MID SEM REPORT

on

Visualization of Linear and Non-Linear Data Structure Using OpenGL

Submitted by

Aman Sharma (Enroll No. R110218018)

Gaurvendra Singh (Enroll. No. R110218056)

Divyanshu Pande (Enroll. No. R110218054)

Under the guidance of

Ms. Ambika Aggarwal

Assistant Professor (SG)
Department of Virtualization

UPES

UNIVERSITY WITH A PURPOSE

SCHOOL OF COMPUTER SCIENCE
UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Bidholi Campus, Energy Acres, Dehradun – 248007.

Aug- Dec (2020)

UPES

School of Computer Science

University of Petroleum & Energy Studies, Dehradun

Project Proposal Approval Form (2020)

Minor | I

PROJECT TITLE: Visualization of linear and non-linear data structure using OpenGL.

ABSTRACT:

This project was inspired by this pandemic situation and online teaching mode, so we are creating a GUI based application to visualize linear and non-linear data structure. A lot of beginners and experienced programmers avoid learning Data Structures and Algorithms because it's complicated and they think that there is no use of all the above stuff in real life. Data structures and algorithms play a major role in implementing software. Knowledge of data structures like Linked list, Trees, Graphs, and various algorithms goes a long way in solving the problems efficiently and the interviewers are more interested in seeing how candidates use these tools to solve a problem. Just like a car mechanic needs the right tool to fix a car and make it run properly, a programmer needs the right tool (algorithm and data structure) to make the software run properly. So, to understand them efficiently and without any mistake, we need to visualize them.

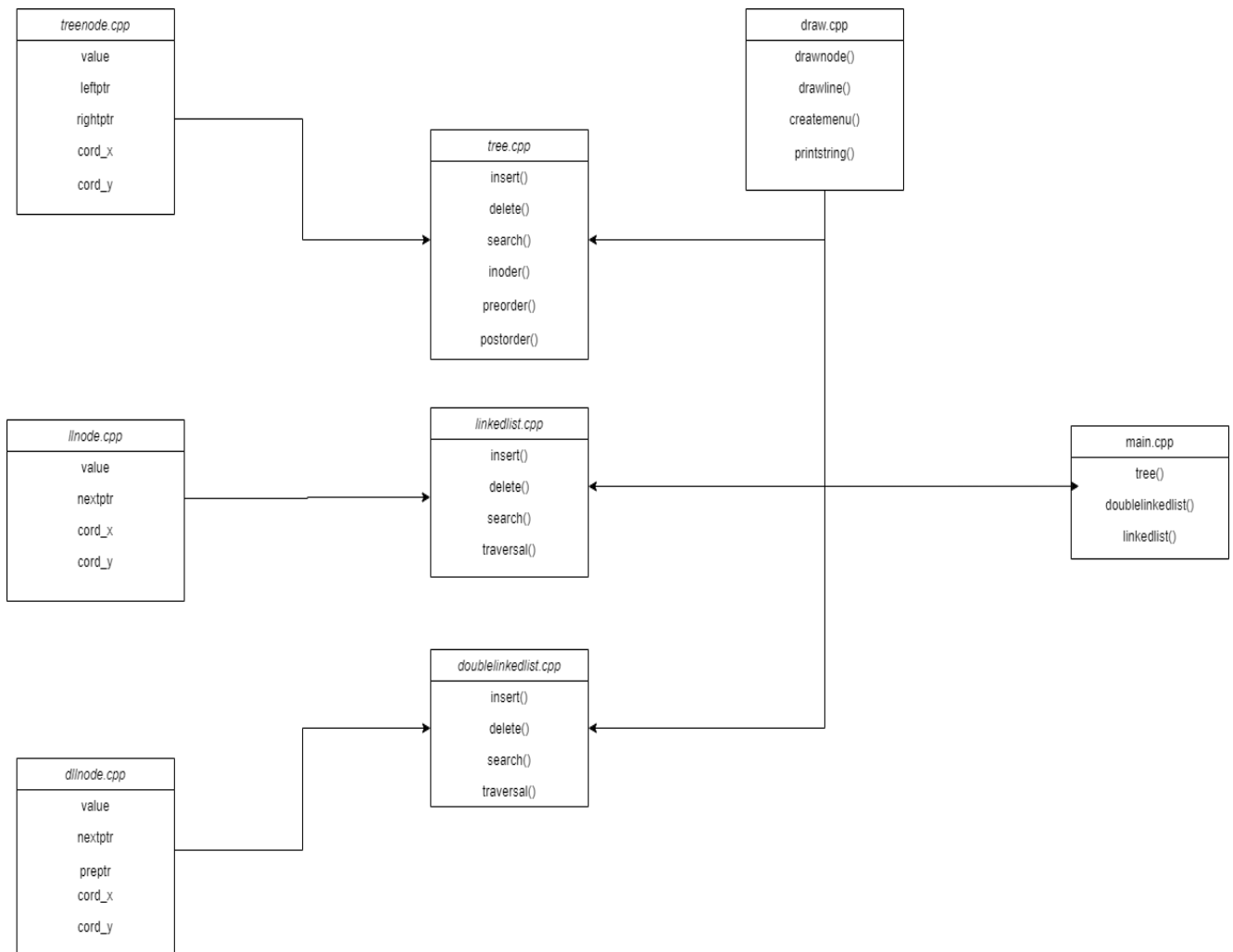KEYWORDS: DATA STRUCTURES, TREES, GRAPH, ALGORITHMS, LINKED LIST.

