TDDD38 - Advanced programming in C++

Course introduction

Eric Elfving (modified by Christoffer Holm)

Department of Computer and information science

LINKÖPING UNIVERSITY

1 What is TDDD38
Syllabus
Organization
Exam
Literature
Plan
2 What is C++?
History
Design rules
Design support rules
Design support rules
Design support rules
Coding style

LINKÖPING UNIVERSITY

1 What is TDDD38
Syllabus
Organization
Exam
Literature
Plan
2 What is C++?
History
Design rules
Design support rules
Design support rules
Design support rules
Coding style

LINKÖPING UNIVERSITY

What is APiC++?

Aim (syllabus)

- Explain non-trivial C++ language constructs and their semantics
- Explain the overall design principle of the C++ standard library

What is APiC++?

Aim (syllabus)

- Explain non-trivial C++ language constructs and their semantics
- Explain the overall design principle of the C++ standard library
- Design and implement usable, correct, error-safe, non-trivial classes and polymorphic classes.
- Design and implement advanced program components

What is APiC++?

Aim (syllabus)

- Explain non-trivial C++ language constructs and their semantics
- Explain the overall design principle of the C++ standard library
- Design and implement usable, correct, error-safe, non-trivial classes and polymorphic classes.
- Design and implement advanced program components
- Use different components from the C++ standard library in combination

What is APiC++?

Prerequisites (syllabus)

- Good knowledge and skills in programming in at least one procedural and/or object-oriented language
- knowledge about fundamentals of object-oriented programming

What is APiC++?

Organization

- Basically a self-study course
- No examining labs
  - Several optional (but highly recommended) exercises
  - No lab time scheduled
- Contact me for assistance - preferably via email
- You can also visit the scheduled office hours
- "Seminars"

What is APiC++?

Seminars in APiC++

- Material will be added to the course webpage and the gitlab group `https://gitlab.ida.liu.se/TDDD38`
  - Lecture slides
  - Links to blogs, videos from conferences, discussions on SO etc.
  - Code examples
- This is only to support your learning, of course it's up to you if you want to attend or not.

What is APiC++?

Computer examination - five hours

- Theory questions - 5 points in total
- Four programming problems - 5 points each
  - Class design, derivation, operator overloading, exception handling
  - STL - standard library
  - Standard techniques and templates
- Grades - 25 points maximum
  - 19-25 - 5/A
  - 15-18 - 4/B
  - 11-14 - 3/C
  - 0-10 - U/FX
- Access to the *reference* part of cppreference.com
- Important to be familiar with the linux system

Extra information, literature, etc.

- Course web page:
  `http://www.ida.liu.se/~TDDD38/`
  - Seminar plan and material
  - Exercises
  - C++ links
  - Contact information
  - Information on the exam

Extra information, literature, etc.

- Course literature - basically up to you
  - C++ Primer, 2012, Lippman, Lajoie, Moo
  - The C++ Programming Language, 4th edition, 2013, Stroustrup
  - A Tour of C++, 2013, Stroustrup
  - The C++ Standard Library, 2012, Josuttis
  - Effective Modern C++, 2014, Meyers
  - cppreference.com
- Friday fun!

Topics

- Seminar 1
  - Basic C++ - data types, variables, declarations, expressions, statements, etc.
  - initializers, arrays, type conversions, memory management, command-line arguments
- Seminar 2: Single class design and operator overloading
- Seminar 3
  - Derived classes, inheritance, polymorphism, RTTI
  - Exception handling
- Seminar 4-5
  - Templates (main focus)
  - Namespaces
- Seminar 6-?: The Standard Template Library (STL)

1 What is TDDD38
Syllabus
Organization
Exam
Literature
Plan
2 What is C++?
History
Design rules
Design support rules
Design support rules
Design support rules
Coding style

LINKÖPING UNIVERSITY

What is C++?

