

# **Знайомство з функціональним програмуванням**

2019

ФП. Знайомство.

1

# Основні питання

---

- Знайомство з функціональним програмуванням:
  - мова функціональним програмування *Haskell*;
  - списки;
  - *list comprehension*.
- Визначення функцій рівняннями. Зіставлення зі зразком. Операційна семантика.
- Лінівність обчислень.



An advanced, purely functional programming language

Declarative, statically typed code.

```
primes = filterPrime [2..]
  where filterPrime (p:xs) =
        p : filterPrime [x | x <- xs, x `mod` p /= 0]
```

## Try it!

Type Haskell expressions in here.

```
λ take 9 (zipWith (+) [1..] [10,20..])
[11,22,33,44,55,66,77,88,99] :: (Enum a, Num a) => [a]
λ
```

## Got 5 minutes?

Type `help` to start the tutorial.

Or try typing these out and see what happens (click to insert):

```
23 * 36 or reverse "hello" or foldr (:) [] [1,2,3] or do line <- getLine;
putStrLn line or readFile "/welcome"
```

These IO actions are supported in this sandbox.

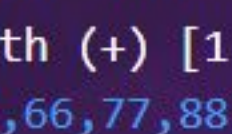
## Videos



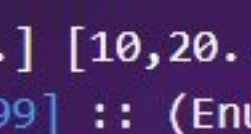
Escape from the ivory tower: The Haskell journey, by Simon Peyton-Jones



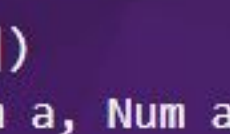
Haskell takes limiting side for parallel programming, by Ryan Newton



Lentzner



Other Tales, by Katie Miller



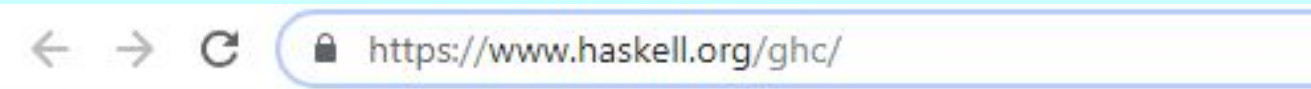
Spöck, by Oskar Wickström

## Try it!

Type Haskell expressions in here.

```
λ take 9 (zipWith (+) [1..] [10,20..])
[11,22,33,44,55,66,77,88,99] :: (Enum a, Num a) => [a]
λ
```

<https://www.haskell.org/ghc/>



## About GHC

- Home
- License
- Documentation
- Blog
- FAQ
- Download
- Report a bug
- Request a feature
- Developers (Wiki)
- The GHC Team

## About Haskell

- Haskell.Org
- The Haskell 2010 Report

## Latest News

- 7 December 2018**  
GHC 8.6.3 Released! [[download](#)]
- 2 November 2018**  
GHC 8.6.2 Released! [[download](#)]
- 14 October 2018**  
GHC 8.4.4 Released! [[download](#)]
- 21 September 2018**  
GHC 8.6.1 Released! [[download](#)]
- 29 May 2018**  
GHC 8.4.3 Released! [[download](#)]
- 19 Apr 2018**  
GHC 8.4.2 Released! [[download](#)]
- 8 Mar 2018**  
GHC 8.4.1 Released! [[download](#)]

# List comprehension. Функція швидкого сортування

Одна із спискових  
родзинок (синтаксич-  
ний цукор)

Генератор

Охорона

```
qsort [] = []  
qsort (x:xs) = qsort [y | y <- xs, y < x] ++ [x] ++ qsort [y | y <- xs, y >= x]
```

