Recursión

Recursión

Cristián G. Sánchez y Carlos J. Ruestes

2020

Programación funcional

Programación funcional

La programación funcional es un *paradigma de programación* donde los programas se sonstruyen por medio de la aplicación y composición de funciones. En la programación funcional estricta las funciones son *funciones puras* y no pueden tener efectos colaterales. Si bien representa un paradigma "elegante" (desde un punto de vista muy Platónico quizás), los programas funcionales son menos eficientes (en tiempo y memoria) que los programas imperativos. La programación funcional es importante en la teoría de la computación ya que permite probar ciertos teoremas muy generales sobre complejidad algorítmica.

A pesar de lo anterior la programación funcional resulta un ejercicio importante que puede ayudarnos en nuestra construcción del pensamiento algorítmico.

Composición

Composición

La definición de funciones en términos de composición de otras nos es intuitiva desde nuestros conocimientos de análisis: $h(x) = f(g(x))$. Como en el caso de los ejercicios de operadores lógicos el uso de composición de funciones nos permite descomponer un problema complejo en "ingredientes" más simples.

Recursión

A diferencia de la definición en base a composición la *recursión* (y su prima cercana la *inducción*) son menos usadas y por lo tanto no tan familiares. Por ejemplo, la definición recursiva de la factorial parte del caso base:

$$0! = 1$$

y de allí se define para todo $n > 0$ de la forma

$$n! = (n-1)!$$

Recursión

El código python para definir la factorial en forma recursiva es:

```
def fact(n):
    if n == 0: # caso base
        return 1
    else: # para todos los otros n
        return n*fact(n-1)
```

Notar que dentro de la definición de `fact` tenemos una invocación a `fact`, esto convierte la definición en recursiva. La recursión converge porque la invocación tiene como argumento `n-1`

Síntesis y recursos:

- Recursión en wikipedia
- Excepciones, utilizadas en el demo.