Object Oriented Programming

Shaobai Kan

chapter 10

Outline

Outline

- using the *this* pointer

Outline

- using the *this* pointer
- *static* class members

using the this *pointer*

This pointer

Question. An object's member functions can manipulate the object's data. How do member functions know which object's data members to manipulate?

This pointer

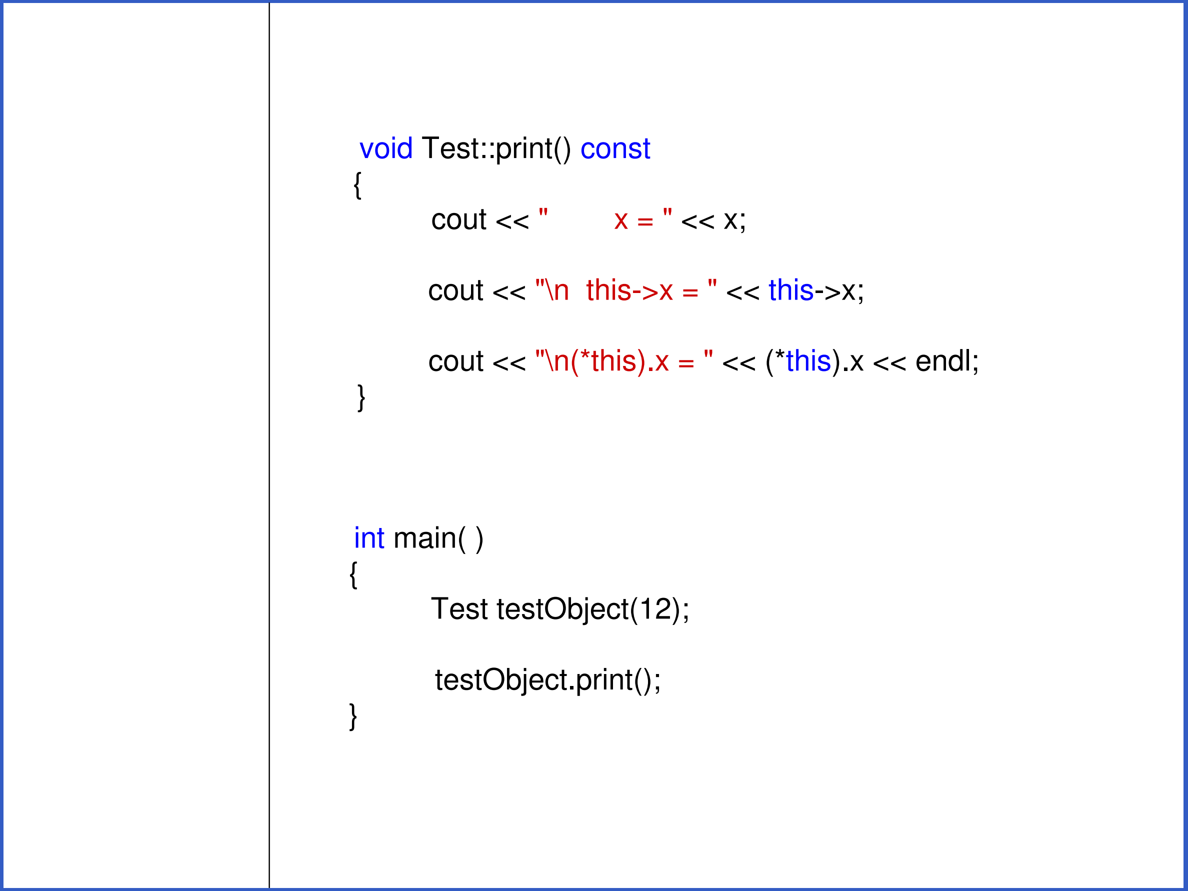
Question. An object’s member functions can manipulate the object’s data. How do member functions know which object’s data members to manipulate?

Answer. Every object has access to its own address through a pointer called *this* (a C++ keyword).

