Tarea 2
Manejo dinámico de estructuras lineales
Curso 2022

1. Introducción y objetivos

Esta tarea tiene como principal objetivo trabajar sobre el manejo dinámico de memoria con estructuras lineales.

Para ello, a partir de esta tarea **se utilizará el analizador de uso de memoria** `valgrind`.

Esta herramienta debe estar instalada en las máquinas en las que se prueban los programas; está disponible en las máqinas `Linux` de la facultad.

El correcto uso de la memoria será parte de la evaluación.

Por más información sobre la herramienta recurrir al material sobre *valgrind*: Detección de errores en uso de memoria, que se encuentra disponbile en el sitio EVA del curso.

Se debe recordar que:

- **Como cada tarea, es individual y eliminatoria.**
- **La evaluación se hace con casos de prueba que se publican al terminar el plazo de entrega.**
- **Quien no apruebe en la entrega podrá realizar una re-entrega en un plazo de hasta 24 horas después de publicado el resultado.**
- **La aprobación en la entrega otorga 2 puntos; la aprobación en la re-entrega no otorga puntos.**
- **La tarea debe compilar y funcionar correctamente con la regla make testing en las máquinas de la Facultad.**
- **El archivo a entregar se debe generar mediante la regla make entrega y no se le debe cambiar el nombre en el proceso de entrega.**

2. Materiales

Los archivos para la tarea se extraen de *MaterialesTarea2.tar.gz*. Para conocer la estructura de los directorios ver la Sección **Materiales** de Estructura y funcionamiento del Laboratorio.

En esta tarea los archivos en el directorio `include` son **utils.h**, **info.h**, **cadena.h**, **colCadenas.h**, **iterador** y **aplicaciones.h**. Estos archivos no se pueden modificar.

En el directorio `src` se incluyen ya implementados **utils.cpp** e **info.cpp**. Además, se incluyen los archivos **plantilla_cadena.cpp**, **plantilla_colCadenas.cpp**, **plantilla_iterador.cpp** y **plantilla_aplicaciones.cpp**. Estos cuatro archivos contienen las funciones que se deben implementar pero con lo mínimo para que pueda compilar. Se pueden usar para lo cual se les debe cambiar el nombre (esto es remover `plantilla`), poner la cédula (7 dígitos) y completar el cuerpo de las funciones y el struct.

3. Desarrollo

Ver la Sección **Desarrollo** de Estructura y funcionamiento del Laboratorio.

En esta tarea se deben implementar los archivos **cadena.cpp**, **colCadenas.cpp**, **iterador.cpp** y **aplicaciones.cpp**.

La generación del ejecutable se hace compilando los archivos implementados **cadena.cpp**, **colCadenas.cpp**, **iterador.cpp**, **aplicaciones.cpp**, **info.cpp**, **utils.cpp** y **principal.cpp** y enlazando los archivos objeto obtenidos:
El archivo **Makefile** (ver el material disponible sobre Makefile) provee para la compilación la regla **principal**, que es la predeterminada. Por lo tanto el ejecutable se obtiene mediante

```
$ make
```

Se sugiere probar cada uno de los test provistos y al final confirmar mediante la regla **testing**:

```
$ make testing
```

Se debe verificar la compilación y ejecución en las máquinas de Facultad ya que es con esas máquinas con las que se hace la evaluación.

4. Entrega

Ver la Sección **Entregas** de Reglamento del Laboratorio.

4.1. Plazos de entrega

El plazo para la entrega es el **lunes 4 de abril a las 15:00**.

4.2. Archivo a entregar y procedimiento

Se debe entregar el archivo **Entrega2.tar.gz**, que contiene los módulos a implementar **cadena.cpp**, **colCadenas.cpp**, **iterador.cpp** y **aplicaciones.cpp**.
Este archivo se obtiene al ejecutar la regla `entrega` del archivo *Makefile*:

```
$ make entrega
```

Con esto se empaquetan los módulos implementados y se los comprime.

