

MIIA0106 #3

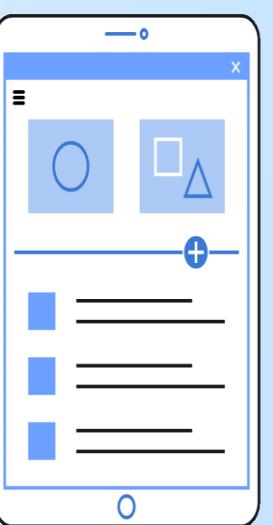


Python and C Programming Language

การโปรแกรมภาษาโปรแกรมและภาษาซี



Sutit Ongart



ตารางสอนรายวิชา MIIA0106

Section A

วัน	คบ	เวลาเรียน	ห้อง
อังคาร (TUE)	คบ 2	12.30 – 15.00 น.	D604
อังคาร (TUE)	คบ 3	15.30 – 18.00 น.	MII203

Section B

วัน	คบ	เวลาเรียน	ห้อง
เสาร์ (SAT)	คบ 4	16.00 – 18.30 น.	D503
อาทิตย์ (SUN)	คบ 4	15.30 – 17.30 น.	MII201

สอบประจำภาค วันอาทิตย์ ที่ 15 มีนาคม พ.ศ. 2569

เช้า 09:00-12:00 น.

ขาดสอบ จะได้เป็น FE

แหล่งค้นคว้าข้อมูลเพิ่มเติม



1. การแสดงข้อมูล
2. ตัวดำเนินการทางคณิตศาสตร์ (**Arithmetic Operators**)
3. คำสั่ง **Input/Output** พื้นฐานใน C และ C++ แบบละเอียด

และบางเนื้อหา สรุปมาจากการสอน CPEN1110 Introduction to Computer Programming

Code

```
#include <iostream>
using namespace std;
int main() {
    // พิมพ์ code ตรงนี้
    return 0;
}
```

Code แสดงข้อมูลออกหน้าจอ

```
#include <iostream>
using namespace std;
int main() {
    cout << "ข้อความที่ต้องการแสดง" << endl;
    return 0;
}
```

Code

```
#include <iostream>

using namespace std;

int main() {
    int a = 10, b = 3;
    cout << "ค่าของ a = " << a << endl;
    cout << "ค่าของ b = " << b << endl;
    return 0;
}
```

Code แสดงข้อมูลออกหน้าจอ

```
#include <iostream>
using namespace std;
int main() {
    cout << "ข้อความที่ต้องการแสดง" << endl;
    cout << "ข้อความ1" << "ข้อความ2" << endl;
    return 0;
}
```

ตัวดำเนินการสำหรับกำหนดค่า (Assignment Operators)

- การกำหนดค่าในภาษาซีจะใช้ตัวดำเนินการ “=” เป็นตัวดำเนินการถ่ายค่าจากตัวถูกดำเนินการทางขวาให้กับตัวถูกดำเนินการด้านซ้าย

```
LValue = RValue
```

```
int A = 5;  
int B = -10;  
A = B;
```

A = ?

- นอกจากนี้ในการกำหนดค่าให้กับตัวแปรสามารถกำหนดค่าให้กับตัวแปรรายลักษณะมาก ๆ ได้ เช่น กัน ดังตัวอย่างต่อไปนี้

```
A = B = C = 10;
```

ตัวดำเนินการสำหรับกำหนดค่า (Assignment Operators)

- ในกรณีที่ตัวแปรเป็นคนละชนิดกันตัวคอมไพล์เตอร์จะทำการเปลี่ยนชนิดข้อมูล (**Implicit Conversion Type**) ให้เหมือนกับชนิดของตัวแปรทางซ้ายมือ

```
int a;  
float b;  
b = 6.78;  
a=b;      a = ?
```

ใช้สำหรับการคำนวณทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร และหาค่า模 residue

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
+	บวก	$a + b$
-	ลบ	$a - b$
*	คูณ	$a * b$
/	หาร	a / b (หารจำนวนเต็มหรือทศนิยม)
%	หารเอาเศษ (Modulo)	$a \% b$ (ให้ได้เฉพาะจำนวนเต็ม)

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

Code

```
#include <iostream>
using namespace std;
int main() {
    int a = 10, b = 3;
    cout << "a + b = " << (a + b) << endl; // บวก
    cout << "a - b = " << (a - b) << endl; // ลบ
    cout << "a * b = " << (a * b) << endl; // คูณ
    cout << "a / b = " << (a / b) << endl; // หารจำนวนเต็ม
    cout << "a % b = " << (a % b) << endl; // หารเอาเศษ
    return 0;
}
```

ผลลัพธ์:

a + b = 13
a - b = 7
a * b = 30
a / b = 3
a % b = 1

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

Python and C Programming Language

- ลำดับความสำคัญของตัวดำเนินการทางคณิตศาสตร์
 1. ไอล์ทั้งงานจากการเล็บที่อยู่ในนิพจน์ก่อน หากไม่มีวงเล็บในนิพจน์จะทำงานตามลำดับความสำคัญของตัวดำเนินการเป็นลำดับต่อไป
 2. ตัวดำเนินการที่มีลำดับความสำคัญสูงได้แก่ * / %
 3. ตัวดำเนินการที่มีลำดับความสำคัญต่ำได้แก่ + -

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

- ตัวอย่างการคำนวณของตัวดำเนินการทางคณิตศาสตร์ในภาษาซี

```
x = 10 - 15 / 3 + 4 - 2 * ( 4 - 1 )
```

1. คำนวณโดยการไล่ตรวจสอบเลข

```
x = 10 - 15 / 3 + 4 - 2 * ( 4 - 1 )
```

```
x = 10 - 15 / 3 + 4 - 2 * 3
```

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

2. คำนวนจากตัวดำเนินการที่มีลำดับความสำคัญสูง

```
x = 10 - 15 / 3 + 4 - 2 * 3
```

```
x = 10 - 5 + 4 - 2 * 3
```

```
x = 10 - 5 + 4 - 6
```

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

3. คำนวณจากตัวดำเนินการที่มีลำดับความสำคัญต่อ

```
x = 10 - 5 + 4 - 6  
x = 5 + 4 - 6  
x = 9 - 6  
x = 3
```

ตัวดำเนินการเปรียบเทียบ (Relational Operators)

ใช้สำหรับการเปรียบเทียบค่าระหว่างตัวแปรหรือค่าคงที่

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
<code>==</code>	เท่ากับ	<code>a == b</code>
<code>!=</code>	ไม่เท่ากับ	<code>a != b</code>
<code>></code>	มากกว่า	<code>a > b</code>
<code><</code>	น้อยกว่า	<code>a < b</code>
<code>>=</code>	มากกว่าหรือเท่ากับ	<code>a >= b</code>
<code><=</code>	น้อยกว่าหรือเท่ากับ	<code>a <= b</code>

ตัวอย่างโค้ด: การใช้ตัวดำเนินการ เปรียบเทียบ

Python and C Programming Language

Code

```
#include <iostream>
using namespace std;

int main() {
    int a = 5, b = 10;
    cout << "a == b: " << (a == b) << endl; // เท่ากับ
    cout << "a != b: " << (a != b) << endl; // ไม่เท่ากับ
    cout << "a > b: " << (a > b) << endl; // มากกว่า
    cout << "a < b: " << (a < b) << endl; // น้อยกว่า
    cout << "a >= b: " << (a >= b) << endl; // มากกว่าหรือเท่ากับ
    cout << "a <= b: " << (a <= b) << endl; // น้อยกว่าหรือเท่ากับ
    return 0;
}
```

ผลลัพธ์:

```
a == b: 0
a != b: 1
a > b: 0
a < b: 1
a >= b: 0
a <= b: 1
```

ใช้สำหรับการเปรียบเทียบค่าและการดำเนินการเชิงตรรกศาสตร์ เช่น AND, OR, NOT

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
&&	AND (และ)	$(a > 0 \&\& b < 5)$
 	OR (หรือ)	$(a > 0 \mid\mid b < 5)$
!	NOT (ไม่)	$!(a > 0)$

ตัวดำเนินการทางตรรกะ (Logical Operators)

- ตัวดำเนินการทางตรรกะ AND (&&)

นิพจน์ 1	นิพจน์ 2	ผลลัพธ์ที่ได้จาก
		นิพจน์1 && นิพจน์2
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

- ตัวดำเนินการทางตรรกะ OR (||)

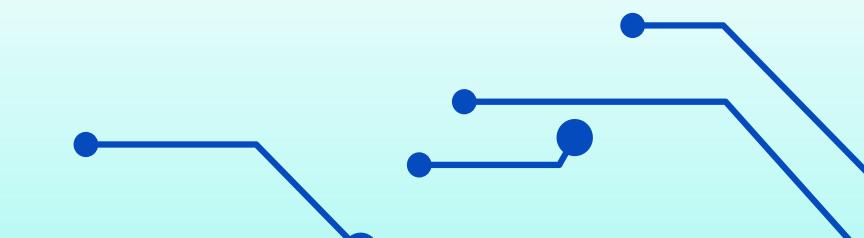
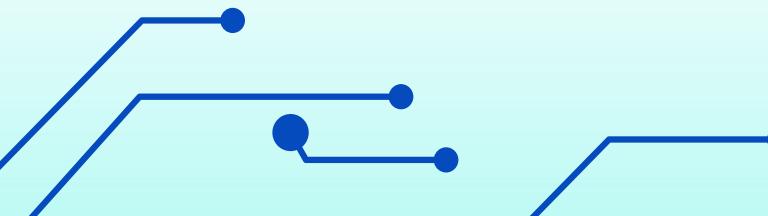
นิพจน์ 1	นิพจน์ 2	ผลลัพธ์ที่ได้จาก
		นิพจน์1 นิพจน์2
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

- ตัวดำเนินการทางตรรกะ NOT (!)

