Corso: Fondamenti, Linguaggi e Traduttori

Paola Giannini

Symbol Table

Symbol Table

- Struttura centrale per **tutte le fasi della compilazione**.
- Contiene informazioni sui **nomi** usati nel programma:
  - variabili
  - etichette
  - classi
  - funzioni
  - metodi
  - struct
  - .....

.

Un esempio molto semplice di costruzione per C-- (linguaggio a blocchi)

Assumiamo di partire da un AST e facciamo una visita che ha 2 scopi

1. Processare le dichiarazioni dei nomi
2. Connettere gli usi dei nomi alle lori dichiarazioni (La connessione avviene attraverso l'uso della symbol table.)

Se non è possibile fare la connessione vuol dire che il nome non è definito, altrimenti le visite successive possono ottenere le informazioni accedendo direttamente all'entry della symbol table.

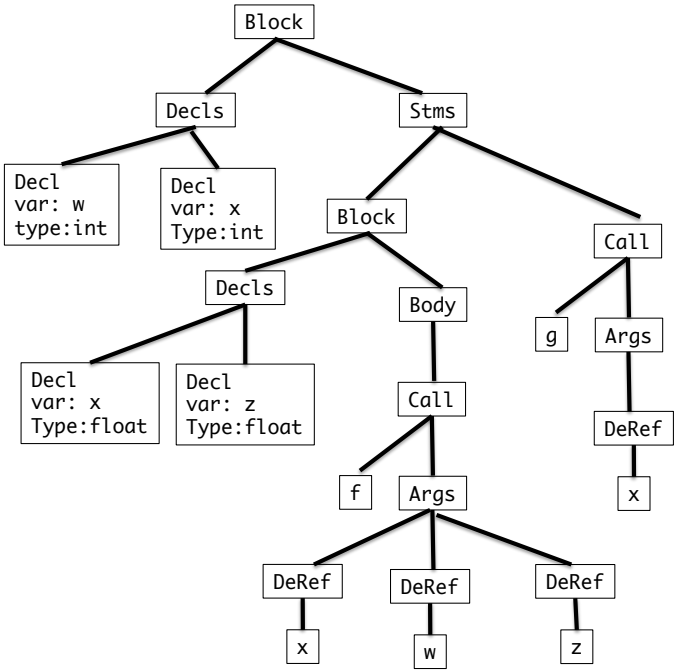
Un programma C--

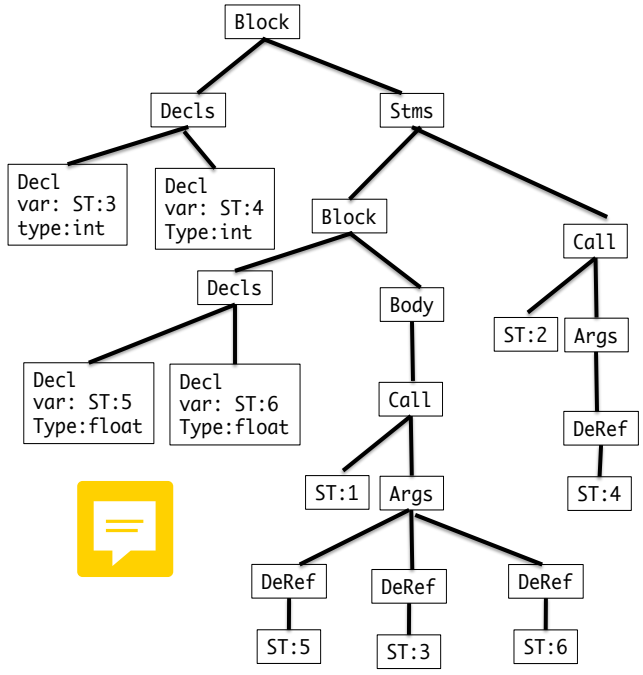
```
include f(float, float, float)
include g(int)

{
  int w,x
    {
       float x,z
       f(x,w,z)
     }
     g(x)
}
```

Ci sono 2 blocchi annidati. Notate che i nomi `float` e `int` non hanno entry nella symbol table perchè lo scanner li ha riconosciuti come parole chiave!

