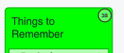
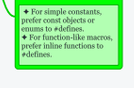
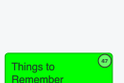
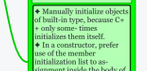
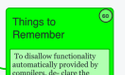
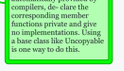
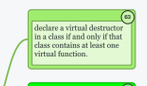
- effective C++
  - Item 1
    - View C++ as a federation of languages
    - C
    - Object-Oriented C++
    - Template C++
    - The STL
    - Things to Remember
      - Rules for effective C++ programming vary, depending on the part of C++ you are using.
  - Item 2
    - Prefer consts, enums, and inlines to #defines
    - Use Const to replace #define
      - const
        - Note, by the way, that there's no way to create a class-specific con- stant using a #define, because #defines don't respect scope. Once a macro is defined, it's in force for the rest of the compilation.
    - the enum hack
      - 
    - Use Inline template function to replace macro
      - 
    - Things to Remember
      - ◆ For simple constants, prefer const objects or enums to #defines.
      - ◆ For function-like macros, prefer inline functions to #defines.
  - Item 3
    - Use const whenever possible.
    - const use
      -
      -
      -
    - const Member Functions
      - The purpose of const on member functions is to identify which mem- ber functions may be invoked on const objects.
      -
      - use mutable keyword: change member variable in const object
        -
    - Things to Remember
      - ◆ Declaring something const helps compilers detect usage errors. const can be applied to objects at any scope, to function parameters and return types, and to member functions as a whole.
      - ◆ Compilers enforce bitwise constness, but you should program using logical constness.
      - ◆ When const and non-const member functions have essentially identi- cal implementations, code duplication can be avoided by having the non-const version call the const version.
  - Item 4
    - Make sure that objects are initialized before they're used.
    - if tfs isn't initilized, it will be wrong.
      - 
    - use Singleton patern (static) to make sure that it's initialized.
      - 
    - Things to Remember
      - ◆ Manually initialize objects of built-in type, because C++ only some- times initializes them itself.
      - ◆ In a constructor, prefer use of the member initialization list to as- signment inside the body of the constructor. List data members in the initialization list in the same order they're declared in the class.
      - ◆ Avoid initialization order problems across translation units by re- placing non-local static objects with local static objects
  - Item 5
    - Know what functions C++ silently writes and calls.
    -
    - Things to Remember
      - Compilers may implicitly generate a class's default constructor, copy constructor, copy assignment operator, and destructor.
  - Item 6
    - Explicitly disallow the use of compiler- generated functions you do not want.
    - 
    - Things to Remember
      - To disallow functionality automatically provided by compilers, de- clare the corresponding member functions private and give no implementations. Using a base class like Uncopyable is one way to do this.
  - Item 7
    - Declare destructors virtual in polymorphic base classes.
    - declare a virtual destructor in a class if and only if that class contains at least one virtual function.
    - Things to Remember
      - ◆ Polymorphic base classes should declare virtual destructors. If a class has any virtual functions, it should have a virtual destructor.
      - ◆ Classes not designed to be base classes or not designed to be used polymorphically should not declare virtual destructors.
  - Item 8
    - Prevent exceptions from leaving destructors.
    - 
    - 
    - Things to Remember
      - ◆ Destructors should never emit exceptions. If functions called in a destructor may throw, the destructor should catch any exceptions, then swallow them or terminate the program.
      - ◆ If class clients need to be able to react to exceptions thrown during an operation, the class should provide a regular (i.e., non-destruc- tor) function that performs the operation.
  - Item 9
    - Never call virtual functions during construction or destruction.
    - Things to Remember
      - ◆ Don't call virtual functions during construction or destruction, because such calls will never go to a more derived class than that of the currently executing constructor or destructor.
  - Item 10
    - Have assignment operators return a reference to *this.
  - Item 11
    - Handle assignment to self in operator=.
    - 
    - 
      -
    - 
    - Things to Remember
      - ◆ Make sure operator= is well-behaved when an object is assigned to itself. Techniques include comparing addresses of source and target objects, careful statement ordering, and copy-and-swap.
      - ◆ Make sure that any function operating on more than one object be- haves correctly if two or more of the objects are the same.