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

Informe N°1

Calculando factoriales

Profesor: Mario Medina

Alumno: Aldo Mellado Opazo

24 de agosto de 2018

Programación Orientada al Objeto
Ingeniería Civil en Telecomunicaciones

1. Calculando Factoriales

Para este problema se debía çalcular.[el] factorial de 1000 y sumar sus dígitos. Para ello, se debía tener en consideración que las variables de tipo `int, long int, long long int`, no dan a basto para almacenar valores de factorial tan grandes -notar que 1000! es $\approx 4{,}0238726007 \cdot 10^{2567}$ - de modo que debía buscarse una solución alternativa.

Para estos efectos se consideró trozar cada número obtenido en el cálculo del factorial de la siguiente manera.

10! =	3	6	2	8	8	0	0

Tabla 1: Representación de vector resultante

de manera que a la siguiente iteración, es decir, al 11!, el número por el cuál hacer el producto no fuese 3628800, sino que el vector mencionado en **??**.

Para este cometido, se crearon las funciones;

```
vector<int> num2vec(vector<int> vector, int value)
```

Lo que hacía dentro del código era precisamente seccionar los números para evitar el overflow. Esto lo hacía usando una división iterativa que dividía el número por potencias de diez hasta que el resto era módulo resultante era 0, lo que se traducía en que este había llegado a la unidad, la cual debía ser empujada como último elemento del vector.

```
int sumofvect(vector<int> v1,vector<int>v2)
```

Esta función debería tomar los elementos del vector y uno a uno irlos sumando. Considerando además el *carry* que se produce para la suma entre números sobre 9. Entrega además un vector resultante (`Suma`) que corresponde al resultado de la suma de los dígitos como se mencionó anteriormente.

```
  si se hace el reverse, se puede trabajar con la funciÃn push_back()*/
```

El factorial, por definición involucra un producto de dos elementos que se relacionan por ser el producto de un número con otro previamente obtenido, de modo que en el código, la función que realiza dicho cálculo es esta y al igual que en la función `sumofvec`, se hace, elemento a elemento el producto, considerando los carry.

Al igual que en la función suma, se consideran aspectos de orden y largo, es decir, se compara el largo de los dos vectores y en caso de no ser iguales, se rellena con ceros tantas veces como dígitos de diferencia tengan respecto del largo.

1.1. Problemas

1. **Largo vectores**: Se tiene que desde casos tan sencillos como 4! y 5!, cuyos resultados son respectivamente, 24 y 120, tienen largo de vector diferente. Para dar solución a ello, se usaron las siguientes líneas de código.

```
    for(it2=v2.begin();it2!=v2.end();it2++)
    {
        for(it1=v1.begin();it1!=v1.end();it1++)
        {
```

2. **Función**: En el caso de esta función, se tiene que al realizar el producto pero no arrastrar a la suma el resultado, no continúa en los valores siguientes. Por ejemplo, si el producto es 9*2 = 18, hace `push_ back(8)`,

mas no hace la suma entre el resultado del producto de los siguientes números y el carry de la multiplicación previa, si lo hubiera. Que para efectos de este ejemplo, sería 1.

El problema podría eventualmente ser resulto si se enlaza la función `prodofvect` con la función `sumofvect`.