Alma Mater Studiorum—Università di Bologna
Ingegneria elettronica per l'energia e l'informazione, Campus di Cesena
Fondamenti di Informatica A, a.a. 2019/2020

Laboratorio 2: Valutazione e typing di espressioni

Sara Montagna

20 Marzo 2020

1 Operazioni preliminari

Un semplice programma per valutare le prime espressioni in C La valutazione dell'insieme di espressioni che vi verranno proposte di seguito, la faremo attraverso il sorgente `valutazione.c`

- Aprire il sorgente `valutazione.c`
- Compilare...
  - Leggere sempre **attentamente** i warning segnalati: la presenza di un warning non impedisce la creazione dell'eseguibile, ma è sempre riflesso di una qualche imprecisione nell'implementazione del programma, i cui effetti si possono vedere all'atto dell'esecuzione
- Eseguire...
- Seguendo il template della valutazione dell'espressione e della stampa del risultato mostrata nel sorgente, aggiungere o sostituire di volta in volta l'espressione da valutare nel sorgente
- **Compilare il sorgente a seguito di ogni modifica**
- **Prima** di eseguirle nel programma, cercare **sempre** di anticipare quale sarà il risultato della valutazione
- Se quanto avete pensato corrisponde a quanto risponde il software, bene!
- Se non corrisponde, bene lo stesso, a patto che ne capiate la ragione
- Se ottenete un risultato inaspettato e non sapete giustificarlo, consultate il docente!

2 Valutazione di Espressioni

Operazioni su interi – elementi di base Valutare le seguenti espressioni attraverso l'istruzione `printf("L'espressione si valuta in %d\n", ...);`

- `1+2`
- `5*8`
- `10/2*4+5/2` — Chi ha la priorità?
- `10/3` — Quanto fa? 3, $3.\bar{3}$ o 4?
- `10%3` — Operatore "modulo"
- `6/2*(1+2)` — Quanto fa? 1 o 9? Perché?
- `4+4*4+4*4-4/4+4*4`

Operazioni su interi – complemento a 2, overflow Premesse:

1. Le seguenti espressioni sono pensate per variabili di tipo `int`. In questo esercizio non dichiarare le variabili e non valutare il risultato dell'espressione come `unsigned int`
2. Verificare, attraverso l'operatore `sizeof(int)` (si veda il sorgente `size.c` allegato all'ercitazione), quanti byte sono necessari per rappresentare una variabile di tipo `int` nella vostra architettura
3. Il seguente esercizio è pensato per la maggior parte delle architetture – compresa quella dei pc del laboratorio – in cui vengono dedicati 4 byte, ovvero **32 bit**, a una variabile di tipo `int`
4. Come per gli esercizi precedenti, valutare le seguenti espressioni attraverso l'istruzione `printf("L'espressione si valuta in %d\n'', ...);`

- `8-7` — Trasformate 7 in complemento a 2, quindi sommate!
- `2147483647+1` — Cos'è successo???
- `2147483647*2`
- `-2147483648-1`
- `-(-2147483648)`
- `-(-21474836481)` — La `l` finale indica al compilatore di interpretare quel numero come tipo `long`: i `long`, a differenza degli `int`, hanno **64bit** nella maggior parte delle architetture. Verificare quanti byte vengono dedicati ai long sul computer che state usando.

Operazioni su numeri in virgola mobile Il linguaggio C consente di operare con numeri decimali. Ne esistono di due tipi `float`, rappresentati nella maggior parte della architetture con **32 bit**, e `double`, rappresentati generalmente con **64 bit**. Valutare le seguenti espressioni attraverso l'istruzione `printf("L'espressione si valuta in %f\n", ...);`

- `1.0` — Aggiungere una parte decimale ad un numero lo fa interpretare come `double`.
- `1+2.1` — Un'operazione fra un `int` e un `double` ha come risultato un `double`
- `10.0/3` — Confrontare il risultato con `10/3`
- `10.0f/3` — Posporre `f` a un numero forza C ad interpretarlo come `float`. Notare come i `float` siano meno precisi dei `double`, confrontando questo risultato con il precedente
- `15.45E4` — Supporto alla notazione scientifica

Errori sintattici Provare queste espressioni ed osservare con cura il tipo di errore

- `1+>3`
- `1+(2`
- `1e`
- `1e/5`
- `1)` — Quale errore è segnalato?
- `1t`
- `"a`

Provate a commettere altri errori di sintassi

Errori semantici Provare queste espressioni ed osservare con cura il tipo di errore. Cambiare l'identificatore in modo opportuno a seconda del tipo associato all'espressione.

- `1/0`
- `1.0/0` — I numeri in virgola mobile supportano ∞!
- `-1.0/0`
- `1%0`
- `1.0%0` — Cosa succede?
- `2147483648` — Come si più risolvere?
- `a` — Come si indicano i caratteri?

3 Piccoli programmi

Esercizio 1 Si faccia rifrimento al sorgente `test_overflow.c`. Il programma dichiara una variabile intera (`int`) `x` e la inizializza a 1. Successivamente, finché `x` risulta strettamente maggiore di 0, la variabile `x` viene incrementata. Gli incrementi cessano non appena il valore di x diventa minore o uguale a zero. Nel caso in cui il ciclo termini, viene stampato un messaggio con l'ultimo valore assunto dalla variabile `x` .

- Il programma termina? Provare a rispondere alla domanda senza eseguire il programma; una volta formulata una risposta, verificarla compilando ed eseguendo il programma. Giustificare il risultato che si ottiene.

4 Esercizi avanzati

Esercizio 2 Tester di palindromi [1].
Un palindromo è un numero o una frase di un testo che si legge all'indietro e in avanti. As esempio, ognuno dei seguenti numeri interi a cinque cifre è palindromo: 12321, 55455, 88888. Scrivere un programma in C che legga un numero intero di cinque cifre e determini se sia o meno un palindromo.

Esercizio 3 Stampare l'equivalente decimale di un numero binario [1] .
Inserite un numero intero di 4 cifre contente soltanto zeri e uni, che interpreterete come numero "binario", e stampate il suo equivalente decimale.

- Non inserire numeri con uno `'0'` come prima cifra;
- Suggerimento: usate gli operatori di divisione e resto per ottenere le cifre del numero "binario" una alla volta da destra a sinistra. Dopodiché moltiplicate la cifra ottenuta per il suo valore posizionale. Ad esempio, l'equivalente decimale del numero binario 1011 è $1 * 2^0 + 1 * 2^1 + 0 * 2^2 + 1 * 2^3$ ovvero $1 * 1 + 1 * 2 + 0 * 4 + 1 * 8$ ovvero 11.

[1] Esercizio preso – a meno di modifiche minori – dal Capitolo 3 del testo *Paul J. Deitel, Harvey M. Deitel, Il linguaggio C: fondamenti e tecniche di programmazione 8/ed, Pearson, 2016, ISBN 978-8891901651*