SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

COURSE NAME : 23CST101– PROBLEM SOLVING & C PROGRAMMING

I YEAR/ I SEMESTER

UNIT-II C PROGRAMMING BASICS

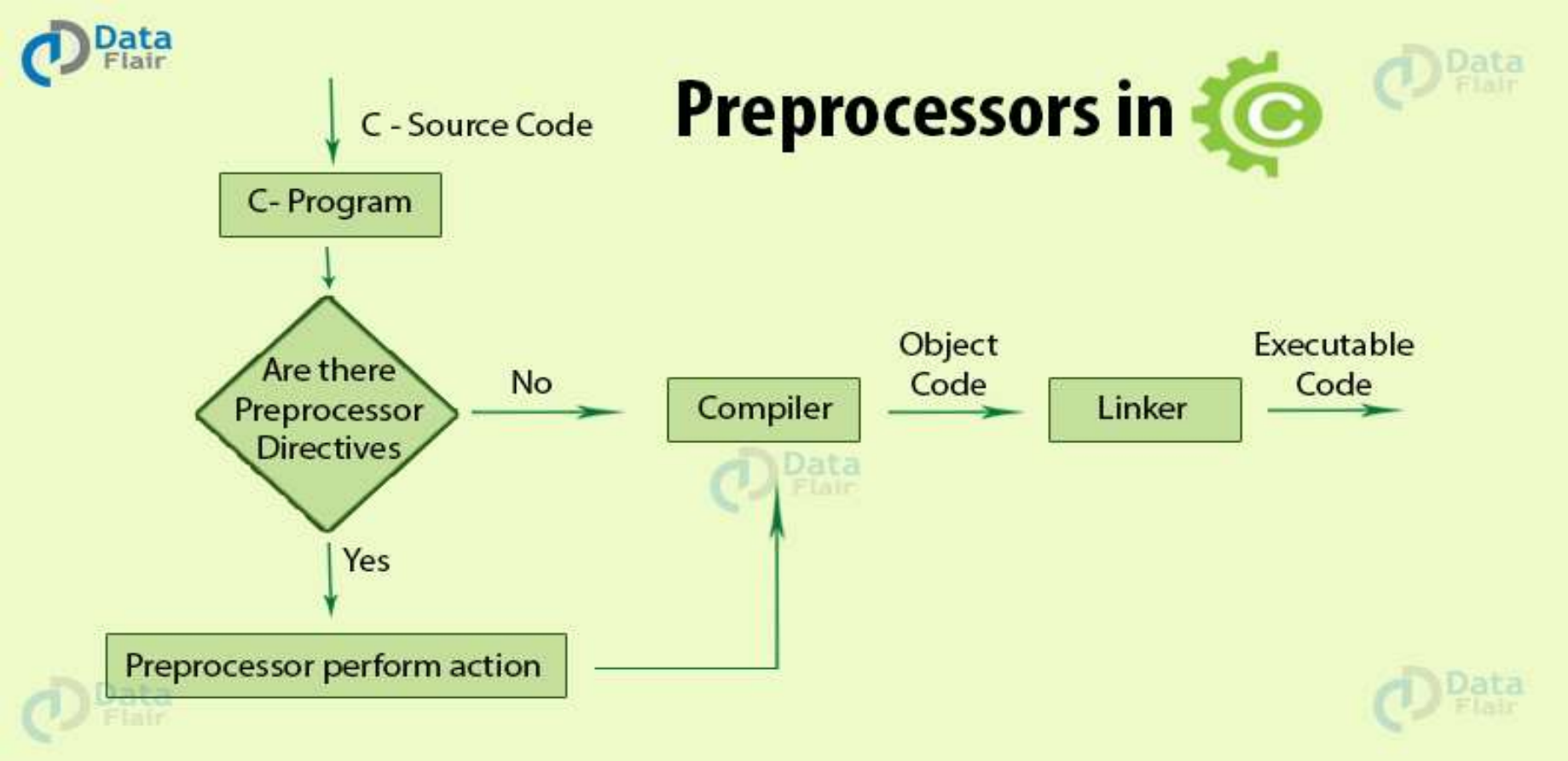
Topic: Preprocssor

Dr. B.Vinodhini
Associate Professor
Department of Computer Science and Engineering

