale è il primo degli operatori aritmetici che andiamo a prendere in considerazione, quando scriviamo il nome della variabile uguale a qualche cosa, andiamo a dire al nostro programma di assegnare il valore che sta a destra dell'uguale alla variabile che si trova alla sua sinistra. Continuando nell'analisi della riga vediamo comparire due numeri e l'operatore di addizione `+`.

Torniamo all'esempio precedente ed esaminiamo le altre righe, vediamo comparire l'operatore di sottrazione `-`, di moltiplicazione `*`, di divisione `/`, e nella Riga 4 vediamo anche le parentesi. Per ora lasciamo stare il discorso delle parentesi che tratteremo comunque in questo tutorial e concentriamoci sugli operatori appena detti.

Volendo riassumere:

- Operatore **assegnamento** `"="` serve per assegnare ad una variabile un valore;
- Operatore **addizione** `"+"` serve per sommare due variabili tra loro (fanno eccezione le variabili di tipo `char` di cui parleremo successivamente);
- Operatore **sottrazione** `"-"` serve per sottrarre dalla variabile sulla sinistra dell'operatore il valore che si trova sulla destra del simbolo (fanno eccezione le variabili di tipo `char` di cui parleremo successivamente);
- Operatore **moltiplicazione** `"*"` serve per moltiplicare tra loro le due variabili (fanno eccezione le variabili di tipo `char` di cui parleremo successivamente);
- Operatore **divisione** `"/"` serve per effettuare la divisione tra le due variabili (fanno eccezione le variabili di tipo `char` di cui parleremo successivamente);

La cosa sembra molto semplice, in realtà non è così, in quanto si parla di variabili che possono avere vari tipi, quindi prestiamo un'attenzione particolare all'operatore di divisione, distinguendo il comportamento nel caso in cui le due variabili da dividere siano intere e nel caso in cui siano valori in virgola mobile (o reali):

```
// Frammento 1
int risultato                = 0;
int primaVariabileIntera    = 9;
int secondaVariabileIntera  = 3;
risultato = primaVariabileIntera / secondaVariabileIntera;
```

```
// Frammento 2
double risultato            = 0.0;
double primaVariabileReale  = 6.0;
double secondaVariabileReale = 1.5;
risultato = primaVariabileReale / secondaVariabileReale;
```

```
// Frammento 3
int risultato          = 0;
int primaVariabileIntera  = 11;
int secondaVariabileIntera = 3;
risultato = primaVariabileIntera / secondaVariabileIntera;
```

Iniziamo a vedere che cosa accade nei tre frammenti di codice. Nel primo abbiamo definito tre variabili intere, una per il risultato e le altre due con due numeri interi, poi abbiamo eseguito l'operazione di divisione, ora ci chiediamo quanto vale risultato alla fine dell'operazione, la risposta ovviamente è molto semplice, il risultato della divisione intera è 3. Passiamo a vedere il frammento di codice numero due, anche in questo caso tre variabili, ma tutte e tre sono dei numeri reali, quindi il risultato della divisione sarà un numero reale, ovvero 4.0 e non 4.

Ora prendiamo in considerazione il terzo ed ultimo frammento, abbiamo una divisione tra due numeri interi e il risultato deve essere un numero intero, se fosse stata la classica operazione di divisione il risultato sarebbe stato 3.66 periodico, ma se voi andate ad eseguire il codice noterete che il risultato è semplicemente 3, perché si tratta appunto di una operazione di divisione intera e ci restituisce la parte intera del risultato della divisione, quindi come potete vedere, in base al tipo di variabile il risultato cambia.

Per completare la carrellata sugli operatori aritmetici aggiungiamo l'operatore per ottenere il resto della divisione:

```
int risultato          = 0;
int primaVariabileIntera  = 11;
int secondaVariabileIntera = 3;
risultato = primaVariabileIntera % secondaVariabileIntera;
```

Il risultato che si ottiene dal frammento di codice non è un valore percentuale, ma bensì il resto della divisione intera tra le due variabili, quindi il risultato finale è 2, ovvero per dirla come facevamo alle elementari il 3 nell'11 ci sta 3 volte con resto di 2.

