Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos, 2022-30
Parcial 1 - 24 de agosto de 2022

Reglas y recomendaciones

Durante el parcial, deben observarse las siguientes reglas:

1. El parcial se debe desarrollar en el computador, escribiendo las respuestas en los campos designados dentro del cuestionario de BrightSpace.

2. Archivos adicionales se aceptarán sólo en el caso del diseño, de acuerdo a las instrucciones que se encuentran en el enunciado del parcial. Guarde regularmente su trabajo (para evitar posibles pérdidas de información), con los nombres de archivo
`par1_apellido1_apellido2_nombre1_nombre2.txt` (para el diseño de los TADs) y
`par1_apellido1_apellido2_nombre1_nombre2_dis.pdf` (para el diagrama de relación entre TADs)
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'. Estos archivos deben enviarse únicamente dentro de la pregunta correspondiente en el cuestionario de BrightSpace.

3. El parcial tiene una duración de dos horas, contadas a partir del inicio normal de la clase. El cuestionario en BrightSpace está programado para cerrarse al terminar las dos horas, por lo que no enviarlo dentro de ese lapso de tiempo 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, Teams, Skype, Zoom, Gtalk, Whatsapp, etcétera).

7. También está totalmente prohibido copiar o transcribir texto o código de enunciados y respuestas de parciales anteriores, o de diseños e implementaciones que no sean propias.

8. Los celulares deben permanecer apagados, y no se debe enviar ni recibir ningún mensaje de texto.

9. 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.

10. Si el estudiante incumple con cualquiera de las reglas, será evaluado con nota 0.0

11. **Recuerde que diseñar no implica implementar.**

1 (15%) Análisis de código

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

```
typedef std::vector< char > TVector;
typedef std::vector< TVector > TMList;

TVector func( TMList ml ) {

  unsigned int res = 0;

  for( int i = 0; i < ml.size(); i++ ) {

    TVector::iterator lIt;
    for( lIt = ml[i].begin( ); lIt != ml[i].end(); lIt++ ) {
      if ( *lIt == 101 )   res++;
    }

  }

  TVector vec;
  for( int i = 0; i < res; i++ ) {
    vec.push_back(101);
  }

  return vec;
}
```

1.1 (7%) ¿Qué hace «func»? (descríbala en 12 palabras o menos)

1.2 (8%) ¿Cuál es el orden de complejidad de «func»? Justifique brevemente su respuesta.

2 (24%) Selección múltiple con única respuesta

2.1 (8%) Un algoritmo de ordenamiento tiene dos posibles formas de implementarse: en la primera se tiene un bloque de código de complejidad $O(\log n)$ seguida de otro bloque de código de complejidad $O(n^2)$; y en la segunda se tiene un bloque de código de complejidad $O(n)$ dentro de un ciclo que se ejecuta n veces. ¿Cuál de las dos versiones del algoritmo es más eficiente (tiene menor orden de complejidad)?

1. La primera versión.
2. La segunda versión.
3. Las dos versiones tienen el mismo orden de complejidad.
4. Las dos versiones no son comparables en términos de complejidad.

2.2 (8%) Los contenedores básicos de la STL (`vector`, `deque`, `list`) proveen todos las mismas operaciones, aunque con diversos órdenes de complejidad. ¿Qué operaciones modificadoras tienen el mismo orden de complejidad en los tres contenedores?

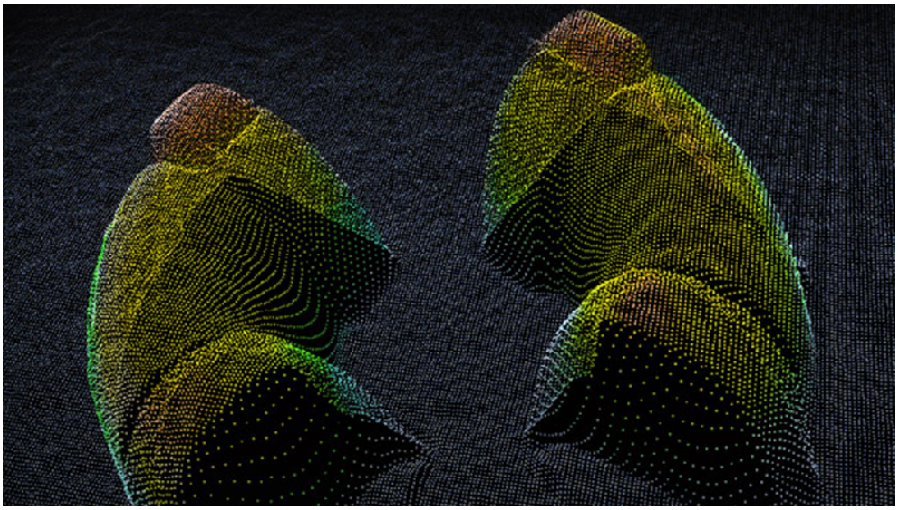
1. `size`, `empty`.
2. `push_front`, `pop_front`.
3. `push_back`, `pop_back`.
4. `insert`, `erase`.

