Software Engineering

Moscow Institute of Physics and Technology

01. Introduction and Overview

- Course Introduction
- Developer Tools
- Standard Library Overview

02. Basics of Programming

- Types and Variables
- Operators and Expressions
- Memory Management
- Lvalue References
- Functions and Algorithms

03. Object - Oriented Programming

- Structures and Classes
- Object Relations
- Dynamic Polymorphism
- Rvalue References
- Operator Overloading

04. Generic Programming

- Function Templates
- Class Templates
- Forwarding References
- Metaprogramming
- Constant Expressions

05. Software Design Patterns

- Generative Patterns
- Constructional Patterns
- Behavioral Patterns
- Mixin - Based Patterns

06. Projects and Libraries

- Preprocessor Directives
- Multi - File Projects
- Modular Programming
- Additional Libraries

07. Debugging and Profiling

- Irrecoverable Errors
- Function Return Codes
- Exception Handling
- Debugging and Profiling

08. Number Processing

-

09. Memory Management

-

10. Data Structures

-

11. Algorithms and Ranges

-

12. String Processing

-

13. Streams and Serialization

-

14. Parallel Programming

-

15. Computer Network Systems

-

- 07.11 – Statements `throw`, `try` and `catch`. Stack unwinding. User-defined exceptions. Attribute `noreturn`.
- 07.12 – Exception safety guarantees. Specifier and operator `noexcept`. Zero-overhead principle.
- 07.13 – Exception safe container `Stack` interface.
- 07.14 – Backtracing. Call stack. Container `std::stacktrace`.

Debugging and Profiling

- 07.15 – Debugging. Debugger GDB. Commands run, continue, next, step, break, print, list and backtrace.
- 07.16 – Memory profiling. Internal compiler sanitizers.
- 07.17 – External sanitizers Valgrind.
- 07.18 – Performance profiling. Profiler Callgrind. Visualizer KCachegrind.
- 07.19 – Logging. Library Boost.Log.
- 07.20 – Library Google.Log.
- 07.21 – Testing. Unit tests. Test suites and cases. Datasets. Fixtures. Library Boost.Test.
- 07.22 – Assertions. Expectations. Library Google.Test.
- 07.23 – Microbenchmarking. Complexity evaluation. Library Google.Benchmark.

Bitwise Processing

- 08.01 – Number systems. Binary, octal, decimal and hexadecimal literals.
- 08.02 – Bitwise logical operators. Bitwise swap algorithm.
- 08.03 – Reflected binary Gray code. Gray code encode and decode algorithms.
- 08.04 – Encoder implementation.
- 08.05 – Bit fields. Type `Timestamp`.
- 08.06 – Benchmarks for bit fields.
- 08.07 – Fixed-size sequences of bits. Container `std::bitset`.
- 08.08 – Enumeration `std::byte`.
- 08.09 – Endianess. Big and little endian byte orders. Enumeration `std::endian`. Function `std::to_integer`.
- 08.10 – Reinterpreting bits. Operator `reinterpret_cast`. Type punning. Function `std::bit_cast`.

Long Arithmetic

- 08.11 – Indian exponentiation algorithm.
- 08.12 – Factorial for type `int`. Utility `std::numeric_limits`.
- 08.13 – Long arithmetic. Type `Integer`. Long arithmetic and comparison operators. Square root algorithm.
- 08.14 – Karatsuba fast multiplication algorithm.
- 08.15 – Appendix: main.
- 08.16 – Factorial for type `Integer`.
- 08.17 – Extended checked and unchecked integer types. Library Boost.Multiprecision.
- 08.18 – Factorial for type `boost::multiprecision::cpp_int`.
- 08.19 – Embedding Python. Python C/C++ API. Global interpreter locker. Library Boost.Python.
- 08.20 – Factorial for type `boost::python::api::object`. Module `math`.

Floating Point Types

- 08.21 – Standard IEEE-754. Precision. Exponent. Infinity. Quiet and signaling NaNs.
- 08.22 – Floating point numbers compare algorithms. Absolute and relative epsilon constants. Function `std::abs`.
- 08.23 – Extended floating point types.
- 08.24 – Numerical methods. Derivatives. Special math functions. Library Boost.Math.
- 08.25 – Weighted mean and variance. Library Boost.Accumulators.
- 08.26 – Complex numbers. Type `std::complex`. Functions `std::real`, `std::imag` and others.
- 08.27 – Discrete Fourier transform algorithm.

Random Numbers

- 08.28 – Non-deterministic generators. Entropy sources. Seeds. Engines. Distributions.
- 08.29 – Appendix: scheme.
- 08.30 – Monte-Carlo methods. Pi constant estimation.
- 08.31 – Appendix: scheme.
- 08.32 – W. L. Putnam mathematical competition problem. Probability estimation. Barycentric coordinate method.
- 08.33 – Benchmarks for branch predictor.

Chrono Management

- 08.34 – Namespace `std::chrono`. System, steady and high-resolution clocks. Time points. Unix epoch.
- 08.35 – Durations. Duration type conversions.
- 08.36 – Durations since epoch. C-style time. Type `std::time_t`. Function `std::time`.
- 08.37 – Utility `Timer`.
- 08.38 – Library Boost.Timer.
- 08.39 – Calendars. Years. Months. Days. Hours. Minutes. Seconds.
- 08.40 – Time zones.
- 08.41 – Namespace `std::literals`. User-defined literals. Literal operators.