— ให้ผลลัพธ์คู่ต่างข้าม เช่น จริง เป็นเท็จ เป็นต้น

```
!(10 <= 5)
```

ผลที่ได้จากตัวดำเนินการเปรียบเทียบและตัวดำเนินการทางตรรกะที่ถูกประมวลผลจากภาษาซีจะให้ผลลัพธ์เป็น 0 หากเป็นเท็จ และให้ผลลัพธ์เป็น 1 หากเป็นจริง



Code

```
#include <iostream>
using namespace std
int main() {
    int a = 5, b = 10;
    cout << "a > 0 AND b > 0: " << (a > 0 && b > 0) << endl; // AND
    cout << "a > 0 OR b < 0: " << (a > 0 || b < 0) << endl; // OR
    cout << "NOT (a > 0): " << !(a > 0) << endl;           // NOT
    return 0;
}
```

ผลลัพธ์:

```
a > 0 AND b > 0: 1
a > 0 OR b < 0: 1
NOT (a > 0): 0
```

ตัวดำเนินการทางตรรกะ (Logical Operators)

Python and C Programming Language

- ระดับความสำคัญของตัวดำเนินการเปรียบเทียบและตรรกะ ได้จากสูงไปต่ำ

ระดับความสำคัญ	ตัวดำเนินการ
ลำดับความสำคัญสูง	!
	> >= < <=
	== !=
	&&
ลำดับความสำคัญต่ำ	

การทดสอบดำเนินการทางคณิตศาสตร์ และตรรกية

Python and C Programming Language

Code

```
#include <iostream>
using namespace std;
int main() {
    int x = 15, y = 20, z = 25;
    if ((x < y) && (z > y)) { // ใช้ทั้งเปรียบเทียบและ AND
        cout << "x น้อยกว่า y และ z มากกว่า y" << endl;
    }
    if ((x + y) > z || z < 30) { // ใช้ทั้งคณิตศาสตร์และ OR
        cout << "ผลรวมของ x กับ y มากกว่า z หรือ z น้อยกว่า 30" << endl;
    }
    return 0;
}
```

ผลลัพธ์:

X น้อยกว่า y และ z มากกว่า y

ผลรวมของ X กับ Y มากกว่า Z หรือ Z น้อยกว่า 30

ตัวดำเนินการเพิ่มค่าและลดค่า (Increment and Decrement Operators)

- Increase operator (`++`) และ Decrease operator (`-`)

เรียกว่าตัวดำเนินการเพิ่มค่าและลดค่า ใช้ในการเพิ่มและลดค่าที่ละหนึ่งค่าของตัวแปรตามลำดับตัวอย่าง เช่น

`A++` จะมีความหมายเช่นเดียวกับ `A = A+1` หรือ `A += 1`

- ลักษณะสำคัญประการหนึ่งของตัวดำเนินการนี้ได้แก่ การนำไปวางไว้ด้านหน้าหรือด้านหลังตัวแปรได้ ตัวอย่าง เช่น
 - `A++;`
 - หากว่างตัวดำเนินการเพิ่มหรือลดค่าไว้หลังตัวแปร การดำเนินการจะทำก็ต่อเมื่อการทำงานของโปรแกรมผ่านเครื่องหมายสิ้นสุดคำสั่ง `;"` จึงจะทำการเพิ่มหรือลดค่า
 - `++A;`
 - หากว่างตัวดำเนินการเพิ่มหรือลดค่าไว้หน้าตัวแปร การดำเนินการนั้นจะทำงานทันที

ตัวดำเนินการเพิ่มค่าและลดค่า (Increment and Decrement Operators)

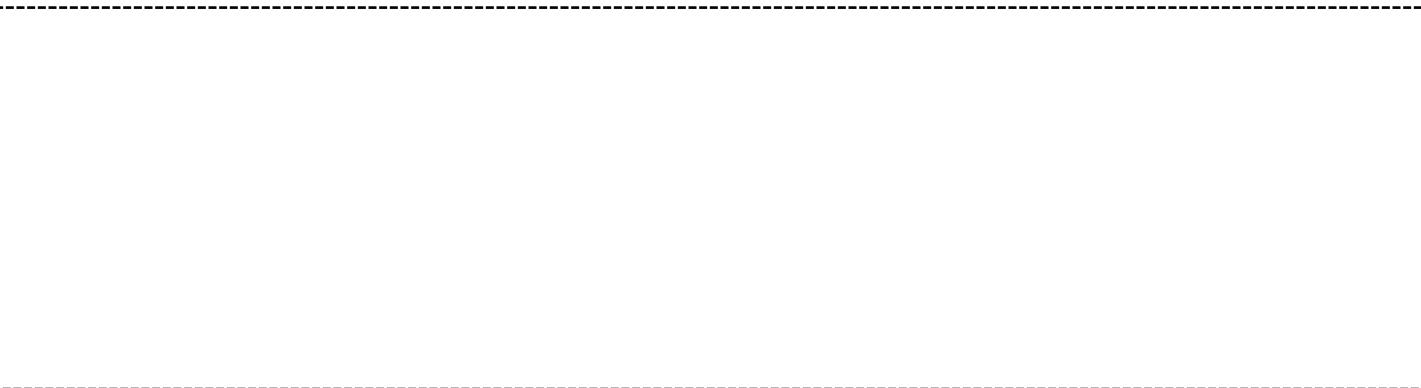
Python and C Programming Language

Code

```
#include <stdio.h>

void main(void)
{
    int a=5;
    printf("variable a = %d \n",++a);
    printf("variable a = %d \n",a++);
}
```

ผลลัพธ์:



ตัวกระทำสำหรับเลือกค่า (Conditional Operator)

- ตัวกระทำสำหรับเลือกค่า (Condition Operator) เป็นลักษณะการทำงานแบบการเลือกกระทำการอย่างใดอย่างหนึ่งโดยพิจารณาเงื่อนไขว่าจริงหรือเท็จโดยมีรูปแบบการใช้งานดังนี้

นิพจน์เปรียบเทียบ
หรือตรรกะ

?

นิพจน์หรือการทำงาน
เมื่อเป็นจริง

:

นิพจน์หรือการทำงาน
เมื่อเป็นเท็จ

;

```
#include <stdio.h>

void main(void)
{
    int a=5;
    a > 0 ? printf("Positive") :printf("Negative");
}
```

ตัวดำเนินการทางบิต (Bitwise Operators)

- มีลักษณะคล้ายกับตัวกระที่ทำการทางตรรกะเพียงแต่จะเข้าไปจัดการกับข้อมูลลิ๊กติ๊งระดับบิตซึ่งใกล้เคียงกับภาษาระดับล่างโดยมีรายละเอียดดังตารางต่อไปนี้

ตัวดำเนินการทางบิต	ชื่อ	รายละเอียด	ชนิดข้อมูลที่ใช้ได้
&	AND	Bitwise AND	จำนวนเต็ม
	OR	Bitwise OR	จำนวนเต็ม
^	XOR	Bitwise Exclusive OR ค่าบิตเหมือนกันได้ 0 ค่าบิตต่างกันได้ 1	จำนวนเต็ม
~	One's Complement	กลับค่าบิต จาก 0 เป็น 1 และ จาก 1 เป็น 0	จำนวนเต็ม
<<	Shift Left	เลื่อนบิตไปทางซ้าย	จำนวนเต็ม
>>	Shift Right	เลื่อนบิตไปทางขวา	จำนวนเต็ม

ตัวดำเนินการทางบิต (Bitwise Operators)

- สำหรับการกระทำการทางบิตได้แสดงไว้ดังตารางด้านล่าง สำหรับค่า p และ q

p	q	$p \& q$	$p q$	$p ^ q$
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

ตัวดำเนินการทางบิต (Bitwise Operators)

- ถ้ากำหนดให้ ถ้า $A = 65$; และ $B = 27$; จะได้ค่าเป็นเลขฐานสอง คือ

```
A = 0100 0001
```

```
B = 0001 1011
```

```
A&B = 0000 0001
```

```
A|B = 0101 1011
```

```
A^B = 0101 1010
```

```
~A = 1011 1110
```

```
A<<1 = 1000 0010
```

```
A>>1 = 0010 0000
```

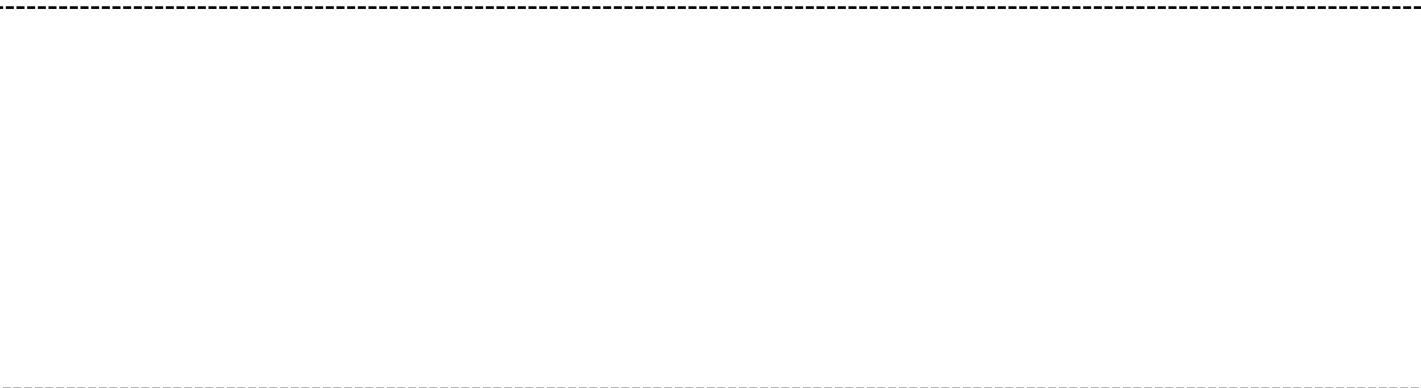
ตัวกระทำบวกขนาด (`sizeof()` Operator)

