# Universidad de Costa Rica
# CI-0113: Programación 2
# Laboratorio #4

**Objetivo:** Familiarizar al estudiante con el concepto y uso de clases en C/C++.

**Enunciado**

Resuelva los siguientes problemas. Escriba un código en C/C++ que resuelva los siguientes problemas:

1. La palabra clave *static* es de gran relevancia en C/C++ ya que permite establecer variables que perduran más allá de su contexto: las variables estáticas pertenecientes a una función conservan su valor entre llamados, las variables estáticas pertenecientes a una clase son compartidas por todos los objetos de la clase y pueden accederse aunque no existan objetos, los métodos estáticos pueden ser invocados aún sin crear ningún objeto de la clase utilizando la sintáxis:

   ```
   Clase::métodoEstático(params...)
   ```

   a. Implemente una clase Primos que posea las siguientes características:
   b. Debe poseer los atributos necesarios para almacenar una cantidad ‘n’ de primos de manera estática.
      i. Los atributos estáticos deben llevar un valor inicial que será almacenado al compilar, de esta manera la declaración del atributo se hace dentro de la clase, pero su inicialización fuera de ella.
      ii. El valor de inicialización debe especificarse fuera de la clase:

      ```
      tipo Clase::atributo = valor_inicial;
      ```

      iii. El valor de inicialización se almacena en compilación, por lo que debe ser un valor estático/fijo (no puede ser el resultado de llamar a **new**).
      iv. El valor 0 se puede usar para representar un puntero “null”.
   c. Debe poseer un método estático: *cargar* que recibe un número entero ‘n’ que representa la cantidad de valores primos que se desea encontrar. Si el tamaño actual del arreglo estático de primos es menor a ‘n’, entonces creará un nuevo arreglo y liberará la memoria del anterior. Deberá encontrar los primeros ‘n’ números primos e introducirlos al arreglo. (Si ‘n’ es menor a la cantidad de primos actual entonces la función no hará nada)
   d. Debe tener métodos métodos estáticos que permitan obtener la cantidad de primos disponibles *getNumberPrimes()* y obtener el i-ésimo número primo *getPrime(unsigned int i)*.
   e. (Opcional: puntos de logro +5) Intente que su código sea capaz de encontrar los primeros 10 000 000 de números primos en menos de dos minutos y medio (la velocidad de su procesador impactará el rendimiento, ajuste a conveniencia). Puede medir el tiempo tomado utilizando el siguiente código:

      ```
      #include <time.h> // Al inicio del código
      unsigned int t_zero = time(0);
      Primos::cargar(10000000);
      cout << “Tiempo tardado: ” << (time(0)-t_zero) << endl;
      ```

2. Los números enteros son el conjunto numérico que posee a los números naturales (aquellos que se utilizan para contar), sus opuestos (los números negativos) y el 0. Este conjunto numérico posee ciertas características interesantes como que nos permite trabajar con aritmética modular, se utiliza de base para el conjunto de números racionales (donde todo número es la relación entre dos números enteros, e.g: ⅔). Además, los números enteros pueden representarse en función de los diferentes factores primos (números mayores a uno que poseen un único factor: si mismos) y sus exponentes respectivos que los componen.
    a. De esta manera un número como 72 se puede representar como:
    $$72 = 2^3 * 3^2$$
    b. Implemente una clase Compuesto que permita representar un número entero según sus componentes. Debe cumplir las siguientes características:
    c. El constructor recibe un entero con signo de 8 bytes y debe descomponer dicho número en sus componentes primos y sus exponentes respectivos.
    d. Debe tener los atributos necesarios para almacenar los factores primos y sus exponentes: considere que en el peor de los casos el número estará compuesto por un máximo de 20 factores primos diferentes (¿de dónde cree que viene este número?).
    e. Debe considerar qué utilizar para diferenciar entre números positivos y negativos. Además debe considerar cómo representar los casos especiales del +- 1 (no posee factores primos) y el 0.
    f. Implemente un método *print* que imprima el número almacenado en formato:
    $$p_1^{e_1} * p_2^{e_2} * \ldots * p_i^{e_i}$$
    g. (Opcional puntos de logro +5) Implemente un método *divisible* que reciba por parámetro otro Compuesto y retorne si el objeto es divisible entre el número compuesto recibido. Debe utilizar los factores primos y sus exponentes para la respuesta, únicamente.
    h. (Opcional puntos de logro +5) Sobrecargue el operador *= de manera que permita multiplicar y almacenar el resultado de la multiplicación.
    i. (Opcional puntos de logro +5) Sobrecargue el operador /= de manera que permita dividir y almacenar el resultado de la división, el numerador debe ser divisible entre el denominador.
    j. (Opcional puntos de logro +5) Sobrecargue el operador == de manera que permita comparar dos objetos Compuestos y retornar si son iguales o no. (Aproveche este operador para sobrecargar el operador !=)