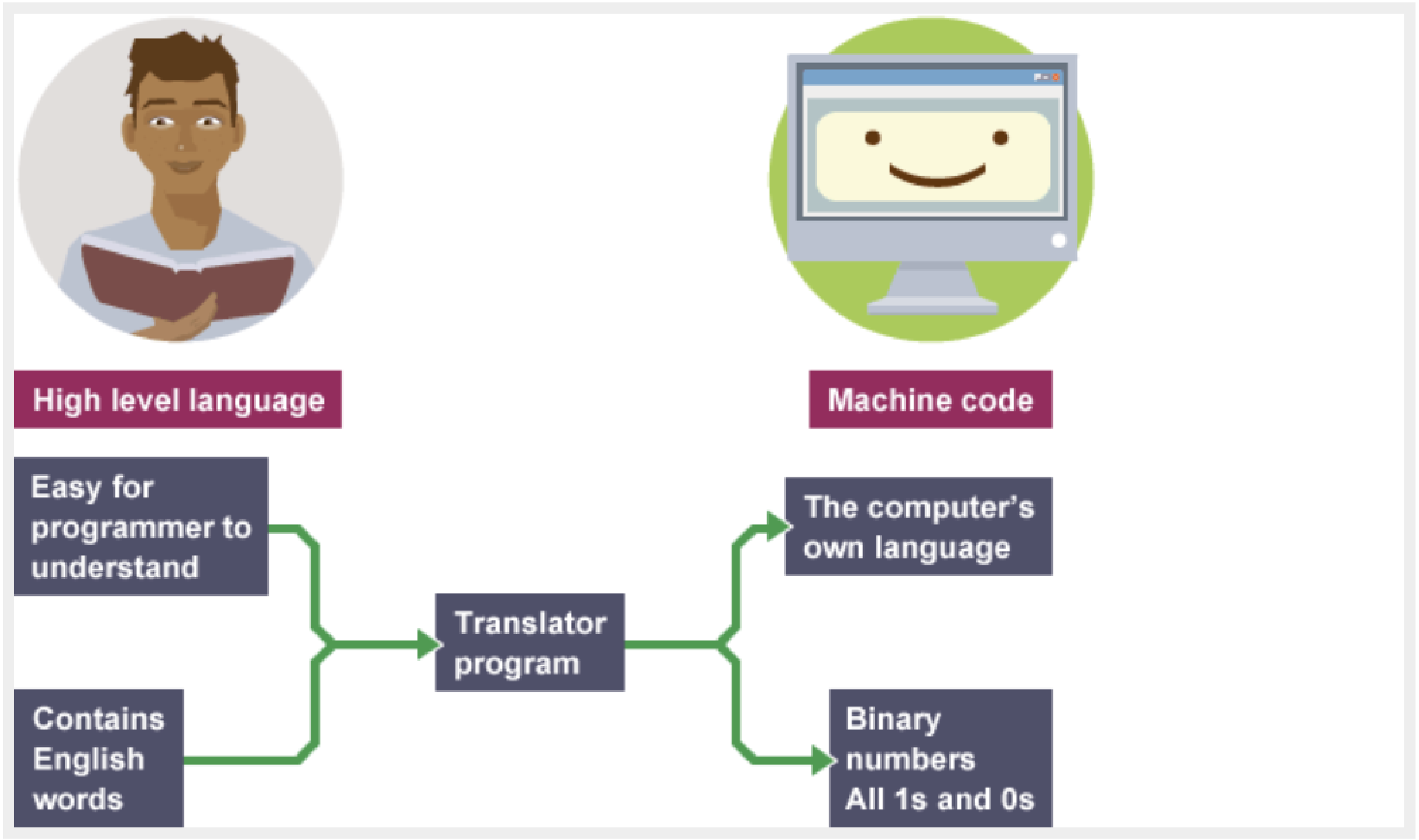
Introduction to C

Introduction

- A computer system does not understand a human language, but a machine language i.e. binary language made of 0s and 1s(low level/machine language).
- Hence, to make a computer capable of running a human formulated algorithm, it is necessary to make the computer understand & execute the algorithm.
- Programming languages are used by humans to implement the algorithms, also known as High level languages.
- A programming language is a set of commands and instructions used to create a software program.


High level language
Machine code
Easy for programmer to understand
Contains English words
Translator program
The computer's own language
Binary numbers All 1s and 0s

Structure of a C program

- C programming is a language developed at AT&T Bell laboratories of USA in 1972, designed and written by ”Dennis Ritchie”.
- Starting with a program. The program consist of :
  - Headers -> Include header files which contain definition of the functions used inside a program.
  - Body -> Here the logic of the code is written meant to serve the purpose of the program

```
#include<stdio.h>
void main()
{
        printf(“Hello World!!”);
}
```

Example program contd..

