C Programming Part 1 - 2

By Tanaroeg O-Charoen

Variable

Variable หรือ **ตัวแปร** เป็นตัวที่เอาไว้เก็บค่าต่างๆ ซึ่งก็จะมี Type ที่ต่างกัน โดยการตั้งชื่อตัวแปรนั้นเป็นจุดสำคัญมากที่จะทำให้ Code เราอ่านง่ายการตั้งตัวแปรควรมีความหมาย พยายามเลี่ยงการใช้ตัวย่อที่ย่อมากๆจนคนอื่นไม่เข้าใจ

โดยจะมีข้อห้ามในการตั้งตัวแปรคือ **ไม่ใช้ตัวเลขนำหน้าหรือตัวอักษรพิเศษ หรือเป็นคำที่เป็น Keyword ของภาษา C** เช่น if, double เป็นต้น เราจะไม่ใช้ชื่อตัวแปรที่ไม่ใช่ค่าคงที่เป็นตัวใหญ่ทั้งหมด และเรายังมีหลักการในการตั้งชื่อตัวแปรในกรณีที่ตัวแปรมีค่าหลายคำอยู่คือ

- snake_case
- camelCase
- CamelCase

Data Type

ในภาษา C นั้นเราจะมี Data type ที่เป็น Primative Data Type อยู่ 2 ประเภทหลักๆคือ จำนวนเต็ม และ ทศนิยม

Integer Types

ตัวแปรประเภท Interger นั้นจะเป็นการเก็บค่าของ **เลขจำนวนเต็ม** ซึ่งก็จะมีประเภทที่แยกย่อยลงไปอีก โดยจะจำแนกตามขนาดของข้อมูล และ ช่วงของข้อมูล

Type	Storage Size	Value range
char	1 byte	-128 to 127
short	2 byte	-32,768 to 32,767
int	4 byte	-2,147,483,648 to 2,147,483,647
long	8 bytes or (4bytes for 32 bit OS)	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

keyword `unsigned` นำหน้า Data type ได้เพื่อจะทำให้ตัวแปรนั้นๆ ไม่เก็บค่าลบ จะเก็บค่าแค่ตั้งแต่ 0 ขึ้นไปได้
keyword `short` นำหน้า data type เพื่อทำให้ขนาดของตัวแปรลดลง (2 เท่า)
keyword `long` นำหน้า data type เพื่อทำให้ขนาดของตัวแปรเพิ่มขึ้น (2 เท่า)

**หมายเหตุ* keyword `short` กับ `long` สามารถใช้นำหน้าตัวมันเองได้เช่น `long long`

Floating-Point Types

ตัวแปรประเภท Floating-Point นั้นจะเป็นตัวแปรที่เอาไว้เก็บค่าของ **เลขทศนิยม** โดยจะจำแนกตามขนาดของข้อมูล และ ช่วงของข้อมูล

Type	Storage Size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

**หมายเหตุ* การเก็บทศนิยมของคอมพิวเตอร์จะเก็บโดยใช้ IEEE-754 เพราะว่าคอมพิวเตอร์นั้นสามารถเก็บได้เฉพาะค่า 1 และ 0 เท่านั้นซึ่งจะทำให้ความแม่นยำของทศนิยมจะไม่ถูกต้องไปทั้งหมด ***หมายเหตุ* เราสามารถใช้ `sizeof(type|variable)` เพื่อแสดงความจุของตัวแปรนั้นได้

Variable Define

ในการสร้างตัวแปรขึ้นมานั้น ตัวแปรนั้นๆหลักๆจะต้องประกอบด้วย
`data type` `variable name` หรือสามารถกำหนดค่าเริ่มต้นของตัวแปรนั้นได้
`data type` `variable name` `=` `value` เช่น

```c
#include <stdio.h>

int main() {
    int a = 5;
    float b;
    int x = 5, y = 10; // ทั้ง x และ y เป็น int เหมือนกัน
    double k = a + 5; // สามารถใช้ Expression ในการสร้างตัวแปรได้
}
```

ข้อควรระวัง

1. การนิยามตัวแปร 2 ตัวติดกันถ้าหากเราใส่ค่าของตัวแรกเกินขนาด ข้อมูลจะเกินไปถึงตัวแปรถัดไปได้
2. การที่เรานิยามตัวแปรแต่ไม่มีค่า ค่านั้นอาจจะไม่เป็น 0 โดยจะเป็นค่าที่ค้างอยู่ใน Memory หรือเป็นค่าขยะนั้นเอง

ในจำนวนเต็มจะมีความพิเศษในการนิยามเราสามารถใช้ character 1 ตัวเพื่อสร้างตัวแปรได้ โดยจะได้ค่าเป็นตัวแหน่งใน ASCII Table

```c
#include <stdio.h>

int main() {
    int x = 'A'; // A จะมีค่าเป็น 65
}
```

Boolean

ตามปกติภาษา C จะไม่มีตัวแปรประเภท boolean แต่เราจะใช้ตัวเลขหรือตัวอักษรมาเป็นค่าความจริงแทน

