And Even More and More C++

Outline

- C++ Classes
  - Friendship
  - Inheritance
  - Multiple Inheritance
  - Polymorphism
  - Virtual Members
  - Abstract Base Classes
- File Input/Output

Friendship

- Friend functions
  - A non-member function in a class marked as “friend” makes it so that other instantiated objects of the **same** type can access each other’s information

Friend Function Example

```
// friend functions
#include <iostream>
using namespace std;

class Rectangle {
    int width, height;
  public:
    Rectangle() {}
    Rectangle (int x, int y) : width(x), height(y) {}
    int area() {return width * height;}
    friend Rectangle duplicate (const Rectangle&);
};

Rectangle duplicate (const Rectangle& param)
{
  Rectangle res;
  res.width = param.width*2;
  res.height = param.height*2;
  return res;
}

int main () {
  Rectangle foo;
  Rectangle bar (2,3);
  foo = duplicate (bar);
  cout << foo.area() << '\n';
  return 0;
}
```

```
24
```

More Friendship

- Friend Classes
  - A friend of a class can access protected and private items within that class

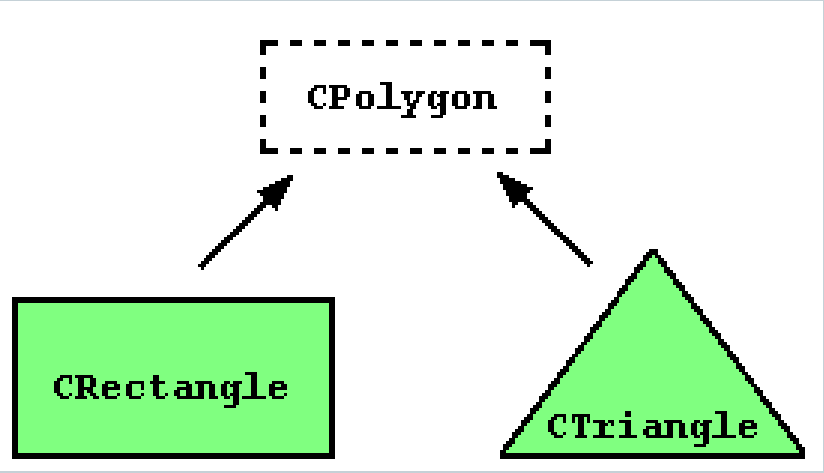
Friend Class Example

```
// friend class
#include <iostream>
using namespace std;

class Square;

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

class Square {
  friend class Rectangle;
  private:
    int side;
  public:
    Square (int a) : side(a) {}
};

void Rectangle::convert (Square a) {
  width = a.side;
  height = a.side;
}

int main () {
  Rectangle rect;
  Square sqr (4);
  rect.convert(sqr);
  cout << rect.area();
  return 0;
}
```

```
16
```

Inheritance

- Base class is the parent class
- Derived classes are the children
  - Children inherit the members of its parent
  - Children can also add their own members




```
class derived_class_name: public base_class_name
{ /*...*/ };
```

Inheritance Example

```
// derived classes
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b;}
 };

class Rectangle: public Polygon {
  public:
    int area ()
      { return width * height; }
 };

class Triangle: public Polygon {
  public:
    int area ()
      { return width * height / 2; }
  };

int main () {
  Rectangle rect;
  Triangle trgl;
  rect.set_values (4,5);
  trgl.set_values (4,5);
  cout << rect.area() << '\n';
  cout << trgl.area() << '\n';
  return 0;
}
```

```
20
10
```

Access Permissions

- External access permission to class data

Access	public	protected	private
members of the same class	yes	yes	yes
members of derived class	yes	yes	no
not members	yes	no	no

- Inherited members inherit access permissions dependent on how they are declared
  - Public – same access permissions (default for struct inheritance)
  - Protected – public and protected members inherited as protected
  - Private – all inherited members are private(default for class inheritance)

Inheritance

- What gets inherited?
  - A publicly derived class inherits everything except:
    - constructors and destructor
    - assignment (operator=)
    - friends
    - private members
      - this means that private variables are not inherited
      - need to provide getters and setters
  - Even though not inherited, constructors and destructor are automatically called by the child class

Inheritance Example

```
// constructors and derived classes
#include <iostream>
using namespace std;

class Mother {
  public:
    Mother ()
      { cout << "Mother: no parameters\n"; }
    Mother (int a)
      { cout << "Mother: int parameter\n"; }
};

class Daughter : public Mother {
  public:
    Daughter (int a)
      { cout << "Daughter: int parameter\n\n"; }
};

class Son : public Mother {
  public:
    Son (int a) : Mother (a)
      { cout << "Son: int parameter\n\n"; }
};

int main () {
  Daughter kelly(0);
  Son bud(0);

  return 0;
}
```

```
Mother: no parameters
Daughter: int parameter

Mother: int parameter
Son: int parameter
```

Multiple Inheritance

- Could do this in Python
  - Not so in all languages
- Done by specifying more than one base class separated by commas

Multiple Inheritance Example

```
// multiple inheritance
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int width, height;
  public:
    Polygon (int a, int b) : width(a), height(b) {}
};

class Output {
  public:
    static void print (int i);
};

void Output::print (int i) {
  cout << i << '\n';
}

class Rectangle: public Polygon, public Output {
  public:
    Rectangle (int a, int b) : Polygon(a,b) {}
    int area ()
      { return width*height; }
};

class Triangle: public Polygon, public Output {
  public:
    Triangle (int a, int b) : Polygon(a,b) {}
    int area ()
      { return width*height/2; }
};

int main () {
  Rectangle rect (4,5);
  Triangle trgl (4,5);
  rect.print (rect.area());
  Triangle::print (trgl.area());
  return 0;
}
```

```
20
10
```

Summary

- C++ Classes
  - Friendship
  - Inheritance
  - Multiple Inheritance
  - Polymorphism
  - Virtual Members
  - Abstract Base Classes
- File Input/Output