Ahmed Bahaa

Lecture #01


# Introduction to Programming Language

# CONTENTS

# [1] History of Machine Language & Programming Languages

Human Language

Human Language

Programming Language

Machine Language
( 0 , 1 )

CPU 1

**ADD** 0000
**SUB** 1001
**DIV** 0010

Assembly 1

CPU 2

**ADD** 0111
**SUB** 1100
**DIV** 1010

Assembly 2

CPU 3

**ADD** 0101
**SUB** 1110
**DIV** 1100

Assembly 3

Assembly Language not unique. So, Human need to study all Assembly Languages to can communicate with all Machines. so, we need common language to make it easy like C / C++

## Low- Level Programming Language

01010010 0101

Machine Code

ADD 2,5

Assembly Language

## Medium - Level Programming Language

X = 2 + 5
printf ( X )

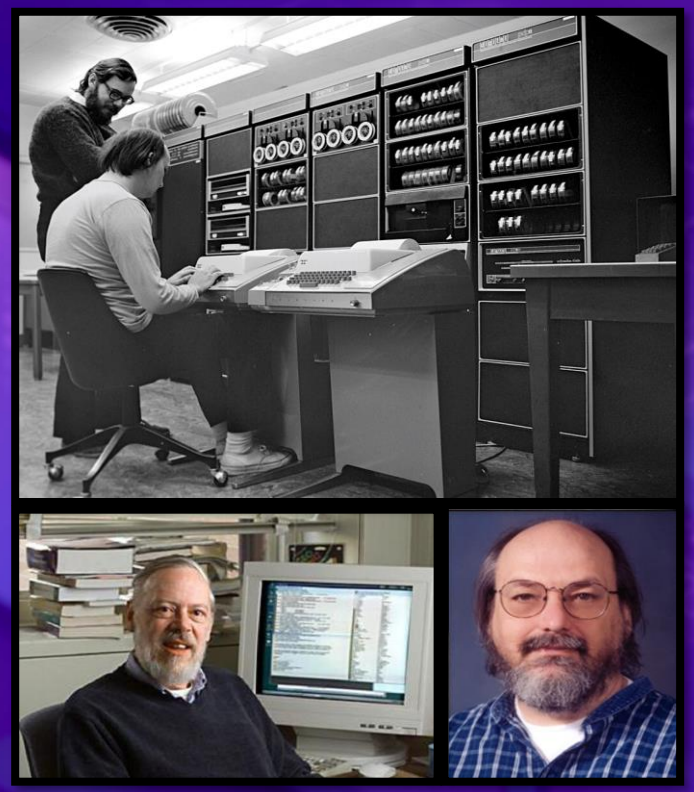
C Language

## High - Level Programming Language

C++, Python, Java, JavaScript

High-Level Programming Language

## [2] History of C Programming Language & C Applications



1960 -> **ALGOL**

1967 -> **BCPL** " Basic Combined Programming Language "

1970 -> ' **B** ' (BCPL + Features)
" **Ken Thompson** " Developed **UNIX** Operating System

1972 -> ' **C** ' (ALGOL + BCPL + B + Features)
" **Dennis Ritchie** " Developed **UNIX** Operating System

1989 -> First standardized specification for ' **C** ' Language was developed by **ANSI** "American National Standards Institute"

1990 -> ' **C89**/**C90** ' Standard C Programming Language from **ISO** " **International Standards Organization** "

1999 -> ' **C99** ' Next version with new features like advanced Data types and other changes.

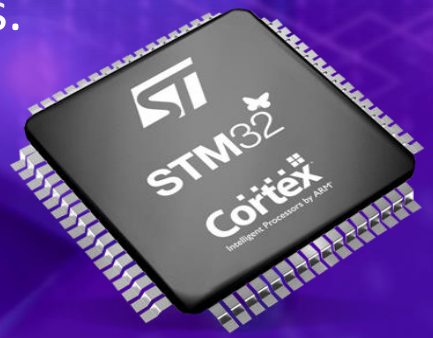
2011 -> ' **C11** ' the current standard for **C** Programming Language.

