

STL Algorithms - Principles and Practice

"Prefer algorithm calls to hand-written loops." - Scott Meyers, "Effective STL"

Victor Ciura - Technical Lead, Advanced Installer

Gabriel Diaconița - Senior Software Developer, Advanced Installer

<http://www.advancedinstaller.com>

CAPHYON

Durata: **120 minute**

Dificultate: **medie**

Stil: **interactiv** (prezentare de exemple pe care să le analizăm prin dialog Q&A cu studenții)

Mediu: **slides, live code demos**

Prerequisites: **C++14** standard compliant compiler toolset (VS2015, Clang, GCC) *** detalii in anexa 1.

Despre conținutul tehnic:

Am colectat și pregătit exemple utile și interesante de aplicare a algoritmilor STL în practica noastră, din codul **Advanced Installer**. Am încercat pe cât posibil să adaptăm exemplele extrase pentru a putea fi ușor analizate *izolat* din contextul lor de aplicare.

Ne vom concentra pe exemple ce reprezintă pattern-uri frecvent necesare (utile) nouă. De asemenea, vom prezenta și câteva situații despre care am constatat, de-a lungul timpului, că generează probleme și confuzie pentru programatorii neexperimentați, precum și greșeli comune pe care le putem face atunci când nu suntem atenți.

Plan de desfășurare:

Începem printr-o scurtă introducere teoretică a algoritmilor din STL. Nu vom insista pe o prezentare detaliată a fundamentelor teoretice necesare, ci în a explica **de ce** este important să cunoaștem instrumentele puse la dispoziția noastră de biblioteca STL de algoritmi și cu ce ne ajută în practica reală de zi cu zi, în afara exemplilor canonice / didactice.

Studenții vor putea să descopere, de asemenea, că stăpânirea algoritmilor și conceptelor STL, în general, îi va ajuta să economisească timp prețios în elaborarea și testarea soluțiilor pentru probleme de genul **competitive programming**.

Restul timpului alocat va fi folosit exclusiv pentru **exemple** de aplicare a algoritmilor și discuții pe marginea acestora. Bucățile de cod vor fi **scurte** (câteva linii care să încapă pe un slide) și bine izolate astfel încât să nu necesite informație suplimentară despre contextul de aplicare în scenariul real.

Pentru fiecare exemplu de cod al prezentării, vom încerca să răspundem la următoarea întrebare, din perspectiva punctelor de mai jos:

Why prefer to reuse (STL) algorithms?

- *Correctness*
- *Code Clarity*
- *Performance*
- *Modern C++ (C++11 / C++14 standards)*

Dorim ca prezentarea să aibă un **stil interactiv**, să întreținem un dialog cu studenții, adresând întrebări legate de fiecare exemplu prezentat. Întrebările nu vor fi în stil quizz, ci mai degrabă de **opinie**, de genul:

“Ce credeți că face codul de pe slide?”

“Ce potențiale probleme are codul acesta?”

“Cum putem îmbunătăți / corecta / simplifica codul?”

Încurajăm studenții să pună întrebări legate de exemplele prezentate, atunci când consideră că un concept a fost neclar sau dacă nu înțeleg anumite părți din cod.

Exemplele vor acoperi, neexclusiv, următoarele **concepte** STL, pe care le considerăm fundamentale (ordinea nu este relevantă):

- the benefits of using lambda functions with standard algorithms
- leveraging standard function objects (bind, less, equal_to, not1, greater, negate, ...)
- using iterator adaptors (inserter, back_inserter, front_inserter, etc)
- std::transform (higher-order function, usage as a type convertor, etc)
- std::copy (using source/destination ranges over different collection types)
- std::find[_if] (complexity, generic algorithm vs. container member function)
- std::sort (complexity, std::list vs. std::vector, stable sorting)
- binary search
- handling unique elements in a collection
- set operations (union, difference, intersection)
- removal algorithms (erase/remove idiom)
- customized algorithms (extend STL algorithms with new interfaces, convenience wrappers, iterator ranges)
- exposing STL-like iterator ranges from your custom collection classes for use with std algorithms

