
boost
C++ LIBRARIES

Boost Libraries

- Boost is a set of libraries for the C++ programming language that provide support for tasks and structures such as *linear algebra*, *pseudorandom number generation*, *multithreading*, *image processing*, *regular expressions*, and *unit testing*
- The Boost community emerged around 1998, when the first version of the standard was released
- It has grown continuously since then and now plays a big role in the standardization of C++.

https://theboostcpplibraries.com/

Boost Libraries

- Less code -> Real productivity
  - Less to write
  - Less to debug
- More *expressive* code
  - Natural to write
  - More self-documenting
  - More likely to be correct the first time

Test-driven development (TDD)

- A software development process that relies on the repetition of a very short development cycle:
  - requirements are turned into very specific test cases, then the software is improved to pass the new tests, only
  - opposed to software development that allows software to be added that is not proven to meet requirements
- Kent Beck (one of the 17 original signatories of the Agile Manifesto) 2003

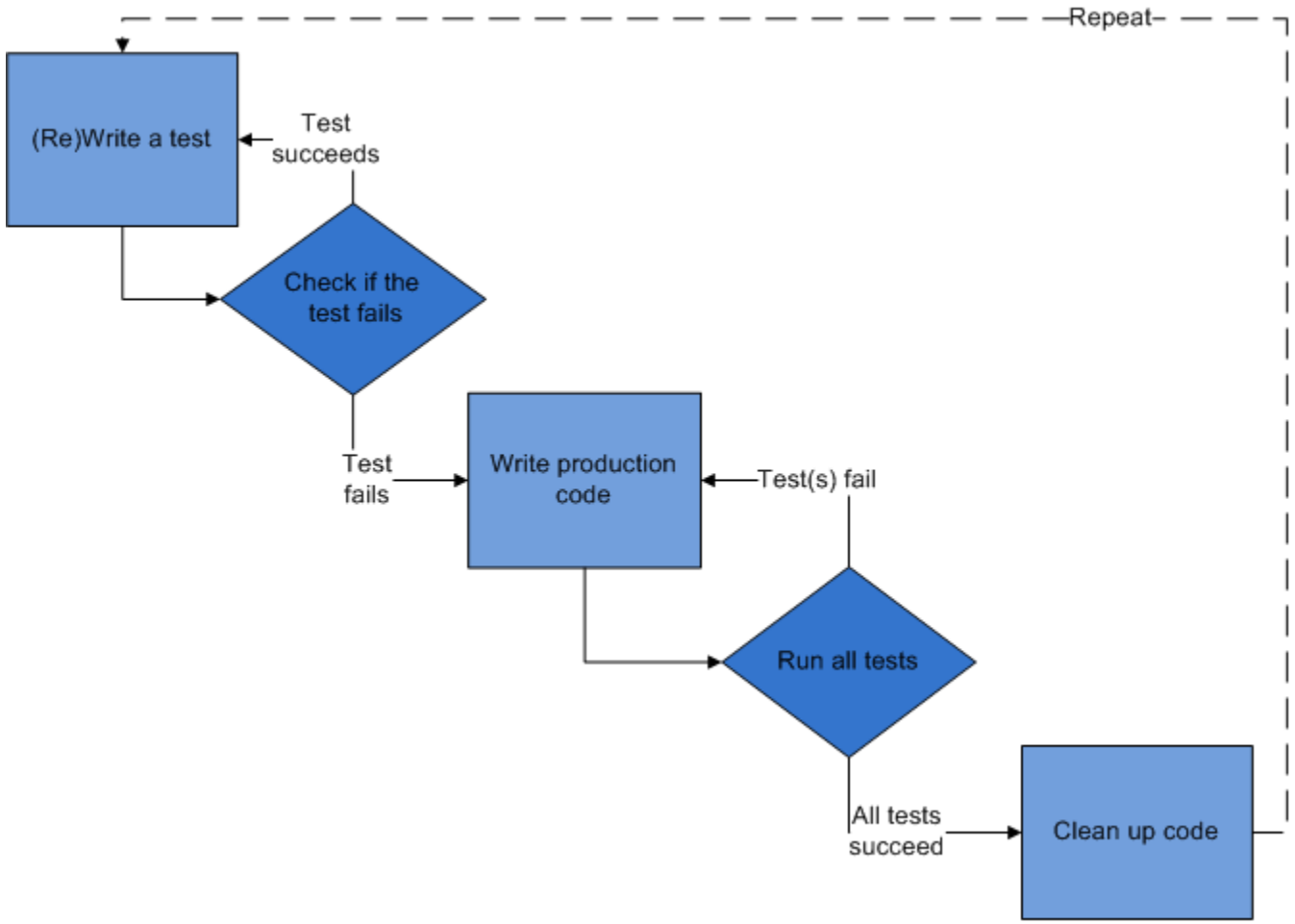
https://en.wikipedia.org/wiki/Agile_software_development#The_Agile_Manifesto

