Estimados Guerreros del Laboratorio,

La tarea está lista, aquí en GitHub. Primero, para **comenzar a trabajar lo antes posible**, lean lo siguiente con cuidado.

1. Hagan git pull del repositorio de materiales del curso (Uds. ya tienen un repositorio local del mismo para hacer git pull) o repitan el git clone si lo borraron.
Liondance/CI-2125 (github.com)

2. Copien la carpeta c01 a su repositorio privado de trabajo.
Quizás es buena idea mover los archivos que ya tienen allí a una subcarpeta c00, para que la numeración de los archivos tenga más sentido.
Aunque, de una manera u otra, no va a haber colisiones de nombres: solo para que el material quede más ordenado.

Acerca de la tarea en sí:

1. El proyecto tiene varios archivos: vamos a trabajar de manera modular, enlazando el código generado a partir de todos ellos en un ejecutable.
- Los archivos con extensión .h son llamados encabezados
- Los encabezados permiten que otros módulos vean la declaración (firma o prototipo) de las funciones exportadas por un módulo.
- Cada archivo con extensión .cxx representa la implementación de un módulo: estos archivos tienen símbolos privados (definidos con *static*) y públicos
- El script *build* muestra cómo compilar el proyecto: como pueden apreciar es muy sencillo. El script *clean* hace limpieza.
- Si usan MS-Visual Studio, deben crear un proyecto que tenga los archivos .cxx.

2. Para ayudarles a que todo vaya bien, ya tienen un esqueleto bien formado ... incluso compila (y hasta corre, pero aún no como debería, lógicamente)
- **No deben cambiar los encabezados en absoluto**: la declaración de las funciones exportadas define la interfaz del módulo.
- En los **archivos de implementación** de los módulos **solo deben completar las funciones marcadas con la palabra TAREA** en su comentario
- Los comentarios explican que hacer: lean las instrucciones y sugerencias con cuidado

3. El archivo main.cxx no tiene encabezado ya que no implementa un módulo: este archivo es el programa de prueba, Como tal, se rige por reglas distintas.
- En principio pueden hacer lo que les da la gana después del comentario ***WILD WILD WEST***!
- Es decir, pueden cambiar todo el archivo menos las inclusiones de encabezados que deben dejar intactas. Sin embargo …
- En práctica es bueno que vayan agregando y reorganizando código de prueba incrementalmente al esqueleto de prueba ya provisto.
- 6 puntos de la evaluación corresponden a lo que hagan con el programa de prueba.
- los 29 puntos restantes van a provenir de la implementación de las funciones.

4. **Don't Panic** (Sin Pánico)
a. Todas las funciones son cortas: ¡de 1 a 6 líneas!
b. Lean los módulos con cuidado (primer paso) y durante la implementación de las funciones (o antes) se van a dar cuenta que pueden reusar código.
c. Trabajen de manera incremental asegurándose primero que las funciones a reusar están bien: esto reduce el *debugging* un 80%.
d. Sigan el siguiente camino ...

5. **La ruta del éxito**: sugerida por la experiencia que hace la diferencia
a. Primero leer cada módulo (encabezado seguido de la implementación) en este orden: *stochastic*, *statistics*, y *arrayops*.
b. Implementar las funciones en los módulos stochastic y statistics. Ya tienen el código de prueba para la varianza en el main.
c. Para probar la generación de números aleatorios normales, impriman unos 20 números y vean si se agrupan alrededor del cero.
d. La mejor forma de probar la generación de números aleatorios normales es mostrar un histograma, pero sugiero dejarlo para luego.
e. Ahora implementen las funciones de búsqueda, intercambio, desordenamiento, y ordenamiento *para arreglos de enteros*.
f. Prueben, prueben, prueben los arreglos de enteros lo suficiente para convencerse que funcionan perfectamente.
g. Ahora generalicen lo anterior a arreglos de números en doble precisión y arreglos de "strings".
h. Asistan a la clase del viernes: voy a pasear sobre el código con explicaciones y sugerencias.
i. Van a aprovechar la clase del viernes más si comienzan a trabajar ya, en serio.
j. Sería bueno que implementaran unas 4 o 5 funciones hoy, jueves. No es difícil hacerlo.

6. **¡Las fórmulas mágicas!**

Estas corresponden a *resultados conocidos* que no es razonable pedirles que descubran, aunque apuesto que algunos ya conocen la fórmula de la varianza.

a) La fórmula mágica para implementar la función ***double normal()*** en el archivo *stochastic.cxx*, es la siguiente:

$$Z = \sqrt{-2 \ln U_1} \cos(2\pi U_2)$$

donde U[1] y U[2] son dos números aleatorios independientes, distribuidos uniformes entre 0 y 1.

b) La fórmula mágica para implementar la función ***double varp_sdf*** en el archivo *stats.cxx* es ...
... **la fórmula usada en todo el mundo** para definir la varianza de un conjunto de observaciones, *considerado como una población*:

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

donde la letra griega μ representa el promedio de las x, es decir el promedio de los valores observados.
Supongo que ya la conocen por sus cursos de estadística. También la pueden apreciar en Variance - Wikipedia.
Mango Peel ALERT: les recuerdo que los índices de colecciones en libros de matemática **comienzan en 1**, pero ...
... en C y C++ los índices de *los índices de los elementos en los arreglos* ***comienzan en cero***.
Parte de la tarea de aprender a implementar algoritmos en C es hacer la traslación con cuidado.

7. Por último ...
Por favor disculpen la tardanza, en parte por problemas logísticos, la pérdida de mi computadora, y el ajetreo del viaje, ...
... pero mayormente por el esfuerzo de darles una tarea de laboratorio bien organizada y con relevancia para lo que sigue,
La fecha de entrega es el 14. Pienso tener las notas de su primera entrega esta semana.

Enzo Alda

Jefe del Laboratorio