First Year Engineer in computer science

CHAPTER 6:

Macros

2023-2024

Sid Ahmed BERRABAH

Macro

A macro is a definition of an identifier (name) that can be substituted for its value at compile time.

Macros are often used to define:

- Constants
- Simple functions with parameters.

Syntax:

#define Macro_Name replacement_text

The **Macro_Name** is the name of the macro, and the **replacement_text** is the value that will be substituted for the macro at compile time.

Constant definition

The **replacement_text** in the constant definition can be any value or segment of code that will replace the **Macro_Name** in the program before complition.

Examples:

```
#define PI 3.14
#define start {
#define end }
#define message "worning: small value"
#define forever for( ; ; )    // infinite loop
```

Constant definition

Example:

```
#define PI 3.14
int main()
{
    float radius = 2.5;
    float area = PI * radius * radius;
    print(“area of circle is %f”, area);
}
```

Output:

Area of circle is 19.625000

Function like macros

A function like macros (parameterized macros) can take arguments, just like functions, and are defined as follows:

#define **identifier**(list-of-parameters) body-of-the-macro

The list of parameters is a list of identifiers separated by commas .

Example:

```
#define MAX(a,b) a > b ? a : b
```

When the preprocessor encounters **MAX(x,y)** in the source code (where **x** and **y** are any values or variables or expressions), it will replace it with the body of the macro (**x > y ? x : y**).

Function like macros

- The distinction between a symbolic constant definition and a macro with parameters is made on the character that immediately follows the macro name:
  - if this character is an opening parenthesis, it is a macro with parameters,
  - otherwise it is a symbolic constant.

Therefore, you should never put a space between the name of the macro with parameters and the opening parenthesis.

Function like macros

Example : if we add a space between the name of the macro and the opening parenthesis:

```
#define MAX (a,b) (a > b ? a : b)
```

the preprocessor will not be able to correctly identify the macro.

The following statement in the code

```
z = MAX(x,y);
```

will be expanded by the preprocessor as:

```
z = (a,b) (a > b ? a : b)(x,y)
```

Function like macros

Example: let the following function like macro:

```
#define SQUARE(a) a * a
```

The preprocessor expands the following statements as:

```
z = SQUARE(x); →   z = x * x;      (correct)

z = SQUARE(x + y);    →   z = x + y * x + y;    (not correct)
```

Function like macros

To write correct macros with parameters you should:

- enclose each occurrence of the parameters in parentheses.
- enclose the body of the macro in parentheses.

Example: The correct form of the SQUARE macro will be:

```
#define SQUARE(a) ((a) * (a))
```

In this case `z = SQUARE(x + y);` will expanded to

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

Side effects in Function like macros

Example:

The following call for the SQUARE macro

```
z = SQUARE(x++);
```

will be expanded to

```
z = (x++) * (x++);
```

The increment operator is applied twice. (Side effect problem)

RULE: To avoid side effect problem, avoid using assignment, increment, and decrement operators

Exercises

Exercise 1 : Define the following macros:

(1) ABS(x) to compute the absolute value a number.

(2) IsEven(x) to check if a number is even or not.

(3) IsOdd(x) to check if a number is odd or not.

(4) SIGN(x) which return 1 if the number is positive and -1 if the number is negative.

(5) ROUND(x) To round a floating-point number to the nearest integer.

Exercises

Exercise 2 :

Define the macro RANGE(val,min_val,max_val) which operates in the following way (hysteresis) :

- If val is less than min_val, val takes the value of min_val.
- If val is greater than max_val, val takes the value of max_val.
- If val is between min_val and max_val, val is unchanged.