Project Report-Group 09

Muhammad Faraz Ul Abrar

Mannan Rauf

Muhammad Osama Nusrat

**ABSTRACT**

Our project is of hospital management system.it has following features

* A program that will guide you according to the needs
* Patient condition
* Fee requirements
* Availability of doctor
* No of persons waiting in the list
* Facility of medicine if patients is wiling
* Final bill

Following are the data structures which we have used

Trees

Arrays

To save the information of the patient visited we have used file handling. Firstly it asks the patient if he is in a state of emergency or normal state. Then it asks the Patient cnic, name. Then it asks which the patient that which type of disease he is suffering from so that he send the patient to the specific doctor. We have used sleep function so that if there is already a patient with doctor the patient outside waits for his entry. If the patients wants to take medicine from same hospital he can take it’s up to the patient several medicines are available for each disease. In the end it calculates the total bill. There is also a manager mode it requires a secret 4 digit pin to access. It has record of the patients who have visited that day. Some improvements could be made to make it better by saving the information using if stream. Secondly we could also have used password instead of pin number.

**Introduction:**

Our problem statement is **‘To Make an Automated User friendly Hospital Management System that provides complete guidance to Patient’**.

For this purpose we implemented Data Structures and Algorithms techniques to make our respected user friendly program. We implemented our program with the help of:

1. **Linked Lists**
2. **Binary Trees**
3. **Arrays**

With the help of this program a user will:

1. Register himself as a normal patient or emergency patient
2. Give program necessary credentials like **name, Id number** to get registered
3. Get appointment to the doctor according to the disease he is suffering from
4. Purchase medicines of his choice
5. Saves a list of visited patients if he logged in as a ‘Manager” in the program

For saving the list of patients we used the concept of file handling. This program can be implemented on small scale hospital sector for its creativity. It provides a patient to select a particular disease from which he is suffering as well as a Manager can save a list of patients visited on a particular day.

**Implementation:**

**Time Complexity:**

* Time complexity of Linked Lists: O(n)
* Time complexity of Binary trees: O(nlogn)

Hence the overall time complexity of this program is: **O(n)** where n is the number of patients registered.

**Data structures used:**

* **Main structure** of program was implemented using the concept of **Binary trees**. We designated specific program to run for each node of tree. We implemented this structure using post order traversal of Binary Tree.
* We chose binary trees for this purpose because we wanted to implement the whole program using all the major techniques of Data structures.

***1) Linked List implementation:***

Basically we have implemented the concept of singly linked list in our project. The concept behind the use of linked list is solely the registry of patient. As we are aware of the fact that in a linked list a node data structure is created with a few basic attributes and pointer pointing to other nodes, likewise the implementation of linked list is done in the same suit thereby making a data structure node i.e. of patients having unique attributes including name, serial no, bill along with the pointers. Furthermore we have used the concept of class here.

***2) Binary Tree implementation:***

As mentioned above the binary tree implementation has been done in a quite exciting way. We have used to concept of tree traversal in which we have implemented DFS (Depth First) “Pre-order” algorithm, the underlying objective is the use the tree data structure in such a way that each major function in the code or each step in the hospital management system that patient will come across is carried through the node traversal, for this purpose we have created 5 nodes along with tree data structure and also each function that will be called, has been linked to a specific node.

***3) Array implementation:***

The concept of array has also been used a number of times in this project. For example:

-BILL of the patient

-String array for the name of patients etc.

* **Registration of Patient** was implemented using **arrays**. Programs will ask user to enter his condition (Normal or Emergency). After that he will enter his name, Id and will specify his disease and then he’ll be proceeded to respected doctor for checkup. Then the program will ask the user if he want to purchase medicine or not. This all is done for one patient using arrays and linked lists. Each array index points the particular patient and we initialized **Linked List** for each array index which will store his Name, Id, Disease and bill etc.
* We chose these structures for this task because it is more simple and efficient and we can easily traverse through the whole list using arrays. In this way we can also access their respected credentials.
* **Manager mode** is the mode in which a program asks a user to enter a pin code to log in to manager portal. The user will be given 4 tries before the program will shut down automatically if he failed to enter the correct pin code. After logging into this portal user will be asked whether he wants to save visited patient list or not. The List will be saved to respected location.

