

# Programming Languages and Tools: Programming with C++ CS:3210:0003

Lecture/Lab #12

# Classes vs Structs

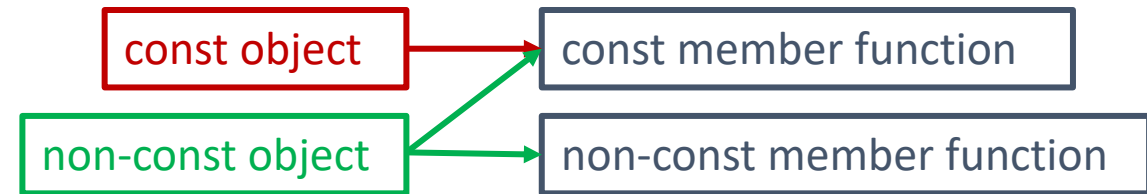
- In C, structs have limited capability.
- Classes were added to C++ to extend this capability,
  - However, most of the extended capability was also given to structs
- Often, classes and structs are interchangeable
- Convention dictates when to use one over the other.
- We'll look at features that are common to both classes and structs and point out differentiating features when they arise
- **Class types** refer to both

# Member Functions

- Class types can have their own functions - **member functions**
- Must be declared inside the class, can be defined inside/outside
- Remember, definitions implicitly contain declarations
- Member functions can be called using the dot (.) operator on an object
- The object on which the member function is called is an implicit argument to the function

# Const Member Functions

- Member functions can be made const to guarantee that
  1. they won't modify object
  2. call non-const member functions



- Syntax:  
`returnType funcName (args) const`
- If your member function isn't going to modify object (member variables), make it const
- If your object isn't going to be changed and you have made your member functions const, call it by const reference

# Member Access

- **Access level** of member determines who can access it:
  1. public: can be accessed by anyone (in scope)
  2. private: can only be accessed by other members of same class/struct
  3. protected: concerned with inheritance
- Illegal access leads to compilation error
- Use access specifiers to make access level explicit:  
public:, private:, protected:

Struct	Class
Members public by default	Members private by default

# Access Functions

- Public functions used to access/change private variables
- **Getters/accessors** return values of private variables
- **Setters/mutators** set values of private variables
- For Fraction class:
  - Getters: `int getNum() const, int getDen() const`
  - Setter: `setFrac(int num, int den)`
- Getters should be read-only: return by value or const reference

# Best Practices

- Start names of member variables with `m_` (especially private)
- Leave all members of structs public (default)
  - This allows us to initialize using aggregate list, ex: `Fraction y {5, 3};`
- No member variables of classes should be public
  - Public member functions can provide access to private members