Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos, 2019-10
Parcial 1 - 22 de febrero de 2019

Reglas y recomendaciones

Durante el parcial, deben observarse las siguientes reglas:

1. El parcial se debe desarrollar en el computador, escribiendo las respuestas en un archivo de texto plano (TXT). Guarde regularmente su trabajo (para evitar posibles pérdidas de información), con el nombre de archivo
   `par1_apellido1_apellido2_nombre1_nombre2.txt`
   Tenga en cuenta que los nombres y apellidos van todos en minúsculas, los espacios se reemplazan por guiones bajos y no se utilizan ni tildes ni 'ñ'. El archivo de respuestas debe enviarse únicamente a través de la actividad correspondiente en UVirtual.
2. Archivos adicionales se aceptarán sólo en el caso del diseño, de acuerdo a las instrucciones que se encuentran en el enunciado del parcial. El archivo debe guardarse como
   `par1_apellido1_apellido2_nombre1_nombre2_dis.pdf`
   y enviarse junto con el archivo de respuestas a través de la actividad correspondiente en UVirtual.
3. El parcial tiene una duración de dos horas, contadas a partir del inicio normal de la clase. La actividad en UVirtual está programada para cerrarse al terminar las dos horas, por lo que cualquier archivo no entregado dentro de ese lapso se considerará como parcial no presentado y tendrá una calificación de 0.0.
4. El parcial es estrictamente individual y se debe desarrollar únicamente en la sala de computadores del día.
5. Se recomienda no utilizar compiladores, intérpretes de lenguajes de programación o entornos de desarrollo de cualquier tipo.
6. Puede utilizar sus apuntes, libros, e Internet para obtener la información que necesite para el parcial. Sin embargo, está absolutamente prohibido comunicarse con cualquier otro ser humano para obtener información sobre el parcial, a través de cualquier medio (conversación directa, MSN, Skype, gtalk, whatsapp, blackberry messenger, etcétera).
7. Los celulares deben permanecer apagados, y no se debe enviar ni recibir ningún mensaje de texto.
8. La única excepción a lo anterior son los profesores y monitores de la asignatura, quienes sólo responderán consultas respecto a la claridad de las preguntas del parcial y no responderán consultas sobre la materia.
9. Si el estudiante incumple con cualquiera de las reglas, será evaluado con nota 0.0
10. **Recuerde que diseñar no implica implementar.**

"A todos nos gusta ganar, ¿pero a quién le gusta entrenar?"

– Mark Spitz

1 (16%) Análisis de código

Considere la siguiente función (se garantiza que los argumentos contienen la cantidad de elementos necesarios):

```
typedef double                 TScalar;
typedef std::vector< TScalar > TVector;
typedef std::list< TScalar >   TList;
typedef std::list< TList >     TList2;
void func( TList2& l, TVector& v )
{
  v.clear( );
  TList2::const_iterator l2It = l.begin( );
  for( ; l2It != l.end( ); l2It++ )
  {
    if( l2It->size( ) == 0 )
      continue;
    TList::const_iterator l1It = l2It->begin( );
    unsigned int i = 0;
    for( ; i < l2It->size( ) / 2; i++, l1It++ )
    {
      // No se hace nada acá
    }
    v.push_back( *l1It );
  }
}
```

1.1 ¿Qué hace «func»? (descríbala en 12 palabras o menos)

1.2 ¿Cuál es el orden de complejidad de «func»? Justifique brevemente su respuesta.

2 (18%) Selección múltiple con única respuesta

2.1 El orden de complejidad del algoritmo de ordenamiento "burbuja" sobre una lista de enteros es:

1. $O(2^n)$
2. $O(n^2)$
3. $O(n)$
4. $O(\log_2 n)$
5. $O(n \log n)$ o $O(1)$, dependiendo de la implementación interna de la lista, si es una lista encadenada o un arreglo dinámico respectivamente.

2.2 Una lista doblemente encadenada de la STL (`std::list< T >`) es equivalente a un arreglo dinámico (`std::vector< T >`) porque ambas:

1. permiten inserción de un elemento en el inicio de la secuencia.
2. son implementaciones en C++ de una secuencia lineal de datos.
3. ofrecen acceso aleatorio a sus datos.
4. permiten eliminación de un elemento en el inicio de la secuencia.
5. ofrecen algoritmos de orden constante ($O(1)$) para modificar su estructura.

3 (66%) Diseño e Implementación de TADs

MiBodeguita es una empresa que ofrece el servicio de almacenamiento de cajas a diferentes empresas que requieren la distribución de sus productos. La cantidad de mercancía que maneja esta empresa es tal que recientemente vieron la necesidad de utilizar un sistema de información para organizar este almacenamiento, de manera que se pudiera obtener rápidamente la ubicación exacta de una caja particular.

El sistema de información debe organizar la información de las mercancías de acuerdo a las políticas de bodegaje de la empresa, las cuales son:

- Las cajas se organizan una encima de otra, en pilas, con un tamaño (altura) máximo. Al estar apiladas, sólo es posible tener acceso a una caja si se mueven las que están encima de ella. Cada caja tiene como identificador un código (cadena de caracteres) de longitud 6, compuesto por 3 caracteres alfabéticos y 3 números dígitos. Otros atributos de cada caja incluyen el material del que está hecha (cartón corrugado, polietileno, ...) y su clasificación como normal o delicada, dependiendo de la mercancía que contenga.

- Las pilas de cajas se ubican de forma consecutiva, una al lado de otra, en la bodega de almacenamiento. Cada pila se identifica dentro de la bodega con una cadena de dos caracteres que contiene una letra del alfabeto y un número dígito.

De esta forma, se han identificado inicialmente tres grandes procedimientos que se deben realizar frecuentemente en *MiBodeguita*, y que el sistema de información debe entrar a apoyar:

1. Contar todas las cajas almacenadas actualmente en la bodega.

2. Agregar una nueva caja en una pila dada, teniendo en cuenta que cada pila de cajas tiene un tamaño (altura) máximo ya definido. Debe informarse si no es posible agregar la caja a la pila deseada.

3. Buscar cajas cuyo identificador se encuentre en un rango especificado. Las cajas que cumplan esta condición deben presentarse al usuario ordenadas por su código.

Se le pide entonces diseñar e implementar (en C++) los componentes ya descritos del sistema de información para la empresa *MiBodeguita*.

3.1 (20%) Diseño

Diseñe el sistema y el (los) TAD(s) solicitado(s). Recuerde que diseñar es un proceso previo a la implementación, por lo que no debería contener ninguna referencia a lenguajes de programación (código fuente). Para simplicidad del diseño, no es necesario incluir los métodos obtener y fijar (*get / set*) del estado de cada TAD. Para el diagrama de relación entre TADs, anéxelo en formato PDF como parte de su entrega.

Algoritmos

De acuerdo al diseño realizado en el punto anterior, escriba la implementación de los siguientes algoritmos en C++. Estas implementaciones deberán tener en cuenta:

- la definición apropiada de los prototipos de los métodos/funciones (i.e. recibir/retornar los datos suficientes y necesarios para su correcta ejecución),

- el **NO** uso de salidas/entradas por pantalla/teclado (i.e. paso/retorno correcto de valores y/o objetos),

- el correcto uso del (de los) TAD(s) diseñado(s) en el punto anterior, y

- la escritura de todo el código que pueda llegar a necesitar que no esté incluido en la STL (excepto los métodos *get / set* del estado de cada TAD).

3.2 (10%) Conteo de cajas.

3.3 (14%) Agregar nueva caja.

3.4 (22%) Búsqueda de cajas con identificador en un rango.