L'AST del programma prima e dopo la visita e la sua symbol table





Numero Simbolo	Nome Simbolo	Attributi
1	f	void func(float,float,float)
2	g	void func(int)
3	w	int
4	x	int
5	x	float
6	z	float

Lo **scope** di una variabile e i blocchi

- Lo **scope di una dichiarazione** è la zona del codice in cui la variabile dichiarata può essere riferita.
- Un nome non può essere dichiarato piú di una volta in un dato scope.
- I **blocchi** delimitano lo scope delle dichiarazioni.
- Nei linguaggi con **scoping statico** i riferimenti sono legati alla dichiarazione nel blocco più interno che li contiene.
- I linguaggi hanno anche uno **scope globale** che contiene dichiarazioni di nomi accessibili da tutte le unità di compilazione
- C permette di avere buchi nello scope delle variabili, mentre Java non lo permette (come la variabile x dell'esempio).
- In Java comunque una variabile locale o parametro di un metodo può avere lo stesso nome di una variabile/campo della classe

UNIVERSITÀ DEL PIEMONTE ORIENTALE

Interfaccia della Symbol Table

```
openScope() // apre un nuovo scope in cui inserire i nomi

closeScope() // chiude lo scope piu' recente (quindi i nomi sono interpretati
                    // nel precedente scope)

enter(nome,infNome) // inserisce il nome nello scope corrente  con
                    // l'informazione sulla  dichiarazione

lookup(nome) // ritorna l'informazione sul nome riferita alla dichiarazione corrente
             // del nome, null se il nome non e' dichiarato

dichLocale(nome) // true se il nome e' dichiarato nello scope corrente false
                 // altrimenti
```

Struttura ad albero dei blocchi, Esempio

```
int value=10;
void pro_one()
   {
   int one_1;
   int one_2;
      {              \
      int one_3;      |_  inner scope 1
      int one_4;      |
      }              /
   int one_5;
      {              \
      int one_6;      |_  inner scope 2
      int one_7;      |
         XX           |
      }              /
   }
void pro_two()
   {
   int two_1;
   int two_2;

      {              \
      int two_3;      |_  inner scope 3
      int two_4;      |
          YY          |
      }              /

   int two_5;
   }
```

