



CS1010

Evan Tay | evantay@comp.nus.edu.sg

https://github.com/DigiPie/cs1010_tut_c09



Today's plan

- Unit 20: C Pre-processor
 - *Problem Set 20.1, 20.2*
- Unit 21: Assert
 - *Problem Set 21.1*
- Unit 22: Efficiency
 - *Problem Set 22.1, 22.2*



UNIT 20

C PRE-PROCESSOR

Recap. PS 20.1. PS 20.2



Recap

- Preprocessor directive
 - A directive which starts with **#**
 - To **#include** a file or,
 - To **#define** a constant

Recap - #include

- #include <stdbool.h>
- #include "cs1010.h"

Recap - #define constant

■ #define constant

- *Use it to define constants which are repeatedly used in code.*

Recap - #define macro

```
#define SQUARE(x) x*x
```

```
#define PI 3.1415926
```

```
int main() {
```

```
    double radius = 4.0;
```

```
    cs1010_print_double(PI*SQUARE(radius));
```

```
}
```

Recap - #define macro

```
#define SQUARE(x) x*x
```

```
#define PI 3.1415926
```

```
int main() {
```

```
    double radius = 4.0;
```

```
    cs1010_print_double(PI*radius*radius);
```

```
}
```


Recap – Macro warnings

- Given:

```
#define SQUARE(x) x*x
```

- SQUARE(radius + 2) evaluates to:

radius + 2*radius + 2

Recap – Macro warnings

- Given:

```
#define SQUARE(x) ((x)*(x))
```

- SQUARE(radius + 2) evaluates to:

```
((radius + 2)*(radius + 2))
```

ENDING NOTE

ALWAYS USE
UPPERCASE WHEN
`#define` CONSTANTS
and MACROS



UNIT 20

C PRE-PROCESSOR

Recap. **PS 20.1.** PS 20.2



Problem Set 20.1 a)

```
#define MIN(a,b) a < b ? a : b
```

```
long i = MIN(10, 20);
```

```
long j = MIN(10, 20) + 1;
```

■ What are the values of **i** and **j**?

Problem Set 20.1 a)

```
#define MIN(a,b) a < b ? a : b
```

```
long i = MIN(10, 20);
```

```
long j = MIN(10, 20) + 1;
```

■ What are the values of **i** and **j**?

– $i = 10 < 20 ? 10 : 20 = 10$

– $j = 10 < 20 ? 10 : 20 + 1$

$j = 10 < 20 ? 10 : 21 = 10$

Problem Set 20.1 a)

```
#define MIN(a,b) (a < b ? a : b)
```

```
long i = MIN(10, 20);
```

```
long j = MIN(10, 20) + 1;
```

■ What are the values of **i** and **j**?

– $i = 10 < 20 ? 10 : 20 = 10$

– $j = (10 < 20 ? 10 : 20) + 1$

$j = 10 + 1 = 11$

Problem Set 20.1 b)

```
#define MIN(a,b) a < b ? a : b
```

```
long i = 10;
```

```
long j = 20;
```

```
long k = MIN(j, i++);
```

■ What are the values of **i** and **k**?

a) $i = 11, k = 10$

b) $i = 11, k = 11$

c) $i = 12, k = 10$

d) $i = 12, k = 11$

Problem Set 20.1 b)

```
#define MIN(a,b) a < b ? a : b
```

```
long i = 10;
```

```
long j = 20;
```

```
long k = MIN(j, i++);
```

■ What are the values of *i* and *k*?

NANI?

a) *i* = 11, *k* = 10

b) *i* = 11, *k* = 11

c) *i* = 12, *k* = 10

d) *i* = 12, *k* = 11

Problem Set 20.1 b)

```
#define MIN(a,b) a < b ? a : b
```

```
long i = 10; long j = 20;
```

```
long k = MIN(j, i++);
```

- $k = 20 < i++ ? 20 : i++$
- $k = 20 < 10++ ? 20 : i++$
- $k = 20 < 10 ? 20 : 11++$
- $k = 11; i = 12$



UNIT 20

C PRE-PROCESSOR

Recap. PS 20.1. [PS 20.2](#)



Problem Set 20.2

■ Original solution

```
#define SWAP(T, x, y) {\n    T temp;\n    temp = x;\n    x = y;\n    y = temp;\n}\n\nint main() {\n    long x = 3.0; long y = -1.0;\n    SWAP(long, x, y);\n}
```

■ Modified version

```
#define SWAP(T, x, y) T temp = x;\n    x = y;\n    y = temp;\n\nint main() {\n    long x = 3.0; long y = -1.0;\n    SWAP(long, x, y);\n}
```

■ What could go wrong?

