Rekursiva typer

- T = …T…
- **Listor** L = Unit + (SxL)
- Pascal, implementeras mha pekare
- ML, direkt i språket
- Kardinalitet: oändligheten
- Operationer: Tom? Första elementet? Resten av listan?

Strängar

- **Lista av tecken, egen typ, flexibel array.....**
- **Primitiva eller sammansatta värden?**
- **Vilka operationer?**
  - Primitiv. (ML)
    Strängoperationer inbyggda i språket
  - Array av tecken (Pascal)
    Arrayop. + spec.op. Fixerad längd
  - Lista av tecken. (Miranda, Prolog)
    Listop. + spec.op.

Typsystem

- Gruppera värden
- Effektiv beskrivning av data
- Typkontroll

Statisk kontra dynamisk typning

- Statisk fastställs under kompilering
- Pascal, C statisk
- Dynamisk fastställs under exekvering
- Lisp, Smalltalk dynamisk

Typekvivalens

- Def. är språkberoende
- Strukturekvivalens : innehåller samma delar, samma mängd värden
- Namnekvivalens: Typerna deklarerade på samma ställe. Pascal, Ada

Typkompletthet

The type completness principle

- Första klassens värden: Kan alla saker nedan
  - Vara argument, utvecklas, tilldelas, vara komponenter i sammansatta värden
- Andra klassens: Kan endast vara argument
- *Typkompletthet*: Ingen operation skall vara *godtyckligt* begränsad när det gäller de inblandade typerna

Uttryck

Literaler

Fixa värden: 1 23.90 'e' "hej" ….

Aggregat

Skapar sammansatta värden:

(23,0.0) {y = 1991, m = "Oct", d = 4}

Funktionsanrop

sin (1.0)

⊕E = ⊕(E) prefix

E1 ⊗E2 => ⊗(E1,E2) infix

alternativt ⊗E1 E2

F (AP) F = funktion AP = aktuell parameter, argument

Val (villkor)

if E1 then E2 else E3

case E of

P1 => E1

|....

| Pn => En

Konstant- och variabelaccess

const pi = 3.141;

var r : real

2 * pi * r

MLs variabler (identifierare) som Pascals konstanter

Variabler och Uppdatering

Variabler innehåller ett värde, ett objekt

Inspekteras och uppdateras: tilldelning

Även filer och databaser

Inte samma som matematiska variabler (förutom i funktionella språk)

Lagring

- minnet är en samling celler
- varje cell är antingen allokerad eller deallokerad
- varje allokerad cell har ett innehåll, antingen ett lagringsbart värde eller odefinierat

Ex:
```
var n : integer
n := 0
n := n + 1
```

Sammansatta variabler

(Composite variables)

- Jämför med sammansatta värden
- Kan inspektera delar av variabler, tar upp flera celler

Total och selektiv uppdatering

```
type T = record
m : integer;
n : real
end;
val a, b : T;
a := b      total
a.m := 3    selektiv
```

Array-variabler

- Statisk: indexgränser sätts vid kompilering, gränserna en del av typen
- Dynamisk: indexgränser sätts vid skapandet av variabeln under exekvering
- Flexibel: begränsas aldrig

Lagringsbara värden

(Storables)

- Icke selektivt uppdaterbara
- Pascal: Primitiva värden, mängder, pekare
- ML: Alla värden, förutom arrayer (komponenterna är alltid referenser)

Livslängd

- Skapas, allokeras
- Tas bort, deallokeras
- Tiden däremellan är livslängden för en variabel
- En variabel tar endast upp utrymme under sin livslängd

Statisk minnesallokering

- Minne binds före exekveringen

Dynamisk minnesallokering

- Minne binds under exekveringen
- Det kan ske både explicit (ex new(ptr)) och implicit (ex lokala variabler vid anrop av underprogram)

Lokala variabler, Globala variabler och Aktivering

- Lokala: inom blocket
- Globala: yttersta blocket
- Aktivering: när blocket exekveras

- Livslängden för ett blocks lokala variabler = aktiveringstiden för blocket.
- En lokal variabel behåller inte sitt värde mellan olika aktiveringar. Gäller även rekursiva anrop.
  Undantag: *Statiska* variabler i t.ex. C.

