**OOPS MID-1 LAB PROGRAMS**

**1.Largest of Four Numbers using Nested If**

#include <iostream>

using namespace std;

int max\_of\_four(int a, int b, int c, int d)

{

 int max;

 if((a>b) && (a>c) && (a>d))

 max = a;

 if((b>a) && (b>c) && (b>d))

 max = b;

 if((c>a) && (c>b) && (c>d))

 max = c;

 if((d>a) && (d>b) && (d>c))

 max = d;

 return max;

}

int main()

{

 int a, b, c, d;

 cout<<"enter a,b,c,d:"<<endl;

 cin>>a>>b>>c>>d;

 int ans = max\_of\_four(a, b, c, d);

 cout<<"max: "<<ans;

 return 0;

}

**2.Largest of Four Numbers using If Ladder**

#include <iostream>

using namespace std;

int max\_of\_four(int a, int b, int c, int d)

{

 int max;

 if((a>b) && (a>c) && (a>d))

 max = a;

 else if((b>a) && (b>c) && (b>d))

 max = b;

 else if((c>a) && (c>b) && (c>d))

 max = c;

 else

max = d;

 return max;

}

int main()

{

 int a, b, c, d;

 cout<<"enter a,b,c,d:"<<endl;

 cin>>a>>b>>c>>d;

 int ans = max\_of\_four(a, b, c, d);

 cout<<"max: "<<ans;

 return 0;

}

**3.Display VIBGYOR using Switch Case**

#include <iostream>

using namespace std;

int main()

{

char choice;

cout<<"Enter a letter from VIBGYOR :";

cin>>choice;

switch(choice)

{

case 'V':

cout<<"Violet";

break;

case 'I':

cout<<"Indigo";

break;

case 'B':

cout<<"Blue";

break;

case 'G':

cout<<"Green";

break;

case 'Y':

cout<<"Yellow";

break;

case 'O':

cout<<"Orange";

break;

case 'R':

cout<<"Red";

break;

default:

cout<<"Not a color letter from VIBGYOR";

}

}

**4.Generate Pascals Triangle using do while**

#include <iostream>

using namespace std;

int main()

{

int no\_row,num=1,blk,i,j;

cout<<"Input number of rows: ";

cin>>no\_row;

i=0;

do

{

for(blk=1;blk<=no\_row-i;blk++)

cout<<" ";

j=0;

do

{

if (j==0||i==0)

num=1;

else

num=num\*(i-j+1)/j;

cout<<num<<" ";

j++;

}

while(j<=i);

cout<<endl;

i++;

}

while(i<no\_row);

}

**5.Check for Armstrong Number:**

#include <iostream>

using namespace std;

int main()

{

int num,r,sum=0,temp;

cout<<"Input a number: ";

cin>>num;

for(temp=num;num!=0;num=num/10)

{

r=num % 10;

sum=sum+(r\*r\*r);

}

if(sum==temp)

cout<<temp<<" is an Armstrong number.\n";

else

cout<<temp<<"%d is not an Armstrong number.\n";

}

**6.Generate prime numbers using while**

#include<iostream>

using namespace std;

int main()

{

int n, i, j, count;

cout<<"Enter any number\n";

cin>>n;

cout<<"Prime numbers between 1 to "<<n<<"\n";

for(i = 1; i <= n; i++)

{

count = 0;

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

if(i % j == 0)

{count++;}

if(count == 2)

cout<<i<<"\t";

}

}

**7.Factorial using for and while loop**

#include <iostream>

using namespace std;

int main()

{

long int fact=1;

int num,i,choice;

cout<<"1.Factorial using for loop\n";

cout<<"2.Factorial using while loop\n";

cout<<"Enter a number: ";

cin>>num;

cout<<"Enter your choice: ";

cin>>choice;

switch(choice)

{

case 1:

for(i=1;i<=num;++i)

fact\*=i;

cout<<fact;

break;

case 2:

i=1;

while(i<=num)

{

fact\*=i;

i++;

}

cout<<fact;

break;

default:

cout<<"Invalid Choice!\n";

}

}

**8.Fibonacci series using do while**

#include<iostream>

using namespace std;

int main()

