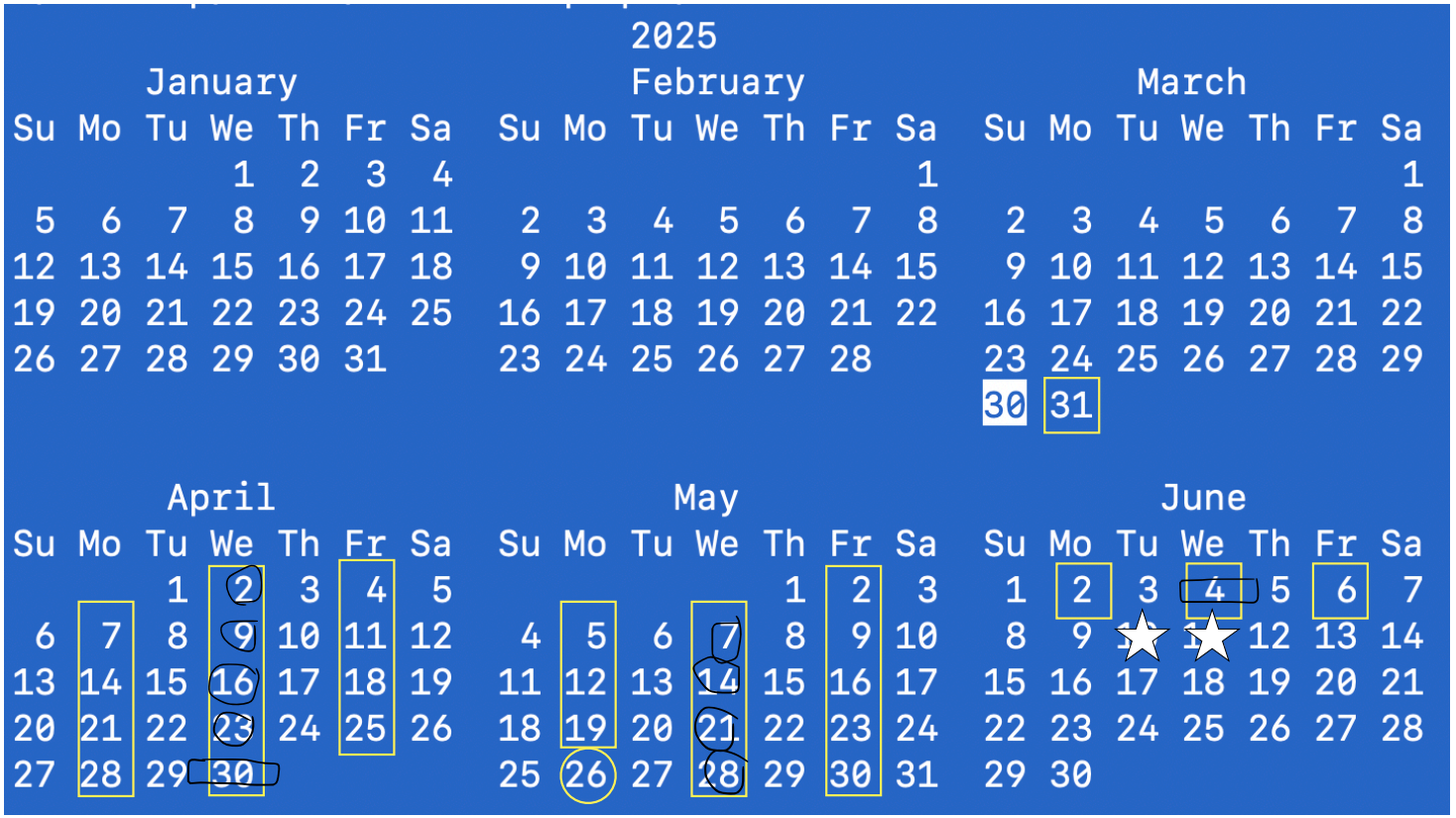
lecture 1

- 29 lectures, ~9 graded events (6 HWs, 3Exams) or (7 HWs, 2Exams) or etc.
- Monday May 26 is a holiday
- 483 - Final Tuesday June 10
- 583 - Final Wednesday June 11

- Title IX (pronouns matter)
- Syllabus / Course Content
- Grading and fairness
- Canvas Page
- Piazza Page
- Course Evaluation (let's make a deal!)
- Visual Studio Code - common framework



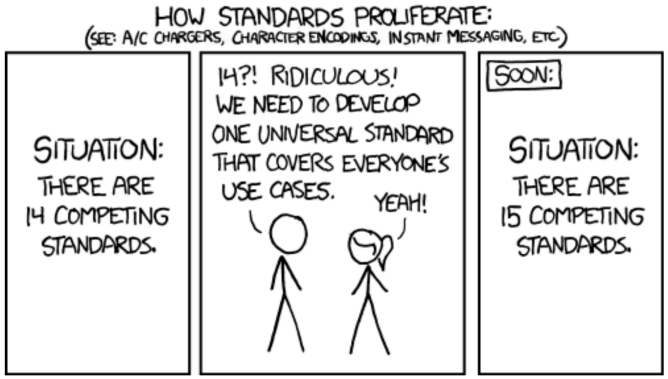
Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