This pointer

```
# include <iostream>
using namespace std;


class Test
{
public:
      Test (int = 0);
      void print () const;

private:
      int x;
};


Test::Test( int value )
              : x(value)
{

}
```

This pointer



```
void Test::print() const
{
        cout << "        x = " << x;

        cout << "\n  this->x = " << this->x;

        cout << "\n(*this).x = " << (*this).x << endl;
}



int main( )
{
        Test testObject(12);

        testObject.print();
}
```

static **class members**

Static members

Fact. Generally, each object of a class has its own copy of all the data members of the class. However, in certain cases, only one copy of a variable should be shared by all objects of a class. A static data member is used for these and other reasons.

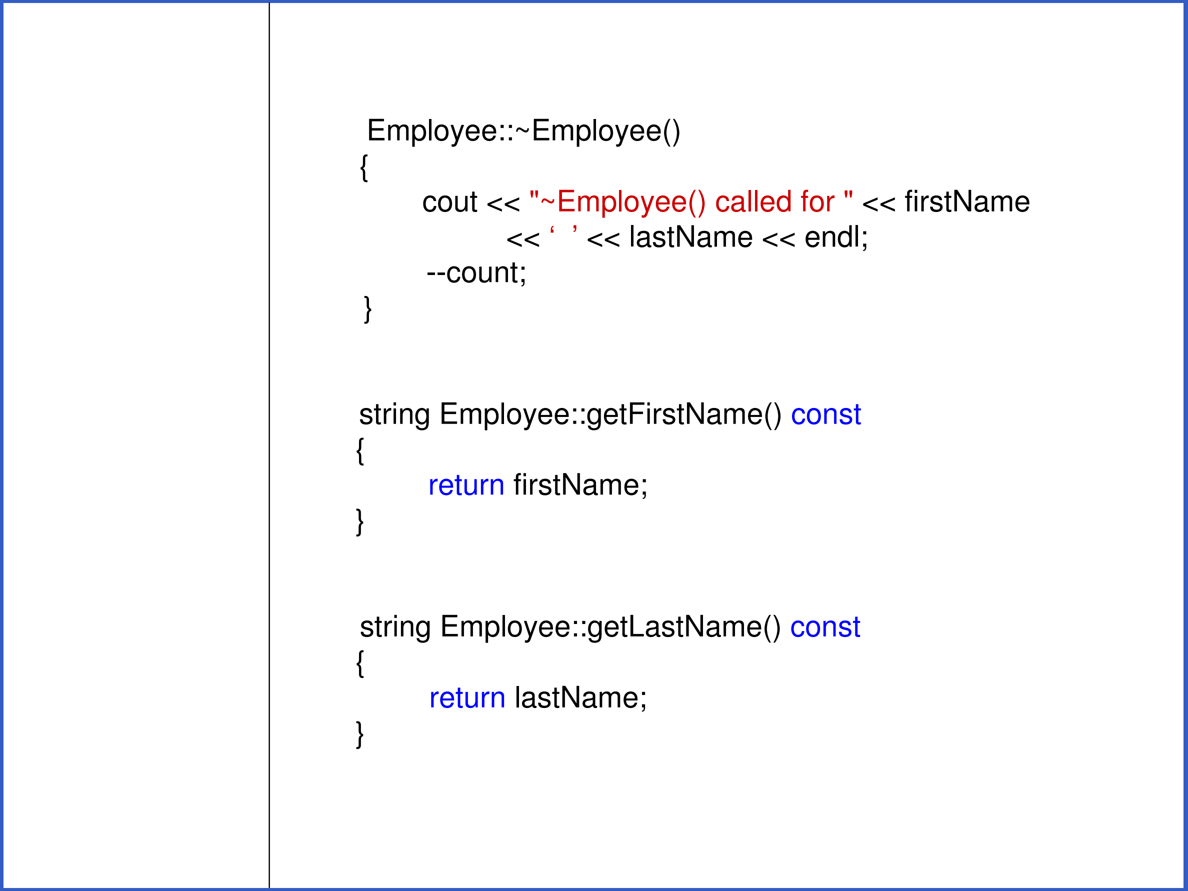
static members

```
#include <string>
using namespace std;

class Employee
{
 public:
      Employee(const string &, const string &);
      ~Employee();
      string getFirstName() const;
      string getLastName() const;

      static int getCount();

 private:
      string firstName;
      string lastName;


      static int count;
};
```