Функція швидкого  
сортування

Варто  
відзначити

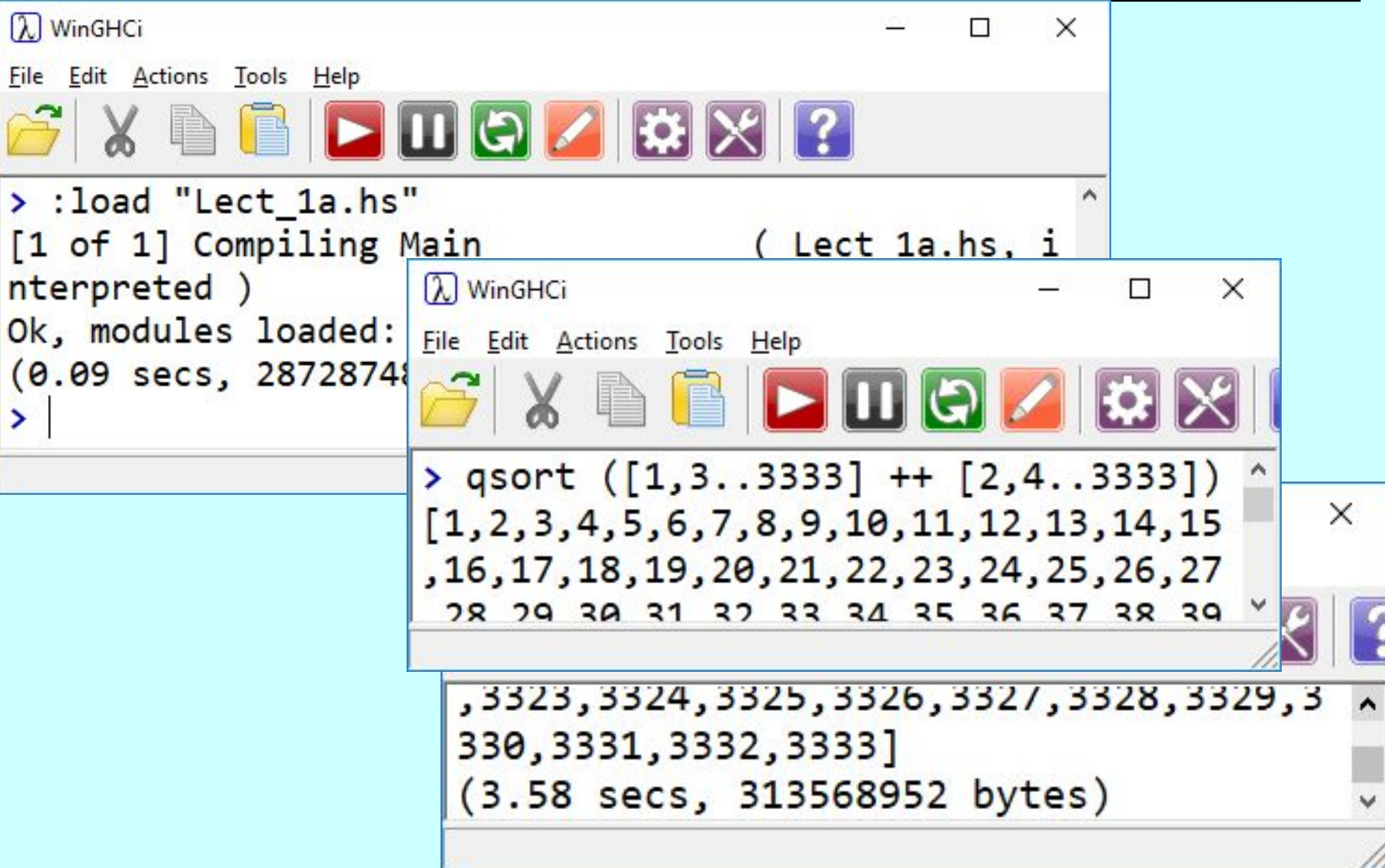
Функції визначаються  
рівняннями

Ключові терміни операційної семантики:

редукція та зіставлення зі зразком (*pattern matching*)

