The Constructors

Lecture 6
Sections 4.4, 4.5, 6.3

Robb T. Koether
Hampden-Sydney College
Wed, Jan 28, 2009
Outline

1. The Four Fundamental Member Functions
2. The Default Constructor
3. The Automatic Default Constructor
4. The Copy Constructor
5. The Automatic Copy Constructor
6. Constructors and the `new` Operator
7. Assignment
The Four Fundamental Member Functions

- The four fundamental functions
  - The default constructor
  - The copy constructor
  - The destructor
  - The assignment operator

- These four member functions are essential to the functioning of any class.
The Default Constructor

The default constructor constructs an object for which no initial value is given.

The default constructor should initialize the data members to neutral values that are appropriate for that type.

```
Type::Type(); // Prototype
Type Object;   // Usage
```
Purposes of the Default Constructor

The default constructor is used when:
- An object is created with no initial value specified.
- The `new` operator is used to create an array.
- A static array is partially initialized.

```cpp
Vectr v;
Vectr* pv = new Vectr[10];
Vectr av[10] = {Vectr(4, 123)};
```
Example

Example (The *Vectr* Class)

- Download *vectr.h*.
- Download *vectr.cpp*.
- Download and run *VectrTest.cpp*.
The automatic default constructor is provided automatically if we write no constructor.

- It
  - Allocates memory for the data members.
  - Invokes each data member’s own default constructor.

- It is *not* provided automatically if we write any constructor; we must then write the default constructor, if we want one.
The Copy Constructor

The copy constructor constructs an object which will be a copy of an existing object.

```cpp
Type::Type(const Type&);  // Prototype
Type ObjectA = ObjectB;    // Usage 1
Type ObjectA(ObjectB);    // Usage 2
```
Purposes of the Copy Constructor

- The copy constructor is used when
  - An object is created and initialized to the value of an existing object of the same type.
  - A local copy of a value parameter is created during a function call.
  - A function returns a value.
The Copy Constructor

```cpp
Vectr v(5, 123);
Vectr u = v;
Vectr w(u);
```
The Copy Constructor

```c
Vectr w;
f(w);
::
Vectr f(Vectr v)
{
    Vectr u;
    ::
    return u
}
```
Point of Style

The Copy Constructor

```
Vectr u = v;    // Good style
Vectr w;       // Poor style
w = u;
```

- The first uses the copy constructor.
- The second uses the default constructor followed by the assignment operator.
Points of Style

The Copy Constructor

```cpp
Vectr(const Vectr& v)
{
    makeCopy(v);
    return;
}

void makeCopy(const Vectr& v)
{
    // Make a copy
}
```

- A handy technique is to write a function `makeCopy()` and simply call on it to do the work of the copy constructor.
The automatic copy constructor

- Allocates memory for the data members.
- Invokes each data member’s copy constructor to copy values from the existing object.
This is called a **shallow copy**.

Pointers get copied with no change in value.

Therefore, the pointer in the new object will point to the very same memory as the pointer in the old object.

Generally, this is not good. Instead, we want a **deep copy**.

What would happen in the `Vectr` class if we made a shallow copy of a vector?
Constructors and the `new` Operator

The `new` Operator and Constructors

```cpp
int* pi = new int(123);
int* pai = new int[123];
string* ps = new string("Hello");
Rational* pr = new Rational(2, 3);
Point2D* ppt = new Point;
Vectr* pv = new Vectr(v);
Vectr* pv2 = new Vectr(10);
Vectr* pv3 = new Vectr[10];
```

- The `new` operator works in conjunction with the constructors.
Assignment

Homework

- Read Section 4.4, pages 158 - 165.
- Read Section 4.5, pages 165 - 169.
- Read Section 6.3, pages 269 - 286.