Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos
Taller 2: Estructuras lineales, 2022-30

1 Objetivo

Completar la implementación de una aplicación que usa estructuras lineales para calcular anagramas. En especial, se desea evaluar las habilidades del estudiante en el desarrollo y uso de las operaciones de recorrido en secuencias y uso de pilas y colas como apoyo algorítmico.

2 Recordatorio: compilación con g++

La compilación con g++ (compilador estándar que será usado en este curso para evaluar y calificar las entregas) se realiza con los siguientes pasos:

1. **Compilación**: de todo el código fuente compilable (**ÚNICAMENTE LOS ARCHIVOS CON EXTENSIONES** *.c, *.cpp, *.cxx)
   `g++ -std=c++11 -c *.c *.cxx *.cpp`
2. **Encadenamiento**: de todo el código de bajo nivel en el archivo ejecutable
   `g++ -std=c++11 -o nombre_de_mi_programa *.o`

   Nota: Estos dos pasos (compilación y encadenamiento) pueden abreviarse en un sólo comando:
   `g++ -std=c++11 -o nombre_de_mi_programa *.c *.cxx *.cpp`
3. **Ejecución**: del programa ejecutable anteriormente generado
   `./nombre_de_mi_programa`

ATENCIÓN: Los archivos de encabezados (*.h, *.hpp, *.hxx) **NO SE COMPILAN**, se incluyen en otros archivos (encabezados o código). Así mismo, los archivos de código fuente (*.c, *.cpp, *.cxx) **NO SE INCLUYEN**, se compilan. Si el programa entregado como respuesta a este Taller no atiende estas recomendaciones, automáticamente se calificará la entrega sobre un 25% menos de la calificación máxima.

3 Desarrollo del taller

Un anagrama es una secuencia que resulta del reordenamiento de los componentes de un conjunto de símbolos. Por ejemplo, las secuencias de caracteres "roma" y "mora" son anagramas de "amor", es decir, usan los mismos símbolos en un orden diferente.

El desarrollo del taller consistirá en completar el código de las funciones escritas en el archivo "NextAnagram.hxx" y, adicionalmente, generar un reporte (en formato PDF) donde responda a algunas preguntas de reflexión.

1. **(5%)**En su reporte, describa paso a paso lo que hace el procedimiento principal (función " main") en el archivo "solve_anagram.cxx". Sea conciso y describa cada paso en 15 palabras o menos.
2. **(5%)** Describa la salida de la función "ReadAsCharacterList" en su reporte. Sea conciso en su descripción (15 palabras o menos).
3. **(5%)** Describa la salida de la función "ReadAsStringList" en su reporte. Sea conciso en su descripción (15 palabras o menos).
4. Estudie el archivo "NextAnagram.h". ¿Entiende todo? si no, ¿le puede preguntar al profesor o al monitor?
5. Estudie el archivo "NextAnagram.hxx". ¿Entiende todo? si no, ¿le puede preguntar al profesor o al monitor? Identifique las secciones marcadas con el comentario /** TODO #n **/, estas son las secciones donde usted debe completar el código del algoritmo. Las variables declaradas en las líneas 15 a 21 son suficientes para completar el código.

6. **(5%)** `TODO #1`: a partir de la secuencia "`lst`" de entrada a la función, llenar la pila "`s`".
7. **(5%)** `TODO #2`: extraer el tope de la pila "`s`" y almacenarlo en la variable "`v_aux`".
8. **(20%)** `TODO #3`: iterativamente, extraer el frente (cabeza) de la cola "`q`" y adicionar este valor a la misma cola. Este ciclo se detiene cuando el elemento que se acaba de extraer es menor que lo contenido en la variable "`pivot`".
9. **(20%)** `TODO #4`: iterativamente, extraer el frente (cabeza) de la cola "`q`" y adicionar este valor a la misma cola. Este ciclo se detiene cuando el elemento que se acaba de extraer es mayor que lo contenido en la variable "`pivot`".
10. **(10%)** `TODO #5`: iterativamente, vaciar la cola "`q`" en la pila "`s`".
11. **(15%)** `TODO #6`: iterativamente, vaciar la pila "`s`" e insertar por la cabeza cada elemento en la secuencia resultado "`res`".
12. **(10%)** `TODO #7`: Completar la función que calcula la cantidad de anagramas que resultan de una secuencia de entrada. Si el tamaño de la secuencia de entrada es n, la cantidad de anagramas posibles es $n!$.
13. Para que pueda probar, se provee un archivo de entradas de muestra (`input_00.in`) y su correspondiente salida (`output_00.in`).
14. **(5%)** Investigue si existe una función que calcule anagramas en la STL. Ponga en su reporte el encabezado de la función y una descripción concisa (10 palabras o menos) de su forma de uso.

4 Evaluación

La entrega se hará a través de la correspondiente asignación de BrightSpace, antes de la medianoche del próximo jueves 19 de agosto. Se debe entregar un único archivo comprimido (únicos formatos aceptados: .zip, .tar, .tar.gz) que contenga dentro de un mismo directorio (sin estructura de carpetas interna), documentos (único formato aceptado: .pdf) y código fuente (.h, .hxx, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

La evaluación del taller tendrá la siguiente escala para cada una de las preguntas o secciones de código a completar:

- **Excelente** (**5.0/5.0**): El código es correcto, compila sin advertencias y/o la respuesta es correcta y concisa.
- **Bueno** (**3.8/5.0**): El código es correcto, pero compila con advertencias y/o la respuesta es correcta pero no es concisa.
- **No es un trabajo formal de ingeniería (3.0/5.0)**: El código es parcialmente correcto y/o compila con advertencias y/o la respuesta es aproximada y/o no es concisa.
- **Necesita mejoras sustanciales** (**2.0/5.0**): El código no es correcto y/o la respuesta no es clara o es incorrecta.
- **Malo** (**1.0/5.0**): El código entregado por el estudiante no compila en el compilador g++ (mínimo versión número 4.5).
- **No entregó** (**0.0/5.0**): No entregó los archivos solicitados, o los archivos no se encuentran en el formato indicado.