Programing Methodology in C

Lecture 3 – Program Writing in C Programing language

By Elubu Joseph

josebulinga@gmail.com

Agenda

- Recap on code transformation to a program
- Writing a program in C
- C Program Structure/Layout
- C Language Syntax Rules

Analysis of the life of a program from Conception, Coding to Execution

Where dose Programs Start?

Programs start out as an idea in a programmer's head/Mind. He uses a text editor e.g. *code blocks* to write his thoughts into a file called a *source file*, containing *source code.*

This file is transformed by the ***compiler*** into Assembly code which the ***assembler*** then transform to an ***object file***. Next, a program called the ***linker*** takes the object file, combines it with predefined routines from a ***standard library***, and produces an *executable program* (a set of machine-language instructions), (Steve O, 1997).

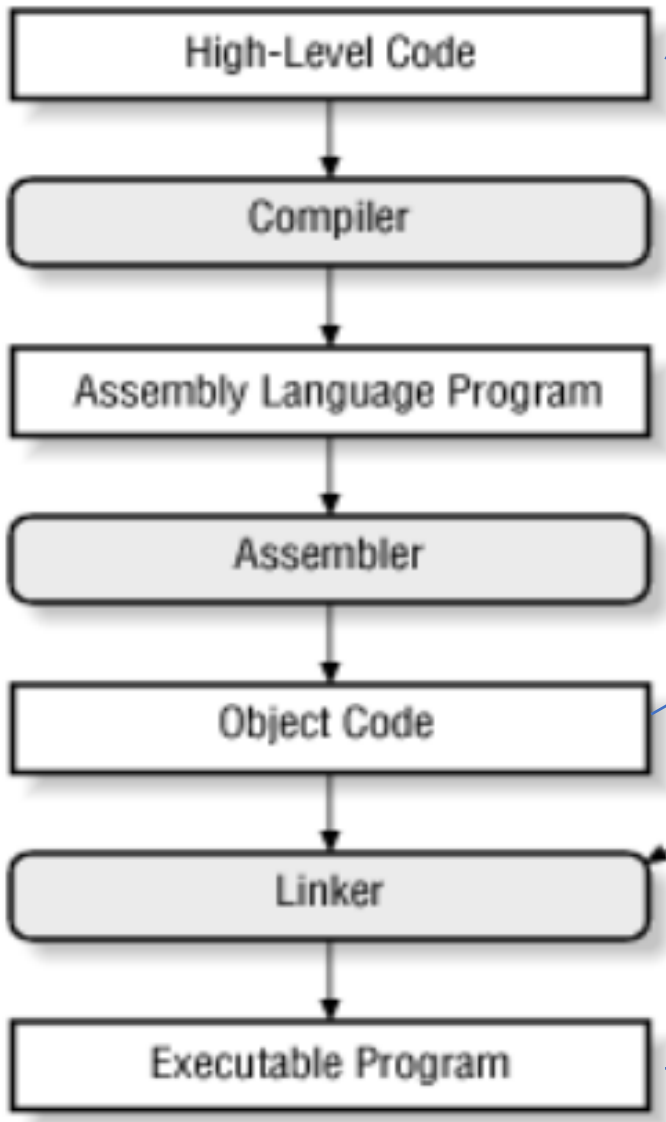
Analysis of the life of a program from Conception, Coding to Execution +

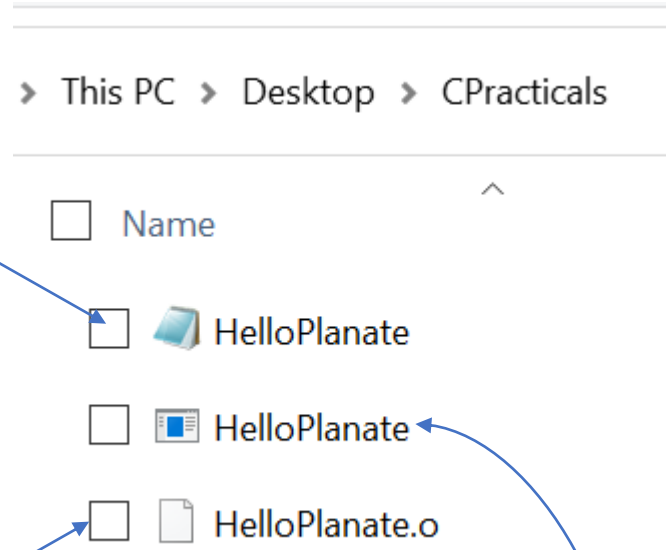
C programs are written in a high-level language using letters, numbers, and the other symbols you find on a computer keyboard.

Computers actually execute a very low-level language called *machine code* (a series of numbers e.g. 11001001 or 10101011- bits). So, before a program level can be used, it must undergo several transformations.

