

C++ assignment

Submitted To:

SMRUTI MA’M

Submitted By:

NITISH RAJ

IT-3

ROLL NO: 11720068

1. **Operator Overloading:**

* #include<iostream>

using namespace std;

class complex

{

int a,b;

public:

void setdata(int x,int y)

{

a=x;

b=y;

}

void showdata()

{

cout<<"values of a:"<<a<<"\nvalue of b:"<<b<<"\n";

}

complex operator+(complex c)

{

complex temp;

temp.a=a+c.a;

temp.b=b+c.b;

return(temp);

}

};

int main()

{

complex c1,c2,c3;

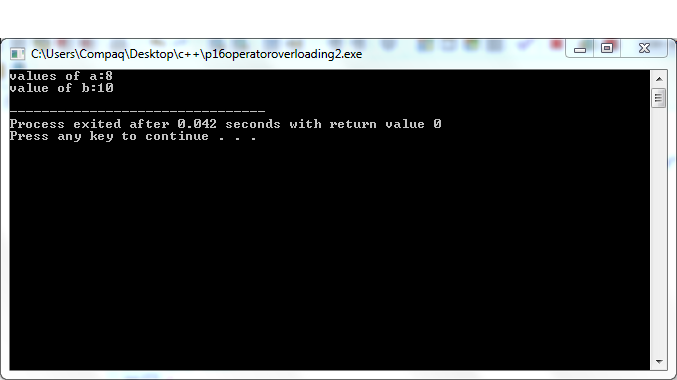
c1.setdata(2,5);

c2.setdata(6,5);

c3=c1+c2;

c3.showdata();

}



1. **Template:**

* #include<iostream>

using namespace std;

template<class T>

class Calculator

{

private:

T num1;

T num2;

public:

Calculator(T n1,T n2)

{

num1=n1;

num2=n2;

}

void display()

{

cout<<"NUMBERS ARE: "<<num1<<"and"<<num2<<endl;

cout<<"\n addition:"<<add()<<"\n";

cout<<"\n subtration:"<<sub()<<"\n";

cout<<"\n division:"<<div()<<"\n";

cout<<"\n multiplication:"<<mul()<<"\n";

}

T add()

{

return num1+num2 ;

}

T sub()

{

return num1-num2 ;

}

T div()

{

return num1/num2 ;

}

T mul()

{

return num1\*num2 ;

}

};

int main()

{

cout<<"\nint results:"<<endl;

Calculator<int>intcalc1(10,5); //declaration of class object

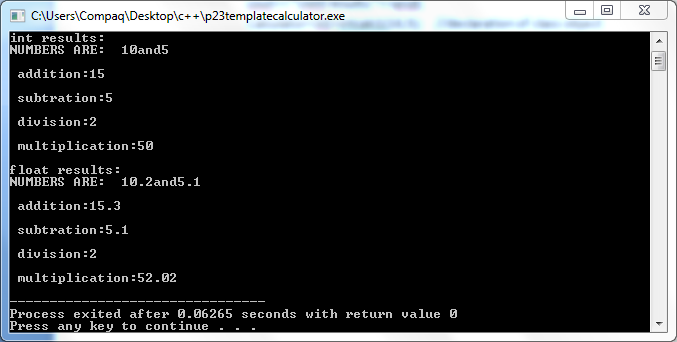
intcalc1.display();

cout<<"\nfloat results:"<<endl;

Calculator<float>floatcalc2(10.2,5.1);

floatcalc2.display();

}



1. **Exception handling:**

* **#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int a,b;**

**cout<<"ENTER THE NUMBERS:"<<endl;**

**cin>>a>>b;**

**int x=a-b;**

**try**

**{**

**if(x!=0)**

**{**

**cout<<"result:(a/x)="<<a/x<<endl;**

**}**

**else**

**{**

**throw(x);**

**}**

**}**

**catch(int i)**

**{**

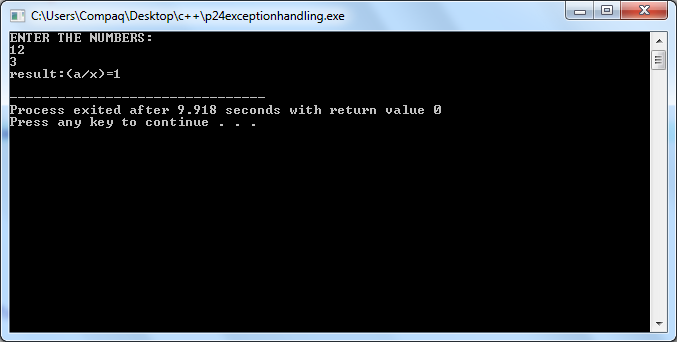
**cout<<"EXCEPTION CAUGHT:divide by zero"<<"\n";**

