C Programming

Lesson 8: Preprocessor

Lesson Objectives

- **In this lesson, you will learn Preprocessor topics related to C Programming.**

Lesson Coverage

- **In this lesson, you will cover:**
  - Introduction to Preprocessor
  - Macro substitution

Introduction to Preprocessor

- **Preprocessor is a program that processes the source text of a C program before it is passed to the compiler**
- **It can be an independent program or its functionality may be embedded in the compiler**
- **It offers a collection of special statements, called "preprocessor directives"**

Introduction to Preprocessor

- **Preprocessor has four major functions:**
  - Macro replacement
  - Conditional compilation
  - File inclusion
  - Error generation

Preprocessor Directives

- **Preprocessor Directives begin with a # symbol**

- **Some of the preprocessor directives are as follows:**
  - #define directive
  - #include directive
  - #undef directive
  - #error directive

Conditional compilation directives

- #if
- #ifdef
- #ifelif
- #ifelse
- #ifndef

Concept of Macro Substitution

- **During macro substitution, the preprocessor replaces every occurrence of a simple macro in the program text by a copy of the body of the macro**
- **The body of the macro may itself contain other macros**
- **It is achieved using the #define directive**
- **The general syntax is as shown below:**

```
#define macro-name  sequence-of-tokens
```

Concept of Macro Substitution

- **The macro-name is associated with the sequence-of-tokens that appear from the first blank after the macro-name to the end of the file**
  - For example:

```
/* Associates macro name MIN with value -5 */
#define MIN -5
```

Sample Code

- **The program given below shows #define directive used to define operators**

```
#include <stdio.h>
#define AND &&
#define OR ||
void main(void)
{long int salary=;
    char gender='F';
      if((salary>200000) AND (gender=='M' OR gender=='F'))
         printf("You have to pay Income Tax.....");
      else
        printf("No Tax.....");  }
```

Concept of Parameterized Macros

- **Parameterized Macros are macros with arguments**
  - For example:

```
#include <stdio.h>
#define AREA( r )  (3.14*r*r)
void main(void)
{
      float radius;
      printf("Enter the radius \t");
      scanf("%f",&radius);
      printf("\nArea of the circle is %f", AREA(radius));
}
```

Concept of Nested Macros

- **In Nested Macros, one macro can be used in the definition of another macro, as shown below:**

```
/* This program shows use of nesting of macros */
#include <stdio.h>
#define  SQUARE(num) (num*num)
#define CUBE(num)         (SQUARE(num) * num)
void main(void)
{
        int no;
        printf("Enter the number  ");
        scanf("%d",&no);
        printf("\nSquare of a number is %d",SQUARE(no));
        printf("\nCube of a number is %d",CUBE(no));
}
```

Concept of Un-defining a Macro

- **Un-defining a Macro is useful to restrict the definition only to a particular part of the program**
  - Syntax:

```
#undef identifier
```

  - For example:

```
#undef SQUARE
```

Concept of File Inclusion

- This preprocessor directive causes one file to be included in another
- It is useful to include the library file that contains common library functions or necessary macros in another file

Concept of File Inclusion

- **Syntax:**

```
#include <filename>
```

  - It searches the specified file in default directory and includes, if it exists.

  or

```
#include "filename"
```

  - It searches the specified file in default directory and also in current directory and includes, if it exists

File Inclusion - Example

- **For example:**
  - Suppose we have the following three files:
    - function.c contains some functions
    - proto.h contains prototypes of functions
    - test.c contains test functions
  - Then we can make use of a definition or function contained in any of these files by including them in the program as shown on the next slide

File Inclusion - Example

- **For example (contd.):**

```
#include <stdio.h>
#include “function.c”
#include “proto.h”
#include “test.c”
#define MAX  50
void main(void)
{
  /* Here the code in the above three files */
  /* is added to the main code */
 /* and the file is compiled */
}
```

Concept of Conditional Compilation

- **Conditional Compilation allows selective inclusion of lines of source text on the basis of a computed condition**

- **Conditional Compilation is performed using the preprocessor directives shown below:**

  - #ifdef
  - #ifndef
  - #elif
  - #else
  - #endif

Concept of Conditional Compilation

- Logical directive tests whether an identifier exists as a result of having been created in a previous #define directive.

  C Conditional Compilation takes the following form:

  #if defined identifier

  ...

  #endif

- If identifier is defined, statements between #if and #endif are included in the program code. If the identifier isn't defined, the statements between the #if and the #endif will be skipped in the absence of an identifier.

- with the help of **#elif and #else** these can be handled well. In fact, this tends to be used more frequently than the form you've just seen.

- #ifndef is used to check whether a particular symbol is defined.

Concept of Error Generation

- **Error Generation is useful to display the user-defined error message on occurrence of an error**
- **The format of the directive is:**
  - #error token sequence
- **It causes the implementation to produce a diagnostic message containing the token sequence**

Concept of Error Generation

- **For example:**

```
#ifndef PI
        #error  "PI NOT DEFINED"
#endif
```

- **If PI is not defined, pre-processor will print the error message "PI NOT DEFINED" and the compilation will stop**

Library

- **A library is a package of code that is meant to be reused by many programs**
- **There are two types of libraries: static libraries and dynamic libraries**
  - There are two types of libraries: static libraries and dynamic libraries

Static Library

- **A static library (also known as an archive) consists of routines that are compiled and linked directly into your program**
- **When you compile a program that uses a static library, all the functionality of the static library becomes part of your executable**
- **On Windows, static libraries typically have a .lib extension. On Linux, static libraries typically have an .a (archive) extension**

Dynamic Library

- **A dynamic library (also called a shared library) consists of routines that are loaded into your application at run time**
- **When you compile a program that uses a dynamic library, the library does not become part of your executable — it remains as a separate unit**
- **On Windows, dynamic libraries typically have a .dll (dynamic link library) extension, whereas on Linux, dynamic libraries typically have a .so (shared object) extension**

Lab

- **Lab 8**

Summary

- **The preprocessor directive causes one file to be included in another.**
- **C provides a facility called type definition, which allows users to define new data types. They are equivalent to existing data types.**

Review Questions

- **True or False?**

1. A macro must always be defined in capital letters.

2. Macro calls and function calls work exactly similar.

3. Every C program will contain AT LEAST ONE preprocessor directive.

Review Question: Match the Following

1. Macro Replacement	1. #include
2. Conditional Compilation	2. #error
3. File Inclusion	3. #undef
4. Error Generation	4. #define