(1 - 1) Computer Software & Software Development
H&K Chapter 1

Instructor - Beiyu Lin

CptS 121 (May 6th, 2018)

Washington State University

WASHINGTON STATE UNIVERSITY
World Class. Face to Face.

Course Collaborators

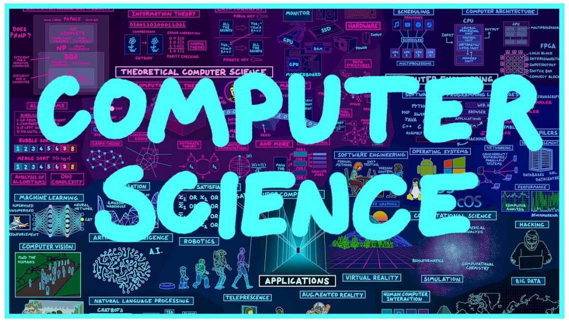
- A lot of material for this course was adapted from Andrew S. O’Fallon and Chris Hundhausen’s course or developed concurrently with them

What is Expected in this Course?

- To learn how to approach and solve problems differently, including some interview like questions
- To build enough programming skills to be one step closer to landing an internship
- Dedication
- And of course, hard work
- You up for the challenge?

What is Computer Science? (1)

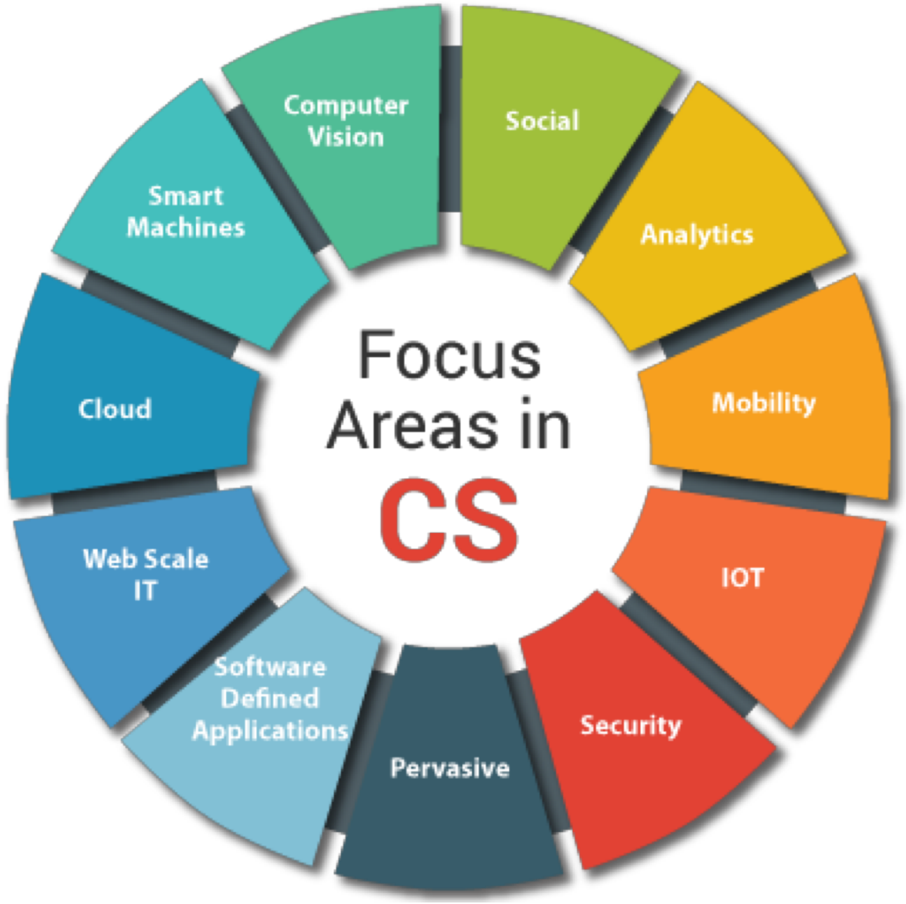
- Computer science is the study of computers and computational systems, with a particular focus on *algorithms*
  - Intersects theory with practice
  - Requires thinking in abstract and concrete terms
  - Not just about building computers and developing programs
  - Involves planning, designing, developing and applying systems
  - Applies analysis to algorithm efficiency, and software performance



