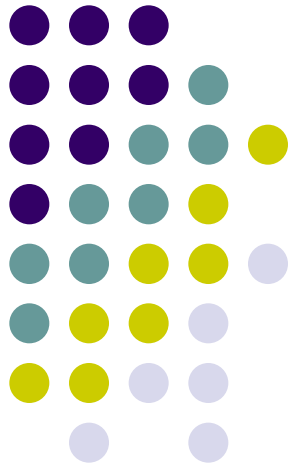
Teach-Me-CS-213

CS 293 Final Project Demo

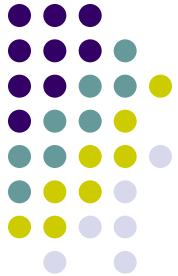
25th November 2012

Sudipto Biswas, 110050048

Abhilash Gupta, 110050058

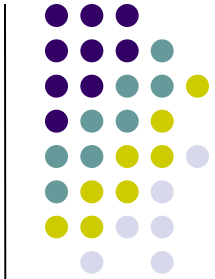


Outline <for a total of 30 mins>



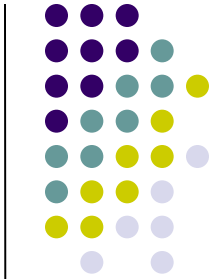
- *Aim of project (1 mins)*
- *Demo (5 mins)*
- *Teamwork Details (0.5 min)*
- *Design Details –Algorithm (5 mins)*
- *Design Details – Implementation (8 mins)*
- *Viva (9 mins)*
- *Transition time to next team (2 mins)*

Aim of the project



- *To help students learn about the ADTs better with real time examples which demonstrate how various inbuilt function of various ADTs work. This project demonstrates the working of List, Stack, Queue, Heap, Binary Search Tree and AVL Trees which coupled with the teacher's explanation will help the students learn CS 213 better.*

Demo



- *Now we will explain the program further while doing the demo.*
- ***NOTE*** *: All the theory in the theory boxes of all the dialogs has been taken from Wikipedia.*

Teamwork Details

- ***Sudipto*** *: Development of classes with extra coupled features so as to facilitate their demonstration and Qt implementation of them. Worked on the Tree structure and heap structure and selection sort. More Coding oriented.*
- ***Abhilash*** *: Development of painter function for most of the ADT implementations. Worked with stacks, queues, lists and insertion sort. Development of connection of windows and dialogs and various other features of UI. More GUI oriented.*

Overall Contribution by Sudipto	*Overall Contribution by Abhilash*
50%	*50%*

Design Details

Algorithm *: The main requirement of algorithm was for the working of "paintevent" function of any Qt window for graphical representation of the data structure. So we worked on understanding how can we draw rectangles/lines and various other figures and how to develop such an algorithm which will be able to draw the desired representation from the given data structure. The paintevent function uses the class Qpainter which paints the various required objects such as Rectangles, Lines, text, etc. Since paintevent is a state based painting facility i.e. it does not update a particular part, we need to update and draw the whole figure again whenever any change is made.*

Design Details

- ***Implementation***

  PaintEvent is the main function for graphical representation of the given data structure.

  'Update' removes all the previously drawn figures on the GUI window for reconstruction of the figure according to updated information whenever PaintEvent is called.

  Used Qpainter class of Qt to paint the structures

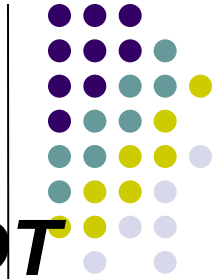
  Used QmessageBox class to show theory.

  Used Qpushbutton for the button facility while Qlineedit for taking inputs in lineEdit and Qlabel for depiction of special features.

