Lesson 1: Generic Programming

Generic programming

C++ templates

C++ standard templates

1.1 Coding Conventions

- Names
- Braces
- Operators

1.2 Generics & STL Review

- What is generic programming?
- STL
  - Containers
  - Iterators
  - Algorithms

1.3 Prefer vector and string

- Dangers of dynamically allocated arrays
- Vector & string benefits

1.4 Iterators

- Find hidden issues with iterator code
- Iterator guidelines

1.5 Case-Insensitive Strings

- What does case-insensitive mean?
- Implement case-insensitive string
- Is case-insensitivity a good idea?

1.6 Case-Insensitive Strings 2

- Is it safe to inherit from char_traits?
- Investigate compilation failures
- Other problems with CI strings

1.7 Reusable Generic Containers

- Implement fixed_vector to hold any type
  - Copy construction
  - Copy assignment
  - Make implementation reusable
- Another approach
  - How it’s done in the STL
- Strong exception safety

1.8 Temporary Objects

- Find issues in code
- Correct the code

$$\underbrace{\underbrace{\texttt{x + x}}_{\text{temp1}} \texttt{ + } \underbrace{\texttt{y * y}}_{\text{temp2}}}_{\text{temp3}}$$

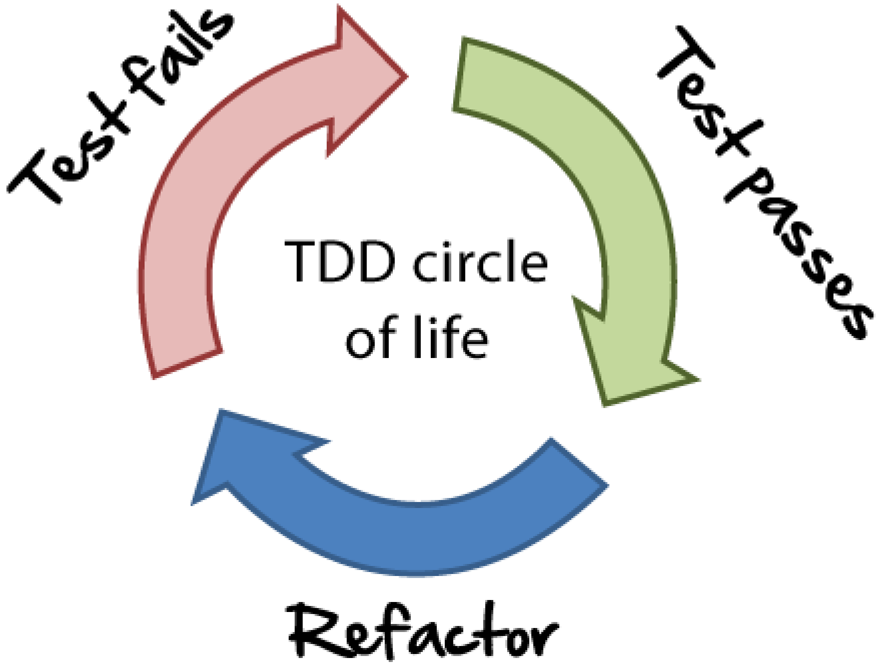
1.9 Using the Standard Library

- Standard library avoids problems
  - Well designed
  - Heavily tested
  - Easier-to-read code

THE C++
STANDARD
TEMPLATE
LIBRARY

1.10 Test Driven Development

- TDD Cycle
- Benefits



1.11 TDD Example

- Use TDD to develop new class
- Tests define new behavior
- Tests ensure old behavior still works