Introduction

1 Basic syntax

This course uses OCaml to teach programming fundamentals. Covered here are the basics of the syntax.

Once installed, one can enter use OCaml in the terminal by typing OCaml and writing your expression after the `#`. An expression should be ended with `;;`.

The simplest way to leave OCaml is to use `exit 0;;`. One can interrupt a running program by pressing Ctrl-C.

2 Concepts

An OCaml program is an *expression* (or a collection of expressions) with a *type*. This expression can be evaluated to give a *value*. Operators can be applied to sub-expressions to yield larger expressions. Some operators only apply to certain types.

The type **int** is the integers. Your machine will have a so called `min_int` and `max_int`, which are the smallest and largest integer that is available, respectively. Adding to `max_int` will wrap back around to `min_int` as `min_int = max_int + 1`. A similar situation occurs when subtracting from `min_int`. Thus, the integers from `min_int, ..., -1, 0, 1, ..., max_int` is are all of the objects with type **int**. It is worth being careful when working close to the edges of this range to prevent unexpected results.

The type **bool** is the Boolean values `true` and `false`. It is useful to have a separate type for Booleans rather than using, say, `0` and `1` as this would introduce potential situations where a non `0` or `1` value is returned inintentionally and acts to prevent unnecessary errors. This is possible as OCaml has a strict type system.

The type **char** is single characters, for example `'a'`, `'!'` and `'E'`.

Mathematical operators only act between expressions of type **int**. These are `+ - * / mod`. The result of such an expression will also have type **int** (remember this when using division operations).

Comparison operators compare expressions of the same type. This means that Booleans and characters can aslo be compared (try this to see the relative values of uppercase and lowercase charaters, and `true` and `false`). These are `= < <= > >= <>` where `<>` is the operator for "not equal to" in OCaml. The result of such an expression will have type **bool**.

Boolean operators `&&` and `||` are "and" and "or" respectively and compare expressions of type **bool**.

Operators of higher precedence are evaluated first. For example, `*` has precedence over `+` and "and" has precedence over "or". Otherwise, operators are evaluated from left to right.

If statements have the following syntax: `if` ***expression1*** `then` ***expression2*** `else` ***expression3*** where *expression1* has type **bool**, and *expression2* and *expression3* have the same type.