INTRODUCTION:

In this project, we shall visualize the linear and non-linear data structure using OpenGl A.P.I. We will use the C++ programming language to implement data structures part and various functions and we will use OpenGL Utility Toolkit (GLUT: a window system independent toolkit for writing OpenGL programs) for making the GUI. The Data structures that we will implement are:

- Linear: - Linked List (with all functions).
- Non-Linear: - Binary Search Tree (with all functions and traversals).

All the functions like insertion, deletion, etc. will take input value from the user and the user will have the option to select the data structure.

CLASS DIAGRAM:

treenode.cpp
value
leftptr
rightptr
cord_x
cord_y

tree.cpp
insert()
delete()
search()
inoder()
preorder()
postorder()

draw.cpp
drawnode()
drawline()
createmenu()
printstring()

llnode.cpp
value
nextptr
cord_x
cord_y

linkedlist.cpp
insert()
delete()
search()
traversal()

main.cpp
tree()
doublelinkedlist()
linkedlist()

doublelinkedlist.cpp
insert()
delete()
search()
traversal()

dllnode.cpp
value
nextptr
preptr
cord_x
cord_y

PROBLEM STATEMENT:

Visualization of linear and non-linear data structure.

OBJECTIVES:

To create a GUI based application that visualizes the data structures.

- ➢ Sub Objectives:
    - Creation and visualization of Linked List.
    - Creation and visualization of Binary Search Trees.
    - Deleting nodes and Inserting nodes.
    - Traversal of the data structure.

METHODOLOGY:

We are creating a program in which the user will be given a menu in which he/she will have the option to select the data structure to visualize. After this, a window will open where the user will have the option to insert, delete, and traverse the data structure. Users will also have the option to exit the program.

The entire implementation of this project can be summarized in the following steps:

1) Taking the input from the user for data structure selection from a user-defined menu.
2) After this, the user will be directed to a GUI window with various options.
3) Then taking input values for the functions like the insertion of nodes, deletion of nodes, and the traversal on a GUI based window.
4) After performing the function, the data structure will be visible in the GUI window in an understandable format.
5) After this user can again perform the function or can exit the program.

SYSTEM REQUIREMENTS:

- ➢ Hardware:

    - RAM: 4GiB

    - Disk Space: 1GiB (min)

- ➢ Software:

    - Visual Studios

- ➢ Operating System:

    - LINUX/WINDOW/OS X

- ➢ Resources:

    - OpenGL API

LITERATURE REVIEW:

Here is the conclusion of some of the reference paper and the links that we review to make our project better and to know more technologies that we can use in our system:

- Using [3], we learned about different data structures, the uses, and various functions, and how to write the code efficiently. In [2] we learned about advance concepts of C/C++ programming like containers.
- OpenGL (GLUT) is very hard to understand. [1] provides a very easy to understand approach for implementing GLUT functions and [4] provides various concepts and ideas to implement various visualization tasks.
- Mathematics behind visualization is difficult. The paper [5] Highlights the idea of using vector and coordinates system to give the accurate position of the node after performing various functions.