Test-driven development (TDD)

- TDD is a software development process which relies on the repetition of a very short development cycle
  1. Add a test case that defines a desired improvement or new function
  2. Run all tests and see if the new one fails
  3. Write some code to pass the test
  4. Run tests
  5. Refactor code
  6. Repeat

https://en.wikipedia.org/wiki/Test-driven_development

Test-driven development (TDD)



http://alexott.net/en/cpp/CppTestingIntro.html

Software testing

- Many different forms of tests:
  - unit tests:
  - integration tests: to test if Individual units are combined and functioned as a group.
  - regression tests: to uncover new software bugs in existing functional
  - performance tests: to determine how a system performs in terms of responsiveness and stability under a particular workload

Unit testing

- Unit testing is a method by which individual units of source code are tested to determine if they are correctly working
- A unit is the smallest testable part of an application.
  - an individual function or procedure

Benefits of unit testing

- Facilitate changes: unit tests allow programmers to refactor code at later date, and be sure that code still works correctly
- Simplify integration: unit tests may reduce uncertainty in the units, and can be used in a bottom-up testing style approach
- Living documentation for the system: easy to gain a basic understanding of implemented Application program interface (API)

How to organize tests

- Each test case should test only one thing
- A test case should be short
- Test should run fast, so it will possible to run it very often
- Each test should work independent of other test
- Tests should NOT be dependent on the order of their execution

Test methods

- How should a test program report errors? Displaying an error message would be one possible way

```
if( something_bad_detected )
std::cout << “Something bad has been detected” << std::endl;
```

  - requires inspection of the program’s output after each run to determine if an error exists.
  - Human inspection of output message is time consuming and unreliable
- Unit testing frameworks are designed to automate those tasks

Test methods:
Unit testing frameworks

- To simplify development of unit tests, unit test frameworks are usually used
- Almost any programming language has several unit testing frameworks

Unit testing frameworks in C++

- There are many unit testing frameworks for C++
  - Boost.Test, Google C++ Testing Framework, etc, etc, etc!
- Boost.Test
  - Suitable for novice and advanced users
  - Allows organization of test cases into test suites
  - Test cases can be registered automatically and/or manually

Unit testing frameworks in C++

- Boost.Test
  - Fixtures (initialization and cleanup of resources)
  - Large number of assertion/checkers
    - ✓ Exceptions
    - ✓ Equal, not equal, greater, less
    - ✓ Explicit fail/success
    - ✓ Floating point numbers comparison etc.

Testing tools/Checkers

- WARN
  - produces warning message check failed, but error counter does not increase and test case continues
- CHECK
  - reports error and increases error counter when check fails, but test case continues
- REQUIRE
  - similar to CHECK, but it is not used to report “fatal” errors.

Testing tools/Checkers

- Expression examples

```
BOOST_WARN( sizeof(int) ==sizeof(short));
BOOST_CHECK(i == 1);
BOOST_REQUIRE (i > 5);
BOOST_REQUIRE('h', s[0]);
```

- If the check fails, **Boost.Test** will report the line in the source code where this occurred and what condition was specified.

