สัปดาห์ที่ 2 Types, Operators, and Expressions

เรียบเรียงโดย ชาคริต วัชโรภาส

วิชา 01418113 Computer Programming 1

ทุกคำสั่งภาษา C จะปิดด้วย ;

1. การใช้งานตัวแปร

ตัวแปรภาษา C ยังไม่สามารถถูกใช้งานได้จนกว่าจะมีการประกาศตัวแปร (declare) ดังตัวอย่างที่แสดงให้เห็นถึงปัญหานี้ในด้านล่าง

คอมเม้น : //

In [1]:
```c
#include <stdio.h>

int main()
{
   x = 10;
}
```

```
/tmp/tmp2diaa4bp.c: In function ‘main’:
/tmp/tmp2diaa4bp.c:5:4: error: ‘x’ undeclared (first use in this function)
    x = 10;
    ^
/tmp/tmp2diaa4bp.c:5:4: note: each undeclared identifier is reported only once for each function i
t appears in
[C kernel] GCC exited with code 1, the executable will not be executed
```

การประกาศตัวแปรทำได้ด้วยการระบุประเภทของตัวแปร แล้วตามด้วยชื่อตัวแปร

```c
int x;
```

ดังที่แสดงข้างบนนี้เป็นการประกาศตัวแปร x ที่มีประเภทของตัวแปรเป็นเลขจำนวนเต็ม (integer)

In [2]:
```c
#include <stdio.h>

int main()
{
   int x;
   float y;

   x = 10;
   y = 2.54;
}
```

การประกาศตัวแปร ประเภทบอกข้างหน้าว่าเป็นชนิดใด

1.1 ชื่อตัวแปร (Variable Names)

ห้ามขึ้นด้วยตัวเลข 1text ✗ text1 ✓

ข้อกำหนดในการตั้งชื่อตัวแปรของภาษา C เป็นดังนี้

A-Z, a-z 0-9

- ชื่อตัวแปรประกอบด้วยตัวอักขระและตัวเลข โดยที่ตัวอักษรตัวแรกของชื่อตัวแปรต้องเป็นอักขระ
- เครื่องหมาย _ นับว่าเป็นอักขระ
- การใช้เครื่องหมาย _ คั่นระหว่างคำในชื่อตัวแปรช่วยให้การอ่านโค้ดทำได้ง่าย
- อักขระภาษาอังกฤษตัวพิมพ์ใหญ่และตัวพิมพ์เล็กถือว่าเป็นอักขระที่แตกต่างกัน
- เรามักจะใช้อักขระภาษาอังกฤษที่เป็นตัวพิมพ์ใหญ่ทั้งหมดกับการนิยามค่า symbolic constant
- ห้ามชื่อตัวแปรซ้ำกับคีย์เวิร์ดในภาษา C เช่น if, else, int, float เป็นต้น

1.2 การตั้งชื่อตัวแปร

- การตั้งชื่อตัวแปรควรคำนึงถึงในเรื่องของ readability คือ ให้อ่านแล้วเข้าใจได้ง่าย
- ถึงแม้ว่า **a = b * c;** จะถูกไวยกรณ์ในภาษา C แต่จุดประสงค์ของการใช้งานไม่ชัดเจน
- ต.ย. **weekly_pay = hours_worked * hourly_pay_rate;** สื่อความหมายมากกว่า
- รูปแบบชื่อตัวแปรที่มักใช้กัน
  - snake_case, student_id, student_name
  - คั่นแต่ละคำในตัวแปรด้วยเครื่องหมาย _ (underscore)
  - CamelCase, StudentID, StudentName
  - ตัวอักษรตัวแรกของแต่ละคำเป็นอักขระพิมพ์ใหญ่นอกนั้นเป็นอักขระพิมพ์เล็ก
  - camelCase, studentID, studentName
  - บางครั้ง CamelCase มีการใช้งานที่ตัวอักษรในคำแรกเป็นอักขระพิมพ์เล็ก หลังจากนั้นใช้หลักการของ CamelCase ตามปกติ
- เราใช้หลักการตั้งชื่อตัวแปรนี้ในการตั้งชื่อฟังก์ชันได้ด้วย