- ตัวกระทำ `sizeof()` เป็นคำสั่งสำหรับใช้บวกขนาดของพื้นที่ในหน่วยความจำของตัวแปรหรือชนิดข้อมูล
- การใช้คำสั่ง `sizeof()` ทำให้โปรแกรมที่เขียนขึ้นมีความเป็นมาตรฐานมากยิ่งขึ้นเนื่องจากบนเครื่องคอมพิวเตอร์ต่างระบบกันหรือที่ระบบปฏิบัติการต่างกันจะมีการกำหนดพื้นที่สำหรับตัวแปรแตกต่างกันออกไป
- **`sizeof(Expression)`** หรือ **`sizeof(Data Type)`**

Code

```
#include<stdio.h>
void main(void)
{
    int a = 9;
    double b = 2.3;
    printf("Data Type Memory size for keeping\n");
    printf("Char %2d Byte\n", sizeof (char));
    printf("Short %2d Byte\n", sizeof (short));
    printf("Int %2d Byte\n", sizeof (int));
    printf("Long %2d Byte\n", sizeof (long));
    printf("Float %2d Byte\n", sizeof (float));
    printf("Double %2d Byte\n", sizeof (double));
    printf("*****\n");
    printf("Variable a %2d Byte\n", sizeof (a));
    printf("Variable a+b %2d Byte\n", sizeof (a+b));
```

ผลลัพธ์:



- เป็นตัวดำเนินการลดรูปในการใช้งานตัวแปรตัวอย่าง เช่น $A = A + 1$
เราสามารถใช้ Compound Assignments เพื่อเรียกใช้งานตัว
แปร A เพียงครั้งเดียวได้เป็น $A += 1$

$a = a + 5;$	ลดรูปได้เป็น $a += 5;$
$a = a - b;$	ลดรูปได้เป็น $a -= b;$
$a = a * b;$	ลดรูปได้เป็น $a *= b;$
$a = a / 5;$	ลดรูปได้เป็น $a /= 5;$
$a = a \% 5;$	ลดรูปได้เป็น $a %= 5;$
$x = x << y;$	ลดรูปได้เป็น $x <= y;$
$x = x >> y;$	ลดรูปได้เป็น $x >= y;$
$x = x \& y;$	ลดรูปได้เป็น $x \&= y;$
$x = x y;$	ลดรูปได้เป็น $x = y;$
$x = x ^ y;$	ลดรูปได้เป็น $x ^= y;$

ตัวกระทำการจัดลำดับให้ค่า (Comma Operator)

- ตัวกระทำการนี้มักใช้ในโครงสร้างที่มีหลายนิพจน์ โดยต้องการให้กระทำพร้อม ๆ กันซึ่งลำดับการทำงานจะกระทำการซ้ายไปขวา ตัวอย่างเช่น

```
int a,b;  
a = 3 , b = 3 + a ;
```

- หมายถึง
 - a มีค่าเป็น 3
 - b มีค่าเป็น $(3 + a)$

ตัวดำเนินการ	ระดับความสำคัญ	ทิศทางการทำงาน	หมายเหตุ
<code>() [] -> . ++ --</code>	ลำดับความสำคัญสูง	ซ้ายไปขวา	<code>++ --</code> ที่วางไว้หลังตัวแปร
<code>+ - ! ~ ++ -- (type) * & sizeof ()</code>		ขวาไปซ้าย	<code>++ --</code> ที่วางไว้หน้าตัวแปร
<code>* / %</code>		ซ้ายไปขวา	
<code>+ -</code>		ซ้ายไปขวา	
<code><< >></code>		ซ้ายไปขวา	
<code>< <= > >=</code>		ซ้ายไปขวา	
<code>== !=</code>		ซ้ายไปขวา	
<code>&</code>		ซ้ายไปขวา	
<code>^</code>		ซ้ายไปขวา	
<code> </code>		ซ้ายไปขวา	
<code>&&</code>		ซ้ายไปขวา	
<code> </code>		ซ้ายไปขวา	
<code>?:</code>		ขวาไปซ้าย	
<code>= += -= *= /= %= >=> <<= &= ^= =</code>	ลำดับความสำคัญต่ำ	ขวาไปซ้าย	
<code>,</code>	ลำดับความสำคัญต่ำ	ซ้ายไปขวา	



คำสั่ง Input/Output พื้นฐานใน C และ C++

และบางเนื้อหา สรุปมาจากการสอน CPEN1110 Introduction to Computer Programming

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

ฟังก์ชัน	ภาษา	ใช้ทำอะไร	รับข้อมูลแบบใด	ต้องระบุ stream ใหม่? หมายเหตุสำคัญ
putchar(ch)	C	พิมพ์ตัวอักษร 1 ตัว	char / ASCII	ไม่
putc(ch, stream)	C	พิมพ์ตัวอักษร 1 ตัวไปยัง stream	char	ต้องระบุ เช่น stdout
puts(str)	C	พิมพ์ string และขึ้นบรรทัดใหม่	char* (string)	ไม่
printf(format, ...)	C	พิมพ์ข้อความพร้อมจัดรูปแบบ	ตัวแปรหลายประเภท	ไม่
printf(): Escape Sequence	C	อักขระพิเศษ เช่น \n, \t	ใช้ร่วมกับ printf	ไม่
printf(): แสดงค่าตัวแปร	C	พิมพ์ค่าตัวแปรด้วย %d %f %c %s	ตัวแปรทุกชนิด	ไม่
cout <<	C++	พิมพ์ข้อมูล (เหมือน printf)	ชนิดข้อมูลได้กี่ได้	ไม่

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

ฟังก์ชัน	ภาษา	ใช้ทำอะไร	รับข้อมูลแบบใด	ต้องกด Enter ไหม?	หมายเหตุ
getc(stream)	C	อ่านตัวอักษร 1 ตัวจาก stream	char	✓ ต้อง	อ่านจาก stdin หรือไฟล์
getchar()	C	อ่านตัวอักษร 1 ตัว	char	✓ ต้อง	เหมือน getc(stdin)
gets(str)	C	อ่าน string ทั้งบรรทัด	char array	✓ ต้อง	✗ ไม่ปลอดภัย (buffer overflow)
scanf(format, ...)	C	รับค่าตัวแปรแบบกำหนดชนิด	ตัวแปรทุกชนิด	✓ ต้อง	ต้องใช้ & หน้าตัวแปร
scanf(): ทำไมต้องมี &	C	ส่ง “ที่อยู่ตัวแปร” ให้ scanf	Address	✓ ต้อง	ยกเว้น array ไม่ต้องใส่ &
getch()	C (conio.h)	อ่านตัวอักษร 1 ตัว	char	✗ ไม่ต้อง	ไม่แสดงผลบนหน้าจอ
getche()	C (conio.h)	อ่านตัวอักษร 1 ตัว	char	✗ ไม่ต้อง	แสดงอักษรที่กด
cin >>	C++	รับค่าแบบ type-safe	ทุกชนิดข้อมูล	✓ ต้อง	ใช้ง่าย ไม่ต้องใช้ &

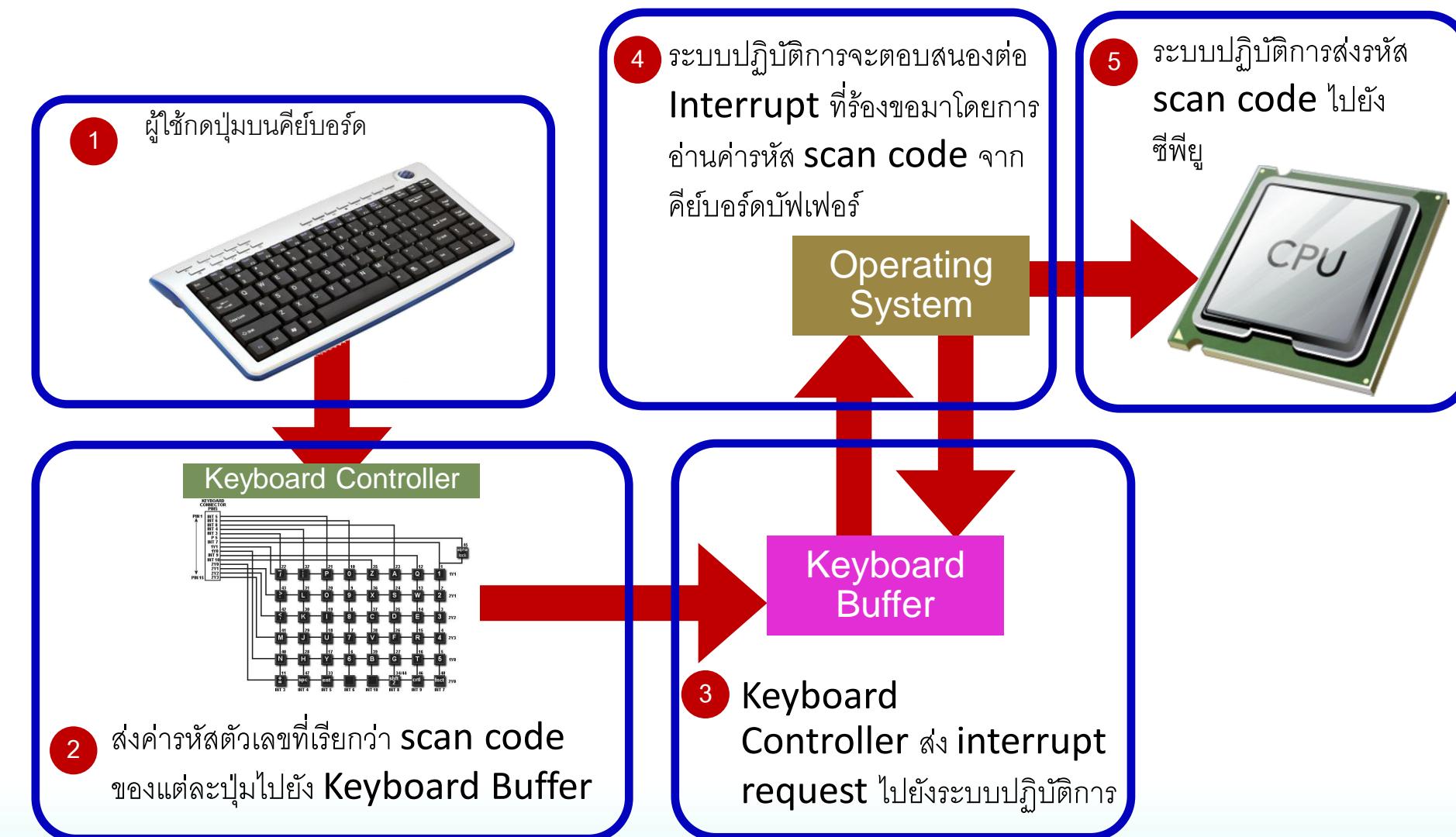
ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