**Applications in C:**

Embedded Systems, Desktop Applications, Compilers, IOT, Data Base, etc..

## [3] Why C Language in Embedded Systems ?


- **C** is a Middle Level Language.
- **C** Programming Supports inline Assembly language programs.
- Using inline Assembly feature in C, we can directly access system registers.
- **C** Programming is used to access memory directly using Pointers.
- **C** Programming also support High Level Language features.
- It is more user friendly as compared to previous languages, so C is a Middle Level Language.



# [4] Tool Chain and Install Tool Chain & VS Code Text Editor

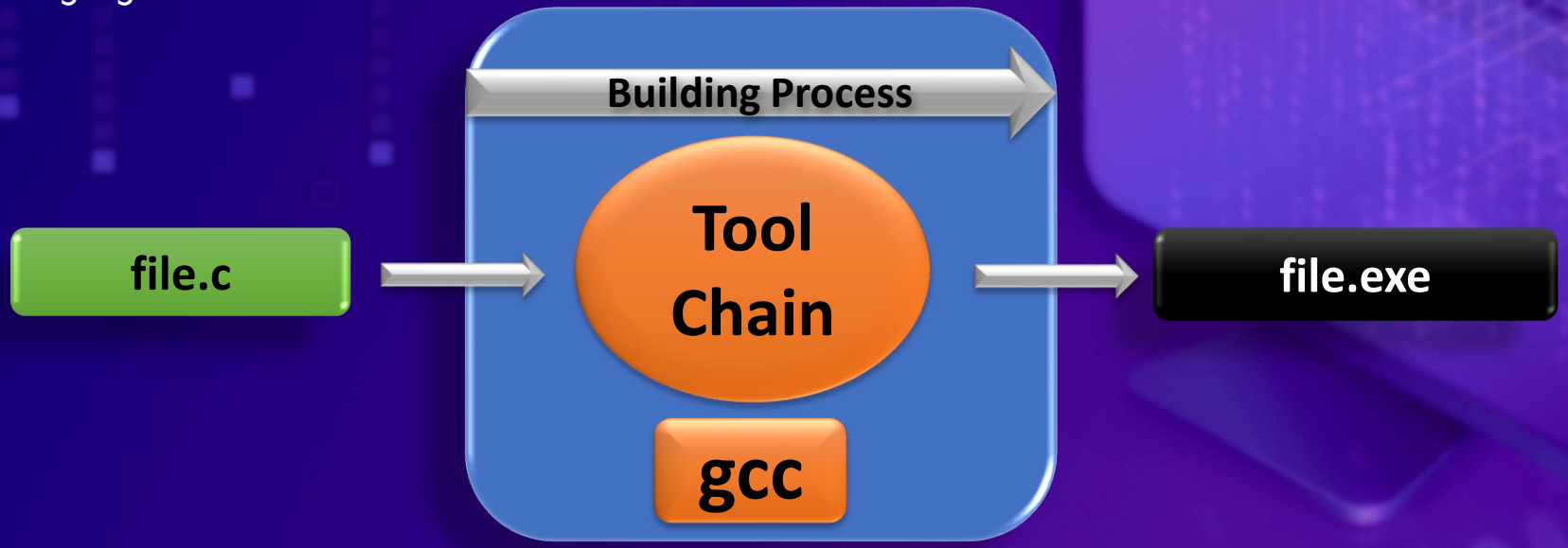
Human Language

C Programming Language

Translator

Binary Code

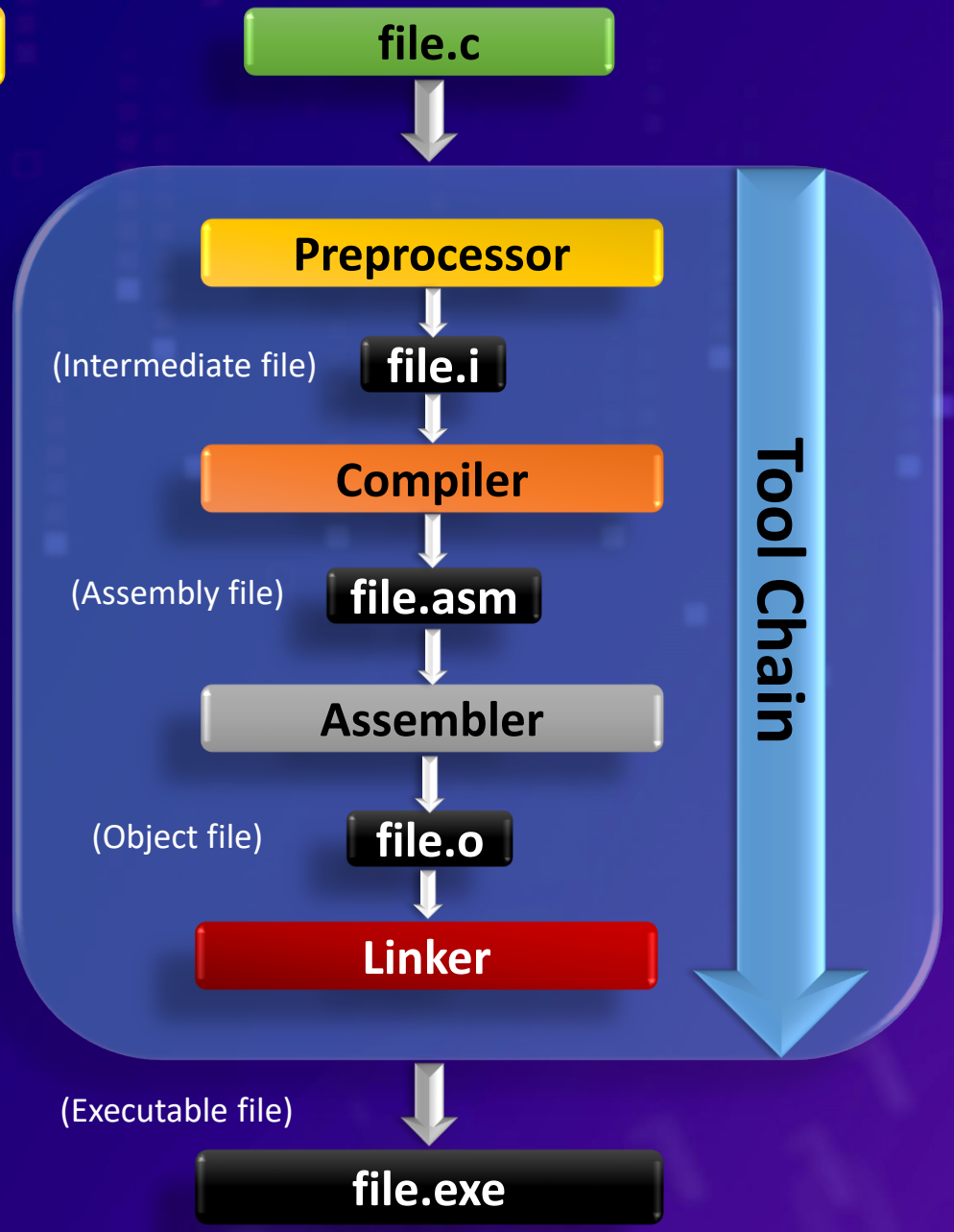
Machine Language ( 0 , 1 )




**Tool Chain:**

A tool chain is a set of tools that are used to building process so it for convert the source file written in C Language, to the executable file written in machine language ( **0** and **1** ) to be run on the processor.