In the following sections, we'll see how these various forms of the IDE program components work together to produce the final program. **See the figure below**

Transformation of a high-level language code into a program



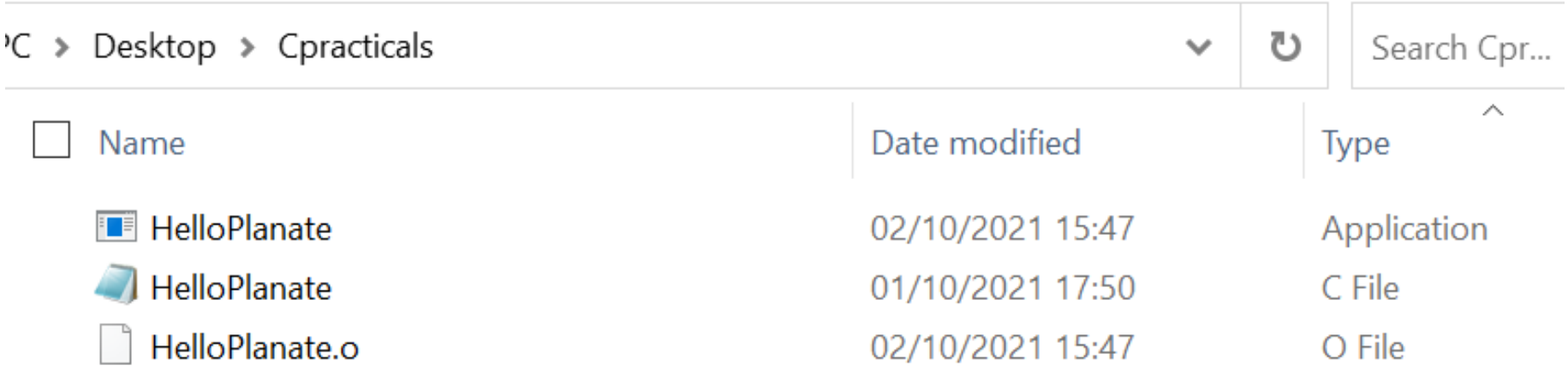


Library

(Steve O, 1997)

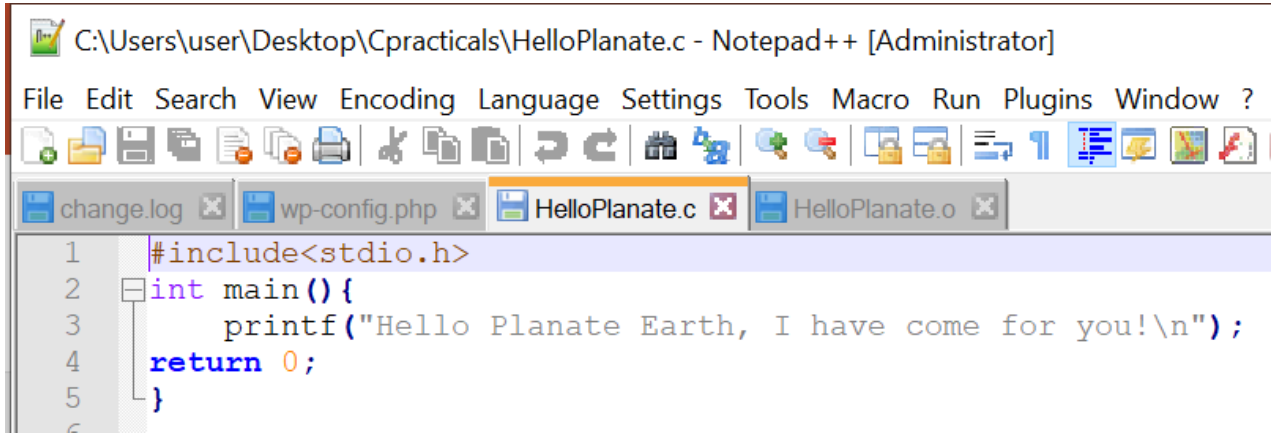
Files produced after Program Transformation

After we transformed our source code to executable program, the transformation systems produced for us three files

