5 assig cpp

#include"admin.h"

Admin::Admin():Employee(){

this->allowance=0;

}

Admin::Admin(int i,char\* nm,double s,double a):Employee(i,nm,s){

this->allowance=a;

}

void Admin::setAllowance(double a){

this->allowance=a;

}

double Admin::getAllowance(){

return this->allowance;

}

void Admin::display(){

cout<<"Allowance:\n";

Employee::display();

cout<<"Allowance:"<<this->allowance<<"\n";

}

ostream& operator<<(ostream& o,Admin& a){

o<<"\nAdmin\n";

o<<"id:"<<a.getid()<<"\n";

o<<"name:"<<a.getname()<<"\n";

o<<"salary:"<<a.getsalary()<<"\n";

o<<"allowance:"<<a.getAllowance()<<"\n";

o<<"---------------------------------------\n"

return o;

}

//

#include"emp.h"

#include<iostream>

#ifndef admin

#define admin

using namespace std;

class Admin:public Employee{

double allowance;

public:

Admin();

Admin(int,char\*,double,double);

void setAllowance(double a);

double getAllowance();

void display();

};

ostream& operator<<(ostream& o,Admin& a);

#endif

//

#pragma once

#include<iostream>

using namespace std;

class Employee{

int id;

char name[20];

double salary;

public:

Employee();

Employee(int,char\*,double);

void setid(int);

void setname(char\*);

void setsalary(double);

int getid();

char\* getname();

double getsalary();

virtual void display();

};

ostream& operator<<(ostream&,Employee&);

#include "emp.h"

#include "hrh.h"

hr::hr():Employee() {

this->commission=0;

}

hr::hr(int i,char\* nm,double s,double c):Employee(i,nm,s){

this->commission=c;

}

void hr::setCommission(double c){

this->commission=c;

}

double hr::getCommission(){

return this->commission;

}

void hr::display(){

cout<<"Hr:\n";

Employee::display();

cout<<"Commission:"<<this->commission<<"\n";

}

ostream& operator<<(ostream& o,hr& h){

o<<"\nHR;\n";

o<<"id:"<<h.getid()<<"\n";

o<<"name:"<<h.getname()<<"\n";

o<<"salary:"<<h.getsalary()<<"\n";

o<<"Commision:"<<h.getCommission()<<"\n";

o<<"---------------------------------------\n"

return o;

}

#include"emp.h"

#ifndef hrh

#define hrh

class hr:public Employee{

double commission;

public:

//hr():Employee();-->this is declaration so we cannot call the function here Employee()

hr();

hr(int,char\*,double,double);

void setCommission(double);

double getCommission();

void display();

};

ostream& operator<<(ostream& o,hr& h);

#endif

//#include <iostream>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

#include "myarr.h"

int main() {

MyArry arr(5);

int exit=0;

int choice;

do{

cout<<"\n";

cout<<"0.Exit\n";

cout<<"1.Add\n";

cout<<"2.Search\n";

cout<<"3.delete\n";

cout<<"4.display\n";

cout<<"Enter the choice:";

cin>>choice;

switch(choice){

case 0:{

exit=1;

cout<<"Exit!";

break;

}

case 1:{

//add element

cout<<"choice the emp:\n";

cout<<"1.Admin\n";

cout<<"2.hr\n";

cout<<"3.salesManager\n";

int e;

cout<<"Enter the choice:\n";

cin>>e;

switch(e){

case 1:{

//admin

int i;

double s;

double a;

char nm[20];

cout<<"Enter the id:";

cin>>i;

cout<<"Enter the name:";

cin>>nm;

cout<<"Enter the salary:";

cin>>s;

cout<<"Enter the allowance:";

cin>>a;

//admin al(i,nm,s,a);//block madhe obj hotoy stack vr ani heap vr adrress assign hotoy

//jas next ieration jaty tas stack nighun janr block ch

//arr.addElement(&a1);

Admin\* a1=new Admin(i,nm,s,a);

arr.addElement(a1);

break;

}

case 2:{

//Hr

int i;

double s;

double c;

char nm[20];

cout<<"Enter the id:";

cin>>i;

cout<<"Enter the name:";

cin>>nm;

cout<<"Enter the salary:";

cin>>s;

cout<<"Enter the commission:";

cin>>c;

hr\* h=new hr(i,nm,s,c);

arr.addElement(h);

break;

}

case 3:{

int i,t;

double s;

double inc;

char nm[20];

cout<<"Enter the id:";

cin>>i;

cout<<"Enter the name:";

cin>>nm;

cout<<"Enter the salary:";

cin>>s;

cout<<"Enter the Incentive:";

cin>>inc;

cout<<"Enter the Target:";

cin>>t;

SalesM\* sm=new SalesM(i,nm,s,inc,t);

arr.addElement(sm);

break;

}

}//out of switch

break;

}

case 2:{

cout<<"case2\n";

int id;

cout<<"Enter id to search:";

cin>>id;

int i=arr.searchElement(id);

cout<<"index:"<<i<<"\n";//index

Admin\* a=dynamic\_cast<Admin\*>(arr.getPtr()[i]);

hr\* h=dynamic\_cast<hr\*>(arr.getPtr()[i]);

SalesM\* sm=dynamic\_cast<SalesM\*>(arr.getPtr()[i]);

if(a!=NULL){

a->display();