True

- ตัวเลขทุกค่าที่ไม่ใช่ 0
- Chacter ทุกตัวที่ไม่ใช่ Null Character ('\0')

False

- ตัวเลขที่เป็น 0
- Null Character ('\0')

String

จะเห็นว่าภาษา C ก็ไม่มี Data type String เช่นกันแต่ตามที่เราทราบว่า String ก็คือสายของอักขระที่มี อักขระ หรือ Character มาเชื่อมต่อกันนั้นเอง ซึ่งเราก็จะใช้เป็น Character Array แทน

```c
#include <stdio.h>

int main() {
    char str2[10] = "String 2"; // เป็นการสร้าง String ที่มีการกำหนดขนาดมีความปลอดภัย
    char str1[] = "String 1"; // เป็นการสร้าง String ที่ไม่ได้มีการกำหนดขนาด แบบนี้ไม่ควรทำเพราะอาจจะเกิด Secmentation Fault (การ Access Memory
}
```

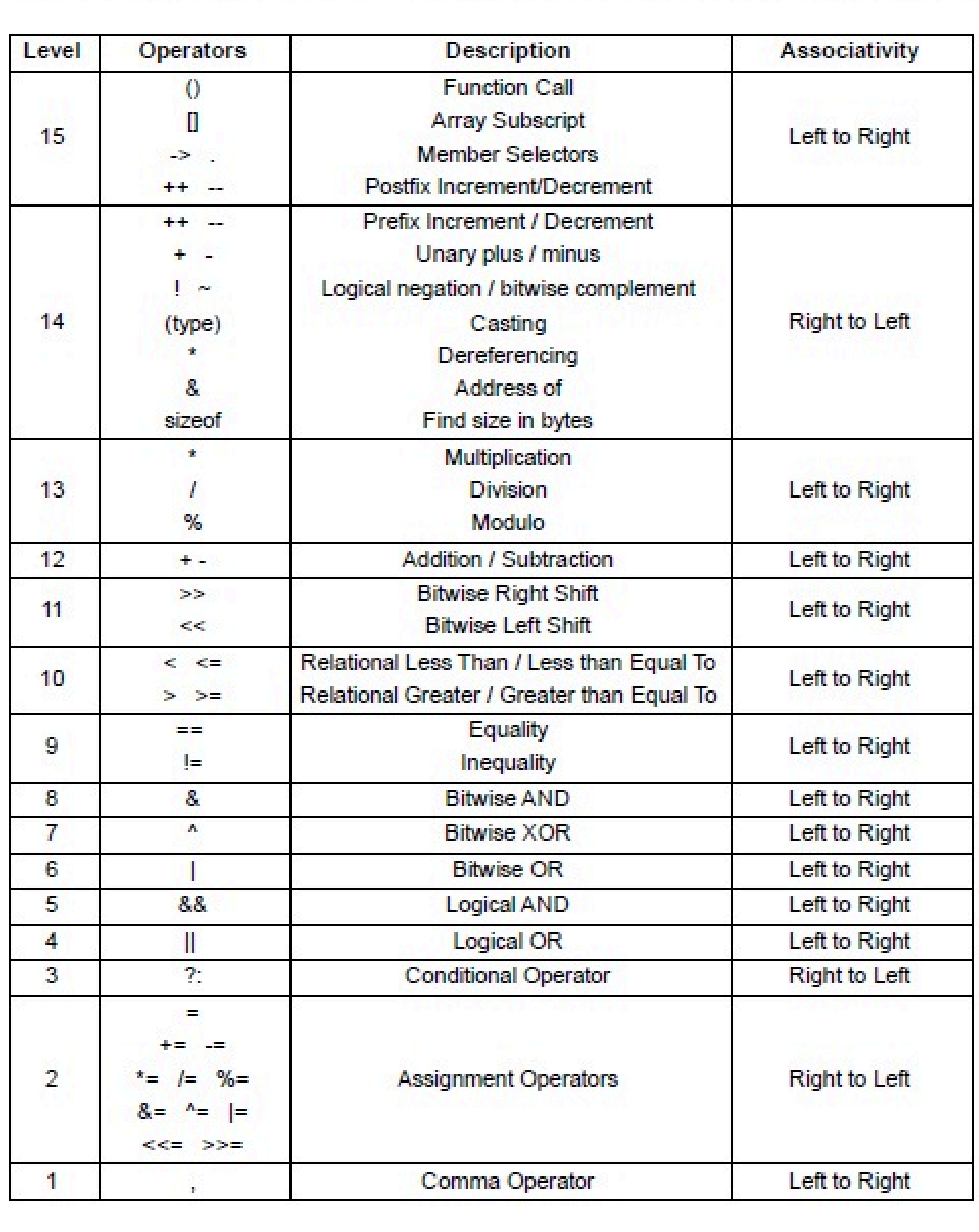
เพิ่มเติม การกำหนดขนาดของ String นั้นเราควรจะเผื่อไว้ 1 ตัวให้กำกับ Null Character (ตัวหยุดของ String)

