Projektrapport F# ”Breaking the cipher”

DVA229 Funktionell programmering med F#

Jonathan Larsson

JLN14010

Mälardalens högskola

21/4-16

**Hur jag löste deluppgift a)** (Your task is to write a program in F# that decodes a text file that has been encrypted using a Caesar cipher. You can assume that only the ASCII letters (A-Z) are shifted in the encrypted text, and that the text contains only capital letters. (Hint: consider how you can find c. How many different c's are there? Is it feasible to use brute force?)**:**

Detta är ett enkelt chiffer där bokstäver är flyttade ett visst antal steg till höger, för att lösa chiffret så flyttar man helt enkelt tillbaka bokstäverna samma antal steg till vänster.

Eftersom chiffret i den här uppgiften bara innehöll stora bokstäver, inga speciella tecken samt att användaren skulle vara involverad i lösningen så var det inte speciellt svårt att komma fram till en lösning. En av delfrågorna i uppgiften var: hur många c (antal steg) kan det finnas? Eftersom det finns 26 bokstäver i alfabetet så kan bokstäverna flyttas 0-25 steg, flyttas de 26 steg så är vi tillbaka på samma sträng som om de skulle flyttas 0 steg. Lösningen jag kom fram till var att läsa in filens innehåll. Sedan för varje nyckel (antal steg) gå igenom strängen, bokstav för bokstav och flytta bokstaven så många steg som nyckeln. När en sträng har skapats med en nyckel så skrivs svaret ut och programmet går vidare till nästa nyckel. Efter att nyckel 0-25 har skrivits ut så är programmet klart och användaren kan klart och tydligt se vilken nyckel som är den rätta. Som svar på andra frågan ”Is it feasible to user brute force?”, ja det är väldigt användbart att använda ”brute force” eftersom det bara finns 26 olika nycklar som kan vara den rätta.

I min kod har jag gjort detta genom att först och främst ha en global variabel ”alphabet” som är en lista med alla bokstäver i alfabetet i alfabetisk ordning. Jag har sedan en funktion ”ShiftLeft” som tar två argument, det första är hur många steg som bokstaven ska ändras, det andra är vilken bokstav som ska ändras. Eftersom alfabetets stora bokstäver alla har ett integer-värde från 65-90 så använda jag detta för att returnera den rätta bokstaven. Om t.ex. bokstavens integer-värde var 66 och den skulle flyttas 2 steg så kunde den inte flyttas till bokstaven med integer-värde 64 eftersom den inte finns i listan. Den fick istället flyttas 1 steg från bokstaven med integer-värde 90. Detta var en enkel lösning med bara en if-else-sats. För att dekrypta en sträng så gjorde jag en funktion ”DecryptCeasar” som tar två argument, det första för antalet steg bokstäverna ska flyttas och det andra en lista med bokstäver som utgör strängen som ska dekryptas. Detta är en rekursiv funktion som går igenom varje element i listan och returnerar en ny lista som är flyttad ett visst antal steg. Den sista funktionen som jag gjorde för att enkelt köra dekryptionen var ”ShowCeasar” som också tar två argument, ett för antal steg alla bokstäver ska flyttas och det andra för listan med bokstäver som utgör strängen. Den är också rekursiv och skriver ut varje dekrypterad sträng med nyckel 0-25.

**Hur jag löste deluppgift b)** (Your task is to write a program that decrypts a text that has been encrypted using a mixed alphabet of the type described above. (Hint: search the web for information about the relative frequencies of letters in the English language):

Detta är ett chiffer där bokstäverna i alfabetet har flyttats runt till en slumpmässig ordning. Om t.ex. ”A” ska skrivas så skrivs den bokstav som är på A:s plats istället. Det här gör att chiffret inte har någon enkel lösning utan att involvera användaren en hel del. Eftersom det finns mycket forskning inom frekvensanalyser i det engelska språket så finns det en ”lösning” man kan utgå ifrån för att få fram en liten del av resultatet. Min lösning var först och främst ha en global variabel ”freqAlpha” som var en lista med bokstäver och tecken i den ordning som vissa frekvensanalyser angav. Den bokstaven som används mest hamnade på plats 0 och så vidare. När jag sedan har läst in filens innehöll så gick jag igenom varje bokstav i det ”alfabetet” och räknade hur många gånger den bokstaven fanns. Efter det sorteras listan efter hur många gånger bokstaven fanns. T.ex. så används mellanslag absolut mest av alla tecken i det engelska språket. Därför kunde den bokstaven som fanns flest gånger i filen bytas mot mellanslag på varje plats. Samma sak gjordes för hela listan och förhoppningsvis hamnar vissa bokstäver på rätt plats. Skulle samma texter som använts med forskningen bakom frekvensanalyserna användas så skulle chiffret lösas direkt. För att lösa chiffret för den här texten så behöver en del bokstäver och tecken byta plats i alfabetet.