Pre-processors in C

- Preprocessing in C is not a part of the compilation process but done just before compilation.
- The term “preprocessor” is self-explanatory.
- The word “pre” means “before” and the word “processor” refers to “making something”.
- Before the source code is compiled, it gets automatically processed due to the presence of preprocessor directives.
- In programming terminology, a preprocessor is nothing but a System Software.
- Translating the source code written in a high-level language to object code written in the machine-level language

Pre-processors in C



Pre-processors in C

Example of Preprocessors in C

- Let us acknowledge the existence of preprocessors by considering a simple example.
- Suppose you forgot to use the **hash/pound (#)** symbol while working with the header file **#include<stdio.h>**, will your program work?
- Lets check. Example
- In the C programming language, you would find an error if you skip the **# sign** while defining a header file.

Pre-processors in C

What are Preprocessor Directives in C?

- Preprocessor Directives are nothing but the commands we use in the preprocessor.
- In the C language, all of the preprocessor directives begin with a hash/pound (#) symbol.
- In C we can use preprocessor directives anywhere in our program.
- But, it is preferable to use them at the beginning of the program.
- This enhances the readability of the code.

Types of Pre-processors in C

There are basically 4 types of preprocessors in C.

1. Macros

- A macro is a segment of code that has the ability to provide the inclusion of header files and specifies how to map a replacement output sequence in accordance to a well-defined series of steps a particular input sequence.
- For example,

  #define MAX 100
- Here, the string MAX has the assigned constant value of 100.
- Also, MAX is called macro template and 100 is repressed to as macro expansion.
- #define macro_template macro_expression does not terminate with a semicolon.

Types of Pre-processors in C

2. File Inclusion

- File inclusion is responsible for instructing the compiler to include a specific file in the source code program.
- Depending on the type of file included, file inclusion is categorized into two types, namely:
  - **Standard header files –** These files refer to the pre-existing files, which convey a specific meaning to the compiler before the actual compilation has taken place.
  - **User-defined files –** The C language gives the programmer the provision to define their own header files in order to divide a complex code into small fragments.

