Problem Solving With C - UE24CS151B

Preprocessor Directives

Prof. Sindhu R Pai
PSWC Theory Anchor, Feb-May, 2025
Department of Computer Science and Engineering

PROBLEM SOLVING WITH C

Preprocessor Directives

- Introduction
- Types of Pre-processor Directives
- Macros
- Points to know about Macros
- Predefined Macros
- Macro vs Enum
- File Inclusion Directives
- Conditional compilation Directives
- Other Directives

Preprocessor Directives

Introduction

- **The lines of code** in a C program which are **executed by the 'C' pre-processor**
- A **text substitution tool** and instructs the compiler to do required pre-processing before the actual compilation
- All pre-processor commands begin with a hash symbol (#)
- It must be the first nonblank character and for readability, pre-processor directive should begin in the first column conventionally

PES UNIVERSITY

Types of Preprocessor Directives

- Types include:
  - Macros
  - File Inclusion
  - Conditional Compilation
  - Other directives
- Possible to see the effect of the pre-processor on source files directly by using the **-E option of gcc**

PES UNIVERSITY

Macros

- A piece of code in a program which has been given a name
- During preprocessing, it substitutes the name with the piece of code
- **#define** directive is used to define a macro.
- Example: #define PI 3.14
- Coding Examples

Points to know about Macros

- Macro **does not** judge anything
- **No memory Allocation** for Macros
- Can define string using macros
- Can define macro with expression
- Can define macro with parameter
- Macro can be used in another macro
- Constants defined using #define cannot be changed using the assignment operator
- Redefining the macro with #define is allowed. But not advisable

Predefined Macros

S.no	Macro & Description
1	**__DATE__** The current date as a character literal in "MMM DD YYYY" format.
2	**__TIME__** The current time as a character literal in "HH:MM:SS" format.
3	**__FILE__** This contains the current filename as a string literal.
4	**__LINE__** This contains the current line number as a decimal constant.
5	**__STDC__** Defined as 1 when the compiler complies with the ANSI standard.

Macro vs Enum

- Macro doesn’t have a type and enum constants have a type int.
- Macro is substituted at pre-processing stage and enum constants are not.
- Macro can be redefined using #define but enum constants cannot be redefined. However assignment operator on a macro results in error

File Inclusion Directives

- Instructs the pre-processor to include a file in the program using **#include** directive
- Two types of files
  - **Header File or Standard files: Included between < and >**
    - Contains the definition of pre-defined functions like printf(), scanf() etc.
    - To work with these functions, header files must be included
  - **User defined files: Included using " and "**
    - When a program becomes very large, it is good practice to divide it into smaller files and include whenever needed.
    - Adding user defined header files in the program

Conditional Compilation Directives

- A few blocks of code will be compiled in a particular program based on the result of some condition
- Conditions can be mentioned using - **#ifdef, #ifndef, #if, #else, #elif, #else, #endif**
- Coding Examples

PES UNIVERSITY

Other Directives

- **#undef Directive:** Used to undefine an existing macro
  - Example: #undef LIMIT
  - After this statement every "#ifdef LIMIT" statement will evaluate to false
- **#pragma startup and #pragma exit:** Helps to specify the functions that are needed to run before program startup and just before program exit
- **#pragma warn:** This directive is used to hide the warning messages which are displayed during compilation using **–rvl, -par and -rch**

THANK YOU

Department of Computer Science and Engineering

Dr. Shylaja S S, Director, CCBD & CDSAML, PESU
Prof. Sindhu R Pai - sindhurpai@pes.edu