Lecture 19

CIS 208

Wednesday, April 06, 2005
Welcome to C++

- Basic program style and I/O
- Class Creation
- Templates.
C++ OOP

- Allows for more complex programs
  - C starts to break down

- Better data protection

- Much more programmer control
some C++ features

- Exception Handling
- an actual string type and bool type
- Function Overloading/Constructors
- Operator overloading.
  - redefine what the + operator can do
OOP basics

- Encapsulation
- Polymorphism
- Inheritance
Encapsulation

- Binding Data and functions together in nice, neat packages.
- Private and public data and functions
- Implemented as Classes.
Polymorphism

- ‘One interface, multiple methods’
- Multiple functions with the same name.
- Don’t care how something works on the interior, just that it does something
Polymorphism Example

- In c, abs(), fabs(), labs(): Absolute value functions.
- Each takes different type of parameter.
- Annoying to remember which one does which.
Function Overloading

- In C++, define a single function name that can handle multiple types of data.

- In C++, there is just a `abs()` function.
Inheritance

- An object may acquire properties of another.

- Example. Red Delicious apple is a type of apple, which is a type of fruit, which is a type of food.
#include <iostream>
using namespace std;

int main() {
    cout << "this is output. \n";
    cout << "Enter a number: ";
    int i;
    cin >> i;
    cout << i << " squared is " << i*i << "\n";
    for (int j = 0; j < i; j++)
    {
        cout << "iteration " << j << "\n";
    }
    return 0;
}
Compiling

*best to use g++*

*Same commands as gcc. (-o for name of executable)*
#include <iostream>
using namespace std;

```cpp
iostream == stdio.
```

Don’t need .h if using namespace
C libraries

- Can use old C headers (math.h, string.h)
- math.h == cmath
- string.h == cstring
namespace

\*\*\*ignore for right now.\*\*\*

#include <iostream.h> ==

#include <iostream>
using namespace std;
function declaration

neither same as C

```c
int main() {...

Don’t need to put void as a parameter, still need void if no return.

Must still have function```
Local Variables

- Can be declared anywhere.

- Normal scope rules. Only know in current code segment and only after declaration.

- Declaring at beginning might make better code though
Basic output

cout << "This is output\n";

prints ‘This is output’ to console.

May still use any of the old functions

- printf, puts, putch, etc
cout <<

<< is the output symbol

cout is the destination (console)

Arrows point towards destination

cout << "hello" << " world";
cout <<

- works on any basic data type.
  - No format specifiers.

- Can chain multiple types together

```cpp
int j; char k; float m;

cout << j << k << m;
```
cout <<

use casting to alter appearance

float j = 1.5;
cout << (int) j;
cout <<

- can use expressions or function calls
int j = -2;

cout << j * j << " " << abs(j);
Basic input

```cpp
int j;
cin >> j;
```

Reads an int from the keyboard and stores it in variable `j`.

No & necessary. `cin >>` knows what to do.
cin >>

- again, arrows point to destination.
- Will still want to flush buffer.
- Works on all basic types, and strings
cin >>

- multiple reads.
char x, y, z;
cin >> x >> y >> z;
- takes first 3 chars and stores in x,y,z;
- blank spaces don’t count
 cin >>

  use multiple types
  int x; char y; float z;

 cin >> x >> y >> z;
 be careful, this can get tricky.