# The Glasgow Haskell Compiler.

## Функція швидкого сортування. Експериментуємо...



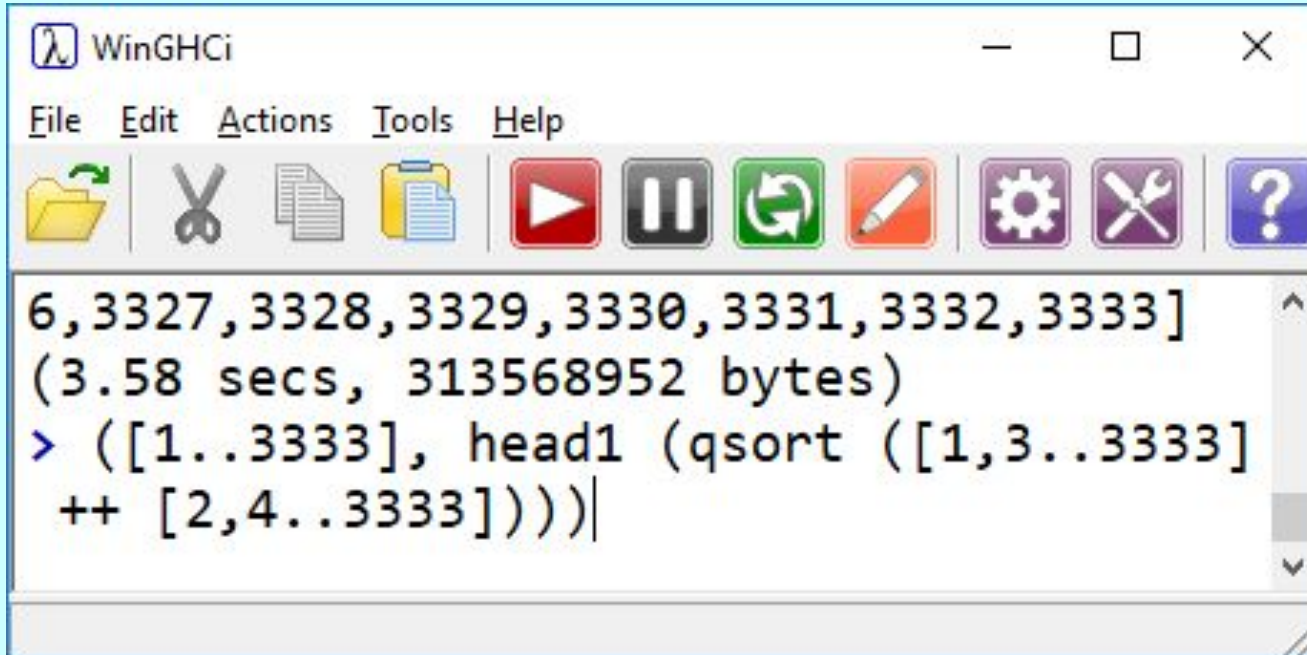
The image shows a screenshot of the WinGHCi terminal window. The window title is "WinGHCi". The menu bar includes "File", "Edit", "Actions", "Tools", and "Help". The toolbar contains icons for file operations (copy, paste, save), execution (run, pause, refresh), and settings (gear, wrench, question mark). The terminal text is as follows:

```
> :load "Lect_1a.hs"
[1 of 1] Compiling Main          ( Lect 1a.hs, i
nterpreted )
Ok, modules loaded:
(0.09 secs, 28728748 bytes)
> |
```

A second WinGHCi window is overlaid on the first, showing the execution of the `qsort` function:

```
> qsort ([1,3..3333] ++ [2,4..3333])
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
,16,17,18,19,20,21,22,23,24,25,26,27
28 29 30 31 32 33 34 35 36 37 38 39
, 3323, 3324, 3325, 3326, 3327, 3328, 3329, 3
330, 3331, 3332, 3333]
(3.58 secs, 313568952 bytes)
```