}

else if(h!=NULL){

h->display();

}

else if(sm!=NULL){

sm->display();

}

break;

}

case 3:{

cout<<"case3";

int id;

cout<<"Enter id to delete:";

cin>>id;

arr.deleteElement(id);

break;

}

case 4:{

cout<<"Display:\n";

arr.displayElements();

break;

}

}

}while(exit!=1);

return 0;

}

#include"myarr.h"

#include"admin.h"

#include "hrh.h"

#include"salesM.h"

MyArry::MyArry(int s)//only parameterised constructor bcz without size arry must not be create

{

this->size=s;

this->index=-1;

this->ptr=new Employee\*[size];

}

bool MyArry::isFull()

{

if(index<=size-1){

return false;

}

else{

//index is greater

return true;//arry is full

}

}

bool MyArry::isEmpty(){

if(index==-1){

return true;

}

else{

return false;

}

}

void MyArry::addElement(Employee\* e){

if(isFull()){

cout<<"\nArray is full\n";

}

else{

//increment index

/\*Admin\* p=dynamic\_cast<Admin\*>(e);

hr\* h=dynamic\_cast<hr\*>(e);

if(p!=NULL){

ptr[++index]=p;

}

else if(h!=NULL){

ptr[++index]=h;

}

\*/

ptr[++index]=e;

cout<<"element successfully added!";

}

}

int MyArry::searchElement(int id){

if(isEmpty()){

//cout<<"Element not found!!";

return -1;

}

else{

for(int i=0;i<=index;i++){

if(id==ptr[i]->getid()){//pointer to one class variable arrow

return i;

}

}

}

//if not found after searching by loop

return -1;

}

void MyArry::deleteElement(int ele){

if(isEmpty()){

cout<<"Array is Empty";

}

else{

//search the index of the element wants to search

int ind=searchElement(ele);

if(ind!=-1){

for(int i=ind;i<index;i++){

ptr[i]=ptr[i+1];

}

//index must be decrese by 1

index--;

}

else{

cout<<"Not Found";

}

cout<<"Element successfully deleted!";

}

}

void MyArry::displayElements(){

if(isEmpty()){

cout<<"Array is empty";

}

else{

cout<<"Array:\t";

for(int i=0;i<=index;i++){

Admin\* a=dynamic\_cast<Admin\*>(ptr[i]);

hr\* h=dynamic\_cast<hr\*>(ptr[i]);

SalesM\* sm=dynamic\_cast<SalesM\*>(ptr[i]);

if(a!=NULL){

// cout<<"I am in admin";

cout<<\*a;

cout<<"\n";

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

}

else if(h!=NULL){

// cout<<"I am in hr";

cout<<\*h;

cout<<"\n";

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

}

else if(sm!=NULL){

cout<<\*sm;

cout<<"n";

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

}

}

}

}

/\*

void MyArry::displayOne(int i){

cout<<"\nEmployee:\n\n";

cout<<"id:"<<ptr[i]->getid()<<"\n";

cout<<"name:"<<ptr[i]->getname()<<"\n";

cout<<"Salary:"<<ptr[i]->setsalary()<<"\n";

}

\*/

Employee\*\* MyArry::getPtr(){

return this->ptr;

}

//

#include <iostream>

using namespace std;

#include"emp.h"

#include"admin.h"

#include"hrh.h"

#include"salesM.h"

class MyArry{

int size;

int index;

Employee\*\* ptr;//storing the addreess of address

public:

MyArry(int);//only parameterised constructor bcz without size arry must not be create

bool isFull();

bool isEmpty();

void addElement(Employee\*);

int searchElement(int);

void deleteElement(int);

void displayElements();

Employee\*\* getPtr();

//void displayOne(int);

};

#pragma once

#include "emp.h"

//#ifndef SalesM

//#define SalesM

class SalesM :public Employee{

double incentive;

int target;

public:

SalesM();

SalesM(int,char\*,double,double,int);

double getIncentive();

void setIncentive(double);

void setTarget(int);

int getTarget();

void display();

};

ostream& operator<<(ostream& o,SalesM& h);

//#endif

#include "salesM.h"

SalesM::SalesM(){

this->incentive=0;

this->target=0;

}

SalesM::SalesM(int i,char\* nm,double s,double inc,int t):Employee(i,nm,s){

this->incentive=inc;

this->target=t;

}

double SalesM::getIncentive(){

return this->incentive;

}

void SalesM::setIncentive(double i){

this->incentive=i;

}

void SalesM::setTarget(int t){

this->target=t;

}

int SalesM::getTarget(){

return this->target;

}

void SalesM::display(){

cout<<"SaleManager:\n";

Employee::display();

cout<<"Incentive:"<<this->incentive<<"\n";

cout<<"Targets;"<<this->target<<"\n";

}

//global

ostream& operator<<(ostream& o,SalesM& h){

o<<"SalesManager:\n";

o<<"Id:"<<h.getid()<<"\n";

o<<"name:"<<h.getname()<<"\n";

o<<"Salary"<<h.getsalary()<<"\n";

o<<"Incentive:"<<h.getIncentive()<<"\n";

o<<"Target"<<h.getTarget()<<"\n";

o<<"---------------------------------------\n"

return o;

}