Haskell

Misc

- GHCi
- 配置文件 ~/.ghci
- 进入命令行交互界面 终端中输入 ghci
- 代码后缀 `.hs`

第一节 基本语法

- 比较运算符左右的类型必须相同，并且可以比较， `4 == 5 return False; 5 \= 4 return True` 这里 `\=` 表示“不等于”
- 负数要用括号包起来，(这点这麻烦)， `5 * -3` 报错， `-3 * 5` 或者 `5 * (-3)` 都是语法正确的.
- 布尔表达式， `&& ||` 表示与和或，真是 `True`，假是 `False`.
- 函数调用： `function parameter1 parameter2 ...` 函数名 空格 参数 (空格间隔)
- 函数调用的优先级高于任何其他运算. 可以这样写： `succ 9 + max 5 4 + 1`
- 一些二元的中缀函数有两种调用方式： `div 92 10; 92 `div` 10` 都是做92 除以10的整数除法.
- 函数迭代的话，要用括号抱起来 `succ (succ 9)` 结果是11.
- if 语句强制性要有else，结构是： `if BOOL then do something else do others`
- 表达式的概念： **An expression is basically a piece of code that returns a value.** if语句也是一个表达式，因为每个判断都是强制要返回一个值的，而且if前置有else，所以总是有返回值
- 单引号可以作为一个字符出现在函数名变量名中，并没有什么特殊含义.
- 函数名不能以大写字母开头.
- 一个函数没有参数， `hehe' = "hehe"` 可以看做一个定义.
- 可以用 `let` 命令在ghci中直接定义函数，变量 `let a = 1`，定义之后a和1等价，不能改变.
- list Haskell中list是一些相同类型的数据的数据，连接连个list，用 `++`， `[1, 2, 3] ++ [4]`.
- `++` 操作的时间复杂度是线性的，会遍历操作符左边的list，如果希望时间效率高，用 `:` (cons operator)，会把一个数据放到list的最前面(不能放在其他地方).
- 数组下标从0开始，用下标访问数组元素，符号是 `!!`.
- 其他数组操作：
    - `head` 返回数组第一个元素.
    - `tail` 返回除了数组第一个元素以外的list.
    - `last` 和 `head` 相对，最后一个.
    - `init` 和 `tail` 相对，除了最后一个元素以外的list.

    - `length` 用一个list作为参数，返回长度.
    - `null` 判断list是不是空的. 空返回`True`，否则 `False`.
    - `reverse` reverse一个list.
    - `take` 传入一个数字k和list，从list中取出前k个元素. 如果list不够长，全部取出来. 返回值是个list. k是负数，抛出异常.
    - `drop` 和 `take` 相反，从头开始丢弃一些元素，丢光为止，负数抛异常.
    - `maximum` 去除最大的元素. 空list 跑出异常.
    - `minimum` 取出最小的元素. 空list 跑出异常.
    - `sum` 求和一个元素是数字的list. 空list，返回0.
    - `product` 求一个元素是数字的list的积，空list，返回 1.
    - `elem` 判断一个元素是不是数组的元素. 类型要相同. 是一个中缀函数，有两种写法，推荐写成中缀形式. `4 `elem` [1, 2, 3, 4]` 结果是`True`.

- 简洁地表达数组，`[1..20], ['a'..'z'], ['K'..'Z'], ['A'..'a']` ，最后一个会输出一个字符串，有所有ASCII码介于'A'和'a'之间的字符按顺序组成.
- 更换步长，`[4, 6..20]` 输出所有4到20之间(包含边界)的偶数，`[3, 6..20]` 输出所有3到20之间的3的倍数.
- 倒着的枚举数字，`[20, 19..1]` 是正确写法，`[20..1]` 是不对的. 不资瓷.
- 枚举数字的时候，注意浮点数会有精度问题. `[0.1 0.3 .. 1]` 会输出 `[0.1,0.3,0.5,0.7,0.8999999999999999,1.0999999999999999]`.
- `cycle` 将一个list循环，产生一个无穷的list，可以用`take`取出一些需要的.
- `repeat` 将一个元素循环，产生一个无穷的list.
- `_` 表示无论从list中取出的元素是什么,我们并不在乎他的值. 这种时候用`_`.
- `tuple` 和C++里类似，但是没有一元的元组，`fst` 取出第一项，`snd` 取出第二项. 这两个函数只能引用在二元组上.
- `zip` 将两个list的元素一次配对，返回一个二元组的list，直到有一个不够用.
- 过滤，`let rightTriangles' = [ (a,b,c) | c <- [1..10], b <- [1..c], a <- [1..b], a^2 + b^2 == c^2, a+b+c == 24]` 范例，这里b循环到c，a循环到b，满足了 $a \leq b \leq c$.