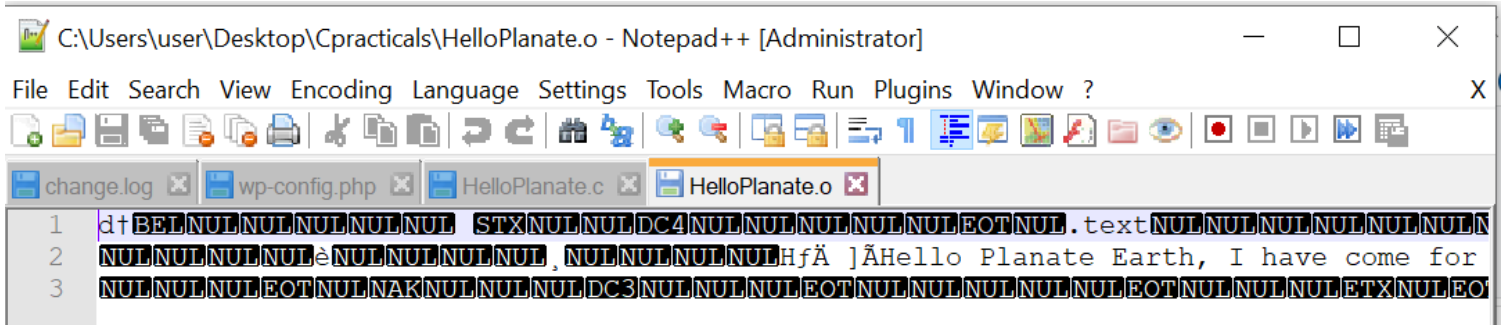


Lets open each of the above files

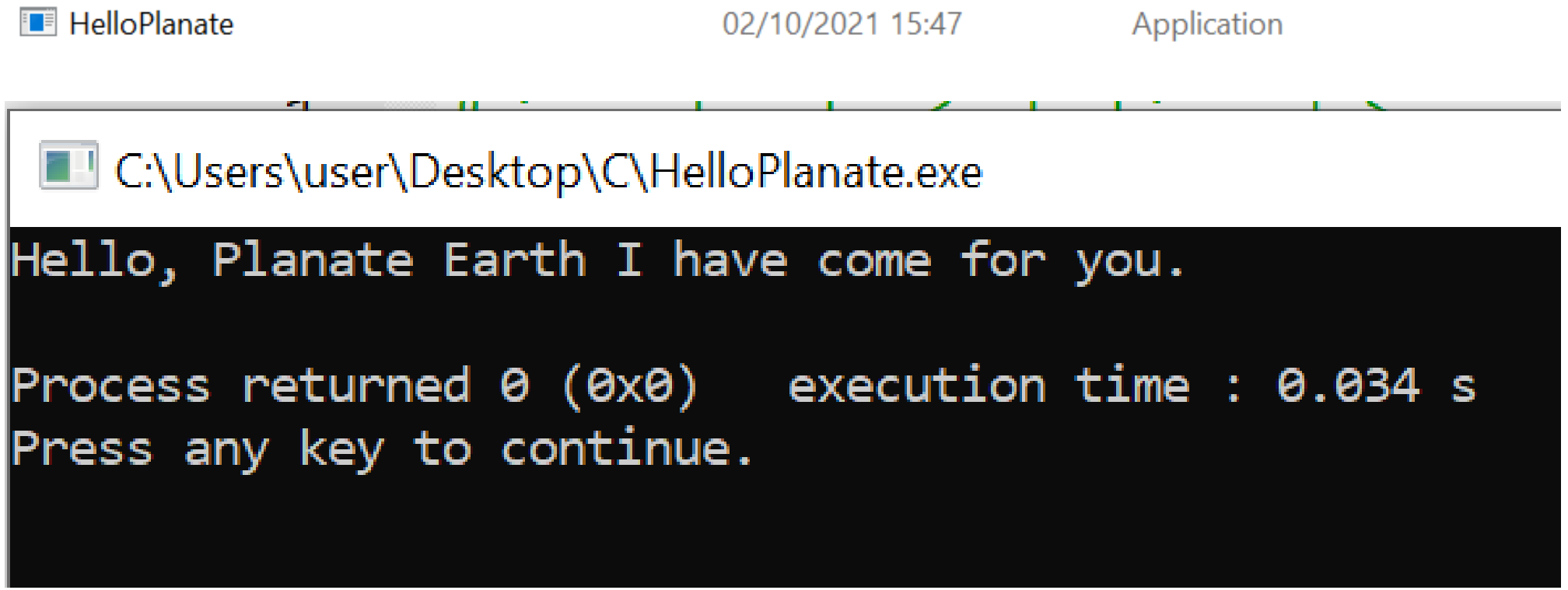
HelloPlanate 01/10/2021 17:50 C File



HelloPlanate.o 02/10/2021 15:47 O File



Lets open each of the above files



Writing a program

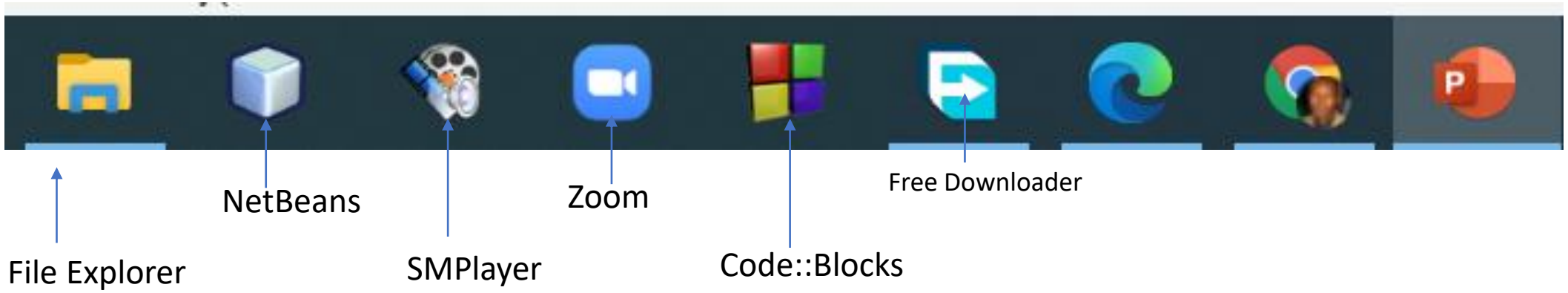
A *program* is a set of step-by-step instructions that directs the computer to do the tasks a user want it to do and produce the results you want.