Types of Pre-processors in C

3. Conditional Compilation

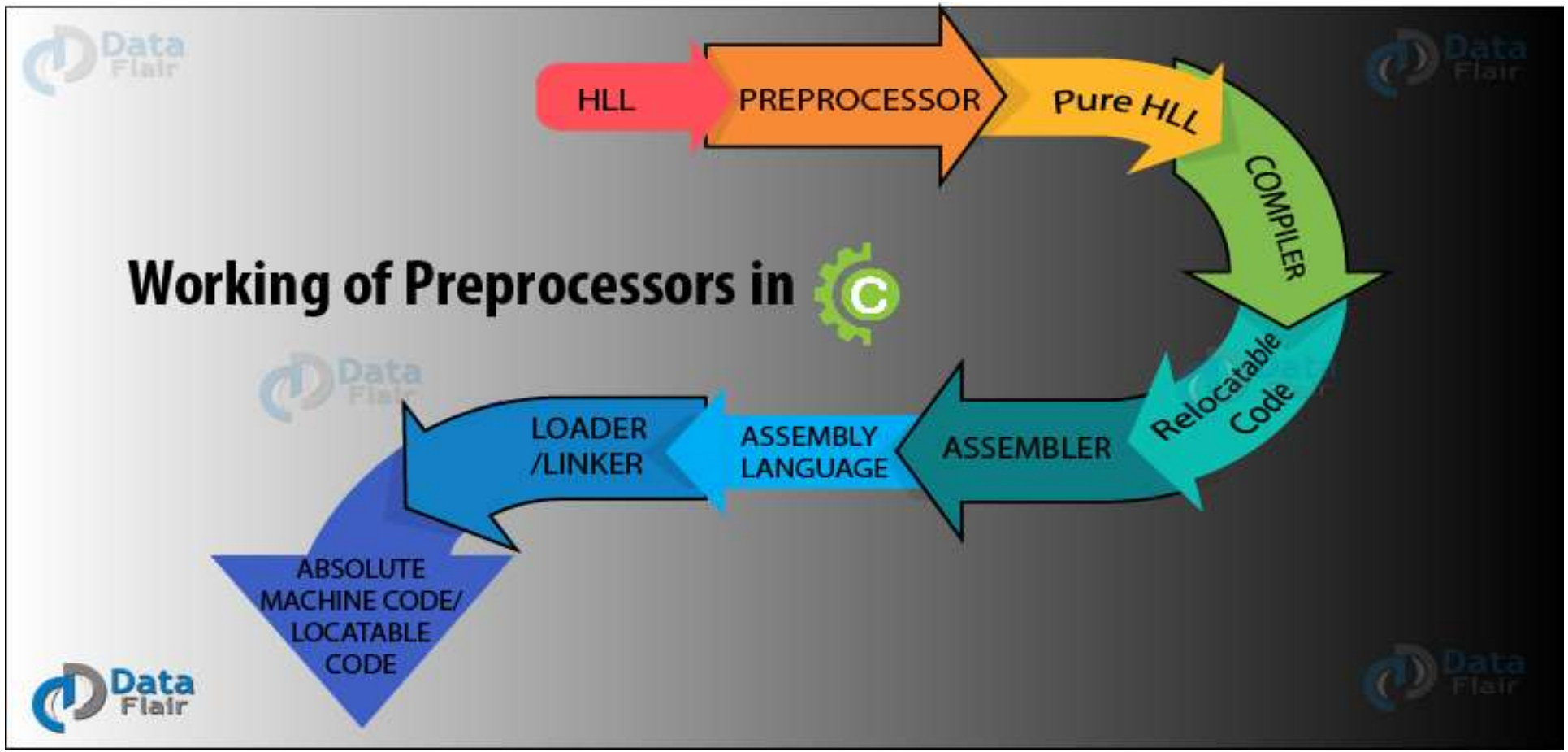
- Just like we use if-else statements for the flow of control over specific segments of code, in the same way, we use the concept of conditional compilation.
- Conditional compilation is a type of preprocessor that gives the programmer the power to control the compilation of particular segments of codes.
- It is done with the help of the 2 popular preprocessing commands, namely:
  - ifdef
  - endif

Types of Pre-processors in C

4. Other Directives

- There are 2 more preprocessor directives in C apart from the ones already discussed.
- They are, namely:
  - **#undef:** As eccentric as it sounds, we use #undef directive to undefine the pre-existing, standard header or a user-defined header file.
  - **#pragma:** We use this type of preprocessor directive to enable or disable certain features. It is important to note that #pragma varies from compiler to compiler.

How Preprocessor Works in C?



Pre-processors in C

Here is a table that summarizes the utility of preprocessors in C:

Preprocessor	Elucidation
#include	Used to insert a specific header from a file.
#define	Used as a replacement of a preprocessor macro.
#ifdef	Used when dealing with conditions. If the macro is defined, it returns true.
#endif	Used to close the preprocessor directive in accordance with a given condition.
#undef	Used to undefine a standard or user-defined header.
#pragma	Used to enable and disable certain features.
#ifndef	If the macro is not defined, it returns true.
#if	Used to check if the condition is true in compile time.
#else	Used to check the next condition if #if proves to be false in compile time.
#elif	Used as a combination of #else and #if.
#error	Used to print error on stderr.

How to Run C Program in Ubuntu?

Running C Program in Ubuntu


Thank You!