ФУНКЦІЙНЕ ПРОГРАМУВАННЯ

ПАРАДИГМИ ПРОГРАМУВАННЯ

- Імперативна

модифікація станів

$$\sigma = \sigma_0 \to \sigma_1 \to \sigma_2 \to \cdots \to \sigma_n = \sigma'$$

шляхом *руйнівних* присвоювань типу *v:=Expr*

порядок виконання команд змінюється операторами типу *if, while* …

ПАРАДИГМИ ПРОГРАМУВАННЯ

- Функційна

  Заключний стан залежить тільки від параметрів фунції

$$\sigma' = f(\sigma)$$

  Без руйнівного присвоювання

  Без явного управління порядком виконання команд

  Програма – декларація + опис функцій

  Обчислення – застосування функції до параметрів

  Рекурсія

ВЛАСТИВОСТІ ФУНКЦІЙНИХ МОВ

- лаконічність і простота;
- сувора типізація + виведення типів;
- модульність;
- чистота (відсутність побічних ефектів);
- відкладені (ліниві) обчислення.

ЛАКОНІЧНІСТЬ І ПРОСТОТА

```
1.  void quickSort (int a[], int l, int r)
2.  {
3.          int i = l;
4.          int j = r;
5.          int x = a[(l + r) / 2];
6.          do
7.          {
8.                  while (a[i] < x) i++;
9.                  while (x < a[j]) j--;
10.                 if (i <= j)
11.                 {
12.                         int temp = a[i];
13.                         a[i++] = a[j];
14.                         a[j--] = temp;
15.                 }
16.         }
17.         while (i <= j);
18.         if (l < j) quickSort (a, l, j);
19.         if (i < r) quickSort (a, i, r);
20. }
```

Швидке сортування (C)

ЛАКОНІЧНІСТЬ І ПРОСТОТА

```
quickSort [ ] = [ ]
quickSort (h : t) =
                    quickSort [y | y <- t, y < h] ++
                    [h] ++
                    quickSort [y | y <- t, y >= h]
```

Швидке сортування

ЛАКОНІЧНІСТЬ І ПРОСТОТА

Зіставлення зі зразком

fib (0) = 1

fib (1) = 1

fib (N) = fib (N – 2) + fib (N – 1)

Число Фибоначчі

ЛАКОНІЧНІСТЬ І ПРОСТОТА

Зіставлення зі зразком

```
fib 0 = 1
fib 1 = 1
fib N = fib (N – 2) + fib (N – 1)
```

Число Фибоначчі

- Сувора типізація
  - Типізація (модель Хіндлі-Мілнера)
- Модульність
- Чистота
  - Відсутність побічних ефектів
  - Детермінованість
  - Розмежування обчислень та дій

ФУНДАМЕНТ ФП

- Алонзо Чорч (США), λ-числення
- Хаскелл Брукс Каррі (США), комбінаторна логіка
- Теорія категорій
- Теорія типів

Інструментарій

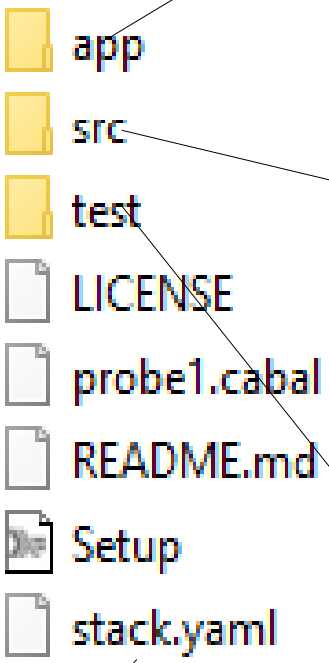
1. Glasgow Haskell Compiler.
   https://www.haskell.org/ghc/
   - ghc
   - ghci
   - Cabal
     (Common Architecture for Building Applications and Libraries; https://www.haskell.org/cabal/)
   - cabal-install

Інструментарій

2. Haskell Platform
   - Glasgow Haskell Compiler
   - Cabal build system
   - **Stack** tool for developing projects https://docs.haskellstack.org/en/stable/README/
   - support for profiling and code coverage analysis
   - core & widely-used packages ghc

Інструментарій

3. IDE + stack + Glasgow Haskell Compiler
   - Glasgow Haskell Compiler
   - Cabal build system
   - **stack** tool for developing projects
   - IDE = Visual Stidio Code + Haskero

