



DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE CONCEPCIÓN
CONCEPCIÓN, CHILE.

Informe N°2

Generando secuencias infinitas

Profesor: Mario Medina

Alumno: Aldo Mellado Opazo

31 de agosto de 2018

Programación Orientada al Objeto
Ingeniería Civil en Telecomunicaciones

1. Generando secuencias infinitas

La secuencia *look and say*, se basa en la comparación, además del conteo, de un elemento previo y otro posterior. Mientras ambos sean iguales, el contador crece. En el caso de que sean diferentes, se anuncia la cantidad de ocurrencias y el número que se estaba contando, así, para la iteración:

$$L_4 = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 1 & 1 \\ \hline \end{array}$$

se comienza contando el primer elemento, es decir el 1, donde al hacer la comparación entre este y el siguiente elemento, se tiene que al ser distintos, el primer y segundo elemento de la línea resultante será la cantidad de ocurrencias y el número que se estaba contando:

$$L_5 = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array}$$

Luego, se contarán la cantidad de 2 que existen, resultando:

$$L_5 = \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 2 \\ \hline \end{array}$$

Finalmente, dado que el tercer elemento de la iteración L_4 es igual al cuarto el contador crecerá.

Luego, como al llegar al cuarto elemento no existen más valores, se tiene entonces que el vector ha sido recorrido en su totalidad, resultando:

$$L_5 = \begin{array}{|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 2 & 2 & 1 \\ \hline \end{array}$$

Dicho esto, se intentó emular la lógica de la resolución del problema, a través de las siguientes líneas de código.

1.1. Solución

Funcion countandSay

```

1  for(int i=1;i<=n;i++)
2  {
3      cout<<"\n L["<<i<<": ";
4      if(i==1)
5      {
6          look.push_back(1);
7          despliegaSecuencia(look);
8      }
9      else if(i==2)
10     {
11         look.push_back(1);
12         despliegaSecuencia(look);
13     }
14     else
15     {
16         look = countandSay(look,i);
17         despliegaSecuencia(look);
18     }
19 }
```

Como se puede ver en la línea 4, se tiene en consideración la inicialización de la secuencia, haciéndola partir con un uno en el primer nivel de ésta.

Luego, se tiene que para el siguiente nivel, deben existir *Dos unos*, y esto se logra inicializando otro valor 1 más.

Finalmente, para el resto de la secuencia se aplica lo que se llama en la línea 16, la función `countandSay(look,i)`, que como parámetros recibe el valor resultante `look`, que no es más que el vector resultante de la iteración anterior de la función.

En la tercera iteración, esta realiza lo siguiente:

```

1 vector<int>countandSay(vector<int> look, int n)
2 {
3     vector<int> aux;
4     aux = look;
5     look.clear();
6
7     int counter=1;
8
9     vector<int>::iterator it=aux.begin();
10
11     for(it;it<aux.end();it++)
12     {
13         if(*(it)==*(it+1) && (it)!=aux.end()-1) //Busca coincidencias entre el valor y el valor posterior a
14             este, haciendo la salvedad de que no se haga esta evaluacion
15             //si es que se ha llegado al valor previo al termino del vector
16             {
17                 counter++;
18             }
19             else
20             {
21                 look.push_back(counter);
22                 look.push_back(*it);
23                 counter=1;
24             }
25     }
26     /*la finalidad de este vector era que se pudiera almacenar una copia de los valores que se empujaron al
27     vector
28     es decir, 21 en el caso de la tercera iteracion, o 1211 en el caso de la cuarta, sin embargo, no lo pude
29     hacer. Se me ocurre que de alguna manera, la copia
30     de este vector, deb  a ir fuera de esta funcion, fuera del for.*/
31     return look;
32 }

```

Esta funci  n recibe un vector, y un entero; el primero corresponde al vector que contiene el resultado del nivel anterior de la secuencia, y el entero, el nivel de secuencia hasta el cual debe llegar.

Lo medular de estas l  neas reside en las siguientes condiciones bajo las cuales se ejecuta el c  digo:

```

1     for(it;it<aux.end();it++)
2     {
3         if(*(it)==*(it+1) && (it)!=aux.end()-1) //Busca coincidencias entre el valor y el valor posterior a
4             este, haciendo la salvedad de que no se haga esta evaluacion
5             //si es que se ha llegado al valor previo al termino del vector
6             {
7                 counter++;
8             }
9             else
10            {
11                look.push_back(counter);
12                look.push_back(*it);
13                counter=1;
14            }
15    }

```

Como se puede ver en la 3a l  nea, se tiene que lo que hace es comparar el valor actual, con el siguiente; si se halla una coincidencia, se aumenta el contador, a la vez que se aumenta el iterador; en caso contrario, se copia el contador hasta el cual hubo coincidencia, y luego, el n  mero que se estaba comparando.

