Mathematical Software Programming (02635)

Module 1 — Fall 2016

Instructor: Martin S. Andersen

Practical information

Format

- 5 ECTS (1 ECTS $\sim$ 28 hours on average)
- Short lectures (**B306-A031**)
- Focus on exercises (**B306-H000vest**, **B306-H001øst**)
- Weekly reading assignments (see Calendar on CampusNet)
- Two written hand-in assignments (more info later)
- Final exam (written)

Instructors

- Martin S. Andersen (`mskan`), DTU Compute
- Bernd Dammann (`beda`), DTU Compute/DCC

Teaching assistant

- Mathias Sorgenfri Lorenz (`s134597`)
- David Frich Hansen (`s144242`)

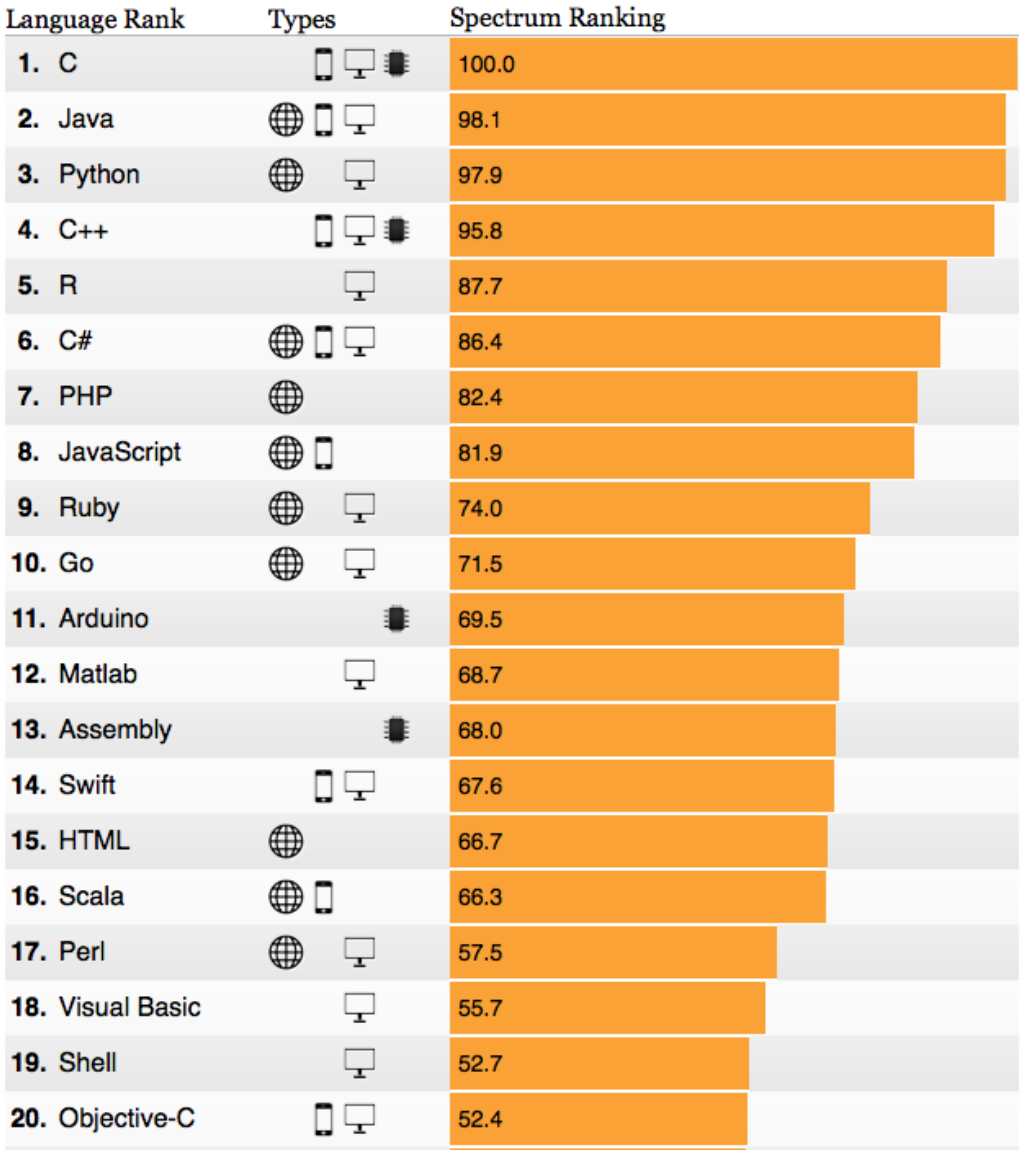
Learning objectives

- Evaluate discrete and continuous mathematical expressions.
- Describe and use data structures such as arrays, linked lists, stacks, and queues.
- Choose appropriate data types and data structures for a given problem.
- Compare iterative and recursive solutions for simple problems.
- Analyze the runtime behaviour and the time and space complexity of simple programs.
- Call external (third party) programs and libraries.
- Design, implement, and document a program that solves a mathematical problem.
- Debug and test mathematical software.
- Describe and use basic object-oriented programming concepts such as classes and objects.

