

Conditional Equations and Tuples

```
maxi(x, y) | x >= y      = x  
           | otherwise   = y
```

Currying

```
plus :: (Int, Int) -> Int  
plus (x, y) = x + y
```



```
plus :: Int -> Int -> Int  
plus x y = x + y
```

Pattern Declarations

```
pin :: Float  
pin = 3.14159
```

```
suc :: Int -> Int  
suc = plus 1
```

```
x0, y0 :: Int  
(x0, y0) = (1,2)
```

```
x1, y1 :: Int  
[x1,y1] = [1,2]
```

```
x2 :: Int  
y2 :: [Int]  
x2:y2 = [1,2]
```

HASKELL-declarations

decl → typedecl | fundecl

typedecl → var₁, ..., var_n :: type, $n \geq 1$

var → string starting with lower case symbol

fundecl → funlhs rhs

funlhs → var pat₁ ... pat_n, $n \geq 1$

rhs → = exp | condrhs₁ ... condrhs_n, $n \geq 1$

condrhs → | exp = exp

HASKELL-declarations

decl → typedekl | fundecl | patdecl

typedekl → var₁, ..., var_n :: type, $n \geq 1$

var → string starting with lower case symbol

fundecl → funlhs rhs [where decls]

funlhs → var pat₁ ... pat_n, $n \geq 1$

rhs → = exp | condrhs₁ ... condrhs_n, $n \geq 1$

condrhs → | exp = exp

patdecl → pat rhs [where decls]

decls → { decl₁; ... ; decl_n }, $n \geq 0$

HASKELL-declarations (2)

decl → typeddecl | fundecl | patdecl | infixdecl

infixdecl → $\left\{ \begin{array}{l} \text{infix} \\ \text{infixl} \\ \text{infixr} \end{array} \right\} \left[\left\{ \begin{array}{l} 0 \\ 1 \\ \vdots \\ 9 \end{array} \right\} \right] \underline{\text{op}}_1, \dots, \underline{\text{op}}_n, \quad n \geq 1$

op → varop | constrop

varop → string of special symbols, not starting with :

constrop → string of special symbols starting with :

HASKELL-expressions

exp → var
| constr
| integer
| float
| char
| [exp₁, ..., exp_n], $n \geq 0$
| string
| (exp₁, ..., exp_n), $n \geq 0$
| (exp₁ ... exp_n), $n \geq 2$
| **if** exp₁ **then** exp₂ **else** exp₃
| **let** decls **in** exp
| **case** exp **of** { pat₁ → exp₁; ...; pat_n → exp_n }, $n \geq 1$
| \ pat₁ ... pat_n → exp

constr → string starting with upper case symbol