Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Arbejdsseddel 5 - individuel opgave

Jon Sporring

7. oktober - 15. oktober.
Afleveringsfrist: lørdag d. 15. oktober kl. 22:00.

I denne periode skal vi arbejde med abstrakte datatype, biblioteker og applikationer. En abstrakt datatype er et koncept som f.eks. lister, som kan beskrives vha. dets interface med signaturfiler og implementeres som et bibliotek i implementationsfiler. Biblioteker i F# kaldes også for moduler. Denne arbejdsseddels læringsmål er:

- at lave et bibliotek og en tilhørende applikation,
- at kunne oversætte biblioteksfiler og applikationsfiler både med `dotnet fsi` og `run`,
- at kunne arbejde med generiske moduler,

Opgaverne er opdelt i øve- og afleveringsopgaver. I denne periode skal I arbejde individuelt med jeres afleveringsopgaver. Regler for gruppe- og individuelle afleveringsopgaver er beskrevet i "'Noter, links, software m.m."'→"'Generel information om opgaver"'.

Øveopgaver (in English)

5ø0 In an earliere assignment, you implemented a small set of functions for vector operations in F#:

(a) addition of vectors

```
add: vec -> vec -> vec
```

(b) multiplication of a vector and a constant

```
mul: vec -> float -> vec
```

(c) rotation of a vector

```
rot: vec -> float -> vec
```

and wrote a small program to test these functions. Convert your earlier programs into a library called `Vec`, consisting of a signature file, an implementation file, and an application file. Create a `.fsproj` project file, and run the code using first `dotnet fsi` and then `dotnet run`.

Afleveringsopgaver (in English)

In the following, you are to work with the abstract datatype known as a queue. Queues appear often in real life: You stand in line at the counter in a shop, orders await in a queue for their turn to be shipped in an online shop, and students line up in a queue to be examined at oral examinations. Queues are defined by their element type and a set of operations typically,

```
type element // an element on the queue such as a plate
type queue // a queue of elements
create: unit -> queue // create an empty queue
// add an element to the end of a queue
enqueue: element -> queue -> queue
// remove the element in the front position of the queue
dequeue: queue -> element*queue
// check if the queue is empty
isEmpty: queue -> bool
```

In this exercise, you are to work with queues in F#.

5i0 In the following, you are to implement your own queue module using lists.

(a) Given the description of the abstract datatype Queue above, write a signature file for the queue module implemented with lists. The module is to be called `queue`.

(b) Write an implementation file, implementing the signature file above.

(c) Write an application file, which as minimum consists of the following series of tests of the stack module:

```
let q0 = create ()
printfn "%A, %A, %A" q0 (isEmpty q0) (dequeue q0)
let q1 = create () |> enqueue 1 |> enqueue 2 |> enqueue 3
printfn "%A, %A" q1 (isEmpty q1)
let (e,q2) = dequeue q1
printfn "%A, %A" e q2
```

5i1 A problem with the integer queue module is, that it does not handle errors, if `dequeue` is applied to an empty queue. Extend your library and the application such that `dequeue` returns an `(element option)*queue`. Write an application demonstrating that your implementation works.

5i2 Convert the integer queue into a generic queue, and demonstrate in your application, that you can build queues of different types with the same generic library.

Krav til afleveringen

Afleveringen skal bestå af

- en zip-fil, der hedder `5i.zip`

- en opgavebesvarelse i pdf-format.

Zip-filen skal indeholde:

- filen `README.txt` som er en textfil med jeres navn og dato arbejdet.
- en `src` mappe med følgende og kun følgende filer:

  5i0.fsi, 5i0.fs 5i0.fsx,
  5i1.fsi, 5i1.fs 5i1.fsx,
  5i2.fsi, 5i2.fs 5i2.fsx,

  svarende til afleveringsopgaverne. Funktionerne skal være dokumenteret med ifølge dokumentationsstandarden ved brug af `<summary>`, `<param>` og `<returns>` XML tagsne.
- pdf-dokumentet skal være lavet med LaTeX, benytte `opgave.tex` skabelonen, ganske kort dokumentere din løsning og indeholde figurer, der viser outputgrafik fra canvas for opgaverne.

God fornøjelse.