ICOM2019 – FINAL

14 de diciembre de 2019

Notas:

1. Al finalizar, enviar por e-mail los archivos fuente de cada ejercicio con nombre **APELLIDO_NOMBRE_Ejer_N.cpp** a icom.cabib@gmail.com
2. Uso de prácticos: **se pueden utilizar los trabajos prácticos propios realizados.**
3. Uso de Internet**: solo para la consulta de referencias de funciones de C/C++**

Problema 1: Números Claros

Se llaman "números claros" a los números autorreferentes que tienen entre una y diez cifras y que cumplen con la siguiente propiedad: la primera cifra indica cuantos "0" tiene el número, la segunda cifra (si existe), cuantos "1", la tercera cifra (si existe), cuantos "2" y así hasta que la décima cifra (si existe) indica cuantos "9" hay en el número.

Ejemplos:

1210	Un "0", dos "1", Un "2", cero "3"
6210001000	Seis "0", dos "1", un "2", un "6" y cero "3","4","5","7","8","9".

Se solicita:

a. Implementar una función que chequee si el argumento que recibe es o no un número claro (deberá retornar **`true`** si lo es, **`false`** si no lo es). El número es recibido a través de su representación decimal en un string, por ejemplo, el número 1234 será recibido como el string "1234". El prototipo de la función debe ser (respetar estrictamente el prototipo):

```
bool esNumeroClaro(const string &numero);
```

b. Implementa una función que busque e imprima todos los números claros existentes en el rango dado [**`0 – rango`**]. El prototipo de la función debe ser (respetar estrictamente el prototipo):

```
void buscaNumerosClaros(int rango);
```

Problema 2: Caminata al azar sin cruces

Una caminata al azar sin cruces es un proceso en el cual un caminante se desplaza como en una caminata al azar normal pero en cuanto trata de volver a visitar un sitio previamente visitado, se detiene y deja de caminar. Este modelo es un paradigma para el estudio de las configuraciones espaciales de cadenas poliméricas y proteicas.

Cada caminante se ubica sobre una red cuadrada, por lo que en cada paso elige aleatoriamente moverse a uno de los cuatro sitios vecinos, ubicados respectivamente, arriba, abajo, derecha o izquierda. El caminante debe ir marcando como visitados los sitios por los que pasa. Si en algún momento, el sitio vecino elegido para moverse ya ha sido previamente visitado, la caminata se corta. Si luego de 10000 (**`MAX_PASOS`**) pasos la caminata aún sigue activa, forzar su interrupción.

Se solicita realizar 1000 (**`NUM_CASOS`**) caminatas al azar sin cruces imprimiendo la cantidad de pasos recorridos en cada caso.

Hint: Modifique el ejemplo 2 de la clase 5, agregando/modificando la funcionalidad que haga falta como para que las caminatas sean sin cruce y se pueda ejecutar con el siguiente main:

```
int main() {
    const int NUM_CASOS = 1000;
    const int MAX_PASOS = 10000;
    Entity e;

    srand(time(0));          // Inicializa generador pseudo-aleatorio

    for (int caso = 0; caso < NUM_CASOS; caso++) {

        // resetea el camino de la entidad, como si empezara nuevamente
        e.reset();

        for (int paso = 0; paso < MAX_PASOS; paso++) {
            // el método move debe retornar true si se pudo realizar el paso, o false
            // si el movimiento pasa por una posición ya visitada
```

```
            if( ! e.move(Direction(rand() % NUM_DIRS)) )
                break;
        }
        cout << e.pathLen() << endl;
    }
    return 0;
}
```

Problema 3: Proto-sistema gráficos

Se desea diseñar e implementar un sistema gráfico elemental para poder dibujar figuras sencillas. En el archivo Shapes.cpp se da un esqueleto del sistema.

Implementar los métodos/clases faltantes.