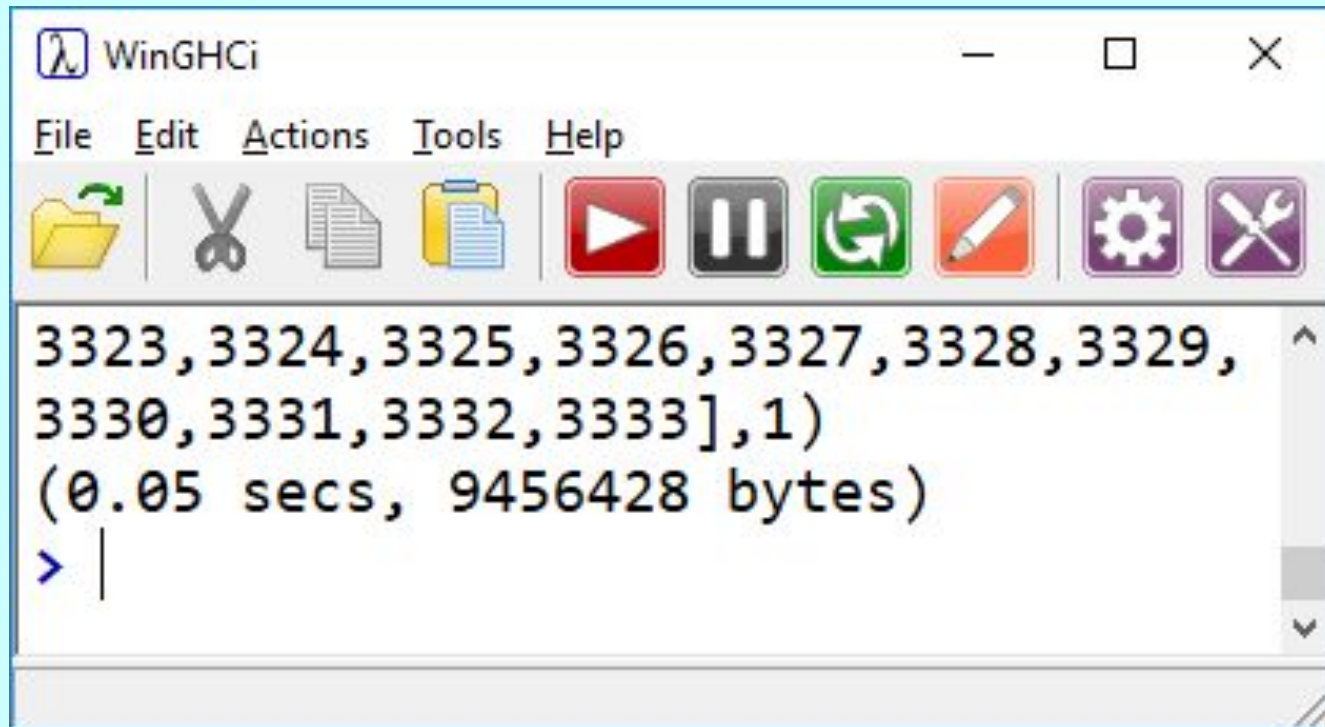
# Лінивість обчислень (1/2)



The screenshot shows the WinGHCi window with the following content:

```
6, 3327, 3328, 3329, 3330, 3331, 3332, 3333 ]  
(3.58 secs, 313568952 bytes)  
> ([1..3333], head1 (qsort ([1,3..3333]  
++ [2,4..3333])))|
```

# Лінивість обчислень (2/2)



The screenshot shows the WinGHCi window with a menu bar (File, Edit, Actions, Tools, Help) and a toolbar with icons for file operations and execution. The main text area displays the following output:

```
3323, 3324, 3325, 3326, 3327, 3328, 3329,  
3330, 3331, 3332, 3333], 1)  
(0.05 secs, 9456428 bytes)  
> |
```



