# Practical Session 2: Pascal

## 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 Flow Control

Pascal is an imperative programming language, like C for instance. As such, it offers the usual common idioms for loops, conditions etc.

Rewrite the AbsMean program of the Assembly session in Pascal.

## 2.2 Procedures and functions

Write a Pascal function which implements the following function $f$:

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

## 2.3 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
```

Write two solutions : one using recursion and one using an iterative process.

# 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.

Pascal allows programmers to define heterogeneous data structures called **record**s (think of C's structs) containing multiple fields of different types, including other records or enumerations.

Write a program which defines a **record** for books in a library. Each book should have a *title* and an *author* (string), a publication date (define an auxiliary **type** date) as well as a *kind* (booktype).

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, the variant fields have a different type depending on their use. This feature of Pascal can be abused to circumvent Pascal's static, strong typing, and allow the programmer to access the same memory location as different types.

Modify your book **record** to have additional records for novels and comic books. More precisely:

- if the book is a novel, it should have an *author* (string) and a ISBN number (Int64).
- 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 strings.

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';
theHobbit.ISBN := 9780007525492;

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


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

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

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

What does the garbage printed by theHobbit.artist correspond to? What happens if you set the ISBN to 31084784824439585 and print the artist field again?

## 4 Something else ????

FIXME