//hybrid inheritance

#include<iostream>

//#include<conio.h>

class A //Base class

{

public:

int l;

void len()

{

std::cout<<"\n\nLenght :\t";

std::cin>>l; //Lenght is enter by user

}

};

class B :public A //Inherits property of class A

{

public:

int b,c;

void l\_into\_b()

{

len();

std::cout<<"\n\nBreadth :::\t";

std::cin>>b; //Breadth is enter by user

c=b\*l; //c stores value of lenght \* Breadth i.e. (l\*b) .

}

};

class C

{

public:

int h;

void height()

{

std::cout<<"\n\nHeight :::\t";

std::cin>>h; //Height is enter by user

}

};

//Hybrid Inheritance Level

class D:public B,public C

{

public:

int res;

void result()

{

l\_into\_b();

height();

res=h\*c; //res stores value of c\*h where c=l\*b and h is height which is enter by user

std::cout<<"\n\nResult (l\*b\*h) :::\t"<<res;

}

};

int main()

{

D d1;

d1.result();

return 0;

}

//hierachical inheritance

#include <iostream>

class Side //base class

{

protected:

int length;

public:

void set\_values (int x)

{

length=x;

}

};

class Square: public Side //class derivation

{

public:

int comp\_square()

{

return (length \*length);

}

};

class Cube:public Side

{

public:

int comp\_cube()

{

return (length \*length\*length);

}

};

int main()

{

Square sq;

sq.set\_values(10);

std::cout<<"\nThe square value is:: \n"<< sq.comp\_square()<<"\n";

Cube cb;

cb.set\_values(20);

std::cout<<"\nThe cube value is:: \n"<<cb.comp\_cube()<<"\n";

return 0;

}

//single inheritance

#include<iostream>

//#include<conio.h>

// Base class

class Shape

{

public:

void setWidth(int w)

{

width = w;

}

void setHeight(int h)

{

height = h;

}

protected:

int width;

int height;

};

// Derived class

class Rectangle : public Shape

{

public:

int getArea()

{

return (width \* height);

}

};

int main()

{

Rectangle Rect;

Rect.setWidth(5);

Rect.setHeight(7);

// Print the area of the object.

std::cout << "Total area: " << Rect.getArea()<<"\n";

return 0;

}

//multilevel inheritance

#include<iostream>

class A

{

public:

void display()

{

std::cout<<"Base class content.";

}

};

class B : public A

{

public:

};

class C : public B

{

public:

void show()

{

std::cout<<"\nmember of class C\n";

}

};

int main()

{

C c;

c.display();

c.show();

return 0;

}

//multiple inheritance

#include<iostream>

//#include<conio.h>

class student // base class

{

protected:

int rno,m1,m2;

public:

void get()

{

std::cout<<"Enter the Roll no :";

std::cin>>rno;

std::cout<<"Enter the two marks :";

std::cin>>m1>>m2;

}

};

class sports //base class

{

protected:

int sm; // sm = Sports mark

public:

void getsm()

{

std::cout<<"\nEnter the sports mark :";

std::cin>>sm;

}

};

class statement : public student,public sports

{

int tot,avg;

public:

void display()

{

tot=(m1+m2+sm);

avg=tot/3;

std::cout<<"\n\n\tRoll No : "<<rno<<"\n\tTotal : "<<tot;

std::cout<<"\n\tAverage : "<<avg;

}

};

int main()

{

statement obj;

obj.get();

obj.getsm();

obj.display();

return 0;

}

**ARRAY PROGRAM**

//matrix display

#include<iostream>

class matrixdemo

{

private:

int matrix[3][3],i,j;

public:

void getdata();

void display();

void transpose();

};

void matrixdemo :: getdata()

{

std::cout<<"\nenter elements of the matrix\n";

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

std::cin>>matrix[i][j];

}

}

}

void matrixdemo :: display()

{

std::cout<<"\ngiven matrix is\n";

for(i=0;i<3;i++)

{

std::cout<"\n\n";

for(j=0;j<3;j++)

{

std::cout<<matrix[i][j]<<"\t";

}

std::cout<<"\n\n";

}

}

void matrixdemo :: transpose()

{

std::cout<<"\nTRANSPOSE IS\n";

std::cout<<"\ngiven matrix is\n";

for(i=0;i<3;i++)

{

std::cout<"\n\n";

for(j=0;j<3;j++)

{

std::cout<<matrix[j][i]<<"\t";

}

std::cout<<"\n\n";

}

}

int main()

{

matrixdemo mat;

mat.getdata();

mat.display();

mat.transpose();

return 0;

}

//searchin program

#include<iostream>

class searching

{

private:

int age[7],i,key;

public:

void getdata();

void search();

};

void searching :: getdata()

{

std::cout<<"\ninitialize aRRAY\n";

for(i=0;i<7;i++)

std::cin>>age[i];

std::cout<<"\nenter the element to be searched\n";

std::cin>>key;

}

void searching :: search()

{

int flag=0;

for(i=0;i<7;i++)

{

if(age[i] == key)

{

std::cout<<"\neleemnt foud\n";

std::cout<<"\nelement is "<<age[i]<<"\n";

std::cout<<"\nlocation is "<<i<<"\n";

flag=1;

break;

}

}

if(flag==0)

std::cout<<"\nelement not found\n";

}

int main()

{

searching ser;

ser.getdata();

ser.search();

return 0;

}

//array initializATION

#include<iostream>

class arraydemo

{

private:

int age[5],i,j;

public:

void getdata();

void display();

void arrange();

};

void arraydemo :: getdata()

{

std::cout<<"\ninitialize the array\n";

for(i=0;i<5;i++)

{

std::cin>>age[i];

}

}

void arraydemo :: display()

{

std::cout<<"\narray elements \n";

for(i=0;i<5;i++)

{

std::cout<<age[i]<<"\t";

}

}

void arraydemo :: arrange()

{

for(i=0;i<5;i++)

{

for(j=i+1;j<5;j++)

{

if(age[i]<age[j])

{

int temp = age[i];

age[i] = age[j];

age[j] =temp;

}

}

}

}

int main()

{

arraydemo ar;

ar.getdata();

ar.display();

ar.arrange();

ar.display();

return 0;

}