Lettura numeri interi

mobytrick

5 settembre 2014

1 Preambolo teorico

Partiamo dalla definizione di *intero*: una quantità numerica intera (ovvero manca la parte decimale), eventualmente preceduta dal segno. Usando la notazione delle *Regular Expression*:

```
[-+]?[0-9]{1,}
```

All'atto pratico, per immettere un intero, sulla tastiera si digiterà un eventuale segno, seguito dalle cifre. Alla fine si premerà il tasto Invio (oppure Enter).

2 Meccanismo della lettura

Consideriamo dapprima la lettura semplice di un intero da standard input (= da tastiera). Per *semplice* si intende che il formato sia `%d`, quindi senza nessuna indicazione circa il numero di cifre da leggere. All'incontro della prima funzione per la lettura si inizia ad immettere il valore. I tasti premuti vengono visualizzati sullo schermo, per controllo. Ad ogni tasto premuto corrisponde una sequenza di bit che viene parcheggiata nel buffer della tastiera. Premendo il tasto Invio i caratteri presenti in tale buffer, **compreso** il carattere newline generato dal tasto Invio, vengono trasferiti nel buffer di input. Contestualmente viene istituito un indicatore che punta al primo carattere. Se la lettura non è la prima, allora sia il buffer di input sia l'indicatore esistono già e quest'ultimo punta al primo carattere da prendere in considerazione. Potrebbe pertanto non corrispondere al primo carattere immesso da tastiera. La lettura consiste nel prelevare dal buffer di input il carattere individuato dall'indicatore e nella sua elaborazione. Se il carattere è considerato valido, allora concorre a formare il valore che poi verrà memorizzato nella variabile. Il carattere viene letteralmente tolto dal buffer di input (*consumato*) e

l'indicatore punta a quello successivo. In caso contrario il carattere rimane e l'indicatore non viene toccato.

Premesso che nella lettura dei dati numerici, non necessariamente interi, i white space character iniziali vengono ignorati (consumati con contestuale avanzamento dell'indicatore), alla lettura del primo carattere non white space si presentano le seguenti possibilità:

1. segno (`+` oppure `-`)
   il segno viene *ricordato*.[1] Si ha l'avanzamento dell'indicatore del buffer, il carattere viene consumato e la lettura prosegue

2. cifra
   il contenuto della variabile da inizializzare viene modificato. Il carattere viene consumato con conseguente adeguamento dell'indicatore

3. altro
   non si ha alcuna modifica del contenuto della variabile da inizializzare. Non ci sono segnalazioni d'errore o di fine dati. L'indicatore nel buffer di input non viene avanzato e la lettura dell'intero è considerata conclusa. Ciò che rimane nel buffer è a disposizione della eventuale successiva operazione di lettura. Non è assolutamente detto che ciò conforme ai desidedata del programmatore.

Se la lettura non è terminata si elabora il carattere individuato dall'indicatore. Si presentano le seguenti possibilità:

1. white space character
   la lettura dell'intero è terminata. Non si ha l'avanzamento dell'indicatore nel buffer di input

2. segno (`+` oppure `-`)
   idem

3. numero
   il carattere concorre a modificare il contenuto della variabile da inizializzare. L'indicatore viene avanzato ed il carattere consumato

4. altro
   la lettura dell'intero è terminata. L'indicatore nel buffer non viene avanzato ed il carattere rimane. Non ci sono segnalazioni

[1] prima della memorizzazione nella variabile, il valore viene complementato

5. end-of-file
   idem

Se la lettura non è terminata si ritorna al primo punto. Al termine della lettura, il valore digitato si trova nella variabile di destinazione. Ma non è detto che ciò che è stato digitato sia poi effettivamente memorizzato. Si pensi ad esempio ad una serie di cifre molto lunga (basta tener premuto un tasto!): il numero immesso non può fisicamente essere memorizzato.

3 Variazioni sul tema

Nella spiegazione del meccanismo di lettura si partiva dal presupposto che il formato di lettura fosse il più semplice possibile (`%d`). Ma esistono delle variazioni sul tema. Ad esempio, si può imporre che la lettura si arresti dopo n cifre. Il formato `%3d` legge al massimo 3 cifre. Se nel buffer ci sono più cifre di quelle previste, la lettura è conforme al format. Se ce ne sono di meno la lettura termina all'esaurimento del buffer di input.

4 Caveat

Spetta al programmatore far coincidere formato di lettura e tipologia della variabile da inizializzare. Per fortuna i moderni compilatori, se richiamati opportunamente, riescono a segnalare le discrepanze. Sia quelle tra formato e tipologia, sia quelle numeriche: deve esserci doppia corrispondenza tra presenza di un formato e variabile da inizializzare.