{

int n,i,n1=0,n2=1,temp=0;

cout<<"Enter number of terms:";

cin>>n;

cout<<n1<<" "<<n2<<" ";

n=n-2;

do

{

temp=n1+n2;

n1=n2;

n2=temp;

cout<<temp<<" ";

n--;

}

while(n>0);

return 0;

}

**9.Matrix Addition**

#include<iostream>

using namespace std;

int main()

{

int a[10][10],b[10][10],c[10][10],i,j,m,n;

cout<<"Enter number of rows and columns: ";

cin>>m>>n;

cout<<"Enter elements of first matrix: "<<endl;

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

{

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

{

cin>>a[i][j];

}

}

cout<<"Enter elements of second matrix: "<<endl;

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

{

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

{

cin>>b[i][j];

}

}

cout<<"Addition Matrix:"<<endl;

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

{

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

{

c[i][j]=a[i][j]+b[i][j];

}

}

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

{

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

{

cout<<c[i][j]<<" ";

}

cout<<endl;

}

}

**10.Matrix Multiplication**

#include<iostream>

using namespace std;

int main()

{

int a[10][10],b[10][10],c[10][10],i,j,k,m,n,p,q;

cout<<"Enter number of rows and columns of first matrix:";

cin>>m>>n;

cout<<"Enter number of rows and columns of second matrix:";

cin>>p>>q;

if(n!=p)

{

cout<<"Matrices cannot be multiplied!"<<endl;

}

cout<<"Enter elements of first matrix: ";

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

{

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

{

cin>>a[i][j];

}

}

cout<<"Enter elements of second matrix: ";

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

{

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

{

cin>>b[i][j];

}

}

cout<<"Multiplication Matrix:"<<endl;

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

{

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

{

c[i][j]=0;

}

}

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

{

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

{

for(k=0;k<p;k++)

{

c[i][j]+=a[i][k]\*b[k][j];

}

}

}

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

{

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

{

cout<<c[i][j]<<" ";

}

cout<<endl;

}

}

**11.Inverse of a Matrix**

#include<iostream>

using namespace std;

int main()

{

int i,j,a[4][4],order;

float det=0;

cout<<"Enter the order of matrix:";

cin>>order;

cout<<"Enter elements of matrix:\n";

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

{

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

{ cin>>a[i][j]; }

}

cout<<"MATRIX:\n";

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

{

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

{ cout<<a[i][j]<<" "; }

cout<<endl;

}

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

{ det=det+(a[0][i]\*(a[1][(i+1)%3]\*a[2][(i+2)%3]-a[1][(i+2)%3]\*a[2][(i+1)%3])); }

cout<<"Determinant of Matrix= "<<det<<endl;

cout<<"INVERSE MATRIX:\n";

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

{

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

{ cout<<((a[(j+1)%3][(i+1)%3]\*a[(j+2)%3][(i+2)%3])-(a[(j+1)%3][(i+2)%3]\*a[(j+2)%3][(i+1)%3]))/det<<" "; }

cout<<endl;

}

}

**12.Factorial using Recursion:**

#include<stdio.h>

long int fact(long int a);

void main()

{

long int n,result;

printf("Enter 'n' value:");

scanf("%ld",&n);

result=fact(n);

printf("%ld",result);

}

long int fact(long int n)

{

int f;

f=1;

if(n==1)

return f;

else

return n\*fact(n-1);

}

**13.Swap numbers using Call by Value**

#include<iostream>

using namespace std;

void swap(int x,int y);

int main()

{

int a,b;

cout<<"Enter a,b values: ";

cin>>a>>b;

cout<<"Swapping using Call by Value:\n";

swap(a,b);

cout<<a<<"\t"<<b<<endl;

}

void swap(int x,int y)

{

int t;

t=x;

x=y;

y=x;

}

**14.Swap numbers using Call by Address**

#include<iostream>

using namespace std;

void swap(int \*x,int \*y);

int main()

{

int a,b;

cout<<"Enter a,b values: ";

cin>>a>>b;

cout<<"Swapping using Call by Address:\n";

swap(&a,&b);

cout<<a<<"\t"<<b<<endl;

}

void swap(int \*x,int \*y)

{

int t;

t=\*x;

\*x=\*y;

\*y=t;

}

**15.Program using class by defining member function inside the class**

#include<iostream>

using namespace std;

class class1

{

int p,n,r;

float si;

public:

void input()