2. ประเภทข้อมูลและขนาด (Data Types and Sizes)

- ประเภทข้อมูลที่ใช้เก็บเลขจำนวนเต็ม
  - char (ใช้หน่วยความจำ 1 ไบท์)
  - int (ส่วนใหญ่ใช้หน่วยความจำ 4 ไบท์ แต่ทั้งนี้ขึ้นอยู่กับตัวคอมไพเลอร์ด้วย)
- ประเภทข้อมูลที่ใช้เก็บเลขจำนวนจริง
  - float (ใช้หน่วยความจำ 4 ไบท์)
  - double (ใช้หน่วยความจำ 8 ไบท์)

- คีย์เวิร์ด short ช่วยลดแพื้นที่จัดเก็บให้กับประเภทตัวแปรจำนวนเต็ม ส่วนคีย์เวิร์ด long ช่วยเพิ่มพื้นที่จัดเก็บข้อมูล
  - short int
  - long int
- คีย์เวิร์ด signed ใช้กำหนดประเภทตัวแปรจำนวนเต็มให้สามารถเก็บค่าที่มีเครื่องหมาย (ซึ่งปกติตัวแปรจำนวนเต็มก็เก็บค่าที่มีเครื่องหมายอยู่แล้ว) ส่วนคีย์เวิร์ด unsigned ใช้กำหนดประเภทตัวแปรจำนวนเต็มให้สามารถเก็บค่าเฉพาะค่าบวก
  - signed int (ซึ่งหากเราไม่ใส่คีย์เวิร์ด signed ก็ยังให้ผลเหมือนกัน)
  - unsigned char
- เราสามารถใช้ sizeof ซึ่งเป็น macro ในการตรวจสอบขนาดตัวแปรและประเภทข้อมูล
- คีย์เวิร์ด void มักจะใช้ในกรณีของการนิยามฟังก์ชันที่ไม่มีการส่งค่ากลับไปยังผู้เรียก และยังใช้ในการกำหนดให้กับตัวแปรที่ไม่มีประเภทข้อมูล

In [3]:

```c
#include <stdio.h>

int main()
{  char c;
   int i;
   float f;
   double d;

   printf("size of c = %d\n", sizeof(c));
   printf("size of i = %d\n", sizeof(i));
   printf("size of f = %d\n", sizeof(f));
   printf("size of d = %d\n", sizeof(d));
}
```

```
size of c = 1
size of i = 4
size of f = 4
size of d = 8
```

In [4]:

```c
#include <stdio.h>

int main()
{
   printf("sizeof(char)                = %d\n", sizeof(char));
   printf("sizeof(int)                 = %d\n", sizeof(int));
   printf("sizeof(float)               = %d\n", sizeof(float));
   printf("sizeof(double)              = %d\n", sizeof(double));
   printf("sizeof(short int)           = %d\n", sizeof(short int));
   printf("sizeof(long int)            = %d\n", sizeof(long int));
   printf("sizeof(unsigend char)       = %d\n", sizeof(unsigned char));
   printf("sizeof(unsigned int)        = %d\n", sizeof(unsigned int));
   printf("sizeof(unsigned short int)  = %d\n", sizeof(unsigned short int));
   printf("sizeof(void)                = %d\n", sizeof(void));
}
```

```
sizeof(char)                = 1
sizeof(int)                 = 4
sizeof(float)               = 4
sizeof(double)              = 8
sizeof(short int)           = 2
sizeof(long int)            = 8
sizeof(unsigend char)       = 1
sizeof(unsigned int)        = 4
sizeof(unsigned short int)  = 2
sizeof(void)                = 1
```

---

3. ค่าคงที่ (Constants)

printf("%.f\n", 1e-2)

