

REPORT OF DS PROJECT

DATA STRUCTURE



NAZEER AHMED

023-18-0062

BS\_CS\_III(A)

|  |  |
| --- | --- |
| Name | Nazeer Ahmed |
| CMS ID | 023 - 18 – 0062 |
| Project Name | UNIVERSITY management System |
| Idea | This project help us to save the data of student according to the department of students. |
| Details | As discussed in idea, there are main features of this project:   1. STUDENT RECORD 2. NEW STUDENT ADD 3. DELETE STUDENT RECODE 4. ADD DEPARTEMNT 5. DELETE DEPARTMENT 6. DESTROY DEPARTMENT 7. SEARCH STUDENT BY NAME AS WELL AS BY DEPARTMENT 8. DISPLAY ALL DEPARTMENT |

**PROJECT:**

University MANAGEMENT SYSTEM is a program the students Record According to their Department.

**Features:**

* Add student
* Search Student By their Id NO
* Search Student By their class name
* Delete student Recode
* Display ONE department Record
* Display All Department as well as Student of that Department

**OBJECTIVE:**

The main Objective of this project is that help to maintain the recode of Students According to their Department.

**STURCTURE OF DEPARTMENT:**

**STURCTURE OF DEPARTMENT WHEN ADD NEW STUDENT IN BST**

**Description:**

Tree Data Structure is used in this program.

In Tree Name of Department is saved and in every Node root is saved of students

**INSERT:**

In Insert Function You can Add New Department in University. Also You can Insert new students in that Department.

1. ADD STUDENT:

In department First of All You Insert the Name of Department then You ADD New Student LIKE:

* Name
* Id
* Age

**DISPLAY:**

In this Project there are two Display Function:

1. In First Display Function First of All You have to Insert the Department then Id of that Student then this program Show all data that is Related to that Students
2. In second Display Function first of all you have to insert the name of department then it will Display all student of that Department.

**DELETE:**

You can Delete the any student recode According to their Department and also you can destroy any class which you want.

SOURCE CODE

#include<iostream>

#include<iomanip>

using namespace std;

struct student\_record

{

string student\_name;

int age;

int id;

student\_record \*left\_next;

student\_record \*right\_next;

};

class List1

{

public:

student\_record \*root1,\*current1,\*trail\_curr;

List1()

{

root1=NULL;

}

void student\_recode(string name, int id1,int age)

{

student\_record \*NewNode1= new student\_record;

NewNode1->student\_name=name;

NewNode1->id=id1;

NewNode1->age=age;

NewNode1->left\_next=NULL;

NewNode1->right\_next=NULL;

if(root1==NULL)

{

root1=NewNode1;

}

else

{

current1 = root1;

while(current1)

{

trail\_curr=current1;

if(current1->id==id1)

{

cout<<" The insert Item is already in the list-";

cout<<" dupplicate are not allowed "

<<"insert iteml"<<endl;

return ;

}

else if(current1->id > id1)

{

current1=current1->left\_next;

}

else

{

current1=current1->right\_next;

}

}

if(trail\_curr->id > id1)

{

trail\_curr->left\_next=NewNode1;

}

else

{

trail\_curr->right\_next=NewNode1;

}

}

}

bool srchbyid(int s\_id)

{

bool flag=false;

student\_record \* curr;

if(root1==NULL)

{

cout<<"NOt found"<<endl;

}

else

{

curr=root1;

while(curr!=NULL && !flag)

{

if(curr->id==s\_id)

{

flag=true;

if(flag)

cout<<"NAME : "<<curr->student\_name<<" "<<endl;

cout<<"AGE : "<<curr->age<<" "<<endl;

}

else if(curr->id > s\_id)

{

curr=curr->left\_next;

}

else

{

curr=curr->right\_next;

}

}

}

}

void InorderTraversal(student\_record \*&q)

{

if(q)

{

InorderTraversal(q->left\_next);

cout<<"---------------------------"<<endl;

cout<<" > NAME : "<<q->student\_name<<endl;

cout<<" > ID : "<<q->id<<endl;

cout<<" > AGE : "<<q->age<<endl;

cout<<"--------------------------"<<endl;

InorderTraversal(q->right\_next);

}

}

void deleteFromNode(student\_record \*p)

{

student\_record \*current;

student\_record \*trailcurrent;

student\_record \*temp;

if(p==NULL)

{

cout<<"ERROR : THE NODE TO BE DELETED IN NULL. "<<endl;

}

else if(p->left\_next==NULL && p->right\_next==NULL)

{

temp=p;

p=temp->right\_next;

delete temp;

}

else if(p->left\_next==NULL)

{

temp=p;

p=temp->left\_next;

delete temp;

}

else if(p->right\_next==NULL)

{

temp=p;

p=temp->right\_next;

delete temp;

}

else

{

current=p->left\_next;

trailcurrent=NULL;

while(current->right\_next!=NULL)

{

trailcurrent=current;

current=current->right\_next;

}

p->id=current->id;

if(trailcurrent==NULL)

{

p->left\_next=current->left\_next;

}

else

{

trailcurrent->right\_next=current->left\_next;

}

delete current;

