

Esercitazione Programmazione Procedurale con Laboratorio

Espressioni

Esercizio 1 Associatività e valutazione espressioni

Valutare le seguenti espressioni (cioè riportare il loro valore):

- $12 / 2 \% 5$
- $a = b = 5$
- $5 + 2 * 3$
- $5 + 2 + ++pluto$ con $int\ pluto = 4$
- $5 + 2 + pluto++$ con $int\ pluto = 4$
- $2 + 4 == 8 - 2$
- $sizeof(int) / 2 > 2$
- $(7 != 6) + 1$
- $3 \&\& 6$
- $3 > 4 \ || \ 6 <= 5$
- $f/4 + f * 2 / ++f$ con $float\ f = 4$
- $f-- + c-- + d - ++e$ con $int\ c = 2, d = 4, e = 6, f = 5$. Una volta apportati i cambiamenti in memoria, quanto valgono f, c, d, e ?

Soluzione dell'esercizio 1

- 1
- 5
- 11
- 12
- 11
- 1
- 0
- 2
- 1
- 0
- 2.6

- $4, c = 1, d = 4, e = 7, f = 4$

Esercizio 2 Valutazione espressioni

Definire il tipo delle seguenti espressioni. Inserire le parentesi in modo da seguire l'ordine della valutazione in base alla priorità degli operatori. Esempio per $-i + 4 == 3$, viene valutata come se le parentesi fossero $((-i) + 4) == 3$.

```

1 char c = 'A';
2 int i = 7, j = 7;
3 double x = 0.0, y = 2.3;
4
5 ! c;
6 ! (i - j);
7 ! i - j;
8 !! (x + y);
9 ! x * ! ! y;
10 i <= j;
```

Soluzione dell'esercizio 2

Le espressioni sono tutte di tipo intero:

- $!c$, valore 0
- $!(i - j)$, valore 1
- $(!i) - j$, valore -7
- $!(!(x + y))$, valore 1
- $(!x) * (!(!y))$ valore 1
- Errore, $<=$ non fa parte della sintassi del linguaggio.

Esercizio 3 Valutazione espressioni

Definire il tipo delle seguenti espressioni. Inserire le parentesi come nell'esercizio precedente.

```

1 char c = 'B';
2 int i = 3, j = 3, k = 3;
3 double x = 0.0, y = 2.3;
4
5 i && j && k;
6 x || i && j - 3;
7 i < j && x < y;
8 i < j || x < y;
9 'A' <= c && c <= 'Z';
10 c - 1 == 'A' || c + 1 == 'Z';
```

Soluzione dell'esercizio 3

Le espressioni sono tutte di tipo intero:

- $(i \&\& j) \&\& k$, valore 1
- $x \parallel (i \&\& (j - 3))$, valore 0
- $(i < j) \&\& (x < y)$, valore 0
- $(i < j) \parallel (x < y)$, valore 1
- $('A' <= c) \&\& (c <= 'Z')$, valore 1
- $((c - 1) == 'A') \parallel ((c + 1) == 'Z')$ valore 1

Esercizio 4 Operatori di incremento

Scrivere cosa stampano le seguenti *printf*, oppure dire se provocano un errore. Rispondere anche alle domande in commento.

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     double a= 3.5;
6     double b = 0.0;
7     printf("%.1f\n", a++);
8     printf("%.1f\n", a);
9
10    printf("%.1f\n", ++b);
11    printf("%.1f\n", b);
12
13    int c= -3;
14    printf("%d\n", -c++); // Aggiungere le parentesi su -c++ seguendo l'ordine di valutazione
15    printf("%d\n", c);
16
17    int d= 0;
18    printf("%d\n", ++d++);
19 }
20

```

Soluzione dell'esercizio 4

Le prime sei stampe sono