Buffer

- บัฟเฟอร์ (Buffer) เป็นหน่วยความจำที่ใช้สำหรับพักข้อมูลเป็นการชั่วคราวในขณะที่มีการส่งผ่านข้อมูลไม่ว่าจะเป็นการรับหรือส่ง
- อุปกรณ์คอมพิวเตอร์บางชนิดจะมี buffer ของตนเอง เช่น เมื่อเราสั่งพิมพ์งานไปยังเครื่องพิมพ์ ข้อมูลที่ต้องการพิมพ์จะถูกส่งไปยังซีพียูเพื่อรอพิมพ์ก่อน หากข้อมูลที่ต้องการพิมพ์มีจำนวนหลายหน้าแล้วหน่วยความจำไม่พอ ระบบปฏิบัติการจะสร้างบัฟเฟอร์ขึ้นมาบนหน่วยความจำเพื่อเก็บข้อมูลที่จะพิมพ์ส่วนที่เหลือต่อไป
- การทำงานที่เกี่ยวข้องกับบัฟเฟอร์จะถูกจัดการโดยไดรฟ์เวอร์ (Device Driver) ซึ่งถูกติดตั้งไว้ในระบบปฏิบัติการ

Buffer



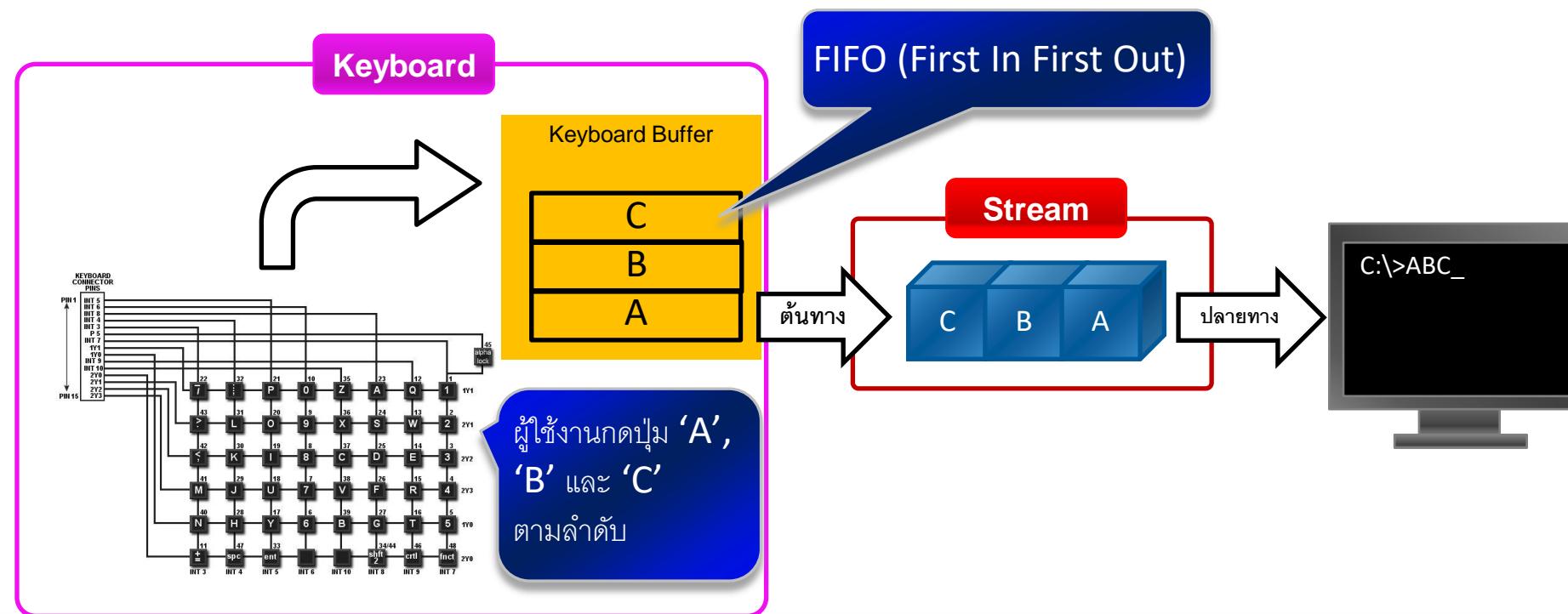
ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

สตรีม (Stream)

- สตรีมหมายถึงกระแสข้อมูลที่ไหลจากต้นทางไปยังปลายทาง เมื่อонกับสายนำ
 - กระแสข้อมูลนี้จะมีการไหลของข้อมูลอย่างเป็นลำดับต่อเนื่องกันไป
 - ลักษณะของข้อมูลดังกล่าวอาจจะเป็น ข้อมูลตัวอักษร ข้อมูลตัวเลข หรือค่าไบนาเรียกได้
 - ในสตรีมหนึ่ง ๆ จะมีตัวบอกลำดับของข้อมูลและประเภทของข้อมูลว่าเป็นข้อมูลที่เป็นการให้เข้าไปในสตรีมทั่วไปหรือเป็นข้อมูลที่บอกว่าข้อมูลนี้เป็นจุดสิ้นสุดของสตรีมแล้ว
- การเก็บข้อมูลในสตรีมจะมีลักษณะเป็น **FIFO (First In First Out)** ซึ่งหมายถึงข้อมูลที่เข้ามาก่อนจะถูกส่งออกไปยังปลายทางก่อน

สตรีม (Stream)



ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

สตรีม (Stream)

- ด้วยหลักการนี้ทำให้การทำงานของสตรีมไม่ขึ้นกับฮาร์ดแวร์ที่นำมาใช้ ซึ่งแหล่งข้อมูลต้นทางและปลายทางของสตรีมที่สามารถเลือกใช้อาจจะเป็นไฟล์ฮาร์ดไดร์ฟ ซีดี ดีวีดี การ์ดแลน
- ในการทำงานของสตรีมอาจมีการใช้งานบัฟเฟอร์เข้ามาเกี่ยวข้องด้วยตัวอย่างเช่นการติดต่อกับคีย์บอร์ด สตรีมจะทำการดึงเอกสารข้อมูลที่ผู้ใช้กดปุ่มต่างๆ ผ่านทางบัฟเฟอร์ของคีย์บอร์ดแล้วส่งข้อมูลไปยังปลายทางต่อไป
- การใช้งานสตรีมทำให้โปรแกรมเมอร์ไม่จำเป็นต้องรู้ว่าเป็นหลังการทำงานของฮาร์ดแวร์ที่จัดการกับข้อมูลนั้น มีเป็นหลังการทำงานอย่างไร
 - หน้าที่ของโปรแกรมเมอร์มีเพียงแค่จับข้อมูลใส่เข้าไปในสตรีมโดย **ระบุถึงต้นทางและปลายทางเท่านั้น** หลังจากนั้นสตรีมจะทำหน้าที่ในการอ่าน เขียน และส่งข้อมูลระหว่างกันเองโดยอัตโนมัติ

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

การแสดงผลตัวอักขรของคอมพิวเตอร์

DEC	HEX	Char	Detail	DEC	HEX	Char	Detail	DEC	HEX	Char	Detail	DEC	HEX	Char	Detail
0	00		Null character	32	20		Space	64	40	@		96	60	`	
1	01		Start of Header	33	21	!		65	41	A		97	61	a	
2	02		Start of Text	34	22			66	42	B		98	62	b	
3	03		End of Text	35	23	#		67	43	C		99	63	c	
4	04		End of Trans.	36	24	\$		68	44	D		100	64	d	
5	05		Enquiry	37	25	%		69	45	E		101	65	e	
6	06		Acknowledgement	38	26	&		70	46	F		102	66	f	
7	07		Bell	39	27	'		71	47	G		103	67	g	
8	08		Backspace	40	28	(72	48	H		104	68	h	
9	09		Horizontal Tab	41	29)		73	49	I		105	69	i	
10	0A		Line feed	42	2A	*		74	4A	J		106	6A	j	
11	0B		Vertical Tab	43	2B	+		75	4B	K		107	6B	k	
12	0C		Form feed	44	2C	,		76	4C	L		108	6C	l	
13	0D		Carriage return	45	2D	-		77	4D	M		109	6D	m	
14	0E		Shift Out	46	2E	.		78	4E	N		110	6E	n	
15	0F		Shift In	47	2F	/		79	4F	O		111	6F	o	
16	10		Data link escape	48	30	0		80	50	P		112	70	p	
17	11		Device control 1	49	31	1		81	51	Q		113	71	q	
18	12		Device control 2	50	32	2		82	52	R		114	72	r	
19	13		Device control 3	51	33	3		83	53	S		115	73	s	
20	14		Device control 4	52	34	4		84	54	T		116	74	t	
21	15		Negative acknowl.	53	35	5		85	55	U		117	75	u	
22	16		Synchronous idle	54	36	6		86	56	V		118	76	v	
23	17		End of trans. Block	55	37	7		87	57	W		119	77	w	
24	18		Cancel	56	38	8		88	58	X		120	78	x	
25	19		End of medium	57	39	9		89	59	Y		121	79	y	
26	1A		Substitute	58	3A	:		90	5A	Z		122	7A	z	
27	1B		Escape	59	3B	;		91	5B	[123	7B	{	
28	1C		File separator	60	3C	<		92	5C	\		124	7C		
29	1D		Group separator	61	3D	=		93	5D]		125	7D	}	
30	1E		Unit separator	62	3E	>		94	5E	^		126	7E	~	
31	1F			63	3F	?		95	5F	_		127	7F	• Delete	