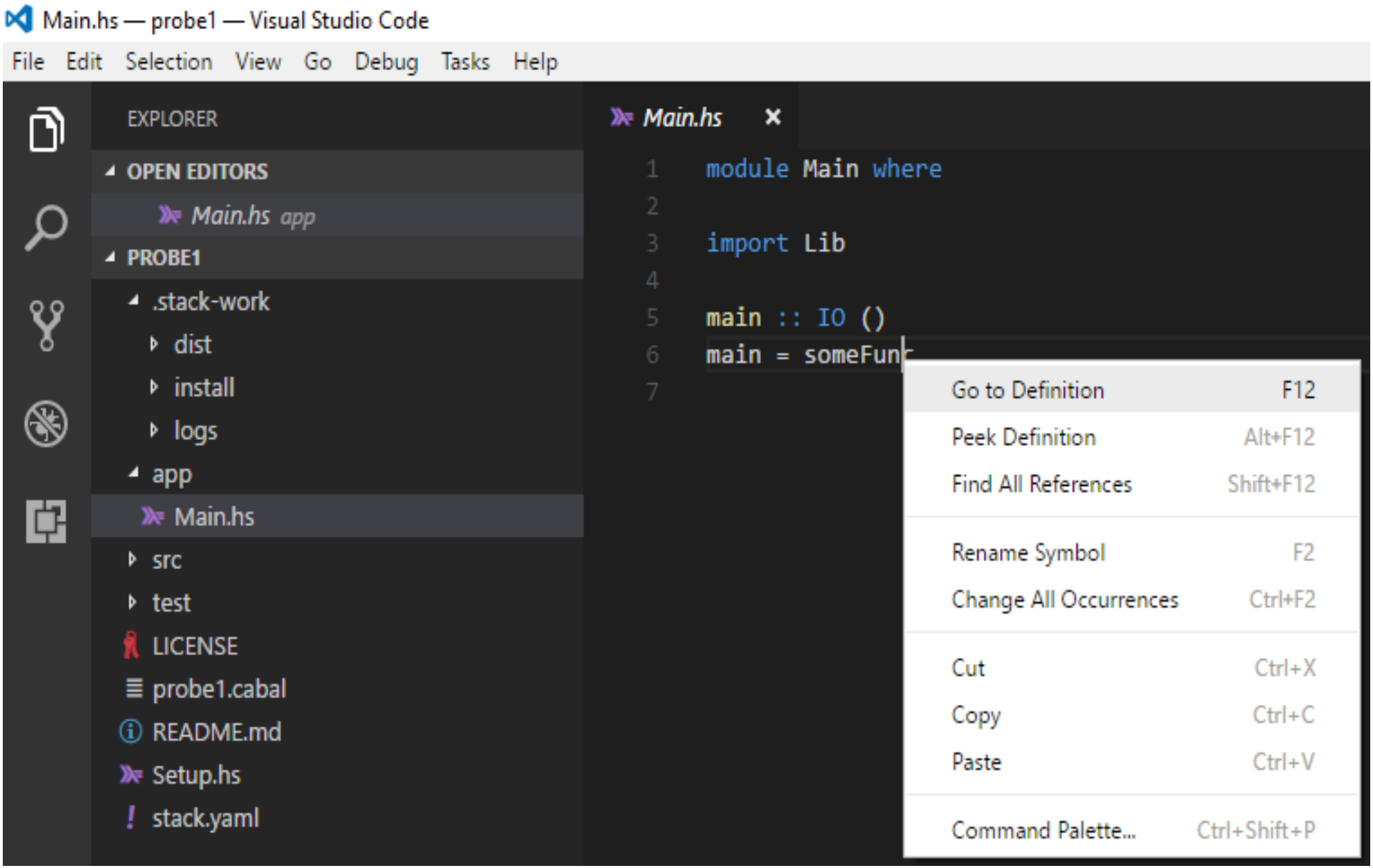


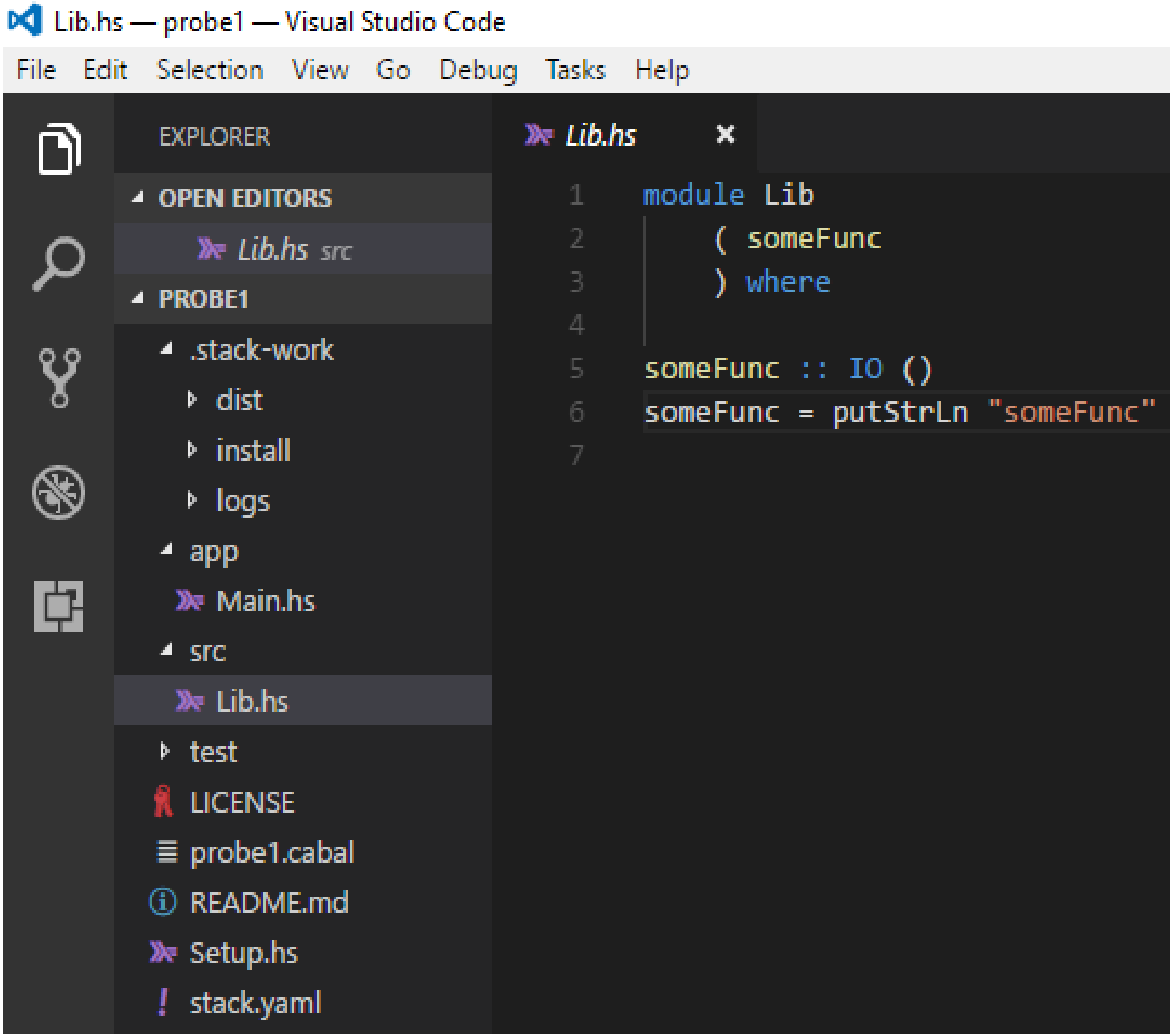

```
module Main where

import Lib

main :: IO ()
main = someFunc
```

