Task 1:

//header

#pragma once

//#ifndef CLASS1\_H

#include<string>

#include<iostream>

using namespace std;

#define CLASS1\_H

//#endif // !CLASS1.H

class person

{

int x;//int x private member is not formed

public:

person(int y=0):x(y)

{ cout << "person::person(int) called" << endl; }

~person()

{

}

};

class faculty:virtual public person

{

public:

faculty(int x) :person(x)

{

cout << "faculty::faculty(int) called \n";

}

~faculty()

{

}

};

class student :virtual public person

{

public:

student(int x) :person(x)

{

cout << "student::student(int) called \n";

}

~student()

{

}

};

class ta :public faculty,public student

{

public:

ta(int x) :faculty(x),student(x)//the sequence is not maintained as it is inherited

{

cout << "ta::ta(int) called \n";

}

~ta()

{

}

};

///driver

#include<iostream>

#include<string>

#include"interface1.h"

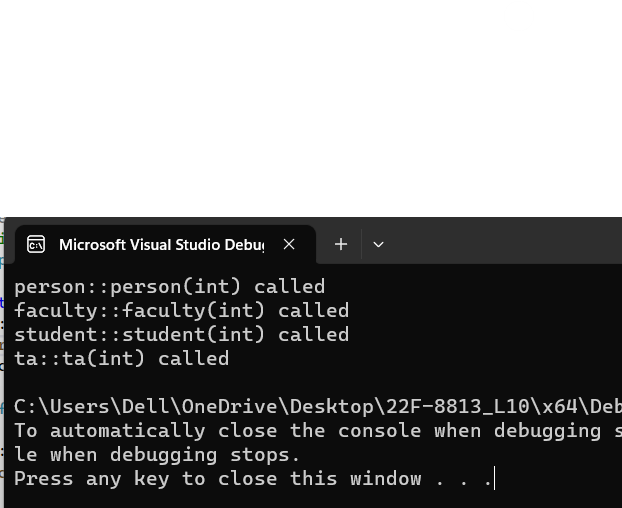
using namespace std;

int main()

{

ta ta1(30);

}



Task 2:

//header

#pragma once

//#ifndef CLASS2\_H

#include<string>

#include<iostream>

using namespace std;

#define CLASS2\_H

//#endif // !CLASS2.H

class animals

{

string name;

public:

void aget()

{

cout << name;

}

void aset(string n)

{

name = n;

}

void sound();

~animals()

{

}

};

class cat :public animals

{

public:

void sound();

};

class dog :public animals

{

public:

void sound();

~dog()

{

}

};

class tiger\_family :public animals

{

public:

void sound();

};

class tiger :public tiger\_family

{

public:

void sound();

~tiger()

{

}

};

class lion :public tiger\_family

{

public:

void sound();

~lion()

{

}

};

class deer :public animals

{

public:

void sound();

~deer()

{

}

};//implementation

#include "interface2.h"

void animals::sound()

{

cout <<"animal sound is called\n";

}

void cat::sound()

{

cout << "cat sound is called\n";

}

void dog::sound()

{

cout << "dog sound is called\n";

}

void deer::sound()

{

cout << "deer sound is called\n";

}

void tiger\_family::sound()

{

cout << "tiger\_family sound is called\n";

}

void tiger::sound()

{

cout << "tiger sound is called\n";

}

void lion::sound()

{

cout << "lion sound is called\n";

}

///driver

#include<iostream>

#include<string>

#include"interface2.h"

using namespace std;

int main()

{

string name;

//animals obj("animal");

animals a;

cout << "enter any naimal name forr instance : ";

cin >> name;

a.aset(name);

a.aget();

a.sound();

deer d;

tiger t;

cat c;

dog dg;

lion l;

c.sound();

dg.sound();

t.tiger\_family::sound();

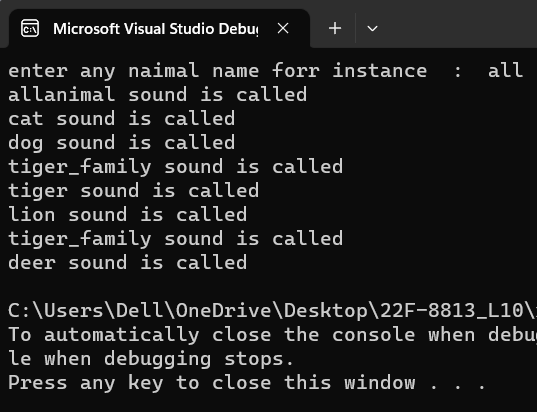
t.sound();

l.sound();

l.tiger\_family::sound();

d.sound();