- Grading and fairness
  - don’t care if you use online resources (fyi - AI often makes errors coding my assignments)
  - grades will be determined by total aggregated points earned / total available
    - 1 HW grade drop will be permitted provided > 80% of enrolled students fill out teacher evaluation
    - final grades are normally curved since I don’t want anyone to fail

- Canvas Page - I will post this by lecture 2

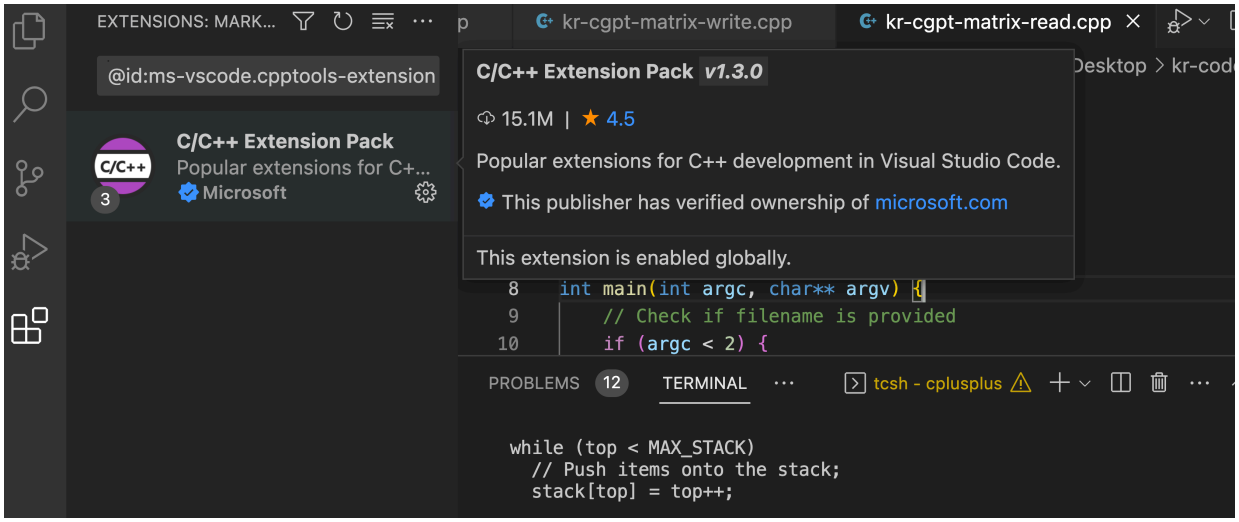
- Piazza Pages:
  - Find AM483 class signup link at: https://piazza.com/washington/spring2025/amath483
  - Find AM583 class signup link at: https://piazza.com/washington/spring2025/amath583

- Visual Studio Code - common framework



download and install Visual Studio Code:
https://code.visualstudio.com/

I will do a short demo in class on ***creating, editing, compiling, and executing*** a C++ code. First touch will require the C/C++ extension pack.
https://code.visualstudio.com/docs/languages/cpp



Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

- Course Content

*Overview of HPC, concepts in (parallel) computing, Top500 and Graph500, Moore's law and Dennard scaling, CPUs and memory, basic code optimization strategies, concurrency and accelerators

*BLAS complexity, forward and backward numerical errors in finite precision, IEEE floating point standard, other finite bit precision issues

*C++ introduction (scope resolution, operators, data types, pointers static and dynamic allocation, references, arrays, vector container class, namespaces), compiling basics

*C++ (namespaces -continued, conditionals, loops (including iterators in containers), functions (parameters by reference and values))

*C++ functions continued, classes (structure, members, methods and access levels), header files and compiling revisited, metric spaces, Cauchy sequence, completeness, vector spaces, normed space, Banach space, metrics induced from norms, equivalent norms, inner product space, Hilbert space, various identities, matrix representations / index mappings for computing - column major vs row major

C++ function overloading, template functions, class constructors, class destructors, constructor overloading and default initialization, constructor initializer, class copy constructor -shallow and deep copy, inheritance -upcasting and downcasting, multiple inheritance, inheritance overriding, poor man's matrix class and operator overloading (,+/-)

*C++ variable scope, header files (avoiding duplicate symbol / multiple definition), include guards, compiling and using relocatable shared object libraries, LD_LIBRARY_PATH environment variable, matrix multiply revisited (vector of vectors), Strassen matrix multiply C=A*B and complexity, loops data dependencies (conditional branching, loop carried dependence, etc.), machine independent code transformations (subexpression elimination, copy propagation, dead code removal, constant folding, strength reduction / induction variable elimination)

*Exception handling in C++ (try and catch), Unix-like processes (memory discussion -stack, heap, BSS, data segment, text segment), C++ STL stack class, compiling / ELF (executable linkable format) and symbol tables, using objdump, LLVM and LLVM-IR tools for optimization, memory safety revisited (stack overflow, etc.), recursive functions; notes on discrete orthonormal bases on digital lattice, Big O notation, orthogonal projections (Gram Schmidt), linearized regression and the normal equations

*Discussion of various HW solutions, primer / review on linear algebra (span, linear independence, row and column and null spaces, rank, nullity), Gaussian elimination (linear systems, elementary matrices, LU solves), data analysis - best fit, C++ Input / Output introduction

*Process creation (fork, exec, execvp, wait), parent and child processes, invoking Python from C++ for plotting example, distributed memory, introduction to MPI (point to point communication, collective communication (broadcast, reduce, etc.))

