

PROYECTO 2

DEFINICIONES RECURSIVAS

Matemáticas Computacionales
Dr. Victor de la Cueva

Hecho por:
Rafael Correa A01019498

14/06/2016

Manual de Usuario

Crear un archivo con un formato específico:

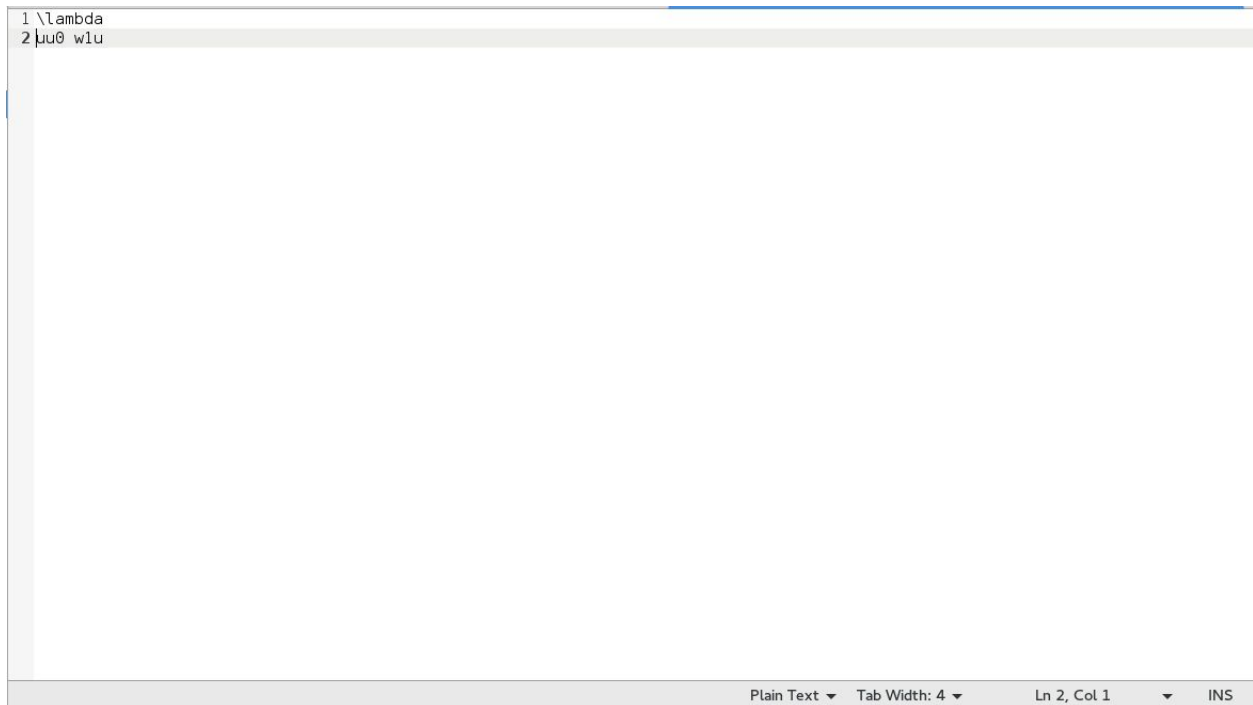
La primer línea debe ser el caso base, separado por espacios.

La segunda línea debe ser el caso recursivo, separado por espacios.

λ se especifica escribiendo '\lambda'.

Actualmente, sólo los caracteres 'u', 'v', 'w', 'x', 'y' y 'z' se utilizan para representar strings. Todos los demás caracteres representan símbolos del alfabeto.

Un archivo ejemplo:



```
1 \lambda
2 pu0 wlu
```

The screenshot shows a text editor window with a light gray background. The top of the window has a title bar. Below the title bar, the first two lines of the file are visible: line 1 contains '\lambda' and line 2 contains 'pu0 wlu'. The rest of the editor area is empty. At the bottom of the window, there is a status bar with the following text: 'Plain Text', 'Tab Width: 4', 'Ln 2, Col 1', and 'INS'.

Correr el script de python (con python2.7) y escribir el nombre del archivo:

```
[rafa@rc proyecto2]$ ls
proyecto2.py  test
[rafa@rc proyecto2]$ python proyecto2.py
Nombre del archivo?
test
Cuantos ciclos?
2
```

Ingresar el nombre del archivo con la información y especificar el número de veces que se debe ejecutar el paso recursivo.

El resultado:

```
[rafa@rc proyecto2]$ ls
proyecto2.py  test
[rafa@rc proyecto2]$ python proyecto2.py
Nombre del archivo?
test
Cuantos ciclos?
2
BASE: \lambda
PASO RECURSIVO: uu0, w1u

iteracion 1: ['1', '0']
iteracion 2: ['11', '10', '01', '010', '011', '000', '111', '110']
STRINGS GENERADOS EN 2 CICLOS: \lambda, 0, 1, 11, 01, 10, 010, 011, 000, 111, 110
[rafa@rc proyecto2]$ |
```

Descripción Técnica

Se definieron caracteres que representan strings al principio del código:

```
#global
_strings = ['u', 'v', 'w', 'x', 'y', 'z']
```

Para generar strings, se recorre una lista de todos los strings generados anteriormente (que en un principio es únicamente la base) y reemplaza un string ('w', 'x', 'y', etc) en específico con cada uno de los strings generados anteriormente.

```
#genera strings con n recursiones
def generarStrings(base, reglas, n):
    generados = base[:]
    viejos = generados[:]
    for x in xrange(n): #numero de iteraciones
        #para cada string generado..
        for s in viejos:
            #para cada regla definida en el paso recursivo..
            for regla in reglas:
                res = crearNuevo(s,regla, viejos)
                generados = list(set(generados).union(res))

    print 'iteracion %d: %s' % (x+1, list(set(generados) - set(viejos)))
    viejos = generados[:]

    generados.sort(key = lambda s: 0 if s == '\lambda' else len(s))
    return generados
```

Para generar los strings a n pasos, se utilizó una función recursiva. La función busca un caracter que pertenezca a los definidos como strings en el código. Una vez que reemplace un string, utiliza el resultado de esto como 'regla' en caso de que todavía contenga strings a reemplazar.

```
def crearNuevo(string, regla, viejos):
    generado = regla[:]
    temp = generado[:]
```

```
res = []
for s in _strings:
    generado = generado.replace(s, string)
    if len(generado) >= 2:
        generado = generado.replace('\lambda', '')
    if temp != generado:
        break

#paso recursivo
if any(x in generado for x in _strings):
    for string in viejos:
        res.extend(crearNuevo(string, generado, viejos))
else:
    if generado not in res:
        res.append(generado)
return res
```