}



Task 3:

///driver

#include<iostream>

#include<string>

#include"interface3.h"

using namespace std;

int main()

{

int num = 0,val=0,extra=0;

videos v;

photos p;

text t;

socialmedia s;

do

{

cout << "enter values any number but 0 for exit : ";

cin >> num;

cout << "1. photos\n2. videos\n 3.text : ";

cin >> val;

switch (val)

{

case 0:

{

break;

}

case 1:

{

cout << "enter any number for photos : ";

cin >> extra;

p.pset(extra);

p.print();

break;

}

case 2:

{

cout << "enter any number for vidoes : ";

cin >> extra;

v.vset(extra);

v.print();

break;

}

case 3:

{

cout << "enter any number for text : ";

cin >> extra;

t.tset(extra);

t.print();

break;

}

default:

break;

}

} while (num!=0);

}

//implementation

#include "interface3.h"

void socialmedia::print()

{

cout <<num <<" socialmedia\n";

}

void videos::print()

{

cout << v<<" videos \n";

}

void text::print()

{

cout << t<< " text \n";

}

void photos::print()

{

cout << p << " photos\n";

}

//header

#pragma once

//#ifndef CLASS3\_H

#include<string>

#include<iostream>

using namespace std;

#define CLASS3\_H

//#endif // !CLASS3.H

class socialmedia

{

int num;

public:

socialmedia() {};

void aset(int n)

{

num = n;

}

void print();

~socialmedia()

{

}

};

class photos:public socialmedia

{

int p;

public:

photos() {};

void pset(int ph)

{

p = ph;

}

void print();

~photos()

{

}

};

class videos:public socialmedia

{

int v;

public:

videos() {};

void vset(int vi)

{

v= vi;

}

void print();

~videos() {};

};

class text :public socialmedia

{

int t;

public:

text() {};

void tset(int te)

{

t = te;

}

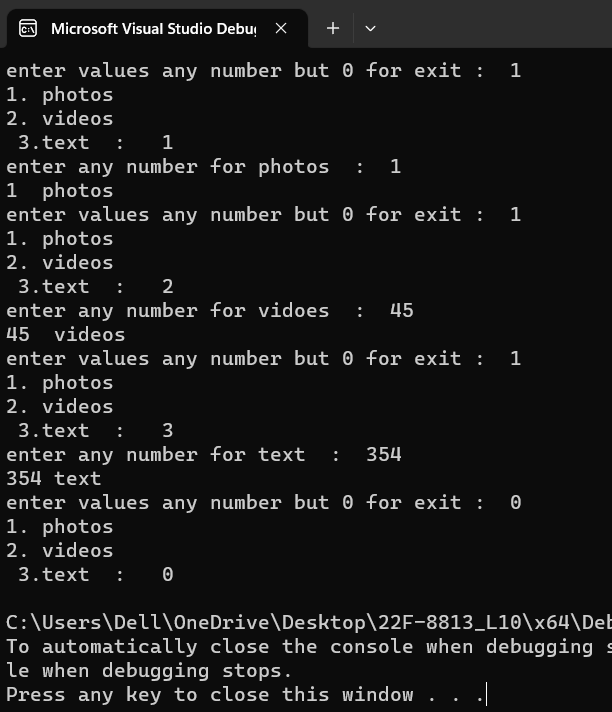
void print();

~text()

{

}

};



Task 4:

//header

#pragma once

//#ifndef CLASS4\_H

#include<string>

#include<iostream>

using namespace std;

#define CLASS4\_H

//#endif // !CLASS4.H

class Time

{

private:

int hr;

int min;

public:

//time();

Time(int h=1, int m=60) :hr(h), min(m) {};

void setTime(int h, int m);

void printTime();

~Time()

{

}

};

class date

{

private:

int month;

int day;

int year;

public:

//date();

date(int m=0, int d=0, int y=0) :month(m), day(d), year(y) {};

void setDate(int m, int d, int y);

void printDate();

~date()