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

การแสดงผลตัวอักษรของคอมพิวเตอร์

- ต่อมาในปี 1986 ได้ถูกปรับปรุงให้มีขนาดเป็น 8 บิต (1 ไบต์) หรือ 256 ตัวอักษรเพื่อให้สามารถแสดงตัวอักษรของแต่ละประเทศได้ ซึ่งตัวอักษร 128 ตัวแรกจะยังคงเหมือนเดิมและเรียกรหัสแอสกีใหม่นี้ว่า รหัสแอสกีแบบขยาย

ตารางสรุปฟังก์ชัน I/O –

ภาษา C และ C++

การแสดงผลตัวอักษรของคอมพิวเตอร์

DEC	HEX	Char	Detail												
128	80	€		160	A0			192	C0	ກ		224	E0	ເ	
129	81			161	A1	ກ		193	C1	ນ		225	E1	ແ	
130	82			162	A2	ຂ		194	C2	ຢ		226	E2	ໂ	
131	83			163	A3	້		195	C3	ຮ		227	E3	ໃ	
132	84			164	A4	ຄ		196	C4	ຖ		228	E4	ໄ	
133	85	...		165	A5	ຄ		197	C5	ລ		229	E5	ໆ	
134	86			166	A6	້		198	C6	ກ		230	E6	໇	
135	87			167	A7	ງ		199	C7	ວ		231	E7	້	
136	88			168	A8	ຈ		200	C8	ສ		232	E8	໊	
137	89			169	A9	່		201	C9	ໝ		233	E9	້	
138	8A			170	AA	ໜ		202	CA	ສ		234	EA	່	
139	8B			171	AB	ໜ		203	CB	ໜ		235	EB	໊	
140	8C			172	AC	ໝ		204	CC	ຟ		236	EC	້	
141	8D			173	AD	ໝ		205	CD	ອ		237	ED	໊	
142	8E			174	AE	ໝ		206	CE	ໝ		238	EE	່	
143	8F			175	AF	ໝ		207	CF	ໍາ		239	EF	ໝ	
144	90			176	B0	ໝ		208	D0	ະ		240	F0	໠	
145	91	'		177	B1	໗		209	D1	ໜ		241	F1	໑	
146	92	'		178	B2	໘		210	D2	ໜ		242	F2	໒	
147	93	"		179	B3	໘		211	D3	ໜ		243	F3	໓	
148	94	"		180	B4	໔		212	D4	ໜ		244	F4	໔	
149	95	•		181	B5	໔		213	D5	ໜ		245	F5	໕	
150	96	-		182	B6	ໜ		214	D6	ໜ		246	F6	໖	
151	97	—		183	B7	ໜ		215	D7	ໜ		247	F7	໖	
152	98			184	B8	ໜ		216	D8	ໜ		248	F8	໖	
153	99			185	B9	ໜ		217	D9	ໜ		249	F9	໖	
154	9A			186	BA	ໜ		218	DA	ໜ		250	FA	໌	
155	9B			187	BB	ໜ		219	DB	ໜ		251	FB	໌	
156	9C			188	BC	ໜ		220	DC	ໜ		252	FC	ໜ	
157	9D			189	BD	ໜ		221	DD	ໜ		253	FD	ໜ	
158	9E			190	BE	ໜ		222	DE	ໜ		254	FE	ໜ	
159	9F			191	BF	ໜ		223	DF	ໜ		255	FF	ໜ	

putchar()

- เป็นคำสั่งที่ใช้สำหรับแสดงตัวอักษรจำนวนหนึ่งตัวออกทางสตรีมเอาต์พุต มาตรฐาน ซึ่งโดยปกติแล้วจะหมายถึงจอภาพ

```
int putchar(int ch);
```

```
#include <stdio.h>

void main(void)
{
    putchar(65);
}
```

- ค่าตัวเลขในคำสั่ง putchar() สามารถใส่เป็นเลขฐาน 16 ได้ เช่น กัน ตัวอย่าง เช่น putchar(0x41)

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

putc()

- เป็นคำสั่งที่ใช้แสดงตัวอักษรหนึ่งตัวโดยสามารถระบุ스트림ปลายทางที่ต้องการได้ อาจจะเป็นได้ทั้ง **stdin, stdout และ stderr** ก็ได้

```
int putc(int ch,FILE *stream);
```

```
#include <stdio.h>

void main(void)
{
    putc(0x41,stdout);
}
```

คำสั่งนี้สามารถระบุถึง스트림ปลายทางเป็นชนิดใดก็ได้ ดังนั้นจึงหมายความว่า
นอกจากการแสดงผลแล้วเรายังสามารถส่งข้อมูลของตัวอักษรหรือตัวเลขไปยัง스트림
ปลายทาง เช่น **stdin** หรือ **stderr** ได้เช่นกัน

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

puts()

- ใช้สำหรับแสดงผลตัวอักษรมากกว่าหนึ่งตัวอักขระทางจอภาพโดยข้อความหรือกลุ่มตัวอักษรที่ต้องการส่งต้องถูกล้อมด้วยเครื่องหมาย “ ” เช่น

```
int puts(const char *s);
```

```
#include <stdio.h>

void main(void)
{
    puts("Who Am I?");
}
```

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

printf()

- คล้ายกับคำสั่ง **puts()** เพียงแต่สามารถจัดรูปแบบในการแสดงผลได้ นอกจากนี้ยังสามารถนำเอกสารตัวเปลี่ยนมาแสดงผลได้อีกด้วย

```
int printf(const char*format[,argument,...]);
```

```
#include <stdio.h>

void main(void)
{
    printf("Hello World!");
}
```

สามารถใช้งานเพื่อแสดงข้อความได้เช่นเดียวกับคำสั่ง **puts()**

printf():Escape Sequence

- การจัดรูปแบบในการแสดงผลของภาษาซีจะอาศัยอักขระพิเศษที่อยู่ในตารางแอสกิลมาใช้งานและเรียกอักขระพิเศษเหล่านี้ว่า Escape Sequence ตัวอย่างเช่นคำสั่ง printf() จะไม่ใส่คำสั่งจบและขึ้นบรรทัดใหม่ให้ หากต้องการให้ขึ้นบรรทัดใหม่ให้ใช้ Escape Sequence “\n” ดังตัวอย่างต่อไปนี้

```
#include <stdio.h>

void main(void)
{
    printf("Your character is c \n");
}
```

printf():Escape Sequence

- Escape Sequence ใช้สำหรับแสดงอักขระที่อยากในการเขียนลงบนโค้ดของโปรแกรมเมอร์ตัวอย่าง เช่น **tab** จะแทนด้วย (\t) ซึ่ง Escape Sequence ทั้งหมดจะเริ่มต้นด้วยตัว **backslash (\)** ดังตารางต่อไปนี้

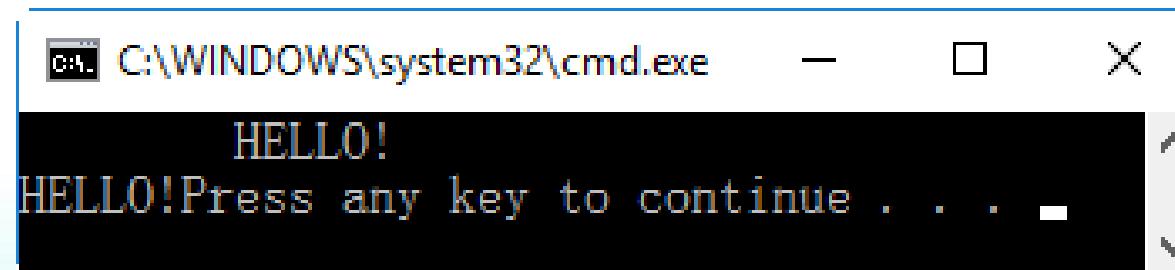
ตัวอักษร	การทำงาน
\"	Double Quotes ("")
\'	Single Quotes ('')
\\"	Backslash
\nnn	ตัวอักษร และสี ในรหัสเลขฐานแปด เช่น \041 จะหมายถึงการแสดงผลตัวอักษร !
\0xnn	เลขฐาน 16
\a	เสียงจากลำโพงของเมนบอร์ดสำหรับระบบปฏิบัติการทดสอบหรือเสียงระฆังในระบบปฏิบัติการวินโดว์
\b	รหัส Backspace (ถอยกลับ 1 space)
\f	Form-feed
\n	รหัสขึ้นบรรทัดใหม่
\r	รหัส Carriage Return
\t	รหัส Tab

printf():Escape Sequence

- สามารถใช้ Escape Sequence แสดงค่าตัวอักษรด้วยเลขฐานแปด หรือ ส్వานสิบหก ได้

```
#include <stdio.h>

void main(void)
{
    printf("\007");
    printf("\tHELLO!\n");
    printf("\110\105\114\114\117\041");
}
```



printf(): การแสดงค่าตัวแปร ในข้อความ

- ในกรณีที่ต้องการแสดงข้อความผสานกับค่าของตัวแปรจะใช้สัญลักษณ์ "%" ตามด้วยตัวอักษรย่อระบุประเภทข้อมูล เช่น **d** หมายถึง **decimal**, **c** หมายถึง **character** แทรกไว้ในประโยคและให้คันตัวแปรที่จะนำค่ามาแสดงผลด้วยเครื่องหมาย "," ดังตัวอย่างด้านล่าง

```
#include <stdio.h>

void main(void)
{
    int value = 5;
    printf("Value = %d \n", value);
}
```

