Tekniska Högskolan i Linköping
Institutionen för Datavetenskap (IDA)
Torbjörn Jonsson 2008-11-07

Luffarschack (miniprojektuppgift)

I denna miniuppgift kommer vi att nyttja alla de saker vi tagit upp i de olika laborationerna. Dessutom kommer vi att använda ett litet hembyggt bibliotek som kallas TJa-biblioteket. Detta finns det mer information om på kurshemsidan.

Mål

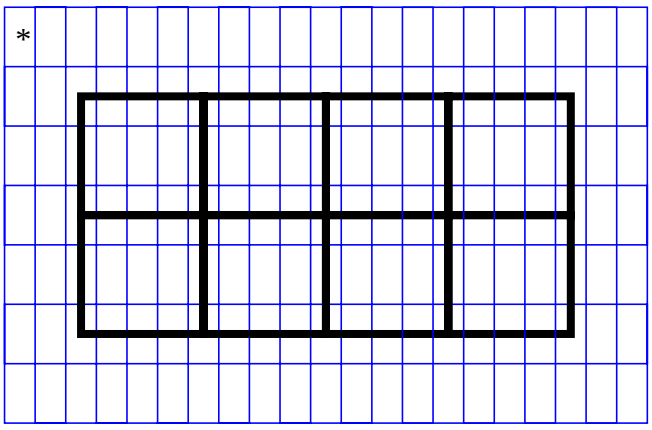
Du ska efter denna miniprojektuppgift ha fått mer känsla för när man använder de olika konstruktionerna ett programspråk kan erbjuda. Förhoppningsvis kan det också vara lite kul att se att man kan skriva något på ”riktigt” och inte bara göra små delmoment som belyser vad en programmerare skall kunna.

Det är INTE meningen att detta skall göras EFTER labserien utan att det skall göras allteftersom det är möjligt att gå vidare. Gör det efter labserien är det ganska mycket som kommer att behöva göras och det leder till att ni inte kommer att hinna med allt innan kursen är slut.

Uppgift 1

Skriv ett program, med namnet Draw_Grid, som ritar ut ett rutmönster på skärmen med ett givet antal rutor i x- respektive y-led. Det är meningen att ni skall använda er av de rutiner som finns i TJa-biblioteket för att rita grafiska tecken och positionera dessa på skärmen.

När man tittar lite närmare på hur ett tecken skrivs ut på skärmen ser man att det inte tar upp en kvadratisk yta. De grafiska tecknen är lika i format som de vanliga. Detta leder till att man får kompensera sin utskrift för att få rutorna så kvadratiska som möjligt. Nedan ser vi hur skärmen (terminalen) kan vara uppdelad och hur man kan tänka sig att de grafiska tecknen placeras in (de blå rutorna avgränsar tecknen på skärmen och övre vänstra koordinaten är (1, 1), markerad med *).



För att kunna bygga vidare på detta program senare inför vi direkt ett antal konstanter (som vi givetvis kan ändra på senare om vi vill). Dessa skall till en början vara (så stämmer de överens med figuren ovan):

```
Upper_Left_Corner_X = 3 Upper_Left_Corner_Y = 2
No_Of_Squares_X = 4     No_Of_Squares_Y = 2
No_Of_Spaces_X = 3      No_Of_Spaces_Y = 1
```

När du fått ditt program att fungera skall du ändra på konstanterna lite för att se att det fungerar även för andra värden. Öka t.ex. No_Of_Squares_X och No_Of_Squares_Y till 10.

Uppgift 2

Ni skall nu skriva om ditt program lite. Det som skall göras är att ni skall skapa ett antal underprogram som löser vissa delproblem istället för att ha allt i iett enda huvudprogram. Därefter skall ni se till att ni kan börja spela luffarschack.

Ett tips är att man kan börja med huvudprogrammet och göra en del i taget så att man kan testa att det fungerar. Det blir mycket jobbigare att göra detta sist om man gör alla delar på en gång.

Underprogrammen

Till att börja med skall ni skriva ett nytt huvudprogram som heter ”Luffarschack”. Till detta skall ni flytta alla konstanterna (som nu kommer att bli globala för hela huvudprogrammet samt alla underprogram).

Placera därefter er procedur Draw_Grid (utan konstanterna) in i detta nya huvudprogram.

Skriv ett nytt underprogram som skall heta Place_Cursor_In_Square som placerar markören i rutan som anges av de två parametrarna X och Y. OBS! Dessa X och Y är inte teckenpositionen på skärmen utan rutkoordinaterna. Inuti detta underprogram räknar ni ut vilken den exakta skärmkoordinaten kommer att bli och flyttar markören till den positionen.

Ett ytterligare underprogram skall skapas. Det är ett underprogram som ni skall kalla Move_Cursor. Detta underprogram skall läsa av tangentbordsinmatningarna och uppdatera X och Y (rutkoordinaterna) till nya värden. Om man trycker Ctrl-C skall underprogrammet returnera ett värde som anger att huvudprogrammet skall avslutas. Om man trycker på SPACE skall underprogrammet returnera den aktuella (X, Y)-koordinaten. Piltangenterna skall ge förflyttning av (X, Y) och returnera det nya (X, Y)-värdet. Övriga tangentnedtryckningar skall ignoreras.