Pictures are from:
https://www.msoe.edu/about-msoe/computer-science/
https://www.youtube.com/watch?v=SzJ46YA_RaA

What is Computer Science? (1)

- What are areas of study in Computer Science?



Pictures are from:
https://www.amrita.edu/program/m-tech-computer-science-engineering

What is Computer Science? (2)

- What is an algorithm?
  - A sequence of instructions that solve a problem
- Why are algorithms so important to computer science?
  - If we can specify an algorithm…
    - We can automate the solution
    - We can also repeat a solution to a problem

Activities: Discuss, Write, and Execute an Algorithm

- Activities (in pairs):
  - (1) Verbally discuss (only) an algorithm for drawing a rectangle on the whiteboard with a dry erase marker
  - (2) Write an algorithm for drawing a rectangle on the whiteboard with a dry erase marker
  - (3) Execute the algorithm and draw the rectangle described by the algorithm on a piece of paper or the whiteboard

Class Analysis of Rectangle Drawing Activity? (1)

- Was the rectangle drawn as expected?
  - If no:
    - Was there any miscommunication between you and your partner about how to write the algorithm?
    - Was it because the algorithm was incomplete and/or ambiguous?
    - Was it because the activity statements were incomplete and/or ambiguous?
      - What kinds of clarifying questions could you ask to better understand the activity statements?

Class Analysis of Triangle Drawing Activity? (2)

- Does your triangle look the same as others in the class?
  - If no, why?
    - Could the activity statements have provided more information?
    - Could you have asked clarifying questions?

Formal Definition of Algorithm

- A well ordered collection. . .
- Of unambiguous and effectively computable operations. . .
- That produces a result. . .
- And halts in a finite amount of time.

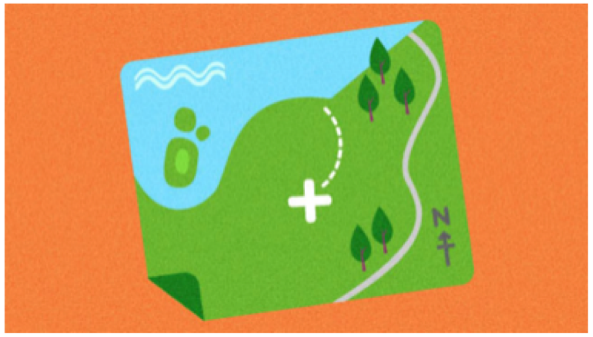
How do we use algorithms in daily life? (4)

I need to make a cake

The algorithm here is a cake recipe. You can find the algorithm to solve this problem in a cookbook!

I can't find the park

The algorithm you need is a set of directions to get to the park. There might be different ways to the park so you can have different algorithms.