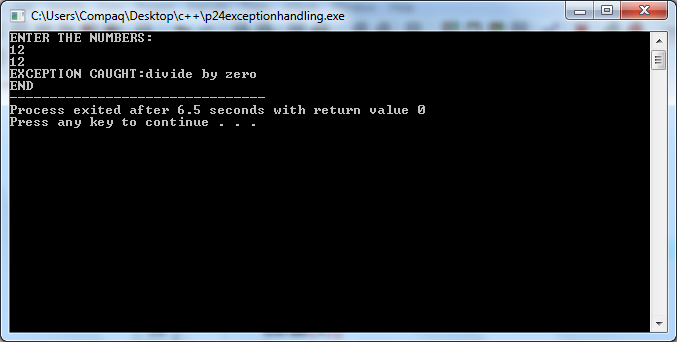
**cout<<"END";**

**return 0;**

**}**

**}**

****

****

1. N**ameless object:**

* **#include<iostream>**

**using namespace std;**

**class nameless**

**{**

**private:**

**int a;**

**public:**

**nameless()**

**{**

**cout<<"\nCONSTRUCTOR:\n";**

**}**

**~nameless()**

**{**

**cout<<"\nDESTRUCTOR:\n";**

**}**

**};**

**int main()**

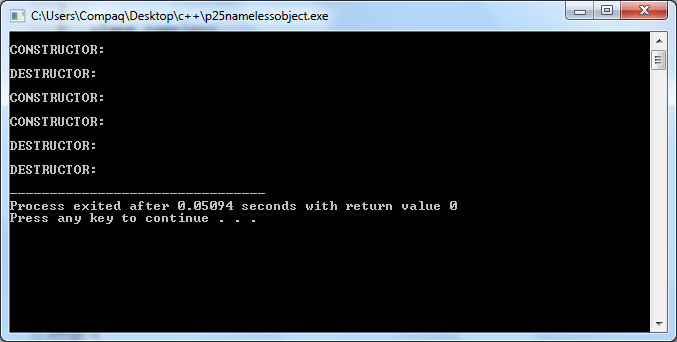
**{**

**nameless();**

**nameless n1; //object created**

**nameless n2;**

**}**



1. C**onstructor and destructor in base class N derived class:**

* **#include<iostream>**

**using namespace std;**

**class A**

**{**

**int x;**

**public:**

**A(int i)**

**{**

**x=i;**

**cout<<"\nA is initialized:\n";**

**}**

**void show\_x()**

**{**

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

**}**

**};**

**class B**

**{**

**float y;**

**public:**

**B(float j)**

**{**

**y=j;**

**cout<<"\nB is initialized:\n";**

**}**

**void show\_y()**

**{**

**cout<<"y="<<y<<endl;**

**}**

**};**

**class C:public A,public B**

**{**

**int z;**

**public:**

**C(int a,float b,int c):A(a),B(b)**

**{**

**z=c;**

**cout<<"\nC is initialized:\n";**

**}**

**void show\_z()**

**{**

**cout<<"z="<<z<<endl;**

**}**

**};**

**int main()**

**{**

**cout<<"\nCONSTRUCTOR IN BASE CLASS and DERIVED CLASS:\n";**

**C obj(1,2,3);**

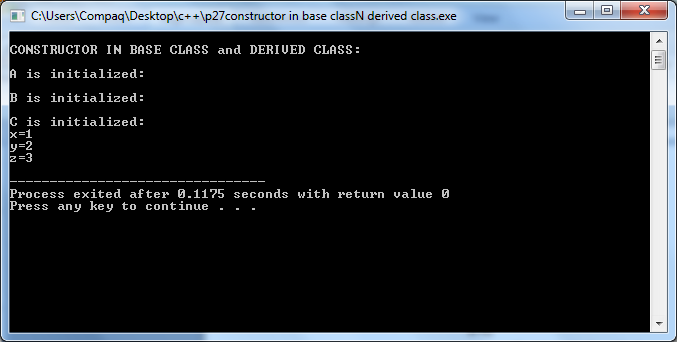
**obj.show\_x();**

**obj.show\_y();**

**obj.show\_z();**

**return 0;**

**}**

****

**6.** **nested class:**

* **#include<iostream>**

**using namespace std;**

**class one**

**{**

**public: class two**

**{**

**private:**

**int a;**

**public:**

**void display(int x)**

**{**

**a=x;**

**cout<<"print value of a="<<a<<"\n";**

**}**

**};**

**};**

**int main()**

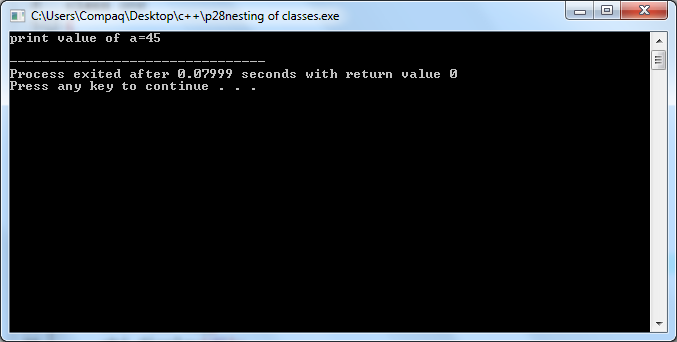