*shared memory and concurrency, introduction to C++ threads, race conditions, threaded quadrature and domain decomposition, mutex, lambda expressions with thread constructors, speedup, thread coordination and condition variables, producer-consumer, MPI hello world, blocking point to point send / receive, broadcast, and reduce, define efficiency, weak and strong scaling

*C++ threads - continued, critical sections, global variables and mutex, using condition variables and lambda functions again, bounded buffer in C++ queue class, asynchronous pipeline execution, MPI - continued, gather, scatter, and (sub)communicators, working on HYAK cluster

*C++ threads - continued, detached threads, MPI - control flow in distributed memory, barriers, math notes on Gamma functions, Fourier series, Fourier transforms, Dirac delta, examples

*Distributed BLAS and parallel linear algebra: data structures and kernels

Looking at the hardware on HYAK, Intel SSE (streaming SIMD extension), AVX (advanced vector extension), AMX (advanced matrix extensions), BFLOAT16 floating point format, special instructions, tiled matrix multiply, OpenBLAS, MPI parallel Input / Output, HPC filesystems (including non-volatile memory) and factors impacting IO performance, HPC IO scenarios, MPI parallel write, MPI parallel read, comparison to sequential, asynchronous point to point send and receive, sequential and parallel (mpi4py) Python IO

*Discrete Fast Fourier Transforms using FFTW, introduction to NVIDIA GPUs and CUDA programming (device discovery, copying symbols between CPU and GPU devices, trivial vector parallelism, GPU to GPU device operations on multiple GPU systems)

*Division of labor in parallel discretizations - round robin, CUDA thread grids, blocks, and identifiers (and their relationships), thread block local shared memory, using templates in CUDA, CUDA streams (common array with offsets, same input different functions, etc.), CUDA BLAS (cuBLAS)

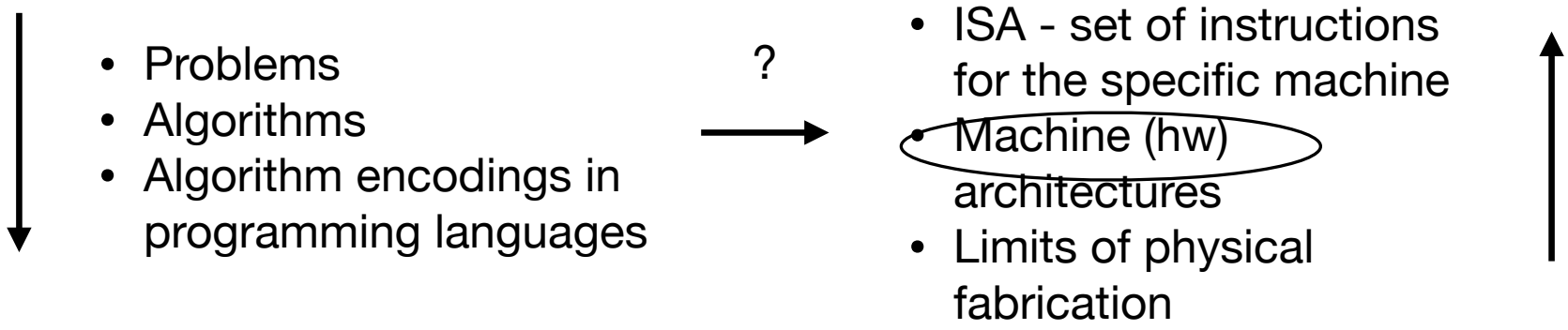
*CUDA programming model support level and architectures (sm_*), atomic operations in CUDA (add, sub, exch, min, max, inc, dec, CAS), use of __device__ - functions for kernel threads, threaded transpose, threaded image analysis (pixels to pixels)

*Python CDLL utility to invoke C, C++ code in Python script, CUDA managed memory, C++ pair (from <utility>) and list of pairs, normal statistical distribution, FFTW implementation of 3d gaussian, CUDA FFT (cuFFT) implementation of 3d gaussian

*More IO intrinsics in C++ - non-contiguous access to data in files for reading and writing, overloaded seekg() and seekp() functions, discrete Fourier transform quick note and complexity discussion

*Discuss final, summarize our quarter, Galois' Dream - Michio Kuga

We will cover these in more detail in the course but …



- What happens when operation counts for an algorithm get huge -like 10^20?
- How about when the size of a data file is larger than a terabyte? A petabyte? Or when there are hundreds or thousands of files in a data set? Etc.
- … we need tools and methods to manage such issues

(clockwise:
Michael Strayer,
Kenneth Roche,
Jack Dongarra,
Michael Guidry,
Aurel Bulgac,
Arthur Kerman

End Intro Lecture