Variablers initialvärde

Vilket värde har en variabel mellan minnesallokeringen och den första tilldelningen?

- **Variabeln kan tilldelas ett värde vid deklarationen (Ada)**
- **Variabeln får det värde som råkar finnas i minnescellen (C)**
- **Variabeln får ett systemdefinierat värde (Simula)**
- **Variabeln får "värdet" *undefined*. Kontroll sker sedan under exekveringen att man inte refererar till en variabel med det värdet (Pascal)**

Dynamiska variabler

(Heap variables)

- Variabeln är på högen inte på stacken, *heap variable*
- Kan skapas och tas bort när som helst
- Är anonym, kan endast kommas åt genom pekare
- Pekare (new) är första klassens värde
- Allokering och deallokering
- Används för dynamiska strukturer

Permanenta variabler (Persistent)

- Livslängd längre än program. Filer, restriktioner

Tillfälliga variabler (Transient)

- Alla variabler som endast existerar under programmets exekvering, dvs lokala och dynamiska variabler

Dangling reference

- Referens till icke levande variabel.
- *Dispose* av en variabel som sedan refereras till
- funktioner och procedurer som första klassens värden

Kommandon

- Programfras för att uppdatera variabler
- Singel entry, singel exit

Skip

- Hoppar över något som borde varit där men som inte behövs

Tilldelning

- Simultan: V1 := V2 := … Vn := E
- Multipel, V1 , V2 , … , Vn := E1, E2,…,En
- Referens
  i ML: (if … then m else n) := 7

Dereferering

n := 1

n := n + 1

(ML) n := !n +1

Proceduranrop

Uppdatera variabler, antigen globala eller AP_i

$P(AP_1,AP_2,...,AP_n)$

AP antingen värde (uttryck) eller ref

Sekvens

C1;C2

Samtidig

C1,C2

- Icke deterministisk n := 7, n := n + 1
- Reellt (effektivt) deterministisk
  n := 7 , m := 6

Villkorskommandon

if E then C1 else C2

- icke deterministisk:

  if E1 the C1
  | E2 then C2
  …
  end if

Case sats

case E of
L1 : C1
…
Ln : Cn
end

Iterativa kommandon

Loopar

Villkorsstyrda (obestämd iteration)

- `while E do C`
- `repeat C until E`
- `repeat C1 while E do C2`

Antalsstyrda (bestämd iteration)

for V := E1 to E2 by E3 do *C*
for V in E do *C*

V av diskret primitiv typ, kallas kontrollvariabel

Algol 60 m fl: **for** V:= E1 **to** E2 **by** E3 **do** *C*

Godtycklig aritmetisk serie

CLU (Miranda): **for** V **in** L **do** *C*

Iteration över listor (och andra datatyper)

Ytterligare frågor:

- Får loopkontrollvariabeln ändras inuti loopen? (ej i **Pascal** eller **Ada**)
- Vilket värde har loopkontrollvariabeln efter loopens slut eller hopp ut ur loopen? (**Ada**: odefinierad)
- Evalueras E2 och E3 en gång för alla (**Pascal**) eller efter varje iteration (**Algol60**)?
- Vad händer om V ej "träffar" E3 precis?

Olika varianter och kombinationer

I Algol68 har man satt ihop 32 loopkonstruktioner i en !
for i from j by k to m while b do..od
En eller flera av dessa fem delar kan utelämnas.
Ada: loop …*body*… end loop.
Man kan hoppa ut ur loopen med exit [when *condition*]

Uttryck med sidoeffekt

Kommando uttryck = Pascal funktioner

Uttrycksorienterade språk

Ingen skillnad på kommandon och uttryck. C, ML med flera