Why C?

- Widely used and mature programming language (developed in the early 1970s)
- Industry standard (ANSI C (C89) / ISO C (C90), C95, C99, C11)
- Many newer programming languages (C++, C#, Objective C, Java, PHP, Go, ...) are syntatically similar to C
- Cross-platform support
- Low-level control (direct access to low level hardware/APIs)
- Low overhead (high performance)
- Statically typed language
- Understanding of memory management (no "magic" under the hood)
- Embedded systems (IoT)
- C *powers* the world (OS kernels, Python, MATLAB, ...)

IEEE: The Top Programming Languages 2016



Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		97.9
4. C++		95.8
5. R		87.7
6. C#		86.4
7. PHP		82.4
8. JavaScript		81.9
9. Ruby		74.0
10. Go		71.5
11. Arduino		69.5
12. Matlab		68.7
13. Assembly		68.0
14. Swift		67.6
15. HTML		66.7
16. Scala		66.3
17. Perl		57.5
18. Visual Basic		55.7
19. Shell		52.7
20. Objective-C		52.4

Resources

Textbooks

- S. Oliveira & D. Stewart, "Writing Scientific Software: A Guide to Good Style", 2006
  - ISBN: 9780521675956
- I. Horton, "Beginning C", 5th ed., 2013
  - ISBN: 9781430248811
  - Ebook available through DTU Library

Supplementary resources (optional)

- I. Horton, Beginning C++, 2014
- M. Olsson, C quick syntax reference, 2015
- M. Olsson, C++ quick syntax reference, 2013
- OnlineProgrammingBooks.com
- Big-O Cheat Sheet
- Learn to Solve It: C programming exercises

Help!?

Instructors/teaching assistants

- Be prepared
- Write down questions
- Get feedback

Piazza

- Post your (anonymous) questions on *Piazza* discussion board
- Learn from and help your peers

Email

- Please use email for personal matters only

Documentation and reference manuals

- GNU C Library
- GNU C Library - function index
- GNU Compiler Collection (GCC) Manual
- Wikipedia: C mathematical functions
- GNU Scientific Library
- Cplusplus.com
- Cprogramming.com
- Boost C++ Library

Compilers

- Linux/Unix
  - `gcc` (Ubuntu/Debian: `sudo apt-get install build-essential`)
  - `clang` (`sudo apt-get install clang`)
- Mac OS X
  - `clang` (`xcode-select --install`)
  - `gcc` (e.g., via Homebrew)
- Windows
  - `gcc` available in TDM-GCC (recommended)
  - Pelles C (mentioned in "Beginning C")
  - C/C++ compiler in *Microsoft Visual Studio 2015* (available via CampusNet/Software)

Software

Cross-platform editors & IDEs

- Atom
- GNU Emacs
- Vim
- Eclipse
- Code::Blocks

Tools

- GNU Make
- GNU plot
- GNU debugger
- GNU profiler
- Valgrind profiler

DTU Resources

- gBar DataBar
- High-Performance Computing
- DTU Computing Center

Todays exercises

Available under “File sharing” on CampusNet

- Part I: Install a C compiler and a text editor or an IDE
- Part II: Do exercises (individually or in small groups)

If you finish early, start preparing for next week!

Compile and run “Hello World!” program

Create a plain text file `main.c` with the following code:

```c
#include <stdio.h>

int main(void) {
    printf("Hello World!\n");
    return 0;
}
```

Compile and run your program:

```
$ cc main.c -o hello
$ ./hello
```

Using the Atom editor

Installation

- Install Atom (provides two commands: `atom` and `apm`)
- Open Atom and install the GCC Make Run package, or using the `apm` command-line tool:
  `$ apm install gcc-make-run`
- Set compiler/options using `ctrl-f6` or `cmd-f6`
- Compile and run your program with `f6`

Useful packages

- linter (flag suspicious code): `linter-gcc`, `linter-clang`
- auto-indentation: `atom-beautify`
- auto-complete (clang-users): `autocomplete-clang`
- highlight current selection: `highlight-selected`
- source code preview: `minimap`