Fondamenti di Programmazione A

Appunti per le lezioni – *Gianfranco Rossi*

Struttura (statica) di un programma C++

Finora visto soltanto programmi costituiti dal programma principale (`main`), più eventuali dichiarazioni globali di libreria, incorporate nel programma tramite direttive `#include`.

In generale, un programma può avere una struttura più complessa, in cui sono presenti più sottoprogrammi e altre dichiarazioni globali definite dal programmatore. La struttura tipica di un programma C++ è la seguente:

```
                                costanti
Dichiarazioni (globali)         tipi (in part., struct, class)
                                variabili (in part., strutture dati)
                                prototipi funzioni
tipo1 f1(...) {
     Dichiarazioni (locali a f1)
     Statement di f1
}
...
tipom fm(...) {
     Dichiarazioni (locali a fm)
     Statement di fm
}
int main() {
     Dichiarazioni (locali al main)
     Statement del main
}
tipom+1 fm+1(...) {
     Dichiarazioni (locali a fm+1)
     Statement di fm+1
}
...
tipon fn(...) {
     Dichiarazioni (locali a fn)
     Statement di fn
}
```

dove *Dichiarazioni* e *Statement* sono insiemi, anche vuoti, di dichiarazioni e di statement, f_1,..., f_n sono nomi di funzioni e $tipo_1$,..., $tipo_n$ sono i tipi delle funzioni f_1,..., f_n ($m,n \geq 0$)

f_1,..., f_n sono i *sottoprogrammi*, mentre `main` è il *programma principale* (di fatto una funzione come le altre, ma con nome speciale, che l'ambiente d'esecuzione riconosce e che usa come “entry point” nel programma, cui passare il controllo quando il programma deve essere eseguito).

Si noti che nel caso $n = 0$ è presente il solo programma principale.

Dichiarazioni globali e locali

Si noti che in base alle “regole di scope” del C++, il **campo d'azione** delle **dichiarazioni globali** di un programma si estende a tutte le funzioni del programma (in altri termini, un nome introdotto da una dichiarazione globale è utilizzabile in una qualsiasi delle funzioni f_1,..., f_n che compongono il programma). Ad esempio:

```
const int max = 10;
struct data
        {...};       // qui max è visibile
int f(...)
        {...}        // qui max è visibile
int main()
        {...}        // qui max è visibile
int g(...) {
        {...}        // qui max è visibile
```

Unica eccezione è costituita dal caso in cui lo stesso nome `id` introdotto in una dichiarazione globale sia introdotto anche in una dichiarazione locale in una delle funzioni f_1,..., f_n; in questo caso la dichiarazione locale “nasconde” temporaneamente—ovvero per tutto il corpo della funzione in cui appare—la dichiarazione globale (in altri termini, un riferimento al nome `id` all'interno della funzione si riferisce alla dichiarazione locale e non a quella globale). Ad es.:

```
const int max = 10;
int f(...)
        {...}        // qui max "globale" è visibile
int main()
        {int max;
         ...         // qui max "globale" non è visibile
        }            // è visibile invece max "locale"
int g(...) {
        {...}        // qui max "globale" è visibile
```

Si noti anche che, in base alle stesse "regole di scope", un nome introdotto (soltanto) in una funzione è locale a questa funzione, e cioè può essere utilizzato soltanto al suo interno.

```
...
int f(...)
        {...}        // qui x non è visibile
int main()
        {int x;
         ...}        // qui x è visibile
int g(...) {
        {...}        // qui x non è visibile
```

Infine si noti che le dichiarazioni delle funzioni sono di fatto delle dichiarazioni globali e quindi i nomi delle funzioni sono visibili in tutto il programma (valendo, comunque, sempre la regola che un nome è visibile dalla sua dichiarazione in poi, e non prima). Ad esempio:

```
int f(...)
        {...}        // f visibile, g non visibile
int main()
        {...}        // f visibile, g non visibile
int g(...) {
        {...}        // f e g visibili
```

Ordine delle dichiarazioni di funzione (prototipi)

Per poter utilizzare una funzione senza dover fornire prima la sua definizione completa, in C++ è possibile dare la dichiarazione della sola testata della funzione (il ***prototipo*** della funzione), separatamente dal suo corpo.

Una volta fornita la dichiarazione della testata (anche senza corpo), il nome della funzione diventa visibile e quindi utilizzabile all'interno delle altre funzioni del programma. La definizione completa della funzione (testata + corpo) potrà essere fornita a parte (eventualmente compilata separatamente), e potrà essere posta anche dopo le funzioni che la utilizzano. Ad esempio:

```
int f(int x);    // prototipo della funzione f
int g(char c);   // prototipo della funzione g
int main()
         {...}   // f e g visibili qui
int f(int x);    // definizione completa di f
         {...}
int g(char c) {  // definizione completa di g
         {...}
```

n.b. Dopo la dichiarazione del prototipo della funzione va inserito un punto e virgola.

La possibilità di specificare prototipi rende di fatto non rigido l'ordine in cui vengono fornite le dichiarazioni delle funzioni (e del main).

Le direttive `#include`, elaborate dal preprocessore, normalmente provvedono ad aggiungere al programma sorgente, sviluppato dal programmatore, tutti i *prototipi* delle funzioni di libreria che si prevede vengano utilizzate nel programma.
La definizione completa di queste funzioni (codice sorgente), invece, non viene aggiunta al programma. Le funzioni di libreria sono invece *compilate separatamente* e il codice *oggetto* ottenuto viene “cucito” a quello del programma utente nella successiva fase di “linking”.