23/2/25 DAY 1

Installation & Environment Setup

- Install MinGW Minimalist GNU for Windows
- Eclipse IDE J2EE
- Get the Path from C Drive for MinGW bin path
- Search Environment Variable -> Path Section -> New Path -> Paste
- Eclipse -> Help -> Eclipse Marketplace
- Search CDT -> Eclipse CDT 12.x -> Confirm -> Restart Eclipse
- Go to Window -> Perspective -> Open -> Other -> C/C++

What is programming?

Giving Precise Instructions to solve a problem with computer. Programming is a process of writing instructions for a computer to perform specific tasks.

- Assembely - 1940-50s
- Fortan - 1950-60s
- Cobol - 1960-70s
- BASIC - 1960-70s
- C - 1969 - 1972

History of C

- C: Development 1969-1972
- Release: 1972
- Where: Bell Labs
- Who: Dennis Ritchie
- Why: For UNIX OS development

Standardization of C (ANSI C, ISO C)

- ANSI -> 1989 -> ANSI C - C89
- ISO
  - C90
  - C99
  - C11
  - C17
  - C23

Struture of C program

```
#include <stdio.h>  //include directives -> header files/macros

// entry point function
int main(){
    printf("Hello World!"); // send a output on screen
    return 0; // return value for successful execution
}
```

Concepts of Language

Tokens

Smallest unit of a program: Everything written in a program.

```
int main(){
    return 0;
}

Tokens: int, main, (, ), {, return, 0, ;, }
```

Keywords

Predefined/Reserved words, which cannot be used for variables or definition.

> e.g. int, return, void, if, else, while, for, switch, case, break, continue, etc.

Identifiers

It is used to identify the variables, functions, arrays, etc. in a program.

> e.g. num, add, main, etc.

Literals

Literals are the fixed values which are directly inserted into the code.

e.g.

- 10 - **integer literal**
- 'A' - **character literal**
- "Hello World" - **string literal**

Variables

Container which stores a value in it. Variables are used to store data in a program. It can be changed during the execution of the program.

e.g.

```
int num = 10; // num is a variable of type int which stores the value 10
```

Operators

We can perform some set of operations with the help of operators.

- Arithmetic - + - * / %
- Relational - > < <= => == !=
- Logical - && || !
- Bitwise - & | !
- Assigment - = , +=, -=, *=, /=
- Unary - ++, --
- Ternary - condition ? true : false

Data types

- Basic Data types:
    - int
    - float
    - double
    - char
    - void
- Derived Data Types:
    - Array
    - Pointer
    - Function
- User Defined Data Types:
    - Structure
    - Union

Functions

Block of code to perform specific task. It is a reusable code which can be called multiple times in a program.

```
int add(int a, int b){
    return a + b;
}

int main(){
    add(10,20);
}
```

SDK (Software Development Kit)

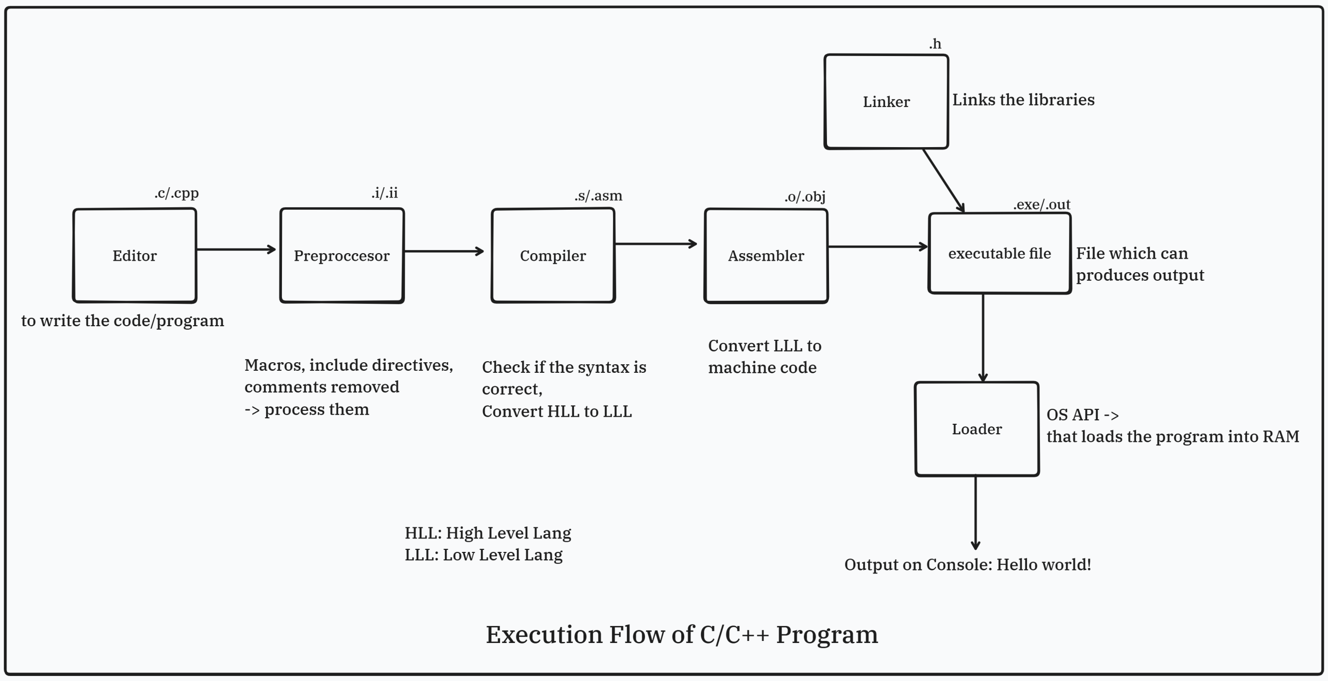
- Development Tools
    - Editor/IDE: vscode, Eclipse IDE, Notepad, Notepad++, Intellij, Code-Blocks, Dev-CPP, Turbo-C