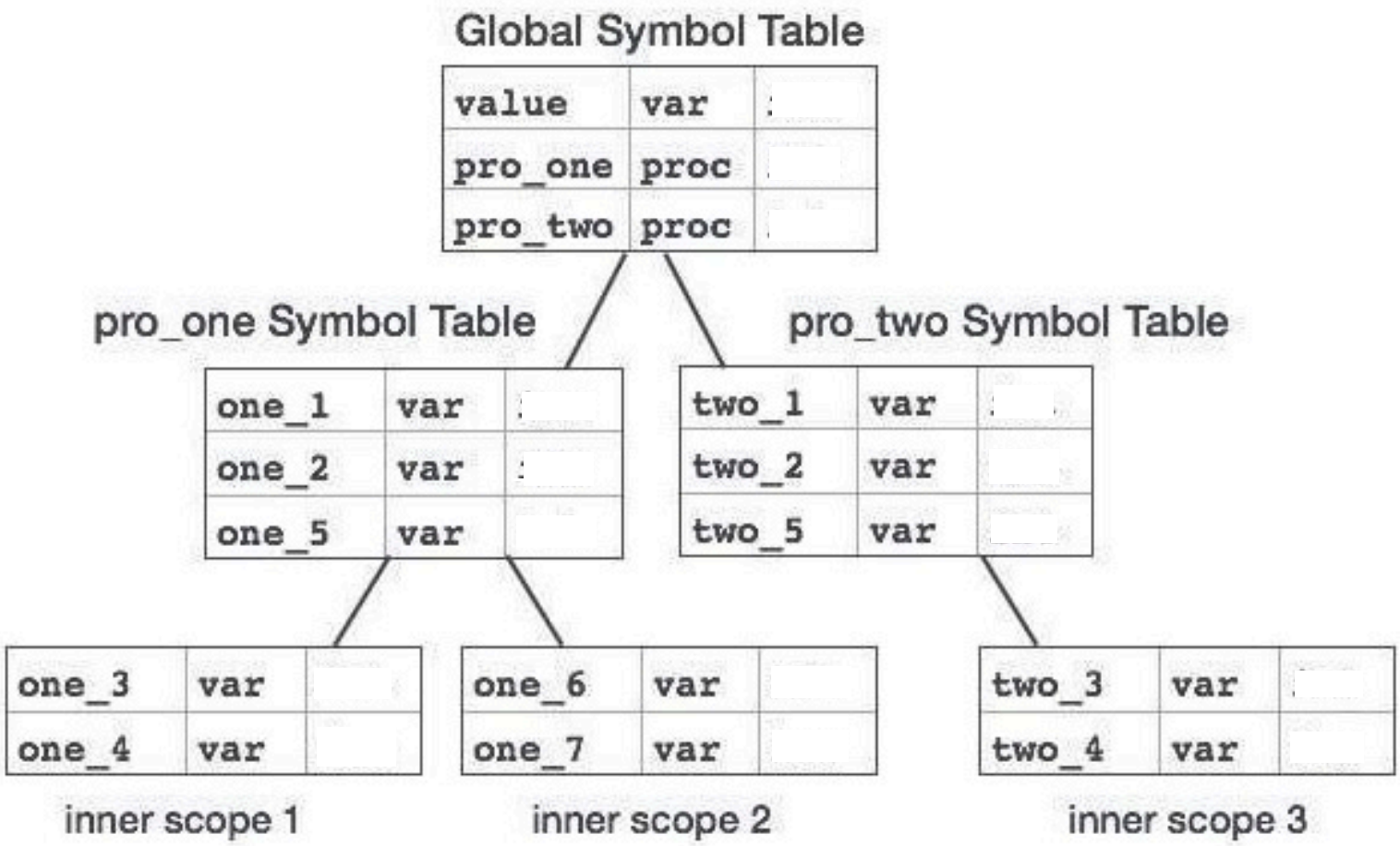
Uso dello scope

- Ogni simbolo occorre nel contesto di un certo numero di scope. Esempio XX e YY
- Quando analizziamo il codice **lo scope corrente** è **quello più interno**.
- Gli **scope attivi** sono quello corrente più **gli scope aperti**.

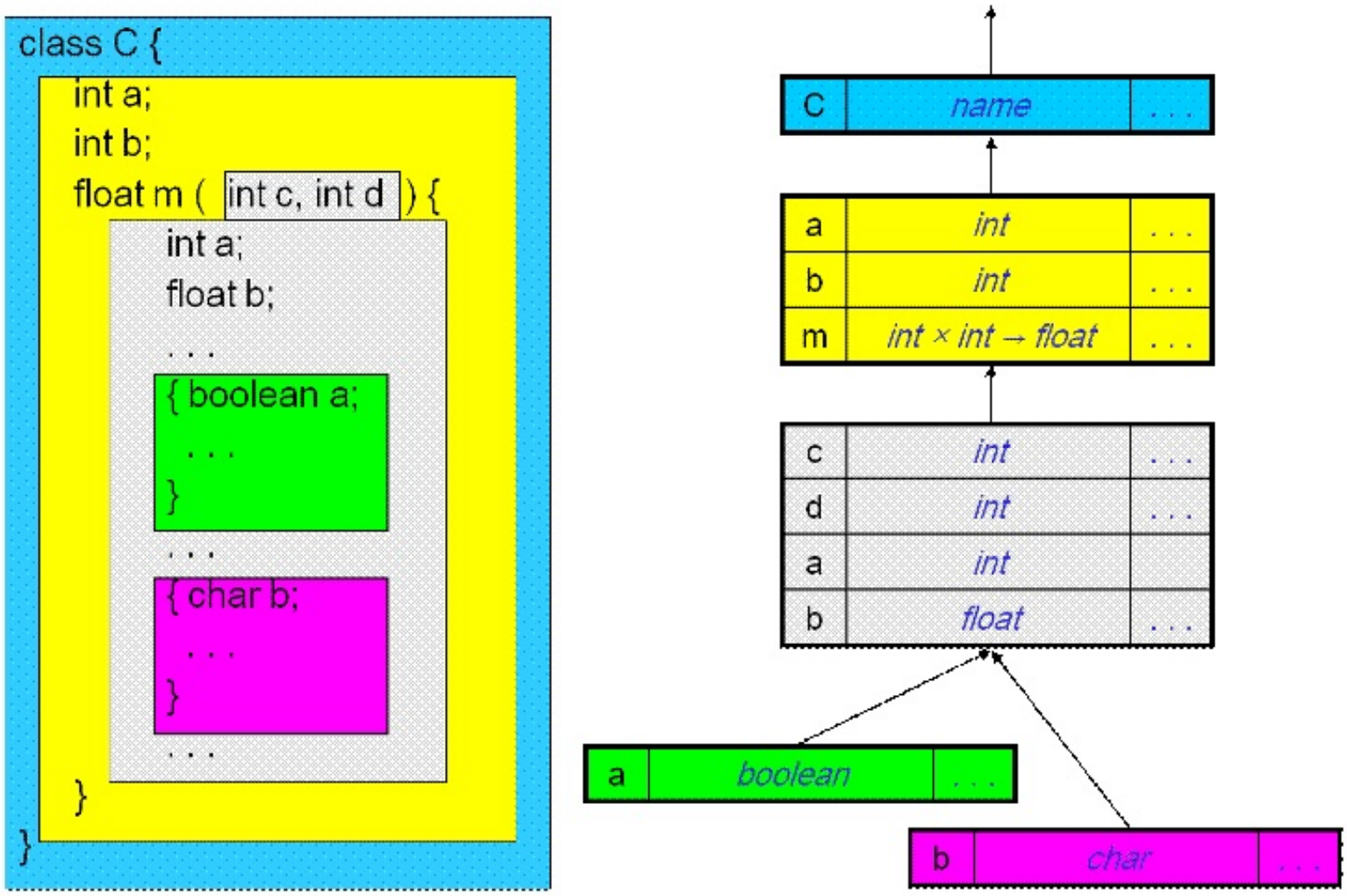
La symbol table deve riprodurre (logicamente) questa struttura, cioè