Shift register

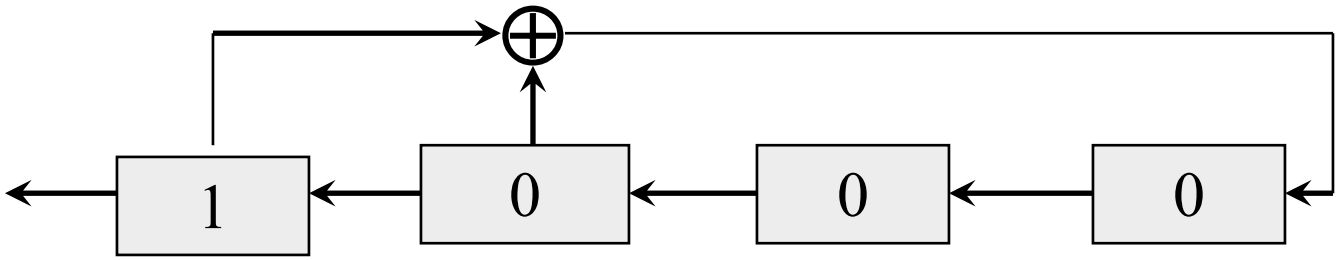
- Shift register consists of an arrangement of flip-flops and are important in applications involving the storage and transfer data in a digital system, it is a type of sequential logic circuit, mainly for storage of digital data.
- They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop.

Shift register: LFSR

- Linear Feedback Shift Register (LFSR) is popularly known as Pseudo-random number generator.
- The random numbers repeat itself after 2^n-1 clock cycles (where n is the number of bits in LFSR)

