Advanced Programming Concepts with C++ CSI2372 – Fall 2018

Jochen Lang
EECS, University of Ottawa
Canada

uOttawa

L'Université canadienne
Canada's university

Université d'Ottawa | University of Ottawa

uOttawa.ca

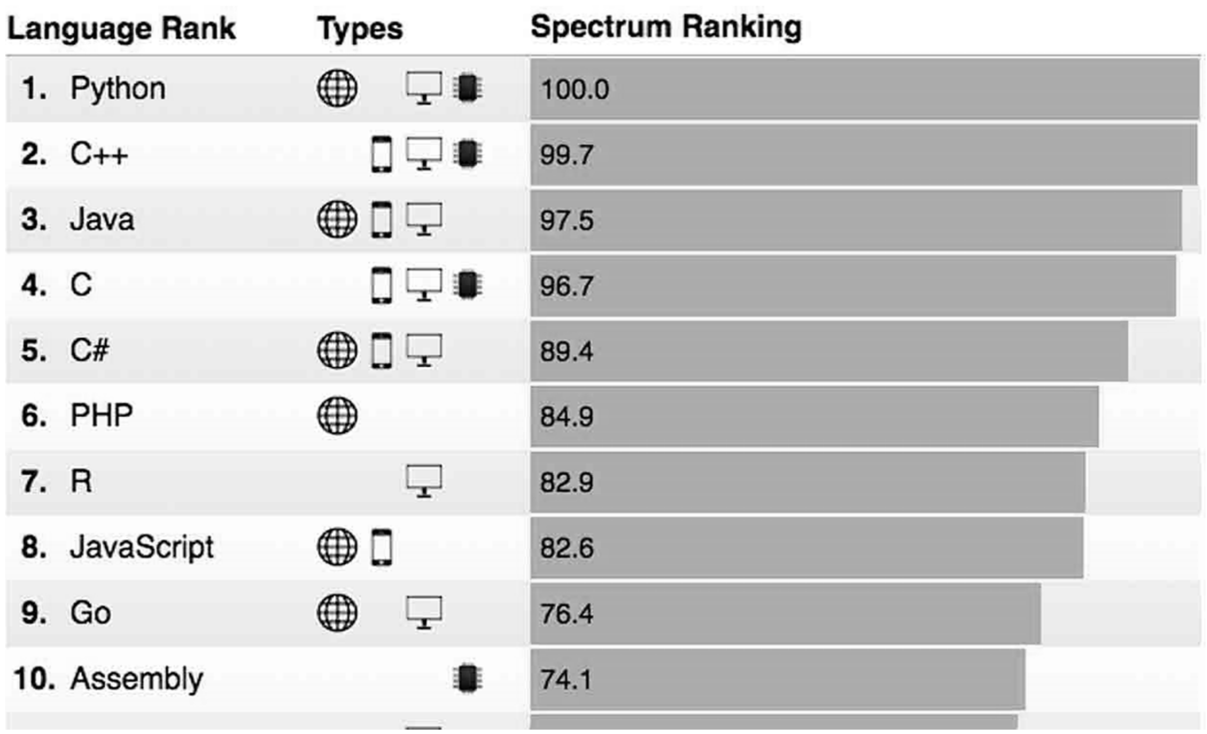
C++

- **Object-oriented programming language**
  - Data abstraction (class concepts)
  - Operator overloading
- **C/C++ (and Objective C) together are (still) the de-facto standard (except for webcentric applications)**
- **Combines a high-level language with low-level features**
  - C++ is a *superset* of C
  - C is a functional programming language

Brief History of C/C++

1967-1980	Development of Unix by Ken Thompson, Denis Ritchie and others at Bell Labs
1969-1973	C by Denis Ritchie, Bell Labs Based on B written by Ken Thompson, most of Unix written in C
1984	C++ by Bjarne Stroustroup, Bell Labs Object oriented programming constructs were added to C
1998	C++ Standard ISO/IEC 14882 and revised in 2003 as ISO/IEC 14882:2003
2011	C++ Standard C++11 “C++0x”, ISO/IEC 14882:2011
2014	C++ Standard ISO/IEC 14882:2014
2017	C++ Standard ISO/IEC 14882:2017
2020	Next scheduled release

