

Enunciado:

Resuelva los siguientes ejercicios en C++14 sobre apuntadores, arreglos y clases. Utilice el estándar C++14 en la solución de sus problemas. No olvide compilar con los *flags* apropiados para detectar *warnings* y errores.

1. Escriba una clase, **Racional**, que permita trabajar con números racionales. Esta clase es útil ya que nos permite representar números como $1/3$ sin pérdida de precisión (si lo representáramos como un real 0,333333 estaríamos perdiendo precisión debido a errores de redondeo). Para ello, almacenaremos en la clase el numerador y el denominador del número racional. Las operaciones que nuestra clase necesitará definir son:
 - a) Construcción del objeto sin parámetros
 - b) Construcción del objeto con parámetros
 - c) Métodos para el acceso a los atributos
 - d) Métodos para la modificación de los atributos
 - e) Operaciones de inserción $<<$ y extracción $>>$
 - f) Operador de asignación, $=$, y de asignación con operación $+=$, $-=$, $*=$, $/=$
 - g) Operadores aritméticos: suma, resta, producto y división
 - h) Operadores relacionales: igualdad, $==$, y desigualdades, $>$, $<$, $>=$, $<=$, $!=$
 - i) Incremento (pre y post), decremento (pre y post) y cambio de signo

Además, todos los números deben ser simplificados después de cada operación. Para ello, podemos calcular el máximo común divisor del numerador y denominador utilizando el algoritmo de Euclides, y luego dividir ambos números por este valor:

```
1 | int gcd(int num, int den) {  
2 |     long gcd, tmp, rest;  
3 |     gcd = abs(num);  
4 |     tmp = den;  
5 |     while (tmp > 0) {  
6 |         rest = gcd % tmp;  
7 |         gcd = tmp;  
8 |         tmp = rest;  
9 |     }  
10 |     return gcd;  
11 | }
```

2. El juego de dominó se juega con piezas que generalmente son rectángulos negros, divididos por una línea central, con algunos puntos blancos en cada lado. Por ejemplo, el dominó

```
||      +-----+-----+  
||      |         | #     # |  
||      |   #     |         |
```



se llama el dominó $[1|4]$, con un punto en su lado izquierdo y cuatro en su lado derecho. Defina una clase `Domino` que representa un dominó tradicional. Su clase debe exportar las siguientes características:

- Todas las variables de instancia deben ser privadas
- Un constructor por defecto que crea el dominó $[0|0]$
- Un constructor parametrizado que toma la cantidad de puntos en cada lado
- Dos métodos `set_left_dots(ldots)` y `set_right_dots(rdots)`
- Dos métodos `get_left_dots()` y `get_right_dots()`
- Un método `to_string()` que crea una representación tipo `string` del dominó
- Sobrecarga inserción `<<` tal que imprime una representación de un dominó
- Sobrecarga de los operadores relacionales `==`, `!=`, `<`, `<=`, `>`, `>=`
- Un método `flip_them()` que invierte los valores de las variables de instancia de los puntos derechos e izquierdo

Escriba la interface `domino.hpp` y la implementación `domino.cpp` que exporta esta clase. Pruebe su implementación de la clase `Domino` escribiendo un programa que cree un conjunto completo de fichas de dominó desde $[0|0]$ hasta $[6|6]$ e imprima estos dominós en la terminal. Un juego completo de fichas de dominó contiene una copia de cada posible ficha en ese rango, impidiendo duplicados que resultan de invertir 180° un dominó. Un juego de dominó, por lo tanto, tiene un dominó $[4|1]$ pero no un dominó $[1|4]$ por separado, son las mismas fichas.

- Usando la clase `Rational`, calcule las primeras 16 sumas parciales, S_n , de la serie geométrica. Es decir,

$$S_n := \sum_{k=0}^n ar^k = a \frac{1 - r^{n+1}}{1 - r}, \quad \text{con } r \neq 1$$

con $a = 1$, $r = 1/2$ y $n = 0, \dots, 15$. En un arreglo dinámico de tipo `Rational` guarde los valores S_n . Recuerde referirse al espacio reservado en el *heap* con un puntero `Rational *pr`, y liberarlo cuando no sea más necesario. Imprima las sumas parciales guardadas en el arreglo usando el siguiente formato:

```
0 1/1 1/1 1.0
1 1/2 3/2 1.5
2 1/4 7/4 1.75
3 1/8 15/8 1.875
...
12 1/4096 8191/4096 1.99976
13 1/8192 16383/8192 1.99988
14 1/16384 32767/16384 1.99994
15 1/32768 65535/32768 1.99997
```

donde las columnas rotulan los valores de n , S_n , ar^n y S_n expresado como un número de punto flotante.

4. Con la clase `Domino` genere un juego completo de dominó “estándar”, que consta de 28 fichas diferentes. El juego debe estar representado como un arreglo dinámico de tipo `Domino`. Refiérase a la memoria reservada en el *heap* usando un puntero `Domino *game`. Sobrecargando el `operator<<` de inserción como

```
1 | ostream & operator<<(ostream &os, const Domino &d) {  
2 |     os << "[" << d.get_left_dots() << " | "  
3 |     << d.get_right_dots() << "];"  
4 |     return os;  
5 | }
```

imprima el juego completo de dominó “estándar” en el siguiente formato:

```
[0|0] [0|1] [0|2] [0|3] [0|4] [0|5] [0|6]  
      [1|1] [1|2] [1|3] [1|4] [1|5] [1|6]  
          [2|2] [2|3] [2|4] [2|5] [2|6]  
              [3|3] [3|4] [3|5] [3|6]  
                  [4|4] [4|5] [4|6]  
                      [5|5] [5|6]  
                          [6|6]
```

Suponga ahora que cada lado de las fichas del dominó toman 3 valores: 0, 1 y 2.
¿Puede su implementación imprimir el juego completo para esta versión del dominó?
¿Cuántas fichas diferentes hay en total en este caso?