MATHFUN
Discrete Mathematics and Functional Programming

Getting started with Haskell and GHCi

Introduction

As part of the MATHFUN unit we will learn the functional programming language Haskell. We will use the Glasgow Haskell Compiler (GHC) software, its interactive enviroment GHCi and, in particular, a version of GHCi called WinGHCi that includes a simple graphical user interface.

Start WinGHCi

Choose Haskell Platform 2013 from MyApps; MyApps is configured so that this will start WinGHCi.

Both versions of GHCi are used to execute ***expressions*** in Haskell typed at a prompt. GHCi is therefore like the Python shell you used last year. The main difference is that the Python shell, as well as evaluating expressions, can also execute statements. Since there are no statements in the Haskell language, GHCi can only evaluate expressions!

Try entering a few arithmetic expressions at the prompt; e.g.:

```
1 + 2.5
2 * (3 + 5)
5 / 2
3 ^ 2
```

etc. to see that GHCi can act as a calculator. Try also:

```
pi
sqrt 2
```

(We see that Haskell appears to have some built in constants and functions.) Try also:

```
"Hello" ++ "World"
```

(So Haskell also includes strings and string operators.)

Start NotePad++

We do not usually define functions directly within the GHCi environment – functions are instead defined using a separate text editor. NotePad++ is an ideal editor for use in the University labs. Often you'll be able to start NotePad++ from within GHCi, but at the University you'll always need to start it first from MyApps (do this now – choose NotePad++ 6.5.2).

Write a Haskell function definition

In a new file in NotePad++, write the following code:

```
mult2 :: Int -> Int
mult2 x = 2 * x
```

The first line is a declaration of your function's type: it says that the function `mult2` takes an `Int` (integer) value and returns another `Int`. The second line is the definition of the function: it says that for any argument value `x`, the function returns the value `2 * x`. (Haskell program files will typically contain a collection of such function definitions.) Save this file to a new folder (e.g. MATHFUN) on your N: drive with the name firstprog.hs (.hs is the suffix you should choose for Haskell programs).

Load and test the Haskell file

Within WinGHCi, load your firstprog.hs file (e.g. using the File menu). You should now be able to use your `mult2` function definition; try for example the following expressions at the prompt:

```
mult2 4
mult2 (3 + 1)
mult2 3 + 1
```

Note the use of brackets here: they are ***not*** used for function calls, but are used to control the order of evaluation.

Edit the firstprog file

Add a second Haskell function definition to the file:

```
mult4 :: Int -> Int
mult4 x = mult2 (mult2 x)
```

Save the file. Every time you make a change to a Haskell file, you need to reload it into WinGHCi (using the green reload button). The new function definition should be available from the GHCi prompt; try:

```
mult4 5
```

The GHCi type command

As well as evaluating expressions, GHCi can also tell you their type. To do this you use a GHCi ***command***; commands in GHCi all begin with a colon and are not part of the Haskell language; they are used within the GHCi environment to change settings, to navigate around the filesystem and to edit, load and reload Haskell files. The `:type` command gives the type of an expression:

```
:type mult2 3
```

Note that, in functional programming, functions themselves are expressions (and therefore have types). Try, for example:

```
:type mult2
```

Configure Notepad++

Whitespace (tabs, spaces, newlines) is used by Haskell to determine the meaning of some code (a little like Python). It is therefore important that your code does not use a mixture of tabs and spaces, since such code might appear well formatted to you but may be rejected by the Haskell interpreter. We recommend that you configure NotePad++ (and any other editor that you might use) to only use spaces (i.e. when you press the tab key, an appropriate number of spaces are added). Go to Settings → Preferences. Click on "Tab Settings". Make sure that "[Default]" is selected, that "Tab size" is 4, and that "Replace by space" is ticked. Select "haskell", and make sure that "Use default value" is ticked. Finally, click close.

Using the non-GUI GHCi

You may prefer to use the standard version of GHCi which does not include a GUI. The advantage of the non-GUI version is that it includes code completion (pressing the tab key to complete words), which can speed up the entering of long expressions. The disadvantage is that you need to use textual commands to move around the file system, and to load and reload files. Start GHCi (under Haskell Platform 2013 in the Start menu).

Navigate to the folder containing your firstprog.hs program using the following commands that move to a folder, move to the parent folder and move to the root folder:

```
:cd folder
:cd ..
:cd
```

You can also use Windows commands by adding '`!`' at the beginning; try the following to display the current folder and its contents.

```
:!cd
:!dir
```

Once in the appropriate folder, load your program into GHCi using:

```
:load firstprog
```

You should be able to start the editor from within GCHi by using:

```
:edit
```

and, if you do this, closing the editor should result in the file being automatically reloaded into GHCi. Otherwise, you can reload the file using:

```
:reload
```

You can quit GHCi using:

```
:quit
```

Note that all these GHCi textual commands can also be used within WinGHCi as an alternative to the GUI controls.

Abbreviations for GHCi commands

Note that the common GHCi commands can be abbreviated as follows:

`:load`	`:l`
`:reload`	`:r`
`:edit`	`:e`
`:type`	`:t`
`:quit`	`:q`

Experiment!

Feel free to experiment with the system. Modify the definition of `mult2`, or add extra function definitions to the firstprog.hs file.