Popularity of Programming Languages

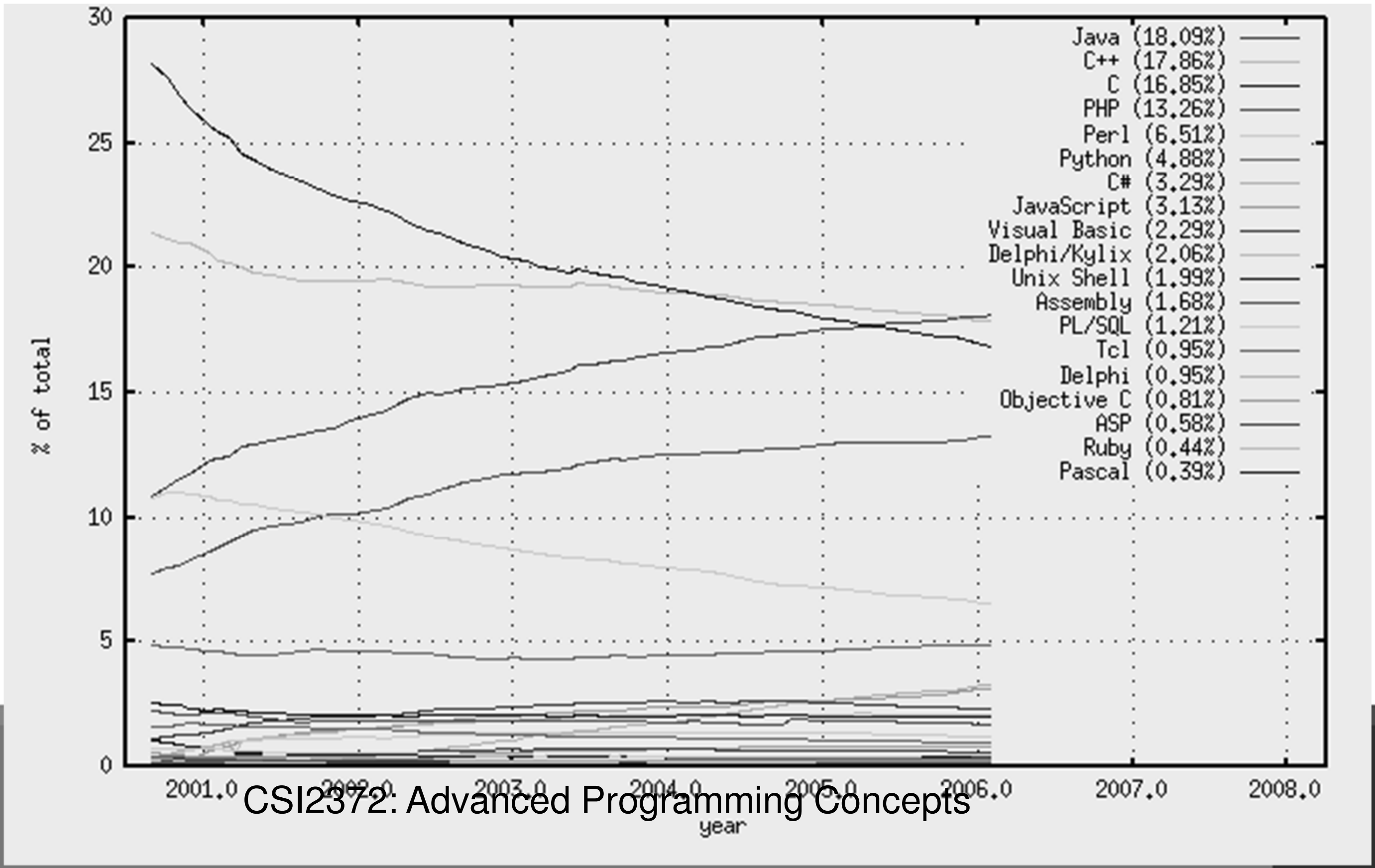


Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥 ▣	100.0
2. C++	📱 🖥 ▣	99.7
3. Java	🌐 📱 🖥	97.5
4. C	📱 🖥 ▣	96.7
5. C#	🌐 📱 🖥	89.4
6. PHP	🌐	84.9
7. R	🖥	82.9
8. JavaScript	🌐 📱	82.6
9. Go	🌐 🖥	76.4
10. Assembly	▣	74.1

“The 2018 Top Programming Languages” IEEE Spectrum ranking [accessed Sep. 1, 2018]. Based on web searches, specific web pages, IEEE digital library etc.

Use of Programming Languages at the Beginning of the Century

- **François Labelle, Programming Language Usage Graph**
  - https://wismuth.com/lang/languages.html
  - Statistics based on open source projects at SourceForge



Is C++ in decline?