These instructions are written by a programmer and the process of putting this instructions to together is writing/coding a program.

Writing a program +

You have used a number of programs including any presentation program you are using to view this presentation or watching our lectures videos. They were all written by someone.

Common programs such as:



etcetera were all coded using various programing languages.

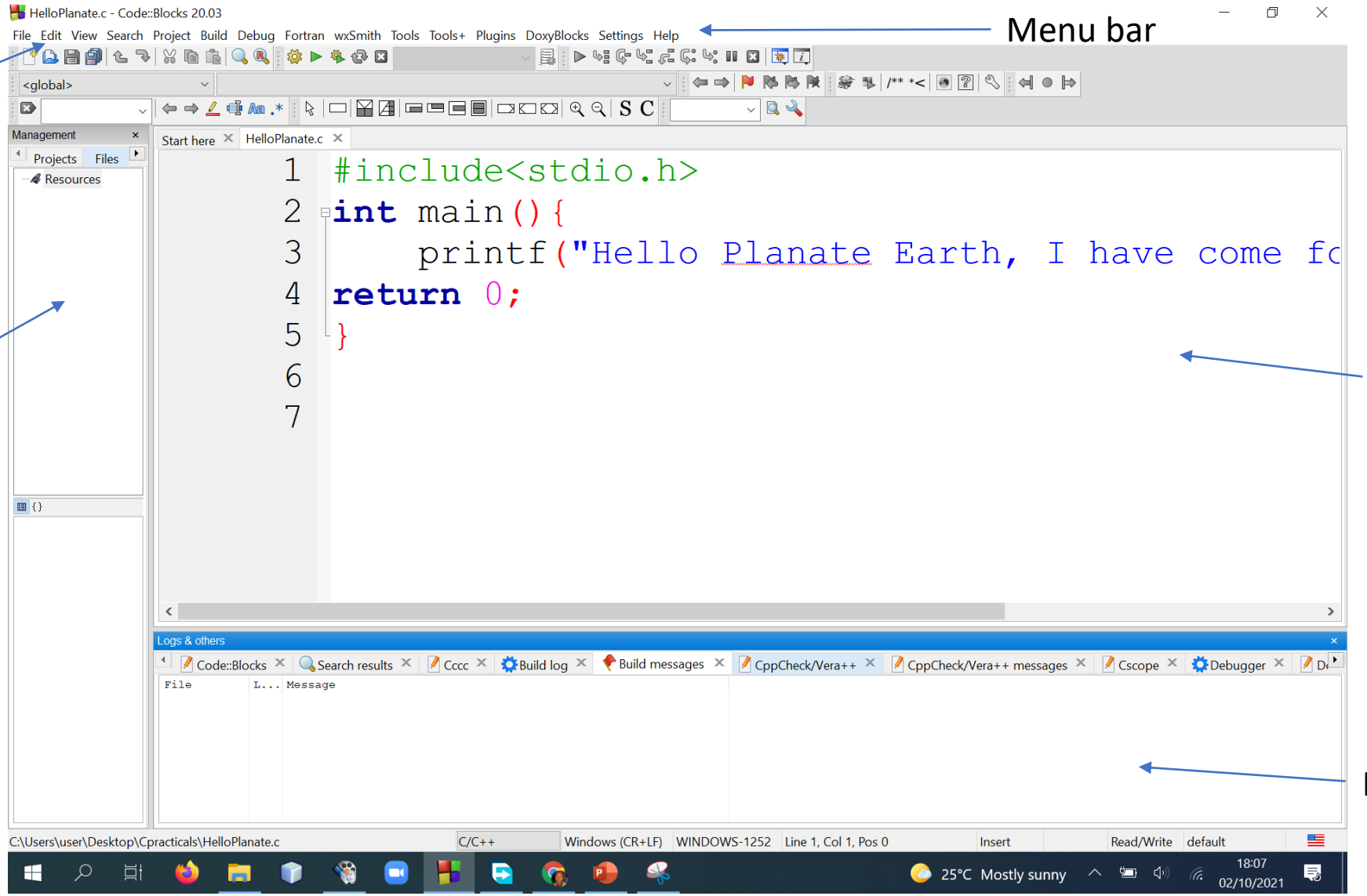
Tools needed for writing a program in C

Refer to Lecture 2 of this course.

We shared need for a fully fledged IDE where we downloaded and installed **Code::Blocks,** Configured it, Used it to write, built and ran our first program.

Now all you need to know is the use of some few important parts of Code blocks to enable you navigate through you develops softly.