{

cout<<"Enter p,n,r values: ";

cin>>p>>n>>r;

}

void process()

{

si=(p\*n\*r)/100;

}

void output()

{

cout<<"Simple interest: "<<si;

}

};

int main()

{

class class1 c1;

c1.input();

c1.process();

c1.output();

}

**16.Program using class by defining member function outside the class**

#include<iostream>

using namespace std;

class class1

{

int p,n,r;

float si;

public:

void input();

void process();

void output();

};

void class1::input()

{

cout<<"Enter p,n,r values: ";

cin>>p>>n>>r;

}

void class1::process()

{

si=(p\*n\*r)/100;

}

void class1::output()

{

cout<<"Simple interest: "<<si;

}

int main()

{

class class1 c1;

c1.input();

c1.process();

c1.output();

}

**17.Generate Fibonacci Series using Constructors**

#include<iostream>

using namespace std;

class fibonacci

{

private:

int f0,f1,fib;

public:

fibonacci()

{

f0=0;

f1=1;

fib=f0+f1;

}

void increment()

{

f0=f1;

f1=fib;

fib=f0+f1;

}

void output()

{

cout<<fib<<"\t";

}

~fibonacci()

{

cout<<"\nDestructed\n";

}

};

int main()

{

fibonacci f;

cout<<"0\t1\t";

for(int i=0;i<8;i++)

{

f.output();

f.increment();

}

}

**18.Default Constructor**

#include<iostream>

using namespace std;

class student

{

private:

char name[25];

int rno;

public:

student()

{

name[0]='\0';

rno=0;

}

void output()

{

cout<<name<<endl;

cout<<rno<<endl;

}

};

int main()

{

student s;

s.output();

}

**19.Fibonacci using Copy Constructor**

#include<iostream>

using namespace std;

class fibonacci

{

private:

int f0,f1,fib;

public:

fibonacci()

{

f0=0;

f1=1;

fib=f0+f1;

}

fibonacci(fibonacci &ptr)

{

f0=ptr.f0;

f1=ptr.f1;

fib=ptr.fib;

}

void increment()

{

f0=f1;

f1=fib;

fib=f0+f1;

}

void output()

{

cout<<fib<<"\t";

}

~fibonacci()

{

cout<<"\nDestructed\n";

}

};

int main()

{

fibonacci f;

cout<<"0\t1\t";

for(int i=0;i<8;i++)

{

f.output();

f.increment();

}

}

**20.Parameterised Constructor**

#include<iostream>

using namespace std;

class class1

{

int n,m;

public:

class1(int x,int y)

{

n=x;

m=y;

}

void output()

{

cout<<n<<endl<<m<<endl;

}

};

int main()

{

class1 c1(10,20);

c1.output();

}

**21.Inline Function and THIS Pointer**

INLINE:

#include<iostream>

using namespace std;

class class1

{

private:

int x;

public:

inline void getdata()

{

cout<<"Enter x: ";

cin>>x;

}

inline void output()

{

cout<<x<<endl;

}

};

int main()

{

class1 c1;

c1.getdata();

c1.output();

}

THIS:

#include<iostream>

using namespace std;

class sample

{

int x;

public:

void output();

};

void sample::output()

{

this->x=10;

cout<<this->x;

}

int main()

{

sample s1;

s1.output();

}

**22.Friend Class and Friend Function**

FRIEND CLASS:

#include<iostream>

using namespace std;

class first

{

friend class second;

private:

int x;

public:

void getdata()

{

cout<<"Enter x: ";

cin>>x;

}

};

class second

{

public:

void output(class first f)

{

cout<<f.x<<endl;

}

};

int main()

{

first f1;

second s1;

f1.getdata();

s1.output(f1);

}

FRIEND FUNCTION:

#include<iostream>

using namespace std;

class class1

{

int x;

friend void output(class class1 c1)

{

cout<<c1.x<<endl;

}

public:

void getdata()

{

cout<<"Enter x: ";

cin>>x;

}

};

int main()

{

class1 c11;

c11.getdata();

output(c11);

}

**23.Static Data Members**

#include<iostream>

using namespace std;

class sample

{

private:

static int c;

public:

void output();

};

int sample::c=100;

void sample::output()

{

for(int i=0;i<=10;i++)

{

c=c+i;

