Aflevering af 2g

Oliver Fontaine Raaschou (vns328)
Peter Asp Hansen (glt832)
Shatin Nguyen (hlv332)

23. februar 2023

Denne aflevering tager udgangspunkt i de to opgaver

- 2g0
- 2g1

1 2g0

Denne opgave består af tre delopgaver, hvor vi skal lave tre funktioner som arbejder med vektorer. En vektor er defineret således: $\vec{v} = (x, y)$, så vi laver en type vec = float*float til at lave en vektor. Den første funktion skal tage imod to vektorer og returnere de to vektorer lagt sammen. Til at løse første delopgave *a* definere vi en funktion *add* som tager imod to vektorer og returnere en vektor. Vi anvender en *let* binding til at opdele vektoren fra dens x og y værdier og anvender formlen $\vec{v_1} + \vec{v_2} = (x_1 + x_2, y_1 + y_2)$.
I næste delopgave skal funktionen tage en vektor og en konstant, og returnere vektoren ganget med konstanten. Til at løse denne opgave definere vi funktionen *mul* og anvender vi igen en *let* binding til at opdele vektoren og anvender formlen $a\vec{v_1} = (ax_1, ay_1)$.
I sidste delopgave skal funktionen tage en vektor og en konstant, og returnere vektoren roteret med konstanten mål i grader. Til at løse opgaven definere vi funktionen *rot* og opdeler igen en vektor ved *let* binding. Nu anvender vi formlen $R_a\vec{v_1} = (xcos(a) - ysin(a), xsin(a) + ycos(a))$. Til at bruge cosinus og sinus i F, bruger vi System.Math.Cos(a) til cosinus og System.Math.Sinus(a) til sinus. Der henvises til XML koden for en beskrivelse af funktionerne. Her vises koden til opgave 2g0.

```
type vec = float*float

///<summary> add tager to vektorer og laegger dem sammen </summary>
/// <param name="a"> vektor. </param>
/// <param name="b"> vektor. </param>
/// <returns> resultatet af de to vektorer i form af en ny vektor </returns>
let add (a:vec)(b:vec): vec =
    let x1,y1 = a
    let x2,y2 = b
    ((x1+x2),(y1+y2))
let vec1:vec = (1.0,2.0)
let vec2:vec = (2.0,3.0)
printfn "Summen af de to vektorer er: %A" (add vec1 vec2)

///<summary> mul tager  n  vektor og ganger den med konstanten k </summary>
///<param name="a"> vektor. </param>
///<param name="k"> konstant. </param>
///<returns> en ny vektor som er k st rre end a. </returns>
let mul (a:vec)(k:float): vec =
    let x,y = a
    ((x*k),(y*k))
printfn "Vektoren ganget med 5 er: %A" (mul vec1 5)
//vektoren ganges med konstanten 5 og printes

///<summary> rot tager  n  vektor og roterer den med vinklen a </summary>
///<param name="b"> vektor. </param>
///<param name="a"> en konstant vinkel. </param>
///<returns> den roterede vektor med vinklen a </returns>
let rot (b:vec)(a:float):vec =
    let x,y = b
    ((x*System.Math.Cos(a)-y*System.Math.Sin(a)),
    (x*System.Math.Sin(a)+y*System.Math.Cos(a)))
printfn "Vektoren roteret med pi er: %A" (rot vec1 System.Math.PI)
//vektoren roteres med pi og printes
```

Figur 1: 2g0 kode

Som det kan ses i koden, er der også inkluderet et par eksempler på brugen af funktionerne. Dette ses fra vores output:
Første vektor: (1.0, 2.0) og anden vektor: (2.0, 3.0).
2g0a: Summen af de to vektorer er: (3.0, 5.0)
2g0b: Første vektor ganget med 5 er: (5.0, 10.0)

2g0c: Første vektor roteret med pi er: (-1.0, -2.0)

