Type Conversion

- In C programming language, when different types of constants and variables are used in expression, C automatically perform type conversion based on some fixed rules. In assignment operation, variable at the right hand side is automatically converted to the type of the variable on the left. The Same can also be possible in C++ programming language.
- But this automated type promotion will work well if both data types are of primary data type or both are of same user-defined data type. But it will create problem when one data type is user-defined data type and another is primary data type. So for that we have to use some special function for type conversion as in such cases automatic type conversion can not be performed by the language itself.

There are three types of possible type conversion

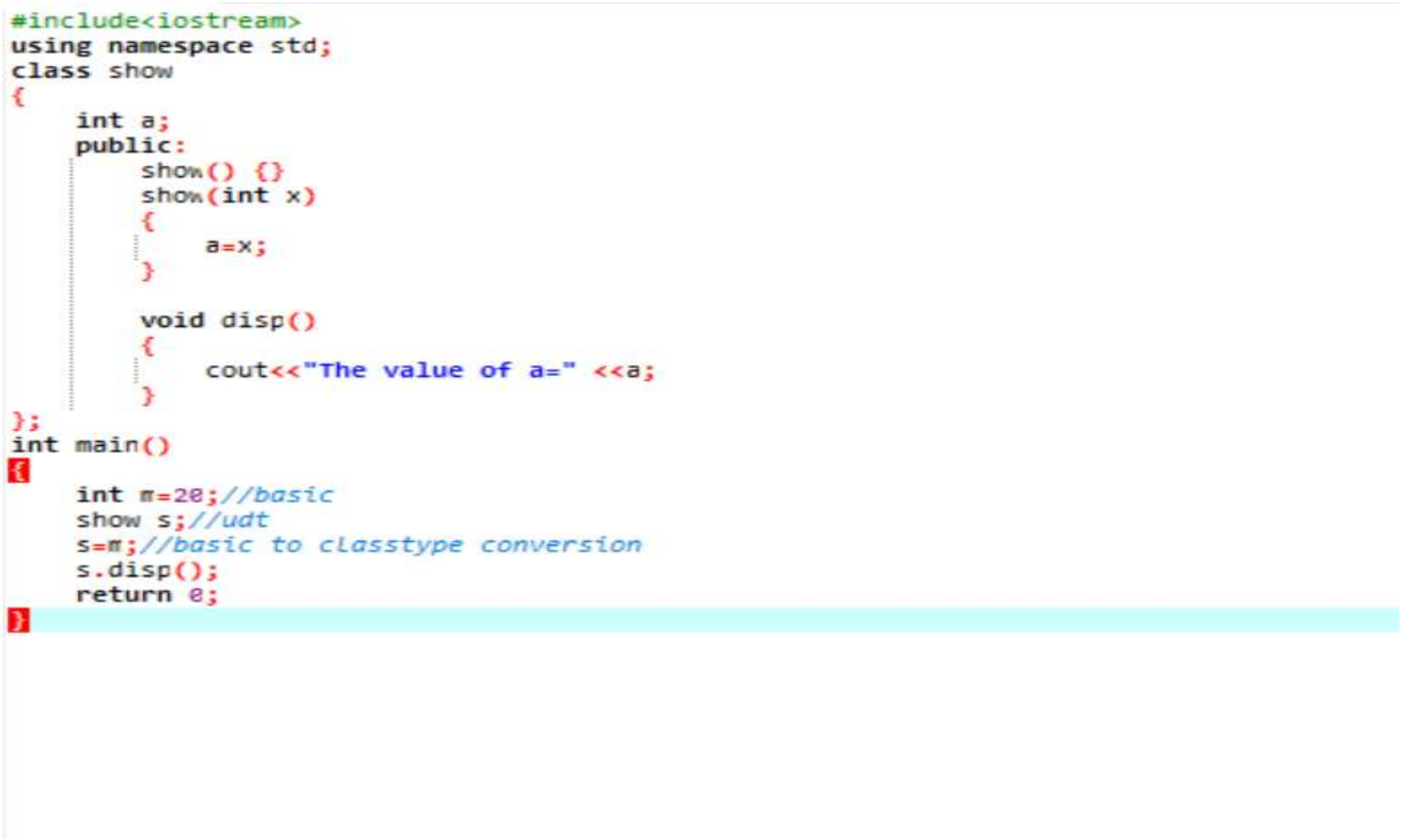
- 1. **Conversion from basic type to class type**
- 2. **Conversion from class type to basic type**
- 3. **Conversion from one class type to another class type**

Conversion from basic to class type

- In this type of conversion the source type is basic type and the destination type is class type. Means basic data type is converted into the class type.