Pictures are from:
https://www.bbc.com/bitesize/articles/z3whpv4

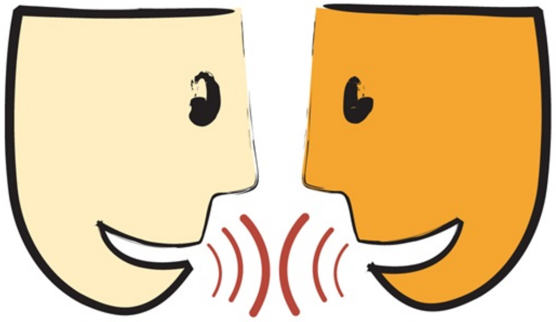
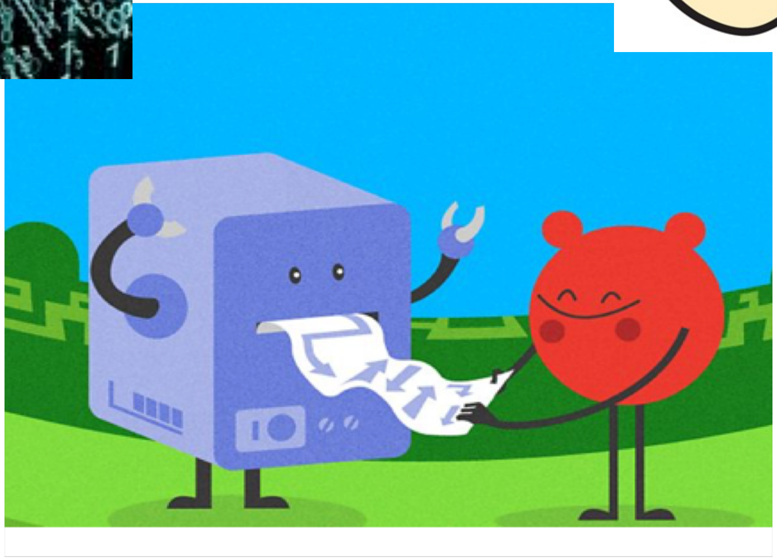
Is this an Algorithm? (4)

- https://www.youtube.com/watch?v=e_WfC8HwVB8 (00:24 – 00:49)
- Count the people in the class room?

How are Algorithms Put Together?

- Sequenced instructions
  - do them in the order given
- Conditional instructions
  - do them if a condition is true
- Iterative instructions
  - do them while a condition is true

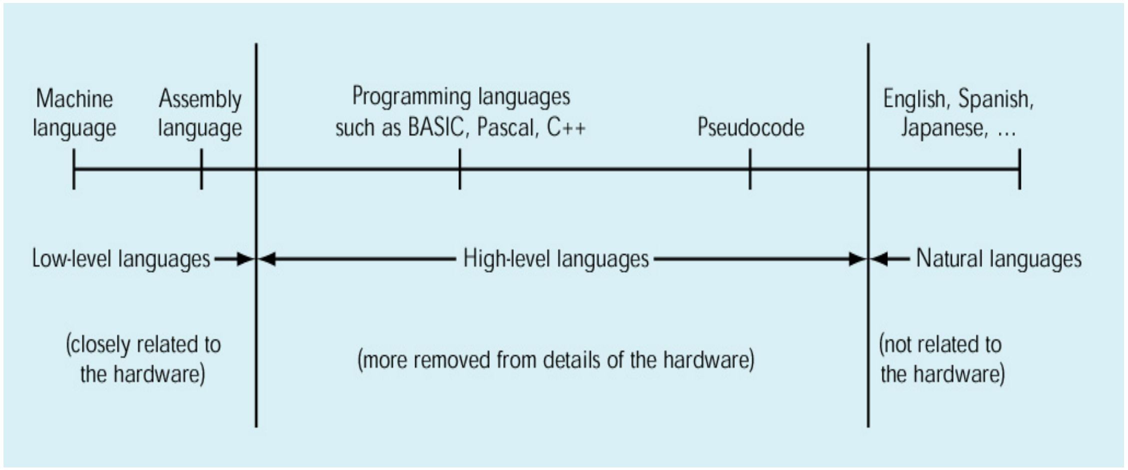
High-Level Programming Languages (1)

Pictures are from:
https://catborrow.wordpress.com/2011/04/17/hello-world/
https://fcw.com/articles/2018/04/10/odni-cio-cdo-organization.aspx

High-Level Programming Languages (1)

