Practical Session 2: Pascal

August 2, 2016

1 Getting started

Download and install FPC, the Free Pascal Compiler[1].FPC is a Pascal standalone command-line compiler. To compile a Pascal source, use:

`$ fpc sourceCode.pas`

which should generate a `sourceCode.o` object file, as well as a `sourceCode` executable binary.

1.1 Error: Can't find unit

If the compiler complains that it can't find a unit with an error message `Fatal: Can't find unit unitName`, it probably means that your compiler is not configured to find the unit packages properly.

The first solution for this is to avoid using units entirely. You should not need them for these practical sessions.

If you really want to use units: locate the `fpc.cfg` config file (on Linux, try `.fpc.cfg` in your home directory) and check that the path to your units installation is included. Typically, on a Linux system, you should have a line with:

`-Fu/usr/lib/fpc/$fpcversion/units/$fpctarget/*`

1.2 Linker warning

The linker may give the error:

`warning: link.res contains output sections; did you forget -T?`

It can be safely ignored.

1.3 Using the compiler

Try to compile and run the following *Hello, World!* program.

```
program HelloWorld;
begin
        writeln('Hello, World!');
end.
```

[1] `http://www.freepascal.org/download.var`

2 Imperative programs

2.1 Procedures and conditionals

Pascal is a procedural programming language.

Write a Pascal function which implements the following function f:

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ 3n+1 & \text{if } n \text{ is odd} \end{cases}$$

2.2 Recursion

Write a program that queries the user for a number and checks the Collatz conjecture for this number.
The Collatz conjecture states that for any starting number, if we repeatedly apply the f function defined above, we will always eventually reach 1. Use recursion to apply f until reaching 1.
Your program should print every step of the convergence. Example of output, starting with $n = 13$:

```
$ ./collatz
Input a positive integer:
13
13 40 20 10 5 16 8 4 2 1
```

2.3 Loops

Re-write your Collatz conjecture program to use a loop instead of recursion.

3 Data Types

Pascal provides nice features for user-defined data types.

Enumerations allow you to enforce a certain defined value for a variable. Only enumerated values can be assigned to variable typed as an enumeration.

Define an enumeration type `booktype`. A book can be either a `novel` or a `comic`.

An important feature of Pascal is variant records. A variant record is a record in which some of the fields are dependent on the value of one specific field. In other words, some parts of the fields

You are in charge of writing the software to manage your local library. Write a program which defines a record for `book`s in a library. Each `book` should have a *title* (`string`) as well as a *kind* (`booktype`). Additionally:

- if the book is a novel, it should have an *author* (`string`).
- if it's a comic book, it should have a *writer* (for the scenario and dialogue) and an *artist* (for the drawings). Both are of course `string`s.

Try to play around with those data structures. What happens if you try to assign a value to the writer field of a novel, or the author field of a comic book? What happens if you read that value? Specifically, try to run the following and understand what is happening:

```
theHobbit.title := 'The Hobbit, or There and Back Again';
theHobbit.author := 'J. R. R. Tolkien';

luckyLuke.kind := comic;
luckyLuke.title := 'Billy the Kid';
luckyLuke.writer := 'Goscinny';
luckyLuke.artist := 'Morris';


{* Tests *}
writeln(theHobbit.kind);
writeln(theHobbit.writer);
writeln(theHobbit.author);
writeln(theHobbit.artist);

writeln(luckyLuke.kind);
writeln(luckyLuke.author);

luckyLuke.author := 'Did I break Pascal yet?';
writeln(luckyLuke.author);
writeln(luckyLuke.writer);
```

3.1