OBS! När man använder TJa-bibliotekets tangentbordshantering fungerar inte Ctrl-C som vanligt.

Huvudprogrammet

Definition av spelet luffarschack: Det spelas på en spelplan bestående av ett stort antal rutor (minst 10x15). Det gäller att få fem stycken av den egna markeringen (X eller O) i rad antingen horisontellt, vertikalt eller diagonalt. Man turas om att sätta en markering av sin sort.

Ni skall nu skriva ett luffarschacksprogram som motsvarar själva spelbrädet och som håller reda på vems tur det är. Programmet skall dock inte lagra data om var man placerat sina markeringar eller om någon vinner.

Skriv luffarschacksprogrammet så att man kan spela det genom att turas om att flytta markören till en ruta och sätta in omväxlande X och O. Programmet skall sköta utskriften av rätt markering automatiskt. Det enda användaren skall göra är att använda piltangenterna för att flytta markören till rätt ruta och sen trycka på SPACE (inte RETURN) när markeringen skall sättas in. Markören skall starta någorlunda i mitten av brädet när spelet börjar. För att avsluta ska användaren trycka på Ctrl-C.

Uppgift 3

Lägg till en datastruktur (en datatyp som är en matris med namnet Board_Type) som lagrar data om vilka rutor som innehåller vilken markering (om det finns någon). I detta fall antar vi att dimensionen på rutnätet är fast så slipper vi problemet med att man inte kan omdimensionera matriser. Använd konstanterna som finns i huvudprogrammet så slipper du en massa problem.

Till denna datastruktur skall ni skapa ett antal underprogram som hanterar datastrukturen. Detta för att inte den som skriver huvudprogrammet skall behöva bry sig om detaljerna om datastrukturen utan istället se allt som lite övergripande gällande luffarschack. De underprogram som kommer att behövas är:

```
function Empty_Board return Board_Type;
-- Skapar ett tomt bräde.

function Empty_Square(Board : in Board_Type;
                      X, Y  : in Positive) return Boolean;
-- Kontrollerar om rutan är tom.

procedure Insert(Board : in out Board_Type;
                 X, Y  : in     Positive;
                 Mark  : in     Character);
-- Stoppar in markeringen (X eller O) på pos (X, Y).
-- Obs! Skriver inget på skärmen.

function Five_In_A_Row(Board : in Board_Type;
                       X, Y  : in Positive) return Boolean;
-- Kontrollerar om det finns fem i rad. (X, Y) anger senaste
-- rutan man stoppat in något i.

function Full(Board : in Board_Type) return Boolean;
-- Kontrollerar om brädet är fullt.
```

Modifiera ditt luffarschacksprogram så att det kontrollerar att användaren stoppar in saker i tomma rutor samt att det avslutas med ett meddelande om vem som vunnit eller om det blivit oavgjort.

Tips: Om man tänker till lite kan man slippa skriva en massa likadan kod. Att ha likadan kod är inte bra programtekniskt. Se speciellt i underprogrammet Five_In_A_Row.

Uppgift 4

Dela upp ditt program i olika moduler (paket). En lämplig uppdelning är att skapa ett paket, Board_Handling, för datatypen Board_Type med tillhörande underprogram och ett paket, Grid_Handling, för det som har med dimensionering och utritning av själva rutnätet att göra.

Det kommer att behövas ett extra paket som innehåller de konstanter vi har definierat. Detta för att vi skall kunna använda dessa i de båda andra paketen samt i huvudprogrammet. I detta paket gör vi dessutom så att vi lägger konstanterna som privata variabler istället för konstanter. Detta gör att vi skulle kunna komma på tanken att man skulle kunna initiera dessa konstanter från huvudprogrammet i en förlängning av programmerandet.

För att komma åt ”konstanternas” värden behöver vi ett antal funktioner som returnerar konstanternas värden. Vi döper därför dessa funktioner till det som konstanterna hette tidigare och döper om ”konstanterna” till något lite annorlunda. Detta gör att vi slipper ändra på alla ställen i de andra modulerna där vi använder konstanterna.

För att man i huvudprogrammet skall kunna sätta initialvärden på dessa variabler inför vi ett antal procedurer som sätter dessa ”konstanters” värden. Dessa namnger vi Set_Upper_Left_Corner, Set_No_Of_Squares och Set_No_Of_Spaces och de skall allihop ta emot två parametrar (X och Y). Dessa indata stoppas sen in i respektive ”konstant”.

OBS! De ”konstanter” ni nu har i detta paket är egentligen globala variabler som inte kommer att skickas in som parametrar i underprogrammen i paketet. Detta är egentligen lite ”galet”, men det är ett bra sätt att lösa just sådana problem.

OM NI VILL kan ni skapa en initieringsprocedur som läser in data och sätter de globala ”konstanterna” i ert huvudprogram. Detta är dock inget krav i uppgiften.