Government Girls' Polytechnic, Bilaspur

Class: CS 3rd sem , IT- 3rd sem
Name of the Lab: Programming Lab

Title of the Practical : **OOPS in C ++ Lab**

Teachers Assessment 30 End Semester Examination :70

EXPERIMENT NO:-1

1.OBJECTIVE :- Program showing the use of data type and basics of C^{++}: **(a)** Program to calculate sum of two numbers. **(b)** Program to calculate simple interest.

2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or windows xp,vista.

3.SOFTWARE REQUIRED :-CPP software

4.THEORY :-

Sum of two number is the mathematical function like a + b
Calculate simple interest is the mathematical function like SI = (P*R*T)/100

5.PROGRAM INPUTS & OUTPUT :-

```
   //  Sum Two Number
#include<iostream.h>
#include<conio.h>
void main( )
{
double A,B,SUM;
clrscr( );
cout<<"\n Enter Number A = ";
cin>>A;
cout<<"\n Enter Number B = ";
cin>>B;
SUM = A+B;
cout<<"\n SUM A+B = "<<SUM;
getch( );
}
```

OUTPUT :- Enter Number A =
Enter Number B =
SUM = A+B

```
// Simple Intrest
#include<iostream.h>
#include<conio.h>
void main( )
{
float P,R,T,SI;
clrscr( );
cout<<"\n Enter Principle P = ";
cin>>P;
cout<<"\n Enter Rate R = ";
cin>>R;
cout<<"\n Enter Time T = ";
cin>>T;
SI = (P*R*T)/100;
cout<<"\n Simple Intrest SI =  "<<SI;
getch( );
}
```

OUTPUT :- Enter Principle P =

```
Enter Rate R =
Enter Time T =
SI = (P*R*T)/100
```

6.OBSERVATIONS :- Task is performed.

EXPERIMENT NO:- 2

1.OBJECTIVE :- Program showing C^{++} programs based on object and classes: **(a)** Program showing arrays within a class. **(b)** Program containing arrays of object.

2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or windows xp,vista.

3.SOFTWARE REQUIRED :-CPP software

4.THEORY :-

Arrays Of Objects -1.Arrays of variables that are of the type class.
2.Such variable are called arrays of objects.

Arrays within a class-1.The arrays can be used as member variables in a class.
2.The arrays variable declared as a private member of the class array, can be used in the member functions,like any other array variable.

5.PROGRAM INPUTS & OUTPUT :-

// Arrays Of Objects

<u>Program :</u>

```
#include<iostream.h>
#include<conio.h>
class employee
{
char name[30];
float age;
public:
void getdata(void);
void putdata(void);
};
void employee :: getdata(void)
{
cout<<"Enter name = ";
cin>>name;
cout<<"Enter age = ";
cin>> age;
}
void employee :: putdata(void)
{
cout<<"name = "<<name<<"\n";
cout<<"age = "<<age<<"\n";
}
const int size=3;
int main( )
{
clrscr( );
employee manager[size];
for(int i=0;i<size;i++)
{
cout<<"\nDetails of manager = "<<i+1<<"\n";
manager[i].getdata( );
}
cout<<"\n";
for(i=0;i<size;i++)
{
cout<<"\nManager = "<<i+1<<"\n";
manager[i].putdata( );
}
getch( );
}
```

OUTPUT :- Enter name =
Enter age =

//Arrays within a class

<u>**Program :**</u>

```
#include<iostream.h>
#include<conio.h>
class item
{
int number[10];
char itcode[10];
float cost[10];
public:
void getdata(void);
void putdata(void);
int i,n;
};
void item::getdata()
{
cout<<"\nEnter total no. of items= ";
cin>>n;
for(i=0;i<n;i++)
{
cout<<"\nEnter item code= ";
cin>>itcode[i];
cout<<"Enter no. of items= ";
cin>>number[i];
cout<<"\nEnter cost= ";
cin>>cost[i];
}
}
void item::putdata()
{
for(i=0;i<n;i++)
{
cout<<"\n Item code= "<<itcode[i];
cout<<"\n Number of Items= "<<number[i];
cout<<"\n total cost= "<<cost[i]*number[i];
}
}
void main()
{
clrscr();
item x;
x.getdata();
x.putdata();
getch();
}
```

OUTPUT :- Enter total no. of items=

```
Enter item code=
Enter no. of items=
Enter cost=
```

6.OBSERVATIONS :- Task is performed.

