

Concepte fundamentale ale limbajelor de programare

Atributele entitatilor de program. Legarea dinamica a atributelor

Curs 05

conf. dr. ing. Ciprian-Bogdan Chirila

Universitatea Politehnica Timisoara
Departamentul de Calculatoare si Tehnologia Informatiei

March 19, 2023



Cuprins

- Domeniul variabilelor
- Durata de viața a variabilelor
- Alocarea memoriei
- Valorile variabilelor
- Tipurile variabilelor



Cuprins

- 1 Atribute de program
- 2 Legarea atributelor
- 3 Momentul legarii atributelor
 - Legarea la definirea limbajului
 - Legarea la compilare
 - Legarea la momentul executiei
- 4 Variabile
 - Domeniul variabilei
 - Legarea statica
 - Legarea dinamica
 - Legare dinamica la cerere
 - Durata de viata a variabilei
 - Alocarea statica
 - Alocarea dinamica
 - Valorile variabilei
 - Tipul variabilei



Atribute de program

- Un limbaj de programare opereaza cu mai multe entitați:
 - variabile
 - constante
 - subprograme, functii, metode
 - tipuri, clase, interfete
 - instrucțiuni
- Entitățile:
 - pot avea un nume atunci când le este asociat un identificator
 - sau pot fi anonime
 - de exemplu când obiectele sunt referite prin pointeri
- Numele este doar unul din posibilele atribute ale unei entitați



Atribute de program

- În limbajele de programare imperative:
 - o variabilă are
 - nume
 - tip
 - adresă de memorie
 - un subprogram are:
 - nume
 - parametri formali
 - secvență de acțiuni asociate
 - o instrucțiune are:
 - acțiuni implicite



Cuprins

- 1 Atribute de program
- 2 Legarea atributelor
- 3 Momentul legarii atributelor
 - Legarea la definirea limbajului
 - Legarea la compilare
 - Legarea la momentul executiei
- 4 Variabile
 - Domeniul variabilei
 - Legarea statica
 - Legarea dinamica
 - Legare dinamica la cerere
 - Durata de viata a variabilei
 - Alocarea statica
 - Alocarea dinamica
 - Valorile variabilei
 - Tipul variabilei



Legarea atributelor

- asocierea dintre o entitate și atributele acesteia se numește **legare**
- limbajele de programare se diferențiază prin modul în care atributele sunt legate la entități
- legarea atributelor poate fi
 - implicită
 - explicită



Exemple

- Pentru variabilele din Fortran
 - legarea numelui se face
 - în primul loc unde variabila este folosită
 - legarea tipului
 - depinde de numele variabilei
 - I,J,K,L,M,N - intregi
 - alte nume - reale
 - sau prin declarare explicită
- În ML, Python:
 - numele, tipul și valoarea variabilei sunt legate în momentul atribuirii
 - `a = 1`
 - `a = 'Hello World'`
 - `a = False`



Exemple

- redundanța utilă se obtine
 - prin legarea explicită a atributelor
 - declarațiile din Pascal, C, C++, C Sharp, Java



Cuprins

- 1 Atribute de program
- 2 Legarea atributelor
- 3 Momentul legării atributelor**
 - Legarea la definirea limbajului
 - Legarea la compilare
 - Legarea la momentul execuției
- 4 Variabile
 - Domeniul variabilei
 - Legarea statică
 - Legarea dinamică
 - Legare dinamică la cerere
 - Durata de viață a variabilei
 - Alocarea statică
 - Alocarea dinamică
 - Valorile variabilei
 - Tipul variabilei



Momentul legării atributelor

- Legarea la definirea limbajului
- Legarea la compilare
- Legarea la execuție



Legarea la definirea limbajului

- În C, C++, Java, C Sharp:
 - identificatorii speciali: `char`, `int`, `float`, etc.
 - sunt asociați cu mulțimile de valori corespunzătoare
- În Pascal:
 - Constantele: `true`, `false`, `Maxint`
 - Tipurile: `integer`, `real`, `char`
 - Funcțiile: `abs`, `trunc`, `chr`, `ord`
- În Java, C Sharp:
 - `null` este legat prin definiție
 - `null` este asociat pointerului `void`



Legarea la compilare

- tipuri de variabile
 - `var i:integer;` (Pascal)
 - `int i;` (C, C++, Java, C Sharp)
- valori constante
 - `const pi=3.14159;` (Pascal)
- tipuri și valori
 - `final double pi=3.14159;` (Java)
 - `public const double Pi = 3.14159;` (C Sharp)