- **Bjarne Stroustrup:**
  - "*No, I don't think so. C++ use appears to be declining in some areas and to be on an upswing in others. If I had to guess, I'd suspect a net decrease sometime during 2002-2004 and a net increase in 2005-2007 and again in 2010-2011, but I doubt anyone really knows. Most of the popular measures basically measures noise and ought to report their findings in decibel rather than "popularity." A professional survey in 2015 estimated the number of C++ programmers to be 4.4 million*."
  - See the Tiobe index at https://www.tiobe.com/tiobe-index/ – a very popular measure
  - *"There are more useful systems developed in languages deemed awful than in languages praised for being beautiful-- many more"*

Bjarne Stroustrup's FAQ: Did you really say that?. Retrieved on 2017-09-03.

Benefits of Learning C++

- Low-level control over many features including memory management, and, the breadth of C++
  - Improves understanding of software design
  - Helps to make informed choices about design
  - Bjarne Stroustrup: *"To use C++ well, you have to understand design and programming technique"* Bjarne Stroustrup's FAQ: Did you really say that?. Retrieved on 2017-09-03.
- Wide use and popularity of C/C++
  - Increases employment prospects
  - Helps to communicate with expert developers
  - Helps to evaluate and adapt projects by others

CSI2372

- **Prerequisite**
  - ITI1121 and ITI1100
    - You should be able to program in Java
    - You can create, compile and run a Java program
    - You are familiar with the basics of oo design
- **Programming Practice**
  - Current environment in the lab
    - Microsoft Visual Studio 2017
  - Code can be compiled with any C++ compiler, e.g., GNU gcc cygwin on windows or gcc in linux

Resources

- **Classic Texts**
  - B.W. Kernighan, D.M. Ritchie, The C Programming Language, 2nd ed, Prentice Hall, 1988.
  - Bjarne Stroustrup, The C++ Programming Language, 4th ed, Addison-Wesley, 2013.
- **On-line Resources**
  - See the course webpage.

Textbooks

- **Required:**
  - S.B. Lippman, J. Lajoie and B.E. Moo, C++ Primer, 5th ed., Addison-Wesley, 2012.
    - The textbook is available at the AGORA bookstore $70.49 + tax.
    - Programming examples available on website.
- **Recommended:**
  - Bjarne Stroustrup, Programming: Principles and Practice Using C++, 2nd ed., Addison-Wesley, 2014.
  - W. Savitch, Problem Solving with C++, Addison-Wesley, 10th ed., 2017.
  - P.J. Deitel and H.M. Deitel, C++ How To Program, 10th ed., Pearson Education, 2016.

Syllabus – 10 Topics

Java in C++

Basic Object oriented C++	Textbook (Lippmann): Chapters 1.1-1.6, 2.1-2.3, 3.1-3.3, 4.1-4.6, 5.1-5.5
Included	+ Java, C and C++ + Basic structure of a C/C++ program + Fundamental and complex data types including classes and strings + Operators for fundamental types + Control and decision statements + Basic use of std::vector, std::array and std::string + Console input/output + Class attributes and methods
Not included	- no inheritance - no pointers - no arrays, no unions, no structures - no dynamic memory management - no bitwise operators

Syllabus – 10 Topics

C-like C++

Data types and memory management	Textbook (Lippmann): Chapters 2.4-2.6, 3.5, 3.6, 4.11, 6.1, 6.2-6.4, 7.2, 12
Included	+ Scope, modifiers, type conversions + Automatic type derivation and conversions + Pointers and arrays + Memory allocation: static, automatic and dynamic + Allocation and de-allocation, C++ vs. C + Pass by value, by reference, by pointer
Not included	- No bit manipulation - No machine architecture specifics

Syllabus – 10 Topics

OO

Object-oriented design	Textbook (Lippman): Chapters 7.1-7.5, 13.1, 13.2, 13.4, 15.1-15.5, 15.7, 18.1
Included	+ Method overloading, default parameters + Class construction and constructor types, destruction + Class relationships: association, aggregation, generalization and inheritance + Pointer attributes + Copy construction and assignment + Polymorphism: Virtual functions, abstract classes and dynamic cast + Exceptions Basics + Inline functions, static members, constexpr
Not included	- No templates yet - No STL yet - No operators yet - No callables yet

Syllabus – 10 Topics

Text is beautiful

Input and output streams	Textbook (Lippman): Chapters 8.1-8.3, 14.2
Included	+ Relevant classes for STL Stream I/O + File handling + Overloading the insertion and extraction operators + String streams
Not included	- No internationalization - No unicode

Syllabus – 10 Topics

Just like int

Abstract data types	Textbook (Lippman): Chapters 14.1-14.4, 14.7, 14.9
Included	+ Operator overloading + Numerical vector and matrix classes in C++ + Friend operator on classes and functions
Not included	

