

Development Tools and Practices

IE 496 Lecture 10

Reading for This Lecture

- Norm Matloff's Debugging Tutorial

Development Tools

Compilers

- GCC
- Intel
- Portland Group
- Borland
- Microsoft Visual Studio
- Oracle Studio
- x1C

Integrated Development Environments

- Microsoft Visual Studio
- Eclipse
- Anjuta
- Dev-C++

Editors

- IDEs
- Emacs
- Notepad++

Other Tools

- Version Control (CVS/SVN)
- Make
- CMake
- Autotools

Debugging

- IDEs
- GDB
- DDD
- Valgrind
- Electric Fence
- Purify

Profiling

- Gprof
- Quantify

Object Oriented Programming

- Object-oriented programming is a paradigm that emphasizes
 - *Data* rather than *methods*
 - Code reuse
 - Separation of *interface* from *implementation*
- Following good object oriented coding practices will lead to more useful code.

Data Types

- A *data type* is a set of *data values* and a set of *operations* that can be performed on those values.
- Data types are a mechanism by which programmer can define new *data structures*.
- What are some examples of data structures?

Classes

- Classes are the mechanisms by which new data types can be defined.
- A class is composed of
 - Data members
 - Member functions
- The data members are the **values**.
- The member functions are the **operations**.
- There are also *constructors* and *destructors* by which objects of the new type can be created and destroyed

Designing Classes

- Ideally, we would like to separate the *definition* of the class from the *implementation*.
 - The *definition* specifies what the data values are and what operations we would like to perform on them.
 - The *implementation* specifies the algorithms used to perform those operations.
- What is the reason for this separation?

The Interface

- The *interface* defines that way in which a client would actually use the data type.
- In C++, the interface consists of the *public members* of the class.
- The *private members* of the class along with the function implementations are the implementation.
- It is good programming style to keep all data members private.
 - Data members define how the data is stored, which is implementation-dependent
 - Access to data values can be provided through query methods.
 - This allows changing the implementation without affecting clients

Some C++ Style Recommendations

- All objects should be explicitly constructed.
- Constructors should initialize all data members.
- All memory allocated with new should be deleted.
- Destructors should delete all allocated memory.
- No public data members.
- No global variables or functions.
- (Almost) no executable code in header files.

Examples

Development Practices

Good Development Practices

- Use version control
- Make code readable
 - Formatting
 - Comments and documentation
 - Naming conventions
- Develop good unit tests
- Make code reusable
 - Modularity
 - API

Debugging

- Debugging is a process of verifying that certain invariants that you expect actually hold.
- When the code is not working, you must look for inconsistencies that indicate a violated invariant.
- Modularity and good unit testing make this process much easier.
- Example: Debugging Insertion Sort

Memory debuggers

- Memory debuggers are tools that can help you find errors such as reads/writes to unallocated memory.
- They can also help you find memory leaks.
- These types of errors are particularly difficult to find in practice because they may not cause a crash.
- They also may cause random behavior that could be different from one run to the next.

Profilers

- A profiler can help you determine where the bottlenecks are in your code.
- The profiler will tell you
 - How many times each function was called
 - How much time was spent in each function

Examples