IL LINGUAGGIO C: GLI OPERATORI

In questo Workshop riprendiamo quanto abbiamo visto sulla programmazione introducendo sia in modo teorico che 'pratico', attraverso una serie di esempi, gli operatori con cui manipoleremo i dati e le variabili.

Nel fascicolo 19 abbiamo introdotto il concetto di **variabile** e abbiamo visto in che modo è possibile **dichiarare** queste 'entità' utili per memorizzare dati e informazioni. Abbiamo, tuttavia, lasciato in sospeso le problematiche relative al loro utilizzo pratico. Archiviare un dato in memoria, infatti, è un'operazione utile nella scrittura di un software, ma è ancora più utile poter **manipolare** ed **elaborare** le informazioni. A questo fine, proprio come avviene in ambito matematico, esiste un vasto insieme di strumenti che permette di interagire con variabili e costanti: stiamo parlando degli 'operatori'. In questo articolo vedremo nel dettaglio i principali operatori forniti dal linguaggio C, dividendoli in categorie associate ad alcuni semplici esempi di utilizzo che ne chiariranno il significato. Tali esempi, rappresentano solo una 'frazione' di codice, non sufficiente a essere testata all'interno dell'ambiente Dev-C++ installato nel Workshop 16.

GLI OPERATORI MATEMATICI

I primi operatori che presentiamo sono quelli 'matematici', la cui funzione è identificabile a partire dal loro nome: permettono di svolgere le operazioni matematiche fondamentali. Vediamo un loro schema riassuntivo nella tabella sottostante:

Operatore	Sintassi	Funzione svolta
=	Var1 = Var2	Operatore di **assegnamento**. Assegna alla variabile *Var1* il valore della variabile (o costante) *Var2*.
+	Var1 + Var2	Operatore di **somma**. Somma la variabile (o costante) *Var1* alla variabile (o costante) *Var2*.
-	Var1 - Var2	Operatore **differenza**. Sottrae dalla variabile (o costante) *Var1* il valore della variabile (o costante) *Var2*.
-	- Var1	Operatore **negazione**. Nega la variabile (o costante) *Var1*.
*	Var1 * Var2	Operatore **moltiplicazione**. Moltiplica la variabile (o costante) *Var1* per il valore della variabile (o costante) *Var2*.
/	Var1 / Var2	Operatore **divisione**. Divide la variabile (o costante) *Var1* per il valore della variabile (o costante) *Var2*.
%	Var1 % Var2	Operatore **modulo**. Calcola il resto intero ottenuto dalla divisione *Var1*/*Var2*.
( )		**Parentesi tonde**. Come in matematica sono utilizzabili per definire le priorità di esecuzione dei calcoli.

Esempio 1: operatori matematici

```
float circonferenza;
float raggio = 15.0;  /*utilizziamo l'operatore 'assegnamento' per inizializzare la
                        variabile */
const float PI_greco = 3.14159;     /*utilizziamo l'operatore 'assegnamento'
                                    per inizializzare la costante */

circonferenza = 2 * PI_greco * raggio; /* utilizziamo l'operatore 'moltiplicazione' per
                                       calcolare i prodotti tra '2', 'Pi_greco' e
                                       'raggio' e l'operatore 'assegnamento' per asse -
                                       gnare il risultato dell'operazione alla varia-
                                       bile 'circonferenza */
```

Esempio 2: operatori matematici

```
float Var1 = 10.0;    /* utilizziamo l'operatore 'assegnamento' per
                         inizializzare le variabili Var1 e Var2*/
float Var2 = 15.0;
float Var3;
float media;

Var3 = 3.0;    /* utilizziamo l'operatore 'assegnamento' per assegnare a
                  Var3 il valore numerico 3.0 */

media = (Var1 + Var2 + Var3) / 3 ;      /* calcoliamo il valore medio delle variabili
                                        Var1, Var2 e Var3: usiamo l'operatore 'somma'
                                        per sommare le variabili, le parentesi tonde
                                        per far eseguire la somma prima della divisione
                                        e l'operatore divisione per dividere la somma
                                        calcolata per 3*/
```

GLI OPERATORI DI 'INCREMENTO' E 'DECREMENTO'

Sono due operatori che agiscono sulla variabile indicata incrementandone o decrementandone il valore di una unità. Possono essere utilizzati in due modi differenti (prefissi o postfissi).

