

w05-Lab

C Coding Style

204111 course standard

Adapted for 204111
by Kittipitch Kuptavanich

Names

ใช้ชื่อที่เหมาะสม

- การตั้งชื่อ เป็นส่วนสำคัญของการเขียนโปรแกรม การตั้งชื่อที่ดีต้องอาศัยกระบวนการคิด และความเข้าใจในตัวโปรแกรม ของโปรแกรมเมอร์ การตั้งชื่อที่เหมาะสม ทำให้ความสัมพันธ์ของแต่ละส่วนของโปรแกรมชัดเจนและมีความหมาย
- If you find all your names could be “Thing” and “DoIt” then you should probably revisit your design.

C Coding Style

- Names
- Formatting
- Documentation
- Miscellaneous

Multiple-word Identifiers

- ในบางกรณีการตั้งชื่อด้วยคำ 1 คำ อาจสื่อความหมายได้ไม่ดีนัก เราสามารถใช้คำมากกว่าหนึ่งคำ (multiple-word) ในการตั้งชื่อ identifier ได้
- ในภาษา C การตั้งชื่อ identifier นั้นห้ามเว้นวรรค จึงต้องแยกคำแต่ละคำด้วยการกำหนดวิธีอื่น ๆ เพื่ออำนวยความสะดวกในการอ่าน วิธีที่ใช้ตั้งชื่อที่พบบ่อยได้แก่
 - Delimiter-separated words
 - Letter-case separated words

Multiple-word Identifiers [2]

- **Delimiter-separated words** คือการหาเครื่องหมาย (ที่ไม่ใช่ตัวเลขและตัวอักษร) มาคั่นระหว่างคำ ในภาษา C (also Pascal, Ruby และ Python) ใช้เครื่องหมาย underscore (also called snake_case)
 - เช่น `word_count`, `eye_color`, `favorite_game`
- **Letter-case separated words** เป็นการแยกคำด้วยตัวพิมพ์ใหญ่ (also called CamelCase) พบมากใน java, C# และ Visual Basic
 - เช่น `wordCount`, `eyeColor`, `favoriteGame`

C Coding Style

5

Function Names (2)

- การใช้คำต่อท้าย (Suffix)
 - `max` - to mean the maximum value something can have.
 - `cnt` - the current count of a running count variable.
 - `key` - key value.
- **For example:**
 - `retry_max`: maximum number of retries
 - `retry_cnt`: the current retry count

C Coding Style

7

Function Names

- แต่ละ **function** (ฟังก์ชัน) มีหน้าที่ หรือกระบวนการที่ทำดังนี้ ชื่อของ function ควรสื่อให้เห็นว่า function นั้นทำอะไร:
- ตัวอย่าง
 - `check_for_errors()` instead of `error_check()`,
 - `dump_data_to_file()` instead of `data_file()`.
- การตั้งชื่อ function ในลักษณะนี้ จะทำให้สามารถแยกได้ชัดเจนว่า นี่คือชื่อ function หรือ ชื่อ data
- ข้อมูลชนิด **struct** (204112) มักมีชื่อเป็นคำนาม (noun) ดังนั้น เพื่ออำนวยความสะดวกในการอ่านและทำความเข้าใจโปรแกรมการตั้งชื่อ function ควรเป็น คำกริยา (verb)

C Coding Style

6

Function Names (3)

- **การใช้คำนำหน้า (Prefix)**
 - `is` - to ask a question about something.
Whenever someone sees “is” they will know it's a question.
 - `get` - get a value.
 - `set` - set a value.
- **For example:** `is_hit_retry_limit()`

C Coding Style

8

Include Units in Names

- ถ้า **variable** ที่ต้องการตั้งชื่อ มีการเก็บค่าของเวลา, หน่วยน้ำหนัก, ความยาว หรือค่าอื่น ๆ ที่มีหน่วยหลายแบบ ควรมีการใส่หน่วยไว้ในชื่อ **variable** ด้วย เช่น

