Lexical Macros in C/C++

Saturday, March 10, 2012

Trenton Computer Festival
The College of New Jersey
Ewing, New Jersey

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 (718) 463 1079
gezelter@rlgsc.com
http://www.rlgsc.com

If it makes a sound, SET IT ON SILENT
(e.g., mobiles, pagers, PDA, smartphones)

Note: These slides will be available on our www site. The last slide in this presentation has the URL.

Why macros?

- common operations
- common constants
- reduce chances of error
- improve performance

Simple cases

- `#define XYZ 15`
- `#define XYZ(x) ((x)+5)`

What is a macro?

- sequence of tokens with substitution
- can be nested
- restrictions/limitations

Macros are different than functions

- inline; no CALL involved
- no type checking in the macro expansion
- if a macro is shared, use a header file

Macros are part of a structured approach

- eliminate repetitive coding
- provide wrappers for common functions
- provide shorter incantations for common operations

Why do macros have a bad reputation?

- misperception
- common mistakes
- solution: understanding, care, and caution

Most common error

- understanding compiler phases
- C/C++ macros are lexical preprocessor constructs
- most common error is underuse of parentheses
  - Error: #define f(x) x+5

Two errors: First, Within

- #define f(x) x*5
- what happens if usage is f(y+5)
- “y + 5 * 5”
- Probably not what was intended. Solution:
  - parenthesize formal parameter
  - to wit: #define f(x) (x)*5

Second Error: Without

- #define f(x) (x)+5
- what happens when “j = 10 * f(y)”
- “j = 10 * (y) + 5”
- Probably not what was intended
- Solution:
  - #define f(x) ((x)+5)

Type conversion hazards

- the preprocessor is purely lexical; no type checking
- consider whether typecasting is needed for safety
- e.g., (int)x, (float)y
- remember to parenthesize parameters correctly, (e.g., (int)(x)

Preprocessor symbols

- `__LINE__`
- `__FILE__`
- useful in preprocessor generated error messages

Speaking of error messages

- `#error`
- `#warning`
- use in concert with #if, #elif, #else, #ifdef, #ifndef, #endif
- particularly useful if macro relies upon other definitions

Macros can be complex.

- continuation lines
- common code sequences
- understand possible use cases

Questions?

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 (718) 463 1079
gezelter@rlgsc.com
http://www.rlgsc.com

Session Notes & Materials:
http://www.rlgsc.com/trentoncomputerfestival/2012/index.html