Important parts of Code Blocks



Quick access bar

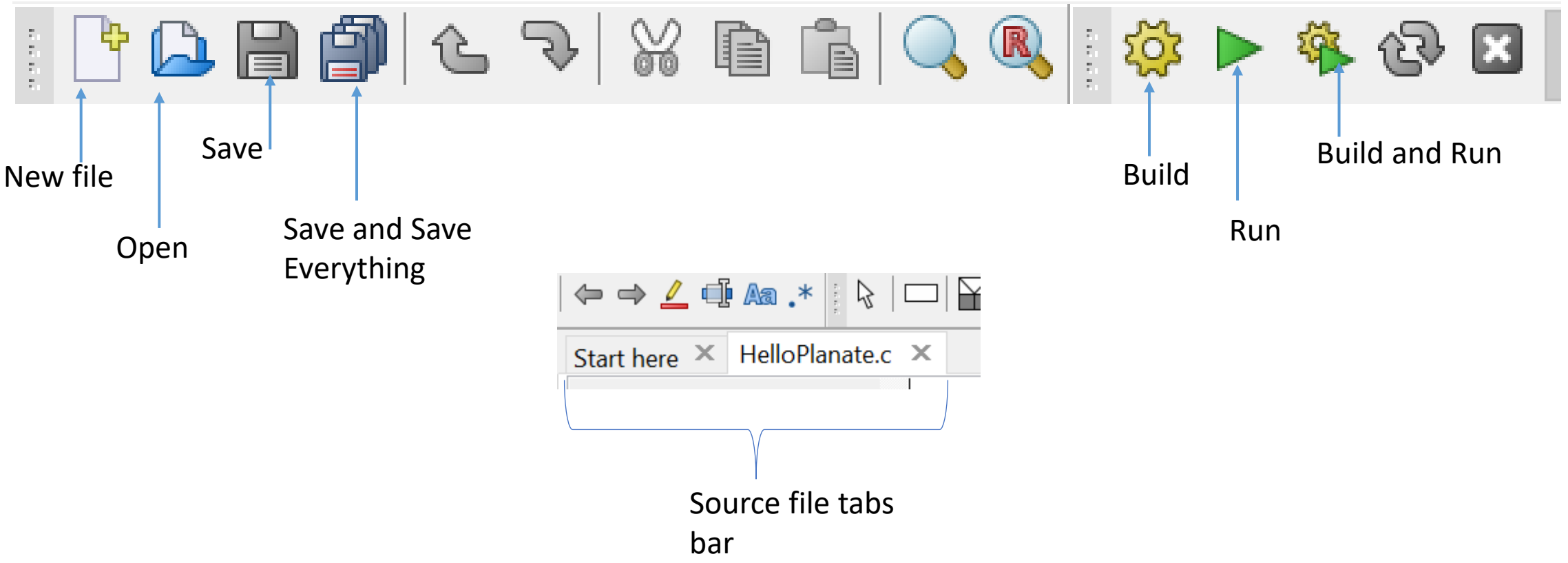
Menu bar

Management section

Source Code Editor

Messages section

Frequently used buttons on Quick access bar



Difference between Compile and Run

You must be thinking why it is a 2 step process, first we compile the code and then we run the code. So, compilation is the process where the compiler checks whether the program is correct syntax wise, and there are no errors in the syntax.

When we run a compiled program, then it actually executes the statements inside the main() function.

With Build, we can’t see the expected output which in this case should be Hello KUMU

```c
#include <stdio.h>
int main() {//Note that int is is being used by main


printf(" Hello, KUMU \n");

return 0; /* Note zero is expected*/
}
```

Run can only produce the result of a Build Code in this case it will be “Hello KUMU” if the program was build before running it.

```c
#include <stdio.h>
int main() {//Note that int is is being used by main


printf(" Hello, KUMU \n");

return 0; /* Note zero is expected*/
}
```

Should you Edit the program and then don’t build, you will not see the results of the new code.

```
#include <stdio.h>
int main() {//Note that int is is being used by main


printf(" Hello, KUMU \n");

Printf (" New code here, at KUMU \n");

return 0; /* Note zero is expected*/
}
```

C Pogroming Language Structure/Layout

C Programing language structure/Layout

For one to be able to speak, write or read any language, one must study the structure, rules and other mandatory components of such a language.

Like any other language, C has its own structure that must be put in its place for the language to be utilize to the fullest.

