Tarea 3
Manejo de un depósito de mercadería
Curso 2022 Contra Semestre

1. Introducción y objetivos

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

Para ello, en 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 máquinas `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:

- **Esta tarea se puede realizar en grupos de hasta 2 estudiantes y es eliminatoria.**
- **La evaluación se hace con casos de prueba que se publican al publicar la tarea y también con casos que se publican al terminar el plazo de entrega, pero que hasta ese momento son privados.**
- **Los grupos que no aprueben en la entrega podrán realizar una re-entrega en un plazo de hasta 24 horas después de publicado el resultado.**
- **La aprobación en la entrega otorga 5 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 *MaterialesTarea3.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 **cadena.h**, **colProductos.h**, **colProductosPorNombre.h**, **controlFecha.h**, **deposito.h**, **fecha.h**, **gestionDeposito.h**, **lista.h**, **listaProductos.h**, **pila.h**, **producto.h**, **utils.h** y **vencimiento.h**.
Estos archivos no se pueden modificar.

En el directorio `src` se incluyen con las representaciones esperadas e implementaciones triviales de los procedimientos y funciones los archivos **colProductosPorNombre.cpp** y **listaProductos.cpp**. Además, el resto de los .cpp se incluyen implementados como se esperaría que lo estuvieran si la tarea 2 hubiera sido entregada con todos los casos de prueba que se liberaron en esa instancia obteniendo los resultados esperados. Recordar completar con la cédula de uno de los integrantes del grupo.

3. Desarrollo

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

En esta tarea se deben implementar los archivos **colProductosPorNombre.cpp** y **listaProductos.cpp**. Además de los archivos anteriores, se deberá realizar los cambios que se consideren necesarios para implementar la nueva funcionalidad que se pide en la letra y para mantener el funcionamiento correcto de las funcionalidades previas.

La generación del ejecutable se hace compilando los archivos .cpp y enlazando los archivos objeto (.h) 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 **Martes 15 de Noviembre a las 18:00hs**.

4.2. Archivo a entregar y procedimiento

Se debe entregar el archivo **Entrega3.tar.gz**, que contiene los archivos **colProductosPorNombre.cpp**, **listaProductos.cpp**. **deposito.cpp** y **gestionDeposito.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 Entrega3.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. Presentación de la realidad

En esta sección de describe la realidad representada en el marco de esta tarea.

La realidad planteada consiste en el modelado de una aplicación para el inventario y gestión de los productos de un depósito. Dentro del depósito se encuentran guardados productos que se identifican por un número y además poseen un nombre. En el depósito puede haber más unidades de un mismo producto. Por ejemplo: un producto puede ser el número 3 y su nombre es Arroz, y las unidades de este producto son todos los paquetes de arroz que se encuentran en el depósito.

Las unidades de los productos se agrupan por fecha de vencimiento y dentro de esta clasificación son indistinguibles. Volviendo al ejemplo del Arroz, este producto podría tener dos unidades con fecha de vencimiento 02/03/2023 y tres unidades con fecha de vencimiento 05/10/2023 totalizando cinco unidades.

Mediante el uso de la aplicación se pretende realizar la alta, baja y modificación tanto de productos como de las unidades de estos que se encuentren en el depósito. Además de estos, se pretende otorgar al usuario ciertos comandos para obtener información del estado actual del depósito desde un punto de vista general como también particular de cada producto.

6. ¿Qué cambia para la tarea 3?

Para la tarea 3, se pide que a las funcionalidades que ya se encuentran implementadas, se le sume la implementación de una forma de obtener la información de los productos buscándolos mediante su nombre. Para buscar los productos por su nombre se crea el comando "vencimientosProductoPorNombre", que se utiliza ingresando el nombre del producto luego del nombre del comando. Para la utilización del comando en esta instancia experimental y dado el poco tiempo que se tiene para implementarlo, se implementará con una restricción: se debe asumir que NO sucederá que un usuario ingrese el nombre de un producto repetido. Si bien hay controles para verificar que no se repita el identificador del producto, estos controles no se han implementado para el nombre del producto.
Para cumplir con este nuevo requerimiento se definieron dos nuevos módulos: **listaProductos** y **colProductosPorNombre**. Ambos módulos se describirán en la siguiente sección. Además de implementar los dos módulos anteriormente mencionados, se deberá realizar los cambios que corresponda en el resto de los módulos que se entregan para que el programa continúe funcionando correctamente y además para que se pueda utilizar la nueva funcionalidad pedida.

7. Descripción de los módulos

En esta sección se describen solamente los nuevos módulos que se incluyen en la tarea 3 y además aquellos que se han cambiado para poder implementar la nueva funcionalidad pedida. Por información sobre los módulos que se presentaron en la tarea 2 se sugiere revisar la letra de esa tarea. Los nuevos módulos que se incluyen en la tarea 3 son: **listaProductos** y **colProductosPorNombre**. Luego, los módulos que cambiaron solamente su especificación son: **deposito** y **gestionDeposito**. Además, el módulo que cambia tanto su especificación como su implementación es **utils**. Finalmente, se cambió también **principal** para incluir el nuevo comando.
En los módulos de especificación como en los de implementación se encuentran comentarios que explican con más detalle qué es lo que se espera de cada una de las operaciones que los componen. Los comentarios pueden incluir, precondiciones, entradas esperadas y el estado de la aplicación luego de la ejecución de dicha operación. Por lo tanto, es recomendable leer bien esos comentarios antes de realizar la implementación de las funciones o procedimientos correspondientes.

7.1. Módulo *colProductosPorNombre*

En este módulo se representa una colección de productos organizados por el nombre del producto. Esta colección está representada como un hash abierto indizado por el nombre de los productos. El índice del producto en el arreglo se calcula en base a una función de hash que se encuentra en el módulo `utils` de nombre `funcionHash`. La función de hash utiliza el nombre del producto para transformarlo en un número que será el lugar que ocupará el producto en el arreglo. La función de hash distribuye los elementos en el arreglo de manera tal que se cumplan las condiciones de orden de hash sin necesidad de realizar reorganización de los elementos. Además de lo anterior, se realizó un estudio apoyado con entrevistas realizadas a las personas que se encargan de llevar el inventario del depósito con la finalidad de identificar un tamaño correcto del arreglo para que también se cumplan las condiciones de orden de hash. El estudio arrojó que el arreglo debe tener tamaño 100. Como ese tamaño es un valor que no va a cambiar luego de que el sistema entre en etapa productiva, se decidió que se guardará en la constante `CANTIDAD_PROD` que se encuentra definida en `utils.h`.

7.2. Módulo *listaProductos*

En este módulo se representa una lista clásica, pero de elementos de tipo Producto. Esta lista sirve como apoyo para la implementación del hash de productos.

7.3. Módulo *deposito*

En este módulo se encuentra representado el depósito. Este módulo fue modificado para incluir dos estructuras en donde se encuentra los productos y además se incluyó una nueva función para mostrar la información de un producto buscándolo mediante su nombre.

7.4. Módulo *gestionDeposito*

En este módulo se realiza la gestión del depósito. Se encuentran implementadas las funciones que controlan y obtienen información del depósito. Para esta tarea se incluyó una nueva función para mostrar la infomación de un producto buscándolo mediante su nombre.

7.5. Módulo *utils*

En este módulo se incluyen nuevas operaciones que pueden ser de utilidad para el desarrollo de la tarea 3.

7.6. Módulo *principal*

En este módulo se incluye un nuevo comando para llamar a la nueva funcionalidad pedida.