# Tool Chain




cmd

```
gcc -E file.c -o file.i
```

```
gcc -S file.c -o file.asm
```

```
gcc -c file.c -o file.o
```

```
gcc file.c -o file.exe
```

# Installing [gcc] Tool Chain

1. Download Tool Chain from this Link https://drive.google.com/drive/folders/1sUPNlik4rcG3fOKjAljhIr3OhN3CUq6V?usp=sharing

2. Extract Here for (mingw-w32) or (mingw-w64) related to your laptop.

3. Open (mingw-w64) and Copy PATH for (bin) folder.

4. Paste this PATH in the Path System Environmental Variables.

5. Test the gcc installation -> Open cmd and write (gcc --version).

**Installation is Correct**

```
C:\Users\Ahmed Bahaa>gcc --version
gcc (x86_64-posix-seh-rev1, Built by MinGW-W64 project) 6.2.0
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

**Installation is Not-Correct**

```
C:\Users\Ahmed Bahaa>gcc --version
'gcc' is not recognized as an internal or external command,
operable program or batch file.
```

6. If installation is Not Correct, Please Restart your Laptop and Repeat this cycle again

7. Now Download VS Code Text Editor from this Link Download Visual Studio Code - Mac, Linux, Windows

8. Install VS Code Text Editor and Create file.c and open it in VS Code and write your first code in C

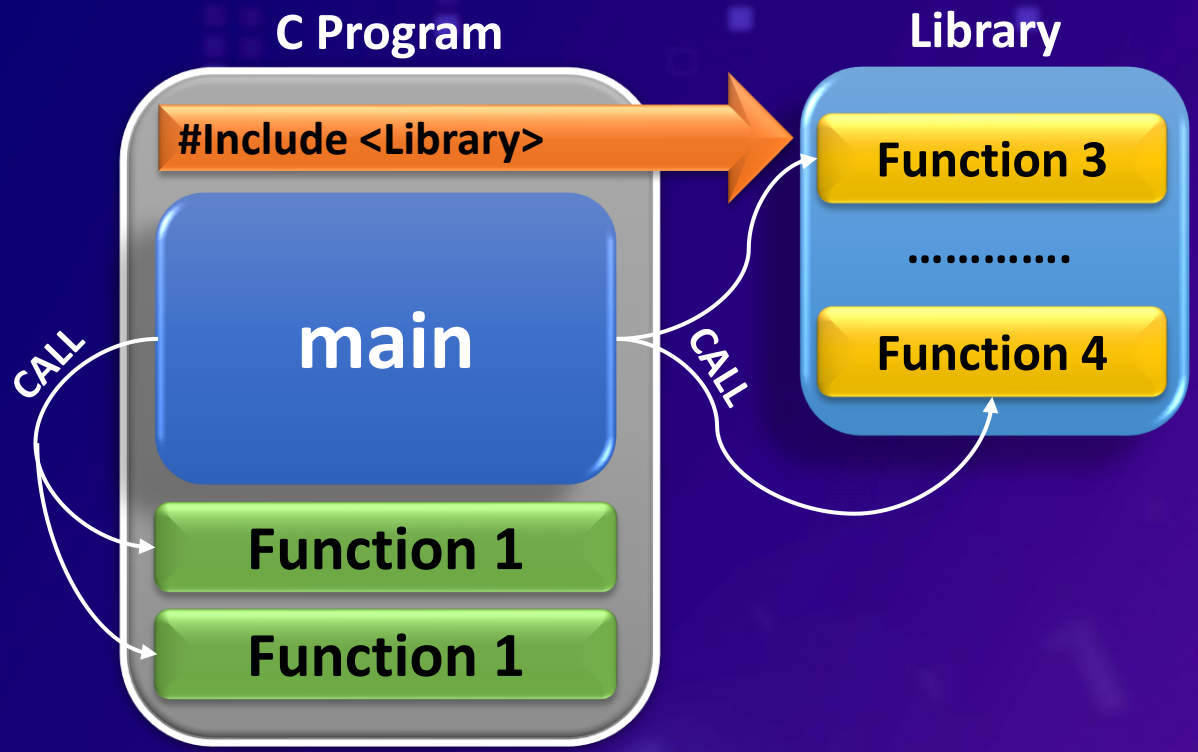
9. Open the folder of your file.c and open (**cmd**) and write this command to get file.exe


