Hassling with the IDE - for Haskell

The first step into anything is a proper work environment.

2019-09-04

This guide is for students that are new to the Haskell language and are in need for a proper **I**ntegrated **D**evelopment **E**nvironment (IDE) to work on projects.

This guide is intended for Windows. It may work similar in Linux and iOS.

For any errors, please contact me at w.b.wijnia@uu.nl.

1. Installing Visual Studio Code

Visual Studio Code (VSC) is a light weight, but rather extensively, code editor. We'll talk about its features throughout this guide. You can download VSC at:

```
https://code.visualstudio.com/download
```

Once installed, take note of the sidebar on the left. From top to bottom, it contains:

- **Explorer**
- **Search**
- Source control
- Debug
- **Extensions**

The **bold** options are important to remember. They will be mentioned throughout this document.

2. Installing GHC (and stack / cabal)

Download the Glasgow Haskell Compiler (GHC). This compiler is used throughout this course and any other course that uses Haskell. You can download GHC here:

```
https://www.haskell.org/platform/windows.html
```

I *really* recommend you to keep the default install paths. It prevents a few warnings (and therefore: potential problems) further down the road.

Once the installation is done, open up a console and type:

```
$ stack update
$ stack upgrade
```

These commands update the list of packages available to stack and upgrades all packages (including stack itself) already installed. This will prevent a few errors down the road.

3. Installing HLint

HLint is used by DomJudge to check for your style. It is a great tool to increase your understanding of your code and experiment with your understanding of the type system of Haskell. You can find more information at:

```
https://hackage.haskell.org/package/hlint
```

Take note that the tool is **not** perfect. Therefore, do not follow it blindly. Always try to understand the hints and tips that HLint is providing you with.

You can install hlint globally with:

```
$ stack install hlint
```

If (read: when) the installation process exits with an error, restart the installation process with the same command. It should properly install the second time.

A few useful commands within a console window:

```
$ hlint <file>.hs
$ hlint <folder>
$ hlint .
```

The output if hlint is directed to the console. The first command applies hlint to a given file, the latter two to all files in a specific folder or to the current working directory.

```
$ pwd

$ cd <absolute / relative path>
$ cd ..

$ ls -a -l
```

The command **pwd** shows your current working directory. The two **cd** commands allows you to **C**hange **D**irectory. Using two dots moves you up one level. You can auto complete folder names with via hitting tab. The command **ls -a -l** shows all files and folders in your current working directory.

4. Setup of Visual Studio Code workspace

Choose a folder in which you'll do all your development. Then open this folder within Visual Studio Code via:

```
File -> Open Folder
```

Once the folder is open, open up a terminal in that folder. If you use the terminal inVSC (*CTRL* + ` *or CTRL* + ~) it will automatically open up in your workspace (folder).

Create a small Haskell file called *notepad.hs* and write the following function within that file:

```
sum' :: [Int] -> Int
sum' (x:xs) = x + sum' xs
sum' [] = 0
```

Save the file. Go back to the terminal in VSC and type 'ghci'. This will start the interactive version of GHC, allowing you to load files and test individual functions in an interactive manner. The following commands are useful within GHCi:

```
$ :load <file>
$ :reload <file>
$ :quit
$ :type <function>
$ :import <library / package>
```

You can load and reload files to check whether they compile and, if they do, test their functionality. You can quit ghci and you can check the type of any function available in the **loaded packages**. When you open the ghci, only the Prelude is opened.

To test all of this, open up ghci and load in our *notepad.hs* file. Then type the following to call the function:

```
$ sum' [1,2,3]
```

You can immediately review its result. This is quite useful for debugging.

5. Visual Studio Code extensions and tips

5.1 Extensions

A useful extension is the 'Haskell Syntax Highlighting' extension. It adds in all kinds of pretty colors in the code editor. Go to the **Extensions** tab, search for it and install it.

Take note that there are a few auto-completion extensions available. Extensions such as Haskero or Haskelly. Installing these can be difficult and system dependent therefore we do not provide support for installing these extensions.

You can program in Haskell without auto completion and it is better for the sake of learning.

5.2 Tips

A few useful hotkeys:

```
CTRL + D              Select a keyword, then press CTRL + D. Press the D
                      repeatedly. Selects all keywords that match. You can
                      change them at once.

CTRL + H              Search for the selected keyword and change them all.
CTRL + F              Search for the selected keyword.
CTRL + SHIFT + F      Search for the selected keyword through all files.

CTRL + B              Open / close the sidebar.

ALT + CLICK           Repeats the cursor at the given location.
ALT + UP / DOWN       Moves the current line up or down.
```

With the console of VSC, you can use hlint in your current work folder via typing:

```
$ hlint .
```

The console always opens up at your work folder. Take note that if you're in GHCi then you cannot use hlint unless you quit. You can, luckily, open up a second console in VSC and use hlint in there.