

CIS 213: Foundations of Computer Science—Lab 8

Binary Search Trees and DDD

Goals: To demonstrate use of Binary Search Tree functions. Do traversals. Show a tree graphically using a graphical debugger: ddd.

Create a lab8 directory in your cis213 directory. Goto your lab8 directory and Copy the file from the lab directory, `~hardnett/pub/cis213/lab7` into your lab7 directory. The file list is:

```
traverse.cc    // Tree creation/traversal program.
makefile      // to compile lab5
BSTree.h      // header file for Binary Search Tree
BSTree.cpp    // implementation file for Binary Search Tree
```

Reading Code

Look at the main file for the program. Note that you'll need to see how the data structure is defined, so track it down via the include files. This is an exercise in finding definitions.

You'll see that tree nodes are defined as

```
struct treeNode
{
    treeItemType Item;
    ptrType      LChildPtr, RChildPtr;

    // constructor:
    treeNode(treeItemType NodeItem, ptrType L, ptrType R);
}; // end struct
```

You'll also need to find what the "functionType" is. What is declared to be of type functionType? What is functionType? Track it down and you'll see that it is:

```
typedef void (*functionType)(treeItemType& AnItem);
```

What does that mean?

Running the Code

Compile and run the program. Use "make" to compile it. Read the makefile and check your understanding.

The program asks for characters to put in a Binary Search Tree, ending with a '\$'. Use `cbafd$` for your first example.

What does this binary search tree look like? Sketch out what it looks like after each letter is added. Convince yourself that the output is correct.

This program uses a `GeneralOrder` traversal. It does 3 operations: when first getting to a node, after going left, and after going right.

Find the code for the `GeneralOrder` member function. How does it work?

Using DDD

Run a debugger called ddd:

```
ddd lab5
```

When it starts up you will see 3 (overlapping) windows:

- A Console window in which you can type commands or inputs (for "cin"s).
- A Command window with the usual buttons like "Run" "Break" etc.
- A Program Source window with the source of the program in it.

1. PLEASE move the "Source" window so it is not hiding the other two windows.
2. In the "Source" window, click on the LINE NUMBER 36 in MyVisit() and click on "Break at ()" at the bottom of the source window. (You should soon see a stop sign.)
3. Click on Run in the command Menu.
4. After the program starts, type in `cbafd$` to add to the tree. Do this in the "Debugger Console" window. (All input/output of your program belongs there.)
5. When it stops, in the Source window select the variable "MyTree" on line 29 and click on Display(). You should see something interesting in the (new) Program Data window. Do the same for TheRecord.Key on line 43.
6. Select "Options/Preferences/Data Prefs." and select "Automatic Layout". (Any window.)
7. Select the "Root" line in the box in the Data window. And click on Display*() or use the right button to get a menu to do Display*.
8. Do the following until bored:
9. Select a childptr in a box and Display* it.
10. Select an Item in a box and "Show Detail" it. (Right button.) Do the same for TheRecord.Key.
You may need to resize/scroll the window after a while.
11. Click "Cont" on the Command Menu to continue execution. Each time it stops it is visiting a node for the 1st/2nd/3rd time. Watch the Debugger Console window and note when the value of the node get printed.
After all nodes in the tree have been visited, the program will next do a InOrder traversal and then a PostOrder traversal.
12. To quit: select "File/Exit"

The ddd program will be very useful to you when you are implementing any program that uses pointers, such as tree programs and linked-list programs. I urge you to experiment with it. Learn the tools of your trade, and programming will be much easier.