* Using \_\_\_ - specify to compiler use of \_\_\_ is from x namespace.
* Why STL array – better functions / has size.
* Struct features – combine data types under a single name (like a tuple)
  + Can be passed by ref or by value
* main parts of class – data memb, function memb, nested types, enumerators, member templates
* classes have 3 funcs – setter, getter, constructor.
* PIE – polymorphism, inheritance, encapsulation
  + Polymorphism – ability to process objects differently based on type/class.
  + Inheritance – subclasses of methods and variables defined in parent/super.
  + Encapsulation – access control, set scope on methods or properties.
* Why better to be vector – dynamic size, functions (push and pop)
* Public – if needs to be accessed outside of scope (getters, setters, and similar)
* Private – anything internal calc or doesn’t need to be accessed beyond
* never use float, use double
* constructors – initialize object / variables.
* overload constructors and functions (poly) – constructor with parameters to initialize object with
* constructors with vectors – don’t need this->, just do vector.push\_back()
* principle – have validation for input.
* Getters not always void, could return value if works/fails
* Why did polymorphism help here? Abstraction, dynamic, recycle code use multi-type/overload
* Where to use struct/class?
  + Struct for data structures, classes for structs with private members/non default constructors
* Array vs vector
  + array is mem efficient, one type, index, 0(1) access, not copy
  + vectors are dynamic, similar to array mostly, copy possible
* for vs for-each
  + for loops are range based, fast, explicit -- for-each is easy, good for output (mostly?)

Quiz content

|  |  |
| --- | --- |
| Headers:  Use #include to use headers  Helps linker with symbols, separate implement from interface, improve adaptability  Full definition of constants  No logic in .hpp, use in complement .cpp  #include rando.h is prob custom library | Class  Class is blueprint (poly)  Real world objects have state and behavior (inhe)  Blocks are collections of statements (encap)  Poly – process things based on type / class  Inher – subclasses of members  Encap – access / scope put on members |
| Overload and encap  Over-constructor is 2nd constructor w/ parameter  Protected means namespace access only  Use instance.function() to access  Internal calculations should be private | Vectors  Better to work w/ vector in function and push rather than pass, better on mem efficient  Vectors do have max size (per machine)  Use .at(), has bounds check  Buffer overflow – heap/stack accessed where it shouldn’t be, overwritten, bad.  Maybe no auto when you know the type you want to work with  Vector <bool> special object tuned for allocation  Vectors can be though of as arrays  Auto type would be dynamic if input changes |