Supplement IV.E: Constructor Initializers

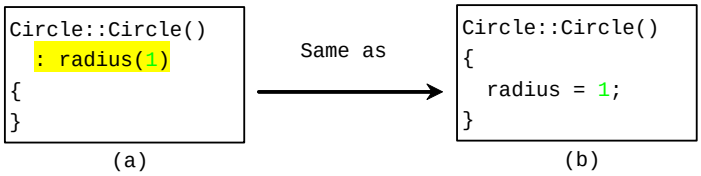
For Introduction to C++ Programming
By Y. Daniel Liang

Data fields may be initialized in the constructor using an initializer list in the following syntax:

```
ClassName(parameterList)
  : datafield1(value1), datafield2(value2) // Initializer list
{
  // Additional statements if needed
}
```

The initializer list initializes **datafield1** with **value1** and **datafield2** with **value2**.

For example,



(b) is actually more intuitive than (a) without using an initializer list. However, using an initializer list is necessary to initialize object data fields that don't have a no-arg constructor.

NOTE
In C++, you can declare an object data field. For example, **name** is declared a **string** object in the following code:

```
class Student
{
public:
  Student();

private:
  string name;
};
```

However, declaring an object data field in a class is different from declaring a local object in a function like this:

```
int main()
{
```

```
  string name;
};
```

As an object data field, the object is not created when it is declared. As an object declared in a function, the object is created when it is declared.

NOTE
In C++, data fields (primitive or object type) cannot be declared with an initial value. For example, the following code is wrong:

```
class Student
{
public:
  Student();

private:
  int age = 5; // Cannot initialize a class member
  string name("Peter"); // Cannot initialize a class member
};
```

The correct declaration is

```
class Student
{
public:
  Student();

private:
  int age; // Declare a data field of the int type
  string name; // Declare a data field of the string type
};
```

When an object of the **Student** class is created, the constructor automatically invokes the **string** class's no-arg constructor to create an object for **name**. However, the primitive data field **age** is not automatically initialized. You have to explicitly initialize it in the constructor. For example,

```
class Student
{
public:
  Student()
  {
    age = 5; // Initialize age
  };
```

```
private:
  int age; // Declare a data field of the int type
  string name; // Declare a data field of the string type
};
```

If a data field is of an object type, the no-arg constructor for the object type is automatically invoked to construct an object for the data field. If the no-arg constructor does not exist, a compilation error will occur. For example, the code in Listing 1 has an error, because the **time** data field (line 24) in the **Action** class is of the **Time** class that does not have a no-arg constructor.

Listing 1 NoDefaultConstructor1.cpp

```
class Time
{
public:
  Time(int hour, int minute, int second)
  {
     // Code omitted
  }

private:
  int hour;
  int minute;
  int second;
};

class Action
{
public:
  Action(int hour, int minute, int second)
  {
    time = Time(hour, minute, second);
  }

private:
  Time time;
};
```

To fix this error, you have to use the constructor initializer list as shown in Listing 2. The data field **time** is initialized using an initializer list (line 19).

Listing 2 NoDefaultConstructor2.cpp

```
class Time
{
```

```
public:
  Time(int hour, int minute, int second)
  {
     // Code omitted
  }

private:
  int hour;
  int minute;
  int second;
};

class Action
{
public:
  Action(int hour, int minute, int second)
    :time(hour, minute, second)
  {
  }

private:
  Time* time;
};
```