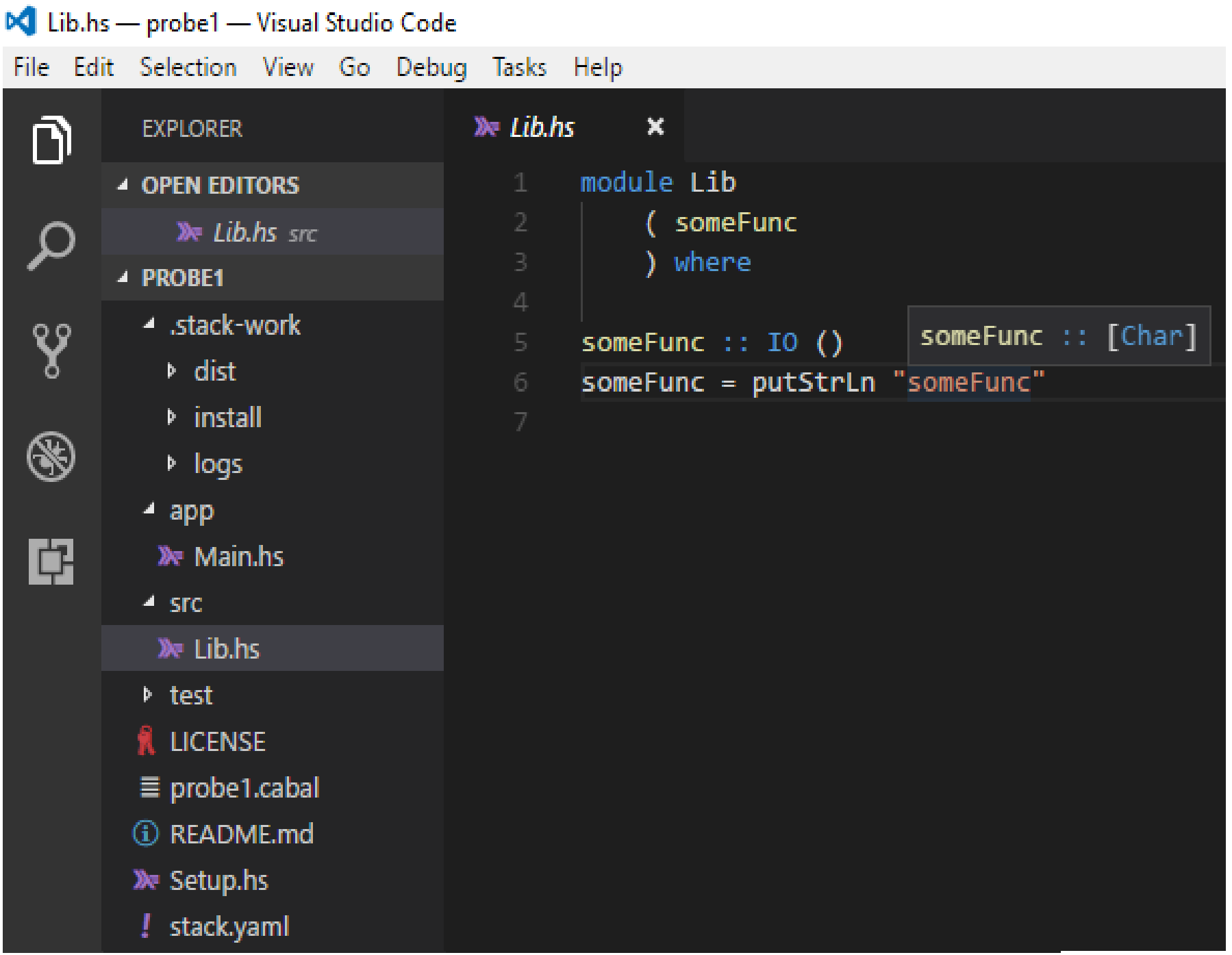
```
module Lib ( someFunc ) where

someFunc :: IO ()
someFunc = putStrLn "someFunc"
```

resolver: lts-9.3

```
main :: IO ()
main = putStrLn "Test suite not yet implemented"
```


Main.hs — probe1 — Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
Main.hs app
PROBE1
.stack-work
dist
install
logs
app
Main.hs
src
test
LICENSE
probe1.cabal
README.md
Setup.hs
stack.yaml
Main.hs
1 module Main where
2
3 import Lib
4
5 main :: IO ()
6 main = someFun
7
Go to Definition F12
Peek Definition Alt+F12
Find All References Shift+F12
Rename Symbol F2
Change All Occurrences Ctrl+F2
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Command Palette... Ctrl+Shift+P


Lib.hs — probe1 — Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
Lib.hs src
PROBE1
.stack-work
dist
install
logs
app
Main.hs
src
Lib.hs
test
LICENSE
probe1.cabal
README.md
Setup.hs
stack.yaml
Lib.hs
1 module Lib
2 (someFunc
3) where
4
5 someFunc :: IO ()
6 someFunc = putStrLn "someFunc"
7


Lib.hs — probe1 — Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
Lib.hs src
PROBE1
.stack-work
dist
install
logs
app
Main.hs
src
Lib.hs
test
LICENSE
probe1.cabal
README.md
Setup.hs
stack.yaml
Lib.hs
1 module Lib
2 (someFunc
3) where
4
5 someFunc :: IO ()
6 someFunc = putStrLn "someFunc"
7
someFunc :: [Char]