- Rendere accessibili i nomi definiti nello scope corrente e in tutti quelli aperti
- Aggiungere le nuove definizioni allo scope corrente.
- Se un nome è dichiarato in più di uno scope risolverlo accedendo a quello più interno

Struttura ad albero degli scope



Scope nel linguaggi a oggetti



Possibili implementazioni delle tabelle (che sia singola o multiple)

Una symbol table è una struttura dati dizionario (associazione chiave/valore)quindi possiamo usare le struttura dati usate per i dizionari.

Possibili implementazioni delle tabelle (che sia singola o multiple)

Una symbol table è una struttura dati dizionario (associazione chiave/valore)quindi possiamo usare le struttura dati usate per i dizionari.
L'operazione più frequente è la ricerca. Una volta inseriti i simboli NON vengono mai cancellati!

Possibili implementazioni delle tabelle (che sia singola o multiple)

Una symbol table è una struttura dati dizionario (associazione chiave/valore)quindi possiamo usare le struttura dati usate per i dizionari.
L'operazione più frequente è la ricerca. Una volta inseriti i simboli NON vengono mai cancellati!

1. Lista/Array di record contenenti per ogni nome la lista delle sue dichiarazioni annotate con un identificatore che dice a quale scope fa riferimento la dichiarazione. Ricerca (O(??))

Possibili implementazioni delle tabelle (che sia singola o multiple)

Una symbol table è una struttura dati dizionario (associazione chiave/valore)quindi possiamo usare le struttura dati usate per i dizionari.
L'operazione più frequente è la ricerca. Una volta inseriti i simboli NON vengono mai cancellati!

1. Lista/Array di record contenenti per ogni nome la lista delle sue dichiarazioni annotate con un identificatore che dice a quale scope fa riferimento la dichiarazione. Ricerca (O(??))
2. Array Ordinata (ricerca più efficiente), e se volessi avere una ricerca efficiente, ma un struttura dinamica?

Possibili implementazioni delle tabelle (che sia singola o multiple)

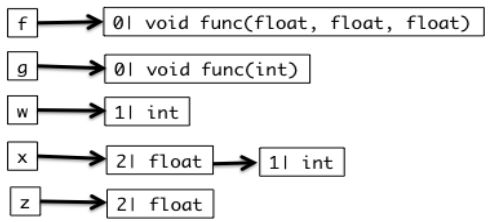
Una symbol table è una struttura dati dizionario (associazione chiave/valore)quindi possiamo usare le struttura dati usate per i dizionari.
L'operazione più frequente è la ricerca. Una volta inseriti i simboli NON vengono mai cancellati!

1. Lista/Array di record contenenti per ogni nome la lista delle sue dichiarazioni annotate con un identificatore che dice a quale scope fa riferimento la dichiarazione. Ricerca (O(??))
2. Array Ordinata (ricerca più efficiente), e se volessi avere una ricerca efficiente, ma un struttura dinamica?
   Albero Binario di ricerca. Ricerca (O(??)) (In entrambi i casi è necessario conoscere quali sono gli scope attivi per cercare il riferimento corretto alla dichiarazione per la variabile.)

Possibili implementazioni delle tabelle (che sia singola o multiple)

Una symbol table è una struttura dati dizionario (associazione chiave/valore)quindi possiamo usare le struttura dati usate per i dizionari.
L'operazione più frequente è la ricerca. Una volta inseriti i simboli NON vengono mai cancellati!

1. Lista/Array di record contenenti per ogni nome la lista delle sue dichiarazioni annotate con un identificatore che dice a quale scope fa riferimento la dichiarazione. Ricerca (O(??))
2. Array Ordinata (ricerca più efficiente), e se volessi avere una ricerca efficiente, ma un struttura dinamica?
   Albero Binario di ricerca. Ricerca (O(??)) (In entrambi i casi è necessario conoscere quali sono gli scope attivi per cercare il riferimento corretto alla dichiarazione per la variabile.)
3. Tabella hash. Ricerca (O(??))



Caratteristiche avanzate dei linguaggi

1. Le strutture del C (hanno associati nomi e struttura ad albero!)
2. Overloading, lo stesso nome deve essere associato a diverse liste di parametri (uso di una lista di definizioni per il nome).
3. Visibilità di Java: **public**, ecc.. Le clausole **import** servono al compilatore per inizializzare la symbol table per risolvere riferimenti a classi abbreviati (es: `java.lang.String` per `String`)
4. In C **static** limita la visibilità della definizione alla unità di compilazione