2.3 (8%) Para que un algoritmo diseñado para invertir el orden de los elementos de una estructura lineal sea más eficiente, podría utilizar como estructura auxiliar:

1. Una cola.
2. Una pila.
3. Una lista.
4. Una multilista.

3 (61%) Diseño e Implementación de TADs

Para la visualización en pantalla de objetos en tres dimensiones se utilizan muchas técnicas, una de ellas se conoce como *point rendering* o dibujado de puntos. Así, los objetos están representados con una estructura llama nube de puntos, que básicamente es un conjunto de puntos tridimensionales ($\boldsymbol{p} = [p_x, p_y, p_z]$) que, vistos desde una distancia específica, dan la impresión de representar una figura geométrica. Un ejemplo se puede ver en la siguiente imagen, donde se intenta representar el perfil de un terreno a partir de puntos de diferentes colores:



`https://metrology.news/time-of-flight-camera-generates-3d-point-cloud/`

De esta forma, cada objeto en la escena puede representarse con una nube de puntos, con sus coordenadas tridimensionales. El color de cada punto se representa con la combinación de 3 colores: rojo, verde y azul, los cuales se expresan en niveles de 0 a 255, cada uno. El objeto en la escena también tiene un nombre asociado, que permite identificar lo que la nube de puntos representa. Y para facilitar el dibujo de los puntos en pantalla, se requiere: la información del punto centroide de la nube de puntos (calculado como el punto promedio de todos los puntos que conforman el objeto) y un indicador de visualización que puede tomar tres valores: completa, cuando el objeto cabe completamente en la escena; parcial, cuando el objeto sólo puede visualizarse parcialmente en la escena y nula, cuando el objeto no puede visualizarse dentro de la escena.

La escena corresponde al espacio que alberga todos los objetos, la cual se visualizará en la pantalla. Esta escena tiene unos límites definidos en los tres ejes de coordenadas, que facilita identificar el indicador de visualización de cada objeto: visualización completa se logra si todos los puntos del objeto están dentro de los límites de la escena; se obtiene visualización parcial cuando sólo algunos puntos están dentro de los límites de la escena (y por tanto otros puntos están por fuera de los límites); y la visualización nula implica que todos los puntos del objeto están fuera de los límites de la escena. Los objetos en la escena deben ubicarse u organizarse de acuerdo a la cercanía al observador de la escena: del objeto más lejano al más cercano. De esta forma, al extraer los objetos se garantiza que aquellos más cercanos se dibujan completamente (de últimas), encima de otros más lejanos (que ya se han dibujado primero).

Dentro de la representación computacional de los objetos en una escena tridimensional, se han identificado inicialmente dos operaciones importantes:

1. Actualizar el indicador de visualización para un objeto dado, revisando todos los puntos de su nube con respecto a los límites de la escena a la que pertenece.

2. Crear un nuevo objeto a partir de la unión de las nubes de puntos de dos objetos específicos en la escena. Si hay un punto con la misma coordenada en los dos objetos, se usa sólo el punto del objeto

ubicado más adelante (más cercano) en la escena. Es necesario calcular el centroide del objeto, así como asignar su indicador de visualización. Finalmente, el nuevo objeto debe quedar ubicado adelante (más cerca) de todos los objetos en la escena, para que pueda ser el último en dibujarse.

Se le pide entonces diseñar e implementar (en C++) los componentes ya descritos del sistema computacional de manejo de objetos y escena tridimensionales.

3.1 (16%) Diseño

Diseñe el sistema y el (los) TAD(s) solicitado(s). Utilice la plantilla de especificación de TADs vista en clase para el diseño. Recuerde que diseñar es un proceso previo a la implementación, por lo que no debería contener ninguna referencia a lenguajes de programación (es decir, si escribe encabezados o código fuente, el punto no será evaluado y tendrá una calificación de cero). Para simplicidad del diseño, no es necesario incluir los métodos obtener y fijar (get/set) del estado de cada TAD.

3.2 (7%) Diagrama de relación

El diseño incluye el diagrama de relación entre TADs, cuando se definen dos o más TADs en el punto anterior. En ese caso, adjúntelo en formato PDF, JPG o PNG como parte de su entrega.

3.3 (17%) Operación 1: Actualizar indicador de visualización.

Dado el (los) TAD(s) ya diseñado(s), escriba la implementación en C++ del algoritmo que permite actualizar el indicador de visualización de un objeto dado. La implementación deberá 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 diseño definido en el punto anterior, y
- la escritura de todo el código que pueda llegar a necesitar que no esté incluido en la STL.

3.4 (21%) Operación 2: Unión de dos objetos.

Dado el (los) TAD(s) ya diseñado(s), escriba la implementación en C++ del algoritmo que permite crear un nuevo objeto a partir de la unión de las nubes de puntos de dos objetos dados en la escena. Así como en el punto anterior, la implementación deberá 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 diseño definido en el punto anterior, y
- la escritura de todo el código que pueda llegar a necesitar que no esté incluido en la STL.