Operatore	Sintassi	Funzione svolta
++	Var++	Operatore di 'postincremento'. Prima svolge l'espressione in cui è inserita la variabile Var e successivamente la incrementa.
– –	Var – –	Operatore di 'postdecremento'. Prima svolge l'espressione in cui è inserita la variabile Var e successivamente la decrementa.
++	++Var	Operatore di 'preincremento'. Incrementa immediatamente la variabile Var e successivamente svolge l'espressione in cui è inserita.
– –	– – Var	Operatore di 'predecremento'. Decrementa subito la variabile Var e successivamente svolge l'espressione in cui è inserita.

Esempio 1: postincremento

```
Var = 2;
Risultato = 2*Var++;  /* dopo questa linea di codice, Risultato vale '4', mentre Var
                         vale '3' */
```

Esempio 2: postdecremento

```
Var = 5;
Risultato = 2*Var--;  /* dopo questa linea di codice, Risultato vale '10', mentre Var
                         vale '4' */
```

Esempio 3: preincremento

```
Var = 2;
Risultato = 2*++Var;  /* dopo questa linea di codice, Risultato vale '6', mentre Var
                         vale '3' */
```

Esempio 4: predecremento

```
Var = 5;
Risultato = 2*--Var;  /* dopo questa linea di codice, Risultato vale '8', mentre Var
                         vale '4' */
```

GLI OPERATORI 'BIT A BIT'

Questo tipo di operatori agisce sulla 'struttura binaria' delle variabili modificando i singoli bit.

Operatore	Sintassi	Funzione svolta
~	~ Var	Restituisce il valore negato (bit per bit) della variabile *Var*.
&	Var1 & Var2	Effettua un AND logico bit a bit tra le variabili (o costanti) *Var1* e *Var2*.
|	Var1 | Var2	Effettua un OR logico bit a bit tra le variabili (o costanti) *Var1* e *Var2*.
^	Var1 ^ Var2	Effettua un XOR logico bit a bit tra le variabili (o costanti) *Var1* e *Var2*.
>>	Var1 >> n	Effettua uno 'shift' di n bit verso destra (ossia i bit della variabile vengono fatti scorrere verso destra).
<<	Var1 << n	Effettua uno 'shift' di n bit verso sinistra (ossia i bit della variabile vengono fatti scorrere verso sinistra).

Esempio 2: assegnamenti composti

```
Var=10;
Var*=20;  /* equivale a Var = Var*20;    Dopo questa linea, Var ha valore '200' */
```

OPERATORI DI TEST E CONFRONTO

Permettono di effettuare test tra due espressioni (o variabili) restituendo un risultato 'booleano'. Ricordiamo, tuttavia, che il C non gestisce il tipo 'booleano', ma utilizza una convenzione particolare per rappresentare i valori 'vero' e 'falso', che vedremo quando affronteremo le strutture di controllo.

Operatore	Sintassi	Funzione svolta
= =	Expr1 = = Expr2	**Uguaglianza**. Confronta le espressioni *Expr1* ed *Expr2* restituendo 'vero' se sono uguali, 'falso' altrimenti.
! =	Expr1 ! = Expr2	**Disuguaglianza**. Confronta le espressioni *Expr1* ed *Expr2* restituendo 'vero' se sono differenti, 'falso' altrimenti.
>	Expr1 > Expr2	**Strettamente maggiore**. Confronta le espressioni *Expr1* ed *Expr2* restituendo 'vero' se *Expr1* è maggiore di *Expr2*, 'falso' altrimenti.
> =	Expr1 > = Expr2	**Maggiore o uguale**. Confronta le espressioni *Expr1* ed *Expr2* restituendo 'vero' se *Expr1* è maggiore o uguale a *Expr2*, 'falso' altrimenti.
<	Expr1 < Expr2	**Strettamente minore**. Confronta le espressioni *Expr1* ed *Expr2* restituendo 'vero' se *Expr1* è minore di *Expr2*, 'falso' altrimenti.
< =	Expr1 < = Expr2	**Minore o uguale**. Confronta le espressioni *Expr1* ed *Expr2* restituendo 'vero' se *Expr1* è minore o uguale a *Expr2*, 'falso' altrimenti.

Esempio 1: operatori di test o confronto

```
int Var1 = 10;  /* inizializzazione delle variabili */
int Var2 = 22;

Var1 == Var2;   /* test con esito 'falso' */
Var1 != Var2;   /* test con esito 'vero' */

Var1 = Var2;   /* reinizializziamo Var1 con il valore di Var2 */

Var1 == Var2;   /* test con esito 'vero' */
Var1 != Var2;   /* test con esito 'falso' */

Var1 >= Var2;   /* test con esito 'vero' */
Var1 > Var2;     /* test con esito 'falso' */
```

OPERATORI LOGICI

Effettuano operazioni logiche su espressioni booleane (ad esempio condizioni, test ecc). Sono utili nella costruzione delle espressioni condizionali.

