C++ Classes: Conversion

Logistics

• Job related news
  – Co-op orientation
    • Friday, Sept 16th 12:00 – 1:30pm 70-1400
    • Friday, Oct 14th 1:00 – 2:30pm Eastman Aud.
  – Career Fair
    • Wednesday, Sept 28th
    • 11am – 4pm
    • Gordon Field House

Project

• Reminder
  – Part 1 (clock and design) due Sunday, Sept 25th
  – Start thinking about partners for Parts 2-3
• Any questions?

Plan

• Today
  – Data type conversions
    • conversion constructors
    • Assignment operators
• Tomorrow
  – Overloading methods and operators
• Thursday
  – Intro to templates

Reminder

• Exam 1
  – Next Thursday, Sept 29th
  – More details to come

Data type conversion

• C++ will perform datatype conversions when at all possible.

  short int a = 12;
  long b = a;  // conversion from short to long
Data type conversion

- Conversion occurs:
  - Assignments
  - Function argument passing
  - Return values
  - Initializers
  - Expression

- C++ can do so with classes as well if conversion constructors are defined.

Constructor

- Special member function used to create an object of a class.
  - Has same name as the class.
  - Does not return anything.
  - Used to initialize data members.

Box.h

```cpp
#ifndef BOX_DEFINED
#define BOX_DEFINED

class Box
{
private:
  int height, width, depth;
public:
  Box (int h, int w, int d);
  int volume ()
};
#endif
```

Box.cpp

```cpp
#include "Box.h"

Box::Box (int h, int w, int d)
{
  height=h; width=w; depth=d;
}

int Box::volume ()
{
  return height * width * height;
}
```

Object initialization

- Objects are initialized using constructors made available by the class.

```cpp
Box mybox (3,4,5);
Box *boxOnHeap = new Box(3,4,5);
```

Conversion constructors

- Converts an object of one type to that of another.
- Will get called when the need for an automatic conversion arises.
Conversion constructor

```cpp
class Cube {
    private:
        int size;
    public:
        Cube (int s);
        int getSize();
};

class Box {
    private:
        int height, width, depth;
    public:
        Box (int h, int w, int d);
        Box (const Cube &C);
};
```

Member conversion constructors

A class can also provide a means for it to be converted to another class or intrinsic datatype.

```cpp
class Date {
    private:
        int month, day, year;
    public:
        Date (int m, int d, int y);
        operator long();  // convert this date to a long
};
```

```cpp
class Box {
    private:
        int height, width, depth;
    public:
        Box ();
        Box (int h, int w, int d);
};
```

```cpp
class Cube {
    private:
        int size;
    public:
        Cube (int s);
        operator Box();
};
```

```cpp
Member conversion function

Box::Box (const Cube &C) : height (C.getSize()),
                        width (C.getSize()), depth (C.getSize())
{};
Cube C(6);
Box convC = C;  // conversion constructor called.
```

Member conversion function

```cpp
Box::Box (int h, int w, int d);
Box::Box (const Cube &C);
```

```cpp
Cube C(6);
Box convC = C;  // conversion constructor called.
```

```cpp
Member conversion function

Date::operator long() {
    static int dys[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    long days = yr - 1900;
    days += 365;
    days += yr/4;
    for (int i=0; i < mo-1; i++) days += dys[i]
    days += da;
    return days;
}
Date today (9, 21, 2005);
l long since = today;
```

Member conversion function

```cpp
Cube::operator Box() {
    return Box (size, size, size);
}
Cube C(6);
Box convC;
convC = C;  // conversion function called.
```
Ways to invoke conversion

• Implicit
  – Assignment, function args, etc.
• Explicit
  – Cast
  – Explicit call to conversion constructor/member conversion function

```cpp
class Cube {
private:
  int size;
public:
  Cube (int s);
  int getSize();
};
class Box {
private:
  int height, width, depth;
public:
  Box();
  Box (int h, int w, int d);
  Box (const Cube &C);
};

Cube C(6);
Box convC;
convC = C;  // implicit conversion.
convC = (Box)C;  // explicit via cast
convC = Box (C); // explicit via constructor call.
```
Conversions

- Questions?

Assignment operator

- operator=
  - Called when an assignment is made
  - Copies all relevant data from object assigner to assignee.
  - Should check for self-assignment!

Assignment operator

```cpp
class Complex
{
    private:
        double re, im;
    public:
        Complex();
        Complex & operator= (Complex &c);
        ...
    }

    Complex c1, c2;
    c2 = c1;  // is the same as saying c2.operator= (c1);
```

Assignment operator

- Note that the assignment operator returns a reference to itself
  - This is to allow statements like:

    ```cpp
    Complex c1, c2, c3;
    c3 = c2 = c1;
    ```

Assignment operator

- If no assignment operator is defined for a class, the default assignment operator is used.
  - Member by member copy of data from one object to another.
  - Can be troublesome if class have pointers as data members.
Assignment operator

```cpp
class Foo
{
private:
    int *array_member;
    int asize;
    ...
};

Foo c1, c2;
c1 = c2;
delete c1;
delete c2;
```

### Constructor

- **Copy vs. Assignment**
  
  ```cpp
  Date::Date (int day, int month, int year) {
    d = day; // constructor + assignment performed
    m = month;
    y = year;
  }
  
  Can also be written using `subobject constructor` (this way is more efficient).
  ```
  
  ```cpp
  Date::Date (int day, int month, int year) :
    d (day), m (month), y (year) {}
  // just copy constructor is called.
  ```

- **Copy Constructor**
  
  - If no copy constructor is defined for a class, the default copy constructor is used.
  - Member by member copy of data from one object to another.
  - Can be troublesome if a class have pointers as data members.
  - Same issues as with the default assignment operator!!!!

### Constructor Summary

- Date d1(3, 10, 2002); // constructor called
- Date d2, d5; // default constructor called
- Date d3 (d2); // copy constructor called
- Date d4 = d1; // copy constructor called
- d5 = d2; // assignment operator called.

Questions?
Assignment operator

Questions?

Summary

• Conversion
  – Conversion constructors
  – Conversion member functions
• Assignment operators
  
• Questions?