ค่าคงที่จำนวนเต็ม

- an integer constant, ต.ย. 1234
- a long constant, ต.ย. 1234567891 หรือ 123456789L
- unsigned constants (ตัวเลขจำนวนเต็มตามด้วย u หรือ U)
- unsigned long constants (ตัวเลขจำนวนเต็มตามด้วย ul หรือ UL)

ค่าคงที่จำนวนจริง

.0 : float

- floating-point constants เช่น 123.4 หรือ 1e-2
- floating-point constants เช่น ตัวเลขจำนวนเต็มตามด้วย f หรือ F
- double constants (ตามด้วย l หรือ L)

ค่าคงที่ของเลขฐาน 2, ฐาน 8, และ ฐาน 16

- 0 (ศูนย์) แล้วตามตัวเลขจำนวนเต็มจะบ่งบอกถึงเลขฐาน 8 (octal)
- 0x หรือ 0X จะบ่งบอกถึงเลขฐาน 16 (hexadecimal)
- ตัวอย่างเช่น 037 จะมีค่าเท่ากับ 31 (ในฐาน 10), 0x1f หรือ 0x1F จะเท่ากับ ...
- 0b หรือ 0B จะบ่งบอกถึงเลขฐาน 2 (binary)
- ตัวอย่างเช่น 0123 เป็นค่าคงที่นเลขฐาน 8, 0x123 เป็นค่าคงที่ในเลขฐาน 16, 0b1001 เป็นค่าคงที่ในเลขฐาน 2

รูปแบบการพิมพ์ (print format) สำหรับเลขฐาน 8 และฐาน 16

- ใช้ %o ในการพิมพ์เลขฐาน 8
- ใช่ %x ในการพิมพ์เลขฐาน 16 (ใช้ %X หากต้องการให้ตัวอักษรเอถึงเอฟเป็นอักขระพิมพ์ใหญ่)

ค่าคงที่ของตัวอักษร (Character Constant)

- character constant จะเขียนอยู่ในรูปตัวอักขระหนึ่งตัวโดยถูกห้อมล้อมด้วยเครื่องหมาย*ฝนทอง* (single quote)
- ต.ย. 'x', '0'
- ASCII character and integer value conversion
- อักขระบางตัวในตาราง ASCII ถูกอ้างถึงโดยใช้ escape sequences เช่น \n หมายถึง newline
- '\0' เป็นอักขระตัวแรกในตาราง ASCII ซึ่งมีค่าโค้ดในตารางเป็น 0 และถูกกำหนดให้เป็น null character

\a	alert (bell) character	\\	backslash
\b	backspace	\?	question mark
\f	formfeed	\'	single quote
\n	newline	\"	double quote
\r	carriage return	*ooo*	octal number
\t	horizontal tab	\x*hh*	hexadecimal number
\v	vertical tab		

ค่าคงที่สตริง (String Constant)

- string constant เป็นกลุ่มของอักขระที่ห้อมล้อมด้วยเครื่องหมาย*ฟันหนู* (double quote)
- ต.ย. "This is a sting constant."
- "" เป็น empty string หรือสตริงเปล่า
- The internal representation of a sting has a null character '\0' at the end, so the physical storage required is one more than the number of characters written between the quotes. เรื่องนี้จะกลับมาพูดอีกครั้งตอนบรรยายถึงเรื่องอาเรย์ของอักขระ

In [5]:
```c
// หากต้องการทดสอบความรู้

#include <stdio.h>

int main()
{
}
```

4. การประกาศตัวแปร (Variable Declarations)

- ตัวแปรในภาษา C ต้องถูกประกาศก่อนถูกใช้งาน
- **ตามข้อกำหนดมาตรฐาน C89 หรือ ANSI การประกาศตัวแปรจะต้องถูกกำหนดไว้ตอนส่วนต้นของ block เท่านั้น**
- การประกาศเป็นการระบุประเภทข้อมูลให้กับตัวแปร

```c
int lower, upper, step; char c, line[1000];
```

