Programación 2
Trabajo Obligatorio – CeRP Suroeste - 2016

Generalidades

• Los grupos deberán tener exactamente tres (3) integrantes. En caso de que la cantidad total de estudiantes no sea múltiplo de 3, se admitirá excepcionalmente algún grupo de dos (2) integrantes.

• El trabajo obligatorio consistirá en la resolución de los ejercicios presentados más adelante. Habrá tres (3) fechas de entrega establecidas para los diferentes ejercicios solicitados, las cuales se detallan a continuación:

- 1er Entrega – domingo 31/07/2016: Ejercicios 1, 2 y 3.
- 2er Entrega – domingo 04/09/2016: Ejercicios 4 y 5.
- 3er Entrega – domingo 16/10/2016: Ejercicios 6, 7 y 8.
- Defensa – martes 25/10/2016 de 13:15 a 16:25

Cada entrega deberá ser enviada por mail al docente antes de las 23:59 del día de la entrega. La entrega consiste en un archivo zippeado por cada ejercicio solicitado en la entrega. Cada archivo zippeado debe contener los archivos .h y .cpp correspondientes al ejercicio en cuestión. En el mail deberán figurar los nombres de los integrantes del grupo.

Este trabajo obligatorio tiene carácter domiciliario. Es decir, debe ser realizado fuera del horario de clase. Los estudiantes podrán consultar dudas puntuales al docente fuera del horario de clase o en los momentos de clase destinados a práctico, siempre que ello no perjudique el desarrollo normal de la clase.

Objetivo

- Puesta en práctica de diversos conocimientos adquiridos durante el desarrollo del curso.

Ejercicio 1

Realice un módulo Lista (archivos .h y .cpp), con al menos, las siguientes operaciones sobre listas de números enteros: crear, esVacía, primero, resto, insFront, insBack, desplegar, pertenece, largo, copiarLista. Estas operaciones serán de utilidad en ejercicios siguientes.

Ejercicio 2

1. Realice un módulo GrafoMatAdy (archivos .h y .cpp) correspondiente a un grafo de N vértices (siendo N una constante simbólica) definido mediante la representación de matriz de adyacencia de booleanos. En dicho módulo, implemente las siguientes operaciones:

- Todas las operaciones solicitadas en el ejercicio 1 del práctico 2.
- Una operación que indique si el grafo es simple.
- Una operación que indique si el grafo es completo.
- Una operación que indique si el grafo es regular.

2. Haga un programa de prueba para las operaciones del módulo anterior. Debe brindar un menú de opciones que permita al usuario probar cómodamente las operaciones.

Ejercicio 3

1. Realice un módulo GrafoListAdy (archivos .h y .cpp) correspondiente a un grafo de N vértices (siendo N una constante simbólica) definido mediante la representación de listas de adyacencia.

En dicho módulo, implemente las mismas operaciones solicitadas en el ejercicio 2. Los cabezales de las operaciones deben ser idénticos al ejercicio 2, sólo que ahora la implementación debe realizarse utilizando la representación de listas de adyacencia.

Para manipular las listas de adyacencia, utilice las operaciones auxiliares del módulo Lista definido en el ejercicio 1. Si necesita alguna operación adicional sobre listas, incorpórela a dicho módulo.

2. Haga un programa de prueba para las operaciones del módulo anterior. Este programa de prueba debe ser idéntico al del ejercicio 2, sólo que ahora debe utilizar el módulo GrafoListAdy en vez del módulo GrafoMatAdy.

Ejercicio 4

1. Esta parte se basa en los ejercicios 3 y 4 del práctico 2. Incorpore al módulo GrafoMatAdy del ejercicio 2 las siguientes operaciones:

- void DesplegarVertices (Grafo G)
  // despliega en pantalla los vértices del grafo, recorriéndolos usando DFS.
- int CantidadComponentesConexas (Grafo G)
  // operación solicitada en el ejercicio 1 del práctico 3.
- Lista CaminoMasLargo (Grafo G)
  // operación solicitada en el ejercicio 2 del práctico 3.

Las tres operaciones solicitadas se basan en el algoritmo de DFS. Para cada operación, deberá definir una operación auxiliar que implemente una variante del algoritmo de DFS con un comportamiento adecuado para resolver la operación solicitada.

2. Agregue opciones al programa de prueba del ejercicio 2 que permitan probar estas tres nuevas operaciones.

Ejercicio 5

Ahora incorpore al módulo GrafoListAdy del ejercicio 3 las mismas operaciones solicitadas en el ejercicio 4, esta vez implementadas mediante la representación de listas de adyacencia.

También agregue opciones al programa de prueba del ejercicio 3 que permitan probar estas tres nuevas operaciones.

Ejercicio 6

1. Realice un módulo Asistente (archivos .h y .cpp) conteniendo un tipo estructurado (struct de C++) con las siguientes componentes: número (int), edad (int), cédula (long). El módulo debe proveer las operaciones de carga, de desplegado y selectoras para el asistente.