SCHEDULE (PERT Chart):

SEPTEMBER	OCTOBER	NOVEMBER	DECEMBER
Project Proposal and Synopsis	Requirement Gathering, Analysis and Design	Coding, Build and Testing/Validation	Project Submission and Presentation
Assigned to:- Aman, Gaurvendra, Divyanshu.	Assigned to:- Aman, Gaurvendra, Divyanshu.	Assigned to:- Aman, Gaurvendra, Divyanshu.	Assigned to:- Aman, Gaurvendra, Divyanshu.

START → ... → END
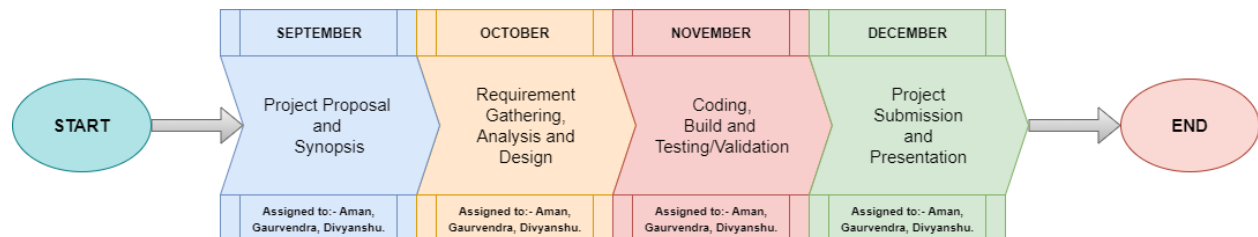
CODE AND OUPUT:

Here is the Implementation code of binary search tree visualization using predefined input.

first2.cpp

```cpp
NODE insert(int item)
{
    //draw_tree();
    int num_of_node=1;
    int node_x=root_centre_x,node_y=root_centre_y;
    int par_x,par_y;
    NODE temp,par=NULL,new_node;
    if(root==NULL){
        root=(NODE)malloc(sizeof(struct node));
        if(root==NULL){
            glColor3f(1.0,0.0,0.0);
            //display_string(not_found,100,600);
            return NULL;
        }
        root->data=item;
        root->left=root->right=NULL;
        glColor3f(0.8,0.5,0.3);
        draw_node(item,node_x,node_y);
        return root;
    }
    temp=root;
    while(temp!=NULL){
        num_of_node*=2;
        par_x=node_x;
        par_y=node_y;
        node_y=node_y-100;
        par=temp;
        if(item<temp->data){
            temp=temp->left;
            node_x=-root_centre_x/num_of_node;
        }
        else{
            temp=temp->right;
            node_x=node_x+root_centre_x/num_of_node;
        }
    }
    new_node=(NODE)malloc(sizeof(struct node));
    if(new_node==NULL){
        printf("there is an error in creating node\n");
        return NULL;
    }
    new_node->data=item;
    new_node->left=new_node->right=NULL;
    if(item<par->data)
        par->left=new_node;
    else
        par->right=new_node;
    glColor3f(0.8,0.5,0.3);
    draw_node(item,node_x,node_y);
    draw_arrow(par_x,par_y,node_x,node_y);
    glFlush();
    return root;
}
void display()
```

cout.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
#define gc getchar_unlocked
#define fo(i,n) for(i=0;i<n;i++)
#define Fo(i,k,n) for(i=k;k<n?i<n:i>n;k<n?i+=1:i-=1)
#define ll long long
#define si(x)   scanf("%d",&x)
#define sl(x)   scanf("%lld",&x)
#define ss(s)   scanf("%s",s)
#define pi(x)   printf("%d\n",x)
#define pl(x)   printf("%lld\n",x)
#define ps(s)   printf("%s\n",s)
#define deb(x) cout << #x << "=" << x << endl
#define deb2(x, y) cout << #x << "=" << x << "," <<
#define pb push_back
#define mp make_pair
#define F first
#define S second
#define all(x) x.begin(), x.end()
#define clr(x) memset(x, 0, sizeof(x))
#define sortall(x) sort(all(x))
#define tr(it, a) for(auto it = a.begin(); it != a.e
#define PI 3.1415926535897932384626
#define inf 1e18
```

Line 1, Column 1   Tab Size: 4   C++

first2.cpp