}

}

void deleteNode(int id2)

{

student\_record \*curr;

student\_record \*t\_curr;

bool found =false;

if(root1==NULL)

{

cout<<"CANNOT DELETE FORM THE EMPTY LIST "<<endl;

}

else

{

curr=root1;

t\_curr=root1;

while(curr!=NULL && !found)

{

if(curr->id==id2)

{

found = true;

}

else

{

t\_curr=curr;

if(curr->id > id2)

{

curr=curr->left\_next;

}

else

{

curr=curr->right\_next;

}

}

if(curr==NULL)

{

cout<<"The delete item is not in the tree "<<endl;

}

else if(found)

{

if(curr==root1)

{

deleteFromNode(root1);

}

else if(t\_curr->id >id2)

{

deleteFromNode(t\_curr->left\_next);

}

else

{

deleteFromNode(t\_curr->right\_next);

}

}

}

}

}

void destroyTree(student\_record \*&p)

{

if(p)

{

destroyTree(p->left\_next);

destroyTree(p->right\_next);

delete p;

p=NULL;

}

}

};

struct School

{

string Room\_name;

School \*Left\_link;

School \*Right\_link;

List1 obj;

};

class List

{

public:

School \*root;

int count\_Room;

List()

{

root=NULL;

}

void Room\_name(string Room\_Name)

{

School \*current;

School \*trail\_current;

School \*NewNode = new School;

NewNode->Room\_name=Room\_Name;

NewNode->Left\_link=NULL;

NewNode->Right\_link=NULL;

if(root==NULL)

{

root=NewNode;

}

else

{

current=root;

while(current)

{

trail\_current=current;

if(current->Room\_name==Room\_Name)

{

cout<<" The insert Item is already in the list-";

cout<<" dupplicate are not allowed "

<<"insert iteml"<<endl;

return ;

}

else if(current->Room\_name > Room\_Name)

{

current=current->Left\_link;

}

else

{

current=current->Right\_link;

}

}

if(trail\_current->Room\_name>Room\_Name)

{

trail\_current->Left\_link=NewNode;

}

else

{

trail\_current->Right\_link=NewNode;

}

}

}

bool Search(string clas\_name)

{

bool flag=false;

string s\_name;

int s\_age,s\_id;

School \* curr;

if(root==NULL)

{

cout<<"There is no element is tree"<<endl;

}

else

{

curr=root;

while(curr!=NULL & !flag)

{

if(curr->Room\_name==clas\_name)

{

flag=true;

if(flag)

{

cout<<"ENTER THE STUDENT NAME : ";

cin>>s\_name;

cout<<"ENTER THE ID OF THE STUDENT : ";

cin>>s\_id;

cout<<"ENTER THE AGE OF THE STUDENT : ";

cin>>s\_age;

curr->obj.student\_recode(s\_name,s\_id,s\_age);

}

}

else if(curr->Room\_name > clas\_name)

{

curr=curr->Left\_link;

}

else

{

curr=curr->Right\_link;

}

}

}

}

bool srchbyclassname(string clsname)

{

bool flag=false;

School \* curr1;

if(root==NULL)

{

cout<<"There is no element is tree"<<endl;

}

else

{

curr1=root;

while(curr1!=NULL && !flag)

{

if(curr1->Room\_name==clsname)

{

flag=true;

if(flag)

{

cout<<"> "<<clsname<<endl;

curr1->obj.InorderTraversal(curr1->obj.root1);

}

}

else if(curr1->Room\_name > clsname)

{

curr1=curr1->Left\_link;

}

else

{

curr1=curr1->Right\_link;

}

}

}

}

void InorderTraversal(School \*&p)

{

if(p)

{

InorderTraversal(p->Left\_link);

cout<<p->Room\_name<<" ";

cout<<endl;

p->obj.InorderTraversal(p->obj.root1);

cout<<endl;

InorderTraversal(p->Right\_link);

}

}

void srchbyid(int srch\_id,string name1)

{

bool flag=false;

School \* curr1;

if(root==NULL)

{

cout<<"There is no element is tree"<<endl;

}

else

{

curr1=root;

while(curr1!=NULL && !flag)

{

if(curr1->Room\_name==name1)

{

flag=true;

if(flag)

{

curr1->obj.srchbyid(srch\_id);

}

}

else if(curr1->Room\_name > name1)

{

curr1=curr1->Left\_link;

}

else

{

curr1=curr1->Right\_link;

}

}

}

}

void DeleteByid(int id , string name2)

{

bool flag=false;

School \*curr1;

if(root==NULL)

{

cout<<"There is no element is tree"<<endl;

}

else

{

curr1=root;

while(curr1!=NULL && !flag)

{

if(curr1->Room\_name==name2)

{

flag=true;

if(flag)

{

curr1->obj.deleteNode(id);

cout<<"DELETE SUCCSEFULLY"<<endl;

}

}

else if(curr1->Room\_name > name2)

{

curr1=curr1->Left\_link;

}

else

{

curr1=curr1->Right\_link;

}

}

}

}

/\* void DeleteClass(School \*p)

{

School \*current;

School \*trailcurrent;

School \*temp;

if(p==NULL)