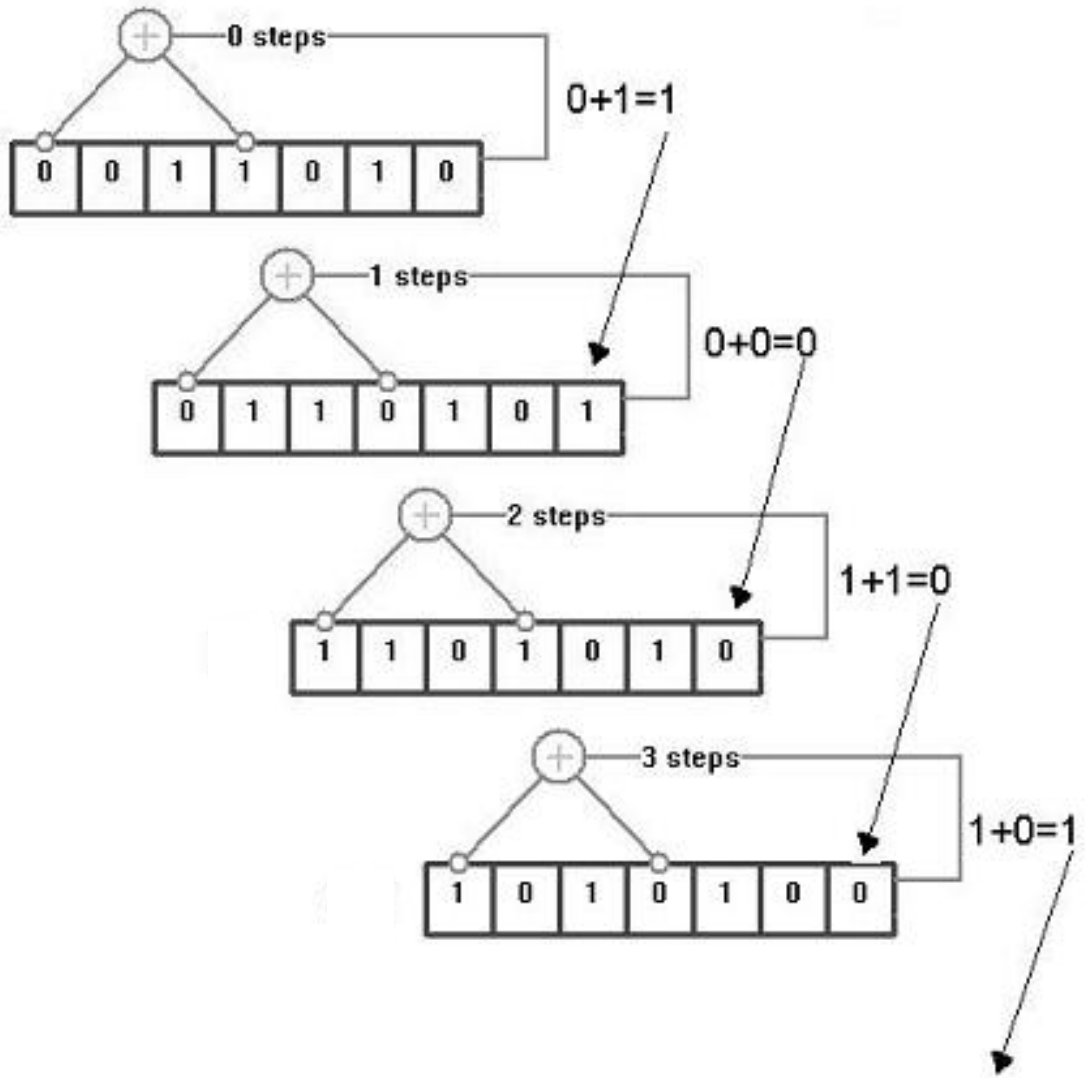
Shift register

- Starting with 1000, the output stream is
  1000 1001 1010 1111 000
- Repeat every $2^4 - 1$ bit
- The seed is the key

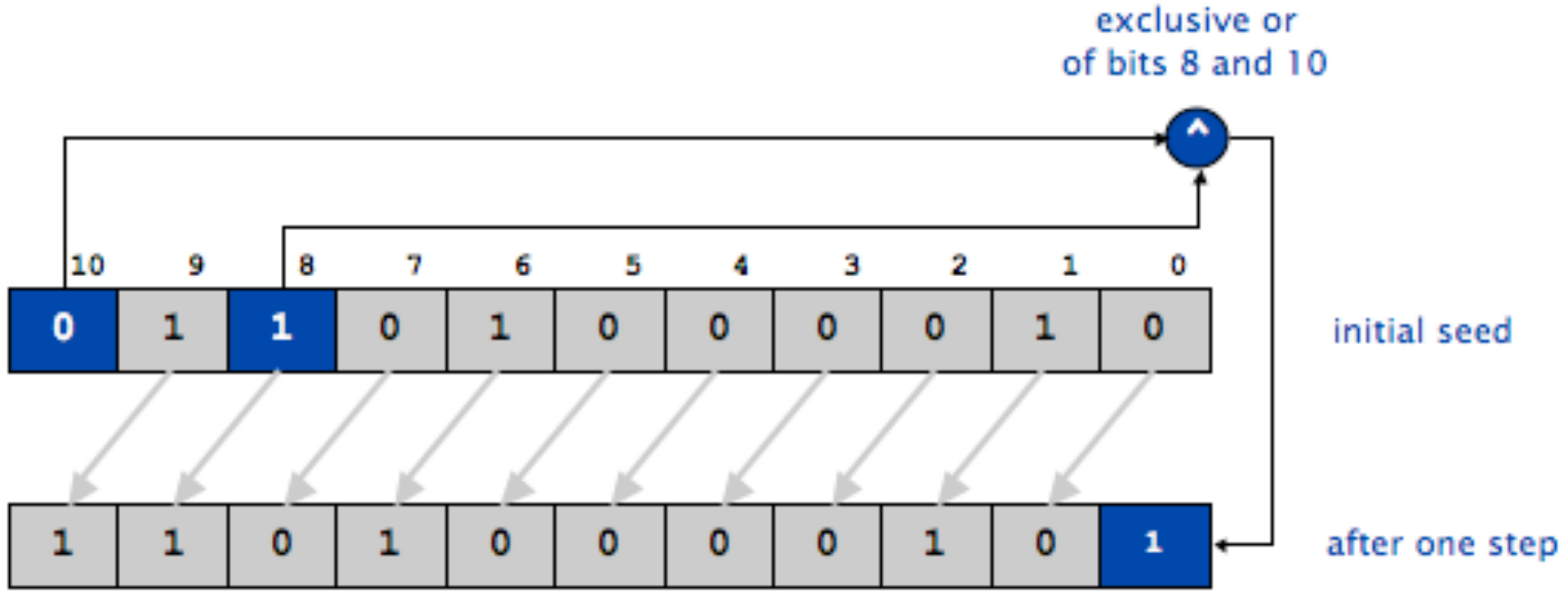


INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Shift register



Shift register



One step of an 11-bit LFSR with initial seed 01101000010 and tap at position 8

Shift register

APPLICATIONS

- As a counters
- Random number generators
- Error detection and correction
- Cryptography
- Jamming
- Test-pattern generation