  Used QtGui and QtCore (includes Qrect,Qfont, Qpen,

  and all such various classes)

Algorithm Design



- ***How to draw any state of a non-linear ADT on the Qt Window/Dialog?***
- *The key to this problem lies in "struct" provided by C/C++ language. Combining a given data value with appropriate coordinates by using a struct of value and coordinates and then drawing rectangles at those coordinates by using simple Qpainter functions.*

(Eg: use of inbuilt data members for coordinates in BST and AVL trees)

- *The main algorithm part was to keep on updating the coordinates of every data value whenever some functionality of the given ADT is shown which might affect the other coordinates.*

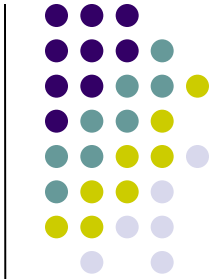
Algorithm Design

- ***How to draw any state of a linear ADT on the Qt Window/Dialog?***
- *For list, stacks, and queue which are linear, we just kept using a variable to get the co-ordinates of its elements and then incrementing the variable.*

Special case- Heap. As a heap can be stored in a vector so we made another vector which stores a struct of co-ordinates for further use as its visual demonstration is not linear but that of a tree.

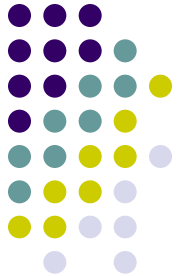
- *The main algorithm part was to keep on updating the coordinates of every data value whenever some functionality of the given ADT is shown which might affect the other coordinates.*

Class Design – High level



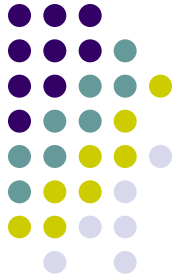
- *Preferably make a picture (e.g. like the DES class design that I presented in class – NOT like your DES submission)*
- *You may use multiple slides for initial "high level" explanation of implementation*
- *Ensure you present any "cool implementation tricks", any special data structures, any design trade-offs done*

Class Design – High level



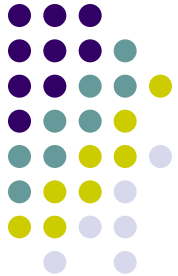
- ***QMainWindow*** *: The main window object that is created at the beginning is of this class. This is the parent window to which all the dialogs are connected.*
- ***QdialogBox*** *: All the other windows that open on request of that particular sort/ADT are of this class DialogBox. It felicitates Painting and organising the whole project because there can be only one mainwindow for main to call which can call these other windows(called dialogs).*

Class Design – High level



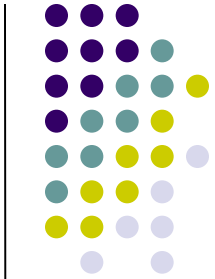
- ***PaintEvent*** *: This is the main function of Qt we have used for painting the window/dialog to represent the data structure correctly. This forms the base constructor for the Painter object in which we draw the structure. It inherits QpaintEvent which we overrode to work according to our needs.*

Class Design – High level



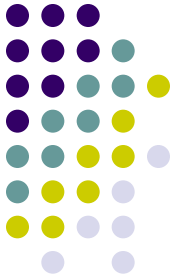
- ***Qpainter*** *: This is the class used extensively for painting different structures on the Window/Dialog. All other Qt classes such as Qrect, Qpolygon, Qlabel, Qpen , Qfont, Qcolor, Qbrush etc were called alongwith this class to facilitate manipulation of the paintings.*
- ***QmessageBox*** *: This is a class we used as a pop-up window to display the Basic theory about the ADT/Sort which comes with a 'information' symbol.*

Class Design – High level



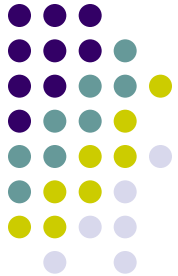
- ***QpushButton*** *: This class is used in all the Applications to create the push buttons objects on any UI, thus has been used throughout our application.*
- ***QLabel*** *: This class has been used to show details on the window on call of pushbutton without changing the other paintings of paintevent which is a state based painter.*
- ***QlineEdit*** *: This class has been used to create input taking objects called lineEdits which takes input from the UI itself.*