EXPERIMENT NO:- 3

1.OBJECTIVE :- At least one C++ program based on the following : **(a)** Constructors and destructors-Program to generate a series of Fibonacci numbers using constructor and destructor. **(b)** Overloading unary operator-Program to generate a Fibonacci series by overloading a postfix operator. **(c)** Overloading binary operator- Program to perform overloading of a plus operator for finding the sum of two given class objects.

2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or windows xp,vista.

3.SOFTWARE REQUIRED :-CPP software

4.THEORY :-

Constructor :

A constructor is a 'special' member function whose task is to initialize the objects of its class.

Destructor :

A destructor,as the name implies, is used to destroy the objects that have been created by a constructor.

Overloading Unary Minus

Unary operator works with one operand.

Overloading Binary Operator

1>when a binary operator is overloaded using a member operator function,then the operator function,then the operator function will have only one argument.

2>This argument will receive an object which will go to the right of binary operator.

5.PROGRAM INPUTS & OUTPUT :-

generation of the fibonacci series using constructor

```
#include<iostream.h>
#include<conio.h>
class fibonacci
{
private:
unsigned long int f0,f1,fib;
public:
fibonacci()
{  //constructor
f0=0;
f1=1;
fib=f0+f1;
}
void increment()
{
f0=f1;
f1=fib;
fib=f0+f1;
}
void display()
{
cout<< fib ;
}
};//end of class construction
void main(void)
{
fibonacci number;
clrscr();
```

```
    for(int i=0;i<=15;++i)
    {
    number.display();
    number.increment();
    }
    getch();
    }
//Destructors
#include<iostream.h>
#include<conio.h>
int count = 0;
class alpha
{
public:
alpha()
{
count++;
cout<<"\n No. of object created "<<cout;
}
~alpha()//(~)MSG :-multiple declaration for 'alpha :: alpha()'
{
cout<<"\n No. of object destroyed "<<count;
count--;
}
};
int main()
{
clrscr();
cout<<"\n\nENTER MAIN\n";
alpha A1,A2,A3,A4;
{
cout<<"\n\nENTER BLOCK1\n";
alpha A5;
}

{
cout<<"\n\nENTER BLOCK2\n";
alpha A6;
}
cout<<"\n\nRE-ENTER MAIN\n";
getch();
}
```

OUTPUT :- //ENTER MAIN

```
//No.of object created 1
//No.of object created 2
//No.of object created 3
//No.of object created 4
//ENTER BLOCK1
//NO.of object created 5
//NO.of object destroyed 5
//ENTER BLOCK2
//NO.of object created 5
//NO.of object destroyed 5
//RE-ENTER MAIN
```

// Overloading Unary Minus

<u>Program :</u>

```
#include<iostream.h>
#include<conio.h>
class space
{
int x;
int y;
int z;
public:
void getdata(int a,int b,int c);
void display(void);
void operator-( );
};
void space :: getdata(int a,int b,int c)
{
x=a;
y=b;
z=c;
}
void space :: display(void)
{
cout<<x<<" ";
cout<<y<<" ";
cout<<z<<"\n";
}
void space :: operator-( )
{
x= -x;
y= -y;
z= -z;
}
int main( )
{
clrscr( );
space s;
s.getdata(10, -20, 30);
cout<<"\n s : ";
s.display( );
-s;
cout<<"\n s : ";
s.display( );
getch( );
}
```

// Overloading Binary Operator

<u>Program :</u>

```
#include<iostream.h>
#include<conio.h>
class complex
{
float x;
float y;
public:
complex( ){ }
complex(float real,float imag)
{ x=real; y=imag;}
complex operator+(complex);
void display(void);
};
complex complex :: operator+(complex c)
{
complex temp;
temp.x = x + c.x;
temp.y = y + c.y;
return(temp);
}
void complex :: display(void)
{
cout<<x<<" +j "<<y<<"\n";
}
int main( )
{
 clrscr( );
complex c1,c2,c3;
c1=complex(2.5,3.5);
c2=complex(1.6,2.7);
c3=c1+c2;
cout<<"c1 = ";c1.display( );
cout<<"c2 = ";c2.display( );
cout<<"c3 = ";c3.display( );
getch( );
}
```
6.OBSERVATIONS :- Task is performed.