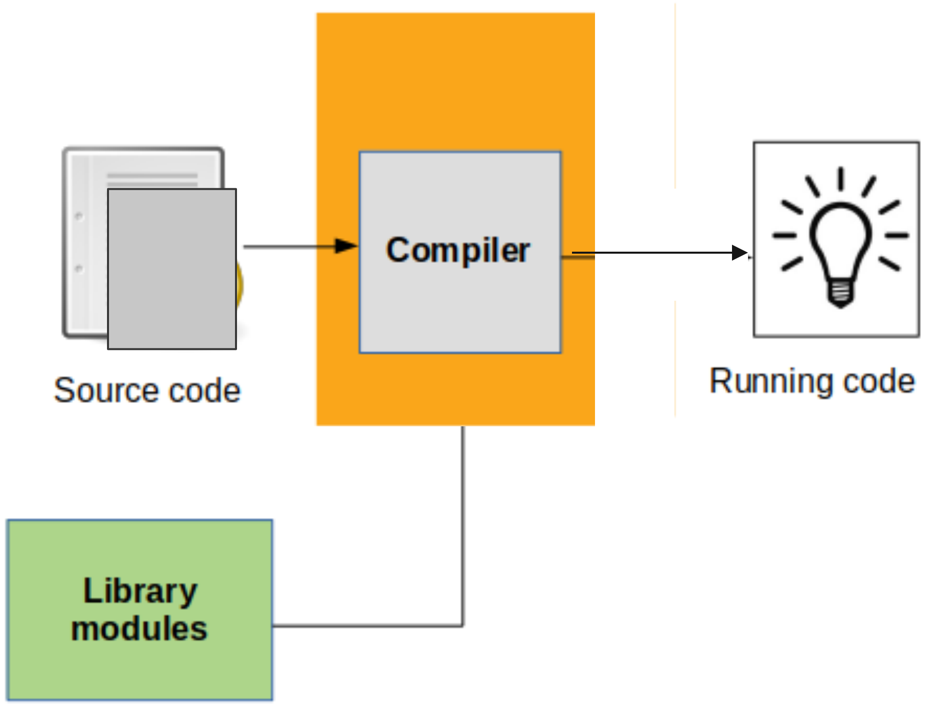
- High-level programming languages
  - The continuum of languages:



Machine language
Assembly language
Programming languages such as BASIC, Pascal, C++
Pseudocode
English, Spanish, Japanese, ...
Low-level languages
High-level languages
Natural languages
(closely related to the hardware)
(more removed from details of the hardware)
(not related to the hardware)

  - Low-level languages were created from the perspective of the machine; working with 1's and 0's, also known as logic levels
  - High-level languages, have natural language like elements

High-Level Programming Languages (2)



High-Level Programming Languages (2)

- Problem: Computers can't understand high-level programming languages
- Solution: They must be translated
  - Programmer uses a text editor to write a text-based source file in a programming language
  - Compiler translates source file
    - Checks to make sure that program is syntactically correct
    - If so, the compiler translates the program into an object file with machine language instructions

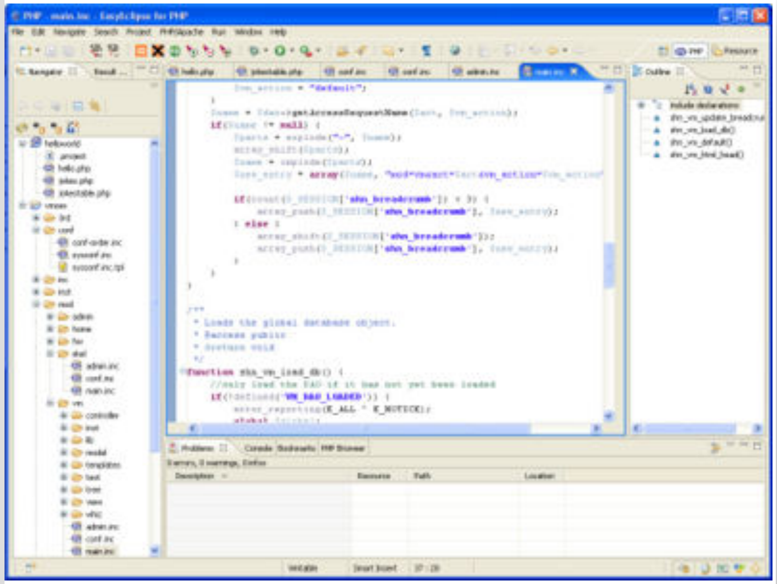
High-Level Programming Languages (3)

