Departamento de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de Concepción
Concepción, Chile.

Informe N°3

Los números pandigitales de 0-9

Profesor: Mario Medina

Alumno: Aldo Mellado Opazo

15 de septiembre de 2018

Programación Orientada al Objeto
Ingeniería Civil en Telecomunicaciones

1. Los números pandigitales de 0-9

Se nos introduce a los números pandigitales explicándonos que es el resultado de la combinatoria entre ciertos digitos cuya principal propiedad es que estos aparecen sólo una vez. Es por esta definición que en la implementación de la resolución de esta tarea se consideró la combinatoria, llevada a cabo en el código por la siguiente función:

```
int factorial(int largol1)
{
  int factorial=1;

  for(int i=1;i<=largol1;i++)
  {

    factorial=factorial*i;
  }
  return factorial;
```

Esto, acorde a la combinatoria, es capaz de obtener el número de posibles combinaciones de dígitos entre 0-9. Y que de acuerdo a lo entregado por la función corresponde a 3628800.
Posteriormente se debían generar dichas permutaciones y para ello se hizo uso de la siguiente función.

```
void permutar(list<int>l1,int factorial)
{
  int j=0;
  list<int>l2;
  l2 = l1;

  ofstream salida("salida.txt");

  while(j<factorial)
  {
        list<int>::iterator it=l2.begin();

        for(auto x:l2)
     {
              salida<<x;
          }
        salida<<"\n";

        next_permutation(l2.begin(),l2.end());
    j++;
  }
  salida.close();
```

Notar que además de valerse de la función `next_ permutation(l2.begin(),l2.end())` para obtener las posibles permutaciones, se hace uso de un archivo de texto generado al cual se le escriben los valores resultantes. Esto será especialmente util para cuando deban hacerse ciertas consideraciones posteriores, pues permitirá que el procesamiento sea más rápido que si se mostrara por consola o se calculasen cada vez que se desee ejecutar una condición sobre los números de la permutación.

Luego se pide que para los números generados, se evalúe cuántos y cuáles de ellos satisfacen las condiciones dadas. Para dicho cometido se genera la siguiente función:

```
void propiedad()
```

Como se aprecia en el código adjuntado en al mail enviado, se tiene que una vez definidos los enteros que servirán de contadores de ocurrencia de cada una de las propiedades pedidas, se procede a transformar, mediante `stoi(temp)`, los números generados que se almacenaron en forma de strings, a enteros para poder así travajar con ellos. Se trabaja además con la función `int2vector`, que nos permite transformar los números, en vectores que serán recorridos por iteradores.

Así, tal y como se aprecia en las siguientes líneas

Se tiene que se evalúan las condiciones, y luego, se establece si hubo o no una ocurrencia.

```
1
2   while(entrada >> temp)
3   {
4     aux = stoi(temp);
5     vector1 = int2vector(aux);
6
7     vector<int>::iterator it = vector1.begin();
8
9     for(it;it!=vector1.end();it++)
10    {
11      d2 = (*(it+1))+(*(it+2))+(*(it+3));
12      d3 = (*(it+2))+(*(it+3))+(*(it+4));
13      d5 = (*(it+3))+(*(it+4))+(*(it+5));
14      d7 = (*(it+4))+(*(it+5))+(*(it+6));
15      d11 =(*(it+5))+(*(it+6))+(*(it+7));
16      d13 =(*(it+6))+(*(it+7))+(*(it+8));
17      d17 =(*(it+7))+(*(it+8))+(*(it+9));
18
19      if(d2%2!=0)
20      {
21        dv2++;
22      }
23      else if(d3%3!=0)
24      {
25        dv3++;
26      }
27      else if(dv5%5!=0)
28      {
29        dv5++;
30      }
31      else if(d7%7!=0)
32      {
33        dv7++;
34      }
35      else if(d11%11!=0)
36      {
37        dv11++;
38      }
39      else if(d13%13!=0)
40      {
41        dv13++;
42      }
43      else if(d17%17!=0)
44      {
45        dv17++;
46      }
```

Finalmente, se pide saber cuántos números son finalmente divisibles por 2, 3, 5, 7, 11, 13 y 17 respectivamente. Esta información es entregada, como dicen las siguientes líneas, por los valores de, `dv2,dv3,dv5,dv7,dv11,dv13,dv17`

```
1 }
2   suma =  d2+d3+d5+d7+d11+d13+d17;
3   cout<<"\nLa cantidad de nÃºmeros divisibles por 2 es: "<<dv2<<endl;
4   cout<<"La cantidad de nÃºmeros divisibles por 3 es: "<<dv3<<endl;
5   cout<<"La cantidad de nÃºmeros divisibles por 5 es: "<<dv5<<endl;
6   cout<<"La cantidad de nÃºmeros divisibles por 7 es: "<<dv7<<endl;
7   cout<<"La cantidad de nÃºmeros divisibles por 11 es: "<<dv11<<endl;
8   cout<<"La cantidad de nÃºmeros divisibles por 13 es: "<<dv13<<endl;
9   cout<<"La cantidad de nÃºmeros divisibles por 17 es: "<<dv17<<endl;
```

1.1. Problemas

1. Uno de los problemas presentados, fue que dada la gran cantidad de números que se debían generar, evaluarlos en el momento o iterar la función `permutar` por todas y cada una de las veces que se deseara evaluar alguna condición no era en lo absoluto eficiente y generar un vector de vectores de 362800 elementos tampoco era muy factible, de modo que la alternativa del archivo de texto era sino, una de las más óptimas.

2. **Cantidades y suma:** se aprecia en el código que tanto la suma, como la cantidad de números divisibles por los distintos números primos, en consideración de la cantidad de números generados es notoriamente baja, esto debido a cómo se concibió el código y es que por ejemplo, para la suma, se tomaron los números resultantes de más de 9 cifras de modo que al sumarlos todos se produjo overflow y el número representado no es el que realmente debió obtenerse.