Using Constructor

- We can use constructor to perform type conversion during the object creation.
- Consider the following example with class ‘Time’ in which we want to assign total time in minutes by integer variable *‘duration’.*



```
#include<iostream>
using namespace std;
class show
{
    int a;
    public:
        show() {}
        show(int x)
        {
            a=x;
        }

        void disp()
        {
            cout<<"The value of a=" <<a;
        }
};
int main()
{
    int m=20;//basic
    show s;//udt
    s=m;//basic to classtype conversion
    s.disp();
    return 0;
}
```

Conversion from class to basic type

- In this type of conversion the source type is class type and the destination type is basic type. Means class data type is converted into the basic type.
- For example we have class *Time* and one object of *Time* class *'t'* and suppose we want to assign the total time of object *'t'* to any integer variable say *'duration'* then the statement below is the example of the conversion from class to basic type.

```
duration= t ; // where, t is object and
duration is of basic data type
```

Here the assignment will be done by converting *“t”* object which is of class type into the basic or primary data type. It requires special casting operator function for class type to basic type conversion. This is known as the *conversion function*. The syntax for the conversion function is as under:

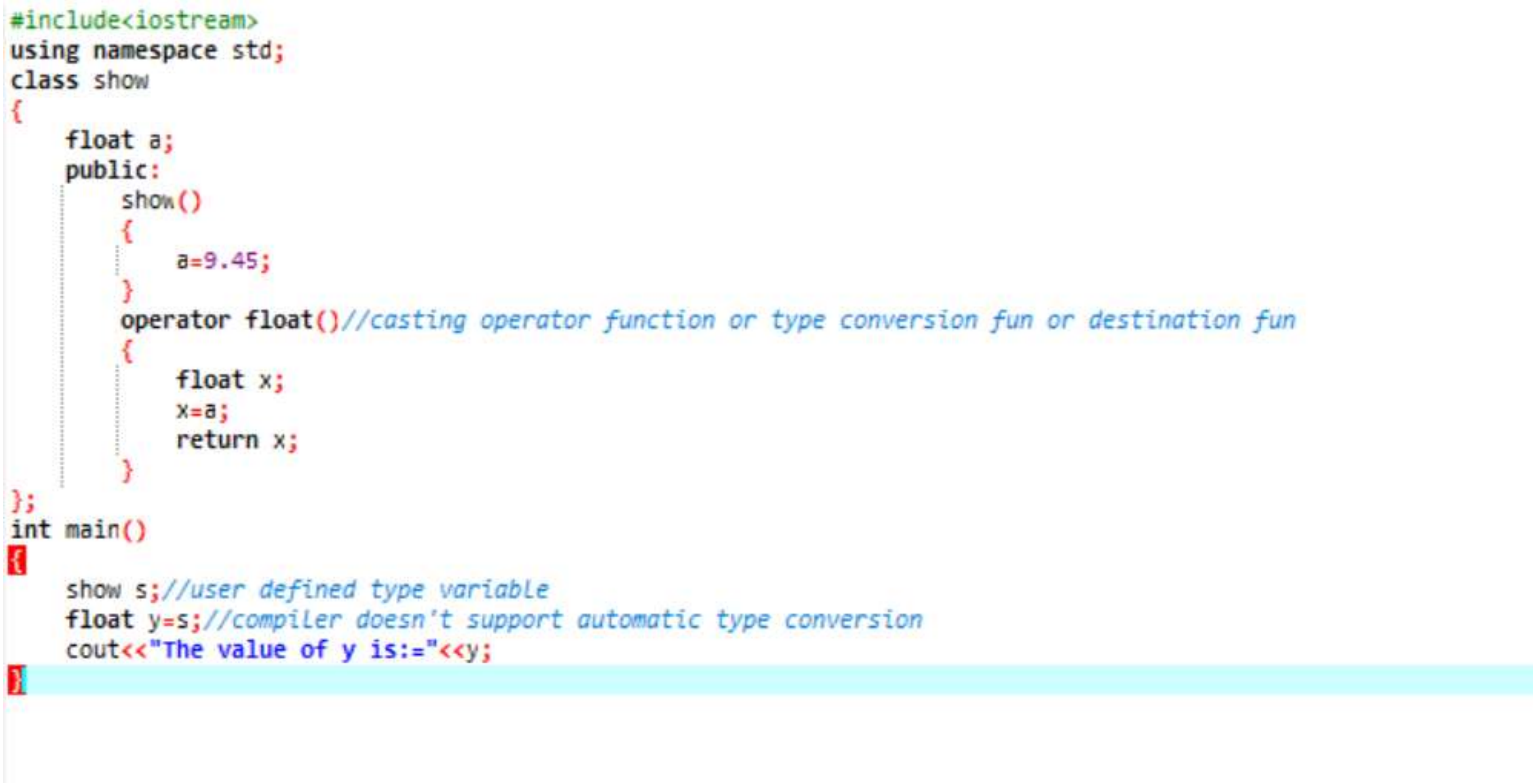
```
operator typename( )
{
     ….
     ….
     ….
}
```

- The conversion function should satisfy the following condition:

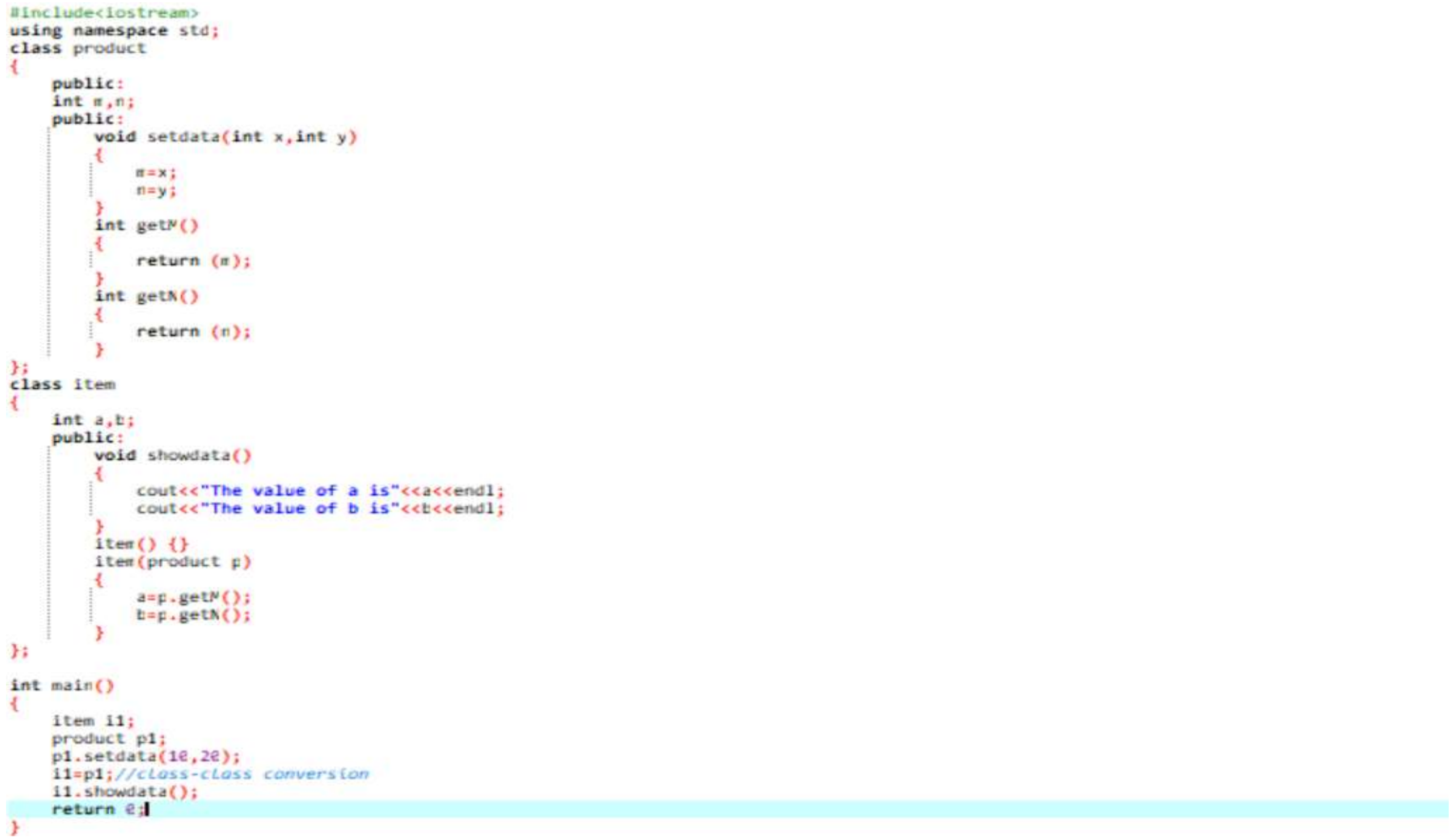
1.It must be a class member.

2.It must not specify the return value even though it returns the value.

3.It must not have any argument.


```
#include<iostream>
using namespace std;
class show
{
    float a;
    public:
        show()
        {
            a=9.45;
        }
        operator float()//casting operator function or type conversion fun or destination fun
        {
            float x;
            x=a;
            return x;
        }
};
int main()
{
    show s;//user defined type variable
    float y=s;//compiler doesn't support automatic type conversion
    cout<<"The value of y is:="<<y;
}
```

Conversion from Class to Class type



```
#include<iostream>
using namespace std;
class product
{
    public:
    int m,n;
    public:
        void setdata(int x,int y)
        {
            m=x;
            n=y;
        }
        int getM()
        {
            return (m);
        }
        int getN()
        {
            return (n);
        }
};
class item
{
    int a,b;
    public:
        void showdata()
        {
            cout<<"The value of a is"<<a<<endl;
            cout<<"The value of b is"<<b<<endl;
        }
        item() {}
        item(product p)
        {
            a=p.getM();
            b=p.getN();
        }
};

int main()
{
    item i1;
    product p1;
    p1.setdata(10,20);
    i1=p1;//class-class conversion
    i1.showdata();
    return 0;
}
```