```

3.5
4.5
1.0
1.0
3
-2

```

$-c++$ viene interpretato come $-(c++)$ dato che l'operatore $++$ postfisso ha priorità più alta dell'operatore $-$.

L'ultima stampa provoca un'errore: l'espressione viene valutata come $++(d++)$ dato che l'operatore $++$ postfisso ha priorità più alta dell'operatore $++$ prefisso: $d++$ viene valutata come 0 (provocando poi l'effetto collaterale (*side effect*) di incrementare d di 1, una volta incontrato un *sequence point*), ma 0 non è un *lvalue* quindi non gli si può applicare l'operatore $++$ prefisso; sarebbe equivalente ad effettuare $++0$.

Esercizio 5 Tipo espressione condizionale

Valutare e dare il tipo delle seguenti espressioni condizionali. Ricordarsi che il tipo di un'espressione $expr1? expr2: expr3$ condizionale è uno solo, e dipende dai tipi delle espressioni $expr2$ e $expr3$: utilizzare le regole di conversione di tipo tra $expr2$ e $expr3$ per trovare il tipo di tutta l'espressione.

```

1 char a= 'a', b= 'b'; // 'a' ha valore 97
2 int i= 1, j= 2;
3 double x= 7.07;
4
5 i == j ? a - 1 : b + 1;
6 j % 3 == 0 ? i + 4: x;
7 j % 3 ? i + 4: x;

```

Soluzione dell'esercizio 5

| | |
|--------------------------------|--------------------------|
| $(i == j) ? (a - 1) : (b + 1)$ | Valore 99, Tipo int |
| $((j \% 3) == 0) ? (i + 4): x$ | Valore 7.07, Tipo double |
| $(j \% 3) ? (i + 4): x$ | Valore 5.0, Tipo double |

Esempio, nel terzo caso sopra, $i + 4$ ha tipo *int*, ma viene ritornato un valore 5.0 di tipo *double*, dato che $expr3$ ha tipo *double*. Il tipo di $expr2$ viene quindi convertito a quello di $expr3$.

Esercizio 6 Operatore virgola

Studiare la seguente porzione di codice; rispondere alle domande nei commenti. Infine controllare la correttezza delle proprie risposte eseguendo tale codice (all'interno di una funzione *main()*).

```

1 int a= 0;
2 double b= 3.5;
3
4 //Aggiungere il della variabile c (per avere una definizione), in modo che non ci siano conversioni di
  tipo
5 c= (a = 5, b = b + 0.3);
6
7 // Cosa stampa?
8 printf("%d\n%f\n%f\n", a, b, c);
9
10 // Aggiungere il tipo della variabile d, in modo che non ci siano conversioni di tipo
11 d= (a++, --a + 2);
12
13 // Cosa stampa?
14 printf("%d\n%d\n", a, d);
15
16 int i, j, k= 3;
17 double x= 3.3;
18
19 // Aggiungere il tipo della variabile f, in modo che non ci siano conversioni di tipo
20 f= (i = 1, j = 2, ++k + 1);
21
22 // Aggiungere il tipo della variabile g, in modo che non ci siano conversioni di tipo
23 g= (j= k != 1, ++x * 2.0 + 1);
24
25 // Cosa stampa?
26 printf("%d\n%d\n%d\n%f\n%d\n%f\n", i, j, k, x, f, g);

```

Soluzione dell'esercizio 6

```

1 int a= 0;
2 double b= 3.5;
3
4 double c = 0.0;
5 c= (a = 5, b = b + 0.3);
6
7 printf("%d\n%f\n%f\n", a, b, c);
8
9 int d= 0.0;
10 d= (a++, --a + 2);
11
12 printf("%d\n%d\n", a, d);
13
14 int i, j, k= 3;
15 double x= 3.3;
16
17 int f= (i = 1, j = 2, ++k + 1);
18 double g= (j= k != 1, ++x * 2.0 + 1);
19
20 printf("%d\n%d\n%d\n%f\n%d\n%f\n\n", i, j, k, x, f, g);

```

Output del programma precedente

```

5
3.800000
3.800000
5
7
1
1
4
4.300000
5
9.600000

```

Esercizio 7 Precedenza operatori

Studiare la seguente porzione di codice e scrivere quello che si pensa stampi. Cosa succede se si toglie la prima o la seconda coppia di parentesi, o entrambe? Controllare solo alla fine la correttezza delle proprie risposte eseguendo il codice.