Problem Set 20.2 - Original

■ Original solution

```
#define SWAP(T, x, y) {\n    T temp;\n    temp = x;\n    x = y;\n    y = temp;\n}\n\nint main() {\n    long x = 3.0; long y = -1.0;\n    SWAP(long, x, y);\n}
```

■ Becomes:

```
int main() {\n    long x = 3.0; long y = -1.0;\n    {\n        long temp;\n        temp = x;\n        x = y;\n        y = temp;\n    }\n}
```

Problem Set 20.2 - Original

■ Modified version

```
#define SWAP(T, x, y) T temp = x;\n    x = y;\n    y = temp;\n\nint main() {\n    long x = 3.0; long y = -1.0;\n    SWAP(long, x, y);\n}
```

■ Becomes:

```
int main() {\n    long x = 3.0; long y = -1.0;\n    long temp = x;\n    x = y;\n    y = temp;\n}
```

■ What could go wrong?

Problem Set 20.2 - Original

- Original becomes:

```
int main() {  
    long x = 3.0; long y = -1.0;  
    {  
        long temp;  
        temp = x;  
        x = y;  
        y = temp;  
    };  
}
```

- Second solution becomes:

```
int main() {  
    long x = 3.0; long y = -1.0;  
    long temp = x;  
    x = y;  
    y = temp;  
}
```

- What could go wrong?

Problem Set 20.2 - Original

- Original becomes:

```
int main() {  
    long temp = 5.0;  
    long x = 3.0; long y = -1.0;  
    {  
        long temp;  
        temp = x;  
        x = y;  
        y = temp;  
    };  
}
```

- Second solution becomes:

```
int main() {  
    long temp = 5.0;  
    long x = 3.0; long y = -1.0;  
    long temp = x;  
    x = y;  
    y = temp;  
}
```

- What could go wrong? This is what happens:
 - *error: redefinition of 'temp'*



UNIT 21

C PRE-PROCESSOR

Recap. PS 21.1.



Recap

```
#include <stdio.h>
#include <assert.h>

int main(){
    char answer;
    printf("Is CS1010 hard? Enter Y/N: ");
    scanf("%c", answer);
    assert(answer == 'N');
}
```

- If wrong answer is given: Assertion 'answer == 'Y'' failed.



UNIT 21

C PRE-PROCESSOR

Recap. [PS 21.1.](#)



Problem Set 21.1

```
void foo(long x) {  
    if (x % 2 == 0) {  
        ...  
    } else {  
        assert(x % 2 == 1);  
    }  
}
```

- Would the assert in Line 5 above ever fail?

Problem Set 21.1

```
void foo(long x) {  
    if (x % 2 == 0) {  
        ...  
    } else {  
        assert(x % 2 == 1);  
    }  
}
```

- Would the assert in Line 5 above ever fail? **Yes**
 - *Consider $x = -1$*

https://github.com/DigiPie/cs1010_tut_c09/blob/master/Tutorial_8/problem21_1.c



UNIT 22

EFFICIENCY

Recap. PS 22.1. PS 22.2



Recap

- In CS1010, we will focus on the efficiency of your code in two senses:
 - *First, your code should not perform redundant work and it should not repeat itself unnecessarily.*
 - *Second, your algorithm should run within a given Big-O running time.*



UNIT 22

EFFICIENCY

Recap. **PS 22.1.** PS 22.2



Problem Set 22.1

- Order the following functions in increasing rate of growth:

1. $\text{Log}_{10} n$

2. $\ln n$

3. \sqrt{n}

4. n

5. $n \ln n$

6. n^2

7. n^4

8. 2^n

9. e^n

10. $n!$



UNIT 22

EFFICIENCY

Recap. PS 22.1. **PS 22.2**



Problem Set 22.2 a)

```
for (long i = 0; i < n; i += 1) {  
    for (long j = 0; j < n; j += 2) {  
        cs1010_println_long(i + j);  
    }  
}
```

- What is the Big-O running time of the following code, in terms of n ?

Problem Set 22.2 a)

```
for (long i = 0; i < n; i += 1) {  
    for (long j = 0; j < n; j += 2) {  
        cs1010_println_long(i + j);  
    }  
}
```

- What is the Big-O running time of the following code, in terms of n ? n^2

Problem Set 22.2 b)

```
for (long i = 0; i < n; i *= 2) {  
    for (long j = 0; j < n; j *= 2) {  
        cs1010_println_long(i + j);  
    }  
}
```

- What is the Big-O running time of the following code, in terms of n ?

Problem Set 22.2 b)

```
for (long i = 0; i < n; i *= 2) {  
    for (long j = 0; j < n; j *= 2) {  
        cs1010_println_long(i + j);  
    }  
}
```

- What is the Big-O running time of the following code, in terms of n ? $(\lg n)^2$

Problem Set 22.2 c)

```
long k = 1;
for (long j = 0; j < n; j += 1) {
    k *= 2;
    for (long i = 0; i < k; i += 1) {
        cs1010_println_long(i + j);
    }
}
```

- What is the Big-O running time of the following code, in terms of n ?

Problem Set 22.2 c)

```
long k = 1;
for (long j = 0; j < n; j += 1) {
    k *= 2;
    for (long i = 0; i < k; i += 1) {
        cs1010_println_long(i + j);
    }
}
```

- What is the Big-O running time of the following code, in terms of n ? $n * (n^2)$



THE END

https://github.com/DigiPie/cs1010_tut_c09