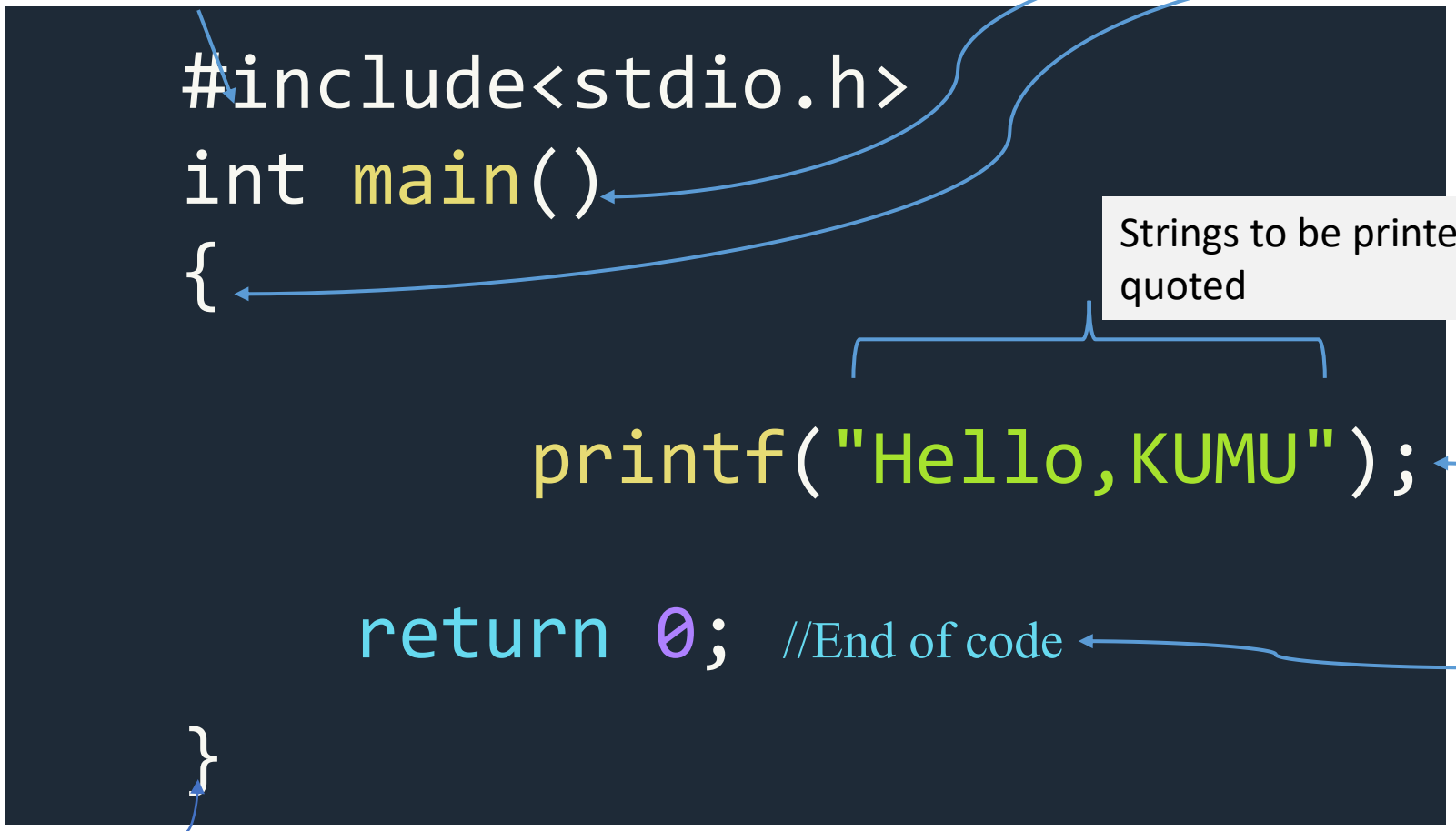
Structure in this context means the **basic layout** of the C programming language. Which layout gives as various parts of C. see the program.

C Program Structure/layout

Lets see how to write a simple and most basic C program.

To do this we must consider including every important part of the program in the layout.

C Program Structure/Layout



Parts of C program

1. Pre-processor
2. Header file/Library file
3. Function
4. Program body
5. Variables
6. Statements & expressions
7. Comments

All these are essential parts of a C language program.

1. Pre-processor

#include is the first word of any C program. It is also known as a **pre-processor**.

The pre-processor initializes the environment of the program, by linking the program source code with the header/library files required.

So, when we say #include <stdio.h>, we are informing the compiler to link the **stdio.h** header file to our code before executing it.

2. C Header file/ C Library file

is a collection of built-in(readymade) functions, which we can directly use in our program. These files contain definitions of the functions which can be incorporated into any C program by using pre-processor #include statement with the header file.

Standard header files are provided with each compiler, and covers a range of areas like string handling, mathematical functions, data conversion, printing and reading of variables.

2. C Header files/Library files +

Header files are readymade piece of source files containing functions which comes packaged with the C language that you can use without worrying about how they work, all you have to do is include the header file in your program.

To use any of the standard functions, the appropriate header file must be included. This is done at the beginning of the C source file.

For example, to use the printf() function in a program, which is used to display anything on the screen, the line #include <stdio.h> is required because the header file **stdio.h** contains the printf() function. All header files will have an extension .h

2. C Header file/ C Library files ++

According to Programiz. (n.d.), Some of the Library Functions in Different Header Files include: -

C Header Files	Description
<stdio.h>	Standard Input/Output functions
<ctype.h>	Character type functions
<locale.h>	Localization functions
<math.h>	Mathematics functions
<setjmp.h>	Jump functions
<signal.h>	Signal handling functions
<assert.h>	Program assertion functions

C Header Files	Description
<stdarg.h>	Variable arguments handling functions
<stdlib.h>	Standard Utility functions
<string.h>	String handling functions
<time.h>	Date time functions

3 main() function

main() function is a function that must be there in every C program. Everything inside this function in a C program will be executed.

