Day 1 (01-12-2025)

- Environment Setups
  - Download and intall MinGW Minimalist GNU for Windows
  - Add Environment Variables from C:\MinGW\bin
  - Download and install Eclispe IDE
  - Install CDT plugin in Eclipse IDE Marketplace
  - Change Perspective to C/C++ in Eclipse
- C brief history
  - Creator: Dennis Ritchie
  - Year: 1972
  - Developed at: Bell Labs
  - ANSI C: 1989
- Standards of C
  - C89/C90
  - C99
  - C11
  - C18
  - C23
- Hello World Program
- Structure of C Program

```
#include <cstdio> //c++ standard library for input and output
int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Strongly Typed Checked Language

```
int num1 = 10; // valid

//  num1 = "shil"; // not valid
```

- Concepts of Language

  - Tokens: Every smlallest element in a program is called a token
  - Keywords: Reserved words in any language
  - Identifiers: Name given to variables, functions, arrays, etc. which is given by the programmer
  - Literals : Constant values that are used in the program
  - Variables: Named memory locations to store data
  - Data Types: Type of data that a variable can hold

- Basic Data Types in C
  - int
  - float
  - double
  - char
  - void
- Derived Data Types : Derived from basic data types to use in complex scenarios
  - Arrays
  - Pointers
  - Functions
- User Defined Data Types
  - Structures

  ```
  struct Student{
      int rollNo;
      char name[100];
  };
  ```

  - Unions

  ```
  union Employee{
      int empId;
      char name[100];
  };
  ```

- Entry point function : main() function which is the starting point of execution of any C/C++ program

```
int main(){
   return 0;  //return 0 -> successful execution of the program -> OS
}
```

```
int main(int argc, char* args[]){
    return 0;
}
```

```
int main(int argc, char* args[], char* envp[]){
    // argc -> stores count of command line arguments
    // args[] -> stores values or commands which are being sent from command line
```

```
    // envp[] -> environment libraries or paths/includes
    return 0;
}
```

void main - `Not a standard way of defining main function in C/C++. Some compilers may allow it but it's not recommended`.

```
void main(){

}
```

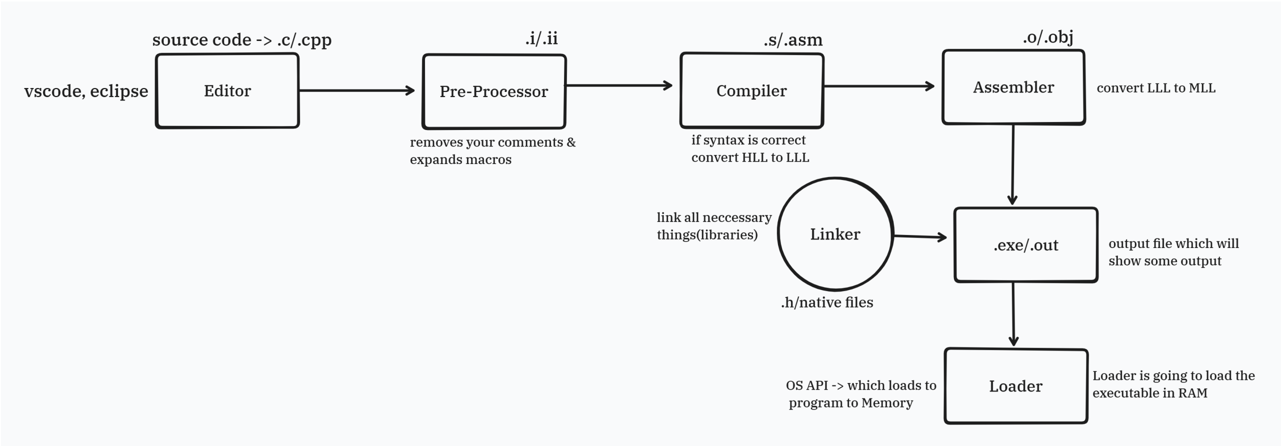
```
void main(int argc, char* args[], char* envp[]){

