Fundamentals of Programming CS-114

Lecture 1

Intro & Affiliations

Area of research: Analysis of medical images/signals using Image/signal processing and Machine Learning Techniques

BIOMISA

BIOmetrics, Medical Image and Signal Analysis Research Group

EMERGING Technologies Lab

RISTECH

RESEARCH AND INNOVATION IN SCIENCE ENGINEERING AND TECHNOLOGY

www.biomisa.org/usman
www.biomisa.org
www.risetech.pk
www.albasr.com
www.ekko.pk

What is your level of expertise in Programming?

a) Never programmed before

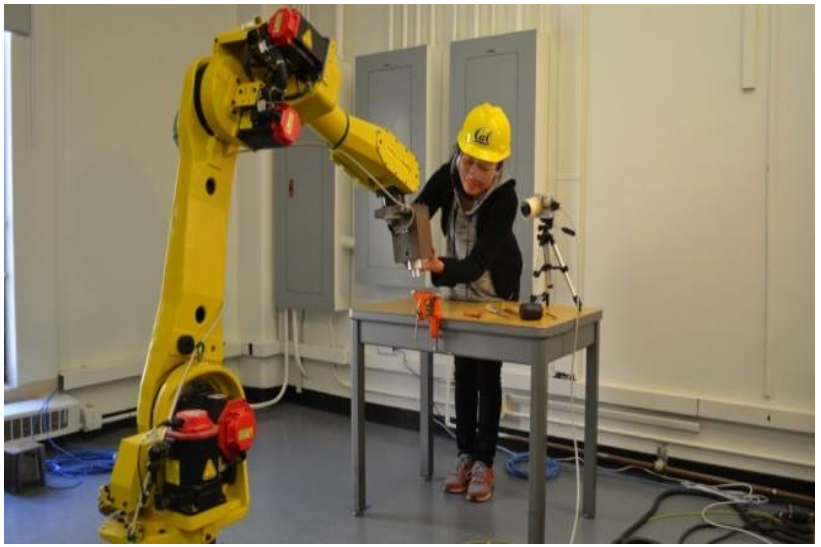
b) Have taken at least one course in programming

c) Have done significant programming

d) I am a master programmer

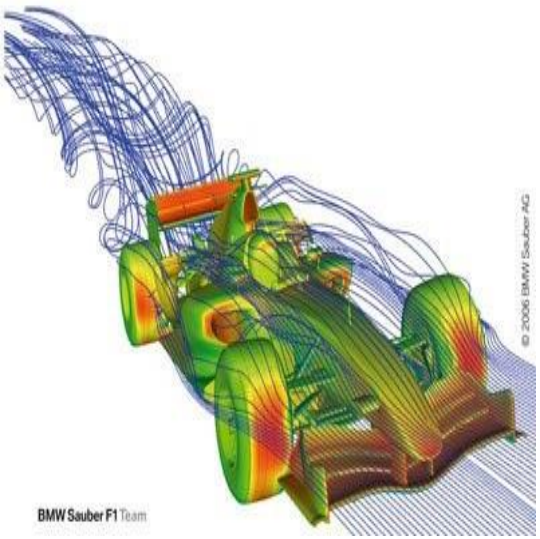
Applications of Programming

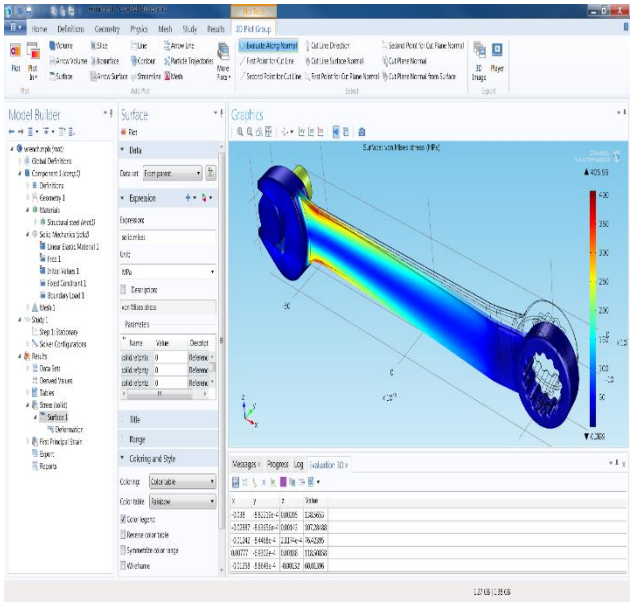
Robotics:

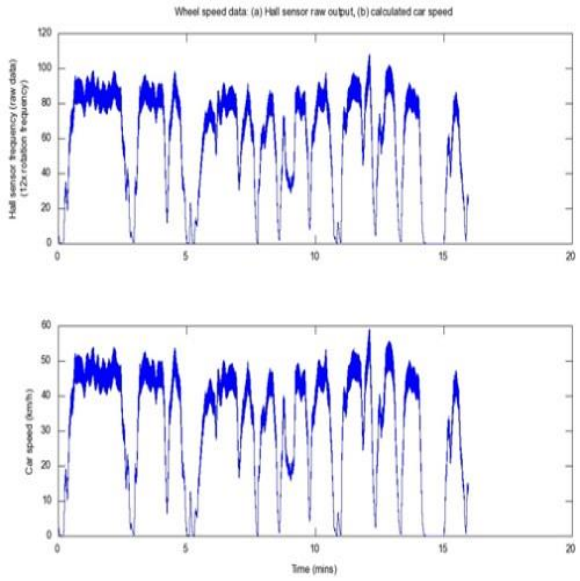


Applications of Programming

Simulation:

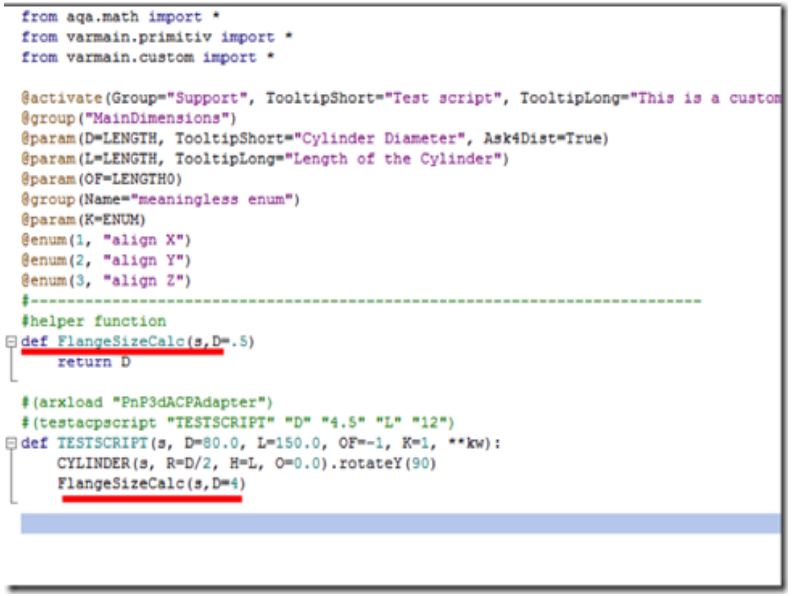






Applications of Programming

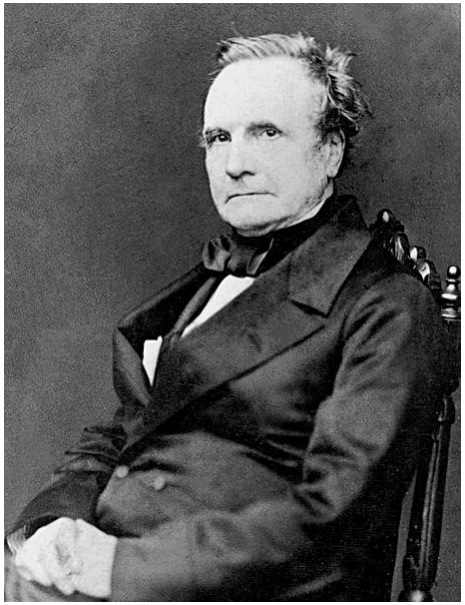
Scripting:

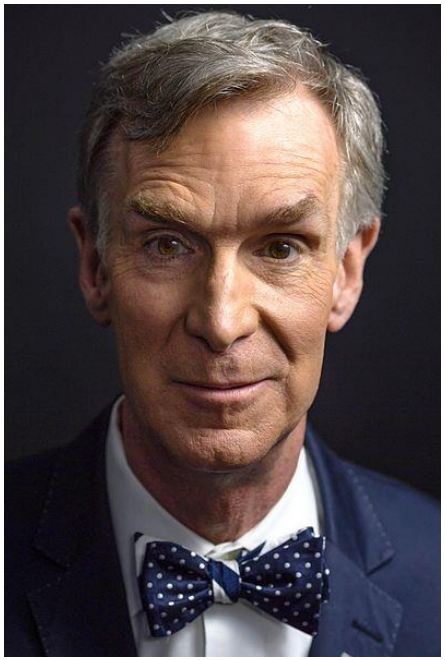


```
from aqa.math import *
from varmain.primitiv import *
from varmain.custom import *

@activate(Group="Support", TooltipShort="Test script", TooltipLong="This is a custom
@group("MainDimensions")
@param(D=LENGTH, TooltipShort="Cylinder Diameter", Ask4Dist=True)
@param(L=LENGTH, TooltipLong="Length of the Cylinder")
@param(OF=LENGTH0)
@group(Name="meaningless enum")
@param(K=ENUM)
@enum(1, "align X")
@enum(2, "align Y")
@enum(3, "align Z")
#-------------------------------------------------------------------------
#helper function
def FlangeSizeCalc(s,D=.5)
    return D

#(arxload "PnP3dACPAdapter")
#(testacpscript "TESTSCRIPT" "D" "4.5" "L" "12")
def TESTSCRIPT(s, D=80.0, L=150.0, OF=-1, K=1, **kw):
    CYLINDER(s, R=D/2, H=L, O=0.0).rotateY(90)
    FlangeSizeCalc(s,D=4)
```

```
import clr
clr.AddReference('ProtoGeometry')
clr.AddReference('DSCoreNodes')
clr.AddReference('DSOffice')
clr.AddReference('Translation')
import DSCore
from DSCore import *
import DSOffice
from DSOffice import *
from Autodesk.DesignScript.Geometry import *

#Assign the intput
convertedCurve1 = IN[0]
toleranceSimplify = IN[1]

#Simplify Converted curves
simplifiedCurve1 = []
for items in convertedCurve1:
    simplifiedCurve1.append(Curve.Simplify(items, toleranceSimplify))

#Convert Curve to NurbsCurve to get ControlPoints
nurbsCurve1 = []
for items in simplifiedCurve1:
    nurbsCurve1.append(Curve.ToNurbsCurve(items))
curvePoints1 = []
for items in nurbsCurve1:
    curvePoints1.append(NurbsCurve.ControlPoints(items))

#Create Polygon to get the Centroid
polygon1 = []
for items in curvePoints1:
    polygon1.append(Polygon.ByPoints(items))
center1 = []
for items in polygon1:
    center1.append(Polygon.Center(items))



#Assign the output
OUT = [curvePoints1, center1]
```

Some Inventors that took a path less travelled



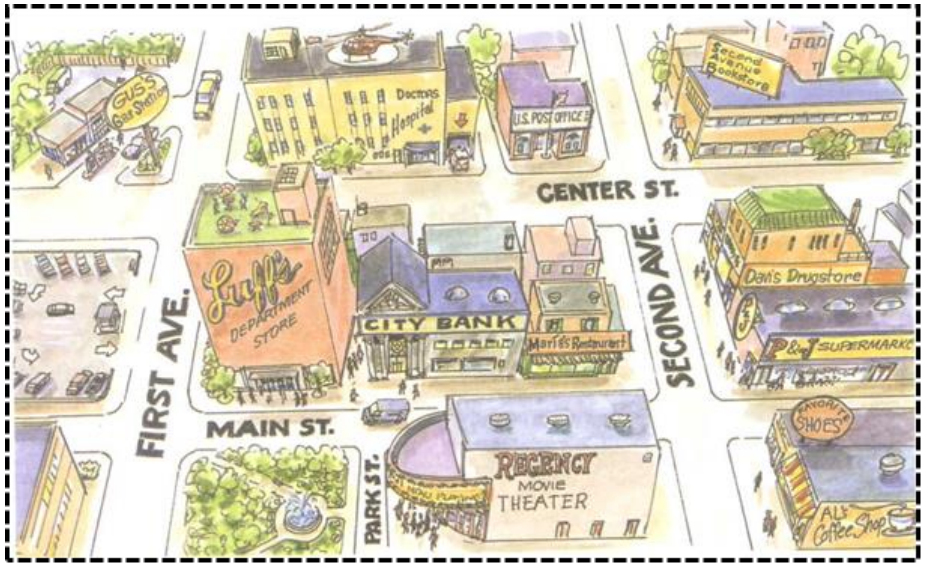


Mechanical Engineers – Programming & Software

- https://mechanicalc.com/posts/software-for-mechanical-engineers
- https://www.youtube.com/watch?v=1_1H7G7DqG4
- https://www.youtube.com/watch?v=gHSZ1S9996U

Programming Languages

Pick up the red pen and
place it on the table



Walk 500 meter on First Ave and
then turn right

Very precise instructions that will result in a single, expected outcome.

