Declaraţii. Compilare separată

30 mai 2006

Variabile globale şi locale

Variabilele pot fi declarate şi *în afara* funcţiilor.
Dacă în declaraţia de variabile nu apar alţi specificatori înainte de tip:

Variabile globale

= o variabilă declarată in afara oricărei funcţii
– are spaţiu de memorie alocat pe întreaga execuţie a programului
– e iniţializată o singură dată (cu valoarea dată explicit în declaraţie, sau implicit cu zero)
– e vizibilă în întreg textul programului începând cu declaraţia ei

Variabile locale (interne)

= o variabilă declarată în interiorul unui bloc (inclusiv de funcţie)
– există doar atât timp cât programul execută blocul respectiv
– sunt iniţializate cu valoarea dată la orice intrare în blocul respectiv (sau au o valoare nedefinită dacă declaraţia nu specifică iniţializare)
– sunt vizibile doar în interiorul blocului respectiv

Domeniul de vizibilitate al identificatorilor

Pt. orice identificator, compilatorul trebuie să-i decidă semnificaţia
Identificatorii obişnuiţi: variabile, tipuri, funcţii, constante enumerare au un *spaţiu de nume* comun (NU: variabilă şi funcţie cu acelaşi nume)

Q1: Un identificator poate fi folosit într-un punct de program ?
R: *Domeniul de vizibilitate* (al unei declaraţii / al unui identificator)
– domeniu de vizibilitate la nivel de *fişier* (*file scope*)
pentru identificatori declaraţi în afara oricărui bloc (oricărei funcţii)
din punctul de declaraţie până la sfârşitul fişierului compilat
– domeniu de vizibilitate la nivel de *bloc* (*block scope*)
pentru identificatori declaraţi într-un bloc { } (corp de funcţie, instrucţiune compusă) şi pentru parametrii unei funcţii
din punctul de declaraţie până la acolada } care închide blocul

Un identificator poate fi *redeclarat* într-un bloc interior şi îsi recapătă vechea semnificaţie când blocul ia sfârşit.

Legătura dintre identificatori (linkage)

Q2: Două declarații ale unui identificator se referă la aceeași entitate?
R: Tipul de legătură (*linkage*) al unui identificator (obiect/funcție)

– *extern*: toate declarațiile identificatorului din toate fișierele care compun un program se referă la același obiect sau funcție
  pentru declarațiile *la nivel de fișier* fără specificator de memorare sau declarația cu specificatorul `extern` a unui identificator care nu a fost deja declarat cu tipul de legătură *intern*

– *intern*: toate declarațiile identificatorului din fișierul curent se referă la același obiect sau funcție; nu se propagă în exteriorul fișierului
  pt. declarațiile *la nivel de fișier* cu specificatorul de memorare `static`

– *fără legături* (no linkage): fiecare declarație denotă o entitate unică
  pentru declarațiile *la nivel de bloc* fără specificatorul `extern`

Durata de memorare a obiectelor

Q3: Ce timp de viață/durată de memorare are un obiect în program?
R: 3 feluri diferite: *static*, *automatic* și *alocat* (discutat ulterior)
Pe întreaga durată de viață, un obiect are o *adresă constantă*
și își *păstrează ultima valoare* memorată.

Durată de memorare *statică*:
pentru obiecte declarate cu tipul de legătură *extern* sau *intern*,
sau declarate cu specificatorul de memorare `static`
– timp de viață: *întreaga execuție* a programului.
– obiectul e *inițializat o singură dată*, înainte de lansarea în execuție.

Durată de memorare *automată*: pentru obiecte fără legătură
– timp de viață: de la intrarea în blocul asociat până la încheierea sa
– la fiecare apel recursiv, se crează o nouă instanță a obiectului
– *valoarea inițială: nedeterminată*;
– o eventuală inițializare în declarație e repetată de câte ori e atinsă

Structura programelor din mai multe fișiere

– fiecare fișier poate fi compilat separat în format obiect
– apoi fișierele sunt legate (linkeditate) pentru a crea executabilul
– orice identificator trebuie declarat în fiecare fișier sursă unde e folosit
– o variabilă va fi *definită* (evtl. cu inițializare) într-un singur fișier, și *declarată* (evtl. cu specificatorul `extern`) în celelalte

Realizarea propriilor biblioteci (ex. tipuri de date cu functiile lor)
– într-un fișier `.h` se *declară* tipurile și funcțiile necesare
– într-un fișier `.c` se *definesc* (implementează) funcțiile
– programul care le folosește va include fișierul `.h` și va fi compilat cu fișierul `.c` (linkeditat cu fișierul obiect rezultat din el)