- การกำหนดค่าเริ่มต้นให้กับตัวแปรสามารถทำได้ในขณะที่ประกาศตัวแปร

```
int lower = 10, upper = 30;
```

- กำหนดค่าเริ่มต้นด้วยนิพจน์

```
char c = 'A' + 1;
```

- เราสามารถประกาศตัวแปรโดยให้เป็นค่าคงที่ที่ไม่สามารถถูกเปลี่ยนค่าได้ตลอดการทำงานของโปรแกรม

```
const int step = 20;
```

- const สามารถใช้กับตัวแปรอาเรย์เพื่อไม่ให้ข้อมูลภายในอาเรย์ถูกเปลี่ยนแปลงได้ โดยเฉพาะในกรณีที่เราต้องการส่งผ่านอาเรย์เข้าไปในฟังก์ชัน แต่ไม่ต้องการให้ฟังก์ชันสามารถเปลี่ยนแปลงค่าในอาเรย์ได้

```
int countCharacters(const char[]);
```

4.1 printf() and puts()

```
#include <stdio.h>
int printf(const char *format, ...);
int puts(const char *s);
```

- รูปแบบการพิมพ์ (format) ของ printf()
  - %d print as decimal integer
  - %6d print as decimal integer, at least 6 character wide
  - %06d print as decimal integer, at least 6 character wide with 0 padding
  - %f print as floating point
  - %6f print as floating point, at least 6 character wide
  - %.2f print as floating point, 2 characters after decimal point
  - %6.2f print as floating point, at least 6 wide and 2 after decimal point
- รูปแบบอื่นที่ใช้ในบางกรณี
  - %o for octal,
  - %x for hexadecimal,
  - %c for character,
  - %s for character string
  - %% for % itself. ถ้าได้ % ออกมา 1 ตัวบนหน้าจอ *

4.2 Character Input and Output รับค่าจาก keyboard

```
#include <stdio.h>
int getchar(void);
int putchar(int c);
```

- getchar() ใช้รับอักขระ 1 ตัวจาก stdin เข้ามาในโปรแกรม โดยส่งค่ากลับไปยังผู้เรียกใช้ฟังก์ชัน
- putchar() ใช้ส่งอักขระ 1 ตัวออกไปยัง stdout

```
char c;
c = getchar();
c = c + 1;
putchar(c);
```

- EOF เป็นค่าคงที่ที่ถูกกำหนด (defined) ไว้ในไฟล์ stdio.h ซึ่งบ่งบอกถึง end-of-file หรือจุดสิ้นสุดข้อมูล
- ฟังก์ชัน getchar() จะส่งค่ากลับเป็น EOF เมื่ออักขระที่ถูกส่งเข้ามาในโปรแกรมหมดแล้ว

- ลองรันโปรแกรมข้างล่างนี้โดยใช้ MSYS2

```
#include <stdio.h>
int main()
{  int c;

   c = getchar();
   while (c != EOF) {
      c = c + 1;
      putchar(c);
      c = getchar();
   }
}
```

5. Arithmetic Operators (ตัวดำเนินการทางคณิตศาสตร์)

- +, -, *, /, %
- % ไม่สามารถใช้กับตัวแปรประเภท float และ double ได้
- นอกจากนี้ +, - เป็น unary operator ได้ด้วย
- + และ - ที่เป็น binary operator มี precedence เท่ากัน ซึ่งมี precedence ต่ำกว่า *, / และ % ซึ่งต่ำกว่า unary + และ -
- หาก operator มี precedence เท่ากัน เราจะพิจารณาลำดับการทำงานของ operator จากซ้ายไปขวา

Operators	Associativity
() [] -> .	left to right
! ~ ++ -- + - * (*type*) sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
&	left to right
^	left to right
|	left to right
&& and	left to right
|| or	left to right
?:	right to left
= += -= *= /= %= &= ^= |= <<= >>=	right to left
,	left to right

6. Relational Operators และ Logical Operators

- Relational Operators
  - >
  - >=
  - <
  - <=
  - ==
  - !=

- Logical Operators
  - &&
  - ||
  - !
  - Evaluation stops as soon as the truth or falsehood is known.