EXPERIMENT NO:- 4

1.OBJECTIVE :- C^{++} program based on the following : **(a)**Inheritance- Program to demonstrate how ambiguity is avoided in single inheritance using scope resolution operator.**(b)** Multiple Inheritance- Program to demonstrate how ambiguity is avoided in multiple inheritance using scope resolution operator.

2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or windows xp,vista.

3.SOFTWARE REQUIRED :-CPP software

4.THEORY :- Inheritance

1>Inheritance is the process by which objects of one class acquire the properties of objects of another class.
2>The mechanism of deriving a new class from an old one is called inheritance.
3>The old class is refered to as the base class and the new one is called the derived class or subclass.

Multiple Inheritance

1>A derived class with several base classes is called multiple Inheritance.

5.PROGRAM INPUTS & OUTPUT :-

Program :

// Single Inheritance : Private
```
#include<iostream.h>
#include<conio.h>
class A
{
int a;
public:
int b;
void get_ab( );
int get_a(void);
void show_a (void);
};
class D : private A
{
int c;
public:
void mul(void);
void display(void);
};
void A :: get_ab(void)
{
cout<<"Enter Values for a and b : ";
cin>>a >> b;
}
int A :: get_a( )
{
return a;
}
void A :: show_a( )
{
cout << "a = " << a <<"\n";
}
void D :: mul( )
{
get_ab( );
c = b * get_a( );
}
void D :: display( )
{
show_a( );
cout << "b = " << b<<"\n";
cout << "c = " << c <<"\n\n";
}
int main( )
{
clrscr( );
D d;
//d.get_ab( );
d.mul( );
//d.show_a( );
d.display( );

//d.b = 20;
d.mul( );
d.display( );
getch( );
}
```

Program :

```
// Multiple Inheritance
#include<iostream.h>
#include<conio.h>
class M
{
protected:
int m;
public:
void get_m(int);
};
class N
{
protected:
int n;
public:
void get_n(int);
};
class P : public M, public N
{
public:
void display(void);
};
void M :: get_m(int x)
{
m = x;
}
void N :: get_n(int y)
{
n = y;
}
void P :: display(void)
{
cout<<"m = "<< m <<"\n";
cout<<"n = "<< n <<"\n";
cout<<"m*n = "<< m*n <<"\n";
}
int main( )
{
clrscr( );
P p;
p.get_m(10);
p.get_n(20);
p.display( );
getch( );
}
```

OUTPUT :-

```
m=10
n=20
m*n=200
```

6.OBSERVATIONS :- Task is performed.

EXPERIMENT NO:- 5

1.OBJECTIVE :- One C^{++} program based on the following: **(a)**Polymorphism-Program to illustrate the dynamicbinding of member function of a class. **(b)** Overloading- Program to illustrate function overloading.

2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or windows xp,vista.

3.SOFTWARE REQUIRED :-CPP software

4.THEORY :-

Polymorephism is a greek world which means ability to take more than one form.

Function Overloading

1>we can use the same function name to creat functions that perform a variety of different tasks.

2>The function would perform the different operations depending on the argument lists in the function call.

5.PROGRAM INPUTS & OUTPUT :-

```
//Polymorephism
#include<iostream.h>
#include<conio.h>
class base_x
{
int p;
public:
void get_basedata(int t)
{
p=t;
}
virtual void test();
};
class der_x:public base_x
{
int q;
public:
void get_derdata(int u)
{ q=u;
}
void test();
};
void base_x::test()
{
cout<<"\nYou are in the base function:";
if(i%2==0)
cout<<"\nThe number is divisible by 2";
else
cout<<"\n The number is not divisible by 2";
}
void der_x::test()
{
cout<<"\n you are in the derived function:";
if(j%3==0)
cout<<"\n The number is divisible by 3";
else
cout<<"\n The number is not divisible by 3";
}
void main()
{
base_x*p;
p=new base_x();
p-> get_basedata(15);
p-> test();
der_x obj;
obj.get_derdata(21);
p=&obj;
p-> test();
getch();
}
```

<u>Program :</u>
// Function Overloading

```
#include<iostream.h>
#include<conio.h>
int volume(int s)                   //cube
{
return(s*s*s);
}
double volume(double r,int h)     //cylinder
{
return(3.14519*r*r*h);
}
long volume(long l,int b,int h)  //rectangular box
{
return(l*b*h);
}
void main( )
{
clrscr( );
cout<<"\n"<< volume(10);
cout<<"\n"<< volume(2.5,8);
cout<<"\n"<< volume(100L,75,15);
getch( );
}
```

OUTPUT :-10

6.OBSERVATIONS :- Task is performed.

