# 12.010 Computational Methods of Scientific Programming

Lecture 18: Finish C++, Classes in Python

# Overview

- C++ Inhertance
- Classes in Python

# C++ inheritance

- A widely used and useful feature of C++ is inheritance
  - inheritance supports creating a class that is the “child” of one or more other classes
  - as name suggests, the ”child” class by default inherits functions/data types from its “parent(s)”
  - code shown declares a `rectangle::` class that is a child of `poly::`
  - the public elements of `poly::` are public in `rectangle::`

```
class poly{
        public:
         poly();
         void append(ppoint);
         void print();
         double area();
         ppoint *parr;
         int    ncur;
         int    nnodes_max;
        private:
         static const int nblk=10;
};


class rectangle:public poly{
        public:
                rectangle(){};
                rectangle(double, double);
};
```

# C++ inheritance

- Inheritance means that `rectangle::` has `append()`, `area()` and `print()` methods from `poly::`
- By using inheritance, we can create a special case of `poly::` for rectangles and reuse the `poly::` code.

```
class poly{
        public:
         poly();
         void append(ppoint);
         void print();
         double area();
         ppoint *parr;
         int    ncur;
         int    nnodes_max;
        private:
         static const int nblk=10;
};

class rectangle:public poly{
        public:
                rectangle(){};
                rectangle(double, double);
};
```

# C++ inheritance

- The `rectangle::` class only needs to define its own "constructor".
- Everything else can be reused from `poly::`
- In the constructor we use the `append()` method to create a `poly::` that is rectangular with size given by arguments `lx` and `ly`.

```
class rectangle:public poly{
        public:
                rectangle(){};
                rectangle(double, double);
};
rectangle::rectangle(double lx, double ly){
        double x0, y0;
        x0=0.;
        y0=0.;
        append( ppoint(x0,y0) );
        append( ppoint(x0+lx,y0) );
        append( ppoint(x0+lx,y0+ly) );
        append( ppoint(x0,y0+ly) );
        append( ppoint(x0,y0) );
}
```

# C++ inheritance

- In our toy main program we can create a rectangle and use the `poly::` methods `area()` and `print()` directly with the `rectangle::` instance.

```
int main(int argc, char *argv[]){
        poly poly1;
        rectangle r1(2.,3.);
        poly1.append( ppoint(0.,0.) );
        poly1.append( ppoint(2.,0.) );
        poly1.append( ppoint(2.,2.) );
        poly1.append( ppoint(0.,2.) );
        poly1.append( ppoint(0.,0.) );
        poly1.print();
        printf("Poly area = %f\n",poly1.area());
        printf("Rectangle area = %f\n",r1.area())
        r1.print();
}
```

# C++ inheritance

- Try Lec19_poly_area_inherit.cc
  - See if you can create a triangle child class.

# C++ summary

- Classes plus inheritance, polymorphism (same function name different behavior with type/class) and overloading (change meaning of e.g. +) are main elements of object oriented programming.

- C++ also provides a standard library that has some pre-built classes for common activities e.g. arrays, strings, random numbers etc…

- A much larger community library “boost” (https://www.boost.org ) holds many more standard sets of functions for C++.

# Classes and namespaces in Python*

- Material here is in Lec19_classes.ipynb
- Scope:
  - global: available in main workspace
  - nonlocal: available within nested functions (i.e., only defined within scope of nested functions.
- Class: Defines a new class with normally (but not always):
  - __init__ method that is invoked when an instance of a class is created.
  - def of methods always includes self as first argument.
- Instances override attributes set in the class definition
- Inheritances: Class definition includes reference to another class (or classes in python)
- Iterators revisit
- Concept here is instance of class contains both data and methods associated with class. Functions: Pass information through calling arguments; classes information imbedded in instance of class.

MIT OpenCourseWare

https://ocw.mit.edu

12.010 Computational Methods of Scientific Programming, Fall 2024