Arrivati a questo punto ultimerei il discorso sugli operatori aritmetici parlando di quanto noi programmatori siamo avari nello scrivere codice, ci sono alcune forme contratte che possono tornare molto utili, sia quando si scrive il codice, che conoscerle per quando si deve leggere il codice scritto da altri. Se volete incrementare il valore di una variabile di 1, oppure diminuire il valore di una variabile di uno voi fareste nel seguente modo:

```
int contatore = 0;           // Inizializzo il contatore
contatore = contatore + 1;   // Incremento di 1
contatore = contatore - 1;   // Decremento di 1
```

Questo è il modo che avete imparato ad utilizzare seguendo il tutorial, ma non è l'unico. Un programmatore può fare la stessa cosa mettendo una coppia di “più” prima o dopo la variabile se si tratta di incrementare il valore o una coppia di “meno” prima o dopo la variabile se si tratta di diminuire di uno il valore della variabile, vediamo come fare:

```

int contatore = 10;    // Il contatore vale 10
contatore ++;        // Ora vale 11
++ contatore;        // Ora vale 12
contatore --;        // Ora vale nuovamente 11
-- contatore;        // Siamo tornati a 10

```

Il perché si mettano prima o dopo il nome della variabile non verrà spiegato in questo tutorial, vi basti sapere che questo modo di scrivere è molto comune, soprattutto se si tratta di contare il numero di volte che viene fatta una determinata operazione.

Ultimo concetto riguarda la composizione dell'operatore di assegnamento con uno qualsiasi degli altri operatori (+, -, *, /), vediamo un esempio e poi spieghiamo a cosa serve:

```

int valore1 = 10;    // Creo una variabile che vale 10
int valore2 = 15;    // Creo una variabile che vale 15
valore1 += valore2;  // valore1 ora vale 25

```

Che cosa abbiamo fatto? Abbiamo creato due variabili e le abbiamo inizializzate con due valori, la prima vale 10 e la seconda vale 15, poi volevamo sommarle tra loro e mettere il valore dentro valore1, prima di questa notazione avreste scritto:

```

valore1 = valore1 + valore2;

```

Ebbene con la composizione di “+” e “=” diciamo che il valore della variabile a sinistra della composizione degli operatori è uguale alla somma di se stessa più la variabile a destra, ovviamente lo stesso vale anche per gli altri operatori. Questa notazione è molto comoda e vedrete che con la pratica la utilizzerete spesso anche voi.

Un piccolo accenno alle parentesi, dato che gli operatori matematici in ANSI C seguono le stesse regole di quelli aritmetici imparati alle scuole, possiamo usare le parentesi per forzare il programma ad eseguirne prima una parte e poi un'altra, esattamente come nella matematica classica, quindi se vogliamo ad esempio fare la somma di “a + b” e poi moltiplicare tutto per un valore “c” abbiamo due possibilità di farlo, o lo facciamo in due passaggi distinti, oppure usiamo le parentesi per forzare prima l'operazione di somma e dopo quella di prodotto. Vediamo per finire un esempio di questo in modo da vedere un piccolo ripasso del resto.

```

int risultato = 0
int valore1 = 10;
int valore2 = 15;
int valore3 = 3;
risultato = (valore1 + valore2) * valore3; //(10+15)*3

```

Operatori Logici

Mi sono dilungato parecchio sulla parte relativa agli operatori aritmetici, anche se in teoria dovrebbe essere una parte abbastanza semplice ed intuitiva, ma per proseguire nei tutorial ho bisogno che tutte le basi siano ben chiare a tutti. Ora entreremo in contatto con gli operatori logici e soprattutto vedremo come questi possono essere sfruttati per prendere decisioni. Questa parte è un pochino più complessa di quella precedente, ma spero di riuscire a renderla il più chiara possibile.

Prima di cominciare a parlare degli operatori logici vorrei fare una breve introduzione sull'algebra booleana, non mi voglio dilungare troppo e sicuramente tutto quello che vi dirò non coprirà nemmeno un centesimo dell'argomento, ma spero di riuscire a dare comunque un quadro generale abbastanza chiaro. Abbiamo detto nei tutorial precedenti che i nostri calcolatori sono delle macchine stupide, che non capiscono le istruzioni così come gliele diciamo, ma che devono prima trasformarle in una serie di impulsi elettrici. Prendiamo quindi un chip che sta all'interno del nostro computer, tutti sappiamo come sono fatti esternamente, hanno un corpo e dei piedini che vengono inseriti all'interno della scheda madre, in base a come facciamo passare la corrente in alcuni piedini, abbiamo una uscita in altri.

Noi possiamo quindi alimentare o non alimentare un piedino, ovvero far passare o non far passare la corrente, ed è quello che succede quando si trasforma un programma, lo si traduce in un linguaggio macchina e si comincia a far passare o non far passare corrente.

Possiamo vedere quindi questa situazione come uno stato dell'architettura:

- 1 = VERO, passa corrente
- 0 = FALSE, non passa nulla

Vediamo quindi che possiamo avere due stati di verità, ovvero se passa la corrente diciamo VERO (in inglese TRUE) e lo associamo al numero 1, se invece la corrente non passa diciamo FALSO (in inglese FALSE) e lo associamo al numero 0. Adesso capite il perché dei numeri binari quando si parla di computer, se ricordate in un tutorial precedente abbiamo parlato della rappresentazione ASCII di un carattere e abbiamo fatto vedere il corrispettivo codice come una sequenza di 1 e 0.

