Chapter 8. Preprocessor

Byoung-Tak Zhang
TA: Hanock Kwak

Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National Univertisy

http://bi.snu.ac.kr

Preprocessor

- Preprocessor is a text substitution tool and they instruct compiler to do required pre-processing before actual compilation.
- C language uses the preprocessor to expend its power and notation.
- *preprocessor directives*
  - lines beginning with a #
  - communicates with the preprocessor

```
#include
#define
```

#include

- The include directive instructs the preprocessor to paste the text of the given file into the current file.

- **#include <headername.h>**
  - It finds the header file from the standard system directory.

- **#include "headername.h"**
  - It finds the header file from the current directory.

#define

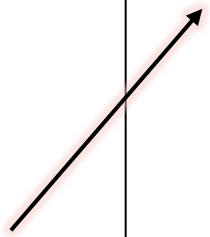
- **#define identifier token_string**
  - The preprocessor replaces every occurrence of ***identifier*** by ***token_string*** in the remainder of the file, except in quoted string.

```
#include <stdio.h>

#define PI 3.14159

int main(void)
{
        double r;
        scanf("%lf", &r);

        printf("pi*r*r = %f\n", r*r*PI);
        return 0;
}
```



```
printf("pi*r*r = %f\n", r*r*3.14159);
```

#define

- It can also summarize some long general codes.

```
#include <stdio.h>

#define PRINT_LINE printf("\n")

int main(void)
{
        ...
        PRINT_LINE;
        ...
}
```

#define

- Syntactic sugar
  - It can alter the syntax of C toward user's preference

```
#define EQ ==
…

while (i EQ 1) {
…
```

Macros with Arguments

- **#define** can be used with parameters.
  - #define identifier(identifier1, identifier2, ...) token_string

```
#define SQ(x) ((x) * (x))
SQ(7 + w)  => ((7 + w) * (7 + w))
SQ(SQ(*p))  => ((((*p) * (*p))) * (((*p) * (*p))))

<Bad Examples>
#define SQ(x) x * x
SQ(a + b)   =>  a + b * a + b

#define SQ(x) (x) * (x)
4 / SQ(2)   =>   4 / (2) * (2)
```

Macros with Arguments

<Bad Examples>

```
#define SQ (x) ((x) * (x))
SQ(7)   =>  (x) ((x) * (x))(7)

#define SQ(x) ((x) * (x));
int a = SQ(2)*3;   =>   int a = ((2) * (2));*3;
```

Macros with Arguments

- Macros are frequently used to replace function calls by inline code.

```
#define min(x, y) (((x) < (y)) ? (x) : (y))
m = min(u, v);

#define min4(a, b, c, d) min(min(a, b), min(c, d))
```

- The macro definition can use both functions and macros in its body.

```
#define SQ(x) ((x) * (x))
#define CUBE(x) (SQ(x) * (x))
#define F_POW(x) sqrt(sqrt(CUBE(x))) /* fractional power: 3/4 */
```

Conditional Compilance

- #if checks whether the value is true, and if so, includes the code until the closing #endif, #elsif, or #else.

```
#if <value>
/* code to execute if this value is true */
#elsif <value2>
/* code to execute if this value2 is true */
#else
/* code to execute otherwise */
#endif
```

[Example]

```
#define DEBUG 1
...

#if DEBUG
printf("debug: a=%d\n", a);
#endif
```

Conditional Compilance

- #ifdef checks whether the given **token** has been #defined earlier in the file or in an included file. If so, it includes everything between it and the closing #else or, if no #else is present, the closing #endif.

```
#ifdef <token>
/* code to include if the token is defined */
#else
/* code to include otherwise */
#endif
```

[Example]

```
#define DEBUG
...

#ifdef DEBUG
printf("debug: a=%d\n", a);
#endif
```

Conditional Compilance

- For instance, the __cplusplus macro is defined in C++ but not in C; you can use this fact to mix C and C++ code using an #ifdef macro.

```
#ifdef __cplusplus
// C++ code
#else
// C code
#endif
```

Predefined Macros (ANSI C)

- __DATE__
  - current date at compile time (e.g. "Jan 14 2012")
- __TIME__
  - current time at compile time (e.g. "22:29:12")
- __FILE__
  - full path to the current file
- __LINE__
  - current line number in the source file