**{**

**one::two obj;**

**obj.display(45);**

**return 0;**

**}**

****

**7.** **constructor:**

* **#include<iostream>**

**using namespace std;**

**class complex**

**{**

**private:**

**int a,b;**

**public:**

**complex(int x,int y) //parameterized constructor**

**{**

**a=x;**

**b=y;**

**cout<<"values:\n"<<a<<"\n"<<b<<"\n";**

**}**

**complex(int k) //parameterized constructor**

**{**

**a=k;**

**cout<<"value of a:\n"<<a<<"\n";**

**}**

**complex() //default constructor**

**{**

**cout<<"no argument,default constructor"<<"\n";**

**}**

**complex(complex &c)**

**{**

**a=c.a;**

**b=c.b;**

**cout<<"values:\n"<<a<<"\n"<<b;**

**}**

**};**

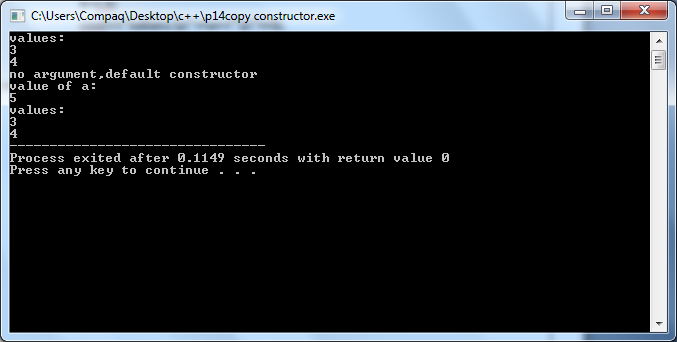
**int main()**

**{**

**complex c1(3,4),c2,c3(5);**

**complex c4(c1);**

**}**

****

**8.** **static member function:**

* **#include<iostream>**

**using namespace std;**

**class item**

**{**

**int number;**

**static float count;**

**public:**

**void getdata(int x)**

**{**

**cout<<"STATIC MEMBER FUNCTION PROGRAM:\n";**

**number=x;**

**cout<<"\n number:"<<x<<"\n";**

**}**

**static void getcount(int y)**

**{**

**count=y;**

**cout<<"\n count:"<<y<<"\n";**

**}**

**};**

**float item::count=3.5;**

**int main()**

**{**

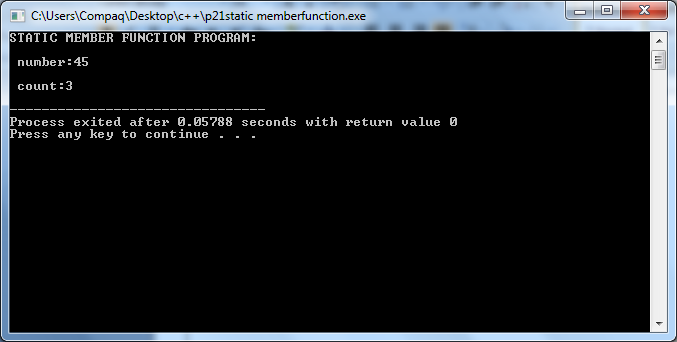
**item n;**

**n.getdata(45);**

**item::getcount(3);**

**return 0;**

**}**

****

**9.** **static data member:**

* **#include<iostream>**