- Compiler: gcc/g++
- Assembler: asm
- Linker: ld

- Documentation
    - cppreference.com, tenouk.com, microsoft for C++
- Runtime Environment
    - c runtime
- Supporting Libraries
    - Standard libraries
    - External libraries

Execution flow of a C/C++ program

- Build -> Run it as C/C++ Program/Application


.c/.cpp
Editor
to write the code/program
.i/.ii
Preproccesor
Macros, include directives, comments removed -> process them
.s/.asm
Compiler
Check if the syntax is correct, Convert HLL to LLL
.o/.obj
Assembler
Convert LLL to machine code
.h
Linker
Links the libraries
.exe/.out
executable file
File which can produces output
Loader
OS API -> that loads the program into RAM
Output on Console: Hello world!
HLL: High Level Lang
LLL: Low Level Lang
Execution Flow of C/C++ Program

main() entry point

main() is the entry point of a C/C++ program. It is the first function which is executed when we run a C/C++ program. It is mandatory to have a main() function in a C/C++ program.

```
int main(){
    return 0;
}
```

```
int main(void){
    return 0;
}
```

```
int main(int argc, char *argv[]){
        // argc -> argument count: how many arguments are present
        // argv -> argument variables: node | index.js
    return 0;
}
```

```
int main(int argc, char **argv){
    return 0;
}
```

```
int main(int argc, char *argv[], char *envp[]){ // envp -> environment parameters:
    return 0;
}
```

```
int main(int argc, char **argv, char **envp){
    return 0;
}
```

```
// not recommend: not standard way of writing the main function
// it may not work in some compilers
void main(){

}
```

Variables & Data types

Variables

Name Given to a memory location to store a value.

```
data_type variable_name;
```

- Rules:
    - Name must be meaningful
    - keywords cannot be used as variable names
    - special characters not allowed, _ can be used
    - cannot start with digit
    - No whitespaces allowed

Data Types

Data Types has some properties:

- Nature: It defines the type of data which can be stored in a variable.
- Size: It defines the size of the data which can be stored in a variable.
    - int: 4 bytes
    - float: 4 bytes
    - double: 8 bytes
    - char: 1 byte
    - void: 0
- Range: It defines the range of values which can be stored in a variable.
    - int: -2,147,483,648 to 2,147,483,647
    - float: 1.2E-38 to 3.4E+38
    - double: 2.2E-308 to 1.8E+308
    - char: -128 to 127

  The above range is for signed data types for 32 bit system.

> The size and range of the data types may vary from compiler to compiler and platform to platform.

16 bit vs 32 bit vs 64 bit

- 16 bit: it can address 2^16 = 65536 memory locations
- 32 bit: it can address 2^32 = 4,294,967,296 memory locations
- 64 bit: it can address 2^64 = 18,446,744,073,709,551,616 memory locations

`The current trend is 64 bit, but 32 bit is still widely used in embedded systems and older hardware.`

- Type Modifiers:
    - short: It reduces the range
    - long: It extends the range
    - signed: It allows both -ve and +ve values
    - unsigned: It allows only +ve values
- Type Qualifiers
    - const: we cannot change the value/readonly values
    - static: accesible throughout the program for all the objects
    - volatile: can be changed by the external factors(hardware).

Comments in C/C++

- This are not the part of your program execution
- Documentation for the fellow developers or for yourself in future
- It is ignored by the compiler

- It is used to explain the code and make it more readable

Single Line Comment:

```
// this is entry point
int main(){
    return 0;
}
```

Multi-line Comment:

```
/*
This is a
Multiline comment
*/
int main(){
    return 0;
}
```

Declaration vs Definition

Declaration

It is the process of declaring a variable, function, etc. It tells the compiler about the name and type of the variable, function, etc. but it does not allocate memory for it.

```
int num; // declaration
```

Definition

It is the process of defining a variable, function, etc. It tells the compiler about the name, type and value of the variable, function, etc. It allocates memory for it.

```
int num = 10; // definition + initialization
```

Initialization vs Assignment

Initialization

It is the process of giving an initial value to a variable at the time of declaration.

```
int num = 10; // definition + initialization
```

Assignment

It is the process of giving a value to a variable after it has been declared.

```
int num;     // declaration
num = 10;    // assignment
```

```
int num; -> having another memory
num = 10; // assignment
// int num = 10; -> error
```