}

cout<<"Sum="<<c<<endl;

}

int main()

{

sample obj;

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

{

cout<<"Count="<<i+1<<endl;

obj.output();

cout<<endl;

}

}

**24.Static Member Function**

#include<iostream>

using namespace std;

class class1

{

static int c;

public:

class1();

static void output();

};

int class1::c=0;

class1::class1()

{

c++;

}

void class1::output()

{

cout<<"Counter Values: "<<c<<endl;

}

int main()

{

cout<<"Before intialization:\n";

class1::output();

class1 c1,c2,c3,c4;

cout<<"After intialization: \n";

class1::output();

}

**25.Single Inheritance Student Information System**

#include<iostream>

using namespace std;

class class1

{

private:

char name[26],dept[5];

int rno,year,sem;

public:

void input();

void output();

};

class class2: public class1

{

private:

int mid1[6],mid2[6],ext[6],Result,i;

int m1t=0,m2t=0,et=0,midavg;

public:

void input();

void output();

};

void class1::input()

{

cout<<"Enter Student name: ";

cin>>name;

cout<<"Enter Roll No.: ";

cin>>rno;

cout<<"Enter Year: ";

cin>>year;

cout<<"Enter Sem: ";

cin>>sem;

cout<<"Enter Department: ";

cin>>dept;

}

void class2::input()

{

class1::input();

for(i=0;i<6;i++)

{

cin>>mid1[i];

}

for(i=0;i<6;i++)

{

cin>>mid2[i];

}

for(i=0;i<6;i++)

{

cin>>ext[i];

}

}

void class1::output()

{

cout<<"Student Name: "<<name<<"\t\tStudent Roll No.: "<<rno<<endl;

cout<<"Year: "<<year<<"\t\tSem: "<<sem<<"\t\tDepartment: "<<dept<<endl;

}

void class2::output()

{

class1::output();

for(i=0;i<6;i++)

{

m1t+=mid1[i];

m2t+=mid2[i];

et+=ext[i];

}

midavg=(m1t+m2t)/2;

Result=(midavg+et)/6;

cout<<"Result: "<<Result<<"%"<<endl;

}

int main()

{

class2 c2;

c2.input();

c2.output();

}

**26.Program using Public, Private and Protected Data members**

#include <iostream>

using namespace std;

class base

{

private:

int x;

protected:

int y;

public:

int z;

base()

{

x = 1;

y = 2;

z = 3;

}

};

class derive: public base

{

public:

void showdata()

{

cout << "x is not accessible" << endl;

cout << "value of y is " << y << endl;

cout << "value of z is " << z << endl;

}

};

int main()

{

derive a;

a.showdata();

return 0;

}

**27.Multiple Inheritance Hospital Management System**

#include<iostream>

using namespace std;

class class1

{

private:

char hos\_name[25],loc[25];

public:

void input();

void output();

};

class class2

{

private:

char doc\_name[26],doc\_dept[26];

int doc\_id;

public:

void input();

void output();

};

class class3:public class1,public class2

{

private:

char pat\_name[26],pat\_type[15],disease[20];

int pat\_id;

public:

void input();

void output();

};

void class1::input()

{

cout<<"enter hospital name and location";

cin>>hos\_name>>loc;

}

void class2::input()

{

cout<<"Enter Doctor name: ";

cin>>doc\_name;

cout<<"Enter Doctor ID: ";

cin>>doc\_id;

cout<<"Enter Doctor Department: ";

cin>>doc\_dept;

}

void class3::input()

{

class1::input();

class2::input();

cout<<"Enter Patient name: ";

cin>>pat\_name;

cout<<"Enter Patient ID: ";

cin>>pat\_id;

cout<<"Enter Patient's disease: ";

cin>>disease;

}

void class1::output()

{

cout<<"hospital name"<<hos\_name;

cout<<"location"<<loc;

}

void class2::output()

{

cout<<"Doctor Name: "<<doc\_name<<"\tDoctor ID: "<<doc\_id<<"\t";

cout<<"Doctor Department: "<<doc\_dept<<endl;

}

void class3::output()

{

class1::output();

class2::output();

cout<<"Patient Name: "<<pat\_name<<"\tPatient ID: "<<pat\_id<<endl;

cout<<"Disease: "<<disease<<endl;

}

int main()