**using namespace std;**

**class item**

**{**

**int number; //instance member function**

**static float count; //static member variable OR class variable**

**public:**

**void getdata(int x)**

**{**

**number=x;**

**cout<<"\nvalue of number:"<<x;**

**count++;**

**}**

**void getcount()**

**{**

**cout<<"\n\nvalue of count="<<count;**

**}**

**};**

**float item::count=1;**

**int main()**

**{**

**item a1;**

**cout<<"\ninstance member function:";**

**a1.getcount();**

**a1.getdata(5);**

**a1.getdata(10);**

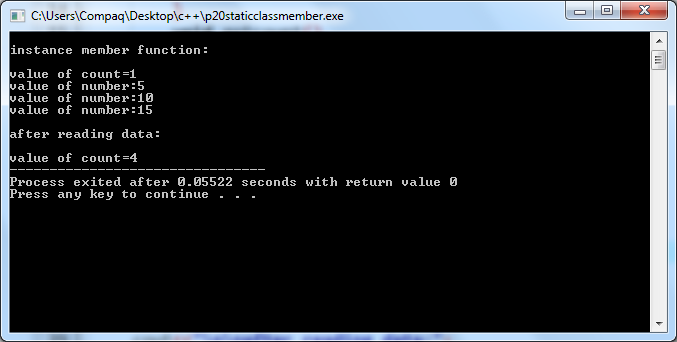
**a1.getdata(15);**

**cout<<"\n\nafter reading data:";**

**a1.getcount();**

**return 0;**

**}**

****

**10. function using inline:**

* **#include<iostream>**

**using namespace std;**

**inline float mul(float x,float y)**

**{**

**return(x\*y);**

**}**

**inline float div(float p,float q)**

**{**

**return(p/q);**

**}**

**inline float add(float w,float r)**

**{**

**return(w+r);**

**}**

**inline float sub(float m,float n)**

**{**

**return(m-n);**

**}**

**int main()**

**{**

**float a=12.345;**

**float b=9.82;**

**cout<<"FUNCTION USING INLINE:\n\n";**

**cout<<"\nmultiplication="<<mul(a,b)<<"\n";**

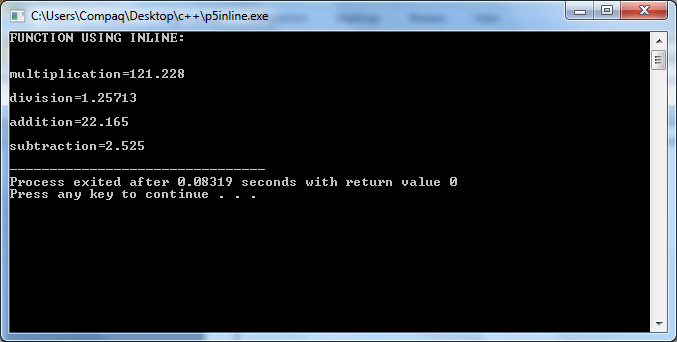
**cout<<"\ndivision="<<div(a,b)<<"\n";**

**cout<<"\naddition="<<add(a,b)<<"\n";**

**cout<<"\nsubtraction="<<sub(a,b)<<"\n";**

**return 0;**

**}**

****

**11.** **default argument:**

* **#include<iostream>**

**#include<conio.h**

**using namespace std;**

**float amount(float p,float t=2,float r=0.06);**

**float amount(float p,float t,float r)**

**{**

**cout<<"principal amount= "<<p<<"\n";**

**cout<<"rate of interest= "<<r<<"\n";**

**cout<<"time= "<<t<<"\n";**

**cout<<"interest amount="<<(p\*r\*t)<<"\n\n";**

**}**

**int main ()**

**{**

**cout<<"results on amount on <2000>"<<amount(2000)<<"\n";**

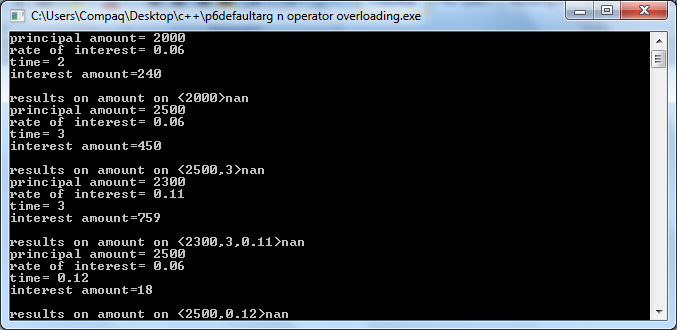
**cout<<"results on amount on <2500,3>"<<amount(2500,3)<<"\n";**