EXPERIMENT NO:- 6

1.OBJECTIVE :- Some C^{++} program should be conducted on each of the following: **(a)** 2d array sorting- Program to sort 2D array using linear representation of the array **(b)** Pointer to objects – Program to illustrate the use of pointer to objects.**(d)** Use of this Pointer- Program to illustrate the use of this pointer. **(c)** Pointers to derived class- Program to illustrate the use of pointers to derived class

2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or windows xp,vista.

3.SOFTWARE REQUIRED :-CPP software

4.THEORY :- Pointers to objects

1>A pointer can point to an object created by a class.
2>An object pointer is used to access the public members of an object.

5.PROGRAM INPUTS & OUTPUT :-

//write a program for sort an array

```
#include<iostream.h>
#include<conio.h>
void main()
{
 clrscr();
int A[20],i,j,l,x;
cout<<"Enter the limit of an array:";
cin>>l;
cout<<"Enter the array elements:";
for(i=0;i<l;i++)
{
cin>>A[i];
}
for(j=0;j<l-1;j++)
{
if(A[j]>A[j+1])
{
x=A[i];
A[j]=A[j+1];
A[j+1]=x;
}
}
for(i=0;i<l;i++)
{
cout<<A[i];
}
getch();
}
```

OUTPUT :- Enter the limit of an array:
Enter the array elements:

//Pointers to objects

<u>Program :</u>

```
#include<iostream.h>
#include<conio.h>
class c1
```

```
{
int i;
public:
c1(int j) {i=j;}
int get_i() {return i;}
};
int main()
{
clrscr();
c1 ob(88),*p;
p=&ob;//get address of ob
cout<<p->get_i();//use ->to call get_i()
getch();
}
```

OUTPUT :-

//Pointers to Derived class

```
#include<iostream.h>
#include<conio.h>
class base
{
int i;
public:
void set_i(int num) {i=num;}
int get_i(){return i;}
};
class derived:public base
{
int j;
public:
void set_j(int num) {j=num;}
int get_j(){return j;}
};
int main()
{
clrscr();
base *bp;
derived d;
bp=&d;//base pointer points to derived object
//access derived object using base pointer
bp->set_i(10);
cout<<bp->get_i()<<" ";
/*The following won't work.you can't access elements of a
derived class using a base class pointer.
bp->set_j(88);//error
cout<<bp->get_j();//error
*/
getch();
}
```

6.OBSERVATIONS :- Task is performed.

EXPERIMENT NO:- 7

1.OBJECTIVE :- At least two program based on file handling: **(a)**Program using getline() and write() functions **(b)** Program showing read and write operations.

2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or windows xp,vista.

3.SOFTWARE REQUIRED :-CPP software

4.THEORY :-

writes a string to the text window on the screen.

5.PROGRAM INPUTS & OUTPUT :-

//getline()and write()function in program

```
//getline()  function
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int size=20;
char city[20];
cout<<"\nEnter city name :";
cin>>city;
cout<<"\nCity name :"<<city<<"\n\n";
cout<<"Enter city name again\n";
cin.getline(city,size);
cout<<"City name now"<<city<<"\n\n";
cout<<"Enter another city name :";
cin.getline(city,size);
cout<<"\nNew city name :"<<city<<"\n\n";
getch();
}
//  write function()
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
clrscr();
char *str ="RAVI";
int i,len =strlen(str);
for(i=0;i<=len;i++)
{
cout.write(str,i);
cout<<"\n";
}
for(i=len;i>0;i--)
{
cout.write(str,i);
cout<<"\n";
}
getch();
}
```