```cpp
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    //displayOptions();
    //glColor3f(1,0,0);
    root_centre_x=700;
    root_centre_y=600;
    node_width=25;
    root=NULL;
    insert(5);
    insert(3);

    insert(4);
    insert(2);
    //insert(1);
    insert(7);
    insert(6);
    insert(8);
    glFlush();
}
void init() {
    glClearColor(0.7,0.2,0.3,1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, SCREEN_SIZE_X, 0.0, SCREEN_SIZE_Y);
}
int main(int argc, char **argv) {
    cout<<"2";
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1500, 800);
    glutInitWindowPosition(0,0);
    //tree.accept_tree();
    glutCreateWindow("Binary Search Tree");
    glClearColor(0.6,0.8,0.1,0);

    glutDisplayFunc(display);
    //glutKeyboardFunc(keyboard);
    //glutMouseFunc(mouse);
    init();
    glutMainLoop();
    return 0;
}
```

Line 1, Column 1   Tab Size: 4   C++

```cpp
#define M_PI 3.142

struct node {
    int data;
    struct node* left;
    struct node* right;
}*root=NULL;
typedef struct node* NODE;
int root_centre_x;
int root_centre_y,node_width;

int count_digit(int n) {
    return (n == 0) ? 0 : floor(log10(n) + 1);
}

void to_string(char *s, int num) {
    int r, d;
    d = count_digit(abs(num));
    if(num == 0)
        s[0] = '0';
    if(num < 0) {
        s[0] = '-';
        d++;
    }
    s[d] = '\0';
    num = abs(num);
    while(num != 0) {
        r = num % 10;
        s[--d] = r + '0';
        num /= 10;
    }
}

void display_string(char s[], float x, float y, float z = 0.0) {
    glRasterPos3f(x, y, z);
    for (int i = 0; s[i]; i++)
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, s[i]);
    glFlush();
}

void draw_node(int item,int centre_x,int centre_y,bool write_value=true){
    char data_string[10];
    glBegin(GL_POLYGON);
    glVertex2f(centre_x+node_width,centre_y+node_width);
    glVertex2f(centre_x-node_width,centre_y+node_width);
    glVertex2f(centre_x-node_width,centre_y-node_width);
    glVertex2f(centre_x+node_width,centre_y-node_width);
    glEnd();
    if(write_value){
        to_string(data_string,item);
        glColor3f(1.0,1.0,0.0);
        display_string(data_string,centre_x-node_width+10,centre_y-5);
    }
```

```cpp
#include<bits/stdc++.h>
using namespace std;
#define gc getchar_unlocked
#define Fo(i,n) for(i=0;i<n;i++)
#define Fo(i,k,n) for(i=k;k<n?i<n:i>n;k<n?i+=1:i-=1)
#define ll long long
#define si(x) scanf("%d",&x)
#define sl(x) scanf("%lld",&x)
#define ss(s) scanf("%s",s)
#define pi(x) printf("%d\n",x)
#define pl(x) printf("%lld\n",x)
#define ps(s) printf("%s\n",s)
#define deb(x) cout << #x << "=" << x << endl
#define deb2(x, y) cout << #x << "=" << x << ","
#define pb push_back
#define mp make_pair
#define F first
#define S second
#define all(x) x.begin(), x.end()
#define clr(x) memset(x, 0, sizeof(x))
#define sortall(x) sort(all(x))
#define tr(it, a) for(auto it = a.begin(); it != a.e
#define PI 3.1415926535897932384626
#define inf 1e18
```


Binary Search Tree

REFERENCES:

1. https://www.opengl.org/resources/libraries/glut/spec3/spec3.html

2. https://en.cppreference.com/w/

3. https://www.geeksforgeeks.org/data-structures/

4. http://www.lighthouse3d.com/tutorials/glut-tutorial/

5. https://www.researchgate.net/publication/338427156_Visualization_Research_of_STL_Model_Based_on_OpenGL_Technology

Synopsis Draft verified by

Ms. Ambika Aggarwal   Dr. Deepshikha Bhargava

                                                                              (H.O.D.)

 (PROJECT GUIDE)   (Department of Virtualization)