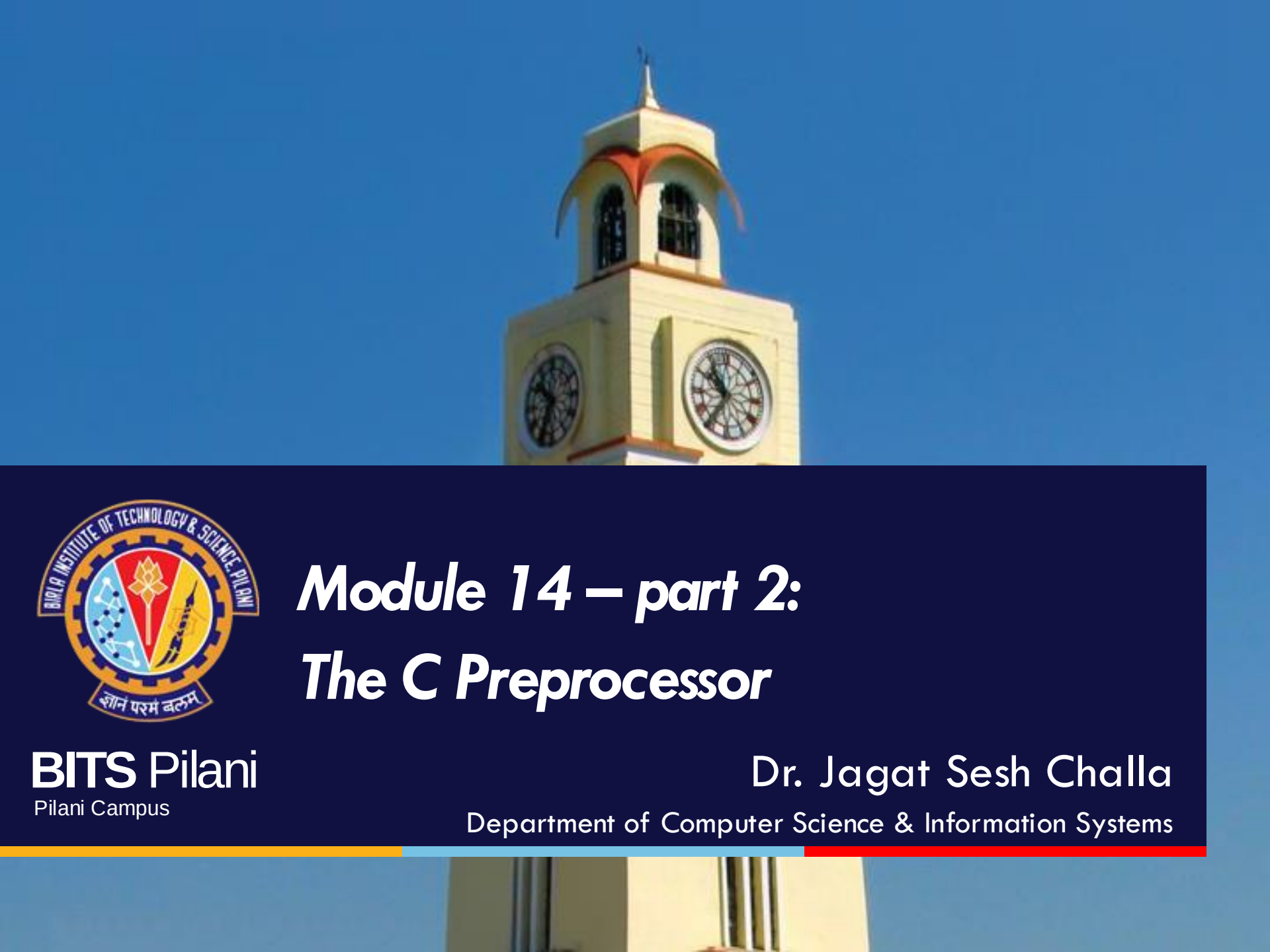


Module 14 – part 2: The C Preprocessor

BITS Pilani
Pilani Campus

Dr. Jagat Sesh Challa
Department of Computer Science & Information Systems

Module Overview

innovate achieve lead

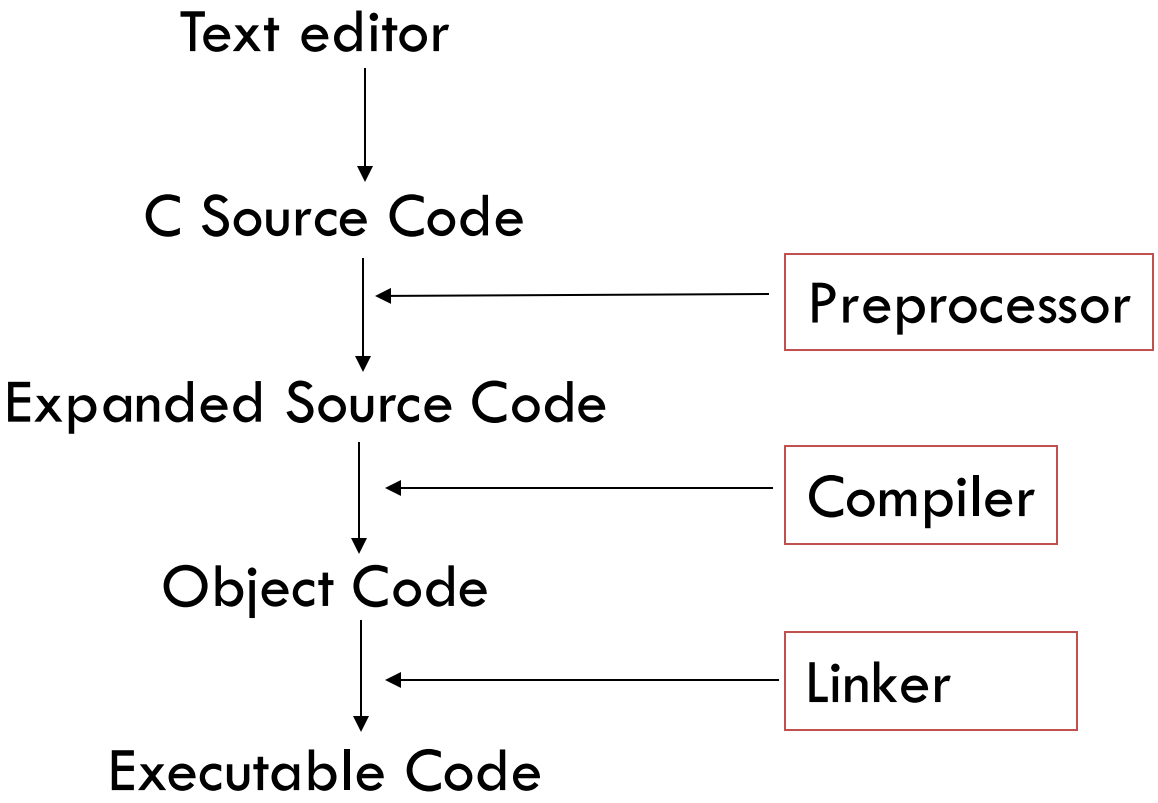
- C Preprocessor
  - Preprocessor Directives
    - Macro expansion
    - File inclusion
    - Conditional Compilation

The C Preprocessor

The C Preprocessor

- The **C Preprocessor** is a "Program that processes source program before it is passed to the compiler"
- Preprocessor commands are called as **Directives**
- Each directive begins with a **# symbol**

C Program execution



innovate achieve lead

BITS Pilani

Pilani Campus

Macro Expansion Directives

Macro Expansion Directives: Example 1 (Defining constants)


```
#define MAX 100
#define SIZE 10
void main()
{
  int k, a[SIZE];
  for(k=0; k<MAX; k+=10)
  printf("%d",k);
}
```

During preprocessing each occurrence of MAX is replaces by 100 and each occurrence of SIZE is replaced by 10.

And then the program is compiled!