//**Read and write operation**

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<iomanip.h>
const char *filename ="BINARY";
int main()
{
float height[4]={175.5,153.0,167.25,160.70};
ofstream outfile;
outfile.open(filename);
outfile.write((char*)&height,sizeof(height));
outfile.close();//close the file for reading
for(int i=0;i<4;i++)//clear array from memory.
height[i]=0;
ifstream infile;
infile.open(filename);
infile.read((char*)&height,sizeof(height));
for(i=0;i<4;i++)
{
cout.setf(ios::showpoint);
cout<<"show(10)"<< setprecision(2)<<height[i];
}
infile.close();
getch();
}
```

OUTPUT :-

6.OBSERVATIONS :- Task is performed.

EXPERIMENT NO:- 8

1.OBJECTIVE :- At least four Ç$^{++}$ programs based on Graphics function :**(a)** Progarm to draw a line.**(b)** Program to draw a circle.**(c)** Program to draw a rectangle. **(d)** Program to draw a polygon.

2.HARDWARE & SYSTEM SOFTWARE REQUIRED :- P2,P4 System or windows xp,vista.

3.SOFTWARE REQUIRED :-CPP software

4.THEORY :-

Circle draws a circle in the current drawing color.
Draws a rectangle (graphics mode)

5.PROGRAM INPUTS & OUTPUT :-

//Line()

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
class sample
{
private:
int x1,x2,y1,y2;
char msg[80];
public:
void getdata();
void setdata();
void draw_line();
};
void sample :: getdata()
{
cout<<"\n      Enter the initial coordinates of x and y ";
cout<<"\n      x = :";
cin>>x1;
cout<<"\n      y = :";
cin>>y1;
}
void sample :: setdata()
{
x2 = getmaxx()/2;
y2 = getmaxy()/2;
setcolor(getmaxcolor());
}
void sample :: draw_line()
{
moveto(x1,y1);
sprintf(msg,"(%d,%d)",getx(),gety());
outtextxy(20,30,msg);
lineto(x2,y2);
sprintf(msg,"(%d,%d)",getx(),gety());
outtext(msg);
}
void main(void)
{
```

```
sample obj;
int gdriver = DETECT,gmode;
initgraph(&gdriver,&gmode,"\\tc\\bgi");
obj.getdata();
obj.setdata();
obj.draw_line();
getch();
cleardevice();
closegraph();
}
```

//circle()

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
const int radius = 100;
class sample
{
private :
int midx,midy;
public:
void getdata();
void draw_circle();
};
void sample :: getdata()
{
midx = getmaxx()/2;
midy = getmaxy()/2;
}
void sample :: draw_circle()
{
circle(midx,midy,radius);
}
void main(void)
{
sample obj;
int gdriver = DETECT,gmode;
initgraph(&gdriver,&gmode,"\\tc\\bgi");
obj.getdata();
obj.draw_circle();
getch();
cleardevice();
closegraph();
}
```

//Rectangle()

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
class sample
{
private:
int midx,midy;
int left,top,right,bottom;
public:
```

```
void getdata();
void setdata();
void draw_rectangle();
};
void sample :: getdata()
{
midx = getmaxx()/2;
midy = getmaxy()/2;
}
void sample :: setdata()
{
left = (midx/2)-50;
top = (midy/2)-50;
right = (midx/2)+50;
bottom = (midy/2)+50;
}
void sample :: draw_rectangle()
{
rectangle(left,top,right,bottom);
}
void main(void)
{
sample obj;
int gdriver =DETECT,gmode;
initgraph(&gdriver,&gmode,"\\tc\\bgi");
obj.getdata();
obj.setdata();
obj.draw_rectangle();
getch();
cleardevice();
closegraph();
}
```

//Polygon()

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
class sample
{
private:
int midx,midy;
int poly[10];
public:
void getdata();
void setdata();
void draw_poly();
};
void sample :: getdata()
{
midx = getmaxx()/2;
midy = getmaxy()/2;
}
void sample :: setdata()
{
poly[0] = 20;
poly[1] = midy/2;
poly[2] = midx-20;
poly[3] = 20;
poly[4] = midx-50;
poly[5] = midy-20;
poly[6] = midx/2;
poly[7] = midy/2;
poly[8] = poly[0];
poly[9] = poly[1];
}
void sample :: draw_poly()
{
drawpoly(5,poly);
}
void main(void)
{
sample obj;
int gdriver = DETECT,gmode;
initgraph(&gdriver,&gmode,"\\tc\\bgi");
obj.getdata();
obj.setdata();
obj.draw_poly();
getch();
cleardevice();
closegraph();
}
```

OUTPUT :-

6.OBSERVATIONS :- Task is performed.