Queens College, CUNY, Department of Computer Science

Object-Oriented Programming in C++

CSCI 211

Summer 2018

Instructor: Dr. Sateesh Mane

Course Outline

1 Review of basics (estimate $1 - 3$ lectures)

1. Variable types (‘strong typing’)
2. Conditional statements, branching
3. **Switch statement?**
4. Loops — nested loops, break, continue
5. **Scoping of variables?**
6. Arrays & references
7. Functions — call by value, call by reference, return type (void, non-void)
8. `const` **references** (use in function arguments)
9. Recursion

2 Functions new properties? (estimate 1 lecture)

1. Function overloading *(beware of ambiguity)*
2. Default arguments *(beware of ambiguity)*
3. Static functions
4. Static variables *(dangers of usage)*

3 Namespaces (estimate 1/2 lecture)

1. Collision of function names (especially in large projects)
2. Disambiguation using **namespaces**

4 Pointers 1 (estimate 1 lecture)

1. Definition of pointer
2. NULL pointer
3. Dereference '*' and address-of '&' operators
4. Relation of pointers and arrays
5. Pointers in function arguments *(compare/contrast to use of references, pointers can be null)*
6. `const` **pointer** — `const int *pci = &a;` value of *pci cannot be changed
7. Array of pointers *(impossible to have array of references, why?)*
8. **dangling pointers** — *(danger of using pointers)*
9. **AVOID FOR NOW:**
   - (a) Dynamic memory allocation (operator `new, delete`)
   - (b) Pointer arithmetic `p+i, p2-p1, p1 < p2`
   - (c) Reference to pointer, pointer to reference
   - (d) `int * const cpi = &a;` cpi cannot be reset to a different memory address

5 Objects: encapsulation (estimate $2 - 4$ lectures)

1. Objects as containers of data
2. Private, public data
3. Private, public class methods
4. Inline, non-inline declarations
5. Mutator/accessor methods
6. **`const` data**
7. **`const` methods** (accessor, etc.)
8. **`static` data** — how to initialize?
9. **`static` methods – (do NOT need to instantiate class object)**
10. Constructor, destructor, copy & assignment, default constructor
11. **Memberwise data initialization**
12. **Necessity of `const` reference in copy constructor**
13. **Necessity of `this` pointer in assignment operator** — ***(must teach pointers first!)***
14. Deep copy & shallow copy

6 Objects: operator overloading & friend functions (estimate 1/2–1 lectures)

1. Concept of overloading
2. Friend functions — access to private data

7 Objects: inheritance (estimate $3-4$ lectures)

1. Objects as **specializations of a base class**
2. **Base class** — *(what is it?)*
3. Constructor of derived class
4. Destructor of derived class
5. **Protected data & methods**
6. Additional data in derived class
7. Additional methods in derived class
8. Order of construction (base to top) and destruction (top to base)
9. Depth of inheritance: A (= base), B, C, etc.
10. **AVOID:**
    (a) Non-public inheritance.
    (b) Multiple inheritance, public virtual, etc.

8 Objects: polymorphism (estimate $3 - 4$ lectures)

1. **Virtual functions**
   (a) Overriding of virtual functions
   (b) **Redefinition of non-virtual functions** = bad idea
2. **Virtual destructor in base class**
3. **Danger of calling virtual functions in constructor** — *(order of construction)*
4. **Danger of default arguments in virtual functions**
5. Use of pointer (or reference) to base class.
6. **Abstract base class**
   (a) Pure virtual functions
   (b) Protected constructor in base class
7. Concept of **interface**

9 Pointers 2 (estimate $1 - 2$ lectures)

1. Dynamic memory allocation = operator `new, delete, new [], delete []`
2. Pointer arithmetic `p+i, p2-p1, p1 < p2`
3. Reference to pointer, pointer to reference
4. `int * const cpi = &a;` cpi cannot be reset to a different memory address

10 String class (estimate 2 lectures)

1. What to say?

11 Templates (estimate 1 lecture)

1. What to say?

12 STL (Standard C++ library) (estimate $2-3$ lectures)

1. Concept of **generic programming**
2. Container classes = vector, set, map
3. Iterators
4. Algorithms