Example 2 (Define Operators)

innovate achieve lead

```
#define AND &&
#define OR ||
#define EQUALS ==
#define MOD %
#define NOTEQUAL !=

void Leap_year(unsigned int yr)
{
  if((yr MOD 4 EQUALS 0) AND (yr MOD 100 NOTEQUAL 0) OR (yr
  MOD 400 EQUALS 0))
      printf("Given Year is Leap Year\n");
  else
      printf("Given Year is Not a Leap Year\n");
}
```

Example 3 (Define Conditions)

innovate achieve lead

```
#define AND &&
#define OR ||
#define RANGE ((ch>='a' AND ch<='z')OR(ch>='A' AND
  ch<='Z'))
void Alpha_Check(char ch)
{
  if(RANGE)
     printf("Entered character is an Alphabet\n");
  else
     printf("Entered character not an Alphabet\n");
}
```

Macros with Arguments

innovate achieve lead

```
#define CUBE(x) (x*x*x)
```

If statements appears in the program like

```
vol = CUBE(side);
```

It will expand to:

```
vol = (side*side*side);
```

If statements appears in the program like

```
vol = CUBE(x+y);
```

It will expand to:

```
vol = (x+y*x+y*x+y);
```

If statements appears in the program like

```
printf("Volume = CUBE(a)"); Expansion???
```

Nesting of Macros

innovate achieve lead

One Macro definition can be used in another Macro

Example:

Consider the Macro Definitions

```
#define SQUARE(p)  ((p)*(p))
#define CUBE(p)    (SQUARE(p))*(p))
#define EIGHTH(p)  (CUBE(p)*CUBE(p)*SQUARE(p))
```

How it will expand???

Nesting of Macros (Contd.)

Macros calls can be nested as function calls

Consider a Macro definition:

```
#define MIN(P,Q)   (((P<Q))?(P):(Q))

int MinThree(int a, int b, int c)
{
   int min;
   min = MIN(a, MIN(b,c)); /* Macro Call */
   return (min);
}
```

Exercise:

Write a macro call for getting min of five values.

Advantages of Macros

- One place replacement is required for any changes
- Easy to handle
- Less error prone
- Easy to debug

Macro vs Function()

innovate achieve lead

- Functions make the program compact and smaller but they slow down the program
  - ***Each call results in a push on the activation stack.***
  - ***The corresponding return is a pop from the stack.***

- Macros make the program run faster but increases the program size
  - ***Macros reduce the function call overhead by performing code replacement at compile time.***
  - ***Code replacement happens once (per compilation), whereas function calls can be repetitive causing performance bottleneck.***

Important Questions:

When is it best to use macro?

And when is it best to use function?

Can we replace all Macros with functions?

- Macros do not have types (for parameters)
- What will happen if we declare

```
#define fact(n) (n==0) ? 1 : n*fact(n-1)
```

- Try this out! Recursive substitutions are not possible
- **Why not?**
  - *Because the compiler does not "execute" code.*
  - *Only code replacement is done.*
  - *Code replacement is a one-time job.*
  - *Code replacement is not recursive.*

innovate achieve lead

BITS Pilani

Pilani Campus

File Inclusion Directives

File Inclusion Directives

innovate achieve lead

How to include a file in your program?

#include “file_name” **OR**
#include<file_name>

What is the difference between above?

1. *If a large program is divided into several files, each containing a set of related functions. These files should be included at the beginning of the main program file.*
2. *Inclusion of header files for using some predefined functions in C library.*

When a header file is included, the preprocessor includes the file in the source code program (.c file) before its compilation.

innovate achieve lead

BITS Pilani

Pilani Campus

Compiler Control Directives

Compiler Control Directives: #ifdef

We can skip over the compilation process of the part of the source program by inserting the preprocessing commands:

#ifdef and #endif

Syntax:

```
#ifdef macroname
        statement 1;
        statement 2;
#endif
```

If macro has been defined, the block of code will be processed as usual; otherwise not.

Compiler Control Directives: #if


```
#define CHECK 1
void  main()
{ #if CHECK==1
        statement1;
        statement2;
  #elif CHECK==0
        statement3;
        statement4;
  #else
        statement5;
        statement6;
  #endif
```

CHECK is Macro Name

What we need?

1. If we don't want to compile some part of the program so we can skip those statements.

2. To make a program portable so it can work on two or more than two types of different architectures.

“ifndef” guards for header files


```
#ifndef HEADER1
#define HEADER1

#include <stdio.h>
...
...

#endif
```

“if not defined”

then define

While linking multiple .o files in a C project, if compiler sees same inclusion in multiple file that are being linked, it throws an error. This guard helps in including the header file only once.

innovate achieve lead

BITS Pilani

Pilani Campus

Thank you

Q & A