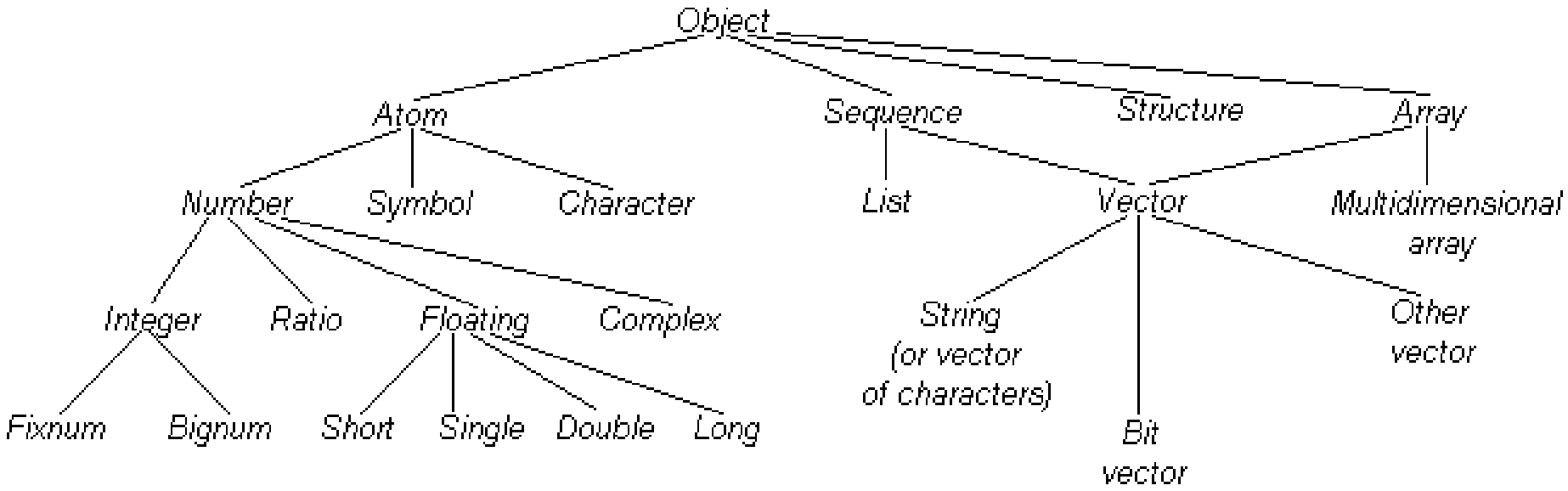
Common Lisp

Типы данных

- В CL тип данных – это множество (возможно бесконечное) объектов языка Lisp. ***Многие*** Lisp-объекты могут принадлежать ***нескольким таким*** **множествам**, и поэтому вопрос о типе объекта не всегда имеет смысл. Вместо этого обычно возникает вопрос о том, **принадлежит ли объект данному типу**.

Типы данных


Object
Atom
Sequence
Structure
Array
Number
Symbol
Character
List
Vector
Multidimensional array
Integer
Ratio
Floating
Complex
String (or vector of characters)
Other vector
Fixnum
Bignum
Short
Single
Double
Long
Bit vector

Типы данных

- В CL типизированы именно объекты данных, а не переменные.

- ***Атом*** – это множество простейших типов данных, объекты которых являются неделимыми. Это множество включает: числа, знаки (элементы строк, символы) и символы.
- ***Символ*** (тип данных) – это специальный тип данных, который используется для обозначения других объектов: переменных, функций, структур.

Вычисления

- Форма – это выражение, которое может корректно вычислено интерпретатором языка:
- – самовычисляющиеся формы (числа, строки, знаки, массивы)
- – символы (если с ними связаны формы)
- – списки.

Вычисления

- Формы, представленные списками, разделяются на три категории:
- – специальные операторы;
- – вызовы функций;
- – вызовы макросов.

Лямбда-выражение

- (lambda list-of-arguments function-body)

- (lambda (x y) (+ x y));*лямбда выражение*
- ((lambda (x y) (+ x y)) 2 3) => 5

Вычисления

- (**defun** function-name lambda-list form)

Лямбда-список

- `lambda-list::=(var* [&optional {var | (var [init-form [supplied]])}*] [&rest var] [&key {var* | ({var | (keyword var)} [init-form [supplied]])}*] [&aux {var* | (var [init-form])}*])`

Корни квадратного уравнения

```
((lambda (a b c)
     (list
          (/ (+ (- b)
                (sqrt (- (* 4 a c) (* b b)))
                )
             (* 2 a)
          )
          (/ (- (- b)
                (sqrt (- (* 4 a c) (* b b)))
                )
             (* 2 a)
          )
     )
) 1 -2 3)
```

Корни квадратного уравнения

```
((lambda (a b c)
     ((lambda (d e f)
          (list
               (/ (+ f d) e)
               (/ (- f d) e)
          )
      ) (sqrt (- (* 4 a c) (* b b)))
        (* 2 a)
        (- b)
     )) 2 -9 18)
```

Опциональные аргументы

- `(defun fopt(x &optional (y (+ x x))) (list x y))`
- `=> fopt`
- `(fopt 2) => (2 4)`
- `(fopt 2 5) => (2 5)`

Оставшийся(иеся) аргументы

- `(defun frest(x &rest y )(list x y)) => frest`
- `(frest 'a) => (A NIL)`
- `(frest 'a 'b 'c) => (a (b c))`

Ключевые аргументы

- **(defun fkey (&key x y (z 100)) (list x y z))=> fkey**
- **(fkey) => (NIL NIL 100)**
- **(fkey :z 300 :y 200) => (NIL 200 300)**
- **(defun fkey1(&key (x 10) (y 20) (z 100)) (+ x y z))=> fkey**
- **(fkey1) =>**
- **(fkey1 :z 1 :x 1 :y 1) =>**

Вспомогательные аргументы

```
(defun sroot(a b c &aux (d (sqrt (-
  (* b b) (* 4 a c)))))
    (list (/ (- (- b) d) (* 2 a)))
          (/ (+ (- b) d) (* 2 a)))
    )
 )

(sroot 1 -2 1) => (1 1)
```