- Lets build the program by ourselves.

PROGRAM

- Write a program to calculate the sum of two numbers: 13,54 and store in a variable and display result
  - **Input**: None
  - **Processing**: assign sum of the two numbers to variable named *sum*
  - **Output**: Print out value of the variable *sum*

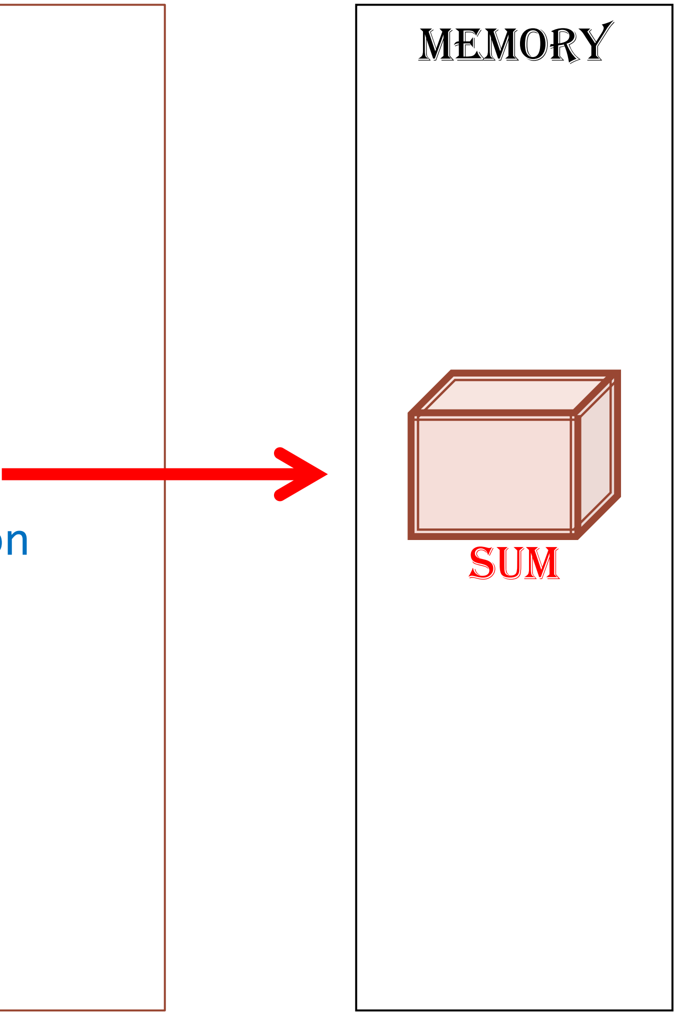
```
#include <stdio.h> //preprocessor
directive

int main() //main function


{
    int sum; //variable declaration
    //Why we need variable declaration




    return 0;
}
```



hitbullseye

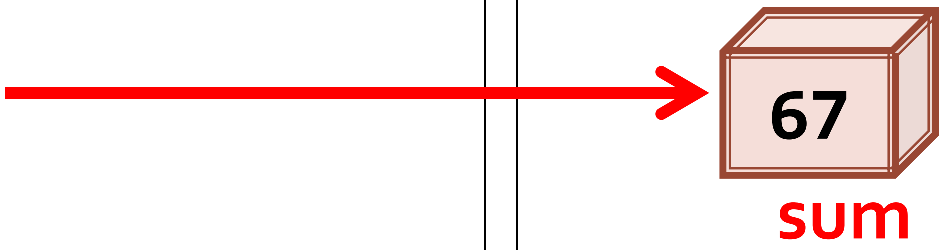
```
#include <stdio.h> //preprocessor directive

int main() //main function

{
    int sum; //variable declaration

    sum = 13 + 54; //processing

    return 0;
}
```



```
#include <stdio.h> //preprocessor directive

int main() //main function


{
    int sum; //variable declaration



     sum = 13 + 54; //processing



    printf("The value of the sum is %d", sum);
//output

       return 0;
}
```