Operator

Operator หรือตัวดำเนินการของภาษา C โดยจะมี Operator 2 แบบใหญ่ๆ

- **Unary Operator** คือเป็นตัวดำเนินการที่ดำเนินการกับ Operand 1 ตัว
- **Binary Operator** คือเป็นตัวแปรที่ดำเนินการกับ Operand 2 ตัว

ซึ่งก็จะแบ่งย่อยลงไปอีก ซึ่งแต่ละ Operator ก็จะมีการเรียงความสำคัญที่ต่างกันดังตาราง



Level	Operators	Description	Associativity
15	() [] -> . ++ --	Function Call Array Subscript Member Selectors Postfix Increment/Decrement	Left to Right
14	++ -- + - ! ~ (type) * & sizeof	Prefix Increment / Decrement Unary plus / minus Logical negation / bitwise complement Casting Dereferencing Address of Find size in bytes	Right to Left
13	* / %	Multiplication Division Modulo	Left to Right
12	+ -	Addition / Subtraction	Left to Right
11	>> <<	Bitwise Right Shift Bitwise Left Shift	Left to Right
10	< <= > >=	Relational Less Than / Less than Equal To Relational Greater / Greater than Equal To	Left to Right
9	== !=	Equality Inequality	Left to Right
8	&	Bitwise AND	Left to Right
7	^	Bitwise XOR	Left to Right
6	|	Bitwise OR	Left to Right
5	&&	Logical AND	Left to Right
4	||	Logical OR	Left to Right
3	?:	Conditional Operator	Right to Left
2	= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operators	Right to Left
1	,	Comma Operator	Left to Right

ตัวที่อยู่สูงกว่า (Precedence สูงกว่า) โปรแกรมก็จะทำ Operator นั้นๆก่อนตัวอื่น แล้ว Operator ยังมีคุณสมบัติในการที่จะ Chain กันได้อีก เช่น

```c
#include <stdio.h>

int main() {
    int x;
    int y;
    x = 5 + 2 * 2;
    y = x = 10;
}
```

Arithmetic Operator

ตัวดำเนินการทางคณิตศาสตร์จะประกอบด้วย

Operator	Description
+	Addition
-	Subtracts
*	Multiplication
/	Division
%	Modulo
++	Increase
--	Decrease

Relational Operators

ตัวดำเนินการที่เอาไว้เปรียบเทียบ

Operator	Description
==	Equality
!=	Inequality
>	Greater
>=	Greater or equal to
<	Less than
<=	Less than or equal to

Logical Operators

ตัวดำเนินการเปรียบเทียบทาง Logical

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

Bitwise Operators

ตัวเดินการกับ Bit

Operator	Description
&	Binary AND
|	Binary OR
^	Binary XOR
~	Binary One's compliment
<<	Binary Left Shift
>>	Binary Right Shift

Assignment Operators

เป็น Operator ที่ใช้สำหรับการกำหนดค่าให้กับตัวแปร

Operator	Description
=	Assignment
+=	Add and assignment
-=	Subtract and assignment
*=	Multiply and assignment
/=	Divide and assignment
%=	Modulus and assignment
<<=	Left SHIFT and assignment
>>=	Right SHIFT and assignment
&=	Bitwise AND and assignment
|=	Bitwise OR and assignment
^=	Bitwise Exclusive OR and assignment

Misc Opeators

Operator	Description
sizeof()	Return size of a variable or type
&	Return the address of variable
*	Pointer to a variable
? :	Condition Expression [Condition ? true : false]
,	Comma Operator (Connect expression)

Type Conversion

Explicit Type Conversion

เราสามารถ Casting Type ของตัวแปรได้ (แต่ไม่ได้เขียนทับค่าของตัวแปรนั้นๆ) โดยการใช้

```
(type) variable
```

ตัวอย่างเช่นเราต้องการให้ตัวแปร Float กลายเป็น Int

```c
#include <stdio.h>

int main() {
    float x = 2.95;
    printf("Result: %d", (int) x);
}
```

```
Result: 2
```

จะสังเกตว่าจะไม่มีการปัดเศษขึ้นแต่จะเป็นการตัดเศษทิ้งเลย

Implicit Type Conversion

ในภาษาซีการที่นำตัวแปรมากระทำกันผ่าน Operator Compiler จะทำการแปลง Type ที่ Narrow -> Wider เพื่อไม่ใช้เสียข้อมูลไป

แหล่งที่มา GeeksForGeek

ตัวอย่างการแปลง

Character <--> integer
boolean <--> integer

Example

ในบางกรณีเราอาจจะใช้ใน Expression ที่กระทำกันแบบคนละ Type ลองพิจารณา Code ข้างล่าง

```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = 1;
    printf("Result: %f", b / a);
}
```

เราจะหยิบ Type casting เข้ามาช่วย

```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = 1;
    printf("Result: %d", b / (float) a);
}
```