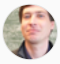

interpreter tweaks
mhe authored 1 hour ago

afad15cf

AbstractSyntax.md 1.37 KB

Abstract syntax

The operations are listed in the order of precedence as in the languages C and C++ (from low to high), where all the operations in the same line have the same precedence, but this is important for parsing only:

```
module AbstractSyntax where

type Identifier = String

data OpName = Or                                 -- ||
            | And                                -- &&
            | Eq                                 -- ==
            | Leq | Less | Geq | Greater         -- <=  <  >=  >
            | Add | Sub                          -- +  -
            | Mul | Div | Mod                    -- *  /  %
            | Not                                -- !
            deriving (Show)

data Expr = Constant Integer
          | Var Identifier
          | Op OpName [Expr]
          deriving (Show)

data Program = Identifier := Expr
             | Block [Program]
             | While Expr Program
             | If Expr Program
             | IfElse Expr Program Program
             deriving (Show)
```

Notice that we are using a constructor `:=` in the `Program` type, written in infix notation. We use monadic parsing to convert from concrete syntax to abstract syntax.

Next: Parser