Exactly Which Years Are Leap Years?

- We add a Leap Day on February 29, almost every four years. The leap day is an extra, or intercalary, day and we add it to the shortest month of the year, February.
- In the Gregorian calendar three criteria must be taken into account to identify leap years:
  - The year can be evenly divided by 4;
  - If the year can be evenly divided by 100, it is NOT a leap year, unless;

    - The year is also evenly divisible by 400. Then it is a leap year.
- This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years.

In [6]:
```c
// ลองเขียนโค้ดทดสอบ leap year
#include <stdio.h>

int main()
{  int years[8] = {2000, 2400, 1800, 1900, 2100, 2200, 2300, 2500};
   int i;

   for (i=0; i < 8; i++) {
      int year = years[i];

      if (1 == 0)   // กำหนดเงื่อนไขให้ถูกต้อง
         printf("%d is a leap year\n", year);
      else
         printf("%d is not a leap year\n", year);
   }
}
```

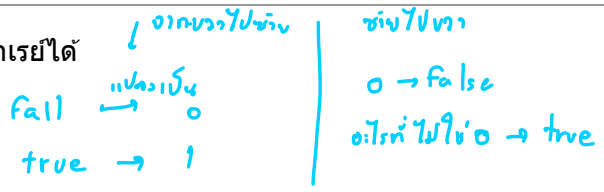
```
2000 is not a leap year
2400 is not a leap year
1800 is not a leap year
1900 is not a leap year
2100 is not a leap year
2200 is not a leap year
2300 is not a leap year
2500 is not a leap year
```

7. Type Conversion

- เมื่อนำค่าของข้อมูลประเภทต่างๆ มาถูกใช้งานผ่าน operator ร่วมกัน คอมไพเลอร์จะมีการแปลงค่าข้อมูลให้เหมาะสม โดยจะแปลงจากประเภทข้อมูลที่ "narrow" ไปยังประเภทข้อมูลที่ "wider" เพื่อไม่ให้สูญเสียข้อมูล
    - ต.ย.

```
i + f จะถูกแปลงให้เป็น f + f
```

- float ไม่สามารถใช้เป็น subscript ในอาเรย์ได้



- char <---> integer
- logic <---> number

- Explicit Type Conversion ทำได้โดยใช้ (type name) expression

- การทดสอบ (c >= '0' && c <= '9') สามารถใช้ฟังก์ชัน isdigit(c) ทดสอบแทนได้ ซึ่งฟังก์ชันนี้ถูกกำหนด function prototype ไว้ใน <ctype.h>

- นอกจากนี้ เรามักจะเห็นโค้ดที่ถูกเขียนในรูปของ

```
if (!valid)
```

แทนการเขียน

```
if (valid == 0)
```

8. Increment and Decrement Operators

- n++
- ++n
- n--
- --n
- (i+j)++ is illegal
- ต.ย.

```
void squeeze(char s[], int c)
{  int i, j;

   for (i = j = 0; s[i] != '\0'; i++)
      if (s[i] != c)
         s[j++] = s[i];
   s[j] = '\0';
}
```

In [7]:

```
#include <stdio.h>

void squeeze(char s[], int c)
{  int i, j;

   for (i = j = 0; s[i] != '\0'; i++)
      if (s[i] != c)
         s[j++] = s[i];
   s[j] = '\0';
}

int main()
{  char my_name[] = "My name is John.";

   printf("%s\n", my_name);
   squeeze(my_name, 'n');
   printf("%s\n", my_name);
}
```

```
My name is John.
My ame is Joh.
```

[ค่า 0, 1]* ฝั่งซ้าย ขวา ควรเป็นชนิดเดียว

9. Bitwise Operators

- & (AND)
  - The bitwise AND operator is often used to mask off some set of bits, for example
  - n = n & 0177;
  - sets to zero all but the low-order 7 bits of n.
- | (inclusive OR)