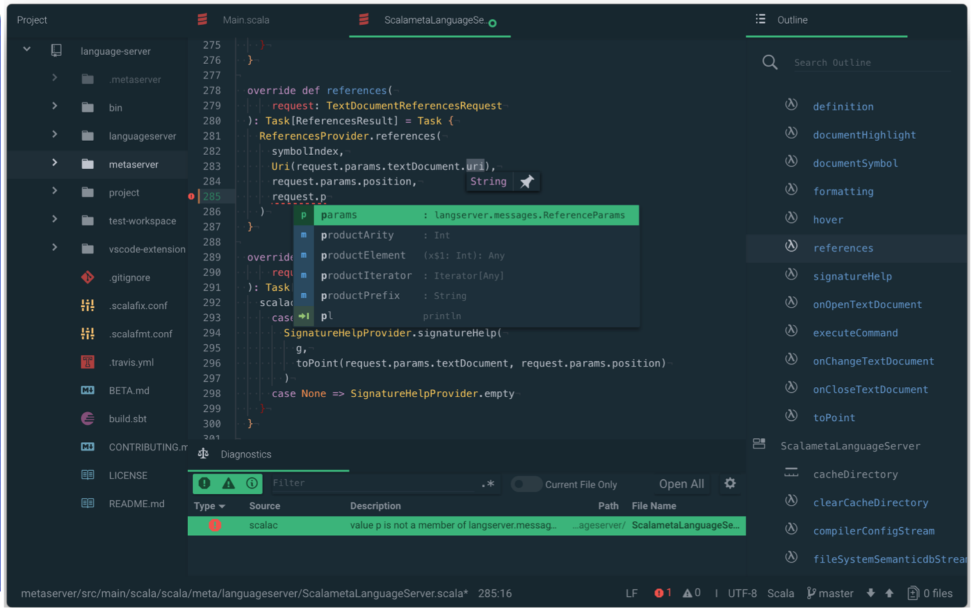
- Object file translated by compiler will not execute!
  - High-level programs often make use of software libraries containing predefined pieces of code, including
    - Math functions
    - Input/output functions
  - In order to execute, object file must be *linked* to object files containing these predefined pieces of code
  - A *Linker* program performs this operation
  - A *Loader* program loads the linked program into memory so that it can be executed

High-Level Programming Languages (4)

- Executing Programs
  - In this class, programs will execute in a text-based window called a *console*
  - Input data can be entered at command-line prompts
  - Output results will be displayed in the console window
  - In the real world, many programs have a graphical user interface (GUI)
  - GUI programming is, however, beyond the scope of this course

High-Level Programming Languages (4)





```
    }
  }

  override def references(
      request: TextDocumentReferencesRequest
  ): Task[ReferencesResult] = Task {
    ReferencesProvider.references(
      symbolIndex,
      Uri(request.params.textDocument.uri),
      request.params.position,
      request.p
    )
  }

  override
      requ
  ): Task
    scala
      case
        SignatureHelpProvider.signatureHelp(
          g,
          toPoint(request.params.textDocument, request.params.position)
        )
      case None => SignatureHelpProvider.empty
    }
  }
```

Type	Source	Description	Path	File Name
	scalac	value p is not a member of langserver.messag...	...ageserver/	ScalametaLanguageSe...

metaserver/src/main/scala/scala/meta/languageserver/ScalametaLanguageServer.scala* 285:16