Benissimo, una volta introdotti questi concetti binari e esserci assicurati che siano chiari, li andiamo ad incasinare ben bene aggiungendo alcune "operazioni" che possiamo fare su questi valori. Vi mostrerò solo i tre operatori di base, tralasciando gli altri, perché questi sono quelli che devono essere chiarissimi nella vostra testa, perché in qualsiasi programma li utilizzerete. Quindi vediamo di addentrarci dentro molto lentamente ma in maniera esaustiva, diciamo che per semplicità utilizzerò non 1 o 0, ma VERO o FALSO, o meglio, dato che il linguaggio di programmazione ha comandi in Inglese utilizzeremo TRUE o FALSE.

Partiamo con il primo operatore, ovvero il NOT logico, che cos'è il NOT? Questo è sicuramente il più semplice degli operatori, l'operatore NOT nega il valore che gli passiamo, o in parole povere se noi diciamo NOT TRUE il risultato dell'operazione logica sarà FALSE e se diciamo NOT FALSE, indovinate un pochino quale sarà il risultato? Ovviamente il contrario, o meglio il negato, o meglio ancora TRUE.

Benissimo nulla di particolarmente complesso, stiamo parlando di valori TRUE e FALSE e stiamo parlando di operazioni che possiamo fare su questi valori, così come $3 + 2 = 5$, qui abbiamo NOT TRUE = FALSE. Piccolo inciso la stessa cosa vale su usassi i numeri binari per le operazioni, ovvero NOT 1 = 0, NOT 0 = 1. Chiaro no?

Quando si parla di linguaggi di programmazione ovviamente entra in gioco la simbologia per scrivere tutto questo, nella stragrande maggioranza dei linguaggi di programmazione l'operatore di negazione è utilizzato spessissimo e per indicare il NOT si utilizza il simbolo "!". Quindi se volessi fare il negato di una variabile chiamata `boolValue`, non farei altro che scrivere

!boolValue.

Quindi per ricapitolare:

- !TRUE = FALSE
- !FALSE = TRUE

Come potete vedere non è molto complesso e questa notazione la utilizzerete spesso in qualsiasi linguaggio di programmazione, dopo parleremo degli operatori di confronto, ma se andiamo a fare una breve anticipazione, per dire che una variabile è diversa da un'altra andremo a dire che una variabile è il non uguale a un'altra, ovvero è il negato dell'uguaglianza, ovvero il NOT dell'uguaglianza, ovvero `variabile1 != variabile2`, NOT uguale, `!=`.

Torniamo agli operatori logici e andiamo a prendere in esame il secondo operatore, si tratta di AND, che cos'è AND? Questo operatore ci restituisce TRUE se e solo se entrambi gli operandi sono TRUE, in pratica basta che uno dei due operandi che entrano in gioco con l'operazione logica di AND è FALSE che il risultato sarà FALSE. Cerchiamo di creare una tabella di esempio:

Operando 1	Operando 2	Operazione Logica	Risultato Finale
FALSE	FALSE	FALSE AND FALSE	FALSE
FALSE	TRUE	FALSE AND TRUE	FALSE
TRUE	FALSE	TRUE AND FALSE	FALSE
TRUE	TRUE	TRUE AND TRUE	TRUE

Esaminiamo la tabella, ovviamente dato che abbiamo due operandi binari, il numero massimo di combinazioni di valori che possiamo ottenere è 4, esattamente come le righe della tabella. Nella colonna Operazione Logica, troviamo l'operazione di AND in base al valore dei due operandi e come dicevamo prima il risultato finale è TRUE se e solamente se entrambi gli operandi sono TRUE, basta che uno sia FALSE, o nel primo caso entrambi FALSE che il risultato sarà false.

Forse vi starete chiedendo, ok ma tutto questo a cosa mi serve? La risposta è molto semplice, potete utilizzare l'AND per controllare quando due condizioni si verificano contemporaneamente, supponiamo che vogliate scrivere un programma per l'assegnazione dei regali di natale ai vostri amici, avete preparato dei cesti con delle leccornie da mangiare, ovviamente volete distinguere tra amici maschi e amiche femmine, lo fate prendendo cesti rosa per le ragazze e azzurri per i ragazzi. Poi sapete che tra questi amici alcuni amano i canditi e altri no, quindi dovete mettere panettone o pandoro in un caso o nell'altro. (Ok un esempio abbastanza assurdo). Benissimo ora volete controllare se mettere un panettone in una cesta azzurra, benissimo vi basta controllare che il vostro amico sia:

RAGAZZO AND AMANTECANDITI

Se entrambe sono vere, il risultato sarà vero e quindi potrete riempire il vostro cesto azzurro con il panettone, ma se almeno una delle due è falsa, tutta l'espressione è falsa, quindi non ha senso che prepariate quel cesto.