Operatore	Sintassi	Funzione svolta
!	! Expr1	Negazione. Nega il valore logico di *Expr1*.
&&	Expr1 && Expr2	Prodotto logico (AND). Effettua il prodotto logico (AND logico) tra le espressioni *Expr1* ed *Expr2*.
||	Expr1 || Expr2	Somma logica (OR). Effettua la somma logica (OR logico) tra le espressioni *Expr1* ed *Expr2*.

Esempio 1: operatori logici

```
(1>0) && (2>=4);   /*questa espressione restituisce 'falso'. Infatti, mentre è
                   vero che 1 è maggiore di 0, l'espressione 2>=4 risulta falsa.
                   Quindi, ricordando le tabelle di verità incontrate quando abbiamo
                   parlato degli operatori logici dell'elettronica digitale avremo:
                   vero AND falso = falso    */
```

Esempio 2: operatori logici

```
int Var = 10;
char c= 'a';
char d= 'a';


!(Var!=15) || (c==d);        /* questa espressione logica è vera. Var è differente da 15
                             (quindi Var!=15 è 'vero'); tuttavia, per effetto
                             dell'operatore '!', il primo termine dell'espressione risulta
                             'falso'. Il secondo termine dell'espressione, invece, è
                             chiaramente vero, poiché la variabile 'c' (di tipo carattere)
                             ha valore 'a' e, allo stesso tempo, anche la variabile 'd'
                             è inizializzata con il carattere 'a'. Risulta quindi vera
                             l'uguaglianza e, con essa, l'espressione logica di destra.
                             L'OR logico finale viene, allora, calcolato come segue:
                             falso OR vero = vero */
```

TESTARE GLI ESEMPI

Come detto inizialmente, negli esempi sono mostrate solo alcune righe di codice C, che però non sono sufficienti a realizzare un vero programma 'compilabile' ed 'eseguibile'. Lo scopo di tali esempi è, invece, quello di mostrare come vengono applicati concretamente gli operatori descritti. Nei prossimi fascicoli, invece, sperimenteremo l'uso delle variabili e degli operatori in modo concreto, provando a scrivere alcuni semplici programmi che utilizzino le due funzioni di input e output scanf e printf (la prima ancora da 'sperimentare', mentre la seconda già incontrata in precedenza, anche se in modo molto semplificato) per acquisire e visualizzare dati elementari.

Esempio 1: operatori 'bit' a 'bit'

```
int RisultatoAND;
int RisultatoOR;
int RisultatoXOR;
int RisultatoSHIFT;
int Var1 = 0b000110;
int Var2 = 0b000101; /* dichiariamo le due variabili Var1 e Var2 e le inizializziamo
                     con valori numerici espressi in binario (il prefisso '0b' indica
                     al compilatore che quello che segue è un numero in forma
                     binaria) */

RisultatoAND = Var1 & Var2;  /* eseguiamo l'AND logico bit a bit tra le due variabili
                              Var1 e Var2 e assegniamo il risultato a RisultatoAND.
                              Il risultato ottenuto sarà 000110 & 000101 = 000100 */

RisultatoOR = Var1 | Var2;  /* eseguiamo l'OR logico bit a bit tra le due variabili
                            Var1 e Var2 e assegniamo il risultato a 'RisultatoOR'.
                            Il risultato ottenuto sarà 000110 | 000101 = 000111 */

RisultatoXOR = Var1 ^ Var2;  /* eseguiamo lo XOR logico bit a bit tra le due variabili
                             Var1 e Var2 e assegniamo il risultato a 'RisultatoXOR'.
                             Il risultato ottenuto sarà 000110 ^ 000101 = 000011 */

RisultatoSHIFT = Var1<<2;    /* 'shift' di Var1 di due posizioni verso sinistra e
                             assegnamento a RisultatoSHIFT, che ora vale 011000 */
```

ASSEGNAMENTI COMPOSTI

Sono operatori che permettono di 'abbreviare' la scrittura di espressioni matematiche che comportano la reinizializzazione di una variabile.

Operatore	Sintassi	Funzione svolta
+=	Var += Espressione	Var = Var + Espressione
-=	Var -= Espressione	Var = Var - Espressione
*=	Var *= Espressione	Var = Var * Espressione
/=	Var /= Espressione	Var = Var / Espressione
&=	Var &= Espressione	Var = Var & Espressione
|=	Var |= Espressione	Var = Var | Espressione

Esempio 1: assegnamenti composti

```
Var = 2;
Var +=10; /* equivale a Var = Var+10;    Dopo questa linea, Var ha valore '12' */
```