Lecture 1: History and Usage of C++

* C++
  + Been around since 1980s, though 2011 were when the big updates began
  + Made features easier to understand and more powerful
  + OO design that is very efficient
* C++11
  + Standardized through ISO and IEC

Lecture 2: Features

* C++ came from C, which was made by Dennis Ritchie at Bell Labs
* C11 came from C
* C++ extended C
  + C++ made by Bjarne at Bell Labs
* C++ standard libraries
  + Classes and functions
  + Use building block approach (build upon existing blocks)
    - Improves portability since they are included in every C++ implementation
* Objects
  + Created by classes
  + Reusable software components
  + Attributes (name, color, size, etc.) and behaviors (calculating, moving, etc.)
    - Each object only knows its own attributes
    - Attributes are specified by the class’s data members
  + OO uses instances of objects instead of procedural approaches
* Classes
  + They are not physical things, but used to create other instances
  + Like blueprints to build a house
  + Instance of a class would have hours, minutes, and sections which are each 4-byte integers, so an instance would take up 12 byes
  + Classes hide information (ex. how they implement functions) inside the classes
    - Encapsulation
* Member functions
  + Performing a task in a program requires a member function
  + Houses the program statement that performs its task
  + Hides these statements from its user
  + Classes house member functions, which perform the class’s tasks
* Instantiation of an object = Build an object from a class
  + Object is an instance of this class
* 4 areas of memory: Code memory, static memory, stack, heap
  + C++ variable could be in the static memory, stack, or heap (not in executable code)
* Encapsulation (information hiding)
  + Classes encapsulate attributes and member functions into objects
  + Objects can communicate with one another, but they do not know how other objects are implemented
* 4 relationships between classes: inheritance, containership, dependency, and realization
* Inheritance
  + One class is a kind of other classes

Lecture 3: Language Comparison

* Fortran
  + Only uses code and data, not extra memory
  + Not powerful
  + No OO
  + Used for computationally intensive algorithms
* COBOL
  + Popular in 80/90s
  + Used for business applications
  + Not OO
  + Generated language
* Pascal
  + Teaching language
* Ada
  + Based on Pascal
  + Mandated by DoD for military applications
* Basic
* C
  + Widely used
  + Used by Linux operating system
* Objective-C
  + Apple computers
* Java
  + Java does not support hard real time (it is not deterministic)
  + Can’t do global optimizations
* Visual Basic
* C#
  + OO but limited to Microsoft OS
* PHP
* Perl
  + Scripting language
* Python
  + OO scripting language that replaced Perl
* JavaScript
* Ruby on Rails
* Scala

Lecture 4: Syntax

* Most syntax is inherited by syntax of C
* Expression performs calculations
  + j =5;
  + x= 2\*s;
* Declaration
  + Declare the variables to be used in program
  + Type followed by name
  + int j;
  + Could include const modifier
* Operands have a value
  + Literals: 10, “Hello World”
  + Variables: x, name
* Operators manipulate values
  + Unary Operators: i++
  + Binar operators: a+b
  + Functions: sin(x)
* Basic C++ operators
  + A table with symbols and numbers

    AI-generated content may be incorrect.
* Order of evaluation: precedence, right-to-left associativity
  + Multiplication comes before edition
  + x=y=5 means y=5 then x=y

Lecture 9: Integers and Doubles

Lecture 10: Arithmetic Operators

* Using namespace std