In the above example, int written before the main() function is the **return type** of main() function. we will discuss about it in detail later.

4 C main() function body

One of the important parts of C program is the main() body which is demarcated by two Carly brazes { }.

The curly braces { } just after the **main()** function encloses the **body** of **main()** function. As seen in the program below.

```
#include<stdio.h>

int main(){

}
```

The body of main function

4 C Program body +

Every code that must be executed should be written/called inside the program body. We will talk about this when dealing with custom functions

```
int main(){
msg();        //msg() function call in the main function body


}
int msg(){
printf("Welcome to programing methodology in c");
return 0;
}
```

Program body

msg() function body

5 Comments

Comments are plain simple text in a C program that are not compiled by the compiler. We write comments for better understanding of the program. It make the code more readable.

To add a single line comment, start it by adding two forward slashses // followed by the comment statement.

To add multiline comment, enclode it between /* .... */, just like in the program above.

5 Comments +

1. / /code to be commented

```
1.  /*
2.   code
3.   to be commented
4.*/
```

```
1.  int main(){
2.  /*printing information
3.    Multi-Line Comment*/
4.
5.    printf("Hello C");
6.    return 0;
7.    }
```

6 Return statement - return 0;

A return statement is just meant to define the end of any C program. The major use of this section is to give a report how program ran.

However sometimes the value returned from one function can be used as an input by another function.

The programmer can tell the program to return which value as long as the expected value matches the return type that function.

Determine a right return value

Lets see how return values are determined

```c
#include <stdio.h>
int main() {//Note that int is is being used by main


printf(" Hello, KUMU \n");

return 0; /* Note zero is expected*/
}
```

C language Syntax Rules

C Language Basic Syntax Rules

C language syntax rules states how to form statements in a C language program - How should the line of code start, how it should end, where to use double quotes, where to use curly brackets etc.

The rule specify how the character sequence will be grouped together, to form **tokens**. *A smallest individual unit in C program is known as **C Token**.*

Tokens are either keywords, identifiers, constants, variables or any symbol which has some meaning in C language. A C program can also be called as a collection of various tokens.

In the following program

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

if we take any one statement:

```c
printf("Hello,KUMU");
```

Then the tokens in this statement

are⟶ printf, (, "Hello,KUMU",) and ;.

So C tokens are basically the building blocks of a C program.

Semicolon/Terminator ;

Semicolon ; is used to mark the end of a statement and beginning of another statement.

Absence of semicolon at the end of any statement, will mislead the compiler to think that this statement is not yet finished and it will add the next consecutive statement after it, which may lead to compilation(syntax) error.

Other basic syntax rule for C Language

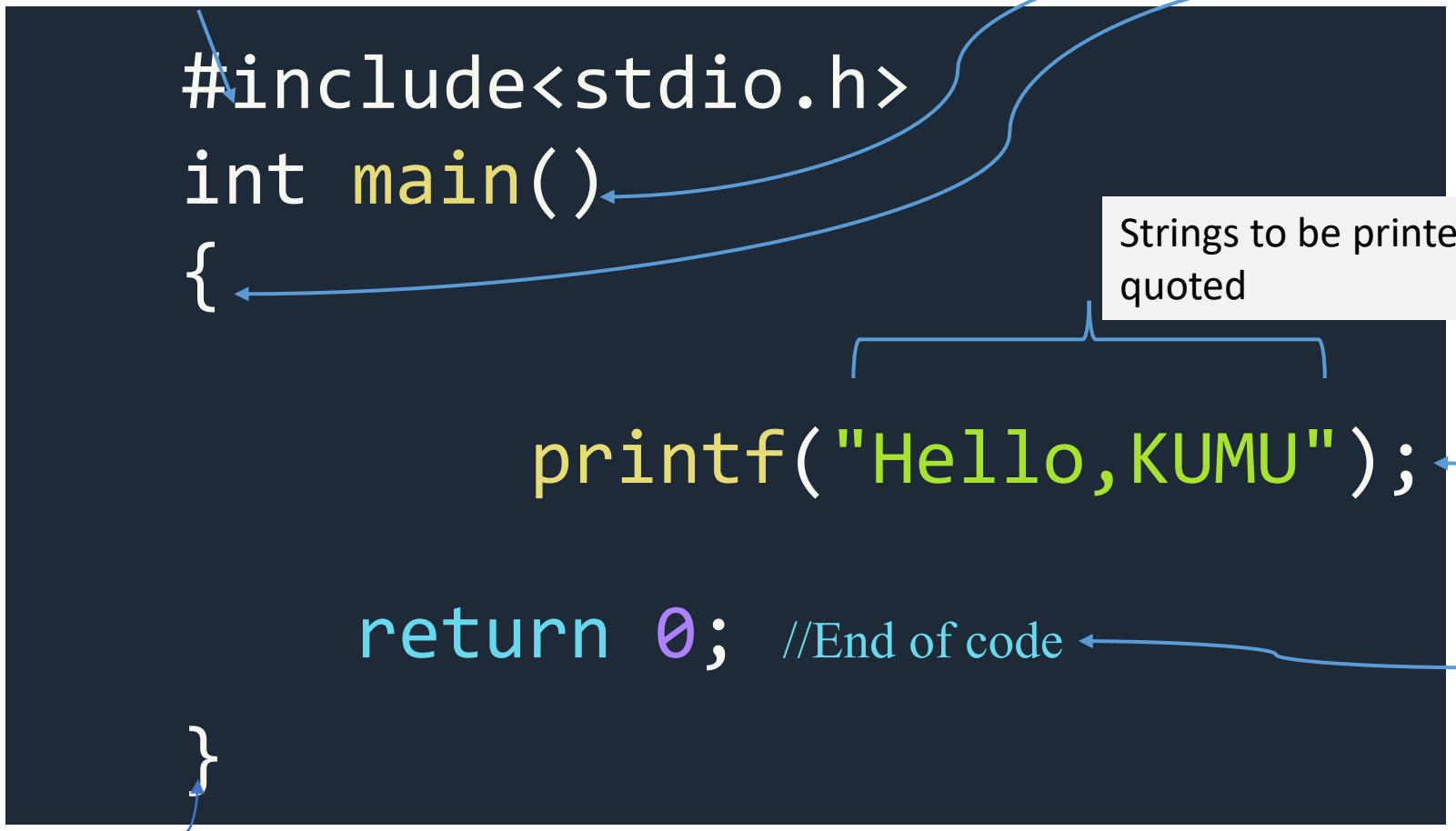
1. C is a case sensitive language so all C instructions must be written in lower case letter.
2. All C statement must end with a semicolon/Terminator.
3. Whitespace is used in C to describe blanks and tabs.
4. Whitespace is required between keywords and identifiers. We will learn about keywords and identifiers in the next Lecture.

