**PTSS Slides Overview**

**week00: (summary of basics)**

* types and their representation
* expressions and operations
* statements (loops, break, continue, etc.)
* memory allocation, pointers
* pass by value, (const) reference or pointer
* casts, namespaces, default function arguments

**week01a: (intro)**

* intro, loop example, swap functions

**week01b: (version control)**

* git cheat sheet!

**week02a: (preprocessing/compiling/linking)**

* macros, includes, assert
* libraries (static or shared)

**week02b: (make)**

**week03a: (cmake)**

* creating and using a library and linking against it
* setting compiler flags and c++ standard

**week03b: (templates and generic programming)**

* generic programming process

**week03c: (classes)**

* data hiding (public, private, friend)

**week04: (more on classes, operators, function objects)**

* public, private, protected (s. week08 inheritance)
* outside of class definition (scope operator)
* constructors, operator overloading, friend, this
* static members, const member functions, mutable
* templates
* assignment, symmetric, conversion, pointer operators

**week05/06a: (more on classes, operators, function objects, templates)**

* first ~40 slides same as in week04
* overview of special member functions (ctor, dtor, copy, move)
* default, delete
* function objects (functors), lambdas
* Argument Dependent Lookup (ADL) or König lookup
* type traits, typename, type and value members
* concepts, documenting a template function

**week06b: (templates, random numbers, timing, exceptions)**

* error handling (termination, error codes/flags, exceptions)
* try, throw, catch, standard exceptions (logic/runtime errors)
* date and time utilities, Monte Carlo methods
* random number utilities (generators, seed, distributions)

**week07: (algorithms and data structures)**

* complexity analysis, big O notation
* Standard Template Library (STL)
* overview of data structures (incl. runtime of operations)
* iterators
* containers and sequences (linear containers, not trees)
* generic algorithms (find, find\_if, push\_back, back\_inserter, etc.)
* algorithms overview

**week08: (inheritance, polymorphism, from modular to generic programming)**

* protected (means public for derived classes, private for others)
* virtual functions
* Abstract Base Classes (ABC), = 0, override
* comparing virtual functions and templates
* runtime and compile time polymorphism
* programming styles (procedural, modular, object oriented, generic)

**week09: (hardware)**

* computer architecture, CPU, von Neumann bottleneck
* machine code and assembly language
* Instruction Set Architectures (ISA), CISC vs RISC
* pipelining, loop unrolling, branch prediction, Moore’s Law
* parallelization, SIMD, shared & distributed memory
* GPU, RAM, caches, temporal/spatial locality, virtual memory

**week10: (optimization and numerical libraries)**

* profiling, choice of data structures & algorithms
* optimization in assembly language, compiler intrinsic functions
* optimization options
* common subexpression elimination, strength reduction, loop unrolling, etc.
* storage order, unit stride, performance model, roofline
* libraries for linear algebra (Fortran, BLAS, ATLAS, LAPACK, etc.)

**week11: (optimization in C++)**

* template meta programming
* expression templates, Blitz++

**week12: (python)**

* built-in types, mutable vs immutable types
* control flow, list and dict comprehension
* function declaration, arguments, pass by assignment
* classes, magic methods, inheritance, decorators (@wrapper syntax)
* modules (to be imported), string formatting, input/output, exceptions

**week13a: (intro to python packages)**

* Numpy, Scipy, Matplotlib,

**week13b: (input/output)**

* standard streams, pipelining/redirecting, formatting, string/file streams, HDF5

**week13c: (Euler [Bonus])**