Do more with less

Macros and Templates	Textbook (Lippman): Chapters 2.9.2, 6.14, 16.1-16.3, 16.5
Included	+ Macros and the C++ preprocessor: debugging, conditional compilation + Templates: template functions and classes + Templates: type and non-type parameters + Template specialization
Not included	- No variadic templates - Not much on macros

Syllabus – 10 Topics

Write even less code

Callable Objects	Textbook (Lippman): Chapters 6.7, 10.3
Included	+ Passing a function: function pointers, functors + C++11 bind + Lambdas
Not included	

No reinventing the wheel

Standard Template Library	Textbook (Lippman): Chapters 10.1-10.2, 10.4, 11.1-11.4
Included	+ Review: Java Collections Framework + Sequential STL containers and container adaptors + STL iterators + Associative STL containers + Generic algorithms

Syllabus – 10 Topics

No more Memory leaks

Smart pointers and data management	Textbook (Lippman): Chapters 12, 12.1-12.2
Included	+ Smart pointers + C++11 smart pointer library types + Move constructor and move assignments
Not included	

A Million other Things

Miscellaneous	Textbook (Lippman): Chapters 18.1, 18.3
Included	+ Multiple inheritance + Virtual inheritance and abstract classes + Exception handling + Interfacing Java and C++
Not included	

Student Participation

- **Mandatory attendance of and participation in lectures, labs and tutorials**
- **Group work and interactive feedback**
  - Using web clicker.
  - You must register for an account (details on Virtual Campus)
  - No marks but there to help with learning and retention
  - Also used to determine if you complied with the Faculty of Engineering rule of minimum attendance of 80% of lectures.
- **First discussion in groups of 4**
  - You are asked develop a financial accounting software. Your client's servers run Linux and the desktop's Windows. What programming language do you choose and why?

Evaluation

- **In all exam: One textbook is allowed (and no other material)!**
- **Midterm A 12 %**
  - Wednesday, October 3rd, 2018, 14:30-16:00
- **Midterm B 12 %**
  - Wednesday, November 7th, 2018, 14:30-16:00
- **Final exam 38 %**
  - Final exam will not overwrite the midterm!
- **Lab programming assignments 12 %**
  - 4 short **individual** assignments
- **Programming project 26 %**
  - Groups of two

Projects and labs will be discussed in the lab, the project will start at the end of October after the lab assignments.

Projects and assignments will have to be submitted via Virtual Campus, exclusively. No other form of submission will be accepted.

Important: If the student's mark in the exam component is less than 50%, i.e., (Midterms + Final) < 31, then the student's mark in the course will be (Midterm + Final) / 62

Reminder: Academic Regulations

- **Mandatory Attendance**
  - As per academic regulations, students who do not attend 80% of the class will not be allowed to write the final examinations.
  - All components of the course (i.e., laboratory assignments, projects, etc.) must be fulfilled otherwise students may receive an INC as a final mark (equivalent to an F). This also holds for a student who is taking the course for the second time.
- **Academic Fraud and Plagiarism**
  - Any form of plagiarism or fraud including on an assignment or the project, will be reported.
  - For any plagiarism or fraud the university regulation on academic fraud applies.
  - https://www.uottawa.ca/vice-president-academic/academic-regulations-explained/academic-fraud explains the University of Ottawa rules. Please familiarize yourself with them.

Contact

- **Office Hours:**
  - Wednesdays at 13:00 – 14:00, STE-5098
  - Virtual Campus Discussion Groups (preferred over e-mail)
  - E-mail: jlang@uottawa.ca
- **TAs (more info on webpage)**
  - Laboratory (3 groups – must stay in your group, starting next week)
    - Shuang Xie (lab 1), Ertuğrul Kara (lab 2), Alireza Parvizimosaed (lab 3)
  - Tutorial (starting Friday, Sep 14) taught by Ahmedou Jreivine
  - Contact hours to be set by TAs in first lab/tutorial

Student Expectation

- **Attend lectures, labs and tutorials**
  - Download examples, compile and run them, try changing and improving them
    - Post improvements on virtual campus!
  - Take notes
- **Get a textbook and be ready to research a question**
- **Seek help early**
  - Rule of thumb: If you are not getting 3 marks in a lab, you will need to review the material.
- **In summary: participate and practice, practice, practice**

A First Look at C/C++

- **Java syntax is based on C**
- **Execution of C/C++ starts with main**
- **System functions are not grouped in a class**
- **C++ has the concept of a namespace**