Legarea la momentul execuției

- atribuirea de valori unei variabile
- Legarea statică
 - înainte de execuție
 - în definiția limbajului
 - la momentul compilării
 - nu poate fi schimbată ulterior
- Legarea dinamică
 - în momentul execuției
 - poate fi schimbată ulterior



Cuprins

- 1 Atribute de program
- 2 Legarea atributelor
- 3 Momentul legarii atributelor
 - Legarea la definirea limbajului
 - Legarea la compilare
 - Legarea la momentul executiei
- 4 Variabile
 - Domeniul variabilei
 - Legarea statica
 - Legarea dinamica
 - Legare dinamica la cerere
 - Durata de viata a variabilei
 - Alocarea statica
 - Alocarea dinamica
 - Valorile variabilei
 - Tipul variabilei



Variabile

- domeniu
- durată de viață
- valoare
- tip
- nume
 - dacă nu este anonimă și referită prin pointer



Domeniul variabilei

- zona de program unde variabila este cunoscută si folositoare
- variabila
 - este vizibilă in domeniu
 - este invizibilă in afara domeniului
- conceptul de domeniu este legat de
 - context
 - mediu



Domeniul variabilei

- context
 - toate variabilele cu valori într-un punct al programului
- mediu
 - subdomeniu definit în mod explicit pentru una sau mai multe variabile
 - de exemplu: corpul funcției este mediul pentru variabilele locale și parametri



Legarea statica a domeniului

- reguli clasice pentru limbajele de programare bazate pe blocuri
 - domeniul unei variabile este blocul unde aceasta a fost declarată și toate blocurile sale interne
 - variabila este invizibilă în afara aceluși bloc în care ea a fost definită
- domeniul variabilei
 - este determinat de structura lexicală a programului
 - este determinată static de textul programului
 - nu depinde de dinamica execuției
 - orice referințe de variabilă va fi relaționată de către compilator cu declararea (implicită sau explicită) a acesteia
- astfel rezultă o **legare statică a domeniului**



Exemplu in limbajul Pascal

```
program domain;  
  var x:integer;  
  
procedure f;  
begin  
  write(x) { se refera la variabila globala x }  
end;  
  
procedure f1(x:integer);  
begin  
  f  
end;  
  
begin { program principal }  
  x:=10;  
  f; { se afiseaza 10 }  
  f1(5); { se afiseaza 10 }  
end.
```



Exemplu in limbajul C

```
#include<stdio.h>
int x;
void f()
{
    printf("%d\n",x); /* se refera la variabila globala x */
}

void f1(int x)
{
    f();
}

int main() /* program principal */
{
    x=10;
    f(); /* se afiseaza 10 */
    f1(5); /* se afiseaza 10 */
}
```



Comentarii la exemplu

- procedura/functia f referă variabila x care este globală
- nu conteaza de unde este aceasta apelată
- în legarea statică a domeniului
 - declaratia validă este cautată în mediul în care este referită
 - daca lipsește atunci este cautată în mediile externe
- este cazul pentru
 - Pascal, Ada, C, C++, C Sharp, Java, Fortran, Modula 2



Legarea dinamică a domeniului

- domeniul variabilei
 - este determinat în timpul execuției programelor
 - depinde de **calea** de execuție
- variabila se leagă la o declarație care
 - este vizibilă în textul programului
 - este determinată în timpul execuției
- o declarație de variabilă devine disponibilă cand
 - este intalnită pe o cale de execuție
 - leagă de ea toate referințele viitoare la numele acelei variabile
 - până când apare o nouă declarație cu același nume



Exemplu in limbajul Lisp

```
(setq x 10)
```

```
(defun f()  
  (print x))
```

; poate referi variabila globala x sau parametrul x

```
(defun f1(x)  
  (f))
```

- print x se poate referi la variabila globală x sau la parametrul x
- legarea se face în momentul execuției



Comentarii la exemplu

- (f)
 - se afișează 10
 - valoarea variabilei globale x
- (f1 5)
 - se afișează 5
 - valoarea parametrului x
- legarea dinamica a domeniului
 - afectează lizibilitatea programelor
 - facilitează implementarea limbajelor interpretate
 - este prezentă în limbajele funcționale de ex. Lisp sau APL



Legarea statică a domeniului în Lisp

- prezentă în versiunile noi de Lisp
 - Scheme
 - Common Lisp

- Exemplu:

```
>(defun f1(x) (f))
```

```
>(defun f() x)
```

```
>(f1 5)
```

```
*** - EVAL: variable X has no value
```



