**Shallow copy**

#include <iostream>

using namespace std;

// declare a class

class rectangle

{

private:

double length;

double width;

public:

rectangle(double len, double wdt) // parameterized constructor

{

length = len;

width = wdt;

}

rectangle(rectangle &r1) // declaring copy contructor

{

length=r1.length;

width=r1.width;

}

double calculateArea()

{

return length \* width;

}

};

int main()

{

rectangle r1(10.5, 8.6); // create an object of Wall class

cout << "\n\tArea of Rectangle 1 : " << r1.calculateArea() << endl; // print area of wall1

rectangle r2 = r1; // copy contents of room1 to another object room2

rectangle r3(r1); // calling copy constructor

cout << "\tArea of Rectangle 2 : " << r2.calculateArea() << endl; // print area of wall2

cout << "\tArea of Rectangle 3 : " << r3.calculateArea() << endl;

return 0;

}

**Shallow copy (with updated one object)**

#include <iostream>

using namespace std;

class shallow

{

int a,b; // by default private

public:

shallow(int x,int y)

{

a=x;

b=y;

}

void update()

{

a=a+2;

b=b+2;

}

void show()

{

cout<<"\tvalue of a "<<a<<"\n\tvalue of b "<<b<<endl;

}

};

int main ()

{

shallow B1 (3,4);

shallow B2=B1;

shallow B3(B1); // copy constructor called which is default in compiler

B1.show();

B2.show();

B3.show();

B1.update(); // updating B1 object

B1.show();

B2.show(); // prints the old value do not copy updated value of B1

B3.show();

}

**Deep copy contructor**

#include <iostream>

using namespace std;

class deep\_copy

{

private:

int \* x;

public:

// Constructor with single parameter

deep\_copy(int m)

{

x = new int; // first location

\*x = m;

}

// Introduce Copy Constructor and perform Deep Copy

deep\_copy( deep\_copy& ob1)

{

x = new int; // second location

\*x = \*(ob1.x);

}

void update()

{

\*x=\*x+2;

}

//Print Function

void PrintX()

{

cout << "Int X=" << \*x << endl;

}

~deep\_copy() //DeAllocate the heap (destructor)

{

delete x;

}

};

int main()

{

deep\_copy ob1(10);

deep\_copy ob2(ob1) ;

ob1.PrintX();

ob2.PrintX();

// Change the Data member value of Object 1

// And print both Object 1 and Object 2

ob1.update();

ob1.PrintX();

ob2.PrintX();

}

// Reason to use deep copy constructor

// 1. In case we use pointer

// 2. We want to change value of object 1 but do not want to copy in object 2