2. Esta parte se basa en el ejercicio 5 del práctico 4. Realice un módulo Colección de Asistentes (archivos .h y .cpp) conteniendo una estructura de Hash para almacenar a los asistentes de la colección (defina el arreglo de cubetas y la estructura de lista de asistentes para el Hash ambos en el mismo módulo). Suponga que la cantidad de cubetas B = 50 y utilice la función de dispersión h(x) = x % B. En este módulo, implemente las siguientes operaciones y calcule el orden para cada una de ellas:

- void Crear(Hash &H)
  // crea una estructura de hash originalmente vacía.
- boolean Pertenece(Hash H, int clave)
  // determina si en la estructura de hash existe un asistente con la clave ingresada.
- void Insertar (Hash &H, Asistente a)
  // inserta al asistente en la estructura de hash.
  // Precondición: el asistente no existe en la estructura de hash.
- Asistente Obtener (Hash H, int clave)
  // obtiene al asistente de la estructura de hash cuya clave es la ingresada.
  // Precondición: el asistente existe en la estructura de hash.
- void ListarFavorecidos (Hash H, int arreglo[N])
  // operación solicitada en el ejercicio 5 del práctico 4.
- float PromedioEdades (Hash H)
  // operación solicitada en el ejercicio 5 del práctico 4.

Para manipular a los asistentes, utilice las operaciones del módulo Asistente. Si necesita operaciones adicionales para manipular las listas de asistentes de las cubetas del Hash, inclúyalas en el módulo Colección de Asistentes.

3. Haga un programa de prueba para las operaciones solicitadas en el módulo anterior. Debe brindar un menú de opciones que permita al usuario probar cómodamente las operaciones.

Ejercicio 7

1. Resuelva el ejercicio 1 del práctico 5. Trabaje con una Queue de números enteros. Realice un módulo QueueACT (archivos .h y .cpp) para la representación de arreglo con tope y otro módulo QueueLDEPPF (archivos .h y .cpp) para la representación de LDEPPF. Los cabezales de las primitivas deberán ser idénticos en ambos módulos.

2. Construya una tabla comparativa donde se muestre el orden de ejecución de las operaciones primitivas del TAD Queue, para las dos implementaciones realizadas en el punto anterior.

3. Haga un programa de prueba para cada uno de los dos módulos anteriores, conteniendo un menú de opciones que permita al usuario probar cómodamente las operaciones. Ambos programas de prueba deberán ser idénticos, con la única diferencia de que el primero incluye al módulo QueueACT y el segundo incluye al módulo QueueLDEPPF.

Ejercicio 8

1. Resuelva el ejercicio 5 del práctico 5. Debe implementar:

- Un módulo Fecha (archivos .h y .cpp) conteniendo las operaciones de carga, desplegado y selectoras.

- Un módulo Consulta (archivos .h y .cpp) conteniendo las operaciones de carga, desplegado y selectoras.

- Un módulo Historial (archivos .h y .cpp) conteniendo las primitivas del TAD Secuencia utilizando la representación elegida por usted para este ejercicio. También debe incluir una operación que liste por pantalla todas las consultas del historial.

- Un módulo Paciente (archivos .h y .cpp) conteniendo las operaciones de carga, desplegado y selectoras. También debe incluir las siguientes operaciones:

    - void InsertarConsulta (Paciente &p, Consulta c)
      // inserta la consulta al historial de consultas del paciente.

- void ListarConsultas (Paciente p)
  // lista por pantalla las consultas del historial del paciente.

- Un módulo Pacientes (archivos .h y .cpp) conteniendo las primitivas Make, Member, Find, Insert, Modify, del TAD Diccionario utilizando la representación elegida por usted para este ejercicio.

- Un módulo Sistema (archivos .h y .cpp) conteniendo las cuatro operaciones que dan solución a los requerimientos solicitados en el ejercicio 5 del práctico 5. Debe invocar a las operaciones definidas en los módulos anteriores.

  - void RegistrarPaciente (Paciente p, Pacientes &d, boolean ok)
    // ok queda en false si el paciente a ingresar existía previamente en el sistema.

  - void NuevaConsulta (long ced, Pacientes &d, Consulta c, boolean &ok)
    // ok queda en false si el paciente con cédula ced no existe en el sistema.

  - void ListarHistorial (Pacientes d, long ced, boolean &ok)
    // ok queda en false si el paciente con cédula ced no existe en el sistema.

  - void ListarPacientes (Pacientes d)
    // lista por pantalla los pacientes ordenados por cédula de menor a mayor.

2. Haga un programa principal que brinde al usuario un menú de opciones para ejecutar los cuatro requerimientos solicitados. Debe invocar a las operaciones definidas en el módulo Sistema.