{

}

};

class event

{

private:

string eventName;

Time eventTime;

date eventDay;

public:

event();

event(Time t, date d, string name = "Christmas") : eventTime(t), eventDay(d), eventName(name) { cout << "overload constructor\n"; };

void setEventData(int hours, int minutes, int m, int d, int y, string name,Time t,date o);

void printEventData();

~event()

{

}

};

///driver

#include<iostream>

#include<string>

#include"interface4.h"

using namespace std;

int main()

{

date objd;

objd.setDate(4, 6, 2002);

Time objt;

objt.setTime(1, 20);

event obje(objt, objd, "fast alumini");

obje.setEventData(1, 20, 6, 12, 5, "dcds", objt, objd);

obje.printEventData();

}

//implementation

#include "interface4.h"

void date:: setDate(int m, int d, int y)

{

month = m;

day = d;

year = y;

}

void date::printDate()

{

cout << month << " " << day << " " << year << " \n";

}

void Time::setTime(int h, int m)

{

hr = h;

min = m;

}

void Time::printTime()

{

cout << hr << " " << min<<" \n";

}

void event::setEventData(int hours, int minutes, int m, int d, int y, string name, Time t, date o)

{

hours = 1;

minutes = 20;

m = 6;

d = 4;

y = 1909;

Time obj;

obj.setTime(hours, minutes);

date objd;

objd.setDate(m, d, y);

}

void event:: printEventData()

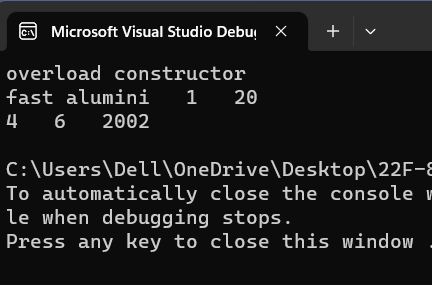
{

cout<< eventName<<" ";

eventTime.printTime();

eventDay.printDate();

}



Tsak5:

///driver

#include<iostream>

#include<string>

#include"interface5.h"

using namespace std;

int main()

{

Date dt(12, 12, 1998);

Student obj1(28, dt);

Bus ob(12);

Engine obj11(28,ob);

perosn cd(12);

bodyparts obj(28,cd);

mobile dc(12);

battery obj1111(28,dc);

fan dtm(12);

wings obj111(28, dtm);

hat r1;

r1.goodperson1(1);

camera r2;

r2.goodBUS(1);

cover r3;

r3.goodmobile1(1);

rings r4;

r4.goodbooks(1);

handle r5;

r5.goodbottle(1);

}

//implementation

#include "interface5.h"

Student::Student(int anAge, Date bD) {

age = anAge;

birthDate = bD;

cout << "Overloaded Constructor";

}

Engine::Engine(int anAge, Bus bD) {

age = anAge;

numBus = bD;

cout << "Overloaded Constructor";

}

bodyparts::bodyparts(int nu, perosn bD) {

n = nu;

numperosn = bD;

cout << "Overloaded Constructor";

}

battery::battery(int nu, mobile bD) {

n = nu;

nummobile = bD;

cout << "Overloaded Constructor";

}

wings::wings(int nu, fan bD) {

n = nu;

numfan = bD;

cout << "Overloaded Constructor";

}

hat::hat() {

person1s = nullptr;

}

person1\* hat::Getperson1(int a) {

if (a > 0 && a < 50)

return person1s;

return nullptr;

}

bool hat::goodperson1(int a) {

if (a > 0 && a < 50)

return person1s->goodperson1();

return false;

}

camera::camera() {

BUSs = nullptr;

}

BUS\* camera::GetBUS(int a) {

if (a > 0 && a < 50)

return BUSs;

return nullptr;

}

bool camera::goodBUS(int a) {

if (a > 0 && a < 50)

return BUSs->goodBUS();

return false;

}

cover::cover() {

mobile1s = nullptr;

}

mobile1\* cover::Getmobile1(int a) {

if (a > 0 && a < 50)

return mobile1s;

return nullptr;

}

bool cover::goodmobile1(int a) {

if (a > 0 && a < 50)

return mobile1s->goodmobile1();

return false;

}

rings::rings() {

bookss = nullptr;

