```
#include <iostream>

using namespace std;

/*
This example illustrates the behavior of the compiler with regards
to generated constructors. Remember that the compiler will always generate
a copy constructor, and will generate an empty constructor only if you do
not provide any constructor. However:
1- the default copy constructor will perform bit to bit copy, which is
   not useful if your class holds pointers
2- the empty constructor will not do anything
So, it is almost always a good idea for the programmer not to rely on
them, but to provide them explicitly
*/


/* A class with a programmer provided constructor: the compiler
   will not generate one */
class X {
private:
        int a;
public:
        X(int v) { a = v; }; // inline constructor. More on this later
        void print() { cout << a << endl; } // inline method
};

/* A class with no programmer provided constructor: the compiler
   will generate one */
class Y {
private:
        int a;
public:
        void print() { cout << a << endl; } // inline method
};

/* A class with two programmer provided constructors, one taking and
   integer and the other taking no parameters */
class Z {
private:
        int a;
public:
        Z() { a = 0; }
        Z(int v) { a = v; }
        void print() { cout << a << endl; } // inline method
};


int main(int argc,char **argv)
{
        X x1(1);
        X x2;  // wrong! no compiler generated constructor
        Y y1;
        Z z1;
        Z z2(2);
        x1.print();
        y1.print();
        z1.print();
        z2.print();

        return 0;
}
```