/\* THIS C++ PROGRAM ILLUSTRATES THE CONCEPT OF

\* C++ OBJECTS AS DATA TYPES\*/

/\*NAME : SAGAR GIRI, ROLL NO. 205 , SECTION : A

\* DATE : 21-FEB-2014\*/

#include <iostream>

using namespace std;

class distances

{

private:

int feet;

float inches;

public:

void getdistance()

{

cout<<"Enter feet and inches:";

cin>>feet>>inches;

}

void setdistance(int ft,float in)

{

feet = ft;

inches = in;

}

void display()

{

cout<<feet<<"\'-"<<inches<<"\"";

}

};

int main()

{

distances d1,d2; //OBJECTS d1 & d2 AS DATA TYPES

d1.getdistance();

d2.setdistance(6,4.3);

cout<<endl<<"distance one:";

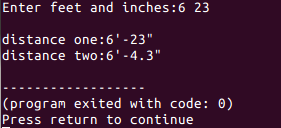
d1.display();

cout<<endl<<"distance two:";

d2.display();

}

OUTPUT:



/\* THIS C++ PROGRAM ILLUSTRATES THE CONCEPT OF CONSTRUCTORS.

\* CONSTRUCTORS:- CONSTRUCTORS ARE THE MEMBER FUNCTION WHICH

\* HAS THE SAME NAME AS THAT OF THE CLASS, AND IS CALLED

\* AUTOMATICALLY WHENEVER THE OBJECT IS CREATED.\*/

/\*NAME : SAGAR GIRI, ROLL NO. 205 , SECTION : A

\* DATE : 21-FEB-2014\*/

#include <iostream>

using namespace std;

class counter

{

private:

unsigned int count;

public:

counter()

{

count = 0;

}

void increment()

{

count++;

}

void display()

{

cout<<count<<endl;

}

};

int main()

{

counter c1,c2;

c1.display();

c2.display();

c1.increment();

c1.increment();

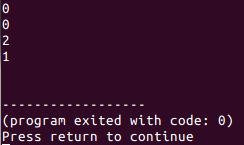
c2.increment();

c1.display();

c2.display();

}

OUTPUT:



/\* THIS IS A SIMPLE C++ PROGRAM ILLUSTRATES THE CONCEPT OF

\* C++ CLASS AND OBJECTS\*/

/\*NAME : SAGAR GIRI, ROLL NO. 205 , SECTION : A

\* DATE : 21-FEB-2014\*/

#include<iostream>

using namespace std;

class student

{

private:

char name[10];

int roll;

public:

void getdata()

{

cout<<"enter name: ";

cin>>name>>roll;

}

void display()

{

cout<<"\nyour name: " << name;

cout<<"\nYour roll: "<<roll;

}

};

int main()

{

student s1,s2;

s1.getdata();

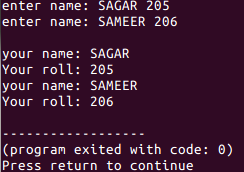
s2.getdata();

s1.display();

s2.display();

}

OUTPUT:



/\*THIS C++ PROGRAM ILLUSTRATES THE CONCEPT OF DEFINING A MEMBER

\* FUNCTION OUTSIDE A CLASS\*/

/\*NAME : SAGAR GIRI, ROLL NO. 205 , SECTION : A

\* DATE : 21-FEB-2014\*/

#include <iostream>

using namespace std;

class complex

{

private:

int real,imag;

public:

complex()

{

real = imag = 0;

}

complex(int r, int i)

{

real = r;

imag = i;

}

void display();

void addcomplex(complex, complex);

};

void complex::addcomplex(complex cc1,complex cc2)

{

real = cc1.real+cc2.real;

imag = cc1.imag+cc2.imag;

}

void complex::display()

{

cout <<real<<"+"<<imag<<"i"<<endl;

}

int main()

{

complex c1(5,7),c2(4,2),c3;

c1.display();

c2.display();

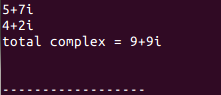
c3.addcomplex(c1,c2);

cout<<"total complex = ";

c3.display();

}

OUTPUT:



/\*THIS C++ PROGRAM ILLUSTRATES THE CONCEPT OF OBJECTS AS

\* FUNCTION ARGUMENTS USING CONSTRUCTORS\*/

/\*NAME : SAGAR GIRI, ROLL NO. 205 , SECTION : A

\* DATE : 21-FEB-2014\*/

#include <iostream>

using namespace std;

class Distance

{

private:

int feet;

float inches;

public:

Distance() //DEFAULT CONSTRUCTOR

{

feet =0;

inches =0.0;

}

Distance(int ft, float in)//TWO ARGUMENT CONSTRUCTOR

{

feet = ft;

inches = in;

}

void adddistance(Distance dd1,Distance dd2)

{

feet = dd1.feet+ dd2.feet;

inches = dd1.inches+dd2.inches;

if(inches >= 12.0)

{

feet++;

inches -= 12.0;

}

}

void display()

{

cout<<feet<<"\'-"<<inches<<"\""<<endl;

}

};

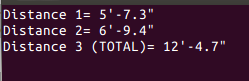
int main()

{

Distance d1(5,7.3),d2(6,9.4),d3;

cout<<"Distance 1= ";

d1.display(); //OUTPUT

cout<<"Distance 2= "; 

d2.display();

d3.adddistance(d1,d2);

cout<<"Distance 3 (TOTAL)= ";

d3.display();

}

/\*THIS C++ PROGRAM ILLUSTRATES THE CONCEPT OF

\* RETURNING OBJECTS FROM THE MEMBER FUNCTION\*/

/\*NAME : SAGAR GIRI, ROLL NO. 205 , SECTION : A

\* DATE : 21-FEB-201\*/

#include <iostream>

using namespace std;

class Distance

{

private:

int feet;

float inches;

public:

Distance()

{

feet = 0;

inches = 0.0;

}

Distance(int f,float i)

{

feet = f;

inches = i;

}

void display()

{

cout<<feet<<"\'-"<<inches<<"\""<<endl;

}

Distance adddistance(Distance);

};

Distance Distance::adddistance(Distance dd2)

{

Distance d;

d.feet = feet+dd2.feet;

d.inches = inches + dd2.inches;

if(d.inches >= 12.0)

{

d.feet++;

d.inches -= 12.0;

}

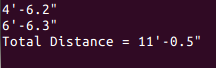
return d; //RETURNING THE OBJECT D

}

int main()

{

Distance d1(4,6.2),d2(6,6.3),d3; //OUTPUT

d1.display(); 

d2.display();

d3 = d1.adddistance(d2);

cout<<"Total Distance = ";

d3.display();

}