.

Espressioni con effetti collaterali

Stefano Ferrari

Università degli Studi di Milano
stefano.ferrari@unimi.it

Programmazione

anno accademico 2016–2017

Operatore di assegnamento

In C l'assegnamento (=) è un operatore, e quindi ha un valore che è il valore dell'espressione a destra di =

Esempio: `i = 5` vale 5 `b = (i > 7)` vale *falso*

Quindi si può assegnare un assegnamento: `l1 = l2 = 5;`

L'assegnamento è associativo a destra: `l1 = (l2 = 5);`

- prima si esegue `l2 = 5;`
- poi si esegue `l1 =`"valore di `(l2 = 5)`"`;`, che è 5

Oltre al valore, l'assegnamento ha un effetto collaterale (side effect): modifica il valore dell'operando a sinistra di =

L'effetto collaterale ha luogo dopo la valutazione dell'espressione

.

Assegnamento e uguaglianza

È facile confondere uguaglianza logica (==) con assegnamento (=)

```
i = 1;
b = (i == 4);
```

b vale *false* (cioè 0), perché `i` vale 1

```
i = 1;
b = (i = 4);
```

b vale 4, che viene interpretato come *true*, e anche `i` vale 4

Poiché l'assegnamento ha un valore convertibile in valore logico, il compilatore non segnala errori

Assegnamento composto

L'assegnamento si può comporre con gli operatori aritmetici

variabile += espressione	→	variabile = variabile + (espressione)
variabile −= espressione	→	variabile = variabile − (espressione)
variabile *= espressione	→	variabile = variabile * (espressione)
variabile /= espressione	→	variabile = variabile / (espressione)

Si notino le parentesi:

1. si valuta l'espressione
2. si esegue l'operazione sui valori della variabile e dell'espressione
3. si assegna il risultato alla variabile

Esempio: `i += j * k` significa `i = i + (j * k)`
`i *= j + k` significa `i = i * (j + k)`

.

Incremento e decremento

L'operatore ++ (incremento) incrementa di 1 una variabile intera

L'operatore -- (decremento) decrementa di 1 una variabile intera

Si possono usare prefissi (`++i` e `--i`) o postfissi (`i++` e `i--`)

- se prefissi, eseguono l'operazione subito
- se postfissi, la eseguono per ultima prima dell'istruzione seguente

`l2 = ++l1;`	`l2 = l1++;`
incrementa `l1` assegna `l2 = l1`	assegna `l2 = l1` incrementa `l1`
`l1` vale `l2`	`l1` vale `l2+1`

È pericoloso usarli in espressioni composte

Tabella delle priorità

Priorità alta	`!` `++` `--` `-` (unario)
	`*` `/` `%`
	`+` `-` (binario)
	`>` `>=` `<` `<=`
	`==` `!=`
	`&&`
	`||`
Priorità bassa	`=`

.

Esempi regolari

Supponiamo di avere dichiarato e inizializzato alcune variabili intere

```
int a, b, c, d, e, f;
a = 1; b = 2; c = 3; d = 4; e = 5;
```

Valutiamo alcune espressioni composte

- `d = a * b / c;`
  `d = 1 * 2 / 3;` — `d` *vale* 0
- `d = ++a * b - c;`
  `d = 2 * 2 - 3;` — `d` *vale* 1, `a` *vale* 2
- `e = 5 + c * d / e;`
  `e = 5 + 3 * 4 / 5;` — `e` *vale* 7 *(non* 5*!)*
- `e = 30 / e++ + 29 % c;`
  `e = 30 / e++ + 29 % c;` — `e` *vale* 9 *(non* 7*!)*

Esempi regolari (2)

Supponiamo di avere dichiarato e inizializzato alcune variabili intere

```
int a, b, c, d, e, f;
a = b = c = d = e = 2; f = 1;
```

Valutiamo l'espressione

```
a = b += c++ - d + --e / -f;
```

```
a = b += 2 - 2 + 1 / -1;
```

```
a = 1
```

- `a` vale 1, `b` vale 1, `c` vale 3, `e` vale 1
- le altre variabili sono invariate

.

Esempi problematici

- L'ordine di valutazione degli operandi può influire sul valore e sull'effetto collaterale dell'espressione

```
(b = a + 2) - (a = 1)
```

- La valutazione cortocircuitata (interrompere la valutazione appena determinato il valore dell'espressione) può influire sul valore e sull'effetto collaterale dell'espressione

```
c = (a = 1) || (b = 2)
```

Lo standard C lascia libertà su questi punti, consentendo comportamenti indefiniti a priori

Perché lo fa?

Istruzioni

In C un'istruzione è qualsiasi espressione terminata da ;

Esempio: questa istruzione valuta `i`, somma 1 al risultato e lo getta via

```
i+1;
```

A che scopo consentire questa stranezza?

Affinché la parte esecutiva di un blocco abbia una struttura regolare manipolabile meccanicamente dal compilatore, cioè sia una sequenza di

- istruzioni semplici, composte da *espressione* e ;
- blocchi, a loro volta con la stessa struttura

In pratica, è ragionevole che ogni istruzione abbia un effetto collaterale

(e che ne abbia uno solo)

.

Istruzioni, direttive, dichiarazioni

In C, istruzione è tutto e solo ciò che termina con ;

Le direttive non sono istruzioni!

- vengono trattate dal precompilatore e non dal compilatore
- il processore non le "vede" nemmeno

Le dichiarazioni di variabili e di procedure sono istruzioni!

- aggiungono righe alla tabella dei simboli, per poterli usare in seguito
- indicano al processore quanta memoria riservare per le variabili e per i dati e i risultati delle procedure