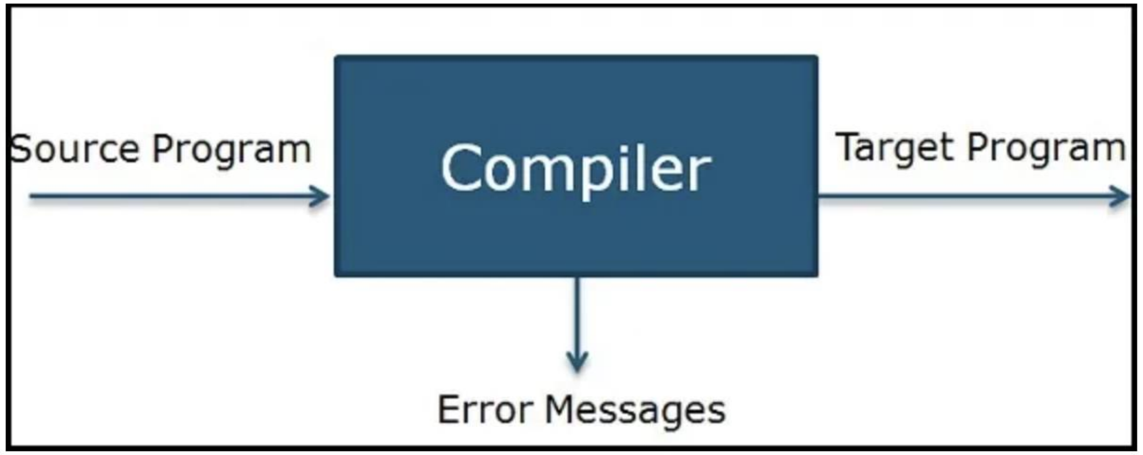
RUN WINDOW

```
The value of the sum is 67
```

hitbullseye

Compiler

- **Compilation:** Translation of a program written in a source language into a semantically equivalent program written in a target language
- **Compiler:** A program that reads a program written in one language and translates it into another language.



Language Levels

High Level Code

- High-level language is a computer language which can be understood by the users.
- The high-level language is very similar to human languages and has a set of grammar rules that are used to make instructions more easily.
- Every high-level language has a set of predefined words known as Keywords and a set of rules known as Syntax to create instructions.
- The high-level language is easier to understand for the users but the computer can not understand it.
- High-level language needs to be converted into the low-level language to make it understandable by the computer. We use **Compiler** to convert high-level language to low-level language.

Language Levels Cont...

Assembly Level Code(middle level language)

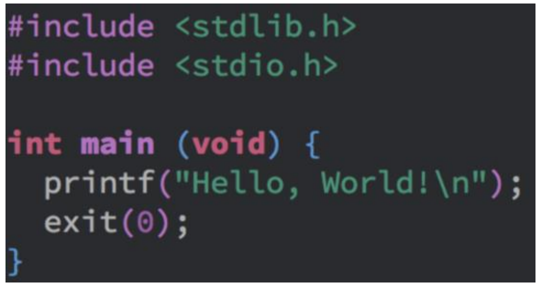
- Middle-level language is a computer language in which the instructions are created using symbols such as letters, digits and special characters.
- **Assembly language** is an example of middle-level language. In assembly language, we use predefined words called mnemonics.
- Binary code instructions in low-level language are replaced with mnemonics and operands in middle-level language.
- But the computer cannot understand mnemonics, so we use a translator called **Assembler** to translate mnemonics into machine language.
- Assembler is a translator which takes assembly code as input and produces machine code as output.
- That means, the computer cannot understand middle-level language, so it needs to be translated into a low-level language to make it understandable by the computer.
- Assembler is used to translate middle-level language into low-level language.

Language Levels Cont…

Low Level Code

- Low-Level language is the only language which can be understood by the computer.
- Low-level language is also known as **Machine Language**.
- The machine language contains only two symbols **1 & 0**.
- All the instructions of machine language are written in the form of binary numbers 1's & 0’s.
- A computer can directly understand the machine language.

Compilation Process



```
#include <stdlib.h>
#include <stdio.h>

int main (void) {
  printf("Hello, World!\n");
  exit(0);
}
```

High Level Language

- Code written using programming Language Constructs

Preprocessor

- Removes comments, Link the libraries, macros expansion - Convert code into pure high level code

Compiler

- Convert Code into Assembly Level Code

Assembler

- Convert code into low level form i.e. Machine Language Code

Linker/ Loader

- Link the libraries and functions, Loads program into main memory

Writing first C program

- To write the first c program, open the C console. C console can be any IDE like CodeBlocks, Dev++, or install gcc compiler and write programs on notepad with extension .c .
- Write the following code

```
#include <stdio.h>
int main(){
printf("Hello C Language");
return 0;
}
```

- **#include <stdio.h>** includes the **standard input output** library functions. The printf() function is defined in stdio.h.