```
gcc    file.c   -o   file.exe
```

10. To display your output, Just write your output name in (**cmd**) > **file.exe**

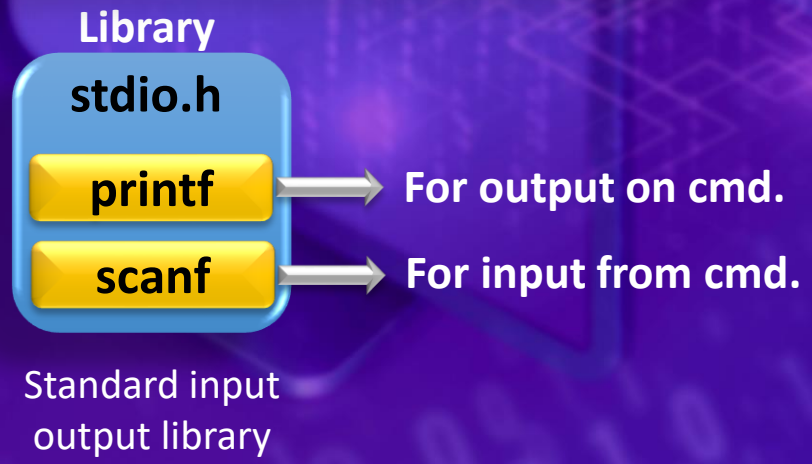
## [5] C Program Structure

- C is structured programming, it means that the C program is composed of small parts each part called *"function"*.
- The first function to be executed (The entry point of the program) is called *"main"*.
- Sometimes, we may write some functions in a stand-alone file for organizing, this file is called *"library"*.




## Inputs/Outputs Library in C <stdio.h>

In **C** Programming we have a Library for inputs and outputs " **Standard input output library** " We need to include **< stdio.h >**




Standard input output library

# [6] First C Program "Hello World"

```c
#include <stdio.h>

void main()
{
    printf("Hello world");
}
```

Standard input output library

Library

stdio.h

printf

scanf

Entry Point

For output on cmd.

```
Hello world
```

Any Line inside a function
must ends with " Semicolon ; "

## [7] Comments in C

- Comments are non-executable text used to provide documentation for the code.
- It provide clarity to the C source code allowing others to better understand what the code was intended to accomplish.
- It is always recommended to use comments in your code, at least one comment for each code line.

### Single Line Comment

Any line preceded by two forward slashes **//**

```
// This is Single Line Comment
```

### Multi Line Comment

Any text starts with **/*** and ends with ***/**

```
/*  This is
    Multi Line
    Comment   */
```

```c
/*  Multi Line Comment
    include stdio Library
    to use printf Function  */
#include <stdio.h>

// Entry Point
void main()
{
    // Calling printf Function to Print on cmd
    printf("  Hello world");
}
```

## [8] Escape Sequences

| | |
|---|---|
| \n | New Line Position the curser at the beginning of the next line. |
| \t | Horizontal Tab |
| \v | Vertical Tab |
| \a | Produce Sound (Windows Alert). |
| \\ | Insert a Backslash character in a String. |
| \” | Insert a Double-Quote character in a String. |
| \’ | Insert a Single-Quote character in a String. |
| %% | Insert a Modulus % in a String. |
| \b | Backspace |
| \0 | NULL |

```
Ahmed  Bahaa
Ahmed  Bahaa
AhmedBahaa
Ahmed"Bahaa"
Ahmed'Bahaa'
Ahmed\Bahaa\
AhmedBahaa
```