Class Design – High level



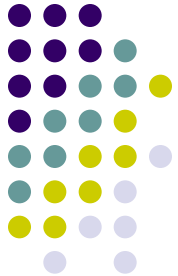
- ***Qrect*** *: This creates a rectangle object with the coordinates of upper corner and the length and breadth for the Qpainter to draw.*
- ***Qpolygon*** *: This has only been used in our application to create a triangle triangle which will be drawn at the tip of line to demonstrate pointer arrows in the depiction in List.*
- ***QpainterPath*** *: Fills a closed path according to the color and brush given to it.*
- ***Qbrush*** *: Is a brush used for painting used in filling the rectangles and polygon.*

Class Design – High level



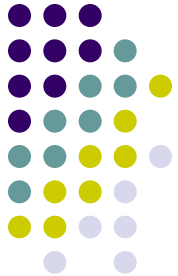
- ***Qpen*** *: Class which sets the pen for drawing figures and writing texts.*
- ***Qfont*** *: Class which sets the Font details for the Qpen class.*
- ***Qcolor*** *: Class which sets the color details for the Qpen class and Qbrush class.*
- ***Qstring*** *: Class which handles the string details for Qt's various usage. Qt's version of std:: string.*

Class Design – High level



- ***QtGui*** *: Class which inherits all the Graphics classes and libraries and including this library along with QtCore has saved us from including all the classes mentioned above and hence a lot of probable possible mistakes.*
- ***QtCore*** *: Class which inherits all the inbuilt Qtcore classes and libraries including Qstring, Qobject, Qwidget etc.*

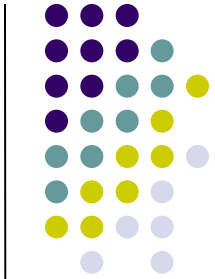
Class Design – High level



- ***Slots and Signals:***

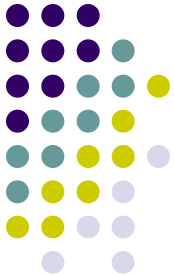
The signals and slots mechanism is a central feature of Qt and probably the part that differs most from the features provided by other frameworks. This is the in-built mechanism by which different objects interact with each other in a given frame. All the pushbuttons all over the dialogs and main window work together with the data values in all the line-edits just by this mechanism of slots and signals. One object sends a signal to another object's slot. Slots can be defined accordingly which are generally a function which will be executed once the signal is received.

Class Design - Details



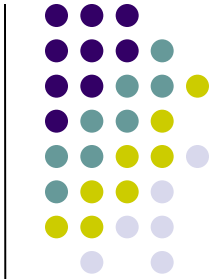
Class Name	Brief Description
AVL	*It is the user defined data structure of AVL Trees which we have written to visually demonstrate the insert functionality of AVL Trees/ Balanced Binary Search Trees. It stores the pointer to the root of the tree in its data members and uses few helper functions to implement the Insert method.*

Class Design - Details



BinaryTree	*It is again the user defined data structure of Binary Search Trees. It stores the pointer to the root node and has the different methods we wanted to showcase in our demonstration of Binary Search Tree as an ADT.*
Heap	*Heap class is again defined to bind in the heap array and heap functions together. And to note the changes made by heap methods on the original heap data structure.*

Element Access Design - Details

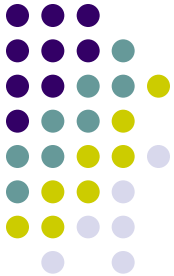


Pointers	*For accessing the elements of the Class defined by us for BST and AVL trees, Pointers have been used.*
Iterators	*For accessing the elements STL used for List,Stack,Queue, Iterators have been used.*
Direct Access	*For the heap implementation as a vector was used as the storage container, direct access is used using an index has been used. Same for Insertion sort and Selection sort.*

Data Structures Used

Purpose for which data structure is used	Data Structure Used	Whether Own Implementation or STL
Implementation of Stack	*List*	*STL*
Implementation of Queue	*List*	*STL*
Implementation of MaxHeap	*Priority Queue implementation using vector*	*Own*
Implementation of List	*List*	*STL*