Programming Languages

- A programming language “tells” the computer precisely what to do
- Instructions for a specific task are encoded in a language
- A computer understands these instructions and performs the task

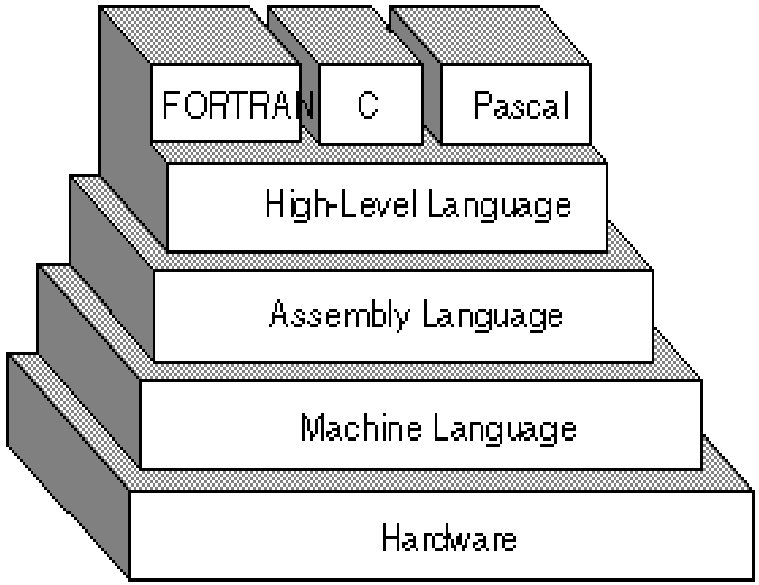
Programming Languages

Walk 10 feet in a straight line

- Cammina per 10 piedi in linea retta
- Camine 10 pies en línea recta
- Пройдите 10 футов по прямой
- Siúil 10 troigh i líne dhíreach

Programming Languages

- Three types of programming languages:
  - Machine
  - Assembly
  - High Level



- Assembly and High Level languages are used more often

- Some examples of high level languages: C++, C#, Java, Python

The Ideal Way to Do Computing

- The ideal way to ask computer to do something is to order it in a natural language e.g.
  - I want to view this webpage
  - Calculate my annual tax
  - etc.
- However, today's computer's are not intelligent enough to understand our orders in natural language completely.

Where We Are in Computers?

- At the very basic level computers use the concept of an electrical pulse.
  - Low voltage is represented as 0
  - High voltage is represented as 1
- To instruct a computer we need ask the computer in the language of 0s and 1s commonly known as ***machine language***.
- For instance 73 in a number in natural language in the language of 0s and 1s, it becomes 1001001

Machine Language: Our First Interaction with the Computer

- *machine language*.

```
10110011   00011001
01111010   11010001  10010100
10011111   00011001
01011100   11010001  10010000
10111011   11010001  10010110
```

  - Finding an average of two numbers
- Not very intuitive way of working
- Not possible for humans to achieve a lot using machine language

One Step Beyond - Assembly Language

- One level above machine language is assembly language

```
MOV  0,  SUM
MOV  NUM,   AC
ADD  SUM,  AC
STO  SUM,  TOT
```

- More understandable but still very difficult for many of us.
- An *assembler* translates assembly language into *machine language*.

Another Step - High-level Languages

- High-level languages is another level above machine language.

X = (Y + Z) / 2

- Much more understandable.
- A *compiler* translates high-level language into *assembly language*.

Where are we going?

- The next step in computing is to use natural language over a high-level language.
- But we are still away from it.
- A lot of research needs to be carried out before we actually see this.
- Until then our task is to use high-level languages in its best possible ways

*Why C

An introductory programming course should use a non-proprietary programming language that adheres to an international standard. A standardized language is stable, and its evolution is supported and maintained by industry and overseen by technical standards committees and other stakeholders.

As a language, C continues to evolve but remains backward compatible. As long as it conforms to the C99 standard, a compiler will work with programs written in C89. Matlab, by contrast, is a proprietary mathematical programming language that makes collaboration difficult with individuals not running Matlab.

C has arguably become the most common programming language, both in engineering and elsewhere. More than 90 percent of desktop computer programs, from operating systems to word processors, are written in C or its relative, C++. C runs on all platforms, and most other languages may be translated into C.

In the Programming Language Popularity Website, C tops the list, while C++ is fourth. FORTRAN is No. 21 and Matlab is nowhere to be seen.

C is especially useful for mechanical engineers because it is the language of choice for hardware interfaces, and commonly used for data acquisition and real-time robotic control. C is also the most widely used language for programming embedded processors: Of the 9 billion microprocessors manufactured in 2005, 8.8 billion were embedded.