Kodningsmässigt så löste jag det här med en hel del rekursiva funktioner. Det första som programmet gör är att göra om strängen till lista av bokstäver. Listan skickas sedan med till den rekursiva funktionen ”FindOcc” som tar frekvensalfabetet och listan med bokstäver som argument. FindOcc går sedan igenom frekvensalfabetet och för varje bokstav går den igenom listan med bokstäver för att räkna hur många gånger den bokstaven finns med och skapar en lista av tupler med antalet gånger bokstaven finns med samt bokstaven. Detta görs genom den rekursiva funktion LoopText som tar en bokstav och en lista som argument. Den går igenom listan och för varje element kollar den om det elementet är bokstaven som har skickats in. När hela listan är tom så returnerar den antalet gånger en viss bokstav fanns med i listan. För att programmet ska köras korrekt måste tupel-listan som returneras sorteras med den högsta siffran först. Tupel-listan, frekvensalfabetet och listan som utgör filen skickas sedan till den rekursiva funktionen ”FindLetters” som går igenom listan med bokstäver, för varje bokstav hittar den platsen som bokstaven finns i frekvensalfabetet och byter den bokstaven mot den på motsvarande plats i tupel-listan genom funktionen ”SwapLetters”. När FindLetters är klar så har den returnerat en lista med bokstäver som utgör den dekrypterade strängen.

**Hur används/körs programmet?**

Programmet har kod i main-funktionen som gör detta till en interaktiv applikation. Konsolen startar och användaren får välja antingen Ceasar- eller Mixed Alphabet-körning. Om Ceasar körs så skrivs helt enkelt de 26 nycklarna med respektive sträng ut i konsolen. Där kan användaren gå igenom alla nycklar och hitta den rätta.

Mixed alphabet är lite klurigare eftersom den kräver mer av användaren. Här har jag gjort så att strängen som har dekrypterats med frekvensalfabetet visas. Om användaren inte är nöjd med dekryptionen så kan han/hon kopiera alfabetet längst ner, kopiera in det i konsolen och byta plats på de bokstäver som han/hon vill. När användaren sedan klickar enter så kommer en ny sträng som har dekrypterats med det nya frekvensalfabetet skrivas ut. När jag själv dekrypterade detta så kollade jag snabbt igenom strängen, såg att ord som t.ex. THAT, HE, TO, TAKE, ANY, WHEN och AT verkade förekomma ganska ofta samt att mellanslagen verkade förekomma tillräckligt ofta. Då antog jag att t, h, a, e, o, k, n, y och w antagligen var på rätt plats. Sedan letade jag efter ord med dessa bokstäver i sig och försökte hitta ord som nästan var riktiga ord. Efter en del byten i frekvensalfabetet så var strängen helt dekrypterad. När användaren byter plats på bokstäver i frekvensalfabetet så betyder det att man byter plats på hur ofta en bokstav finns i filen. Det tecknet som används mest är mellanslag och näst mest bokstaven E. Om användaren då byter plats på dessa i alfabetet kommer filen dekrypteras genom att det tecknet som finns flest gånger i filen kommer bytas till E och näst flest gånger till mellanslag.

Om man vill se den rätta dekryptionen direkt så kan man byta det man skickar in som argument i funktionen ”FindLetters” på rad 101 i koden från freqAlpha till decryptFreqAlpha som är en global lista med det rätta frekvensalfabetet för den här filen.

Som ”bibliotek” skulle en programmerare använda mina funktioner så som jag har gjort. Ceasar-dekryptionen används enkelt genom att skicka in 0 och filens text till ”ShowCeasar”. Mixed alphabet dekryptionen skulle användas genom att skicka in sitt frekvensalfabet och strängen som ska dekrypteras till ”FindOcc”, sortera listan som returneras på högsta siffra först. Sedan skicka in den, frekvensalfabetet och strängen som ska dekrypteras till FindLetters som returnerar den dekrypterade listan. Om man vill kunna fortsätta försöka gör man en while-loop som i mitt program där ”FindLetters” fortsätter köra med nya frekvensalfabet tills den rätta har hittats.