Safe Language (Strong Type)

Ada, Perl, Python, Java, C#, and even Visual Basic

- Have automatic bounds checking, and do not have direct memory access

C-derivatives: Rust (Mozilla 2010)

- Designed to be a "safe, concurrent, practical language", supporting functional and imperative-procedural paradigms
- Does not permit null pointers, dangling pointers, or data races
- Memory and other resources are managed through "Resource Acquisition Is Initialization" (RAII).

Go: type-safe, garbage-collected but C-looking language

- Good concurrency model for taking advantage of multicore machines
- Appropriate for implementing server architectures.

Outline

- Safe Programing
- **Software Testing**
- Compiler and System Support

Manual Code Reviews

Peer review

- Very important before shipping the code in IT companies

Code review checklist

- Wrong use of data:

  variable not initialized, dangling pointer, array index out of bounds, …
- Faults in declarations

  undeclared variable, variable declared twice, …
- Faults in computation

  division by zero, mixed-type expressions, wrong operator priorities, …
- Faults in relational expressions

  incorrect Boolean operator, wrong operator priorities, …
- Faults in control flow

  infinite loops, loops that execute n-1 or n+1 times instead of n, ...

Writing Software Tests

Unit tests

- Test individual components or functions of the software in isolation
- Unit tests should cover all code, including error handling

Regression tests

- Test that new code changes do not negatively affect existing functionality
- Verify that the software continues to function correctly after updates

Integration tests

- Test the interaction between multiple software modules or systems
- Ensure that components work together as expected.

Static Analysis

Analyze the source code or binary before running it (during compilation)

- Explore all possible execution consequences with all possible input
- Approximate all possible states
- Identify issues during development, reducing the cost of fixing vulnerability
- Rely on predefined rules or policies to identify patterns of insecure coding practice

Static analysis tools

- Coverity: https://scan.coverity.com/
- Fortify: https://www.microfocus.com/en-us/cyberres/application-security
- GrammarTech: https://www.grammatech.com/

Limitations

- May produce false positives, requiring manual review
- Cannot detect runtime issues, e.g., logical errors, dynamic environment-specific flaws