# Функція швидкого сортування. Експериментуємо...

```
C:\Program Files (x86)\Hugs98_SOE\hugs.exe
Main> qsort [333,332..1]
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,
38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,7
2,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,
105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,130
,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,15
6,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,1
82,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,
208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,229,230,231,232,233
,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,25
9,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281,282,283,284,2
85,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,
311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,329,330,331,332,333]
(1004722 reductions, 1343326 cells, 2 garbage collections)
Main> ([333,332..1], head1 (qsort [333,332..1]))
([333,332,331,330,329,328,327,326,325,324,323,322,321,320,319,318,317,316,315,314,313,312,311,310,309,3
08,307,306,305,304,303,302,301,300,299,298,297,296,295,294,293,292,291,290,289,288,287,286,285,284,283,
282,281,280,279,278,277,276,275,274,273,272,271,270,269,268,267,266,265,264,263,262,261,260,259,258,257
,256,255,254,253,252,251,250,249,248,247,246,245,244,243,242,241,240,239,238,237,236,235,234,233,232,23
1,230,229,228,227,226,225,224,223,222,221,220,219,218,217,216,215,214,213,212,211,210,209,208,207,206,2
05,204,203,202,201,200,199,198,197,196,195,194,193,192,191,190,189,188,187,186,185,184,183,182,181,180,
179,178,177,176,175,174,173,172,171,170,169,168,167,166,165,164,163,162,161,160,159,158,157,156,155,154
,153,152,151,150,149,148,147,146,145,144,143,142,141,140,139,138,137,136,135,134,133,132,131,130,129,12
8,127,126,125,124,123,122,121,120,119,118,117,116,115,114,113,112,111,110,109,108,107,106,105,104,103,1
02,101,100,99,98,97,96,95,94,93,92,91,90,89,88,87,86,85,84,83,82,81,80,79,78,77,76,75,74,73,72,71,70,69
,68,67,66,65,64,63,62,61,60,59,58,57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,35,
34,33,32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1],1)
(400040 reductions, 575825 cells)
Main>
```



# *List comprehension.* Декартовий добуток

```
dekart xs ys = [(x,y) | x <- xs, y <- ys]
```

Два генератори

```
> dekart [1,2,3] ['a', 'b', 'c']  
[(1, 'a'), (1, 'b'), (1, 'c'), (2, 'a'), (2, 'b'), (2, 'c'), (3, 'a'), (3, 'b'), (3, 'c')]
```

## ***List comprehension:***

- генератори;
- охорони;
- локальні визначення (конструкції ***"let ... in..."*** та ***"where ..."***).

# Список простиx чисел. (1/3)

Функція, що генерує (нескінченний!) список простиx чисел

```
allPrimeNumbers_v0 = [n | n <- [2..], isPrimeLoc n == True]
  where isPrimeLoc n | n <= 1      = False
                  | otherwise    = getDivisorsLoc n == [1,n]
  getDivisorsLoc n
    | n < 1 = []
    | otherwise = [x | x <- [1..n], mod n x == 0 ]
```

```
> otherwise
True
```

## *List comprehension:*

- генератори;
- охорони;
- локальні визначення (конструкції **"let ... in..."** та **"where ..."**).

# Список простых чисел. (2/3)

## Операторна форма бінарних функцій

Функція обчислення (списку) дільників

```
getDivisors num  
| num < 1           = []  
| otherwise        = [x | x <- [1..num], num `mod` x == 0 ]
```

Бінарна функція *mod* подана в **операторній** формі (із використанням зворотних апострофів)

**Операторна форма**  
бінарної функції (*mod*)

Функція перевірки числа на простоту

```
isPrime num  
| num <= 1         = False  
| otherwise        = getDivisors num == [1,num]
```