C. Hundhausen, A. O'Fallon, B. Lin

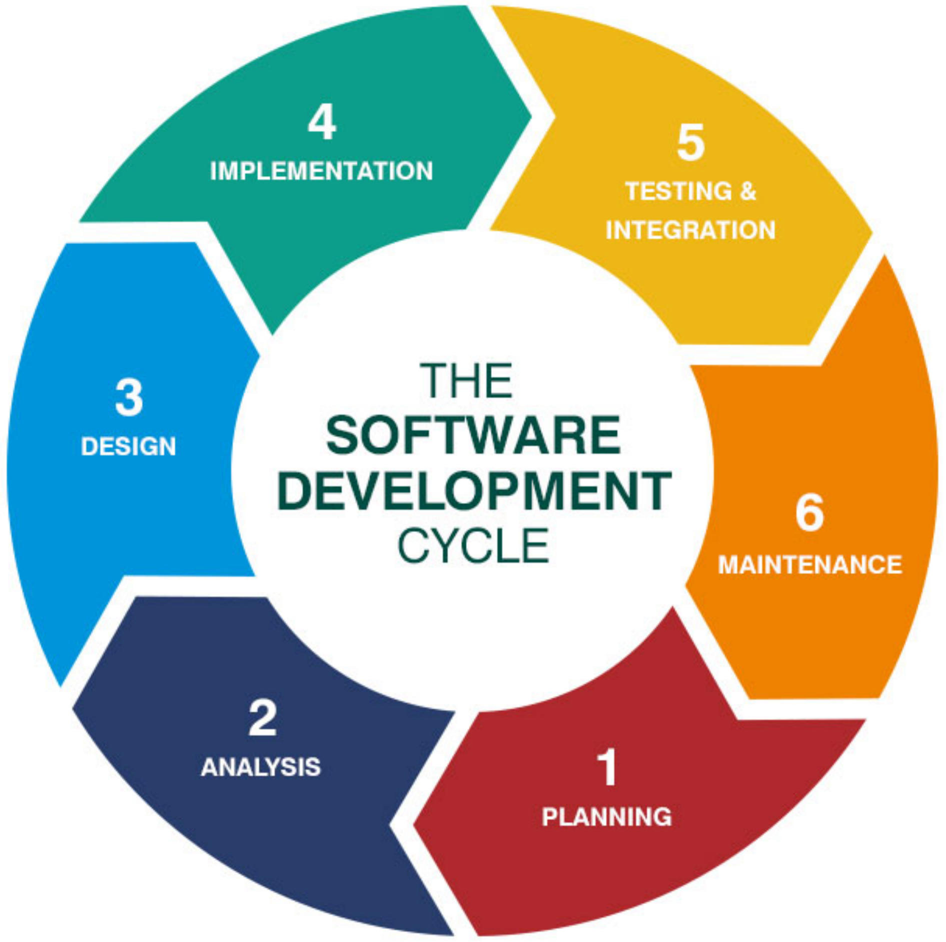
Pictures are from:
http://www.cs.trincoll.edu/hfoss/wiki/F09_Notes:_Integrated_Development_Environment_(IDE):_Eclipse
https://atom.io/packages/ide-scala

Pictures are from:
http://www.cs.trincoll.edu/hfoss/wiki/F09_Notes:_Integrated_Development_Environment_(IDE):_Eclipse
https://atom.io/packages/ide-scala

High-Level Programming Languages (5)

- Integrated Development Environments (IDE)
  - Combine compiler, linker, and loader with a source code editor
    - Generally a single button will start the translation process
  - Provide a variety of tools to assist programmers, for example,
    - Source code syntax highlighting
    - Autocompletion lists ("Intellisense")
    - A debugger, which allows a programmer to step through programs, one instruction at a time
    - A testing framework for developing unit tests

