Labb X2: Programspråkstidsresa

Emilie Ruixin Cao & Nils Malmberg
ercao@kth.se & nilsmal@kth.se

7 april 2025

1 COBOL (50/60-tal)

COBOL har hemska indentations och språkets syntax var otroligt olikt något vi hållt på med innan. Däremot var tillvägagångssättet relativt straightforward. Det var dock väldigt skafft att ha ALLA variabler definierade under 'working-storage section'. Det var som sagt rätt så hemskt att implementera i Cobol och det tog ganska mycket tid, men approachen var ganska simpel så fort man kommit in i Cobol tänket.

2 Smalltalk (70-tal)

Smalltalk hade hemsk syntax, men hade en ganska straightforward approach. Man definierade två funktioner i enlighet med pseudokoden och sen kallar man på bernoulli-funktionen (*b(n,k)*) för att hitta det n:te bernoullitalet. Smalltalk är nog det språk som hade sämst resurser online av de som gjort hittills (alltså alla språk förutom cobol). Därför var det traggligt att få till syntaxen.

3 Erlang (80-tal)

Erlang var, rent ut sagt, förjävligt att jobba med. Det blev ganska traggligt att lösa pga att språket var funktionellt samt saknar loopar som i imperativa språk. Man behövde jobba med rekursion eller andra hjälpfunktioner för att lösa uppgiften.Det blev väldigt osmidigt att översätta pseudokoden då denna verkade utgå från ett mer imperativt språk. Då kunde man inte bara direktöversätta pseudokod till Erlang. Därför behövde vi kolla upp definitionen för det N:te bernoullitalet rent matematiskt för att säkerställa att beräkningen blev korrekt. Däremot var det skönt att Erlang (som är funktionellt) har liknande drag med Haskell eftersom man då kunde använda sig av sina kunskaper från Haskell till att applicera på Erlang och detta underlättar kodandet aningen. Exempelvis användandet av foldl som även finns i Haskell.

4 PHP (90-tal)

PHP var relativt likt många språk som vi har arbetat rent tankemässigt. Vi definierade två funktioner som tog in inparametrar och sedan returnerade en variabel. Dessa funktioner kan sedan kallas på för att outputta i terminalen. Det som var enkelt var tankesättet då man bara kunde följa pseudokoden, det som däremot var svårt var att lära sig syntaxen som ex. klamrarna som behövs över hela programmer <?php ... ?>samt ex. att alla variabler alltid måste ha $ framför sig.

5 Clojure (00-tal)

Clojure var väldigt svårt att skriva i eftersom det inte är ett särskilt vanligt språk och har helt ny syntax. Det fanns inte superbra dokumentation på språket när man sökte på nätet och dessutom så är syntaxen så annorlunda att det är svårt att riktigt komma in i hur man skriver programmet trots att man redan har pseudokoden. Dessutom använder man extremt många parenteser i clojure vilket blir lite luddigt att hålla koll på ibland. Det finns vissa instanser där det är typ 5 högerparenteser på samma rad. Däremot var det enklare såklart strukturen på själva programmet då man definierar funktioner som vanligt och sedan kallar på dem. När man väl började förstå syntaxen var det ganska straightforward.

6 Rust (10-tal)

Rust är relativt lik många språk vi redan har jobbat med i det sättet man definierar funktioner och sedan anropar dessa från en main-funktion. Det som framförallt var krångligt var att se till att alla typer var korrekta och definiera vilka variabler och parametrar som skulle vara av vilken typ samt se till att beräkningar görs med variabler av samma typ. Men som sagt var det relativt enkla att språkets struktur påminner mycket om språk vi har jobbat med såsom Java då man anropar definierade funktioner från en main-funktion och har method-headers som i stora drag liknar Javas.

7 Allmän reflektion

Vi började från nutid till dåtid vilket innebar att det började enkelt och blev jobbigare och jobbigare då vi kom längre och längre ifrån den programmeringen man själv är van vid. Däremot var det en rolig uppgift att få upptäcka så många olika språk. Det uppskattas också att programmet var relativt simpelt samt att man fick pseudokod att utgå ifrån. Slutligen skulle vi båda påstå att det var en väldigt rolig labb och den uppskattas!