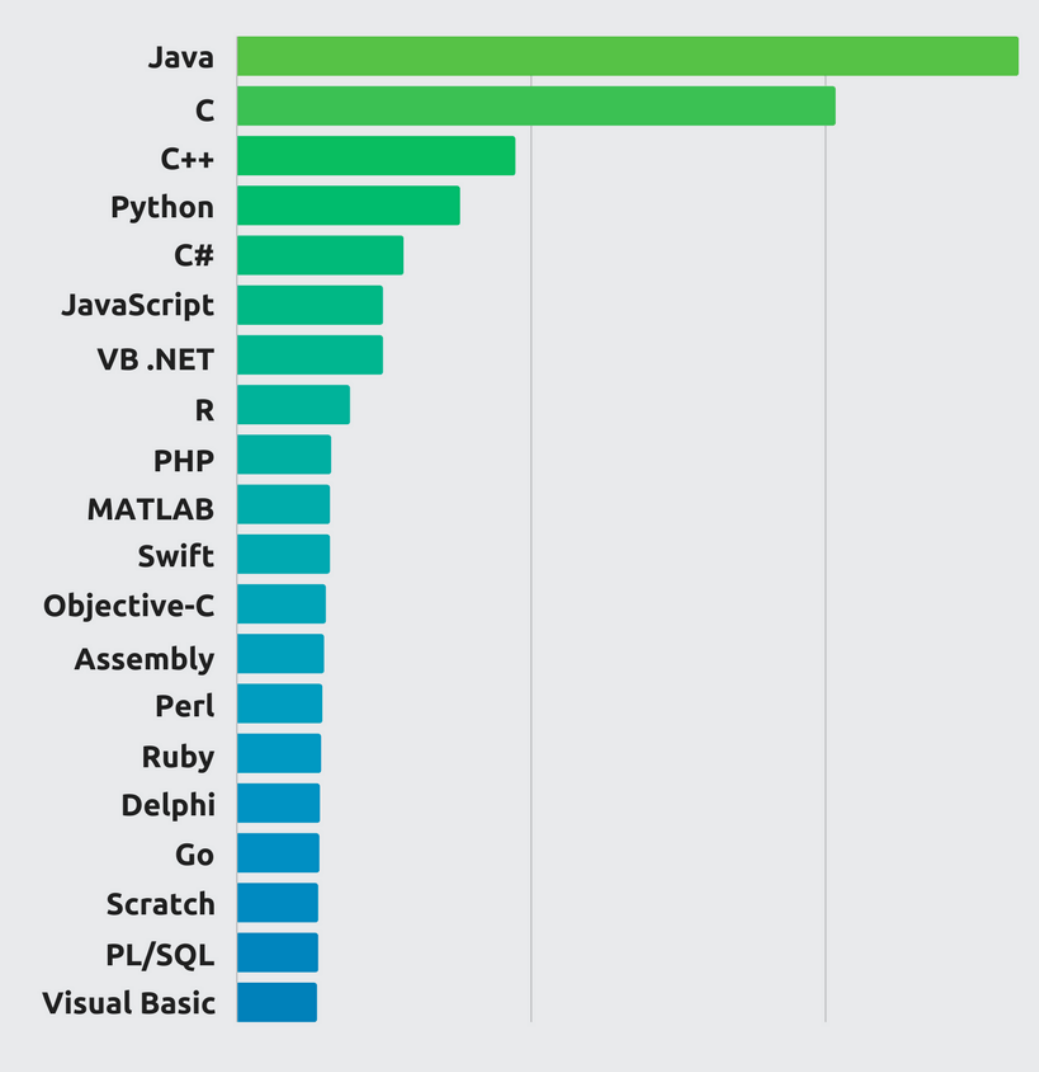
*University of California

Back-end Programming Languages

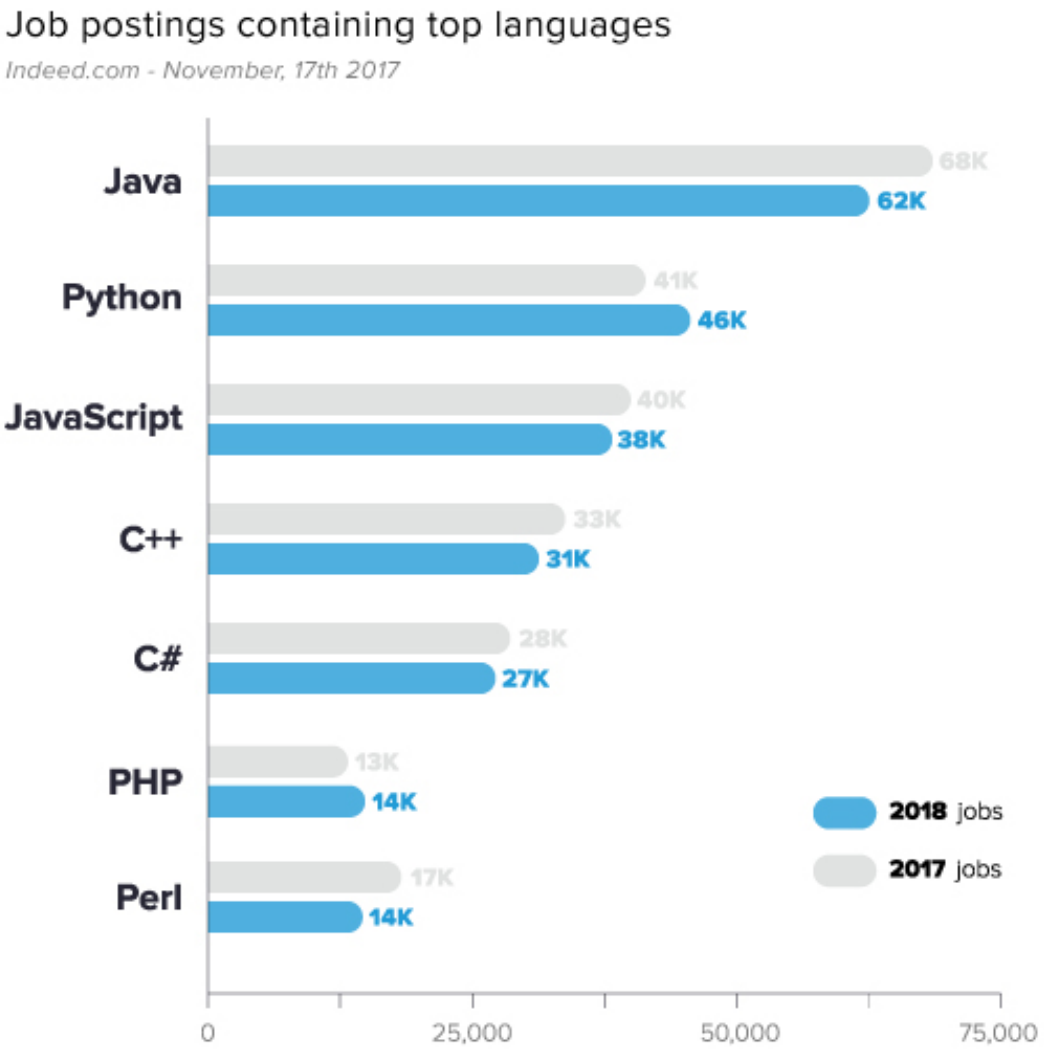
Back-end (Server-side) table in most popular websites

Websites	C#	C	C++	D	Erlang	Go	Hack	Java	JavaScript	Perl	PHP	Python	Ruby	Scala	Xhp
Google.com	No	Yes	Yes	No	No	Yes	No	Yes	No	No	Yes	Yes	No	No	No
YouTube.com	No	Yes	Yes	No	No	Yes	No	Yes	No	No	No	Yes	No	No	No
Facebook.com	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	Yes
Yahoo	No	Yes	Yes	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Amazon.com	No	No	Yes	No	No	No	No	Yes	No	Yes	No	No	No	No	No
Wikipedia.org	No	No	No	No	No	No	Yes	No	No	No	Yes	No	No	No	No
Twitter.com	No	No	Yes	No	No	No	No	Yes	No	No	No	No	Yes	Yes	No
Bing	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No
eBay.com	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	Yes	No
MSN.com	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Linkedin.com	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	Yes	No
Pinterest	No	No	No	No	Yes	No	No	No	No	No	No	Yes	No	No	No
WordPress.com	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No