Writing first C program

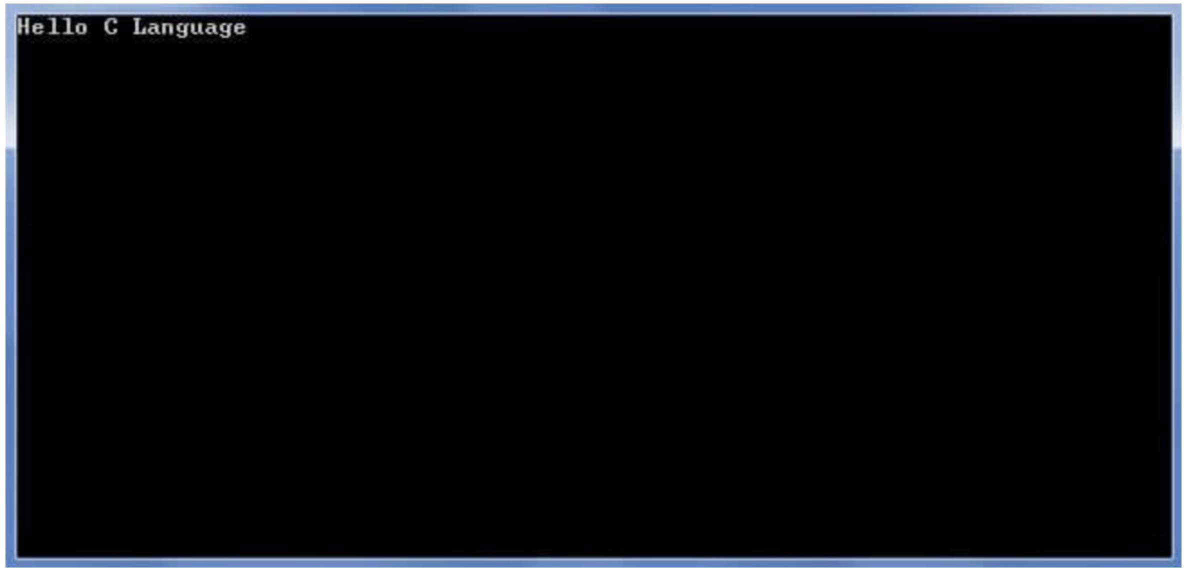
```
#include <stdio.h>
int main(){
printf("Hello C Language");
return 0;
}
```

- **int main()** The **main() function is the entry point of every program** in c language.
- **printf()** The printf() function is **used to print data** on the console.
- **return 0** The return 0 statement, returns execution status to the OS. The 0 value is used for successful execution and 1 for unsuccessful execution.

Writing first C program

Run the program

- **Click on the compile menu then compile sub menu** to compile the c program in the IDE.
- Then **click on the run menu then run sub menu** to run the c program.

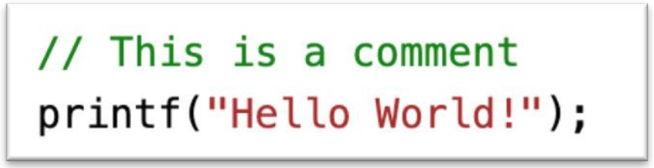


```
Hello C Language
```

Comments in C Program

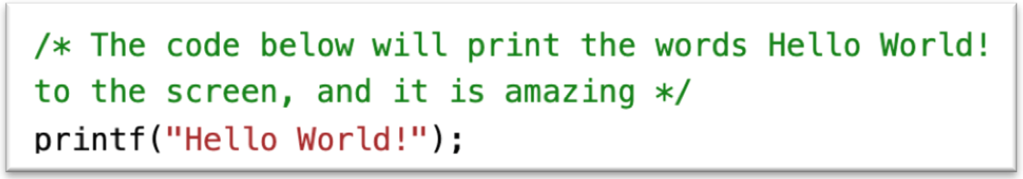
- Comments can be used to explain code, and to make it more readable. It can also be used to prevent execution when testing alternative code.
- Comments can be **singled-lined** or **multi-lined**.
  - ✓ **Single-line** comments start with two forward slashes (//). Any text between // and the end of the line is ignored by the compiler (will not be executed).
  - ✓ **Multi-line comments** start with /* and ends with */.

  Any text between /* and */ will be ignored by the compiler.



```
// This is a comment
printf("Hello World!");
```

Single line comment



```
/* The code below will print the words Hello World!
to the screen, and it is amazing */
printf("Hello World!");
```

Multi-line comment

References Link

- ✓ https://www.w3schools.com/c/index.php
- ✓ https://archive.nptel.ac.in/courses/106/104/106104128/
- ✓ https://www.tutorialspoint.com/cprogramming/index.htm

THANK YOU

hitbullseye