```c
#include <stdio.h>
void main()
{
    printf("Ahmed Bahaa    \n");
    printf("Ahmed\tBahaa   \n");
    printf("Ahmed\aBahaa   \n");
    printf("Ahmed\"Bahaa\" \n");
    printf("Ahmed\'Bahaa\' \n");
    printf("Ahmed\\Bahaa\\ \n");
    printf("Ahmed \bBahaa  \n");
}
```

LAB.01

# Write a C code that will print your Biography

Expected Output

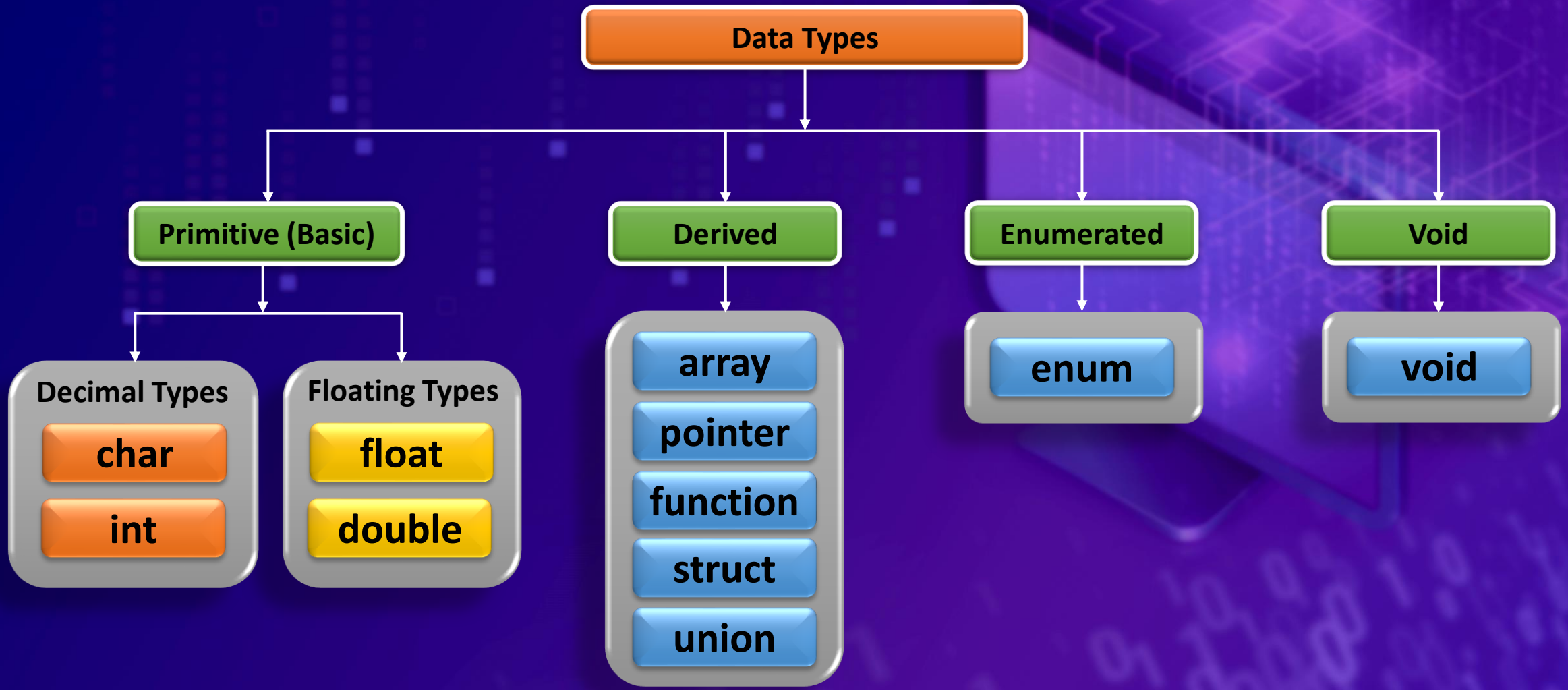
```
I'm Ahmed Bahaa