Use of C++

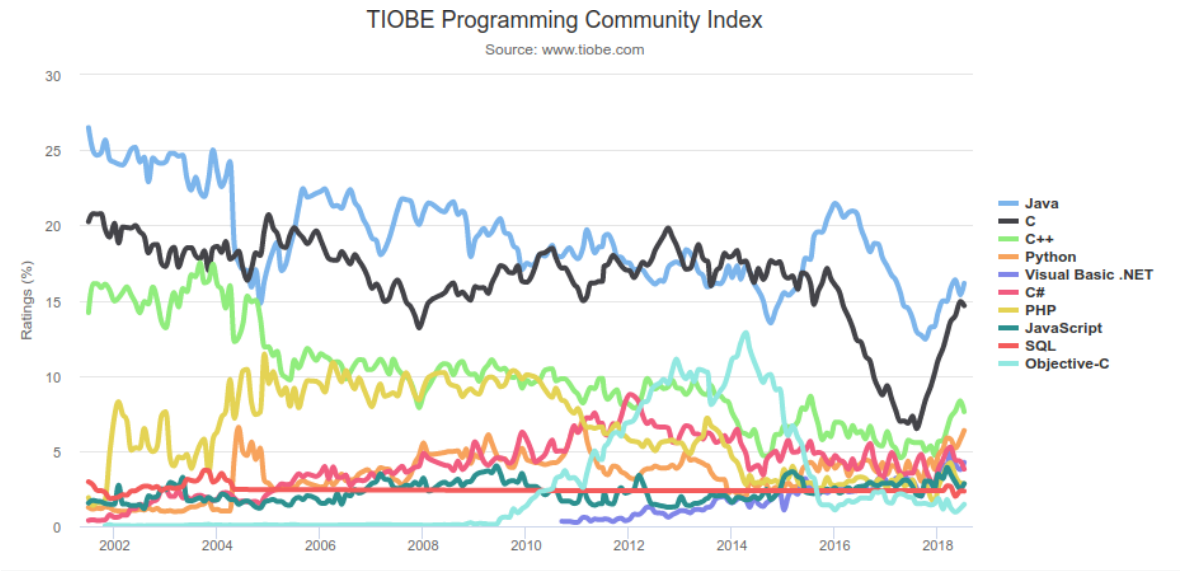

Figure : TIOBE index - http://www.tiobe.com

What is C++?

Goal of C++

"C++ was designed to provide Simula's facilities for program organization together with C's efficiency and flexibility for systems programming. It was intended to deliver that to real projects within half a year of the idea. It succeeded."
- Bjarne Stroustrup

What is C++?

Definition

> *C++ is a general purpose programming language based on the C programming language [...]. In addition to the facilities provided by C, C++ provides additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities*

ISO Draft N4687 (C++17), §1/2

History

1979	First implementation of "C with classes"
1983	Rename to C++
1985	The C++ Programming Language first edition, first external release
1990	The Annotated C++ Reference Manual (ARM)
1998	First ISO standard (C++98)!
2003	Small amendments (C++03)
2011	C++11 standard! - The "new" C++
2014	December release of C++14
2017	Latest release (C++17)
2020?	Next version

Language Design Rules

General rules

- C++ must be useful *now*
  - C++ must be useful to someone with average skills, using an average computer
- Provide comprehensive support for each supported style
- Don't try to force people

Language Design Rules

General rules

- C++ must be useful *now*
- Provide comprehensive support for each supported style
  - Features must be designed to be used in combination
- Don't try to force people

Language Design Rules

General rules

- C++ must be useful *now*
- Provide comprehensive support for each supported style
- Don't try to force people
  - Programmers are smart people and need all the help they can get from a programming language. Trying to seriously constrain programmers will fail

Language Design Rules

- All features must be affordable
  - A feature is only added to C++ when there is no other way of achieving similar functionality at significantly lesser cost.
- It is more important to allow a useful feature than to prevent every misuse
- Support composition of software from separately developed parts

Language Design Rules

- All features must be affordable
- It is more important to allow a useful feature than to prevent every misuse
  - You can write bad programs in any language. Try to minimize accidental misuse but we can't prevent a determined programmer from breaking the system.
- Support composition of software from separately developed parts

Language Design Rules

- All features must be affordable
- It is more important to allow a useful feature than to prevent every misuse
- Support composition of software from separately developed parts
  - As applications get larger and more complex, they must be composed of semi-independent parts to be manageable.

Language Design Rules

Language-Technical Rules

- No implicit violations of the static type system
  - C++ inherits stuff from C which make it impossible to detect every type violation at compile time, but wherever possible checking is done at compile time.
- Provide as good support for user-defined types as for built-in types
- Locality is good
- If in doubt, pick the variant of a feature that is easiest to teach

Language Design Rules

Language-Technical Rules

- No implicit violations of the static type system
- Provide as good support for user-defined types as for built-in types
  - User-defined types are intended to be central to C++ programs, they need as good support as possible from the language.
- Locality is good
- If in doubt, pick the variant of a feature that is easiest to teach

Language Design Rules

Low-Level Programming Support Rules

- Leave no lower-level language below C++ (except assembler)
- What you don't use, you don't pay for (zero-overhead principle)

Language Design Rules

Low-Level Programming Support Rules

- Leave no lower-level language below C++ (except assembler)
- What you don't use, you don't pay for (zero-overhead principle)
  - Large languages have a well-earned reputation for generating large and slow code; let's store cleanup data in all objects or force indirect access to all data. Having this in C++ would leave room for languages at a lower level.
    This rule is still often used to reject new features in the language

What is C++?

Prgramming style?

- There are (sadly) no real proposed styles for formatting
- I will of course use a style, but will not force you to follow that one
- A good (non-format) style guide is the Cpp Core Guidelines
- The course will focus on modern C++ (that is new stuff from C++11/14/17) and penalize usage of the "C parts" (even if they are technically part of C++)

What is C++?

Two very important terms

Implementation defined A program with implementation defined behavior is valid C++ but the results may differ between systems or even compilers on the same system

Undefined Behavior (UB) A program with UB is invalid and a standards-complient compiler can do anthing it wants.

www.liu.se

LINKÖPING UNIVERSITY