}

books\* rings::Getbooks(int a) {

if (a > 0 && a < 50)

return bookss;

return nullptr;

}

bool rings::goodbooks(int a) {

if (a > 0 && a < 50)

return bookss->goodbooks();

return false;

}

handle::handle() {

bottles = nullptr;

}

bottle\* handle::Getbottle(int a) {

if (a > 0 && a < 50)

return bottles;

return nullptr;

}

bool handle::goodbottle(int a) {

if (a > 0 && a < 50)

return bottles->goodbottle();

return false;

}

//header

#pragma once

//#ifndef CLASS5\_H

#include<string>

#include<iostream>

using namespace std;

#define CLASS5\_H

//#endif // !CLASS5.H

//////////////////////////////////////////////////////////composition examples//////////////////////////////////////////////////////////////////////////

class Date {

int day, month, year;

public:

//Date() {}

Date(int d = 1, int m = 1, int y = 1901) :day(d), month(m), year(y) {}

~Date() {}

};

class Student {

protected:

int age;

Date birthDate;

public:

Student(int anAge, Date bD);

int getAge()

{

return age;

}

void setAge(int anAge)

{

age = anAge;

}

};

//////////////////////////////////////////////////////////composition examples//////////////////////////////////////////////////////////////////////////

class Bus {

int num;

public:

Bus() {}

Bus(int n) :num(n) {}

~Bus() {}

};

class Engine {

protected:

int age;

Bus numBus;

public:

Engine(int anAge, Bus bD);

int getAge()

{

return age;

}

void setAge(int anAge)

{

age = anAge;

}

};

//////////////////////////////////////////////////////////composition examples//////////////////////////////////////////////////////////////////////////

class perosn {

int num;

public:

perosn() {}

perosn(int n) :num(n) {}

~perosn() {}

};

class bodyparts {

protected:

int n;

perosn numperosn;

public:

bodyparts(int nu, perosn bD);

int getAge()

{

return n;

}

void setAge(int nu)

{

n = nu;

}

};

//////////////////////////////////////////////////////////composition examples//////////////////////////////////////////////////////////////////////////

class mobile {

int num;

public:

mobile() {}

mobile(int n) :num(n) {}

~mobile() {}

};

class battery {

protected:

int n;

mobile nummobile;

public:

battery(int nu, mobile bD);

int getAge()

{

return n;

}

void setAge(int nu)

{

n = nu;

}

};

//////////////////////////////////////////////////////////composition examples//////////////////////////////////////////////////////////////////////////

class fan {

int num;

public:

fan() {}

fan(int n) :num(n) {}

~fan() {}

};

class wings {

protected:

int n;

fan numfan;

public:

wings(int nu, fan bD);

int getAge()

{

return n;

}

void setAge(int nu)

{

n = nu;

}

};

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////aggregation/////////////////////////////////////////////////////////////////

class person1 {

int numLegs;

public:

person1() {}

bool goodperson1() { return true; }

};

class hat {

private:

float area;

person1\* person1s;

public:

hat();

person1\* Getperson1(int);

bool goodperson1(int);

};

////////////////////////////////////aggregation/////////////////////////////////////////////////////////////////

class BUS {

int whel;

public:

BUS() {}

bool goodBUS() { return true; }

};

class camera {

private:

float area;

BUS\* BUSs;

public:

camera();

BUS\* GetBUS(int);

bool goodBUS(int);

};

////////////////////////////////////aggregation/////////////////////////////////////////////////////////////////

class mobile1 {

int cov;

public:

mobile1() {}

bool goodmobile1() { return true; }

};

class cover {

private:

float area;

mobile1\* mobile1s;

public:

cover();

mobile1\* Getmobile1(int);

bool goodmobile1(int);

};

////////////////////////////////////aggregation/////////////////////////////////////////////////////////////////

class books {

int cov;

public:

books() {}

bool goodbooks() { return true; }

};

class rings {

private:

float area;

books\* bookss;

public:

rings();

books\* Getbooks(int);

bool goodbooks(int);

};

////////////////////////////////////aggregation/////////////////////////////////////////////////////////////////

class bottle {

int cov;

public:

bottle() {}

bool goodbottle() { return true; }

};

class handle {

private:

float area;

bottle\* bottles;

public:

handle();

bottle\* Getbottle(int);

bool goodbottle(int);

};