static members

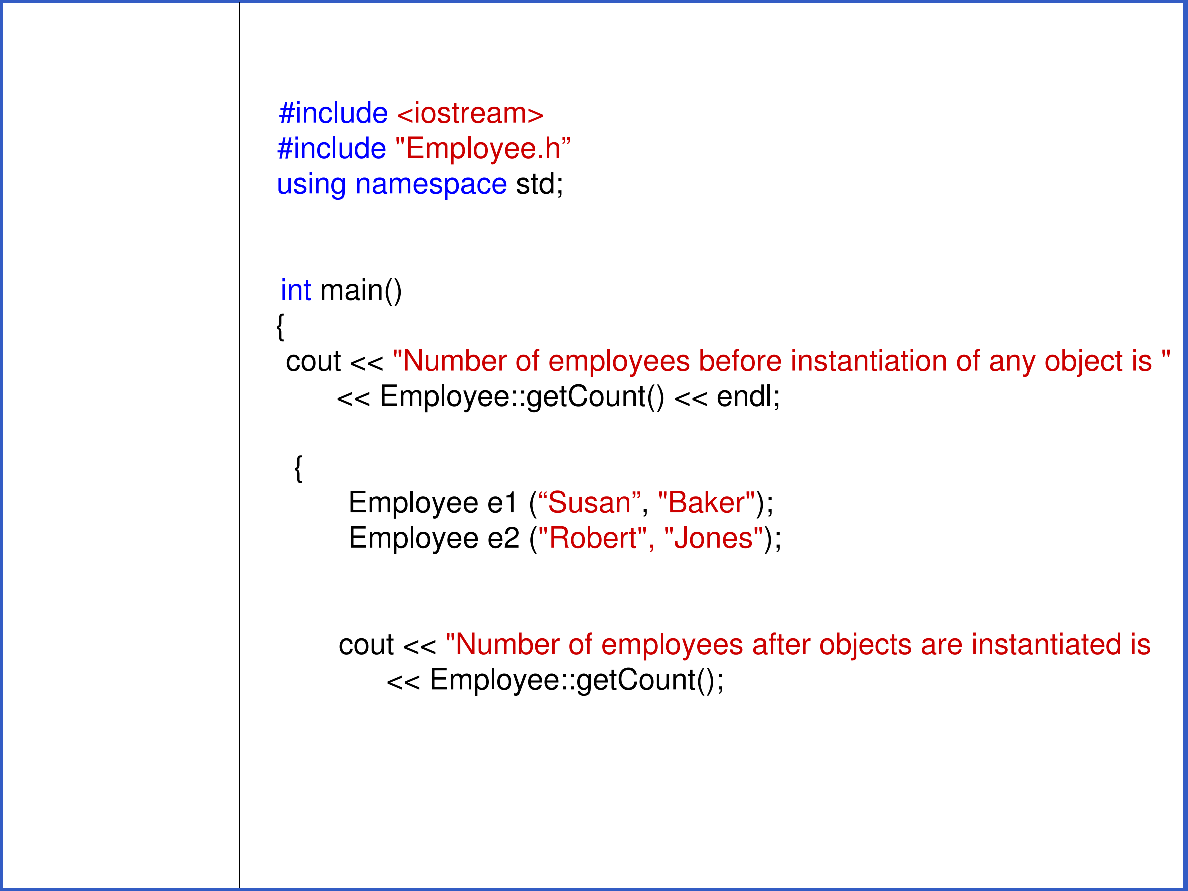
```
#include <iostream>
#include "Employee.h"
using namespace std;

int Employee::count = 0;      //cannot include keyword static

int Employee::getCount()
{
        return count;
}


Employee::Employee(const string &first, const string &last)
        : firstName(first), lastName(last)
{
        ++count;
        cout << "Employee constructor for " << firstName
          << ' ' << lastName << " called." << endl;
}
```