Data Structures Used



Implementation of Elementary Sorting algorithms	*Vector*	*STL*
Implementation of AVL tree	*AVL Tree implementation using BST*	*Own*
Implementation of BST	*BST*	*Own*
For saving coordinates to draw a heap	*Vector*	*STL*

Source Code Information

File Name	Brief Description	Team Member
StackDialog.h	Declares QDialog Class for implementation of stack and its methods	Abhilash
StackDialog.cpp	Implements the above class methods	Abhilash
QueueDialog.h	Declares QDialog Class for implementation of stack and its methods	Abhilash

Source Code Information

File Name	Brief Description	Team Member
QueueDialog.cpp	Implements the QueueDialog class methods	Abhilash
ListDialog.h	Declares QDialog Class for implementation of list and its methods	Sudipto
ListDialog.cpp	Implements the ListDialog class methods	Sudipto

Source Code Information

File Name	Brief Description	Author (Team Member)
HeapDialog.h	*Declares QDialog Class for implementation of Heap and its methods*	*Abhilash*
HeapDialog.cpp	*Implements the Heap QDialog class methods*	*Abhilash*
Heap.h	*Declares and Implements Template Heap (Maxheap) Class and its methods*	*Sudipto*

Source Code Information

File Name	Brief Description	Team Member
BinaryTree.h	Declares BinaryTree Class for implementation of BST and its methods	Sudipto
BinaryTree.cpp	Implements the BinaryTree class methods	Sudipto
BstDialog.h	Declares QDialog Class for implementation of BST and its methods	Sudipto

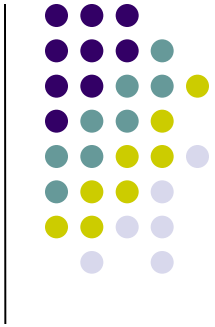
Source Code Information

File Name	Brief Description	Team Member
AVL.h	*Declares AVL Class for implementation of AVL Tree and its methods*	*Sudipto*
AVL.cpp	*Implements the AVL class methods*	*Sudipto*
AvlDialog.h	*Declares QDialog Class for implementation of AVL class and its methods*	*Sudipto*

Source Code Information

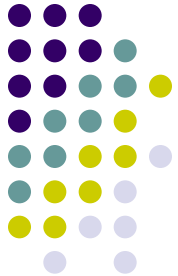
File Name	Brief Description	Team Member
AvlDialog.cpp	Implements the Avl QDialog class methods	Sudipto
Mainwindow.h	Declares QMainWIndow Class for implementation of MainWindow class	Abhilash
MainWindow	Implements the MainWindow class to connect to whichever ADT required	Abhilash

Source Code Information



File Name	Brief Description	Team Member
sortdialog.h	Declares the Sort QDialog class methods	Sudipto
sortdialog.cpp	Implements the Sort dialog class methods declaredd in the sortdialog.h	Sudipto

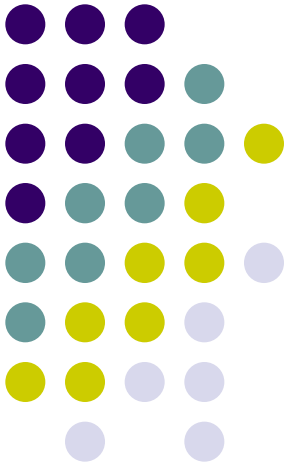
Brief Conclusion



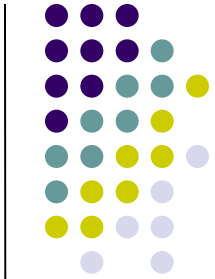
- *Our project Teach-Me-CS-213 thus has been successful in visually depicting the primary **6 ADTs** and the **2 elementary Sort Algorithms** taught in the course properly on a window/dialog. Along with the small Theory Messages provided with each dialog and the professor's explanations, we think this project would be able to provide a more clearer understanding of the course.*

Thank You

Questions?



Back Up Slides



- *Small Qt classes used all over-*

QtGui, QtCore, Qstring, Qrect, Qpen, Qfont, Qcolor, QLineEdit, QpushButton, Qlabel, QPolygon, QPainterPath, Qbrush.