ANEXA 1 - Prerequisites

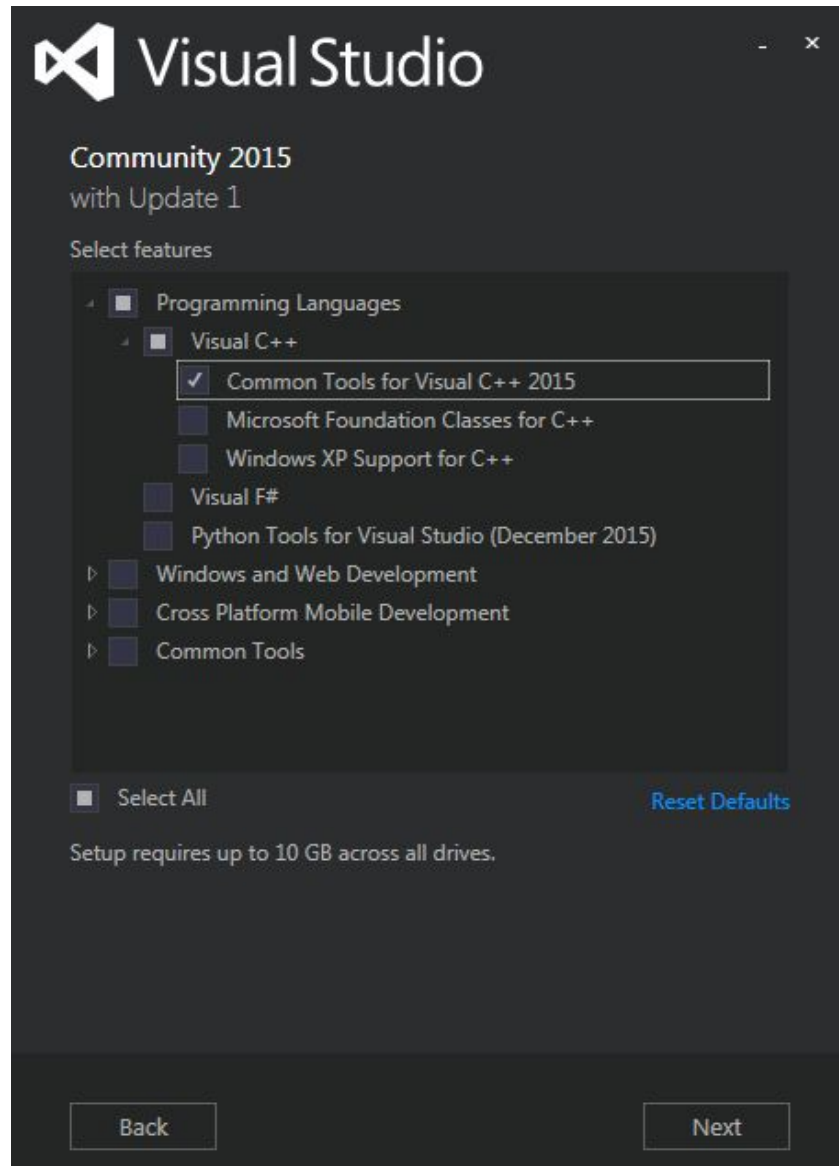
Datorita limitarilor tehnice (PC software, Internet access) ale salii de laborator unde se va desfasura acest workshop, este necesar ca fiecare student sa vina cu propriul **laptop** pe care sa instaleze, in avans unul dintre urmatoarele **C++14** compiler toolsets:

Visual Studio 2015 Community Edition (FREE for non-commercial use).

Download link: <https://www.visualstudio.com/products/visual-studio-community-vs>

Custom install (**minimum**):

[Programming Languages] >> [Visual C++] >> [Common Tools for Visual C++ 2015]



LLVM/Clang 3.7 (minimum version 3.4 required)

Download link: <http://llvm.org/releases/download.html>

GCC 5.3 (minimum version 4.8 required)

Download link: <https://gcc.gnu.org/>