Calculo Lambda
Numerais de Charch
1 ≡ λsλz.s(z)
2 ≡ λsλz.s(s(z))
3 ≡ λsλz.s(s(s(z)))
…

SUC = λwλyλx.y(wyx)
Por exemplo o sucessor de zero pode ser obtido aplicando a função SU C na representção de zero:
(λwλyλx.y(wyx))(λsλz.z)
→ λyλx.y((λsλz.z)yx)
→ λyλx.y((λz.z)x)
→ λyλx.y(x)

ADD = λxλyλwλu.xw(ywu)
Como exemplo podemos aplicar a função de adição para os números naturais 2 e 3:
(λxλyλwλu.xw(ywu))(λsλz.s(s(z)))(λsλz.s(s(s(z))))
→ (λyλwλu.(λsλz.s(s(z)))w(ywu))(λsλz.s(s(s(z))))
→ (λyλwλu.(λz.w(w(z)))(ywu))(λsλz.s(s(s(z))))
→ (λyλwλu.w(w(ywu)))(λsλz.s(s(s(z))))
→ λwλu.w(w((λsλz.s(s(s(z))))wu))
→ λwλu.w(w((λz.w(w(w(z))))u))
→ λwλu.w(w(w(w(u))))

Booleanos
V ≡ λxλy.x
F ≡ λxλy.y

Lógica
E = λx.λy.xy(λuλv.v)
OU = λxλy.(x(λuλv.u))y
N AO = λx.(x(λuλv.v))(λa.λb.a)

Lambda em HASKELL
```
zero::(a -> a) -> a -> a
zero = \s z -> z
um::(a -> a) -> a -> a
um = \s z -> s z
dois = \s z -> s (s z)
tres = \s z -> s (s (s z))
quatro = \s z -> s (s (s (s z)))


suc = \w y x -> y (w y x)
prede = \n f x -> n (\g h -> h (g f)) (\u -> x) (\u -> u)
add = \x y w u -> x w (y w u)
mul = \x y w u -> x (y w) u
```

Arvore Binária
```
data Arvore a = No a (Arvore a) (Arvore a) | Folha
deriving Show

--Insere elemento na Arvore
ins a Folha = No a Folha Folha
ins a (No x esq dir) | a == x = (No x esq dir)
                     | a < x = (No x (ins a esq) dir)
                     | otherwise = (No x esq (ins a dir))
--listToTree
listToTree ls = listToTree' Folha ls
listToTree' arv [] = arv
listToTree' arv (x:xs) = listToTree' (ins x arv) xs

--treeToList
treeToList Folha = []
treeToList (No x esq dir) = (treeToList esq) ++ [x] ++ (treeToList dir)
```