static members



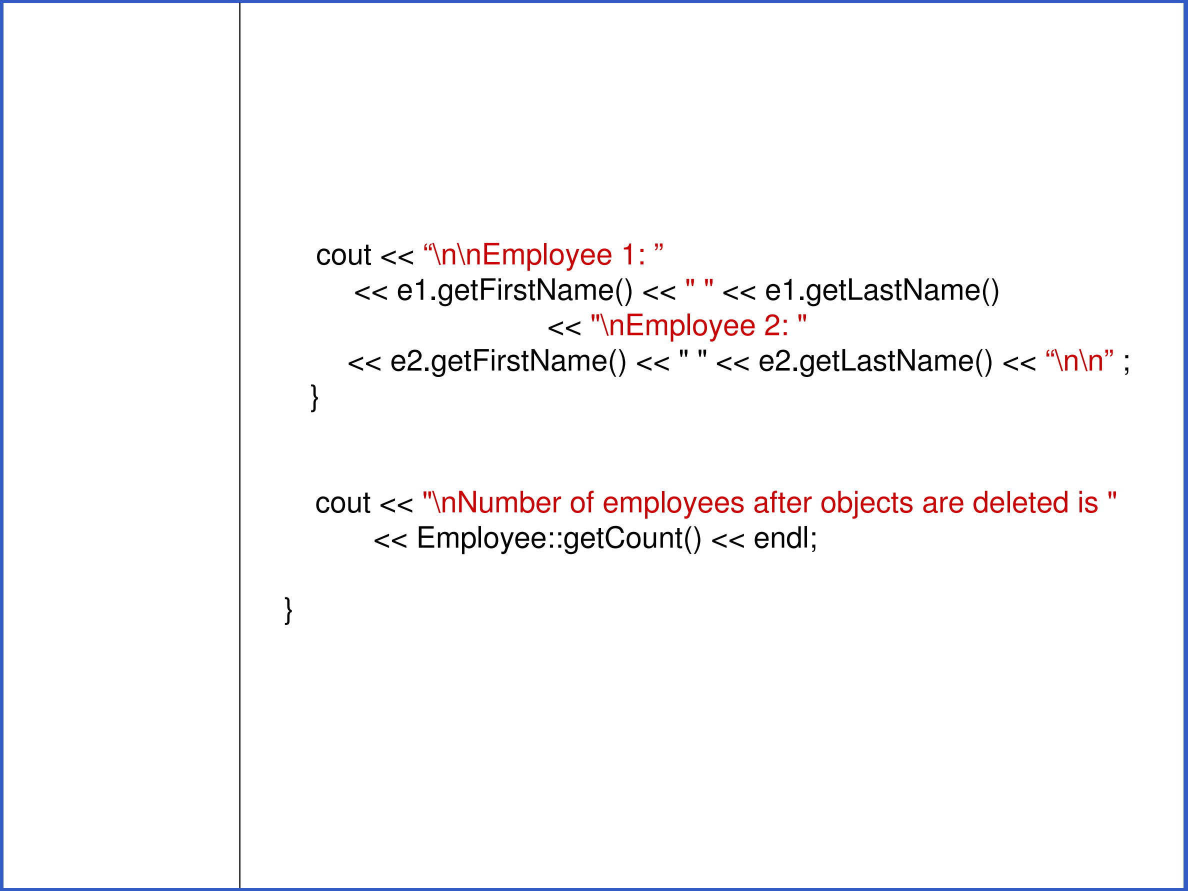
```
Employee::~Employee()
{
        cout << "~Employee() called for " << firstName
                << ' ' << lastName << endl;
        --count;
}


string Employee::getFirstName() const
{
        return firstName;
}


string Employee::getLastName() const
{
        return lastName;
}
```

static members



```
#include <iostream>
#include "Employee.h”
using namespace std;


int main()
{
 cout << "Number of employees before instantiation of any object is "
      << Employee::getCount() << endl;

  {
       Employee e1 (“Susan”, "Baker");
       Employee e2 ("Robert", "Jones");


      cout << "Number of employees after objects are instantiated is
           << Employee::getCount();
```

static members



```
    cout << "\n\nEmployee 1: "
        << e1.getFirstName() << " " << e1.getLastName()
                              << "\nEmployee 2: "
        << e2.getFirstName() << " " << e2.getLastName() << "\n\n" ;
   }


   cout << "\nNumber of employees after objects are deleted is "
         << Employee::getCount() << endl;

}
```

Homework:

- Read Sec. 10.2, 10.4, 10.5, 10.6.