Fondamenti di Programmazione A
Appunti per le lezioni – *Gianfranco Rossi*

Lettura e scrittura di caratteri

Oltre agli operatori >> e << che realizzano lettura e scrittura per i tipi semplici primitivi `int`, `float`/`double` e `char`, oltre che per le stringhe di caratteri (stile C),
sono presenti anche funzioni per la lettura e scrittura di (soli) caratteri: funzioni `get` e `put`.

Lettura di caratteri (funzione `get`)

```
s.get()
```

legge (= estrae) dallo stream di input `s` il carattere corrente, se presente, e lo restituisce come suo risultato. n.b.: se il dato non è presente, allora attende.

Esempio. Dato

```
char c;
c = cin.get();
```

con input (std input = `cin`)

```
abc
```

assegna a `c` il carattere `'a'`
e lascia sullo stream `cin` i caratteri `bc`.

Successive `get` estraggono i caratteri successivi:

```
c = cin.get();  //assegna 'b' a c
c = cin.get();  //assegna 'c' a c
```

Nell'ipotesi che la sequenza `abc` sia seguita da un “a capo”, una successiva `get`

```
c = cin.get();
```

assegna `'\n'` (“a capo”) a `c`.

n.b. Un singolo carattere può essere letto anche tramite l'operatore di estrazione `>>` applicato ad una variabile di tipo `char`. A differenza della `get`, però, l'operatore `>>` ignora i caratteri “spazio” e “a capo” eventualmente presenti sullo stream di input (li estrae, ma li scarta).

Esempio. Dato

```
char x,y,z;
cin >> x;
cin >> y;
cin >> z;
```

con input

```
a bc
```

(n.b., tra `a` e `b` è presente uno “spazio”) assegna `'a'` a `x`, `'b'` a `y` e `'c'` a `z`, mentre la sequenza di istruzioni

```
char x,y,z;
x = cin.get();
y = cin.get();
z = cin.get();
```

con lo stesso input, assegna `'a'` a `x`, il carattere “spazio” (codice ASCII 32) a `y` e `'b'` a `z`.

Per verificare se la `get` è stata eseguita correttamente si può usare la funzione `fail` applicata alla stream di input (es., `cin.fail()`). Nel caso di fallimento dell'operazione di input il valore restituito dalla `get` è, in generale, non prevedibile. In particolare, per controllare se il carattere letto

è l'"`end_of_file`" si può usare la funzione `eof` applicata allo stream di input (es., `cin.eof()`) o alternativamente, verificare se il valore restituito dalla `get` è uguale alla costante (predefinita) `EOF` (ad es., `cin.get() == EOF`).

Esistono altre forme di `get`. In particolare:

```
s.get(c)
```

estrae dallo stream di input `s` il carattere corrente, se presente, e lo assegna alla variabile `c` (`c` parametro per riferimento, di tipo `char`). Analogo a `c = s.get()`.

Differenze rispetto a `get()`:

- Può essere usata come espressione booleana: se l'estrazione del carattere ha avuto successo, restituisce `true`; altrimenti, restituisce `false` (come l'operatore `>>`)
- Se l'estrazione ha avuto successo, restituisce come suo risultato lo stream di input modificato; è quindi possibile concatenare più `get` tra loro (`get` in cascata). Ad es.:

  ```
  char x,y,z;
  cin.get(x).get(y).get(z);
  ```

  con input

  ```
  abc
  ```

  assegna a `x` il carattere `'a'`, a `y` il carattere `'b'` e a `z` il carattere `'c'` (n.b., operatore `'.'` associativo a sx).

Scrittura di caratteri (funzione `put`)

```
s.put(e)
```

con `s` stream di output ed `e` espressione di tipo `char`. Scrive (= inserisce) il carattere ottenuto dalla valutazione di `e` sullo stream di output `s`.

Esempio. Dato

```
char c = 'a';
cout.put(c);
```

oppure

```
cout.put('a');
```

scrive sullo std output il carattere `'a'` (equivalente a

```
cout << c;
```

(oppure

```
cout << 'a';
```

)).

La `put` restituisce come suo risultato lo stream di output modificato; è quindi possibile concatenare più `put` tra loro (`put` in cascata). Ad es.:

```
cout.put('a').put('\n').put('b');
```

output prodotto (n.b., operatore `'.'` associativo a sx):

```
a
b
```

n.b. Il carattere da stampare può anche essere rappresentato dal suo codice ASCII. Ad es., `cout.put(97)` stampa sullo std output il carattere 'a'.