printf(): การแสดงค่าตัวแปรในข้อความ

- การแสดงค่าของตัวแปรนั้นสามารถแสดงค่ากี่ตัวก็ได้แต่ให้ค้นระหว่างตัวแปรแต่ละตัวในคำสั่ง `printf()` ด้วยเครื่องหมาย “,” นอกจากนี้จำนวนของ % ในประโยคและจำนวนของตัวแปรที่ตามหลังมาต้องมีจำนวนที่เท่ากันไม่เช่นนั้นอาจทำให้เกิดการแสดงผลที่ผิดพลาดได้

printf(): การแสดงค่าตัวแปรในข้อความ

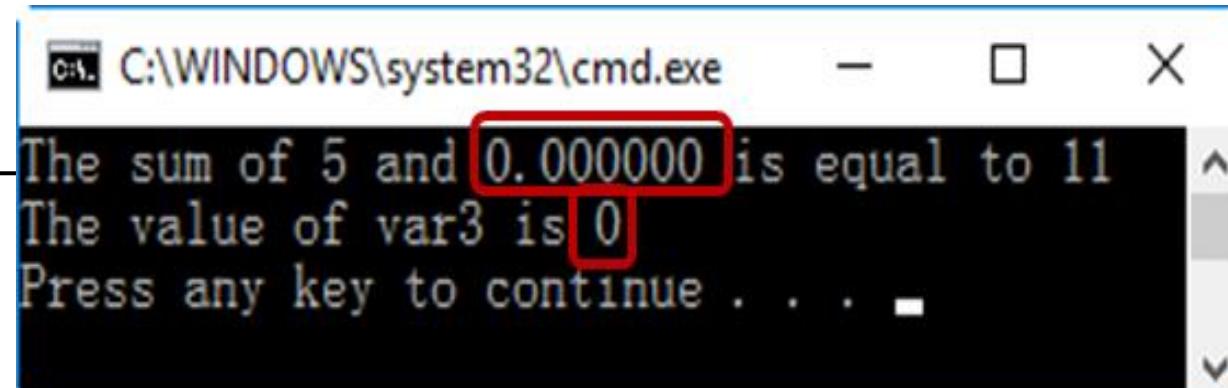
รูปแบบ	การแสดงผล	รูปแบบ	การแสดงผล
%c	ตัวอักษร (char)	%x	เลขจำนวนเต็มฐานสิบหก หากต้องการแสดงเลขฐานสิบหกแบบสี่หลักโดยมีเลขอ 0 นำหน้าให้ใช้เป็น %04x
%d	เลขจำนวนเต็มฐานสิบไม่มีเครื่องหมาย (int)	%lx	เลขจำนวนเต็มฐานสิบหกไม่มีเครื่องหมาย (unsigned long)
%u	เลขจำนวนเต็มฐานสิบไม่มีเครื่องหมาย (unsigned int)	%o	เลขจำนวนเต็มฐานแปด (int)
%ld	เลขจำนวนเต็มฐานสิบไม่มีเครื่องหมาย (long int)	%g	เลขทศนิยมปกติหรือเลขยกกำลัง
%lu	เลขจำนวนเต็มฐานสิบไม่มีเครื่องหมาย (unsigned int)	%e	เลขทศนิยมแบบยกกำลัง
%f	เลขทศนิยม (float)	%E	เลขทศนิยมแบบยกกำลัง
%lf	เลขทศนิยมอย่างละเอียด (double)	%s	สตริง (char)

printf(): การแสดงค่าตัวแปรในข้อความ

- หากประเภทของรูปแบบข้อมูลและตัวแปรไม่สอดคล้องกันการแสดงผลจะผิดพลาดดังนี้คือตัวอย่างด้านล่าง

```
#include <stdio.h>

void main(void)
{
    int var1,var2;
    float var3;
    var1=5;  var2=6;  var3=15.5;
    printf("The sum of %d and %f is equal to %d \n", var1,var2,var1+var2);
    printf("The value of var3 is %d \n",var3);
}
```



printf(): การจัดรูปแบบการพิมพ์

- คำสั่ง printf() สามารถจัดรูปแบบการแสดงผลได้ เช่น การจัดชิดซ้าย ชิดขวา เป็นต้น โดยทั่วไปจะมีรูปแบบดังนี้

```
printf(" Flag Width . Precision s ", ข้อความที่ต้องการพิมพ์ );
```

flags	คำอธิบาย
-	ใช้ในการจัดชิดซ้าย โดยปกติแล้วคำสั่ง printf() จะจัดชิดขวา
+	ใช้ในการจัดข้อความชิดขวา โดยปกติแล้วหากไม่กำหนดสัญลักษณ์ใด ๆ ไว้จะหมายถึงการใช้ค่า flags นี้โดยอัตโนมัติ
width	คำอธิบาย
เลขจำนวนเต็ม	ตัวเลขระบุถึงจำนวนตัวอักษรขั้นต่ำที่จะแสดงผลออกทางหน้าจอ หากข้อความที่ต้องการแสดงให้จำนวนตัวอักษรน้อยกว่าที่กำหนดไว้จะทำการเพิ่มช่องว่างเข้าไป
precision	คำอธิบาย
เลขทศนิยม	กำหนดจำนวนตัวอักษรที่มากที่สุดที่นำมาแสดงผลหากจำนวนตัวอักษรในข้อความมีจำนวนมากกว่าตัวเลขที่ระบุไว้ตัวอักษรที่เกินมาจะไม่ถูกนำมาแสดงผล

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

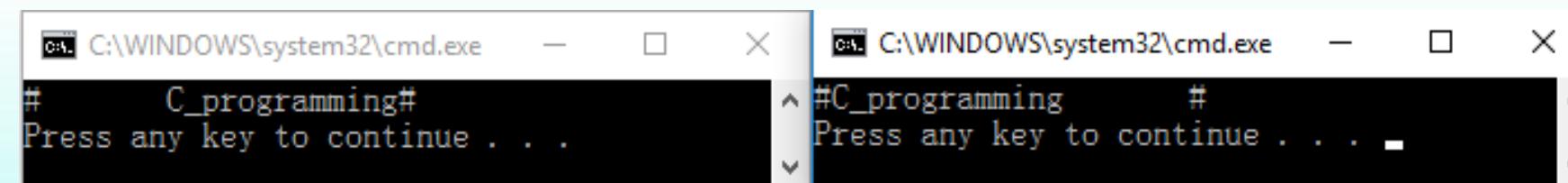
printf(): การจัดรูปแบบการพิมพ์

```
#include <stdio.h>

void main()
{
    printf("#%20s# \n", "C_programming");
}
```

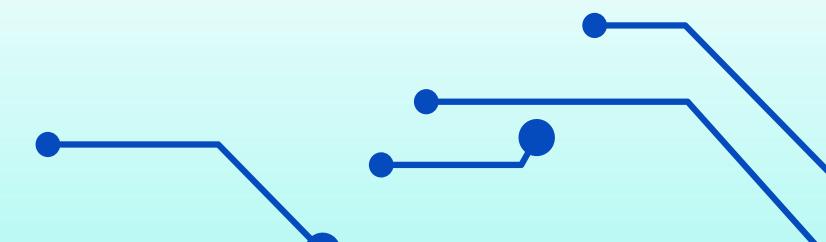
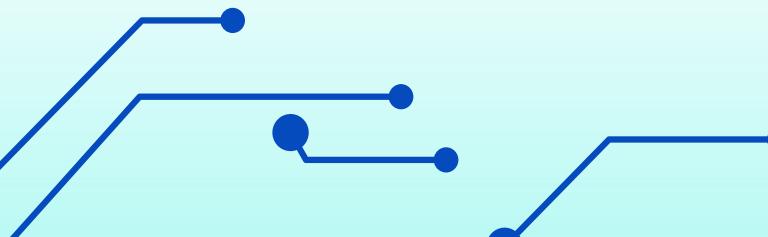
```
#include <stdio.h>

void main()
{
    printf("#%-20s# \n", "C_programming");
}
```



The image shows two separate command-line windows (cmd.exe) running on a Windows operating system. Both windows display the same command: "#%20s# \n" or "#%-20s# \n" followed by the string "C_programming". The left window's output is "#C_programming#", where the hash symbols are aligned at the start of the line. The right window's output is "#C_programming #", where the hash symbols are aligned at the start of the line, and there are five blank spaces between the first and second hash symbol. Both windows have a standard Windows title bar and a message at the bottom: "Press any key to continue . . .".

