Assignment Project Exam Help

COMP0020 Functional Programming

Lecture 4

https://powcoder.com

Miranda

(and her friend Amanda)

Add WeChat powcoder

Contents

Assignment Project Exam Help

- Miranda/Amanda
- Miranda demonstration

https://powcoder.com

- Comments
- Legal names and binding
- Types and type checking
- Tuples

Add WeChat powcoder

- Simple functions

Miranda / Amanda

Assignment Project Exam Help

- Miranda will run on Unix variants such as Linux, OS X / MacOS and Cygwin
- For Windows, you can choose either to install Cygwin on Windows and then install Miranda on Cygwin **or** you can try installing Amanda
- Amanda is a PC version of Miranda
- Amanda is almost (but not quite) the same as Miranda !
- Get it from the link on Moodle - although it hasn't been tested on the latest version of Windows.

https://powcoder.com

Add WeChat powcoder

Miranda Installation on OS X

- OS X's System Integrity Protection (SIP) feature (El Capitan +) prevents changes to /usr (but not /usr/local).
- Miranda installs into /usr/bin and /usr/lib by default.
- Thus, to install Miranda on OS X you must either :
  - (i) force installation in a different place such as /usr/local/bin etc, or
  - (ii) disable SIP, install Miranda in the standard place, then re-enable SIP
- The first way is better : it doesn't require rebooting OS X and there is no danger of forgetting to re-enable SIP. But the procedure is slightly ugly :
  - tar xpf mira-2044-x86_64-Darwin.tar -C /usr/local
  - sudo ln -s /usr/local/usr/local/bin/mira   /usr/local/bin/mira

Miranda Demonstration

- Interpretive environment
- Simple use as a calculator
- Simple definitions stored in a file — default name "script.m" (the file "script.x" contains the intermediate code that is interpreted)
- Main definition — the function called "main" is special in Amanda but not in Miranda
- Define and use functions

Miranda Demonstration

- Similar to the Lambda Calculus

$$Miranda \ (3 + 4) * (6 + 7)$$

- But (inside a program file) you can give names to expressions

$$x \ = \ 3 + 4$$
$$y \ = \ 6 + 7$$
$$main \ = \ (x \ * \ y)$$

- Also give names to functions (no lambdas!)

$$inc \ x \ = \ x + 1$$
$$main \ = \ inc \ 56$$

Comments

- VERY important !
- Use them from the start
- Example :

```
||a simple definition for some text :
message = "hello mum"

||here is a function which adds one to a number :
inc x = x + 1    ||another comment here
```

Legal names

- BINDING :
  - ▶ NAME = EXPRESSION
  - ▶ funcname argname = EXPRESSION
    - ★ Binds funcname when defined (static)
    - ★ Binds argname when applied to an argument (dynamic)
  - ▶ Each binding is <u>unique</u>, within specified scope
    - ★ Scope of argname is the function body (only)
    - ★ Nested scopes (see later) permit nested bindings for the same name (as we have seen previously, the λ-calculus supports such name games, but they are generally not a good idea)
- Names MUST start with an alphabetic character
- Names MUST NOT start with a capital letter
  - ▶ and thereafter may contain uppercase and lowercase characters, numbers and underscores

Types

- Data can be, for example :
  - ▸ Numbers (42) is of type **num**
  - ▸ Characters ('A') is of type **char**
  - ▸ Text ("string") is of type **[char]**
  - ▸ Truth values (True) is of type **bool**
  - ▸ Functions are of type **argtype -> resulttype**. E.g. the function ($f\ x = x + 1$) is of type **num -> num**
- Types help organise data and programs better
- Helps detect errors - the type system is a debugger

Type Checking and Type Inference

- Performed at compile time, before the program is run
- Checks that operators (e.g. +) are executed on data of the correct type
- Checks that functions are applied to data of the correct type
- On the command line you can ask "what type is this?"
- In a program you can specify "this is a Boolean value" etc.
- If types are not given, Miranda will attempt to INFER types from the way that variable names are used

Tuples

- A simple data structure
- ("Sheila Bloggs", 23, "2 Turnabout Road, NW3", 50, False)
- (34, True) is of type (num, bool)
- (34, True) DOES NOT EQUAL (True, 34)
- ("increment", (+1)) is of type ([char], num->num, num->num)

Simple Functions

```
inc : : num -> num
inc x = x + 1

|| the hello function
hello : : num -> [char]
hello x = "good morning", if x < 10
        = "goodbye", otherwise
```

Summary

Assignment Project Exam Help

https://powcoder.com

- Miranda/Amanda
- Miranda demonstration
- Comments
- Legal names and binding
- Types and type checking
- Tuples
- Simple functions

Add WeChat powcoder

Assignment Project Exam Help

END OF LECTURE

https://powcoder.com

Add WeChat powcoder