CC-112 Programming Fundamentals

The C Preprocessor

Nazar Khan
Department of Computer Science
University of the Punjab

The C Preprocessor

- The preprocessor executes before a program is compiled.
- All preprocessor directives begin with #.
- Only whitespace characters and comments may appear before a preprocessor directive on a line.

include

- The *#include preprocessor directive* includes a copy of the specified file.
- If the filename is enclosed in quotes (“. . .”), the preprocessor begins searching in the same directory as the file being compiled.
- If the filename is enclosed in angle brackets (< and >), as is the case for C standard library headers, the search is performed in an implementation-defined manner.

Symbolic constant using `# define`

- The *#define preprocessor directive* is used to create *symbolic constants* and *macros*.
- A *symbolic constant* is a name for a constant.

Macro using # define

- A *macro* is an operation defined in a #define preprocessor directive.

```
1   #define PI 3.14159
2   #define CIRCLE_AREA(x) ((PI) * (x) * (x))
3   void main()
4   {
5     area = CIRCLE_AREA(4);
6   }
```

  What does the preprocessor do?

  1. First, `PI` gets replaced by `3.14159` and `x` by `4` in the replacement text.
  2. Then this expanded replacement text is substituted in line 5 to get `area = ((3.14159) * (4) * (4));`

- Macros may be defined with or without arguments.
- Symbolic constants and macros can be undefined by using the *#undef preprocessor directive*.
- Scope of symbolic constant or macro is from its definition until it's undefined with `#undef` or until the end of the file.

Avoid `# define`

- It is better to declare constants using the `const` keyword instead of using `#define`.
- It is better to use functions instead of macros.

Debugging using conditional compilation

- *Conditional compilation* enables you to control the execution of preprocessor directives and the compilation of program code.
- The conditional preprocessor directives evaluate constant integer expressions. Cast expressions, sizeof expressions and enumeration constants cannot be evaluated in preprocessor directives.
- Every #if construct ends with #endif.
- Directives #ifdef and #ifndef are provided as shorthand for #if defined(name) and #if !defined(name).
- Multiple-part conditional preprocessor constructs may be tested with directives #elif and #else.

Debugging using `#error` and `#pragma` directives

- The *#error directive* prints an implementation-dependent message that includes the tokens specified in the directive.
- The *#pragma directive* causes an implementation-defined action. If the pragma is not recognized by the implementation, the pragma is ignored.

Debugging via `assert`

- Macro `assert` is defined in the header file `<assert.h>` header.
- It tests the value of an expression.

```
assert(x<=10);
```

- If the value is 0 (false), it prints an error message and calls function `abort` (defined in `stdlib.h`) to terminate program execution.
- Useful debugging tool for testing whether a variable has a correct value.
- When debugging is no longer needed, use

```
#define NDEBUG
```

  to ignore all `assert` commands instead of manually removing or commenting them out.