C++ Notes Day-1 Date: 08-12-2024

Setting-Up Environment C/C++

- Toolchain: gcc
    - Install MingW Gcc/G++
    - Setup path of bin folder to env
- IDE: Eclipse
    - Install CDT extention from eclipse marketplace
    - Change perspective to C/C++

Introduction to C/C++

- C brief history
    - Founder of C: Denis Riche
    - Year: 1972
    - Bell Lab
    - Hardware: PDP-11
    - C/C++ are the statically type checked as well as strongly types checked programming language
    - C is known as general purpose programming language

```
int num=100; //OK
bool a=90; //NOT OK
num='A';    //NOT OK
```

```
a=900;  //OK
a="Malkeet";    //OK
```

- ANSI is looking after standerization of C/C++
- standards of C
    - 89
    - 95
    - 99
    - 11
    - 17
    - 23

Data Types and Working of C/C++

- Tokens: Every indivisual entity written in program is known as token.
    - Keywords: Keywords are used to write the program.
    - Identifiers: User defined entities inside the program. eg. variable, method/function/, class, structure
    - Literals:90,100.

    - Operators: eg. +,-,==,=
- Data Types in C: To store some data of specific type inside the program. We need data types.
    - Basic Data Types:
        - int
        - char
        - float
        - double
        - void
    - Derived Data Types:
        - Array
        - Pointer
        - Functions
    - User Defined Data Types:
        - structure
        - union
- Type Modifiers
    - short
    - long
    - signed
    - unsigned
- Type Specifiers/Qualifiers
    - const
    - volatile
- Variable: Variable specify the following terms: eg. int num1=100;
    - Data Type
    - Size
    - State
    - Range

```
int a;  //a is variable of type integer having 2 bytes memory and storing GV as
value and range of this variable can be min (-32678) to max (32677)
//Action:1. Reserve 2 bytes memory, labled that memory with 'a' and put GV as
value
```

- Entry point function
    - ANSI specify that there should be entry point function inside a .c source file.
    - 'main' inside the .c source file is know as entry point function.
    - 'main' is user defined function
    - It is the sole responsibility of OS to call a main function, thats the reason it is known as Callback Function
    - 'main' function in C/C++ is a global function
    - if we not define 'main' function, then linker generate error.
- Syntax of main
- Syntax-1

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

- Syntax-2

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

}
```

- Syntax-3

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

- Syntax-4

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

}
```

- Syntax-5

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

- Syntax-6

```
void main(void)
{

}
```

- Syntax-7

```c
void main()
{

}
```

- Reference: https://en.cppreference.com/w/c
- Software Development Kit: Development Tools + Documentation + Runtime Environment + Supporting Libraries
- Development Tools:
    - Editor: It is used to create/edit the source file (.c/.cpp)
    - Example:
        - Windows: Notepad, Notepad++, Visual Code, Wordpad etc.
        - Linux: nano. vim, vi, gedit, VS Code etc.
        - Mac OS: Vim, vi, TextEditor, VS Code etc.
    - Preprocessor
        - It is a system program whose primary job is to:
            - To remove comments from the source file.
            - To expand macros.
        - Example: CPP (C/C++ Preprocessor)
        - Preprocessor generates intermediate file (.i/.ii)
    - Compiler
        - It is a system program whose primary job is to:
            - To check Syntax.
            - To convert a HLL code into low-level language code (Assebmly Language / Code)
        - Example:
            - Turbo C: tcc.exe
            - MS Visual Studio: cl.exe
            - Linux: gcc
        - Compiler generates .asm/.s file.
    - Assembler
        - It is a system program which is used to convert low-level language code into machine level language.
        - Example:
            - Turbo: Tasm
            - MS Visual Studio: Masm
            - Linux: as
        - Assembler generates .obj/.o file.
    - Linker:
        - It is system program whose job is to link machine code with Supporting Libraries files.
        - It is responsible for generating executed file (.exe).
        - Example:
            - Turbo: Tlink.exe
            - MS Visual Studio: link.exe
            - Linux: ld
    - Loader:

        - It is an OS API.
        - It is used to load executable file from HDD to Main memory (RAM).
    - Debugger:
        - Logical error inside a program is known as bug.
        - To identify the bugs we need Debugger.
        - Example: gdb,ddd
- Documentation
    - It can be in the form .html/.pdf/.txt format.
    - Example: https://www.tenouk.com/ModuleW.html
- Runtime Environment
    - It is responsible for the entire execution of program / application.
    - Example: C Runtime
- C Flow of execution Image
- Reference : https://www.tenouk.com/ModuleW.html
- Comments: Comments inside the program are used to maintain Documentations
    - Single Line Comments: //Double Forward slash is used to put single line comments
    - Multiline Comments / Block Comments: /* */ is used to put block comment

```
//Single Line Comment
```

```
/*
Multiline or Block Comments
*/
```

- Declaration and Definition of Functions
    - Declaration of Function:
        - Syntax: return_type nameofthefunction(parameterlist)

```
int Add(int x, int y);
```

- Local Function Declaration

```
int main()
{
    void Add(); //Function Declaration (Local Declaration of Add Function)
    cout<<"Am Main"<<endl;
    Add();  //Function Call
    return 0;
}
void Add() //Function Definition
{
    cout<<"Am Add";
}
```

- Global Function Declaration

```
void Add(); //Function Declaration (Global Declaration of Add Function)
int main()
{

    cout<<"Am Main"<<endl;
    Add();  //Function Call
    return 0;
}
void Add() //Function Definition
{
    cout<<"Am Add";
}
```

- Function Definition as Declaration

```
void Add()
{
    cout<<"Am Add";
}
int main()
{
    Add();
    cout<<"Am Main"<<endl;
    return 0;
}
```

- Linker Error:
    - If we call a function without giving its definition

```
#include<iostream>
using namespace std;
void Add();
int main()
{
    Add();  //If you call a method without giving its definition then linker will
generate error
    cout<<"Am Main"<<endl;
    return 0;
}
```

- Initialization and Assignment
    - Todo: Do it for variables of type int, char, float etc.

Will be discussed tomorrow (10-12-2024)

- Function Activation Record
- Pointer
    - Concept
    - Declaration
    - Wild pointer
    - Initialization and Assignment
    - NULL and Null Pointer
- Const qualifier
- Constant and Pointer Combination.