Функція, що генерує (нескінченний!) список простих чисел

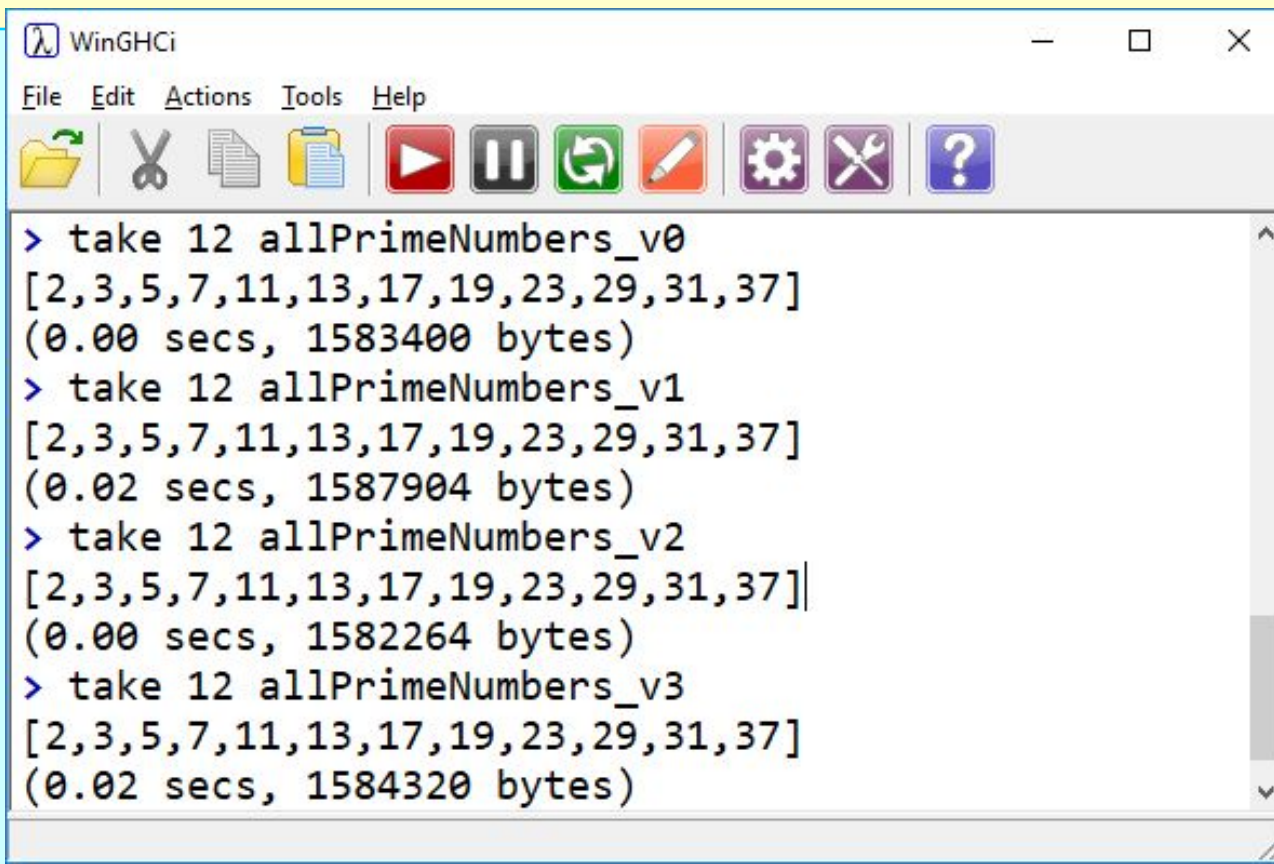
```
allPrimeNumbers_v1 = [2] ++ [n | n <- [3,5..], isPrime n == True]
```

# Список простых чисел. (3/3)

## Ще дві версії

```
sieve1 (x : xs) = x : (sieve1 ys)
  where ys = [n | n <- xs, n `mod` x /= 0]
allPrimeNumbers_v2 = sieve1 [2..]
```

```
sieve2 (x : xs) = let ys = [n | n <- xs, n `mod` x /= 0] in
  (x : (sieve2 ys))
allPrimeNumbers_v3 = sieve2 [2..]
```



The screenshot shows a terminal window titled "WinGHCi" with a menu bar (File, Edit, Actions, Tools, Help) and a toolbar with various icons. The terminal output shows the execution of four commands to generate the first 12 prime numbers using different sieve implementations. Each command returns the same list of primes: [2,3,5,7,11,13,17,19,23,29,31,37]. The execution times and memory usage are also displayed for each command.

```
> take 12 allPrimeNumbers_v0
[2,3,5,7,11,13,17,19,23,29,31,37]
(0.00 secs, 1583400 bytes)
> take 12 allPrimeNumbers_v1
[2,3,5,7,11,13,17,19,23,29,31,37]
(0.02 secs, 1587904 bytes)
> take 12 allPrimeNumbers_v2
[2,3,5,7,11,13,17,19,23,29,31,37]
(0.00 secs, 1582264 bytes)
> take 12 allPrimeNumbers_v3
[2,3,5,7,11,13,17,19,23,29,31,37]
(0.02 secs, 1584320 bytes)
```

# Парадигма функціонального програмування

- Опис (визначення) функцій
- Обчислення функцій

Не імперативний стиль із незмінюваними даними (не використовується поняття послідовного виконання, немає змінних, немає присвоювань)

Чистота функцій (відсутність побічних ефектів): два обчислення функції з однаковими аргументами завжди дають один і той же самий результат!