Top Programming Languages



Top Languages wrt Jobs



Introduction to C++

- Where did C++ come from?
  - Derived from the C language

- Why the ‘++’?
  - ++ is an operator in C++

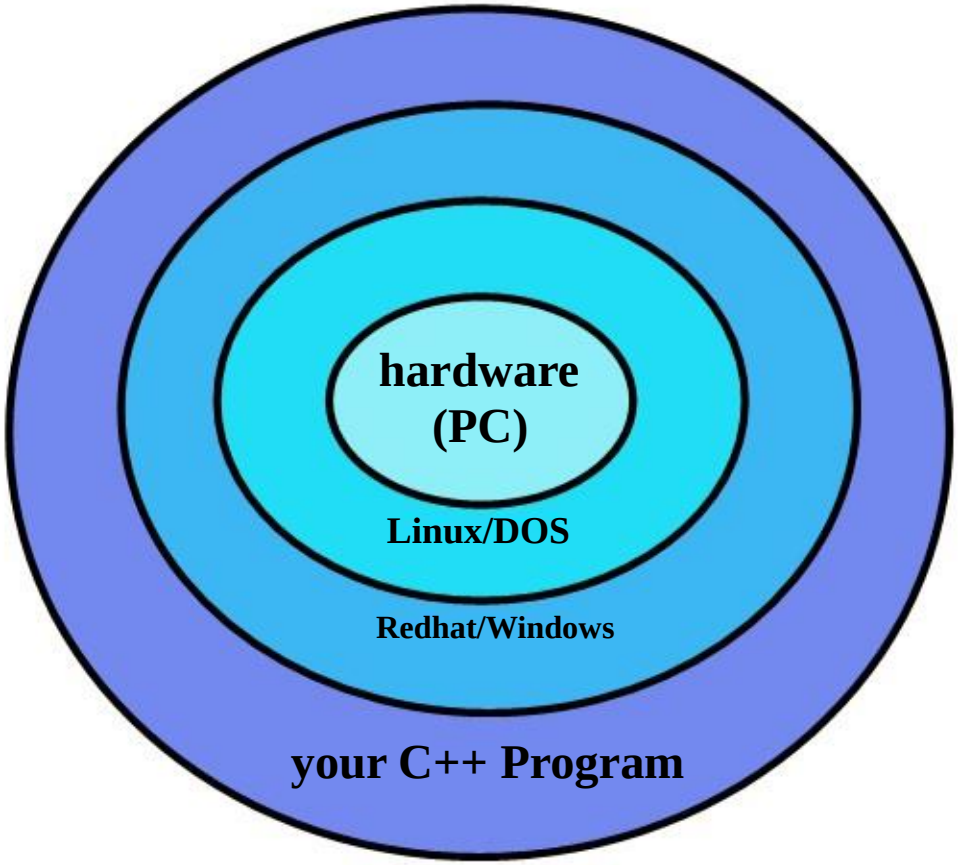
C++ History

- C developed by Dennis Ritchie at AT&T Bell Labs in the 1970s.
  - Used to maintain UNIX systems
  - Many commercial applications written in c
- C++ developed by Bjarne Stroustrup at AT&T Bell Labs in the 1980s.
  - Overcame several shortcomings of C
  - Incorporated object oriented programming
  - C remains a subset of C++

Object-Oriented Programming

- OOP: Now the dominant way to program, yet it is almost 40 years old! (Simula ’67 and Smalltalk ’72 were the first OOPLs)
  - Dr. Alan Kay received ACM's Turing Award, the "Nobel Prize of Computing,“ in 2003 for Smalltalk, the first complete dynamic OOPL
- It was slow to catch on, but since the mid-90’s everybody’s been doing it!
- OOP emphasizes objects, which often reflect real-life objects
  - have both properties and capabilities
  - i.e., they can perform tasks: “they know how to...”

The Computer Onion



How to Learn Programming

- Everybody learns programming at their own pace.
- So do not be impressed by the person sitting next to you because he coded a given program in 20 minutes and you are taking more than an hour.
- **Speed programming does not necessarily mean quality of the final output.**

How to Learn Programming (cont’d)

-

1. Writing a good description of the problem.
2. Breaking down the given problem into small pieces.
3. Turning small pieces into pseudo-code.
4. Deciding the integration mechanism of the pieces.
5. Writing the program for each piece.
6. Integrating all the pieces together.

Scare of Programming?

- Why most students are afraid of programming
  - Paradigm Change
    - Programming is totally different paradigm. You are working on something and you cannot even touch the final output you can only feel it. It is different then other subjects like Physics, Chemistry, Biology, etc.
  - Peer Pressure
    - Some people are naturally good in programming so others think that this is a natural ability and they cannot learn it.

Scare of Programming? (cont'd)

- Lack of Understanding in Fundamental Concepts
  - Some people start programming without a clue of what is going on behind the scene in the computer. As a result they have a flawed understanding from day one of their programming experience
- Time Factor: Programming takes a lot of time
  - Programming may take a lot of time at the start but once a person is comfortable with the concepts and has mastered the basic skills it is just like any other profession.

A Word of Advice

- Without good command on programming any engineering qualification in current era is almost
"***worthless***".

- There is an acute shortage of programmers in the global market and with time this shortage is increasing.