}
```

```
void main(int argc, char* args[]){

}
```

- Software Development Kit
  - Development Tools, Runtime Environment, Documentation, Supporting Libraries Development Tools: Editor -> vscode, eclipse, notpad, notepad++ Compiler: gcc, g++
- C/C++ Flow of execution


source code -> .c/.cpp
vscode, eclipse
Editor
.i/.ii
Pre-Processor
removes your comments & expands macros
.s/.asm
Compiler
if syntax is correct convert HLL to LLL
.o/.obj
Assembler
convert LLL to MLL
link all neccessary things(libraries)
Linker
.h/native files
.exe/.out
output file which will show some output
OS API -> which loads to program to Memory
Loader
Loader is going to load the executable in RAM

  - Preprocessor - handles directives like #include, #define, etc. & removes comments from the code.
  - Compiler - translates the preprocessed code into assembly code (HLL to LLL).
  - Assembler - converts assembly code into machine code (LLL to MLL).
  - Linker - combines multiple object files into a single executable.
  - Loader - loads the executable into memory for execution.

- Comments - to create good documenataion about your code, used to make code readable and understandable for another programmer.

```
// This is a single line comment
```

```
/* This is a
   multi line comment */
```

- Declaration and Definition

```
int num;        // Declaration + Definition
```

```
int num = 10;        // Definition
```

- Initiation and Assignment

```
int num = 10;        // Initiation
```

```
int num;         // Declaration
num = 20;        // Assignment
```

- Working with functions Function - A block of code that performs a specific task Function Declaration - Declaring the function prototype Function Definition - Defining the actual body of the function Function Call - Invoking the function to perform the task

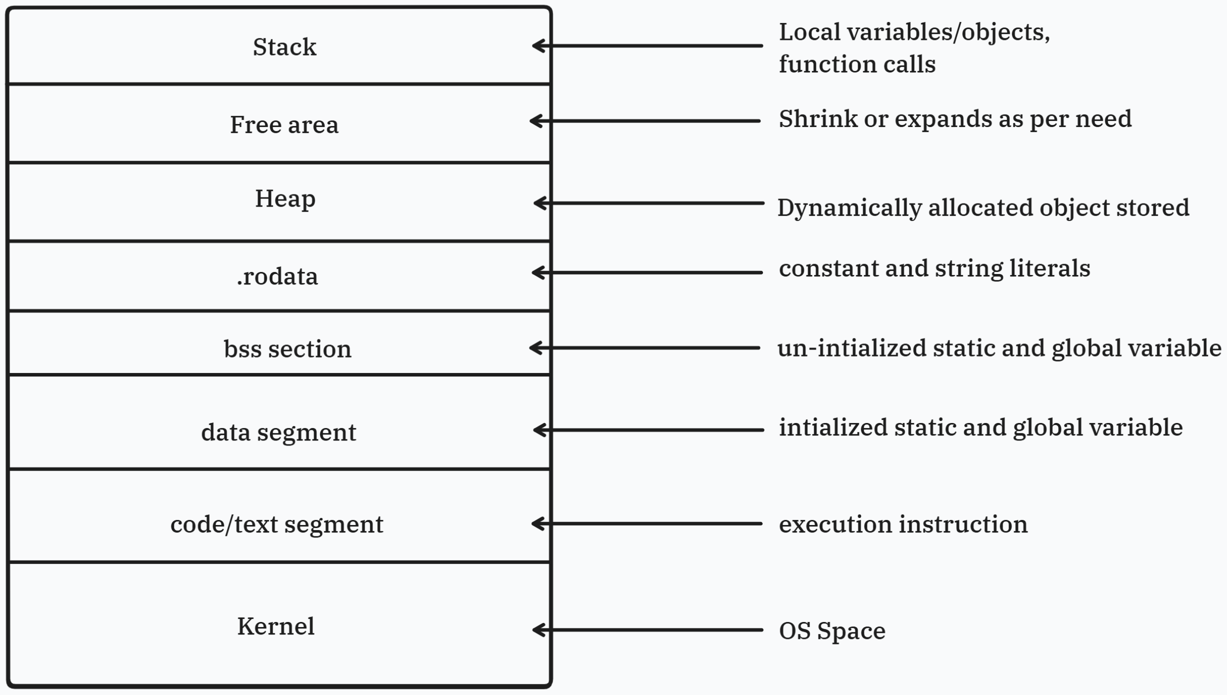
```
// Function Declaration
// Also known as Function Prototype
int add(int a, int b);