**cout<<"results on amount on <2300,3,0.11>"<<amount(2300,3,0.11)<<"\n";**

**cout<<"results on amount on <2500,0.12>"<<amount(2500,0.12)<<"\n";**

**getch();**

**}**



**12.** **friend function:**

**#include<iostream>**

**#include<conio.h>**

**using namespace std;**

**class base {**

**int val1, val2;**

**public:**

**void get() {**

**cout << "Enter two values:";**

**cin >> val1>>val2;**

**}**

**friend float mean(base ob);**

**};**

**float mean(base ob) {**

**return float(ob.val1 + ob.val2) / 2;**

**}**

**int main() {**

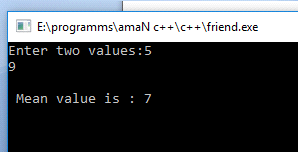
**base obj;**

**obj.get();**

**cout << "\n Mean value is : " << mean(obj);**

**getch();**

**}**



13. armstrong no:

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int n,p=0,r,temp;**

**cout<<"Enter number to check is it armstrong or not: ";**

**cin>>n;**

**temp=n;**

**while(temp!=0)**

**{**

**r=temp%10;**

**p=p+r\*r\*r;**

**temp=temp/10;**

**}**

**if(n==p)**

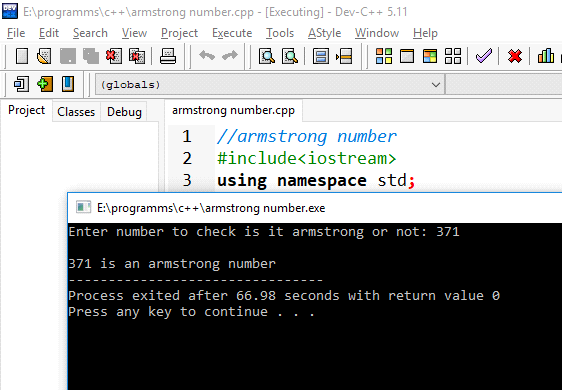
**cout<<"\n"<<n<<" is an armstrong number";**

**else**

**cout<<"\n"<<n<<" is not an armstrong number";**

**return 0;**

**}**



**14 call by reference:**

#include<iostream>

using namespace std;

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

{

int t;

t=\*x;

\*x=\*y;

\*y=t;

}

int main()

{

int a,b;

cout<<"enter a b: ";

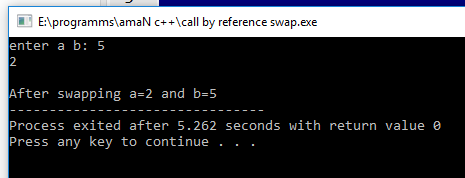
cin>>a>>b;

swap(&a,&b);

cout<<"\nAfter swapping a="<<a<<" and b="<<b;

return 0;

}



**15 palindrome:**

*}#include<iostream>*

*using namespace std;*

*int main()*

*{*

*int n,p=0,r,temp;*

*cout<<"Enter number to check is it palindrome or not: ";*

*cin>>n;*

*temp=n;*

*while(temp!=0)*

*{*

*r=temp%10;*

*p=p\*10+r;*

*temp=temp/10;*

*}*

*if(n==p)*

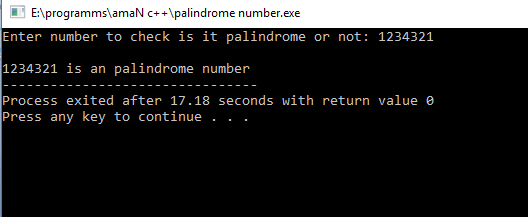
*cout<<"\n"<<n<<" is an palindrome number";*