    - The bitwise OR operator | is used to turn bits on:
    - x = x | SET_ON;
    - sets to one in x the bits that are set to one in SET_ON.
- ^ (exclusive OR)
- << (left shift)
- >> (right shift)
- ~ (one's complement (unary))
- masking x = x & ~077 --> sets the last six bits of x to zero
- ความแตกต่างระหว่าง x & y กับ x && y

---

10. Assignment Operators ใช้สำหรับกำหนดค่า [เครื่องหมาย : ตัวเดียว]

ฟังก์ชันเก็บข้อมูล [ไม่รวมพวกเป็นค่าคงที่]

- c = getchar() และ c = d = getchar() — ฟังก์ชันหรือตัวแปรเป็นค่าได้
- i = i + 2 ==> i += 2
- x = x op y ==> x op= y โดย op อยู่ใน {+, -, *, /, %, <<, >>, &, |, ^}
- x *= y + 1 ==> x = x * (y + 1)
- ลองเขียนโปรแกรมเพื่อใช้นับจำนวน bit ที่มีค่าเป็น 1 ในจำนวนเต็มที่รับค่าเข้ามา

---

11. Conditional Expression

- expr1 ? expr2 : expr3
- ลองแปลงโค้ดด้านล่างในอยู่ในรูปของ conditional expression

```
if (a > b)
  z = a;
else
  z = b;
```

- printf("I bought %d book%s\n", n, n==1 ? "" : "s");

---

12. เกริ่นนำเกี่ยวกับอาเรย์ (Array Introduction)

- ตัวแปรอาเรย์สามารถเก็บข้อมูลมากกว่า 1 จำนวนที่เป็นประเภทเดียวกันไว้ด้วยกัน
- ต.ย.

```
int a[10];
```

- a[i] อ้างอิงขึ้นข้อมูลตัวที่ i ในอาเรย์ a โดยข้อมูลตัวแรกในอาเรย์คือ a[0]
- ลองเขียนโปรแกรมเพื่อนับจำนวนตัวเลขแต่ละตัวในอาเรย์
- อาเรย์ของอักขระเรียกได้อีกอย่างว่า *สตริง*

    - The bitwise OR operator | is used to turn bits on:
    - x = x | SET_ON;
    - sets to one in x the bits that are set to one in SET_ON.
- ^ (exclusive OR)
- << (left shift)
- >> (right shift)
- ~ (one's complement (unary))
- masking x = x & ~077 --> sets the last six bits of x to zero
- ความแตกต่างระหว่าง x & y กับ x && y

10. Assignment Operators

- c = getchar() และ c = d = getchar()
- i = i + 2 ==> i += 2
- x = x op y ==> x op= y โดย op อยู่ใน {+, -, *, /, %, <<, >>, &, |, ^}
- x *= y + 1 ==> x = x * (y + 1)
- ลองเขียนโปรแกรมเพื่อใช้นับจำนวน bit ที่มีค่าเป็น 1 ในจำนวนเต็มที่รับค่าเข้ามา

11. Conditional Expression

- expr1 ? expr2 : expr3
- ลองแปลงโค้ดด้านล่างในอยู่ในรูปของ conditional expression

```
if (a > b)
  z = a;
else
  z = b;
```

- printf("I bought %d book%s\n", n, n==1 ? "" : "s");

12. เกริ่นนำเกี่ยวกับอาเรย์ (Array Introduction)

- ตัวแปรอาเรย์สามารถเก็บข้อมูลมากกว่า 1 จำนวนที่เป็นประเภทเดียวกันไว้ด้วยกัน
- ต.ย.

```
int a[10];
```

- a[i] อ้างอิงขึ้นข้อมูลตัวที่ i ในอาเรย์ a โดยข้อมูลตัวแรกในอาเรย์คือ a[0]
- ลองเขียนโปรแกรมเพื่อนับจำนวนตัวเลขแต่ละตัวในอาเรย์
- อาเรย์ของอักขระเรียกได้อีกอย่างว่า *สตริง*