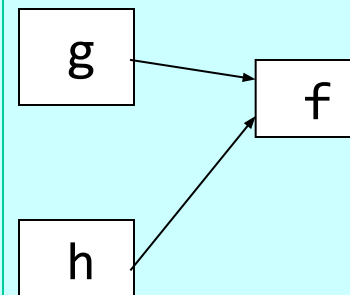
Повернення до **ЧИСТИХ** функцій.

- Машинні бібліотеки функцій.
- *Algol-60*: функції із **побічними ефектами** (червоність).

Що дає функціональна парадигма?

- Можливість заміни виразу на результат обчислення цього виразу (не потрібні переобчислення).
- Спрощене тестування.
- Природні підходи до розпаралелювання.
- Спрощена інтеграція (інтеграція – один з найбільших ризиків в інженерії програмних систем).

ФП. Знайомство.



# Функціональне програмування. Теоретичні засади

---

- Формалізація семантики функціональних програм фактично спряжена з *уточненням власне функцій* та *уточненням аплікації* (уточненням застосування функції до аргументів).

Теоретичні засади такої формалізації давно відомі. Це – **лямбда-числення**.

Окремі штрихи теоретичного підґрунтя :

- чисте лямбда-числення – це числення **анонімних** функцій;
  - **бета-редукція** – основа трактування аплікації (тобто є засадою **операційної семантики**);
  - можливі різні **стратегії** використання бета-редукції, зокрема, є стратегія, спряжена із так званими **лінивими** обчисленнями, є стратегія, спряжена із так званими **енергійними** обчисленнями;
  - теорема про нерухому точку (для визначення функцій, що задаються рекурсивними рівняннями).
- Можна обмежитись **одноаргументними** функціями, спираючись на **каррінг** функцій.

На *Haskell* реалізовано багато складних проектів. Ось деякий їх перелік за джерелом

[http://www.ibm.com/developerworks/ru/library/1-haskell/?S\\_TACT=105AGX99&S\\_CMP=GR01](http://www.ibm.com/developerworks/ru/library/1-haskell/?S_TACT=105AGX99&S_CMP=GR01)

- Компілятори й інші засоби розробки.
- Розподілена система керування версіями *Darcs*.
- Віконний менеджер *xmonad*.
- Сервер *Web*-додатків *HAppS*.
- Інтерпретатор/компілятор *Pugs* для мови *Perl 6*.
- Операційна система *House*.
- Мова опису апаратних засобів *Lava*.
- Система обробки природної мови *LOLITA*.
- Системи доведення теорем *Equinox / Paradox* і *Agda*.