- `unsigned int timeout_msecs;`
- `unsigned int my_weight_lbs;`

C Coding Style

9

Variable Names

- use all lower case letters
- use `'_'` as the word separator.
- Justification
 - With this approach the scope of the variable is clear in the code.
 - Now all variables look different and are identifiable in the code.
- Example

```
int handle_error (int error_number) {
    int           error= OsErr();
    Time          time_of_error;
    ErrorProcessor error_processor;
}
```



C Coding Style

11

Structure Names (204112)

- Use underscores (`'_'`) to separate name components
- Declare variables from large to small size (same size together) then by name in alphabetical order
- More at:

<http://users.ece.cmu.edu/~eno/coding/CCodingStandard.html>

C Coding Style

10

Pointer Variables (204112)

- Place the `*` close to the variable name not pointer type
- Example

```
char *name= NULL;      /* YES */
char* name, address;  /* NO!! */
```

C Coding Style

12

Miscellaneous

- Global Constants
 - Global constants should be all caps with '_' separators.
 - Example


```
const int A_GLOBAL_CONSTANT= 5
```
- Enum Names
 - Labels All Upper Case with '_' Word Separators
 - Example


```
enum PinStateType {
    PIN_OFF,
    PIN_ON
};
```

Blocks

- In computer programming, a block is a section of code which is grouped together. Blocks consist of one or more declarations and statements
- In C programming language family, blocks are delimited by curly braces - "{" and "}"

C Coding Style

- Names
- Formatting
- Documentation
- Miscellaneous

Indentation and White Space

- Within a block, all program statements are indented
 - Use the same indentation for similar groups of statements
 - Indentation and spacing should reflect the block structure of the code
 - DON'T do this:

```
int i;main(){for(;i["<i;++i){--i;}"]];read('---'
,i+++"hell\
o, world!\n",'//'))};}read(j,i,p){write(j/p+p,i---
j,i/i);}
```

Brace Placement

- Of the three major brace placement strategies one is recommended (K&R style):

- each function has its opening brace at the next line

```
int main(int argc, char *argv[])
{
    if (condition) {
        ...
    }

    while (condition) {
        ...
    }
}
```

- For other cases, braces openings are on the same line

C Coding Style

17

When Braces are Needed

- Always Uses Braces Form

- even if there is only a single statement within the braces. For example:

```
if (1 == some_value) {
    some_value = 2;
}
```

- **เหตุผล** หากไม่ใส่ เมื่อมีการเพิ่ม code ในบรรทัดถัด ๆ ไปที่อยู่ภายใต้เงื่อนไข if อาจลืมใส่วงเล็บปีกกดได้

- One Line Form

```
if (1 == some_value) some_value = 2;
```

- It provides safety when adding new lines while maintaining a compact readable form.

C Coding Style

18

Parenthesis ()

- DO NOT put parenthesis next to keywords. Put a space between

- `while (condition)` and **NOT** `while(condition)`

- DO put parenthesis next to function names.

- `printf()` and **NOT** `printf ()`

- Do not use parenthesis in return statements when it's not necessary.

- `return 1` and **NOT** `return (1)`

C Coding Style

19

Line Length

- A Line Should Not Exceed 78 Characters

- Justification

- Even though with big monitors we stretch windows wide our printers can only print so wide. And we still need to print code.

- The wider the window the fewer windows we can have on a screen. More windows is better than wider windows.

- We even view and print diff output correctly on all terminals and printers.

C Coding Style

20

If - Else Formatting

- Layout

```
if (condition) {
} else if (condition) {
} else {
}
```

C Coding Style

21

switch Formatting

- Falling through a case statement into the next case statement shall be permitted as long as a comment is included.
- The default case should always be present and trigger an error if it should not be reached, yet is reached.
- If you need to create variables put all the code in a block.