ฟังก์ชันอินพุต (Input Functions)



getc()

- ดึงค่าตัวอักษรหนึ่งตัวจากสตรีม
 - ในการใช้งานต้องระบุประเภทของสตรีมที่ต้องการใช้งานด้วย
 - ในกรณีที่สตรีมตั้นทางดึงข้อมูลมาจากคีย์บอร์ด พังก์ชันนี้จะทำงานเสร็จสิ้นก็ต่อเมื่อผู้ใช้งานโปรแกรมกดปุ่ม Enter ซึ่งถือว่าเป็นการส่งสัญญาณตัวสิ้นสุดข้อมูลในสตรีมแล้วเท่านั้น โดยมีรูปแบบการประกาศของพังก์ชันดังนี้

```
int getc(FILE *stream);
```

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

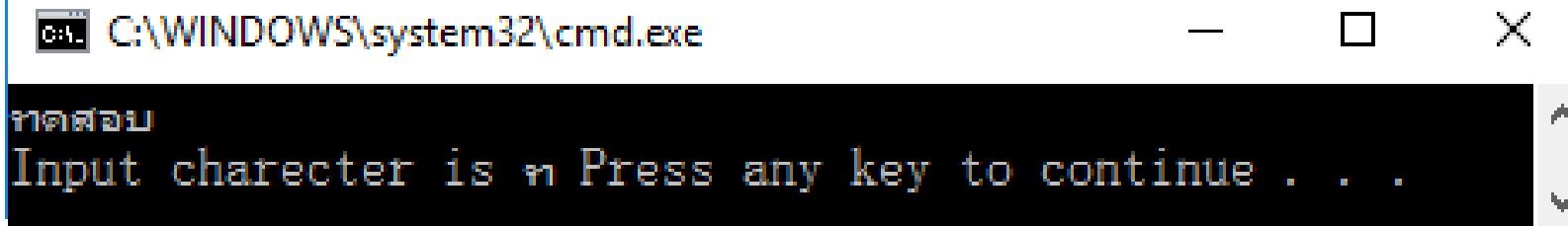
Python and C Programming Language

getc()

- ตัวอย่างการใช้งาน

```
#include <stdio.h>

void main(void)
{
    char in;
    in = getc(stdin);
    printf("Input character is %c ", in);
}
```



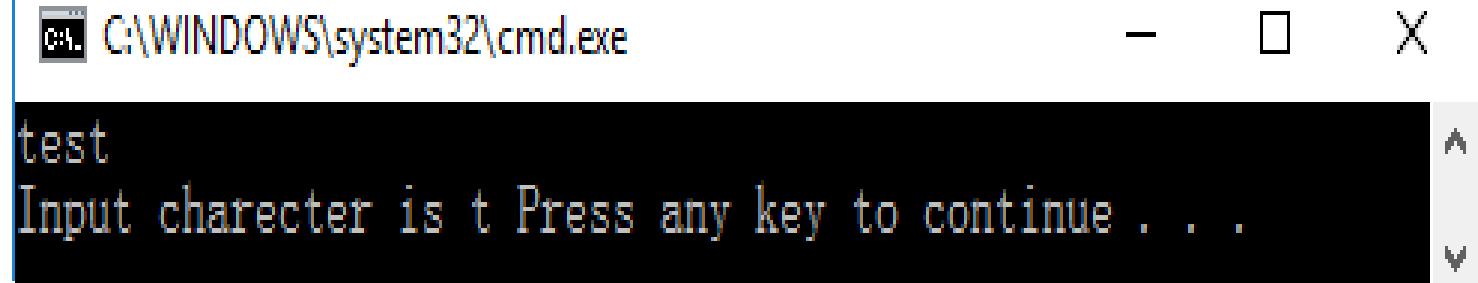
getchar()

- คล้ายคำสั่ง `getc()` แต่ไม่ต้องระบุสตรีม เนื่องจากใช้ `stdin` โดยอัตโนมัติ

```
int getchar(void);
```

```
#include <stdio.h>

void main(void)
{
    char in;
    in = getchar();
    printf("Input character is %c ", in);
}
```



gets()

- ดึงเอกสารุ่มของตัวอักษรหรือข้อความ (String) ที่มีทั้งหมดในสตรีมอินพุตมาตຽานหรือ **stdin** ซึ่งหมายความว่าคำสั่งนี้จะได้ดึงข้อมูลจากสตรีมที่ละตัวอักษรไปเรื่อย ๆ จนกว่าจะพบกับตัวสิ้นสุดสตรีมหรือ **EOF** นั่นเอง

```
char *gets(char *str);
```

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

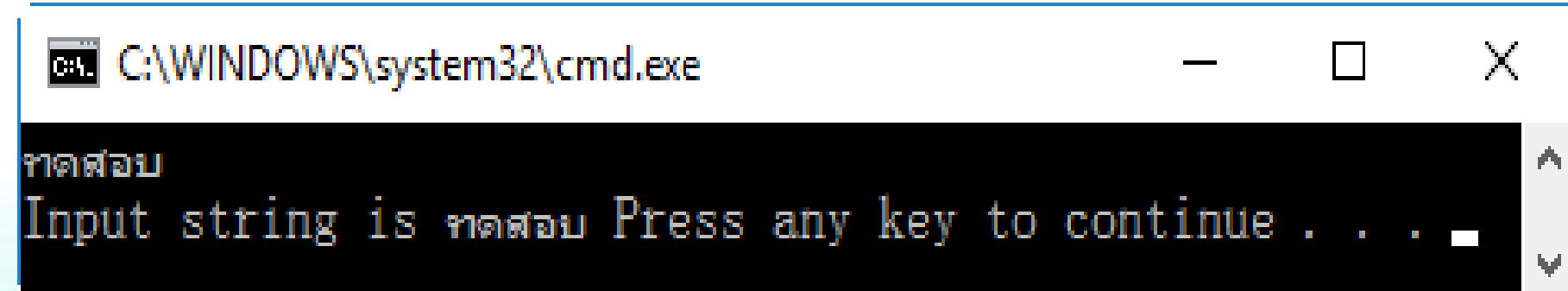
gets()

- ตัวอย่างการใช้งาน

```
#include <stdio.h>

void main(void)
{
    char in[20];
    gets(in);
    printf("Input string is %s ", in);
}
```

ตัวแปรอาร์เรย์ใช้สำหรับเก็บตัวอักษรจำนวน 20 ตัว



ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

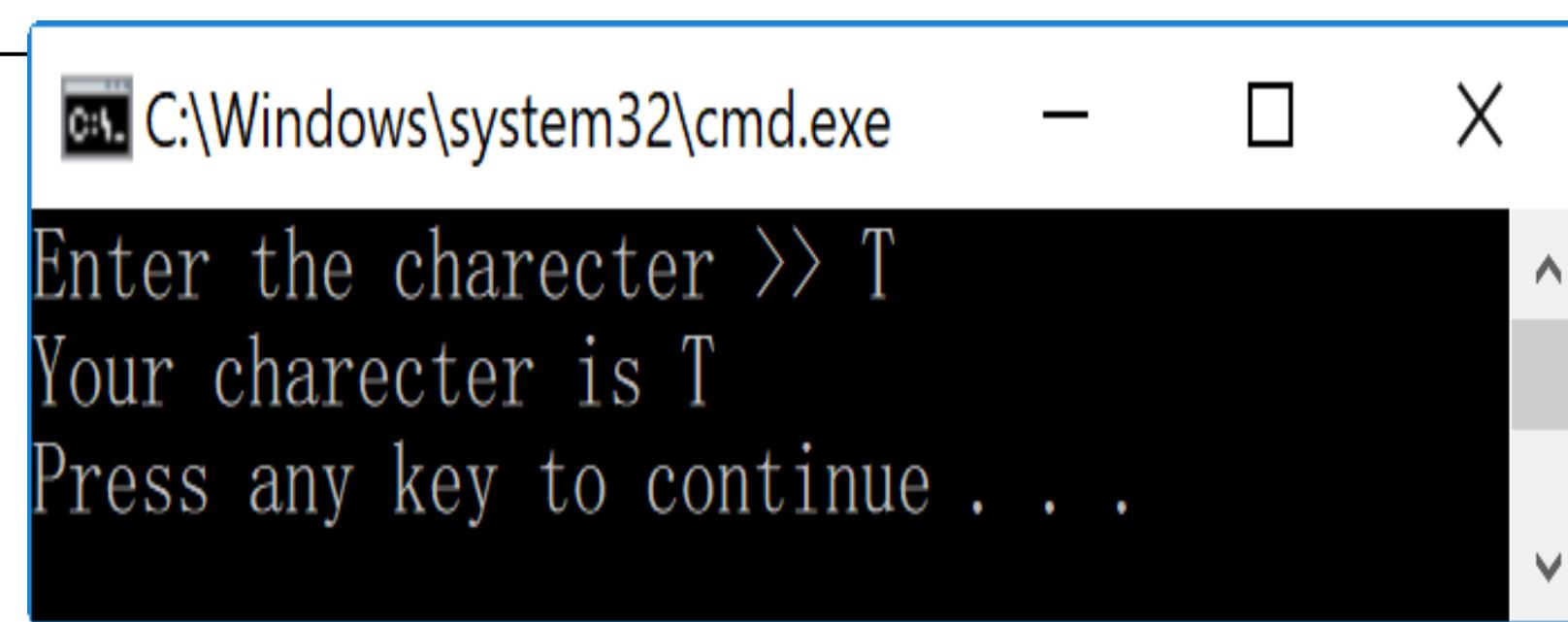
scanf()

- ดึงค่าจากสตรีมมาเก็บไว้ในตัวแปรประเภทอื่นนอกเหนือจากประเภท char ได้

```
int scanf("format1, ..., formatN", &arg1, . . . ,&argN);
```

```
#include <stdio.h>

void main(void)
{
    char character = 'm';
    printf("Enter the character >> ");
    scanf("%c", &character);
    printf("Your character is %c \n", character);
}
```



C:\Windows\system32\cmd.exe

```
Enter the character >> T
Your character is T
Press any key to continue . . .
```

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

`scanf()`: สาเหตุการใช้ &
นำหน้าตัวแปรในคำสั่ง `scanf()`

- การใช้คำสั่ง `scanf()` เพื่อถ่ายค่าให้กับตัวแปรนั้นจะทำการส่งข้อมูลที่ได้ไปยังตำแหน่งหน่วยความจำของตัวแปรโดยตรง
 - หากนำเข้าสู่ลักษณ์ & วางไว้ด้านหน้าตัวแปรจะทำให้ได้ตำแหน่งของหน่วยความจำที่เก็บตัวแปรนั้นอย่างมา
 - สำหรับตัวแปรอาร์เรย์หรือตัวซึ่งต้องออกไปเนื่องจากแค่เพียงอ้างชื่อตัวแปรเหล่านี้ก็จะทำให้ได้ตำแหน่งหน่วยความจำของมหันท์ที่โดยไม่ต้องอาศัยสัญลักษณ์ &

