Scoping

- binding: associazione tra un nome e un'entità del linguaggio, introdotti dalle dichiarazioni
- scope: parte del programma in cui un dato binding è attivo
- i linguaggi compilati cercano di risolvere i binding staticamente
- i linguaggi interpretati devono farlo dinamicamente (ovviamente non significa che non è possibile implementare scoping statico)

Le regole di visibilità degli identificatori sono basate su:

- testo del programma (scoping statico), o
- flusso di esecuzione (scoping dinamico)

Nello scoping statico, un identificatore viene cercato, in ordine, tra:

- le associazioni locali al blocco
- quelle dei blocchi più esterni, dal più vicino al più lontano
- l'ambiente predefinito del linguaggio

A runtime,

- scoping statico: l'ambiente non locale di una funzione è quello esistente al momento in cui viene valutata l'astrazione
- scoping dinamico: al momento in cui avviene l'applicazione

Controlli statici:

- scoping statico: verifica dell'esistenza dei legami e del tipo degli identificatori
- scoping dinamico: i legami dipendono dal particolare flusso di esecuzione, quindi non si possono eseguire controlli di tipo né rilevare identificatori non legati

Inoltre lo scoping statico permette un'implementazione efficiente (vedi sotto) che non richiede di mantenere a runtime i nomi degli identificatori.

Ambiente

- l'ambiente è l'insieme delle associazioni nome-entità esistente a runtime in uno specifico punto del programma e in uno specifico momento dell'esecuzione
- tipicamente basato su blocchi: regioni testuali che possono contenere dichiarazioni
- le modifiche all'ambiente avvengono all'ingresso e uscita dai blocchi, con politica LIFO (stack): creazione, disattivazione (shadowing) e distruzione di legami
- operazioni: naming, referencing, disattivazione, riattivazione, unnaming

Tipi di ambiente:

- locale: legami del blocco
- non locale: legami ereditati da altri blocchi e chiusure
- globale: associazioni valide in ogni punto del programma

Implementazione efficiente di scoping statico

- nel record di attivazione il puntatore di catena statica (static link) punta all'ambiente (record) in cui devono essere risolti i riferimenti non locali
- questo è determinato in base alla sintassi: posizione in cui si trova il blocco / funzione
- la catena statica dunque permette di ricostruire a runtime la struttura lessicale del programma
- durante la compilazione, i nomi delle variabili locali vengono sostituiti con l'offset a cui si troveranno all'interno del record
- per i riferimenti non locali si calcola la rispettiva distanza statica, ovvero la differenza tra le profondità di utilizzo e dichiarazione nell'AST (non dipende dal flusso di esecuzione)
- la distanza statica determina quante volte bisogna seguire il puntatore di catena statica per trovare il record in cui la variabile è memorizzata
- una volta trovato il record, si accede alla variabile tramite offset
- le variabili non locali diventano quindi distanza + offset durante la compilazione
- nei blocchi il puntatore di catena statica coincide con quello di catena dinamica (control link)
- in C tutti i riferimenti non locali sono allo scope globale o ai blocchi che contengono quello corrente per l'assenza di chiusure/funzioni annidate, quindi non è necessaria la catena statica
- le chiusure hanno un puntatore al record del blocco in cui sono state definite
- restituire una chiusura da una funzione fa sì che il record di attivazione di quella funzione venga mantenuto (flag retain)

Shallow binding

Implementazione alternativa al "deep binding" presentato sopra.

- invece di eseguire una ricerca lungo i record di attivazione, manteniamo tutti i legami (locali e non locali) in un'unica tabella centrale, in modo che i riferimenti siano ridotti ad un singolo offset

- complica la gestione dello shadowing: l'associazione nascosta deve essere salvata in una pila e ripristinata alla disattivazione della nuova