第二节 类型

- 在gchi中输入`:t`加上要判断类型的数据，可以得到数据的类型. Haskell不需要制定类型，可以自己进行类型推断.
- 函数显示的声明类型 `function name :: input type -> output type`，例如，`removeUpper :: String -> String`
- 类型都是以大写字母开头的.
- `Int` 是32为整型，`Integer`是大整型，`Float`，`Double`是浮点型.
- 函数的类型推断，`:t head` 返回是 `head :: [a] -> a` 这里`a`，不是一个类型，他是一个类型变量，`type variable`，用于表示任意的类型，可以多于一个字母.
- 函数有多个参数的时候，类型声明用`->`分割各个参数，最后一个类型是返回值的类型.
- `==, *, /, +, -` 都是函数，而且默认是中缀函数，但是可以被当做前缀函数，只要左右加上括号，`(==) 1 2` 返回 `False`.

- `type class`, **A typeclass is a sort of interface that defines some behavior.**
- `class constraint`, `:t (==)` 返回 `(==) :: (Eq a) => a -> a -> Bool` 这里 `=>` 符号就是一个 `class contraint` 表示 `(==)` 这个函数，将两个类型是 `a`， `a` 是某种可以被比较相等 `Eq` 的类型，这样的参数作为参数，返回一个bool．其中 `a` 类型必须是 `Eq` 这个 `typeclass` 的一个成员．
- `Eq` *Eq is used for types that support equality testing.*
- `Ord` *Ord is for types that have an ordering.*
- `compare` 函数的类型是 `compare :: Ord a => a -> a -> Ordering`，参数必须是 `Ord` 这个 `typeclass` 的成员，然后返回值是 `Ordering` 类型，即 `GT, LT, EQ` 这三种结果．
- `show` 这个函数，将一个输入的参数的值转化成字符串． `typeclass` 是 `Show`．
- `Read` 和 `show` 对应，将输入的字符串转化成一个表达式的值．但是这个值的类型是不知道的，只有 `read` 的结果被拿去做其他计算了，haskell才会通过类型推断推断出他的类型，所以 `read "4"` 这样的语句是不能通过编译的，因为返回的数据类型不能被推断，但是 `read "4" + 2` 结果就是6． `:t read` 返回的结果是 `read :: Read a => String -> a`，这里 `a` 是一个类型变量，但并不知道具体的类型．
- `Enum` 是另一个typeclass，这个typeclass的每个数据被定义了一个前驱和后继，所以可以被list range，以及应用 `succ` 和 `pred` 这两个函数，例如: `[LT..]` 就是 `[LT EQ GT]`． `succ 'B'` 及i `'C'`． `(), Bool, Char, Int, Ordering, Integer, Float, Double` 都在这个typeclass中．
- `Bounded` 这个typeclass的成员(这些成员是数据类型)都有一个上下届，可以用 `minBound, maxBound` 这两个函数得到上下届，这两个函数是多态(polymorphic)的．
- `Num` 是一个算数类，成员具有能被当成数字去操作 的属性．
- `*` 的类型，可以通过 `:t (*)` 看到，是 `(*) :: (Num a) => a -> a -> a`，可见他是一个多态函数，接受所有数字作为参数．
- `Integral` 也是一个算数类，但是只包含 `Int` 和 `Integer` 这两个整形．
- `Floating` 和 `Integral` 类似，只包含 `Float` 和 `Double` 这两个实数类型．