*else*

*cout<<"\n"<<n<<" is not an palindrome number";*

*return 0;*

*}*



**16 prime or not:**

***#include<iostream>***

***using namespace std;***

***int main()***

***{***

***int n,i;***

***cout<<"Enter number to check is it prime or not: ";***

***cin>>n;***

***for(i=2;i<n;i++)***

***{***

***if(n%i==0)***

***break;***

***}***

***if(i==n)***

***{***

***cout<<"Prime Number";***

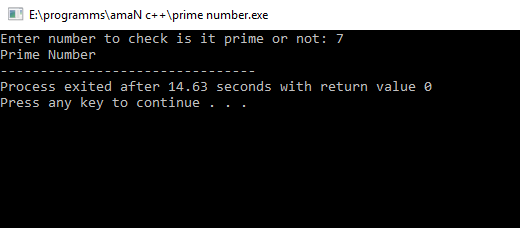
***}***

***else***

***cout<<"Not a Prime Number";***

***return 0;***

***}***



**19.hierarchical inheritance:**

**#include <iostream>**

**#include <conio.h>**

**using namespace std;**

**class person**

**{**

**char name[100],gender[10];**

**int age;**

**public:**

**void getdata()**

**{**

**cout<<"Name: ";**

**fflush(stdin); /\*clears input stream\*/**

**gets(name);**

**cout<<"Age: ";**

**cin>>age;**

**cout<<"Gender: ";**

**cin>>gender;**

**}**

**void display()**

**{**

**cout<<"Name: "<<name<<endl;**

**cout<<"Age: "<<age<<endl;**

**cout<<"Gender: "<<gender<<endl;**

**}**

**};**

**class student: public person**

**{**

**char institute[100], level[20];**

**public:**

**void getdata()**

**{**

**person::getdata();**

**cout<<"Name of College/School: ";**

**fflush(stdin);**

**gets(institute);**

**cout<<"Level: ";**

**cin>>level;**

**}**

**void display()**

**{**

**person::display();**

**cout<<"Name of College/School: "<<institute<<endl;**

**cout<<"Level: "<<level<<endl;**

**}**

**};**

**class employee: public person**

**{**

**char company[100];**

**float salary;**

**public:**

**void getdata()**

**{**

**person::getdata();**

**cout<<"Name of Company: ";**

**fflush(stdin);**

**gets(company);**

**cout<<"Salary: Rs.";**

**cin>>salary;**

**}**

**void display()**

**{**

**person::display();**

**cout<<"Name of Company: "<<company<<endl;**

**cout<<"Salary: Rs."<<salary<<endl;**

**}**

**};**

**int main()**

**{**

**student s;**

**employee e;**

**cout<<"Student"<<endl;**

**cout<<"Enter data"<<endl;**

**s.getdata();**

**cout<<endl<<"Displaying data"<<endl;**

**s.display();**

**cout<<endl<<"Employee"<<endl;**

**cout<<"Enter data"<<endl;**

**e.getdata();**

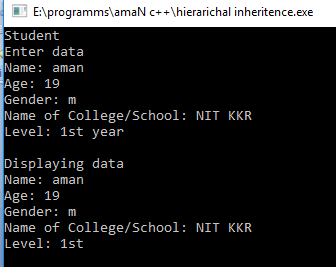
**cout<<endl<<"Displaying data"<<endl;**

**e.display();**

**getch();**

**return 0;**

**}**



20 multiple inheritance:

**#include <iostream>**

**using namespace std;**

**class Mammal {**

**public:**

**Mammal()**

**{**

**cout << "Mammals can give direct birth." << endl;**

**}**

**};**

**class WingedAnimal {**

**public:**

**WingedAnimal()**

**{**

**cout << "Winged animal can flap." << endl;**

**}**

**};**

**class Bat: public Mammal, public WingedAnimal {**

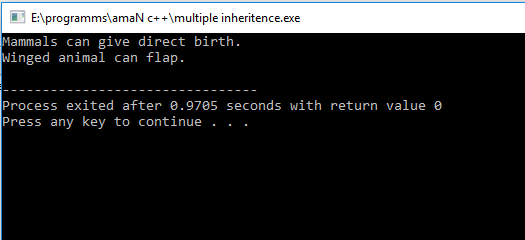
**};**

**int main()**

**{**

**Bat b1;**

**return 0;**



**21 single inheritance:**

**#include<iostream>**

**using namespace std;**

**class B**

**{**

**int a;**

**public:**

**int b;**

**void get\_ab();**

**int get\_a(void);**

**void show\_a(void);**

**};**

**class D: public B**

**{**

**int c;**

**public:**

**void mul(void);**

**void display(void);**

**};**

**void B::get\_ab(void)**

**{**

**a=5,b=10;**

**}**

**int B::get\_a()**

**{**

**return a;**

**}**

**void B::show\_a()**

**{**

**cout<<"a="<<a;**

**}**

**void D::mul()**

**{**

**c=b\*get\_a();**

**}**

**void D::display()**

**{**

**cout<<"\na= "<<get\_a();**

**cout<<"\nb= "<<b;**

**cout<<"\nc= "<<c;**

**}**

**int main()**

**{**

**D d;**

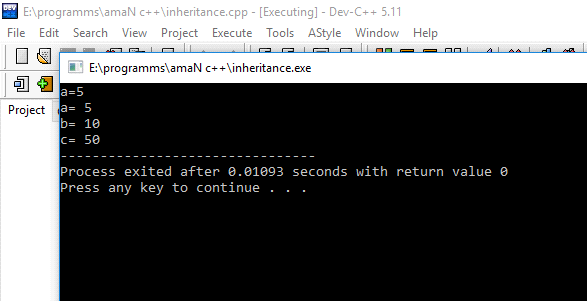
**d.get\_ab();**

**d.mul();**

**d.show\_a();**

**d.display();**

**}**



**23 . pointer to object:**

**#include<iostream>**

**using namespace std;**

**class item**

**{**

**int l,b,h;**

**public:**

**void getdata(int x,int y,int z)**

**{**

**l=x;**

**b=y;**

**h=z;**

**}**

**void showdata()**

**{**

**cout<<"\nl="<<l<<"\nb="<<b<<"\nh="<<h<<"\n";**

**}**

**};**

**int main()**

**{**

**item object,\*p;**

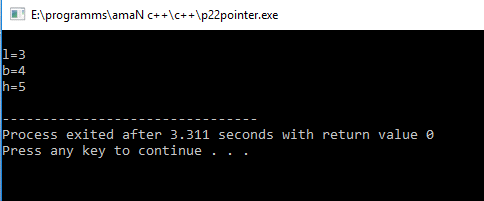
**p=&object;**

**p->getdata(3,4,5);**

**p->showdata();**

**return 0;**

**}**



**24 this pointer:**

**#include<iostream>**

**using namespace std;**

**class maxima**

**{**

**int no;**

**public:**

**maxima(int no=0)**

**{**

**this->no=no;**

**}**

**maxima & greater(maxima & x)**

**{**

**if(x.no>no)//no is calling object's no..... x.no is of object which is passed**

**return x;**

**else**

**return \*this;**

**}**

**void disp (void)**

**{**

**cout<<"greater value: "<<no;**

**}**

**};**

**int main()**

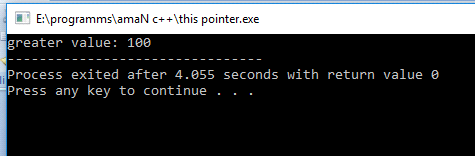
**{**

**maxima obj1(100), obj2(20), obj3;**

**obj3=obj1.greater(obj2);**

**obj3.disp();**

**}**



**25 abstract class:**

**#include<iostream>**

**using namespace std;**

**class Base**

**{**

**int x;**

**public:**

**virtual void fun() = 0;**

**int getX() { return x; }**

**};**

**class Derived: public Base**

**{**

**int y;**

**public:**

**void fun() { cout << "fun() called"; }**

**};**

**int main(void)**

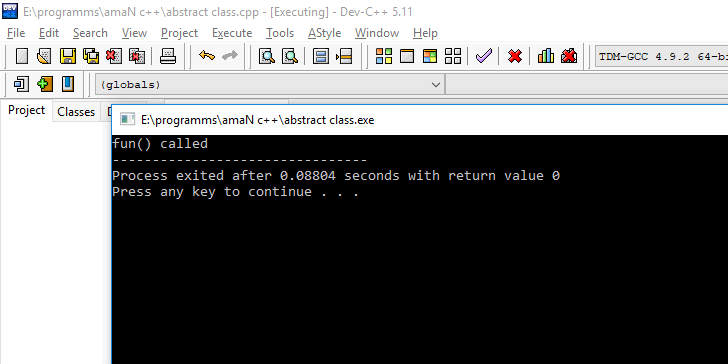
**{**

**Derived d;**

**d.fun();**

**return 0;**

**}**



**26 display using base class pointer:**

**#include <iostream>**

**using namespace std;**

**class Polygon {**

**protected:**

**int width, height;**

**public:**

**void set\_values (int a, int b)**

**{ width=a; height=b; }**

**};**

**class Rectangle: public Polygon {**

**public:**

**int area()**

**{ return width\*height; }**

**};**

**class Triangle: public Polygon {**

**public:**

**int area()**

**{ return width\*height/2; }**

**};**

**int main () {**

**Rectangle rect;**

**Triangle trgl;**

**Polygon \* ppoly1 = &rect;**

**Polygon \* ppoly2 = &trgl;**

**ppoly1->set\_values (4,5);**

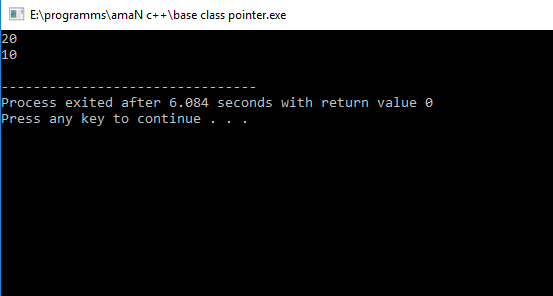
**ppoly2->set\_values (4,5);**

**cout << rect.area() << '\n';**

**cout << trgl.area() << '\n';**

**return 0;**

**}**



27 display using derived class pointer:

**#include <iostream>**

**using namespace std;**

**class Polygon {**

**protected:**

**int width, height;**

**public:**

**void set\_values (int a, int b)**

**{ width=a; height=b; }**

**};**

**class Rectangle: public Polygon {**

**public:**

**int area()**

**{ return width\*height; }**

**};**

**class Triangle: public Polygon {**

**public:**

**int area()**

**{ return width\*height/2; }**

**};**

**int main ()**

**{**

**Rectangle \*R;**

**Triangle \*T;**

**Rectangle rect;**

**Triangle trgl;**

**R= &rect;**

**T= &trgl;**

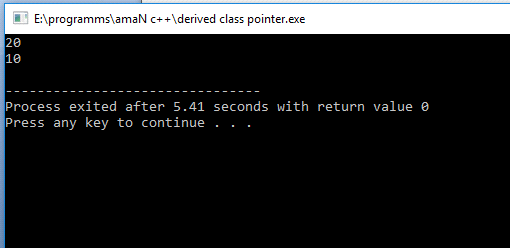
**R->set\_values (4,5);**

**T->set\_values (4,5);**

**cout << rect.area() << '\n';**

**cout << trgl.area() << '\n';**

**return 0;**



**28 virtual function:**

**#include<iostream>**

**using namespace std;**

**class base**

**{**

**public:**

**virtual void print ()**

**{ cout<< "print base class" <<endl; }**

**void show ()**

**{ cout<< "show base class" <<endl; }**

**};**

**class derived:public base**

**{**

**public:**

**void print ()**

**{ cout<< "print derived class" <<endl; }**

**void show ()**

**{ cout<< "show derived class" <<endl; }**

**};**

**int main()**

**{**

**base \*bptr;**

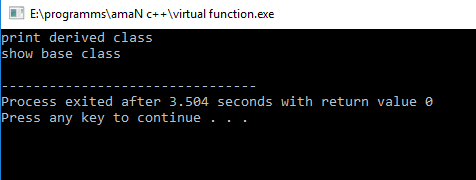
**derived d;**

**bptr = &d;**

**bptr->print();**

**bptr->show();**

**}**



**29 pure virtual function:**

**#include<iostream>**

**#include<conio.h>**

**using namespace std;**

**class base**

**{**

**private: int x;**

**float y;**

**public : virtual void getdata( );**

**virtual void display( );**

**};**

**class dev : public base**

**{**

**private: int roll;**

**char name[20];**

**public : void getdata( );**

**void display( );**

**};**

**void base :: getdata( ) { }**

**void base :: display( ) { }**

**void dev :: getdata( )**

**{**

**cout<<"Enter Roll of the Student " ;**

**cin>> roll;**

**cout<<"Enter name of the student";**

**cin>>name;**

**}**

**void dev :: display( )**

**{**

**cout<<" Enter name of the student"<<name<<endl;**

**cout<<" Roll no is :"<<roll <<endl;**

**}**

**int main( )**

**{**

**base \* ptr;**

**dev obj;**

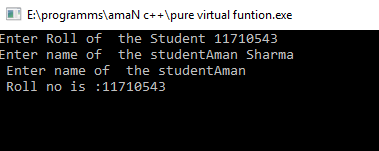
**ptr = &obj;**

**ptr -> getdata( );**

**ptr -> display( );**

**getch( );**

**}**



**30. 2-d array:**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int test[3][2] =**

**{**

**{2,-5},**

**{4, 0},**

**{9, 1}**

**};**

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

**{**

**for(int j = 0; j < 2; ++j)**

**{**

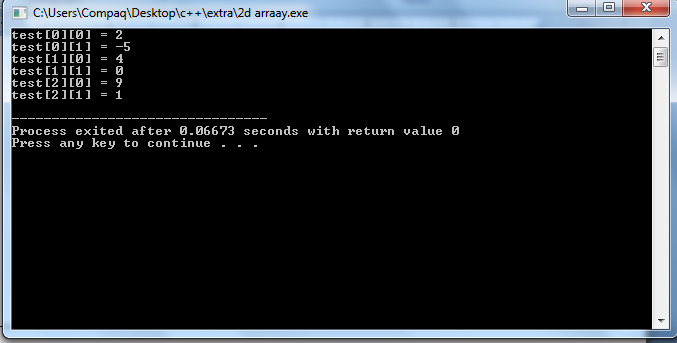
**cout<< "test[" << i << "][" << j << "] = " << test[i][j] << endl;**

**}**

**}**

**return 0;**

**}**

****

**End……….**