{

class3 c3;

c3.input();

c3.output();

}

**28.Hierarchieal Inheritance Hospital Management System**

#include<iostream>

using namespace std;

class class1

{

private:

char hos\_name[25],loc[25];

public:

void input();

void output();

};

class class2: public class1

{

private:

char doc\_name[26],doc\_dept[26];

int doc\_id;

public:

void input();

void output();

};

class class3:public class1

{

private:

char pat\_name[26],pat\_type[15],disease[20];

int pat\_id;

public:

void input();

void output();

};

void class1::input()

{

cout<<"enter hospital name and location";

cin>>hos\_name>>loc;

}

void class2::input()

{

class1::input();

cout<<"Enter Doctor name: ";

cin>>doc\_name;

cout<<"Enter Doctor ID: ";

cin>>doc\_id;

cout<<"Enter Doctor Department: ";

cin>>doc\_dept;

}

void class3::input()

{

class1::input();

cout<<"Enter Patient name: ";

cin>>pat\_name;

cout<<"Enter Patient ID: ";

cin>>pat\_id;

cout<<"Enter Patient's disease: ";

cin>>disease;

}

void class1::output()

{

cout<<"hospital name"<<hos\_name;

cout<<"location"<<loc;

}

void class2::output()

{

class1::output();

cout<<"Doctor Name: "<<doc\_name<<"\tDoctor ID: "<<doc\_id<<"\t";

cout<<"Doctor Department: "<<doc\_dept<<endl;

}

void class3::output()

{

class1::output();

cout<<"Patient Name: "<<pat\_name<<"\tPatient ID: "<<pat\_id<<endl;

cout<<"Disease: "<<disease<<endl;

}

int main()

{

class2 c2;

c2.input();

c2.output();

class3 c3;

c3.input();

c3.output();

}

**29.Swap numbers using Function Overloading**

#include<iostream>

using namespace std;

int swap(int &ix,int &iy);

float swap(float &fx,float &fy);

char swap(char &cx,char &cy);

int main()

{

int ix,iy;

float fx,fy;

char cx,cy;

cout<<"Enter ix,iy: ";

cin>>ix>>iy;

swap(ix,iy);

cout<<"Enter fx,fy: ";

cin>>fx>>fy;

swap(fx,fy);

cout<<"Enter cx,cy: ";

cin>>cx>>cy;

swap(cx,cy);

cout<<ix<<" "<<iy<<endl;

cout<<fx<<" "<<fy<<endl;

cout<<cx<<" "<<cy<<endl;

}

int swap(int &ix,int &iy)

{

int t;

t=ix;

ix=iy;

iy=t;

}

float swap(float &ix,float &iy)

{

float t;

t=ix;

ix=iy;

iy=t;

}

char swap(char &ix,char &iy)

{

char t;

t=ix;

ix=iy;

iy=t;

}

**30.Assignment Operator without Operator Overloading**

#include<iostream>

using namespace std;

class assign

{

int x;

float y;

public:

void output();

assign(int,float);

};

assign::assign(int a,float b)

{

x=a;

y=b;

}

void assign::output()

{

cout<<x<<" "<<y<<endl;

}

int main()

{

assign a1(10,10.5);

assign a2(20,20.5);

a2=a1;

a1.output();

a2.output();

}

**31.Arithematic Operator with Operator Overloading**

#include<iostream>

using namespace std;

class class1

{

int x;

public:

class1();

class1(int a);

class1 operator+(class1 b);

void output();

};

class1::class1()

{

x=0;

}

class1::class1(int a)

{

x=a;

}

class1 class1:: operator+(class1 b)

{

class1 c1;

c1.x=x+b.x;

return(c1);

}

void class1::output()

{

cout<<"x= "<<x<<endl;

}

int main()

{

class1 obj1(10);

class1 obj2(20);

class1 obj3;

obj3=obj1+obj2;

obj1.output();

obj2.output();

obj3.output();

}

**32.Dynamic Allocation**

#include<iostream>

using namespace std;

int main()

{

int \*ptr1=new int;

int \*ptr2=new int;

int \*ptrsum=new int;

int \*ptrsub=new int;

int \*ptrmul=new int;

int \*ptrdiv=new int;

cout<<"Enter values: ";

cin>>\*ptr1>>\*ptr2;

