Vad menas med Abstraktion?

- Ordet abstraktion betyder: *Att bortse ifrån, avskilja*, dvs bortse ifrån oväsentliga detaljer
- Att fokusera på det nödvändiga och viktiga

Varför abstraktioner

- Synliggör skillnaden mellan *vad* som ska göras och *hur* det görs genom att anrop av abstraktioner handlar om *vad* och implementationer handlar om *hur*
- Ger stöd för generalitet, flexibilitet och återanvändning

Abstraktioner i programspråk

- Konstruktioner i programspråk kallas *abstraktion* om de stöder möjligheten att abstrahera
- En abstraktion innefattar något slags evaluering
- Bindningar räknas inte med i abstraktionen

När abstraktioner?

- När är det aktuellt med abstraktioner?
  - funktionsabstraktioner
  - procedurabstraktioner
  - abstraktioner för andra syntaktiska klasser

Funktionsabstraktioner

- innefattar ett uttryck som evalueras
- vid anrop resulterar den i ett värde

Procedurabstraktioner

- innefattar ett kommando som evalueras
- vid anrop uppdateras variabler

Abstraktionsprincipen

- Det är möjligt att konstruera abstraktioner över varje syntaktisk klass, om fraser ur den klassen specificerar något slags beräkningar.
- Selektorabstraktion över variabel access
- Generisk abstraktion är en abstraktion över en deklaration

Parametrar

Varför behövs parametrar?

- För att generalisera
- Mer kontrollerat än utan

Formell/Aktuell parameter

- Formell: används inne i en abstraktionskropp för att beteckna ett argument
- Aktuell: Ett uttryck som resulterar i ett argument
- Argument är ett värde som får skickas till en abstraktion.

Parameteröverföring

Hur koppla ihop formell och aktuell parameter:

- Call by value, result, constant, reference, variable, name, procedure, function
- Kopierande
- Definierande

Kopierande

- den aktuella parametern kopieras in i och/eller från den formella parametern.

värde-parametrar (value parameter):

- första klassens värde, formell parameter X := argumentet vid anropet

Kopierande (forts.)

result-parametrar:

- argumentet en variabel, odefinierad vid anrop; argument := X vid uthopp

value/result-parametrar:

- argumentet en variabel, X:= argument vid anrop och argument:= X vid uthopp

Definierande

Den formella parametern X binds direkt till argumentet

- Konstant-parametrar
- Referens-/variabler-parametrar
- Procedur-parametrar
- Funktions-parametrar

Som om abstraktionskroppen var omgiven av ett block där formella parametern X binds till argumentet i en definition

Parametrar och definitioner

- Konstantparametrar motsvarar konstantdefinitioner.
- Variabelparametrar motsvarar variabeldefinitioner (renaming)
- Value-parametrar motsvarar ny-variabel-deklarationer med initiering

Motsvarighetsprincipen

(Correspondence principle)

För varje form av deklaration finns en motsvarande parameter-överföringsmekanism och vice versa

Evalueringsordning

När exakt evalueras den aktuella parametern vid ett anrop?

- Applikativ ordning
- Normalordning
- Lat evaluering

Applikativ ordning

(eager evaluation)

- Aktuell parameter evalueras vid ett tillfälle då alla förekomster av formell parameter ersätts med det evaluerade värdet

Normal ordning

- Aktuell parameter evalueras inte genast, istället ersätts varje förekomst av formell parameter med aktuell parameter

Lat evaluering

- Aktuell parameter evalueras och lagras undan när det först behövs. Värdet tas fram om det behövs senare, dvs när formell parameter påträffas igen.
- Man kan tänka sig fall där aktuell parameter aldrig evalueras

"Church-Rosser property"

- *Om* ett uttryck kan evalueras, så kan det evalueras genom att använda normal-ordningsevaluering på ett konsekvent sätt. Om ett uttryck kan evalueras på olika sätt (ordningar) så leder evalueringen enligt alla ordningar till samma resultat.

Parametrar i olika språk

- Pascal: Value (kopiering), reference (bindning)
- ML: Överföring genom bindning
- Algol60: Value, name
- C: Value
- Ada: in, out, in out