My Birth date is 20 Aug 1997

I Graduated from Mechatronics Engineering

HTI - I Graduated at 2020
```

# [9] Data Types in C



# Primitive Data Types (Basic)

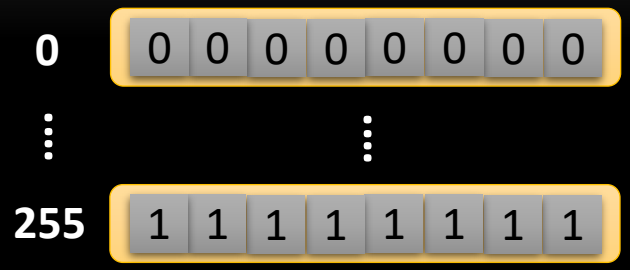
## Decimal Types

### char

**1 Byte → 8 Bit**

Size = 2 Power (8) = 256

Numbers Range = 0 -> 255

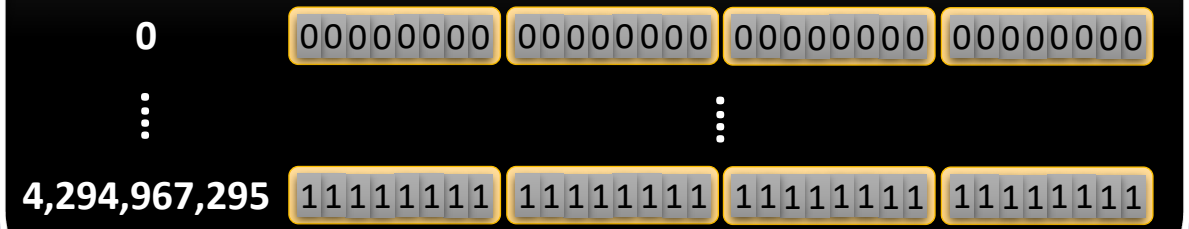



### int

**4 Byte → 32 Bit**

Size = 2 Power (32) = 4,294,967,296

Numbers Range = 0 -> 4,294,967,295




## Floating Types

| | |
|---|---|
| float | 4 Byte → 32 Bit |
| double | 8 Byte → 64 Bit |

Size is Compiler dependent

Some compilers have **int** = 4 byte

And some others have **int** = 2 byte

# [10] Variables in C

Variable is a part from memory used to hold a piece of Data.

## Variable Declaration

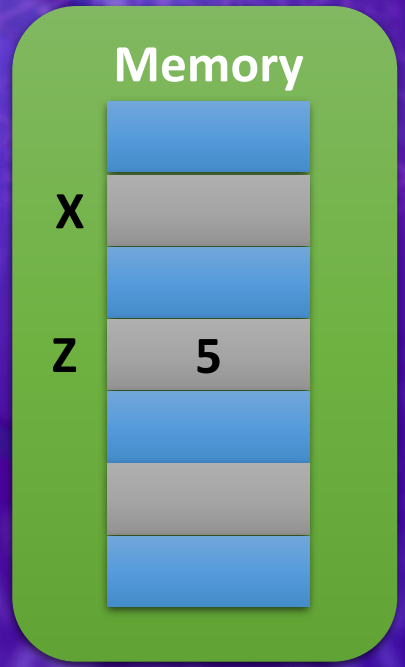
Hints the Compiler about the Type, Name and Size of the variable in compile time, and No Data in memory Location for this variable.

```
Data_Type   Variable_Name   ;   ⟶   int   X   ;
```

## Variable Definition

Means Declaring the variable and allocating space in memory to hold the Data.
Definition = Declaration + Memory Reservation with initial Data

```
Data_Type   Variable_Name   =   initial_Data   ;   ⟶   int   Z   =   5   ;
```



# [10] Variable Naming Rules

(1) Variable Name can contain:

- Capital Litters **A, B, C …. Z**
- Small Litters **a, b, c …. z**
- Numbers **0, 1, 2 …. 9**
- Underscore **_**

(2) First character must be Alphabets or Underscore:

- int 1B ; **Not Allowed**
- int A1 ; **Allowed**
- int _Bahaa ; **Allowed**

(3) Variable Name shouldn't be Keyword in C:

- int main ; **Not Allowed**
- int char ; **Not Allowed**
- int int ; **Not Allowed**

(4) Variable Name can not be repeated in the same scope { }:

```
{
    int X ;
    char X ;      Not Allowed " Multi Definition Error "
}
```

(5) No Special Symbols other than Underscore Allowed ( **#, $, ?, *, &,** …. ):

- int A#B ; **Not Allowed**
- int A_B ; **Allowed**

(6) Blanks and Commas are Not Allowed:

- int Ahmed Bahaa ; **Not Allowed**
- int Ahmed,Bahaa ; **Allowed** but that's mean **2** Variables " **int Ahmed ;** " & " **int Bahaa ;** "

# What will be the Output of the following code.. ?

