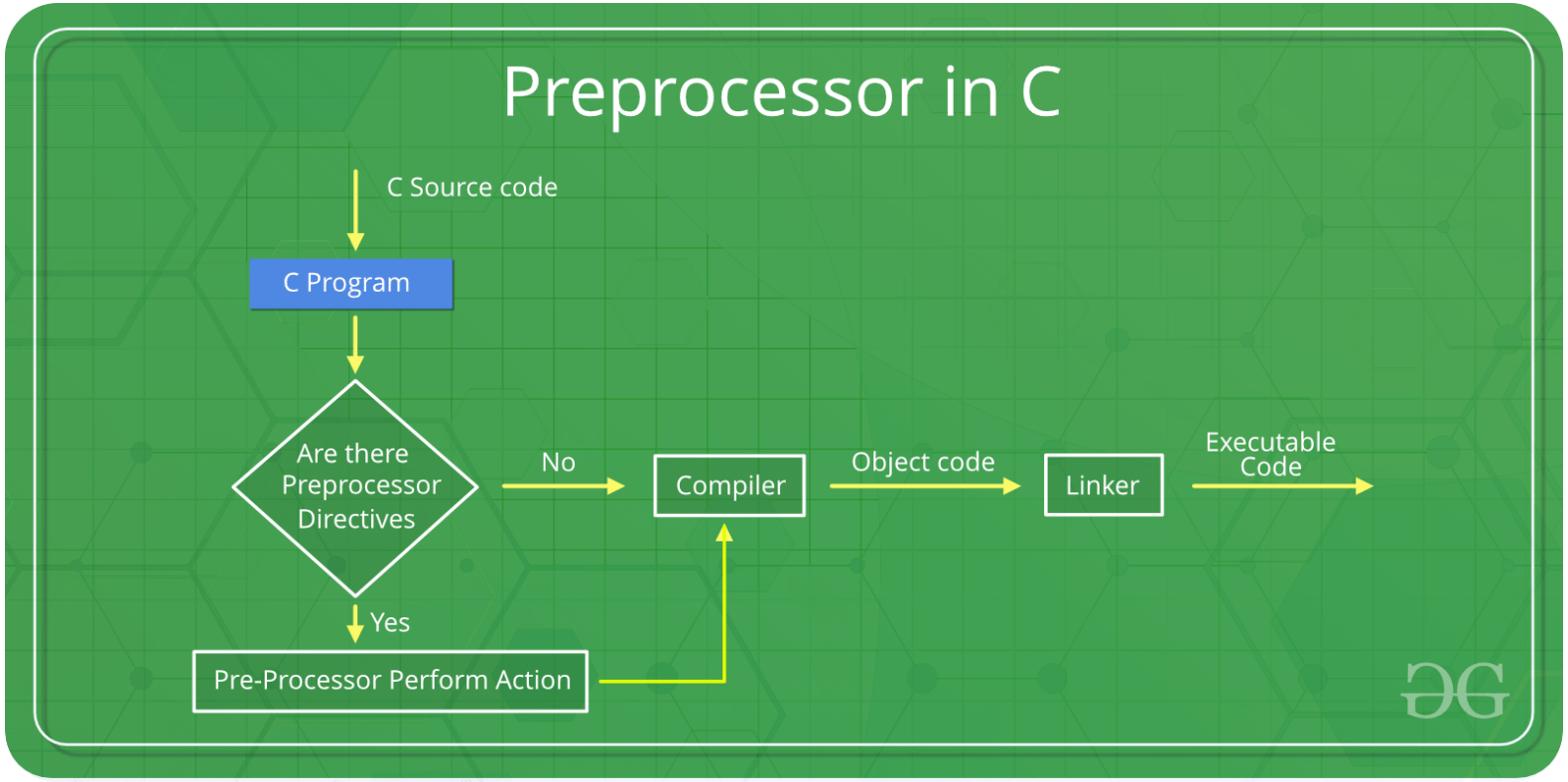
Session 15

Mostafa Akram

C Preprocessors

- Preprocessors are programs that process the source code before compilation. Several steps are involved between writing a program and executing a program in C. Let us have a look at these steps before we actually start learning about Preprocessors.



Preprocessor Directives in C

- Preprocessor programs provide preprocessor directives that tell the compiler to preprocess the source code before compiling. All of these preprocessor directives begin with a ‘**#**’ **(hash) symbol**. The ‘**#**’ symbol indicates that whatever statement starts with a ‘**#**’ will go to the preprocessor program to get executed. We can place these preprocessor directives anywhere in our program.

List of preprocessor directives in C

Preprocessor Directives	Description
#define	Used to define a macro
#undef	Used to undefine a macro
#include	Used to include a file in the source code program
#ifdef	Used to include a section of code if a certain macro is defined by #define
#ifndef	Used to include a section of code if a certain macro is not defined by #define
#if	Check for the specified condition
#else	Alternate code that executes when #if fails
#elif	Combines else and if for another condition check
#endif	Used to mark the end of #if, #ifdef, and #ifndef

Types of C Preprocessors

- **There are 4 Main Types of Preprocessor Directives:**

1.Macros

2.File Inclusion

3.Conditional Compilation

4.Other directives

1. Macros

- In C, Macros are pieces of code in a program that is given some name. Whenever this name is encountered by the compiler, the compiler replaces the name with the actual piece of code. The '**#define**' directive is used to define a macro.
- ***Note*** *There is no semi-colon (;) at the end of the macro definition. Macro definitions do not need a semi-colon to end.*

```
#define token value
```

Macros With Arguments (macro like function)

We can also pass arguments to macros. Macros defined with arguments work similarly to functions.

```
#define foo(a, b) a + b
#define func(r) r * r
```

2. File Inclusion

- This type of preprocessor directive tells the compiler to include a file in the source code program. The **#include preprocessor directive** is used to include the header files in the C program.
- **There are two types of files that can be included by the user in the program:**

1- Standard Header Files

2- User-defined Header Files

3. Conditional Compilation

1.#if Directive

2.#ifdef Directive

3.#ifndef Directive

4.#else Directive

5.#elif Directive

6.#endif Directive

- **#endif** directive is used to close off the #if, #ifdef, and #ifndef opening directives which means the preprocessing of these directives is completed.

4. Other Directives

- Apart from the above directives, there are two more directives that are not commonly used. These are:

1.#undef Directive

2.#pragma Directive

1. #undef Directive

- The #undef directive is used to undefine an existing macro. This directive works as:

```
#undef LIMIT
```

2. #pragma Directive

- This directive is a special purpose directive and is used to turn on or off some features. These types of directives are compiler-specific, i.e., they vary from compiler to compiler.

```
#pragma directive
```

Task

- Install Microchip studio
- Install Proteus
- File guard

Links