\*ptrsum=\*ptr1+\*ptr2;

\*ptrsub=\*ptr1-\*ptr2;

\*ptrmul=\*ptr1\*\*ptr2;

\*ptrdiv=\*ptr1/(\*ptr2);

cout<<"ptrsum="<<\*ptrsum<<endl;

cout<<"ptrsub="<<\*ptrsub<<endl;

cout<<"ptrmul="<<\*ptrmul<<endl;

cout<<"ptrdiv="<<\*ptrdiv<<endl;

delete ptr1;

delete ptr2;

delete ptrsum;

delete ptrsub;

delete ptrdiv;

delete ptrmul;

}

**33.Multilevel Inheritance using Library Information System**

#include<iostream>

using namespace std;

class class1

{

private :

int rno;

char name[26];

public:

void getdata();

void output();

};

class class2:public class1

{

private:

int book\_no;

char book\_name[26];

public:

void getdata();

void output();

};

class class3:public class2

{

private:

int book\_issuedate, book\_return;

public:

void getdata();

void output();

};

void class1::getdata()

{

cout<<"Enter student name: ";

cin>>name;

cout<<"Enter Roll no: ";

cin>>rno;

}

void class2::getdata()

{

class1::getdata();

cout<<"Enter book name: ";

cin>>book\_name;

cout<<"Enter book number: ";

cin>>book\_no;

}

void class3::getdata()

{

class2:: getdata();

cout<<"Enter issue date: ";

cin>>book\_issuedate;

cout<<"Enter return date";

cin>>book\_return;

}

void class1::output()

{

cout<<"Student Name: "<<name<<endl<<"\tRoll No: "<<rno<<endl;

}

void class2::output()

{

class1::output();

cout<<"Book Number: "<<book\_no<<endl<<"\tBook Name: "<<book\_name<<"\t"<<endl;

}

void class3::output()

{

class2::output();

cout<<"Book issue date"<<book\_issuedate<<endl<<"\tBook return date"<<book\_return<<endl;

}

int main()

{

class3 c3;

c3.getdata();

c3.output();

}

**34.Overloading Constructors**

#include<iostream>

using namespace std;

class class1

{

int x;

public:

class1()

{

x=0;

}

class1(int n)

{

x=n;

}

int getdata()

{

return x;

}

};

int main()

{

class1 obj1[10];

class1 obj2[10]={1,2,3,4,5,6,7,8,9,10};

int i;

for(i=0;i<10;i++)

{

cout<<obj1[i].getdata()<<" ";

cout<<obj2[i].getdata();

cout<<endl;

}

}

**35.Overloading Comparison Operator**

#include<iostream>

using namespace std;

class class1

{

int x;

public:

class1();

class1(int a);

int operator<(class1 c1);

void output();

};

class1::class1()

{

x=0;

}

class1::class1(int a)

{

x=a;

}

int class1::operator<(class1 c1)

{

return(x<c1.x);

}

void class1::output()

{

cout<<x;

}

int main()

{

class1 obj1(10);

class1 obj2(200);

cout<<(obj1<obj2)<<endl;

cout<<(obj2<obj1)<<endl;

}

**36.Early Binding and Late Binding**

EARLY BINDING

#include<iostream>

using namespace std;

class class1

{

int x;

public: void output()

{

cout<<"one";

}

};

class derived2:public class1

{

int x;

public: void output()

{

cout<<"two";

}

};

class derived3: public derived2

{

int x;

public:void output()

{

cout<<"three";

}

};

int main()

{

class1 b1;

derived2 d1;

derived3 d2;

class1 \*ptr[3];

ptr[0]=&b1;

ptr[1]=&d1;

ptr[2]=&d2;

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

{

ptr[i]->output();

}

}

LATE BINDING

#include<iostream>

using namespace std;

class class1

{

int x;

public:virtual void output()

{

cout<<"one";

}

};

class derived2:public class1

{

int x;

public:virtual void output()

{

cout<<"two";

}

};

class derived3: public derived2

{

int x;

public:virtual void output()

{

cout<<"three";

}

};

int main()

{

class1 b1;

derived2 d1;

derived3 d2;

class1 \*ptr[3];

ptr[0]=&b1;

ptr[1]=&d1;

ptr[2]=&d2;

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

{

ptr[i]->output();

}

}