Thursday, July 6, 2017

12 Sigma

PH IV with Yuanpeng Wu on 6/29 @ 2-3pm

- **Differences between reference and pointer**
  - My summary: Pointers are more flexible and dynamic, they can change what they point at, and can point to nothing (nullptr). References are more static since they cannot "point" towards something else and are more simple since references do not need to be dereferenced in order to change the value of the referred object.
    - From Bjarne Stroustrup's book [CH2, Pg 46]: "A reference is similar to a pointer, except that you don't need to use the prefix * to access the value referred to by the reference. Also, a reference cannot be made to refer to a different object after its initialization."
- **Virtual questions for constructors / destructors**
  - My summary: It is common for abstract classes to NOT have a constructor, since they have no memory or variables to initialize. Instead, "resource acquisition is initialization" (RAII) occurs when a derived object is instantiated. However, most abstract classes do have a virtual destructor, since someone creating then destroying a object derived from an abstract class has no idea about the underlying implementation details for releasing resources appropriately. At runtime, the appropriate virtual destructor(s) will be invoked.
    - From Bjarne Stroustrup's book [CH 3, Pg 64]: "The technique of acquiring resources in a constructor and releasing them in a destructor, known as Resource Acquisition Is Initizliation or RAII, allows us to eliminate "naked new operations," that is, to avoid allocations in general code and keep them buried inside the implementation of well-behaved abstractions. Similarly, "naked delete operations" should be avoided. Avoid naked new and naked delete makes code far less error-prone and far easier to keep free of resource leaks."
    - From Bjarne Stroustrup's book [CH 3, Pg 65]: "Types such as complex and Vector are called concrete types because their representation is part of their definition. In that, they resemble built-in types. In contrast an abstract type is a type that completely insulates a user from implementation details. To do that, we decouple the interface from the representation and give up genuine local

variables. Since we don't know anything about the representation of an abstract type (not even its size), we must allocate objects on the free store an access them through references or pointers."

- The word virtual means "may be redefined later in a class derived from this one." Unsurprisingly, a function declared as virtual is a virtual function. The curious =0 syntax says the function is pure virtual; that is, some class derived from the [abstract class] must define the function"
- From Bjarne Stroustrup's book [CH 3, Pg 70]: "A virtual destructor is essential for an abstract class because an object of a derived class is usually manipulated through the interface provided by its abstract base class. In particular it may be deleted through a pointer to a base class. Then, the virtual function call mechanism ensures that the proper destructor is called. That destructor then implicitly invokes the destructors of its bases and members."

- **Write C++ function to reverse a string "in place" and write a C++ function to reverse words in a string**