Comentarii

- valoarea lui x din funcția f este căutată
 - static în mediul lui f
 - apoi în manieră globală
- dacă nu este definit acolo
- se generează o eroare



Legarea dinamică a domeniului la cererea programatorului

- în Common-Lisp
- variabile locale speciale

```
>(defun f1(x)
  (declare (special x))
  (f))
```

```
>(defun f()
  x)
```

```
>(f1 5)
5
```



Legarea dinamica a domeniului la cererea programatorului

- variabile definite global

```
>(defvar x)
```

```
>(defun f1(x)  
(f))
```

```
>(defun f(x)  
x)
```

```
>(f1 5)  
5
```



Durata de viața al variabilei

- intervalul de timp în care o zonă de memorie este asociată cu variabila
- asocierea unei zone de memorie unei variabile se numește **alocare**



Alocarea statica

- înainte de execuție
- o anumită zonă de memorie decisă la compilare
- va rămâne asociată variabilei pe tot parcursul execuției programului



Alocarea dinamica

- alocarea este făcuta în timpul execuției programului
- zona de memorie poate fi eliberată ulterior
- poate fi refolosita de alte variabile



Alocarea dinamica

- automată
 - fără cerere din partea programatorului
- la cerere
 - prin cerere din partea programatorului
 - cu instrucțiuni de tipul: `malloc()`, `calloc()`, `realloc()`, `new`



Alocarea memoriei

- nu este specifică limbajului de programare
- depinde de decizia implementatorului



Exemple de alocare de memorie

- Fortran si Cobol
 - alocarea memoriei variabilelor se face static în majoritatea implementărilor
 - pot fi echipate de asemenea cu alocarea dinamica a memoriei
- Pascal, C, C++, C Sharp, Java
 - pentru variabilele declarate local se folosește alocarea dinamică
 - la fel ca la un apel de funcție toate variabilele locale sunt alocate pe stivă
 - dupa apel stiva este curățată
 - alocarea memoriei este bazată pe organizarea de stivă



Alocare definită de programator

- În limbajul de programare C
 - în interiorul funcției
 - în mod implicit este dinamică
 - poate fi statică dacă este folosit cuvântul cheie `static`
 - în afara funcțiilor
 - în mod implicit este statică
- Lisp, Prolog, Python, JavaScript
 - alocarea și eliberarea memoriei
 - nu sunt bazate pe modelul de stivă
 - obiectele pot fi create și distruse la momente arbitrare la rulare
 - sunt limbaje dinamice



Valorile variabilei

- valoarea este legată dinamic
 - atribuirea schimbă valoarea variabilei
- valoarea poate fi legată static
 - în cazul constantelor
 - valoarea nu poate fi modificată în timpul vieții acestora



Momentul legarii

- la compilare
 - Constantele în Pascal
 - constante sau expresiile formate din constante în Ada
 - constante definite cu directiva `#define` în C
 - la compilare constantele legate se numesc **constante manifest**
- la executie
 - expresia de constante poate conține variabile și operatori
 - C, Ada, Algol
 - `const int k= 3*i+j;`
 - `k: constant integer:=3*i+j;`



Tipul variabilei

- Determină
 - valoarea pe care o variabilă o poate avea
 - setul de operații ce poate crea și modifica aceste valori
- legarea statică
 - la compilare
 - implicită
 - în Fortran tipul este dat de prima literă a identificatorului
 - constantele Pascal `const k=3;`
 - explicită
 - Pascal `var x:integer;`
 - C, C++, C Sharp, Java `int x;`



Tipul variabilei

- Lisp, ML, Python, JavaScript
 - tipul este legat dinamic
 - aceeași variabilă poate avea tipuri de valori asociate diferite
- CAML (ML dialect)

```
# let a=2*2
val a:int=4
# a;;
-:int=4
```



Tipul variabilei

- Lisp

```
defun f(x) (car x))  
(setq y 'a)
```

```
(setq y '(a b))
```

```
(f y)
```



Bibliography

- 1 Brian Kernighan, Dennis Ritchie, C Programming Language, second edition, Prentice Hall, 1978.
- 2 Carlo Ghezzi, Mehdi Jarayeri – Programming Languages, John Wiley, 1987.
- 3 Horia Ciocarlie – Universul limbajelor de programare, editia 2-a, editura Orizonturi Universitare, Timisoara, 2013.