C Coding Style

22

One Variable Per Line

- DON'T

```
char **a, *x;
```

- DO

```
char **a = 0; /* comments*/
char *x = 0; /* comments */
```

- Reasons

- Comments can be added for the variable on the line.
- It's clear that the variables are initialized.
- Declarations are clear
 - reduces the probability of declaring a pointer when you meant to declare just a char.

switch Formatting (2)

- Example

```
switch (...) {
    case 1:
        ...
        /* ถ้าไม่ใส่ break ระหว่าง case ต้องมี commentแสดงเหตุผล
         * ของการไม่ใส่ break */
    case 2:
    {
        int v; /*ถ้าจะ declare variable ต้องมีปีกกา (block) */
        ...
    }
    break;

    default: /* default ต้องมีเสมอ เพื่อจับกรณีที่ไม่เข้า case ใด ๆ */
}
```

C Coding Style

23

C Coding Style

24

Initialize all Variables

- You shall **always initialize variables**. Always.
Every time. The compiler gcc with the flag -W may catch operations on uninitialized variables, but it may also not.

C Coding Style

25

Usually Avoid Embedded Assignments

- DON'T**

```
d = (a = b + c) + r;
```

- DO**

```
a = b + c;
d = a + r;
```

- Reason**

- Embedded statement are harder to read and understand (especially in the long run)

Note: In some constructs there is no better way to accomplish the results without making the code bulkier and less readable.

```
while (EOF != (c = getchar())){
    process the character
}
```

C Coding Style

27

Short Functions

- Functions should limit themselves to a single page of code.
- Reasons
 - The idea is that each method represents a technique for achieving a **single objective**.
 - Most arguments of inefficiency turn out to be false in the long run.

C Coding Style

26

C Coding Style

- Names**

- Formatting**

- Documentation**

- Miscellaneous**

C Coding Style

28

Comments Should Tell a Story

- ให้ระลึกเสมอว่า comment ใน program เป็นการบอกเรื่องราวของระบบของ program
- comment ในแต่ละส่วน เมื่อรวมเข้าด้วยกัน ควรสามารถอธิบายให้ผู้อ่าน (ณ เวลาภายหลัง) ทราบถึงกระบวนการต่าง ๆ ที่เกิดขึ้นใน program และจุดประสงค์ของกระบวนการนั้น ๆ

Include Statement Documentation

- #include statements should be documented, telling the user why a particular file was included.

Commenting Function Declarations

- Function headers should be in the file where they are declared. This means that most likely the functions will have a header in the .h file

C Coding Style

- Names
- Formatting
- Documentation
- Miscellaneous

General advice

- Don't use **floating-point** variables where discrete values (integers) are needed.
 - Don't use a float for a loop counter
 - Always test floating-point numbers as <= or >=, never use an exact comparison (== or !=) as they are estimations

General advice (3)

- No global variable
- No goto

General advice (2)

- = and == are not the same

- DON'T

```
if (abool= bbool) { ... } /* confusing */
  • Does the programmer really mean assignment here?
  • Is it a typo?
```

- DO

```
abool = bbool;
if (abool) { ... }
```

No Magic Numbers

- A magic number is a bare naked number used in source code. It's magic because no-one has a clue what it means including the author inside 3 months.

- For example:

```
if (22 == foo) { start_thermo_nuclear_war(); }
else if (19 == foo) { refund_lotso_money(); }
else if (16 == foo) { infinite_loop(); }
else { cry_cause_im_lost(); }
```

No Magic Numbers (2)

- Use `#define`, `const` or `enum` instead

```
#define PRESIDENT_WENT_CRAZY 22
const int WE_GOOFED= 19;
enum {
    THEY_DIDNT_PAY= 16
};

if (PRESIDENT_WENT_CRAZY == foo) {
    start_thermo_nuclear_war();
} else if (WE_GOOFED == foo) {
    refund_lotso_money();
} else if (THEY_DIDNT_PAY == foo) {
    infinite_loop();
} else {
    happy_days_i_know_why_im_here();
}
```

We will learn the differences of `#define`, `enum`, and `const int` later in the course

References

- <http://users.ece.cmu.edu/~eno/coding/CCodingStandard.html>
- <http://www.doc.ic.ac.uk/lab/cplus/cstyle.html>
- [http://en.wikipedia.org/wiki/Naming_convention_\(programming\)](http://en.wikipedia.org/wiki/Naming_convention_(programming))