Class timings

- DE-42 Mechanical
  - Lecture
    - CRM-06
    - Tuesday 1400—1540
  - Lab
    - Computing Lab
    - Wed 1225—1630
  - Office Hours
    - Tue 1540 – 1630
    - Thursday 1225 - 1400
  - Course Material

    http://biomisa.org/usman/fundamentals-programming-fall-2020.html

Course Outlines

- Introduction to Programming
- Introduction to C++
- Basic Structure of C++
- Basic input and output statements
- Operators
- Variables
- Loops
- Decision operators
- Functions
- Arrays & Strings
- Structures
- Pointers
- Files I/O

Course Book

- Object Oriented Programming in C++ by Robert Lafore
- C++ How to Programme by Dietel and Dietel

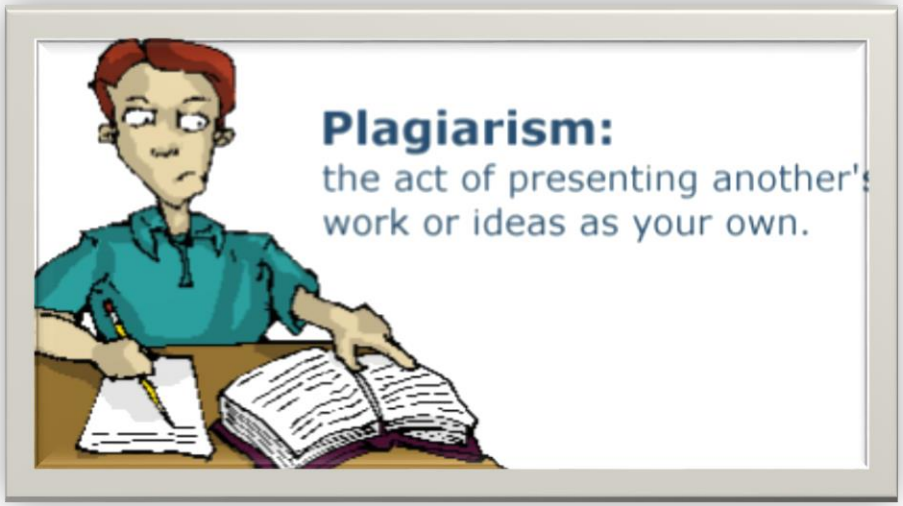
- ***References:***
- A Structured Programing Approach Using C++ by Behrouz A. Forouzan
- www.cplusplus.com

CODE OF ETHICS

- All students must come to class on time (Attendance will be taken in first 5 to 10 mins)
- Students should remain attentive during class and avoid use of Mobile phone, Laptops or any gadgets
- Obedience to all laws, discipline code, rules and community norms
- Respect peers, faculty and staff through actions and speech
- Student should not be sleeping during class
- Bring writing material and books
- Class participation is encouraged

Policies

- No extensions in assignment deadlines.
- Exams will be closed book.
- Never cheat.
  - "Better fail NOW or else will fail somewhere LATER in life"
- Plagiarism will also have strict penalties.



Courtesy Dr. Khawar

Adapted from *What is Plagiarism* PowerPoint
http://mciu.org/~spjvweb/plagiarism.ppt

Learning Outcomes

- **Enabling Knowledge:** The process of designing algorithmic solutions to computable problems; the syntax and control structures of a programming language i.e. C++, which enable you to code these algorithmic solutions using standard coding conventions

- **Critical Thinking and Analysis:** Ability to analyze the requirements for solving simple algorithmic problems.

Learning Outcomes (cont’d)

- **Problem Solving**: Ability to design and implement programs to solve simple algorithmic computing problems, based on analysis of the requirements.

- **Communication:** Ability to explain key concepts of algorithmic design, in written form, to IT specialists.

Course Learning Outcomes

Course Learning Outcome (CLOs)		PLOs	Learning Level
CLO 1	**Understanding** fundamental concepts of computer programing to solve a problem	1	C2
CLO 2	**Implementing** and **applying** the concepts of functions and Structures in C++	1	C3
CLO 3	**Analyse** real life problems and **relate** their knowledge of programing and algorithms to solve those complex problems using arrays, functions and pointers	3	C4
CLO 4	To code, document, test and implement a well-structured, robust computer program in Visual Studio using C++ programming language	5	P2

Acknowledgements

1. Deitel and Deitel: C++ How to Program, 7th Edition, Prentice Hall Publications
2. Robert Lafore: Object-Oriented Programming in C++, Fourth Edition, December 2001,Sams Publishing .
3. A Structured Programing Approach Using C++ by Behrouz A. Forouzan
4. www.cplusplus.com

Material in these slides has been taken from, the following resources