Universidad Simón Bolívar
Departamento de Computación y T.I.
CI-3825, Sistemas de Operación I
Enero-Marzo 2013

Tarea (5 %)

1. Objetivos Generales

Familiarizarse con los aspectos básicos de programación en lenguaje C.

2. Objetivos Específicos

1. Adquirir destrezas en el lenguaje C.
2. Adquirir destrezas en la definición de estructuras compuestas y tipos.
3. Adquirir destrezas en el uso y manejo eficiente de memoria dinámica.

3. Definición del Problema

Trees on the level
Tomado del ACM Programming Contest

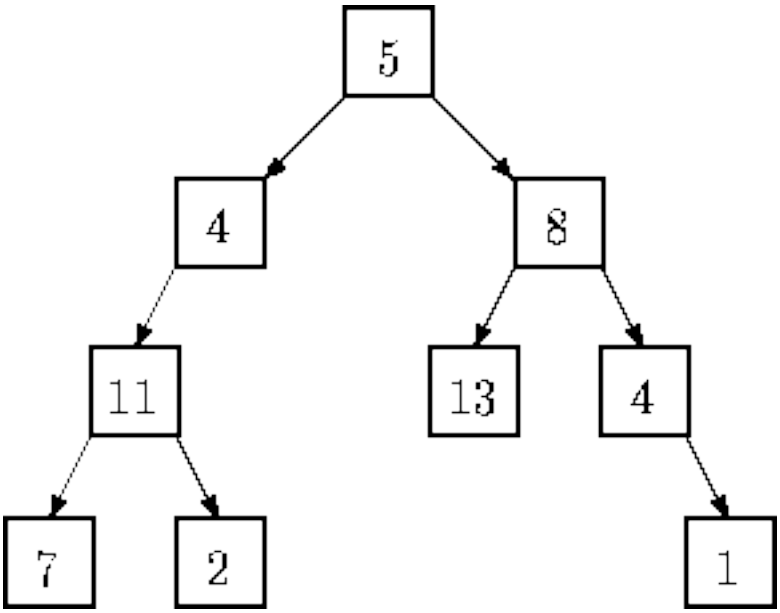
La estructura de árbol es fundamental en muchas ramas de la ciencia de la computación. Este problema involucra construir y recorrer árboles binarios.

3.1. El problema

Dada una secuencia de árboles binarios, debe escribir un programa que imprima un recorrido por nivel de cada árbol. En este problema cada nodo del árbol binario contiene un entero positivo y todos los árboles binarios tienen a lo sumo 256 nodos.

En un recorrido por nivel, los datos en todos los nodos de un nivel son impresos en orden de izquierda a derecha y todos los nodos de nivel k se imprimen antes que los nodos del nivel $k + 1$.

Por ejemplo, un recorrido por nivel del árbol



es: 5, 4, 8, 11, 13, 4, 7, 2, 1.

En este problema un árbol binario está especificado por una secuencia de pares (n, s) donde n es el valor en el nodo cuyo camino desde la raíz está dado por la cadena de caracteres s. Un camino esta dado por una secuencia de I's y D's donde I indica una rama izquierda y D indica una rama derecha. En el árbol diagramado anteriormente, el nodo que contiene 13 está especificado por (13,DI), y el nodo que contiene 2 está especificado por (2,IID). La raíz está especificada por (5,) donde la cadena vacía indica el camino desde la raíz a si misma. Un árbol binario es considerado *completamente especificado* si cada nodo en todos los caminos desde la raíz al nodo en el árbol se le da un valor exactamente una vez (el árbol del diagrama anterior es completamente especificado).

Los árboles binarios deben representarse haciendo uso de estructuras y apuntadores.

3.2. La entrada

La entrada viene especificada en un archivo de texto. El nombre de este archivo será recibido como argumento a la hora de llamar su programa. La sintaxis para la invocación del programa será:

tarea archivo_de_entrada

La entrada es una secuencia de árboles binarios especificados como se describió anteriormente. Cada árbol en una secuencia consiste de varios pares (n, s) como se describió anteriormente separados por espacio en blanco. La última entrada de cada árbol es (). No hay espacio en blanco entre el paréntesis izquierdo y el derecho.

Todos los nodos contienen un entero positivo. Cada árbol en la entrada consistirá de al menos un nodo y no más de 256 nodos. La entrada se termina

por fin de archivo (EOF).

3.3. La Salida

Para cada árbol binario completamente especificado en el archivo de entrada se debe imprimir por pantalla el recorrido por nivel de ese árbol. Si un árbol no está completamente especificado, es decir, a algún nodo en el árbol no se le da un valor o a un nodo se le da un valor más de una vez, se debe imprimir «incompleto». Se imprimirá un resultado por línea

3.4. Entrada ejemplo

(11,II) (7,III) (8,D)
(5,) (4,I) (13,DI) (2,IID) (1,DDD) (4,DD) ()
(3,I) (4,D) ()

3.5. Salida Ejemplo

5 4 8 11 13 4 7 2 1
incompleto

4. Recomendaciones

1. Diseñe su solución completa antes de proceder a implementar.
2. Siga las recomendaciones de estilo publicadas en la sección de documentos de Aula Virtual.
3. Trabaje en forma ordenada e incremental.
4. Pruebe que cada una de sus funciones opera correctamente.
5. Estructure bien su código.
6. Tenga presente que es mejor tener más funciones pequeñas que menos funciones largas.
7. Realice la documentación de su código a medida que vaya programando. Dejarlo para el final se traduce en invertir más tiempo para hacerlo.
8. Tome en cuenta el uso y manejo eficiente de la asignación dinámica de memoria usando las funciones *malloc* y *free*.

5. Entrega de la Tarea

Lugar: Salón de clase.
Hora: 1:30 pm del lunes de la semana 4.

La entrega de la tarea debe hacerse antes de la hora de clase del lunes de la semana 4. Usted deberá colocar el archivo con su tarea en Aula Virtual, para lo cual deberá crear el directorio Tarea dentro de la carpeta documentos de su grupo. Note que debe estar suscrito a algún grupo en Aula Virtual para poder optar a esta opción, no espere al día de la entrega para notificar que tiene problemas o que no se ha registrado. En este directorio colocará el archivo *tar.gz* que contenga los fuentes de su programa. No se corregirán proyectos que no sean colocados en esta forma.

Adicionalmente, ese día usted debe entregar al comienzo de la hora de clase el código fuente impreso en un sobre manila identificado con los nombres de los integrantes del grupo y número del grupo. El código debe tener una cantidad adecuada de comentarios. Siga la guía de estilo de programación en lenguaje C publicada en la sección de documentos.

6. Notas importantes

1. Tarea que no sea entregada en el lugar, fecha y hora prevista, NO será recibida.

2. Proyecto que no cumpla con alguna de las especificaciones establecidas en este enunciado corre el riesgo de no ser corregido.

3. El código debe estar documentado y debe generar la documentación en html con doxygen, vea el ejemplo que está en la carpeta Documentos - Ejemplos - ejemplo_valgrind_doxygen.tar.gz.

4. Verifique que no ocurran fugas de memoria utilizando valgrind. Hay un ejemplo en el archivo mencionado en el punto anterior.

5. Evite que ocurran advertencias (warnings) en la compilación de su código.

6. Su tarea debe compilar y funcionar en las computadoras del LDC.