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:

$$L_4 = \boxed{1}\boxed{2}\boxed{1}\boxed{1}$$

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:

$$L_5 = \boxed{1}\boxed{1}$$

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

$$L_5 = \boxed{1}\boxed{1}\boxed{1}\boxed{2}$$

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:

$$L_5 = \boxed{1}\boxed{1}\boxed{1}\boxed{2}\boxed{2}\boxed{1}$$

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

```
vector<int> countandSay(vector<int> lass)
{
  int l,counter =1;
  vector <int> ssal,ds;
    lass.push_back(1);
    lass.push_back(1);

  ssal = lass;
  ds = ssal;
  lass.clear(); //borra todos los elementos que contiene el vector sobre el cual se harÃ¡n push back los
     valores posteriores de la secuencia

  vector<int>::iterator it=ssal.begin();

  for(it;it<ssal.end();it++)
           {
       if(*(it)==*(it+1) && (it)!=ssal.end()-1)
                   /*Busca coincidencias entre el valor y el valor posterior a este, haciendo la salvedad de
    que no se haga esta evaluacion
                   si es que se ha llegado al valor previo al termino del vector*/
               {
                   counter++;
               }
```

Mediante el uso de un ciclo for se recorren los elementos almacenados en el vector resultante de la iteración previa, y se evalúan según lo que establece la secuencia look and say, evaluándo elemento a elemento, comparándolos si es que son iguales el elemento actual, con el siguiente, y si es que este último, no corresponde al final del vector a recorrer.

```
                else if(it==ssal.end()-1) //en caso contrario, haz una copia de los elementos almacenados en
    el vector, borra el vector actual
                {   /*si es que se llega al final del vector, se debe hacer lo siguiente*/
                    lass.push_back(counter);
                    lass.push_back(*it);
                    counter=1; //vuelve el contador a uno luego de haber llegado al termino luego de recorrer
     del vector
                    continue;
                }
```

En este caso, la condición a evaluar es que si ya se ha recorrido el vector, entonces se haga un `push_back(counter)`, es decir, se indique cuántas coincidencias hubieron para cuando se recorrió completamente el vector, y además, se hace un `push_back(*it)`, es decir, el valor de lo apuntado por el iterador a la hora de terminar el vector. Finalmente, se reinicia el valor del contador.

Un ejemplo de como debería operar la lógica de esto, es que si la secuencia es esta:

$$L_2 = \boxed{1}\boxed{1}$$

Y es que el primer elemento se compara con el siguiente y como son iguales, avanza, el contador aumenta, es decir, `counter=2`, y como al avanzar el iterador, se encuentra que por la condición que establece el for, se ha llegado al final del vector, entonces se hace un `push_back` del contador y del número del cual se estaba buscando coincidencias.

Con esto lo que se hace es que la secuencia resultante, es decir.

$$L_3 = \boxed{2}\boxed{1}$$

Tabla 1: *Dos unos*

1.1. Problemas

- Aún cuándo en otros intentos el código era capaz de evaluar las condiciones, es decir, de reconocer cuándo había llegado al final o bien, detectado una coincidencia. Sin embargo, para las iteraciones actuales, el código es incapaz de realizar dicha tarea.
- Parte del mismo problema mencionado anteriormente, es que el vector generado, es decir, para la segunda tercera iteración, cuyos valores son 2-1, al entrar en la función, countandsay, esta es incapaz de retener el valor generado de manera tal que la próxima vez que se le llame, parte del vector generado previamente. En mi opinión, es un tema de orden, de saber dónde poner éstas líneas.

```
  int l,counter =1;
  vector <int> ssal,ds;
    lass.push_back(1);
    lass.push_back(1);

  ssal = lass;
  ds = ssal;
  lass.clear(); //borra todos los elementos que contiene el vector sobre el cual se harÃ¡n push back los
     valores posteriores de la secuencia
```

Puesto que, si bien se hace una copia del vector resultante en la iteración anterior, según lo que se explica en las líneas:

```
  ssal = lass;
  ds = ssal;
```

Cuando se desea trabajar sobre el vector `ssal`, este no se traduce como la copia de los valores previos, sino como el resultado de valores actuales obtenidos a partir de las primeras líneas de código.