# Contents



# 1 Le basi

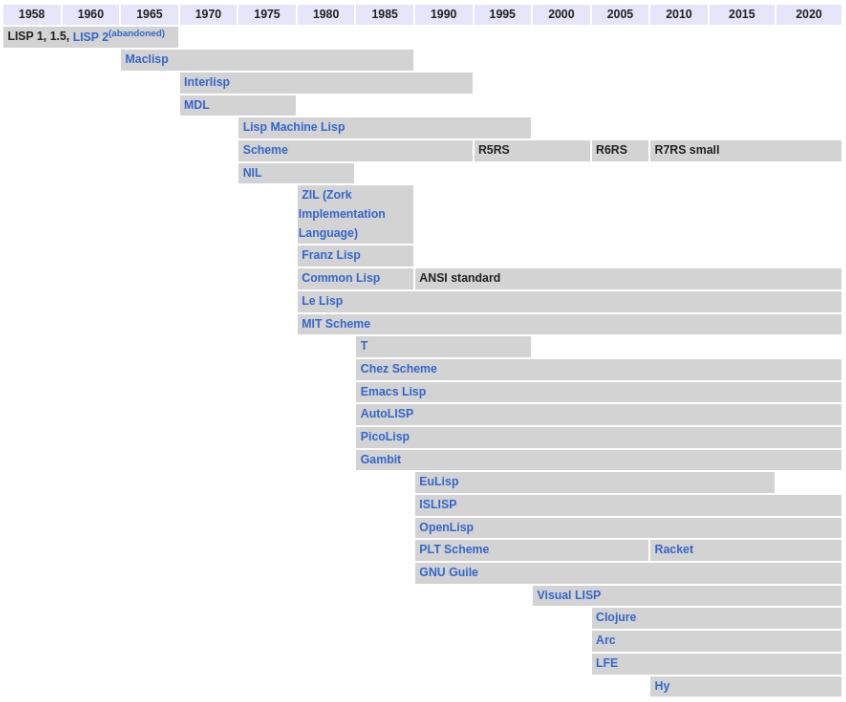
Lisp è un linguaggio di programmazione caratterizzato, tra i molti altri, da due fatti

- è stato fatto negli anni 60
- è stato fatto (tra gli altri) da matematici[1]

è abbastanza uno scempio per chi non ha passato anni ed anni a sviluppare varii strati di sindrome di stoccolma verso di esso, quindi è comprensibile che possa farvi schifo

per vostra sfortuna lisp non è un singolo linguaggio, ma è una famiglia di linguaggi, tutti basati sulle stesse 2/3 idee di base, ma ci hanno fatti tutti il cazzo che volevano giusto per dire, ecco da wikipedia la timeline dei lisp

[1] che l'informatica non esisteva troppo all'epoca come disciplina




Lo schifo che intendo fare sarà più uno scheme fatto male con un paio di cose fottute da common lisp. Scheme perchè scheme è quello più facile da approssimare (lo standard del linguaggio sono sulle 80 pagine, e la gente (inclusi gli autori dello standard) si fa seghe su quanto sia corto lo standard)

## 1.1 Ma sto lisp com'è fatto?

in lisp, quasi tutto è una lista, una lista è fatta così

```
( roba roba roba )
```

con quasi tutto si intende che anche il codice è una lista, ad esempio, un `if` è una lista messa così

```
(if <condizione> <qua il then> <qua l'else>)
```

ai fini di renderci la vita ancora peggiore, `<condizione>` non viene testato se è vero o falso, ma se è `nil`[2] o meno (noto anche come "non nil")

e una chiamata a funzione è una lista messa così

[2] nil sarebbe sia il `false` che il `null` del lisp

```
(<funzione> <argomento>*)
```

con funzione si intende anche roba stra builtin come `+`, `=`, `/`, ..., ad esempio per sommare due(o più) numeri si fa

```
(+ 2 3)
(+ 2 3 4 5)
...
```

se vogliamo controllare se `x == y` si fa[3]

```
(= x y)
```

o per dire `square(x)`, si fa

```
(square x)
```

mettiamo di voler dire "se x è dispari $print(x^2)$, altrimeti $print(\frac{x}{2})$" questo si tradurrebbe con

```
(if (= 0 (mod x 2))
    (print (/ x 2))
  (print (* x x)))
```

## 1.2 Qualche cagata da tenere a mente

TBD

## 1.3 Perchè lisp?

Due motivi

- mi piace il linguaggio[4]
- è una cagata farlo[5]

[3] all'epoca == non era ancora diventato il simbolo del "sono uguali?", i primi standard di questo affare, essendo anni 60, sono antecedenti al C, che era più anni 70

[4] comunque versioni un po' più moderne di quella fatta qui, con cose bellissime quali struct, classi, e ambienti di sviluppo decenti

[5] rispetto ad altri interpreter, poi grazialcazzo che fare un interpreter è comunque un dito in culo

### 1.3.1 Parsing

Fare parsing di lisp è una minchiata, la sintassi è all'incirca (si avvisa il lettore che questo non è mai nella vita un ebnf valido, dovrebbe solo rendere l'idea)

```
lettera :: [a-z] | [A-Z]
cifra :: [0-9]
simbolo :: (<qualsiasi carattere che non sia uno spazio>)*
stringa :: " <qualsiasi carttere>* "
numero :: <cifra>*
espressione :: simbolo | stringa | numero | ( <espressione>* )
```

e certi manuali[6] iniziano, giusto per, definendo l'intera sintassi ed evaluator del linguaggio, perchè gli andava[7]

### 1.3.2 Evaluator

le regole di valutazione del lisp sono anch'esse abbastanza una cagata da descrivere, ci vorrano sui 2/3 check neanche per capire se un'espressione è un if, un loop, o un che, e da lì mezzo secondo e l'hai già scomposta, ez.

[6] vecchi quanto la merda ma comunque

[7] e perchè il manuale l'aveva scritto il primo autore del linguaggio[8]

[8] più dello standard/definizione, il codice l'hanno fatto vari suoi studenti che programmavano decisamente meglio di lui, lui insegnava elettronica credo