El archivo a entregar **DEBE** ser generado mediante este procedimiento. Si se lo genera mediante alguna otra herramienta (por ejemplo, usando un entorno gráfico) **la tarea no será corregida**, independientemente de la calidad del contenido. Tampoco será corregida si el nombre del archivo se modifica en el proceso de entrega. Si alguna de estas dos irregulardaides ocurre en la entrega la calificación será *Insuficiente* lo que obligará a hacer la reentrega. Si ocurre en la reentrega la calificación será `No Aprobado` lo que implicará la pérdida de la tarea, y por lo tanto del curso.
El archivo que queda en el receptor debe poder descomprimirse con el comando:

```
$ tar zxvf Entrega2.tar.gz
```

4.3. Identificación de los archivos de las entregas

Cada uno de los archivos a entregar debe contener, en la primera línea del archivo, un comentario con el número de cédula del estudiante, sin el guión y sin dígito de verificación.

Ejemplo:

```
/* 1234567 */
```

4.4. Individualidad

Ver la Sección **Individualidad** de Reglamento del Laboratorio.

5. Descripción de los módulos

En esta sección se describen los módulos que componen la tarea. Estos son *utils*, *info*, *cadena*, *colCadenas*, *iterador*, *aplicaciones* y *principal*.

5.1. Módulo *utils*

Igual al de la tarea anterior.

5.2. Módulo *info*

Igual al de la tarea anterior.

5.3. Módulo *cadena*

El mismo concepto que en la tarea anterior pero ahora las secuencias **no son acotadas** y tienen **comportamiento circular**. Se agregan operaciones, en particular la que libera la memoria asignada.

Si la cadena no tuviera comortamiento circular la cadena siguiente de

$$c = [(7, 7{,}00), (3, 3{,}00), (5, 5{,}00)]$$

sería

$$s = [(3, 3{,}00), (5, 5{,}00)]$$

En cambio que tenga comportamiento circular significa que la cadena siguiente es

$$s = [(3, 3{,}00), (5, 5{,}00), (7, 7{,}00)].$$

Las cadenas c y s comparten memoria, por lo que si el elemento $(8, 8{,}0)$ se inserta al inicio de s, obteniendo

$$s = [(8, 8{,}0), (3, 3{,}00), (5, 5{,}00), (7, 7{,}00)]$$

la cadena c queda

$$c = [(7, 7{,}00), (8, 8{,}0), (3, 3{,}00), (5, 5{,}00)].$$

5.4. Módulo *colCadenas*

El mismo concepto que en la tarea anterior. Se agregan operaciones, en particular la que libera la memoria asignada.

5.5. Módulo *iterador*

Un iterador es una secuencia de naturales, de la cual solo se puede acceder a una posición, la *posición actual*. Se puede acceder al elemento que está en la posición actual. La posición actual se puede hacer avanzar, esto es, pasar a la siguiente posición, o volverla al inicio de la secuencia. La posición actual queda indefinida si la secuencia de naturales es vacía, o si estando en la última posición se pasa a la siguiente. Se puede agregar un nuevo elemento al final.
Por ejemplo en el iterador

$$[5, 3, 8]$$

la posición actual permite acceder al 3.
Si se avanza el iterador pasa a ser

$$[5, 3, 8]$$

en donde la posición actual permite acceder al 8.

Si se avanza otra vez la posición actual queda indefinida:

$$[5, 3, 8].$$

En este estado no se puede acceder a ningún elemento.

Si se reinicia, la posición actual permite acceder al 5:

$$[5, 3, 8].$$

Se puede reiniciar aunque la posición actual esté definida. Si se reinicia el iterador

$$[5, 3, 8]$$

su estado pasa a

$$[5, 3, 8].$$

5.6. Módulo *aplicaciones*

Consiste en funciones que se implementan usando las operaciones de los otros módulos.

5.7. Módulo *principal*

Similar al de la tarea anterior al que se agregan comandos para las nuevas operaciones.