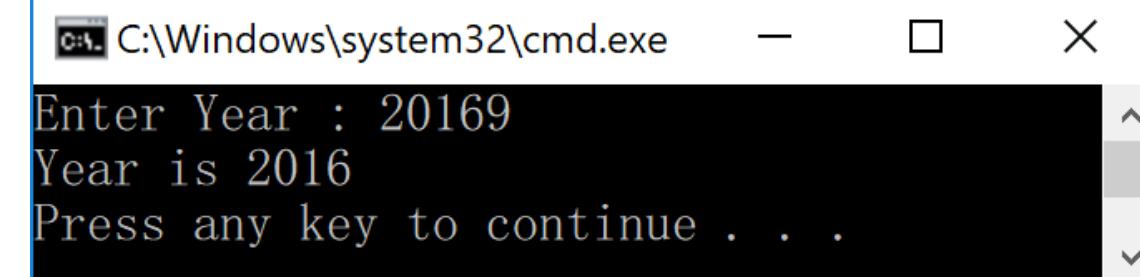
ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

scanf(): การกำหนดจำนวนตัวอักษรที่รับได้มากสุด
สำหรับตัวแปรในคำสั่ง **scanf()**

- ในบางกรณีอาจมีการกำหนดรูปแบบข้อมูลที่แน่นอน เช่น การรับข้อมูลในรูปแบบของ ปี ค.ศ. ซึ่งปกติจะรับค่าเป็นเลขสี่หลัก หากผู้ใช้ป้อนค่าเกินสี่หลัก ตัวเลขที่เกินไปจะไม่ถูกนำมาร้านสำหรับทำได้โดยระบบ ตัวเลขหน้าตัวอักษรย่อจะบุบไปพร้อมกับข้อมูล ดังตัวอย่างต่อไปนี้

```
#include <stdio.h>

void main(void)
{
    int year;
    printf("Enter Year : ");
    scanf("%4d", &year);
    printf("Year is %d \n", year);
}
```



ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

scanf():รับค่าหลายตัวแปรในครั้งเดียว

- คำสั่ง **scanf()** สามารถรับค่าให้กับตัวแปรได้หลาย ๆ ตัวได้ เช่นเดียวกับที่คำสั่ง **printf()** สามารถแสดงค่าจากตัวแปรได้หลาย ๆ ตัวในคราวเดียวดังนี้

```
#include <stdio.h>

void main(void)
{
    int date,month,year;
    printf("Please enter date in dd/mm/yy :");
    scanf("%d/%d/%d", &date, &month, &year);
    printf("Date is %2d/%2d/%2d\n", date, month, year);
}
```

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

`scanf()`: ข้อผิดพลาดที่เกิดจากการใช้ `scanf()` รับค่าตัวอักษร

- เนื่องจากในการรับอินพุตจากคีย์บอร์ดด้วยคำสั่ง `scanf()` ต้องยืนยันการส่งข้อมูลด้วยปุ่ม `Enter` เสมอทำให้ข้อมูลของปุ่ม `Enter` จะถูกส่งเข้าสตรีมด้วย
- ดังนั้นในกรณีที่มีการวนรอบค่าตัวอักษรมาหากว่าหนึ่งครั้งจะไม่นิยมใช้คำสั่ง `scanf()` แต่จะใช้ `getc()`, `getch()` หรือ `getche()` แทน

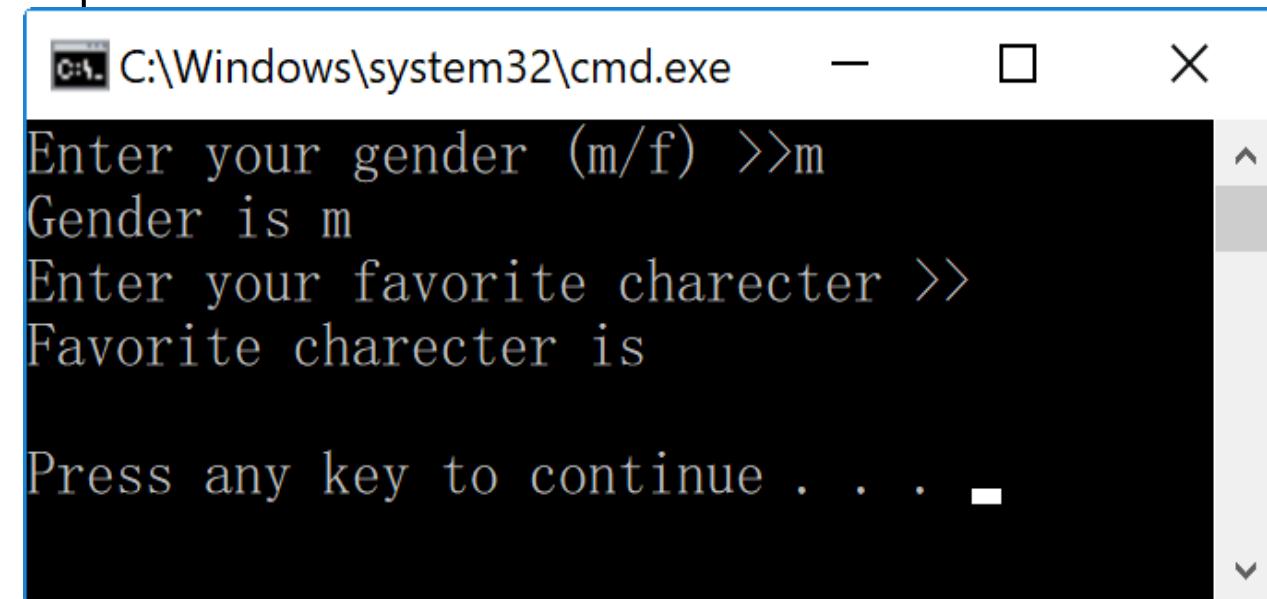
ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

scanf():ข้อผิดพลาดที่เกิดจากการใช้ scanf() รับค่าตัวอักษร

```
#include <stdio.h>

void main(void)
{
    char gender,favchar;
    printf("Enter your gender (m/f) >>");
    scanf("%c",&gender);
    printf("Gender is %c \n", gender);
    printf("Enter your favorite character >>");
    scanf("%c",&favchar);
    printf("\nFavorite character is %c \n", favchar);
}
```



```
C:\Windows\system32\cmd.exe - X
Enter your gender (m/f) >>m
Gender is m
Enter your favorite character >>
Favorite character is

Press any key to continue . . .
```

ตารางสรุปฟังก์ชัน I/O – ภาษา C และ C++

Python and C Programming Language

getch()

- เป็นการรับค่าตัวอักษรจากคีย์บอร์ดจำนวนหนึ่งตัวโดยไม่ต้องกดปุ่ม Enter เพื่อยืนยันการส่งข้อมูล
- ข้อมูลที่เกิดจากการกดแป้นคีย์บอร์ดจะไม่แสดงข้อมูลที่รับบนจอแสดงผล
- ส่วนใหญ่จะใช้เพื่อไม่ให้โปรแกรมปิดตัวโดยทันทีเมื่อทำงานเสร็จแต่ผู้ใช้งานต้องกดปุ่มใด ๆ บนคีย์บอร์ดก่อนจึงจะปิดโปรแกรมได้

```
int getch(void);
```

- ตัวอย่าง

getch()

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int ch;
    printf("Input Data\n Enter one character from keyboard : ");
    ch=getch();
    printf("\nOutput Data\n Entered character is %c",ch);
    getch();
}
```

getche()

- มีการทำงานเช่นเดียวกับ **getch()** เพียงแต่จะแสดงข้อมูลตัวอักษรที่ผู้ใช้ป้อนเข้ามาบนจอแสดงผลด้วย

```
int getche(void);
```

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int ch;
    printf("Input Data\nEnter one character from keyboard : ");
    ch=getche();
    printf("\nOutput Data\nEntered character is %c ",ch);
}
```

Code

```
#include <iostream>  
  
using namespace std;  
  
int main() {  
  
    // พิมพ์ code ตรงนี้  
  
    return 0;  
}
```

Code แสดงข้อมูลออกหน้าจอ

```
#include <iostream>  
  
using namespace std;  
  
int main() {  
  
    cout << "ข้อความที่ต้องการแสดง" << endl;  
  
    return 0;  
}
```

Code

```
#include <iostream>

using namespace std;

int main() {
    int a = 10, b = 3;
    cout << "ค่าของ a = " << a << endl;
    cout << "ค่าของ b = " << b << endl;
    return 0;
}
```

Code แสดงข้อมูลออกหน้าจอ

```
#include <iostream>
using namespace std;
int main() {
    cout << "ข้อความที่ต้องการแสดง" << endl;
    cout << "ข้อความ1" << "ข้อความ2" << endl;
    return 0;
}
```

ใช้คำสั่ง cin พร้อมตัวดำเนินการ >> เพื่อรับค่าจากผู้ใช้งาน

Python and C Programming Language

cin >> ตัวแปร;

```
#include <iostream>
using namespace std;
int main() {
    int age;
    cout << "Enter your age: ";
    cin >> age; // รับค่าจำนวนเต็มจากผู้ใช้
    cout << "Your age is " << age << endl;
    return 0;
}
```

cin >> ตัวแปร;

```
Enter your age: 25
Your age is 25
```

Code

```
#include <iostream>
using namespace std;
int main() {
    int age;
    float height;

    cout << "Enter your age and height: ";
    cin >> age >> height; // รับค่า hely ตัว
    cout << "Age: " << age << ", Height: " << height << endl;

    return 0;
}
```

ผลลัพธ์:

```
Enter your age and height: 25 5.8
Age: 25, Height: 5.8
```

ภาษา C:

- แสดงผล: printf()
- รับค่า: scanf()
- ต้องกำหนดรูปแบบข้อมูล (%d, %f, %c) และใช้ & สำหรับตัวแปร

ภาษา C++:

- แสดงผล: cout <<
- รับค่า: cin >>
- อ่านง่ายกว่า C และไม่ต้องใช้รูปแบบหรือ & สำหรับตัวแปร

การเลือกใช้งานขึ้นอยู่กับภาษาโปรแกรมที่เลือกพัฒนา โดย C++ สะดวกและกระชับกว่าสำหรับ

Input/Output!



Sutit Ongart
Sutit@mut.ac.th
sutit.ongart@gmail.com

END



www.mut.ac.th