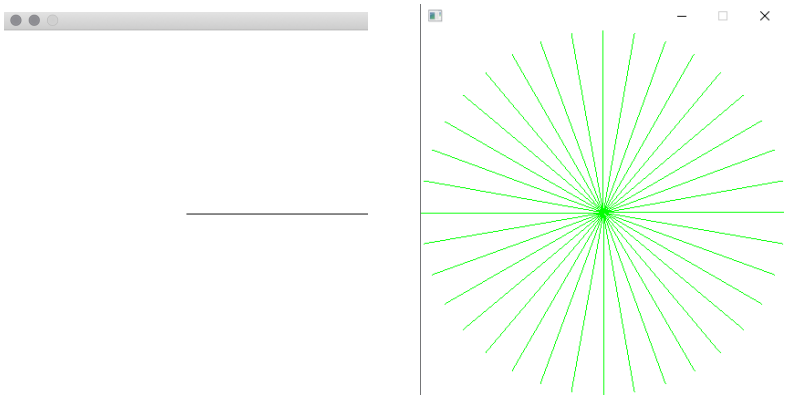
Opgaven går ud på at lave en funktion 'toInt' som tager to vektorer af floats og omdanner dem til to integers. Dernæst skal vi bruge vores funktion fra tidligere 'add' kombinere den med 'toInt' til en ny funktion 'setVector'. Denne funktion skal danne en linje fra p til p+v ved hjælp af `setLine`. Til sidst skal vi så bruge vores nye funktion `setVector` og den tidligere funktion 'rot' til at lave 36 linjer på et Canvas. Dette gør vi vha funktion `draw` som tager input `w h` til at specificere højde og bredde af Canvas. I funktionen benyttes en rekursiv funktion fan som matcher med n. Vi overvejede at hvis man ikke skal overholde præmisserne for funktionel programmering, kunne man også have benyttet et while-loop. Funktionen returnerer n linjer som roteres med en vinkel `n*j`.
Her vises koden til opgave 2g1

```
type vec = float*float

let toInt (a:vec):int*int=
    let x,y = a
    (int(System.Math.Round(x,0)),(int(System.Math.Round(y,0))))

let vec1:vec = (1.0,2.7)
printfn "Vektoren er: %A" vec1
printfn "Vektoren afrundet til int er: %A" (toInt vec1)

#r "nuget:DIKU.Canvas, 1.0"
open Canvas

let w = 400;
let h = 400;
let C1 = create w h
let p1:vec=(200.0,200.0)
let v1:vec=(200.0,0.0)
let c1 = black

///<summary> add tager to vektorer og laegger dem sammen </summary>
///<param name="a"> vektor. </param>
///<param name="b"> vektor. </param>
///<returns> resultatet af de to vektorer i form af en ny vektor </returns>
let add (a:vec)(b:vec): vec =
    let x1,y1 = a
    let x2,y2 = b
    ((x1+x2),(y1+y2))
```

```
///<summary> setVector tager fire argumenter, et canvas, en farve, en vektor
/// og en position, og tegner en linje. </summary>
///<param name="C"> canvas. </param>
///<param name="c"> farve. </param>
///<param name="v"> vektor. </param>
///<param name="p"> punkt. </param>
///<returns> En horisontal linje fra p til p+v. </returns>
let setVector (C) (c) (v:vec) (p:vec) =
    let pv:vec = add p v
    do setLine C c (toInt p) (toInt pv)

setVector C1 c1 v1 p1

///<summary> rot tager en vektor og roterer den med vinklen a </summary>
///<param name="b"> vektor. </param>
///<param name="a"> en konstant vinkel. </param>
///<returns> den roterede vektor med vinklen a </returns>
let rot (b:vec)(a:float):vec =
    let x,y = b
    ((x*System.Math.Cos(a)-y*System.Math.Sin(a)),
    (x*System.Math.Sin(a)+y*System.Math.Cos(a)))



///<summary> draw tager to argumenter w og h
/// og danner et canvas vha. et while loop </summary>
///<param name="w"> bredde af canvas. </param>
///<param name="h"> h jden af canvas </param>
///<returns> 36 linjer som er roteret med 10 grader per gang.</returns>
let n = 36
let j = (2.0*System.Math.PI)/float n
let C = create w h
let g = (0.0, float(h/2))
let draw (w: int) (h:int) =
     let rec fan C col g j n =
         match n with
            0 -> ()
            | _ ->
                let v = rot g (float n*j)
                setVector C col v ((float w)/2.0,(float h)/2.0)
                fan C col g j (n-1)
     fan C green g j n
     do show C "Canvas"

draw 400 400
```

Figur 2: 2g1 kode

Denne kode har følgende output:
Vektoren er: (1.0, 2.7)
Vektoren afrundet til int er: (1, 3)



(a) 2g1b (b) 2g1c

Figur 3: Canvas sider

Dette output returnere et eksempel på funktionen toInt benyttet på vektoren (1.0, 2.0) samt de to canvas sider 2g1b og 2g1c.