- **Example**
  - Hello World in Java and C

Hello World

```java
/* Hello World in Java */
public class HelloWorld {

  static public void main( String args[] ) {
    System.out.println( "Hello World!");
    return;
  }
}
```

```cpp
#include <iostream>

/* Hello World in C++ */
int main() {
  std::cout << "Hello World!" << std::endl;
  return 0;
}
```

Standard Input and Output

- **Output stream cout**

```
std::cout << myVar;
```

  - Object-oriented printing to console
  - Built-in types can be printed using the left-shift operator
  - Similar than System.out.print in Java but more flexible (stream modifiers; more later)
- **Input stream cin**

```
std::cin >> myVar;
```

  - Object-oriented input from console
  - Built-in types can be converted and assigned with the right-shift operator

Using Definitions of the Standard Namespace

- **iostream library necessary for console input and output.**
- **Declarations are in the namespace std (standard).**
  - Using a single declaration:
    - just once

```
std::cout
```

    - in the whole scope

```
using std::cout;
```

  - Using all the declaration within a namespace in a scope (avoid!)

```
using namespace std;
```

Main Function

- **C/C++ program entry point main which is of type**

```
int main( void );
int main( int argc, char *argv[] );
```

- **All source files in a project are allowed to define only one main function.**
  - Note: Visual Studio defines additionally program entry points (other “main” functions). Standard compliant C++ code will only use the above.

Java and C++

- **Java**
  - Compiled to byte code
  - Executed by virtual machine
    - Object-oriented
    - Platform-independent byte code
- **C++**
  - Preprocessor
  - Compiled to object code
  - Linked to binary executable
    - Object-oriented, generic and functional features
    - Object code and executable are platform-specific

C++ Fundamentals

- Fundamental and complex data types including classes and strings
- Operators for fundamental types
- Control and decision statements

Variable and Function Names

```
identifier :
   underscore
   letter
   identifier following-character


following-character :
   letter
   underscore
   digit

letter : one of
   A B ... Z a b ... z
digit : one of
   0 1 2 ...  9
underscore : _
```

Exactly like in Java

Case sensitive!

Examples:

`i5`

`__do_not_use__`

`butUseThis`

`myFavoriteVariable`

Declarations

- **Declarations introduce names into a program. Declarations may occur in different places in a program.**
- **What to declare?**
  - variables
  - functions
  - classes, structures and union components
  - types
  - type tags
  - enumeration constants
  - namespace
  - statement labels
  - preprocessor macros

Definition vs. Declaration

- **Java and C++ provide definitions in one file and use it in many files**
- **Java**
  - Name is imported into another file.
- **C++ (Each file is compiled separately – if not #include'd)**
  - Linker ensures that name (according to scoping rules) refers to the same entity everywhere.
    - Definition allocates a variable.
    - Declaration introduces only the name.

Fundamental Data Types

- **Three categories integral, floating and void.**
- **integral**
  - `bool, char, short, int, long, long long` (in C++11)
    - `intN_t` with N = 8,16,32 or 64 (only C99);
    - MSVC: _`intN` with N = 8,16,32 or 64
- **floating**
  - `float, double, long double`
- **void**

… close to Java

BUT size may vary with C++ compiler/OS

Standard defines minimum sizes

Type Modifiers and Size

- **Modifiers**
  - `unsigned, signed, short, long`
- **Sizes in MSVC++**
  - 1 byte

    `bool, char, unsigned char, signed char`
  - 2 bytes

    `short, unsigned short`
  - 4 bytes

    `int, unsigned int, long, unsigned long, float`
  - 8 bytes

    `double, long long`
  - 18 bytes

    `long double`

Derived Data Types

- **Directly derived data types**
  - *Arrays, functions, pointers, object references, constants*

To be defined later !

- **Composed derivative types**
  - classes, *structures, unions, scoped enumerations*

```
class myClass
{
…
};
```

Automatic Typing with `auto`

- **Most often initialization can be done better (less error prone) by using auto types.**

```
auto iVal=65;
auto oiVal=iVal;
auto fVal=3.0f;
auto ofVal=fVal;
```

- **Aside: Arithmetic literals**

```
1 is an int
1U is an unsigned int
1L is a long
1LL is a long long
1.0f is a float
1.0 is a double
'\1' is a char.
```

Next week:

Java in C++

- **Basic Object-oriented C++**
  - Strongly-typed Enumerations
  - Operators, Ch. 4.1-4.9
  - Selection and Iteration Statements, Ch. 1.4, 5.3-5.5
  - Static casts, Ch. 4.11.3-5.12.6
  - Overview of std::string
  - Introduction to std::array and std::vector