Terminator omitted

```
Lin1: #include<stdio.h>
Lin2: int main() {
Lin3: printf("Hello,KUMU")
Lin4:
Lin5: return 0;
Lin6 :}
```

In the above program, we have omitted the semicolon from the printf("...") statement, hence the compiler will think that starting from printf uptill the semicolon after return 0 statement, is a single statement and this will lead to compilation error

Note! when writing a program



Summary

In summary therefore, we studied about: -

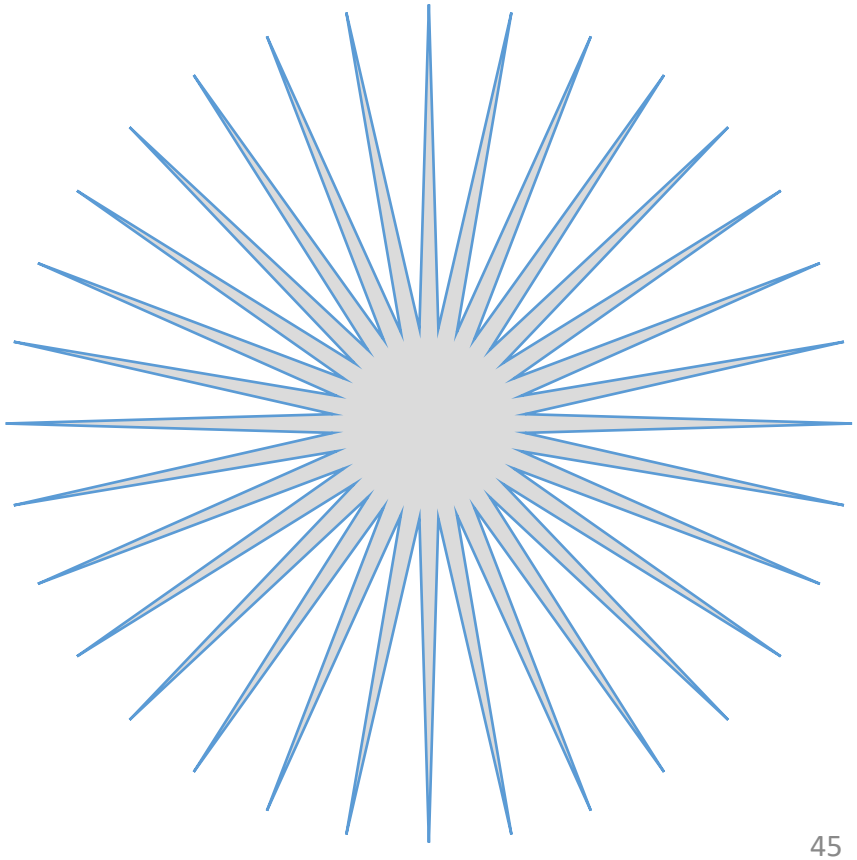
1. Writing a program in C
2. C Program Structure

   Layout of the program, Parts of C language
3. C Language Syntax Rules

Review Questions

1. Explain the difference between Building and running the program.
2. Mention any three parts of C Language.
3. Define a comment and give two examples.
4. Mention any three Library/Header files you know.
5. Write a c program that can display your name on the screen.

Thank you for you attention



References

C standard library functions. Programiz. (n.d.). Retrieved October 2, 2021, from https://www.programiz.com/c-programming/library-function.