



Визначення нової функції. λ -обчислення

Лекція 5

Теоретична основ обчислень у функціональному програмуванні


Автор λ -числення - американський математик Алонзо Чорч.

λ -числення - формальна система, розроблена для формалізації і аналізу поняття обчислюваності.

А. Чорч на основі двох логічних операторів: аплікації та абстракції збудував λ -числення, яке мало прислужитися для серйозного дослідження основ математики.

Стівен Коул Кліні довів, що λ -числення суперечливе. Подальші розробки теорії оператора лямбда належать Хаскеллу Каррі.


Попри суперечливість, λ -числення знайшло практичне застосування, полягши в основу функціональних мов програмування.



Аплікація – це побудова нових обчислень на основі комбінації інших обчислень.

Функціональна абстракція відділяє спосіб використання функції і деталі її реалізації в термінах більш примітивних функцій.

λ-числення може розглядатися як сімейство прототипних мов програмування. Їх основна особливість полягає в тому, що вони є мовами вищих порядків, тобто функції/оператори мають аргументи і значення у вигляді інших функцій/операторів.



Визначення функцій та їх обчислення в функціональних мовах ґрунтується на ***λ-численні*** Чорча.


λ-вираз є важливим механізмом у програмуванні, який полягає у простому та точному формалізмі при побудові нових функцій.

В λ-численні Чорча функція записується у вигляді:

$$\text{lambda } (x_1, x_2, \dots, x_n) . f$$

В LISP λ-вираз має вигляд:


$$(\mathbf{LAMBDA} (x_1 \ x_2 \ \dots \ x_n) f)$$



Символ LAMBDA говорить нам про визначення функції.

Символи x_i – це формальні параметри, f – тіло функції.


Тілом функції може бути довільна S-вираз, значення якого може обчислити інтерпретатор LISP.




Функцію, яка обчислює суму квадратів двох чисел, можна визначити так:

$(\text{LAMBDA } (x \ y) (+ (* x \ x) (* y \ y)))$.

Тут визначено правило – як обчислювати суму квадратів двох чисел. Самі обчислення не задано.



Формальність параметрів вказує на те, що ми можемо замінити їх на інші символи, але від цього не зміниться суть обчислення функції.



λ вираз – це визначення обчислення та параметрів функції в чистому вигляді без фактичних параметрів або аргументів.

Для застосування такої функції до певних аргументів, необхідно поставити λ -вираз на місце імені функції:

(λ -вираз $a_1 a_2 \dots a_n$).

Тут a_i – форми, що задають фактичні параметри.




Наприклад, множення $(*\ 3\ 4)$ можна записати з використанням лямбда виклику:

$((\text{LAMBDA}\ (x\ y)\ (*\ x\ y))\ 3\ 4)$



```
((LAMBDA (x y) (CONS x (CONS y '())))  
  'dog 'cat)
```




Таку форму виклику – застосування λ -виразу до фактичних аргументів називають λ -викликом.

Обчислення λ -виклику відбувається в два етапи.


- Спочатку обчислюються значення фактичних параметрів та відповідні формальні параметри зв'язуються з отриманими значеннями.
- На другому етапі обчислюється форма, яка є тілом λ -виразу. Отримане значення повертається в якості значення λ -виклику. По завершенню обчислення формальним параметрам повертаються зв'язки, які існували до λ -виклику.

Весь цей процес називається λ -перетворенням




λ- вираз без фактичних параметрів є лише визначення, а не форма, яку можна обчислити.

Сам λ- вираз інтерпретатором не сприймається. Якщо ви введете:
(LAMBDA (x y) (CONS x (CONS y NIL))),
то інтерпретатор видасть повідомлення про помилку.



λ-вираз є як чисто абстрактним механізмом для визначення та опису обчислення, так і механізмом для зв'язування формальних та фактичних параметрів під час виконання обчислення.

λ-вираз є функцією без імені.




λ-вираз використовується в мовах програмування Python, C#, F#, Visual Basic .NET для оголошення функцій в довільному місці програми.

Визначення нової функції

У функціональному програмуванні нові функції створюються з допомогою базових на основі принципів композиції та рекурсії.

Композиція – це правило створення нової функції, коли одна функція виступає аргументом для іншої.

Наприклад, (CDR (CAR '((a d g) r t y))).




Для задання нової функції програмістом мовою Scheme використовується конструкція визначення функції DEFINE, формат використання якої:

```
(DEFINE <name> (LAMBDA (<arguments_fun>)  
  (<body_fun>))),
```

де <name> - ідентифікатор функції;

<arguments_fun> - список аргументів (через пробіл);

<body_fun> - правильний S-вираз, який набуває значення.



Функція `DEFINE` з'єднує символи імені з λ -виразом, після чого символи починають іменувати обчислення, яке визначається λ -виразом.

Значенням функції `DEFINE` є ім'я нової функції



Приклад визначення функції