Software Development Method



Pictures are from:
https://online.husson.edu/software-development-cycle/

Software Development Method

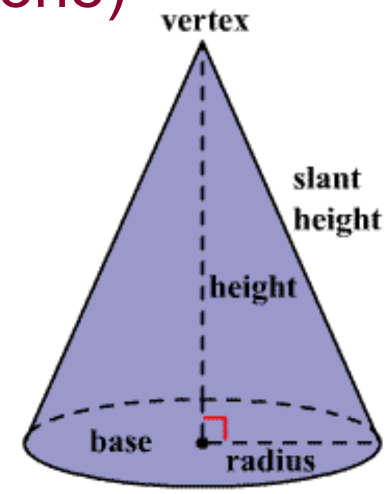
- Equivalent to the "Scientific Method" in the sciences, and the "Systems Approach" in business
- Six basic steps:
  1. Specify problem requirements
  2. Analyze the problem
  3. Design an algorithm to solve the problem
  4. Implement the algorithm
  5. Test and verify the completed program
  6. Maintain and update the program

Applying the Software Development Method (1)

- Developing software is an iterative process, your first solution is generally not your best!
- Your understanding of software your required to build evolves as you understand the problem more!
- At this point don't be afraid to make mistakes!
- Example problem: *Compute the volume of a cone*

Applying the Software Development Method (2)

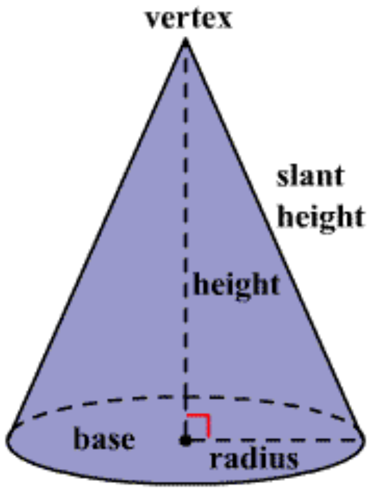
- Data Requirements
  - Problem input:
    radius (of the base), height (of the cone)
  - Problem output:
    volume (of the cone)
  - Relevant formula:
    volume = 1 / 3 * pi * radius^2 * height



Pictures are from:
https://www.varsitytutors.com/hotmath/hotmath_help/topics/volume-of-a-cone

Applying the Software Development Method (3)

- Design
  - Algorithm
    - Get the radius and height for the cone
    - Compute the volume of the cone
    - Display the resultant volume of the cone
  - Refined algorithm
    - Get the radius and height for the cone
    - Compute the volume of the cone
      - volume = 1 / 3 * pi * radius^2 * height
    - Display the resultant volume of the cone



Pictures are from:
https://www.varsitytutors.com/hotmath/hotmath_help/topics/volume-of-a-cone

Applying the Software Development Method (4)

- Implementation (in C)

```
#include <stdio.h> /* Needed for printf (), scanf () */
#define PI 3.14159 /* Constant macro */

int main (void)
{
          int height = 0, radius = 0;
          double volume = 0.0;

          printf ("Enter height of cone as integer: "); /* Displays prompt message */
          scanf ("%d", &height); /* Gets the value from the user/keyboard */
          printf ("Enter radius of base of cone as integer: ");
          scanf ("%d", &radius);

          /* Compute the volume of the given cone */
          volume = ((double) 1 / 3) * PI * radius * radius * height;

          /* Display the resultant volume of the given cone */
          printf ("Volume of cone with radius %d and height %d is %lf.\n", radius, height, volume);

          return 0;
}
```

Applying the Software Development Method (5)

- Note: At this point, don't worry about understanding the details of C syntax! We'll get to that later
- Testing
  - We would execute the program, trying several different input data values and observing the results
    - Debugging is NOT testing! It’s a result of testing!
  - Each test is defined by a test case
    - A test case provides actual inputs, system state or configuration information, and expected results
  - Should always test “boundaries” of inputs and conditions

Applying the Software Development Method (6)

- Maintenance
  - Most software requires continual improvements, adaptations, and corrections; software patches are a result of maintenance

Next Lecture…

- We've covered the general software development method
- It's time to start learning the C language!

References

- J.R. Hanly & E.B. Koffman, *Problem Solving and Program Design in C (8th Ed.)*, Pearson Education, Inc., 2016