A Look at Modern C++

Dmitri Nesteruk, technical evangelist @ JetBrains

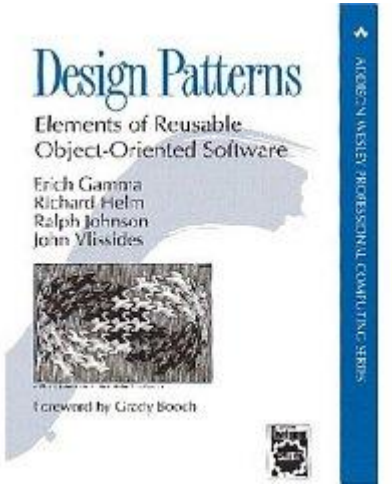
dmitrinesteruk@gmail.com

@dnesteruk

(Dead?) Languages

[redacted]

- C (1972)
- Objective C (1983)
- C++ (1983)
- Java (1995)
- C# (2000)
- Swift (2014)



"IMITATION IS THE *sincerest form* OF FLATTERY."

What is C++?

- A successor to C (an imperative language), which is (arguably) the most popular language ever
  - C with classes and a million other things
  - C++ is backwards compatible with C
- Compiles to native code
  - There is no virtual machine
  - Resulting program ready to be executed without additional JIT compilation process
- Fast
  - Low-level hardware access (raw memory, SIMD, ASM)
- Can be portable
- Some very mature libraries and tools
  - We’re working on IDEs
- Rapidly evolving

Key fields

- Key industries
  - Game Development
  - Embedded Development
  - Quant Finance
- Anyone who needs top performance
  - Device drivers
  - High Performance Computing (HPC)
  - Interoperability with other languages possible
- C/C++ used to leverage custom hardware
  - CUDA C
  - Intel Xeon Phi
  - OpenCL for FPGA

Problems

- Has no garbage collection mechanism
  - Manual memory management
  - Deterministic destruction
- Historically verbose
  - But getting better
- Lack of metadata
  - Lack of attributes/annotation, proper reflection, etc.
  - Run-Time Type Information (RTTI) a poor substitute
- Has a preprocessor
  - Possible source of bugs and ambiguities
- Unreadable compiler messages
  - Reduces your chances to find out what's wrong and where
- Compilation is slow
  - Cluster builds!
- Poor Unicode support
- Backward compatibility with C
- Allows write-only magic (e.g. template metaprogramming)
- Testing is more difficult
  - No metadata, remember?
- Third-party libraries often shipped as source code

Compilers

- Lots of compiles out there
- **Microsoft Visual C++ Compiler** (MSVC), shipped with Visual Studio, supports Windows, Microsoft-specific technologies (e.g., C++ AMP), .NET-compliant C++ dialects
- **Intel C++ Compiler**
  commercial, part of Intel Parallel Studio, includes a large number of libraries and tools, integrates into Visual Studio, works on Win/Linux/OSX
- **LLVM (Clang** front-end)
  modern, good diagnostic messages
- ... and many others

Build Systems

- Local build systems
  - MAKE
  - Cmake
  - MSBuild
- Distributed build systems
  - Distcc
  - ElectricAccelerator
  - IncrediBuild

Libraries

- **C++ Standard Library**
  Standard Template Library (STL)
- **Boost**
  Slowly migrating to C++ Standard Library
- **Qt Framework**
  Cross-platform UI development
  Separate IDE or Visual Studio integration
- **Intel Libraries** (also part of Intel Parallel Studio)
  - **Math Kernel Library (MKL)**
  - **Threading Building Blocks (TBB)** compatible with Microsoft PPL
  - **Integrated Performace Primitives (IPP)**
  - … and many more

Preprocessor

- Initial compilation stage
- Your chance to send hints and instructions to the compiler
  - For example, to pick which parts of a file get compiled and which do not
- Preprocessing instructions start with #
  - #include “foo.h” includes a file in the current file
- C# has similar functionality
  - And good alternatives such as [Conditional]
- C++ functionality is much more powerful
  - And much more dangerous

Compilation

- After processing, all #include statements are implemented
  - In other words, each .cpp file has all its includes included
    - But note before the includes' includes directives
      - And so on
- Each .cpp becomes one *very big* file
  - Self-contained!
- This explains why compilation is
  - Slow
  - Can be parallelized at file level

Linking

- Linking joins all the object files together into an executable or library
- In addition to your own code, you (probably) want to link to other, external libraries
- External libraries can be
  - Static – the library is included wholesale in your program. There are no additional files that need to be shipped
  - Dynamic – the library exists as a separate file
    - DLL on Windows, Shared Library on Linux
    - If the library is not available, program will crash
- To link to a library you need
  - Its header files (some libs are header-only, nothing else needed)
  - Library files

Interop Options

- Windows options listed here
- Platform invocation services (P/Invoke)
  - Export from C++, [DllImport] in C#/VB
  - Functions only!
- Component Object Model (COM)
  - Cross-language interoperability model
  - Still used by Microsoft Office and major Windows applications
  - Automation interface, dynamic keyword
- Managed C++, C++/CLI
  - C++ variants that include .NET extensions
  - Can be used to build a bridge between native and .NET worlds

That’s it!

- Questions?