```c
#include <stdio.h>
void main()
{
    // Define int variable and initialize with 5
    int X = 5 ;
    printf("The Variable Value = X \n" );
}
```

OUTPUT

```
The Variable Value = X
```

**The String will be Printed as it is.**
**It will not replace X with its Value.**
**Printed X as a normal character not a variable.**
**We need to know how to print value of  variable.**

# Printing a Variable

***printf*** function can print a variable inside the string, it could be done by inserting **format specifier** that will be replaced by the values specified in subsequent additional arguments.

```c
#include <stdio.h>
void main()
{
    // Define int variable and initialize with 5
    int X = 5 ;
    printf("The Variable Value = %d \n", X );
}
```

**OUTPUT**

```
The Variable Value = 5
```

**This Format Specifier will be Replaced by the Value of X**

## [11] Format Specifier

| | |
|---|---|
| %x | Format Specifier for **HEXADECIMAL** value. |
| %o | Format Specifier for **OCTAL** value. |
| %p | Format Specifier for **Pointer** or **Address**. |
| %i | Format Specifier for **Integer** value. |
| %hd | Format Specifier for **Short integer**. |
| %ld | Format Specifier for **Long integer**. |
| %lld | Format Specifier for **Long Long integer**. |
| %u | Format Specifier for **unsigned integer** = **Decimal**. |
| %hu | Format Specifier for **Short unsigned integer**. |
| %lu | Format Specifier for **Long unsigned integer**. |
| %llu | Format Specifier for **Long Long unsigned integer**. |
| %lf | Format Specifier for **double**. |
| %LF | Format Specifier for **Long Double**. |

### Common Format Specifiers

| | |
|---|---|
| %d | Format Specifier for **Decimal** value. |
| %f | Format Specifier for **Floating** value. |
| %c | Format Specifier for **Character** value. |
| %s | Format Specifier for **Strings**. |

### Example:

```c
#include <stdio.h>
void main()
{
    int   X = 100 ;
    float Y = 5.7 ;
    char  Z = 'A' ;
    printf(" Decimal   Value of X = %d   \n",X );
    printf(" Floating  Value of Y = %.2f \n",Y );
    printf(" Character Value of Z = %c   \n",Z );
}
```

```
Decimal   Value of X = 100
Floating  Value of Y = 5.70
Character Value of Z = A
```

Number of Digits after **. dot**

# Scanning a Variable

***scanf*** function is a part from the <***stdio.h***> library, it is used to get value from the user and save it in a variable

**Syntax:**

```
Scanf ( " Format_Specifier " , & Variable_Name ) ;
```

**Address Operator**
Plz, Don't Forget it

**Example:**

```c
#include <stdio.h>
void main()
{
    int  X  ;
    // Get the value from user
    scanf("%d", &X );
}
```

LAB.02

# Write a C code that will ask the user to entre a value then print it

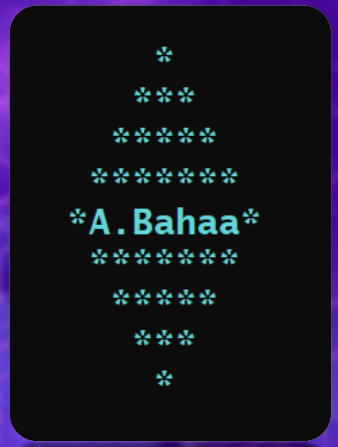
Expected Output

```
Please Entre the Value : 200

The Value you entered is = 200
```

## Assignment.01

# Write a C code that Draw this Shape




## Assignment.02

# Write a code that scan 3 numbers from the user and print them in reversed order

```
Please enter Number 1 : 10
Please enter Number 2 : 20
Please enter Number 3 : 30
Number 3 = 30
Number 2 = 20
Number 1 = 10
```

# THANKS

Ahmed Bahaa