*Week 1*

**Outline and Addendum Overview**

**BB and MS Team**

**Recap on VPN, Putty and WinSCP**

**Intro to VS IDE :**

* working area of the developer , from notepad (basic editor) to sophisticated environment (Visual Studio). VS makes life a bit easy on one click sometimes. Provides rich editor
* Steps to create Project and Add file:

1. Create empty Project -🡪 name it 🡪 keep the directory path same and don’t change it 🡪 create

* Project is your working space that will keep all of your code files and output files in one place. You can check the path in the Files and Folders.
* Once Project is created, you are ready to add program file which we call as source file

2, To Add file Click Project on top main menu and select Add New Item or right click your project name on side panel 🡪Add🡪New Item

* C/C++ compiler are not same but C++ supports C mode so you have to mention that by providing source file name with **.c extension**.
* If .c is not added to filename it will create .cpp source file and will use C++ compiler and some of C commands will not work
* Matrix will also not take any other file ext then .C so **be careful.**
* Check location, it will show same path with Project name as sub folder

3. Add and it will open .c empty editor inside the VS window. That’s where you write code . check name of file and makesure it has .c extension

* To get output, there are 2 steps

1. Under Build on top menu, click Build solution--- this will build whole project source files and not just one file

2. Go to Debug and click Start without Debugging ----this will allow the program to execute on Console Window or command prompt. the program right now does not support GUI (Graphical User Interface).

* You can lookup Console Programming and Windows programming
* Also lookup Shell programming or interactive programming

1. intro to programming languages

2. Types of programming language now used : very low level (Assembly) , low level (c though it is high level), oop (C++, java, Csharp, python, swift, Ruby), scripting (java script, python, Ruby)

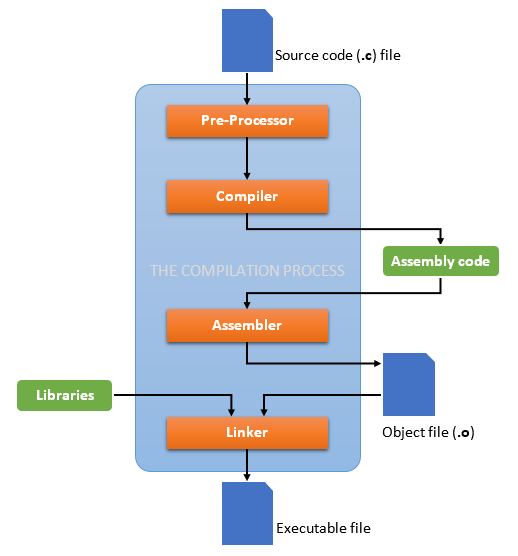
Generations : discussed in Compiler section

3. Other Types : Web Development (ASP, JSP, PHP, Ruby on Rail) Cloud dev (JS (node.js), java, ruby, python, , dot net, php , R , Go), Query Lang (SQL), Markup lang (XML, HTML)

3. Why C? : oldest but still in top 10

* works well with embedded system becuz of access to OS, memory, core machine operations, developing compilers, Operating System itself, reading from sensors, working with Arduino, PI etc.
* basis of other languages like C++, Java, C#, etc.
* Some areas are Robotics, IOT, Industrial Automation. Biomed.
* But should not be the only lang to know.

4. C requires Compiler to run the code .

* there are 3 types of languages, compiler based or interpreted or hybrid. Python is interpreted and Java is Hybrid
* 

Reference : <https://medium.com/@laura.derohan/compiling-c-files-with-gcc-step-by-step-8e78318052>

Filename.obj is object file in windows , filename.out is object file in GCC

5. Every languages needs a grammar, rules to write sentences in that language. C lang grammar is widely adopted by later languages and became more of standard for easy transitioning of programmer from one lang to another.

6. basic programming lang components are same.

7. C program structure

* Header files at top e.g stdio.h ,
  + # symbol is preprocessor directive ( any thing after that is prep processed code files already with the C package. These are called header files. This symbol can also be used for other stuff (not required now)
  + Filename with .h extension are header files , they are also code files but without main . what ever built in commands which we call functions we use ,are in the header files .
  + if you don’t include header file and will use those build in function, compiler will not know where to look for and will give error.
  + E.g. Printf() is in stdio.h
* Main program : This is the start point for execution. If it is not there program will compile but will not execute or run. The syntax is important.
* Curly brackets {} signify block (will be used with different constructs)
* Semicolon ; : to end statement . it is full stop of C . Remove this and you will cc a bunch of error popping out . Why ? we will discuss is some other class .
* Displaying Output:
  + printf(). To display text as output it has to be in “ “ (double quotation marks).

8. Case sensitive. A **is not** a , Main **is not** main. Be careful , Most beginner level errors are because of mixing capital and small letter .

**Remember Matrix matches your output character by character with the output we have put. So change of case will cause your code to fail on Matrix. Other issues may be spelling mistake in output or some letter missing .**

**Make sure the output provided in workshop questions matches 101% with your program output**

9. Errors and debugging.

* Read error , they will give you detail on where code went wrong or where to begin. They don’t bite.
* Always read first error and try to work on that. you will be surprised how rest falls in place.

10. computer knowhow, internal organization, CPU breakup, Memory breakup

11. Information and Data , Bits and Bytes, ASCII and other codes

12. Compiler vs Interpreter

* Compiler , Linker loader , executer
* Program -🡪compiles-🡪machine level code-🡪executes-🡪output
* Every machine have different codes and hence compiler for every machine is different e.g C compiler for Linux is gcc and for windows is VS C compile.
* Read whole code in one go. slow process. once m/c converted execute fast
* Complex debugging. One error causes ripple effect and many errors pops up. makes it hard to debug. pin point one bug and many goes away.