Spero che l'esempio sia stato abbastanza chiaro, anche nel caso dell'AND come per il NOT noi programmatori abbiamo una notazione tutta nostra, ovvero al posto della parola AND usiamo il doppio "ampersand", ovvero `&&`.

N.B: per controllare se una variabile e un'altra sono vere si usa la doppia `&&` e non singola, che invece si riferisce ad un'altra cosa che non tratteremo in questo tutorial.

Tabella di AND trasformata con il doppio ampersand:

Operando 1	Operando 2	Operazione Logica	Risultato Finale
FALSE	FALSE	FALSE && FALSE	FALSE
FALSE	TRUE	FALSE && TRUE	FALSE
TRUE	FALSE	TRUE && FALSE	FALSE
TRUE	TRUE	TRUE && TRUE	TRUE

Benissimo, veniamo ora al terzo ed ultimo operatore che prenderemo in esame in questa mini guida, si tratta dell'operatore OR, anche questo prende due operandi e restituisce TRUE se almeno uno dei due operandi è TRUE, evitiamo di dilungarci esageratamente in spiegazioni e vediamo subito la tabella di verità, che sicuramente rende tutto più chiaro.

Operando 1	Operando 2	Operazione Logica	Risultato Finale
FALSE	FALSE	FALSE OR FALSE	FALSE
FALSE	TRUE	FALSE OR TRUE	TRUE
TRUE	FALSE	TRUE OR FALSE	TRUE
TRUE	TRUE	TRUE OR TRUE	TRUE

Vedete che il valore è FALSE solo quando entrambi sono FALSE, e questo quando si usa? Semplice, se volete che almeno una delle due condizioni sia vera e non vi interessa sapere che lo siano entrambi. In informatica l'OR viene sostituito con il simbolo ||.

N.B: si usa il doppio pipe || e non il singolo.

Ecco la tabella con l'operatore valido per l'ANSI C

Operando 1	Operando 2	Operazione Logica	Risultato Finale
FALSE	FALSE	FALSE FALSE	FALSE
FALSE	TRUE	FALSE TRUE	TRUE
TRUE	FALSE	TRUE FALSE	TRUE
TRUE	TRUE	TRUE TRUE	TRUE

Spero che fino a qui tutto sia stato chiaro, abbiamo dei valori di verità, ovvero vero o falso e poi abbiamo degli operatori per interagire con questi valori, ora andiamo a vedere gli operatori di confronto che uniscono i concetti descritti all'inizio del tutorial con quelli descritti adesso.

Operatori di Confronto

Ora che abbiamo chiari i concetti di variabili, operatori aritmetici, valori booleani e operazioni logiche, possiamo fare un passettino in avanti prendendo in esame gli operatori di confronto. Gli operatori di confronto prendono due operandi, spesso si tratta di due variabili e le

confrontano tra loro, il risultato del confronto poi è un valore di verità, ecco un esempio:

Vogliamo verificare se la variabile `altezzaTriangolo` è maggiore della variabile `latoQuadrato`, faremo:

```
altezzaTriangolo > latoQuadrato = ?
```

Il risultato sarà `TRUE` se è maggiore o `FALSE` se è minore.

La cosa è molto semplice, prendiamo due valori, li mettiamo a confronto e vediamo uscire fuori `TRUE` o `FALSE`, benissimo allora quali sono gli operatori di confronto? Ne abbiamo già mostrati due nel corso del tutorial, ma ora vediamo una tabella riassuntiva:

Operando	Simbolo ANSI C
uguale a	<code>==</code>
diverso	<code>!=</code>
maggiore	<code>></code>
minore	<code><</code>
maggiore o uguale	<code>>=</code>
minore o uguale	<code><=</code>

N.B: Fate molta molta moltissima attenzione all'operatore di confronto UGUALE A, in pratica io voglio controllare se una variabile è uguale a un'altra e avere `TRUE` o `FALSE`, quindi uso il doppio simbolo di `=`, se usassi il singolo simbolo di `=` farei un'assegnazione e non è quello che voglio, quindi guardate l'esempio sotto

```
int variabile1 = 10;  
int variabile2 = 20;
```

```
variabile1 == variabile2 //Il risultato è FALSE perché diverse  
variabile1 = variabile2 //Il risultato è che ora variabile1  
                        //vale 20 perché gli ho assegnato il  
                        //valore di variabile2.
```

Credo sia tutto, siamo giunti alla fine di questo lungo tutorial, spero vivamente che i concetti siano abbastanza chiari, nel prossimo tutorial andremo a vedere come queste cose sono utilizzate all'interno del codice, aggiungendo nuovi costrutti del linguaggio. Buono studio.

Manuel (aka Jekrom)

per segnalazioni : 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/>