

io

Jon Sparring

November 21, 2019

1 Lærervejledningn

Emne Input/output

Sværhedsgrad Let

2 Introduktion

3 Opgave(r)

1. Write a program `myFirstCommandLineArg` which takes an arbitrary number of arguments from the command line and writes each argument as, e.g.,

```
$ mono myFirstCommandLineArg.exe a sequence of args
4 arguments received:
0: "a"
1: "sequence"
2: "of"
3: "args"
```

The program must exit with the status value 0.

2. Make a program `myFirstReadKey` which continuously reads from the keyboard using the `System.Console.ReadKey()` function. The following key-presses must result in the following:

- 'a' writes "left\n" to the screen
- 's' writes "right\n" to the screen
- 'w' writes "up\n" to the screen
- 'z' writes "down\n" to the screen
- shift+'q'** quits the program

All other key-presses must be ignored. When the program exits, then the exit status must be 0.

3. Make a program `myFirstReadFile` which

- (a) opens the text file “`myFirstReadFile.fsx`” as a stream using the `System.IO.File.OpenText` function,
- (b) reads each individual character using the `System.IO.StreamReader.Read` function,
- (c) writes each character to the screen using the `printf` function, and
- (d) closes the stream using `System.IO.FileStream.Close`.

The program’s exist status must be 1 or 0 depending on whether there was an error or not.

4. Make a program `myFirstWriteFile` which

- (a) opens a new text file “`newFile.txt`” as a stream using the `System.IO.File.CreateText` function,
- (b) writes the characters ‘a’ ... ‘z’ one at a time to the file to using `System.IO.StreamReader.Write`, and
- (c) closes the stream using `System.IO.FileStream.Close` function.

The program’s exist status must be 1 or 0 depending on whether there was an error or not.

5. Write a function,

```
filenameDialogue : question : string -> string
```

which initiates a dialog with the user using the question. The function should return the filename the user inputs as a string. If the user wishes to abort dialogue, then the user should input an empty string.

6. Make a program with the function,

```
printFile : unit -> unit
```

which initiates a dialogue with the user using `filenameDilaogue` from Exercise 5. The function must ask the user for the name of a file, and if it exists, then the content is to be printed to the screen. The program must return 0 or 1 depending on whether the specified file exists or not.

7. Make a program with the function,

```
printWebPage : url:string -> string
```

which reads the content of the internet page `url` and returns its content as a string option.

8. Make a calculator program

```
simpleCalc : unit -> unit
```

which starts an infinite dialogue with the user. The user must be able to enter simple expressions of positive numbers. Each expression must consist of a value, one of the binary operators `+`, `-`, `*`, `/`, and a value. When the user presses `<enter>`, the the expression is evaluated and the result is written as `ans = <result>` with the correct result entered. The input-values can either be a positive integer or the string “ans”, and the string “ans” should be the result of the previous evaluated expression or 0, in case this is the first expression typed. As an example, a dialogue could be as follows:

```
$ simpleCalc
>3+5
ans=8
>ans/2
ans=4
```

Here we used the character > to indicate, that the program is ready to accept input.

If the input is invalid or the evaluation results in an error, then the program should give an error message, and the input should be ignored.

9. Make a program with a function,

```
fileReplace :
  filename:string -> needle:string -> replace:string -> unit
```

which replaces all occurrences of the string needle with the string replace in the file filename. Your solution must use the System.IO.File.OpenText, ReadLine, and WriteLine functions.

10. In the html-standard links are given by the <a> tags. For example, a link to Google's homepage is written as Press to go to Google.

Make a program

```
countLinks : url:string -> int
```

which reads the page given in url and counts how many links that page has to other pages. You should count by counting the number of <a substrings.

Notice, most internet pages requires a valid certificate before they will allow your program to access it. By default, Mono has no certificates installed. One way to install useful certificates is to use mozroots, which is part of the Mono package. On Linux/MacOS you do the following from the console:

```
mozroots --import --sync
```

On Windows you type the following (on one line)

```
mono "C:\Program Files (x86)\Mono\lib\mono\4.5\mozroots.exe" --import
--sync
```

Note that your installation of mozroots may be in a different path, and you may have to adapt the above path to your installation. After running the above, your program should be able to read most pages without being rejected.