**Main Code:**

**Binary trees:**

struct node\* newNode(int data)

{

struct node\* node = (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return(node);

}

void PostorderTraversal(node\* node)

{

if (node == NULL)

{

Z: return;

}

PostorderTraversal(node->left);

PostorderTraversal(node->right);

if ( ent == 3)

{

goto Z;

}

node->hos();

}

**Linked Lists:**

struct nod

{

int srno;

string name;

int cnc;

int bill;

nod \*next;

};

class linkedlist

{

nod \*head;

nod \*current;

public:

linkedlist()

{

head = new nod();

current = head;

}

**Main Function:**

int main()

{

struct node \*root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

A:

PostorderTraversal(root);

getchar();

int exe;

cout << "\n Do you wanna exit \n";

cout << "Press 1 for yes and 2 for no" << endl;

cin >> exe;

if (exe == 1)

{

system("pause");

return 0;

}

else if(exe !=1)

{

i = 0;

k++;

ent = NULL;

cin.clear();

cin.ignore();

goto A;

}

}

**Main menu of Program:**

cout << "Welcome to our Hospital" << endl << endl;

P:

cout << "Select the appropriate option" << endl;

cout << "Select your state :" << endl;

cout << "\n\n1- Normal Patient "<<endl<<"2- Emergency patient"<< endl <<"3- manager mode"<<endl<<"4- Exit the program \n";

cin >> ent;

if (ent == 1)

{

return;

}

else if (ent == 2)

{

cout << "we are equally sorrow in this grieve moment ";

}

else if (ent == 3)

{

manager();

}

else if (ent == 4)

exit(0);

else {

cout << " Invalid Choice" << endl << endl;

goto P;

}

}

**Registration of Patient:**

if (i == 2)

{

B:

cout << "\n Enter patient's CNIC no.(without dashes) ";

cin >> cnic[k];

if (cnic[k] > 9999999999999 || cnic[k] < 100000)

{

cout << "\nInvalid CNIC ! Try again\n";

cin.clear();

cin.ignore();

system("pause");

goto B;

}

cout << "\n Enter Patient's name " << endl;

cin >> name[k];

patientlist.newnod(cnic[k], name[k]);

}

**Manager Mode:**

void manager()

{

int givenkey;

int key = 1111;

int chance = 0, max = 4;

cout << "You are About To enter The Manager Portal (You have Only 4 tries to Enter Correct key to open the portal)" << endl;

system("pause");

H:

cout << "\nEnter the secret key ";

cin >> givenkey;

if (max == 0) {

J:

exit(0);

}

else if (key == givenkey)

{

cout << "Access Granted" << endl << endl;

patientlist.display();

system("pause");

}

else

{

cout << "Invalid Key---You have "<<(max-1)<<" tries remaining" << endl;

chance++;

max--;

if (max == 0)

{

cout << "Max Invalid Tries Reached-----You are exiting the portal" << endl;

Sleep(2500);

goto J;

}

goto H;

}

}

void display()

{

current = head;

current = current->next;

cout << endl << "\n\n-The List of patients that visited today are: \n\n";

cout << "Srno. Patient's Name CNIC Bill\n";

cout << endl << current->srno << " " << current->name << " " << current->cnc << " " << current->bill;

while (current->next != NULL)

{

cout << endl << (current->next)->srno << " " << (current->next)->name << " " << (current->next)->cnc << " " << (current->next)->bill;

current = current->next;

}

//"File Handling"

int re;

cout << "\n\nDo you want to save this Important List Permanently?(press 1 for save and 2 for no)\n";

cin >> re;

if (re == 1)

{

ofstream myfile;

myfile.open("HospitalSafe.txt");

current = head;

current = current->next;

myfile << endl << "\n\n-The List of patients that visited today are: \n\n";

myfile << "Srno. Patient's Name CNIC Bill\n";

myfile << endl << current->srno << " " << current->name << " " << current->cnc << " " << current->bill;

while (current->next != NULL)

{

myfile << endl << (current->next)->srno << " " << (current->next)->name << " " << (current->next)->cnc << " " << (current->next)->bill;

current = current->next;

}

myfile.close();

cout << "Save Successful";

}

else

{

cout << "OK";

}

cout << endl << endl;

}

**Results:**

This project, demonstrating the management in Hospital, is limited to serving a single patient at a time as it was the 1st generation version of our attempt at it. The time complexity of the program is Big-O (N) which is quite good in case of a small hospital.

The system can be used at a hospital which is for a not-so-busy hospital at the moment we have a single doctor at the duty and there is a waiting mechanism for patients.

**Discussion & conclusion:**

As for its further extension, we can implement a system which has the possibility that if a doctor is busy with medication examination of a patient, for a next patient to come, the system could direct him to another specialist’s supervision. So, if any patient comes, the program checks for the first doctor. If he is busy, it asks the different doctor and so on.

Another possibility is that, if a limited number of doctors are available, a **queue** can be implemented which allow the serving of patients on first come, first serve basis i.e. the first patient to come get the first priority to be examined by doctor, the second to come gets the second priority and so on. So even if one doctor is available only, a queue would allow us to have a robust system.

Moreover, we can introduce to our program, a report producer at the end of examinations. The report will show all the tests taken by patient and their results with deviation from a normal patient’s result. The details of the patient, his name, contact, address and medical details would also be shown on report such as his blood group etc.