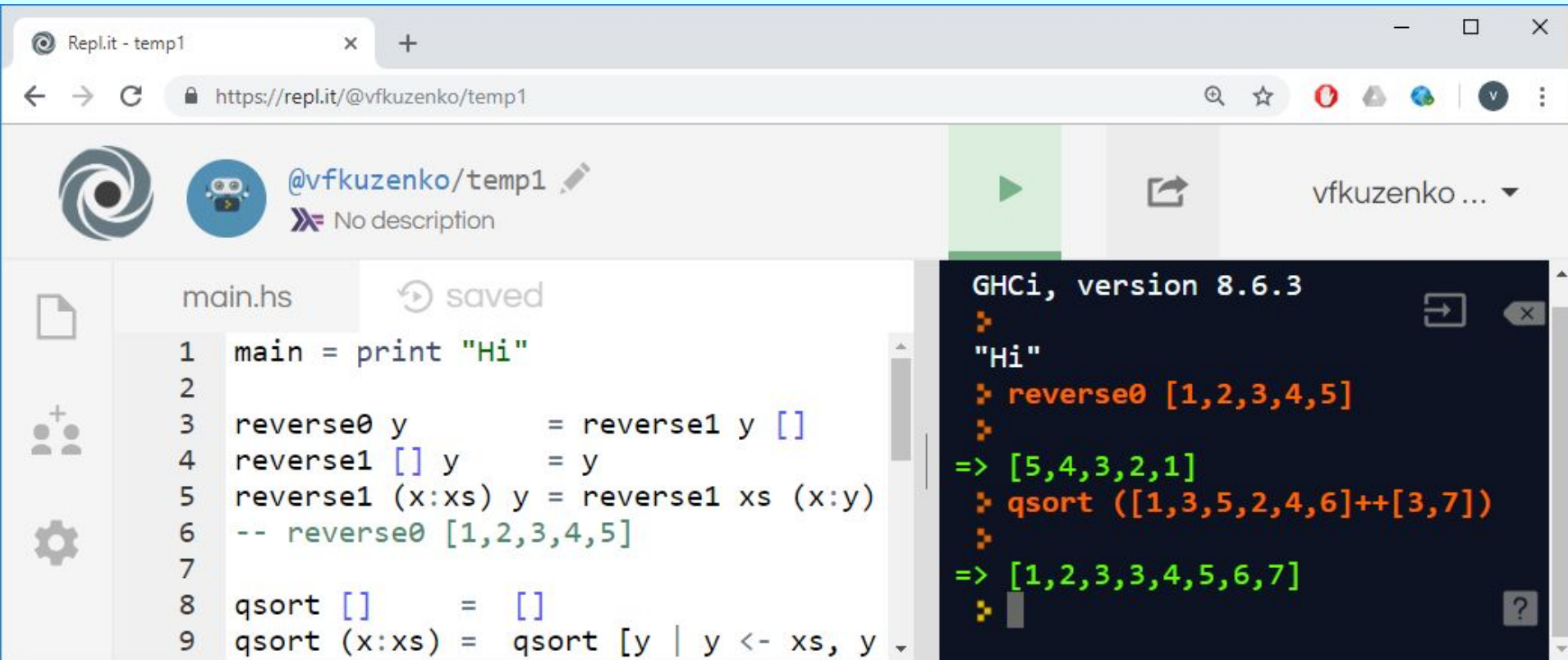


# Література

---

- 1) Душкин Р. В. Функциональное программирование на языке Haskell. М.: ДМК Пресс, 2007.
- 2) Душкин Р. В. Справочник по языку Haskell. М.: ДМК Пресс, 2008.
- 3) Душкин Р. В. Практика работы на языке Haskell. М.: ДМК Пресс, 2009.
- 4) Липовача М. Изучай Haskell во имя добра! М.: ДМК Пресс, 2012.
- 5) Lipovača M. Learn You a Haskell for Great Good! Miran Lipovača. : No Starch Press», 2011.
- 6) Роганова Н. А. Функциональное программирование, 2002.
- 7) Филд А., Харрисон П. Функциональное программирование. М.: Мир, 1993.
- 8) Хендерсон П. Функциональное программирование. Применение и реализация. М.: Мир, 1983.
- 9) Хьюдак П., Петерсон Дж., Джозеф Фасел Дж. Мягкое введение в haskell. Учебник.

# https://repl.it



The screenshot shows a web browser window with the URL `https://repl.it/@vfkuzenko/temp1`. The page header includes the Repl.it logo, the user profile `@vfkuzenko/temp1` with a "No description" note, and a play button. The main content area is split into two panes. The left pane shows a code editor with a file named `main.hs` containing the following Haskell code:

```
1 main = print "Hi"
2
3 reverse0 y      = reverse1 y []
4 reverse1 [] y   = y
5 reverse1 (x:xs) y = reverse1 xs (x:y)
6 -- reverse0 [1,2,3,4,5]
7
8 qsort []        = []
9 qsort (x:xs) = qsort [y | y <- xs, y < x]
```

The right pane shows the GHCi terminal output for version 8.6.3:

```
GHCi, version 8.6.3
>
"Hi"
> reverse0 [1,2,3,4,5]
=> [5,4,3,2,1]
> qsort ([1,3,5,2,4,6]++[3,7])
=> [1,2,3,3,4,5,6,7]
>
```