```
1 int x= 0, y= 0, z= 0;
2 x = (y = 2) + (z = 3);
3 printf("%d %d %d\n" , x, y, z);
```

Soluzione dell'esercizio 7

5 2 3 (senza togliere parentesi)

5 5 3 (togliendo la prima coppia di parentesi)

error: expression is not assignable (togliendo la seconda coppia)

error: expression is not assignable (togliendo entrambe le coppie)

Togliendo la prima coppia, viene valutata per prima l'espressione $(y = 2)$, a cui poi si somma z . $(y = 2) + z$ NON rappresenta però un *lvalue*, quindi non può stare a sinistra di un operatore di assegnamento. Togliendo la seconda coppia di parentesi invece, viene valutata per prima $2 + z$ (che non è un *lvalue*), a cui si cerca poi di assegnare 3.

Esercizio 8 lvalue e rvalue

Valutare se i seguenti comandi o sequenze di comandi generano un errore del compilatore “*error: expression is not assignable*”, cioè quando un *rvalue* viene (erroneamente) considerato un *lvalue*.

- `int var = 4;`
- `(5 + 1) = 6;`
- `4 = var;`
- `abs(-3) = 3;`
- `enum {JAN,FEB,MARCH}; JAN = 20;`
- `int a = -3 + --3;`

Soluzione dell'esercizio 8

A parte il primo, tutti gli altri punti dell'esercizio generano errore. L'errore è dovuto al fatto che si cerca di assegnare un valore ad un *rvalue*.

Esercizio 9 Effetti collaterali (side effects)

Studiare la seguente porzione di codice e scrivere cosa si pensa stampi a video. Controllare programmando poi l'esempio.

```
1 int a, b =0, c =0;
2 a =++b + ++c;
3 printf("%d %d %d\n" , a, b, c);
4 a= b+++c++;
5 printf("%d %d %d\n" , a, b, c);
6 a= ++b + c++;
7 printf("%d %d %d\n" , a, b, c);
8 a= b-- + --c;
9 printf("%d %d %d\n" , a, b, c);
10 int q;
11 printf("%d\n" , q);
12 int r = 2+ q++;
13 printf("%d %d\n" , r, q);
```

Soluzione dell'esercizio 9

Sul mio ambiente di sviluppo viene stampato:

```
2 1 1
2 2 2
5 3 3
5 2 2
1499863904
1499863906 1499863905
```

Alcuni compilatori C possono inizializzare una variabile a 0, ma, in generale, una variabile (eccetto le variabili globali ed altri casi che poi vedremo) può essere assegnata a qualsiasi valore. INIZIALIZZARE SEMPRE SUBITO TUTTE LE VARIABILI. Questo per rendere il codice portabile su compilatori che non effettuano automaticamente l'inizializzazione a 0.

Esercizio 10 Numero effetti collaterali

Quanti effetti ci sono su ciascuna delle variabili delle seguenti espressioni? In base a questo e agli operatori delle espressioni dire se al momento della compilazione l'espressione genera un warning “*warning: multiple unsequenced modifications*”.

- $a = a++ + b$
- $b = a++ \ \&\& \ a++$
- $a += a++ + (b = 5)$
- $c = a++ + b++$
- $b = (a++, a++, a++)$
- $b += b++ + (a++, a++)$

Soluzione dell'esercizio 10

- Due effetti collaterali su a . I due effetti collaterali sulla stessa variabile non sono separati da un sequence point: l'espressione genera un warning.
- Due effetti collaterali su a ed uno su b . I due effetti collaterali sulla stessa variabile sono separati da un sequence point scandito dall'operatore $\&\&$.
- Due effetti collaterali su a e uno su b . I due effetti collaterali sulla stessa variabile non sono separati da un sequence point: l'espressione genera un warning.
- Un effetto collaterale su a , uno su b , e uno su c . Un solo effetto collaterale su una variabile non può provocare un warning.
- Tre effetti collaterali su a e uno su b . I tre effetti collaterali sulla stessa variabile sono separati da un sequence point scandito dall'operatore virgola.
- Due effetti collaterali su a e due su b . I due su a sono separati da un sequence point dato da $,$, mentre i due su b non sono separati, generando un warning “*warning: multiple unsequenced modifications to 'b'*”.