{

cout<<"ERROR : THE NODE TO BE DELETED IN NULL. "<<endl;

}

else if(p->Left\_link==NULL && p->Right\_link)

{

temp=p;

p=temp->Right\_link;

delete temp;

}

else if(p->Left\_link==NULL)

{

temp=p;

p=temp->Left\_link;

delete temp;

}

else if(p->Right\_link==NULL)

{

temp=p;

p=temp->Right\_link;

delete temp;

}

else

{

current=p->Left\_link;

trailcurrent=NULL;

while(current->Right\_link!=NULL)

{

trailcurrent=current;

current=current->Right\_link;

}

p->Room\_name=current->Room\_name;

if(trailcurrent==NULL)

{

p->Left\_link=current->Left\_link;

}

else

{

trailcurrent->Right\_link=current->Left\_link;

}

delete current;

}

}

void deleteNode(string class\_name)

{

School \*curr;

School \*t\_curr;

bool found =false;

if(root==NULL)

{

cout<<"CANNOT DELETE FORM THE EMPTY LIST "<<endl;

}

else

{

curr=root;

t\_curr=root;

while(curr!=NULL && !found)

{

if(curr->Room\_name==class\_name)

{

found = true;

if(found)

{

curr->obj.destroyTree(curr->obj.root1);

cout<<"DELETE SUCCUFULLY !"

<<endl;

}

}

else

{

t\_curr=curr;

if(curr->Room\_name > class\_name)

{

curr=curr->Left\_link;

}

else

{

curr=curr->Right\_link;

}

}

if(curr==NULL)

{

cout<<"The delete item is not in the tree "<<endl;

}

else if(found)

{

if(curr==root)

{

DeleteClass(root);

}

else if(t\_curr->Room\_name >class\_name)

{

DeleteClass(t\_curr->Left\_link);

}

else

{

DeleteClass(t\_curr->Right\_link);

}

}

}

}

}\*/

};

int main()

{

List object;

List1 object1;

char ch1 ,ch;

string class\_name,clas\_\_name,n;

int srch\_id;

string classname;

object.Room\_name("CS");

object.Room\_name("MATH");

object.Room\_name("SE");

object.Room\_name("BBA");

object.Room\_name("AF");

object.Room\_name("BE.D");

cout<<setw(55)<<" -----------------------------------------"<<endl;

cout<<setw(50)<<" >UNIVERSITY MANAGMENT SYSTEM" <<endl;

cout<<setw(55)<<" -----------------------------------------"<<endl;

block:

cout<<" > ADD STUDENT (1)"<<endl;

cout<<" > VIEW STUDENT BY ID (2)"<<endl;

cout<<" > VEIW ALL STUDENT BY CLASS NAME (3)"<<endl;

cout<<" > DELETE STUDENT (4)"<<endl;

cout<<" > DISPLAY ALL CLASS DATA (5)"<<endl;

cout<<endl;

cout<<"PRESS : ";

cin>>ch1;

if(ch1=='1')

{

cout<<"> MATH"<<endl;

cout<<"> CS"<<endl;

cout<<"> SE"<<endl;

cout<<"> BBA"<<endl;

cout<<"> AF"<<endl;cout<<"> BE.D"<<endl;

cout<<"SELECT DEPARTMENT "<<endl;

cin>>classname;

object.Search(classname);

cout<<endl;

cout<<" > DATA ADDED SUCCESFULLY ! "<<endl;

cout<<"Press y for go back : ";

cin>>ch;

if(ch=='y' || ch=='Y')

system("cls");

cout<<endl;

goto block;

}

else if (ch1=='2')

{

cout<<"ENTER THE CLASS NAME :";

cin>>clas\_\_name;

cout<<"ENTER THE STUDENT ID : ";

cin>>srch\_id;

object.srchbyid(srch\_id,clas\_\_name);

cout<<"Press y for go back : ";

cin>>ch;

if(ch=='y' || ch=='Y')

system("cls");

cout<<endl;

goto block;

}

else if (ch1=='3')

{

cout<<"ENTER THE CLASS NAME ";

cin>>class\_name;

object.srchbyclassname(class\_name);

cout<<"Press y for go back : ";

cin>>ch;

if(ch=='y' || ch=='Y')

system("cls");

cout<<endl;

goto block;

}

else if(ch1=='4')

{

cout<<"ENTER THE CLASS NAME :";

cin>>clas\_\_name;

cout<<"ENTER THE STUDENT ID : ";

cin>>srch\_id;

object.DeleteByid(srch\_id,clas\_\_name);

cout<<"Press y for go back : ";

cin>>ch;

if(ch=='y' || ch=='Y')

system("cls");

cout<<endl;

goto block;

}

else if(ch1=='5')

{

object.InorderTraversal(object.root);

cout<<"Press y for go back : ";

cin>>ch;

if(ch=='y' || ch=='Y')

system("cls");

cout<<endl;

goto block;

}

else

{

cout<<"You entered wrong number : "<<endl;

cout<<"Press y for go back : ";

cin>>ch;

if(ch=='y' || ch=='Y')

system("cls");

cout<<endl;

goto block;

}

}