Fondamenti di Programmazione A

Appunti per le lezioni – *Gianfranco Rossi*

Costanti con nome

Nei linguaggi di programmazione è solitamente possibile associare un nome simbolico ad un valore costante e quindi usare il nome al posto del valore all'interno del programma.

Un modo per ottenere questo è tramite *dichiarazioni di costante*.

In C++ la dichiarazione di costante con nome ha la seguente foma sintattica di base:

```
const t c = e;
```

dove: `t`: tipo qualsiasi
`c`: identificatore (= nome della costante)
`e`: espressione costante di tipo `t` (= valore della costante)

Il suo significato è: dichiara una costante di tipo `t`, di nome `c`, con valore (il risultato della valutazione dell'espressione) `e`.

Esempi:

```
const int num_elem = 10;       // costante di nome
                               // num_elem e valore 10
const float pi_greco = 3.14;   // costante di nome
                               // pi_greco e valore 3.14
```

Dunque la forma sintattica della dichiarazione di costante è molto simile a quella della dichiarazione di variabile (con inizializzazione). La sua semantica, però, è profondamente diversa: una dichiarazione di costante infatti introduce soltanto un'associazione tra un nome ed un valore, senza richiedere la presenza di una locazione di memoria cui legare il nome e in cui memorizzare il valore corrente, come invece avviene nel caso delle variabili.

Una dichiarazione di costante può essere inserita ovunque in un programma (anche nella parte delle dichiarazioni globali, come spesso avviene).
La visibilità dei nomi di costanti è determinata in base alle normali regole di “scope” previste per le variabili e per tutti gli altri nomi. In particolare una costante potrà essere usata dalla sua dichiarazione in avanti (non prima) e non sarà visibile in un blocco più esterno a quello in cui dichiarata.

Una volta dichiarato, il nome di una costante può essere usato ovunque possa apparire un valore costante.

Esempio:

```
const int num_elem = 10;
int main() {
    ...
    int A[num_elem];
    if (i > num_elem) ...;
```

ma

```
num_elem = 12;
```

viene segnalato come errore in quanto una costante non può apparire nella parte sx di uno stmt di assegnamento.

Vantaggi ad usare costanti con nome al posto di valori costanti:

- maggior leggibilità del programma
- maggior modificabilità del programma: una eventuale modifica del valore della costante circoscritta alla sola dichiarazione della costante (l'uso resta invariato).

Vantaggi ad usare costanti con nome al posto di variabili (in pratica, omettere la specifica `const` nella dichiarazione di costante):

- maggior affidabilità del programma: evita modifiche non volute
- possibile maggior efficienza del codice generato: il compilatore può evitare l'allocazione di memoria per contenere le costanti (usando ad es. istruzioni con operando immediato), oppure le può allocare nella parte della memoria statica (invece che sullo stack).

N.B. Il C++ ammette che nella dichiarazione

```
const t c = e;
```

l'espressione `e` possa essere un'espressione qualsiasi, anche contenente variabili (il cui valore può essere determinato in generale solo a run-time).

A rigore, in questo caso, la dichiarazione `const` non introduce una costante con nome, quanto piuttosto una *variabile di sola lettura* (o *variabile costante).* Si tratta a tutti gli effetti di una variabile, ma il compilatore controlla che non possa essere modificata (e quindi, ad esempio, non può apparire nella parte destra di un assegnamento). Può essere usata come una costante con nome, ma la sua dichiarazione, nel caso in cui l'espressione `e` contenga variabili, non può apparire nella parte delle dichiarazioni globali di un programma.

N.B. In C++ è possibile associare un nome simbolico ad un valore anche tramite la direttiva di preprocessore `#define`.
Ad esempio:

```
#define num_elem 10
#define pi_greco 3.14
```

Concettualmente molto diverso dalla dichiarazione `const`, anche se l'uso può essere simile. Una direttiva

```
#define s1 s2
```

viene trattata in fase di preprocessing e comporta una semplice sostituzione di stringhe nel programma sorgente: tutte le occorrenze della stringa `s1` vengono rimpiazzate dalla stringa `s2`. Dunque è una trasformazione da programma sorgente a programma sorgente. `s1` e `s2` non sono identificatori, ma stringhe qualsiasi (senza alcun tipo associato). La visibilità dei nomi introdotti non è determinata dalle regole di “scope”, ma si applica all'intero programma, dalla direttiva in poi.