CS3157: Advanced Programming

Lecture #15
Apr 23
Shlomo Hershkop
shlomo@cs.columbia.edu
Outline

- C++
  - class abstractions
  - class inheritance
  - Example code

- Reading:
  - chapters 16, 17, 18 (C++, classes, overloading)
Overview

- 3 more classes (not C++ classes)
- Will meet here on Wednesday
- Before the end of the semester, want to show off C++ and Ruby/Rails
Help with lab…

- Any specific questions?
get

- one character at a time
- even whitespaces
- EOF

- different version of get
  - 21_12
  - 21_13
getline

- read till delim
- throws it out
- replaces it with \0

- 21_14
low level

- low level read/write

- 21_15
Random question

- What is the difference between constructor and destructor

- Can you trigger a destructor/how?
  - Non pointer
  - Pointer
- Mentioned last time..

- Code from fig18_03 (c book)
Phone Numbers

- Say you want to code a phone number class
class PhoneNumber {

private:
    char areaCode[ 4 ];    // 3-digit area code and null
    char exchange[ 4 ];   // 3-digit exchange and null
    char line[ 5 ];       // 4-digit line and null

};

Any observations on phone number representation ??
class PhoneNumber {
    friend ostream &operator<<( ostream&, const PhoneNumber & );
    friend istream &operator>>( istream&, PhoneNumber & );

private:
    char areaCode[ 4 ]; // 3-digit area code and null
    char exchange[ 4 ]; // 3-digit exchange and null
    char line[ 5 ];    // 4-digit line and null
};
// Overloaded stream-insertion operator (cannot be
// a member function if we would like to invoke it with
// cout << somePhoneNumber;).
ostream &operator<<( ostream &output, const PhoneNumber
 &num )
{
    output << "\"" << num.areaCode << "\"\" << num.exchange << "-" << num.line;
    return output;       // enables cout << a << b << c;
}
istream &operator>>( istream &input, PhoneNumber &num )
{
    input.ignore();                     // skip (  
    input >> setw( 4 ) >> num.areaCode; // input area code
    input.ignore( 2 );                  // skip ) and space
    input >> setw( 4 ) >> num.exchange; // input exchange
    input.ignore();                     // skip dash (-)
    input >> setw( 5 ) >> num.line;     // input line
    return input;      // enables cin >> a >> b >> c;
}
int main()
{
    PhoneNumber phone;  // create object phone

    cout << "Enter phone number in the form (123) 456-7890: \n";

    cin >> phone;

    cout << "The phone number entered was: " << phone << endl;
    return 0;
}
unary

- $Y += Z$
- $Y\text{.operator}+= ( Z )$

- $++D$
- member
  - $D\text{.operator}++()$
- Non member
  - operator++($D$)
Instantiation

- Remember as soon as you declare a class you are instantiating it
- String s;
- Sometimes would like to have classes which can not be instantiated
- Abstract classes
  - A class is made abstract by having a pure virtual function.
virtual functions

- in C++ virtual functions allow you to define a specific function in the base class, which is undefined, and each of the subclasses need to override (implement a definition)

- `virtual char * md5sum();`
so if we use a base class pointer at a derived class object, calling \texttt{md5sum} will call the correct one

- compile time resolution
  - static binding
Abstract

- virtual char * md5sum() =0;

- any ideas on what error will be thrown if you instantiate it?
non virtual base functions

if you have a parent class A.foo()

derived class B defines B.foo()

A *a_ptr = B_object

a_ptr.foo()

- which foo will be triggered?
- why?
abstract classes II

- remember that making a pointer doesn’t instantiate anything
- can create pointers of type abstract classes
- used to enable polymorphic behavior

Example: Operating system device
  - read/write behavior
destructors

- when creating and manipulating objects in a polymorphic context, destructors will only be called on base class
solution

- define a virtual base class destructor
- will correct destructor will be called
In C++ a class has a default constructor only if you don’t define any

class Example1 {
    public:
        int a, b, c;
        void multiply (int n, int m)
        {
            a = n; b = m; c = a * b;
        }
};
Example

- So can say
  
  ```
  Example1 ex;
  ```

- Which will create a constructor for you

- As soon as you define any constructor (say taking one arg) the above line stops working

- Blame the compiler
By default

Compiler will create

- copy constructor
- copy assignment operator
- default destructor.
Differences

- Should understand the difference between
  - X
  - *X
  - &X
  - X.Y
  - X→Y
  - (*X).Y
friendship

- Functions (which you’ve seen) and classes which can access private class members
- So you’ve friend functions for overloading’
- Here is an example of friend class
// friend class
#include <iostream>
using namespace std;

class CSquare;

class CRectangle {
   int width, height;
public:
   int area ()
   {return (width * height);}
   void convert (CSquare a);
};

class CSquare {
private:
   int side;
public:
   void set_side (int a)
   {side=a;}
   friend class CRectangle;
};

void CRectangle::convert (CSquare a) {
   width = a.side;
   height = a.side;
}
int main () {
    CSquare sqr;
    CRectangle rect;
    sqr.set_side(4);
    rect.convert(sqr);
    cout << rect.area();
    return 0;
}

C++ hierarchy

- **composition:**
  - creating objects with other objects as members

- **derivation:**
  - defining classes by expanding other classes
  - like “extends” in java

```cpp
class SortedIntArray : public IntArray {
public:
    void sort();
private:
    int *sortBuf;
}; // end of class SortedIntArray
```

- “base class” (IntArray) and “derived class” (SortIntArray)
- derived class can only access public or protected members of base class
public derivation means that users of the derived class can access the public portions of the base class

private derivation means that all of the base class is inaccessible to anything outside the derived class

private is the default