// Function Call
int main(){
    int result = add(5, 10);
    return 0;
}

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

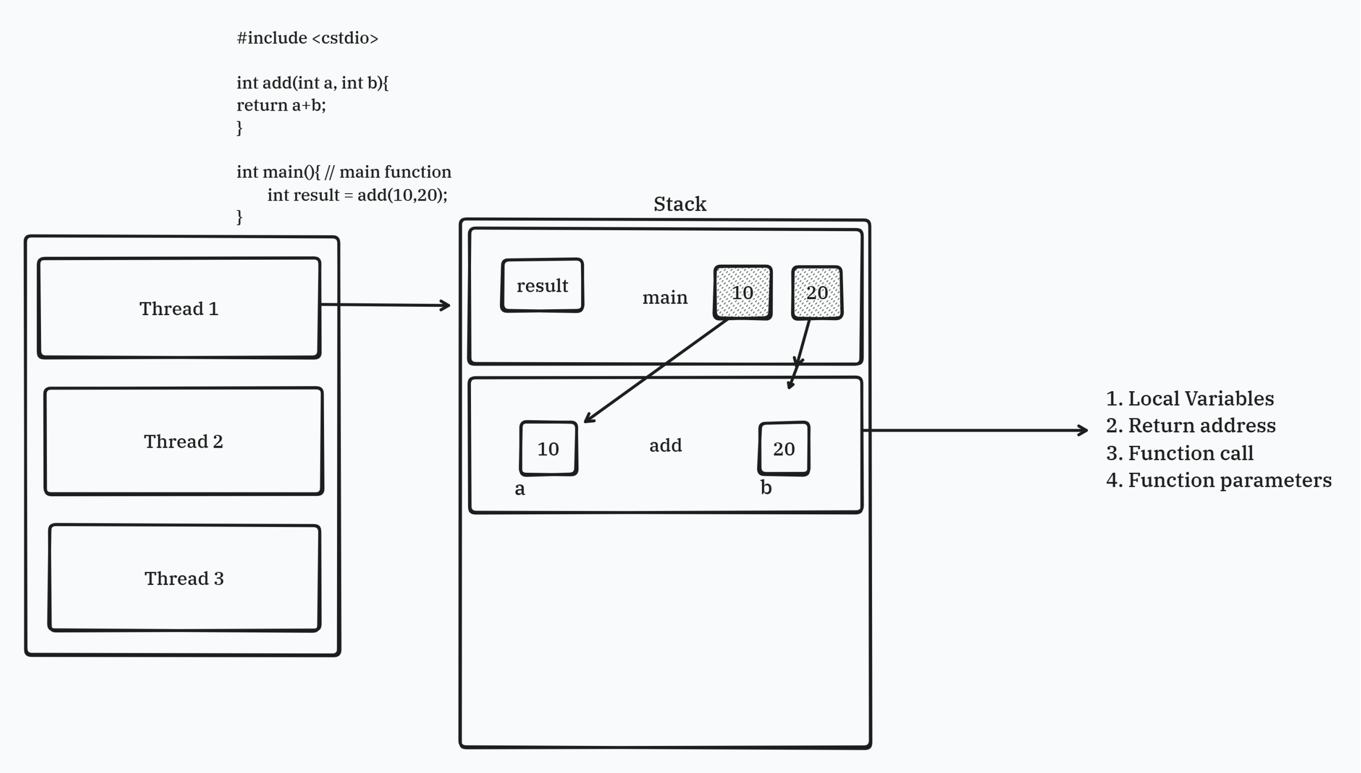
- Types of Functions
    - with arguments and with return type
    - with arguments and without return type
    - without arguments and with return type
    - without arguments and without return type

- Memory Structure of C/C++ Program


Stack
Local variables/objects, function calls
Free area
Shrink or expands as per need
Heap
Dynamically allocated object stored
.rodata
constant and string literals
bss section
un-intialized static and global variable
data segment
intialized static and global variable
code/text segment
execution instruction
Kernel
OS Space

memory outline/diagram

- Function Activation Record - A data structure that contains information about a function call, including local variables, parameters, return address, etc.



- Pointer Concept : A pointer is a variable that stores the memory address of another variable.
- Declaring a pointer

```
int *ptr; // valid

int* ptr; // valid

int * ptr; // valid
```

```
int main(){

    int num = 10;

    int *ptr = &num; //reference

    *ptr = 20; //valid

    printf("num value: %d \n", num);
    printf("num address: %p\n", &num);

    printf("====PTR Values==== \n");
    printf("num value from ptr: %d\n", *ptr);  //dereference
    printf("num address from ptr: %p\n", ptr);
    printf("address of ptr: %p\n", &ptr);

    return 0;
}
```