Computer Programming

(120213600)

Week 06 : Operators and Expressions

Panadda Kongsilp

Electrical and Automation Engineering Technology

Faculty of Engineering and Technology

King Mongkut's University of Technology North Bangkok, Rayong Campus

Learning Content (Weekly Schedule)

Status	Week	Topic/Content	Deliverables
✔	1	Course Introduction	Practice #1 Coding: Hello World" in VS Code + GitHub
✔	2	Fundamentals of Computer System	Practice #2 Group activity: variety of computer language and its application.
✔	3	Algorithmic Thinking	Practice #3 Flowchart from case study problem
✔	4	C Basics I: Program structure, Data types & variables	Practice #4 Coding: main, headers, variables
✔	5	C Basics II: Console I/O Functions, Input/Output	Practice #5 Coding: printf, scanf
☝	6	Operators and Expressions	Practice #6 Coding: operators
	7	Selection statements: If & Switch Statement	Practice #7 Coding: if, switch
Mid-Term Examination			
	8	Iteration / Loop Statements: while & for loops	Practice #8 Coding: while, for loops
	9	Iteration / Loop Statements: do-while	Practice #9 Coding: do-while loops and complex loops
	10	Arrays & strings	Practice #10 Coding: Arrays & strings
	11	Functions & modular code	Practice #11 Coding: Functions
	12	Pointers & memory model	Practice #11 Coding: Pointer
	13	Mini-Project planning	Project proposal
	14	Mini-Project Sprint #1: Program structure & Coding	Prototype code compiles & passes baseline tests
	15	Mini-Project Sprint #2: Testing & refinement + Finalize	Unit-test coverage report ≥ 70 % pushed + Final code (v1.0) tagged
	16	Project demo & final project presentation	Live demo (slide deck submitted) + Assessment (Self & peer assessment)
Final Examination			

Content

- **นิยามของตัวดำเนินการ และนิพจน์ (Definition of Operator and Expression)**
- **ตัวดำเนินการในภาษา C (Operator in C)**
  - ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)
  - ตัวดำเนินการเปรียบเทียบ (Relational Operators)
  - ตัวดำเนินการทางตรรกะ (Logical Operators)
  - ตัวดำเนินการกำหนดค่า (Assignment Operators)
  - ตัวดำเนินการเพิ่ม/ลดค่า (Increment/Decrement Operators)
- **นิพจน์ในภาษา C (Expression in C)**
- **ลำดับการประเมินผลของนิพจน์ (Order of Evaluation in C Expressions)**
- **การเปลี่ยนประเภทของข้อมูล (Type Conversion/Type casting)**

นิยามของตัวดำเนินการ และนิพจน์ (Definition of Operator and Expression)

ตัวดำเนินการ (Operator) คือ สัญลักษณ์ หรือ เครื่องหมาย ที่ใช้ในการคำนวณหรือเปรียบเทียบค่าของข้อมูล เช่น +, -, *, /, ==, && ฯลฯ

<u>หน้าที่ของตัวดำเนินการ</u>

- ใช้ดำเนินการทางคณิตศาสตร์
- ใช้เปรียบเทียบค่าข้อมูล
- ใช้จัดการกับข้อมูลในรูปแบบต่าง ๆ เช่น บิต หรือ ตรรกะ

ในภาษา C จะมี operator หลายประเภท ซึ่งมีลำดับความสำคัญ (precedence) และการจัดกลุ่ม (associativity) ที่แตกต่างกัน

นิยามของตัวดำเนินการ และนิพจน์ (Definition of Operator and Expression)

นิพจน์ (Expression) คือ กลุ่มของตัวแปร ค่าคงที่ และตัวดำเนินการ ที่เขียนรวมกันเพื่อให้ได้ผลลัพธ์บางอย่าง

ตัวอย่างของนิพจน์

a + b * c	-> a + b * c เป็นนิพจน์ที่มีการคำนวณ
x > 10	-> x > 10 เป็นนิพจน์เชิงเปรียบเทียบ
y = z + 5	-> y = z + 5 เป็นนิพจน์ที่ใช้ในการกำหนดค่า

นิยามของตัวดำเนินการ และนิพจน์ (Definition of Operator and Expression)

ตัวอย่างโค้ด

```c
int a = 5, b = 2;
int result;

result = a + b * 3;
printf("Result = %d", result);
```

จากโค้ด:

- + และ * คือตัวดำเนินการ
- a + b * 3 คือนิพจน์
- result = a + b * 3 เป็นนิพจน์แบบกำหนดค่า

นิยามของตัวดำเนินการ และนิพจน์ (Definition of Operator and Expression)

ตัวอย่างโค้ด

```
int x = 10, y = 5;
int z = x - y + 2;
```

จากโค้ด:

- **ตัวดำเนินการ**
  -
- **นิพจน์**
  -

ตัวดำเนินการในภาษา C (Operator in C)

ลำดับ	ประเภท	คำอธิบาย	ตัวอย่าง
1	Arithmetic Operators	ใช้สำหรับคำนวณทางคณิตศาสตร์	+, -, *, /, %
2	Relational Operators	ใช้เปรียบเทียบค่าระหว่างตัวแปร	==, !=, <, >, <=, >=
3	Logical Operators	ใช้ตรวจสอบเงื่อนไขแบบตรรกะ	&&, || , !
4	Assignment Operators	ใช้กำหนดค่าให้ตัวแปร	=, +=, -=, *=, /=, %=
5	Increment / Decrement	เพิ่มหรือลดค่าทีละ 1	++, --
6	Bitwise Operators	ทำงานระดับ bit	&, `
7	Conditional (Ternary) Operator	คำสั่ง if แบบย่อ	condition ? expr1 : expr2
8	Sizeof Operator	ใช้ตรวจขนาดของตัวแปร/ข้อมูล	sizeof(int)
9	Comma Operator	ใช้ประเมินหลาย expression ในบรรทัดเดียว	a = (b = 3, b + 2);

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators) คือตัวดำเนินการที่ใช้ในการคำนวณทางคณิตศาสตร์พื้นฐานในภาษา C เช่น การบวก ลบ คูณ หาร และหารเอาเศษ

หมายเหตุเพิ่มเติม:

- ถ้าตัวแปรทั้งสองเป็น int การหาร / จะให้ผลลัพธ์เป็น int เช่น 5 / 2 = 2
- ตัวดำเนินการ % ใช้สำหรับหาเศษ เช่น 5 % 2 = 1

ตารางแสดงตัวดำเนินการทางคณิตศาสตร์

สัญลักษณ์	ความหมาย	ตัวอย่าง	ผลลัพธ์
+	บวก (addition)	a + b	ผลรวมของ a และ b
-	ลบ (subtraction)	a - b	ผลต่างของ a และ b
*	คูณ (multiplication)	a * b	ผลคูณของ a และ b
/	หาร (division)	a / b	ผลหารของ a กับ b (หารจำนวนเต็มหากทั้งคู่เป็น int)
%	หารเอาเศษ (modulo)	a % b	เศษจากการหาร a ด้วย b (ใช้ได้เฉพาะจำนวนเต็ม หรือ int เท่านั้น)

Practice: cp_pt_06-1

```c
#include <stdio.h>

int main() {
    int a = 10, b = 3;

    printf("assign >> a = %d and b = %d\n", a, b);

    printf("a + b = %d\n", a + b);
    printf("a - b = %d\n", a - b);
    printf("a * b = %d\n", a * b);
    printf("a / b = %d\n", a / b);
    printf("a mod b = %d\n", a % b);

    return 0;
}
```

Output

ตัวดำเนินการเปรียบเทียบ (Relational Operators)

ตัวดำเนินการเปรียบเทียบ (Relational Operators) ใช้ในการเปรียบเทียบค่าของตัวแปร 2 ค่า แล้วให้ผลลัพธ์เป็น จริง - True (1) หรือ เท็จ - False (0)

ตัวดำเนินการ	ความหมาย (ไทย)	ความหมาย (อังกฤษ)	ตัวอย่าง (a=5, b=10)	ผลลัพธ์
==	เท่ากัน	Equal to	a == b	0
!=	ไม่เท่ากัน	Not equal to	a != b	1
>	มากกว่า	Greater than	b > a	1
<	น้อยกว่า	Less than	a < b	1
>=	มากกว่าหรือเท่ากับ	Greater than or equal to	a >= b	0
<=	น้อยกว่าหรือเท่ากับ	Less than or equal to	a <= b	1

Practice: cp_pt_06-2

```c
#include <stdio.h>

int main() {
    int a = 5, b = 10;

    printf("a = %d, b = %d\n", a, b);
    printf("a == b: %d\n", a == b);  // Equal → 0
    printf("a != b: %d\n", a != b);  // Not Equal → 1
    printf("a > b : %d\n", a > b);   // Greater → 0
    printf("a < b : %d\n", a < b);   // Less → 1
    printf("a >= b: %d\n", a >= b);  // Greater or Equal → 0
    printf("a <= b: %d\n", a <= b);  // Less or Equal → 1

    return 0;
}
```

Output

การใช้ตัวดำเนินการเปรียบเทียบกับชนิดข้อมูลอื่น ๆ

1. ชนิด float (เลขทศนิยม)

แม้ว่าเราสามารถใช้ ==, !=, <, > กับ float ได้ในภาษา C แต่ต้องระวังเรื่อง ความคลาดเคลื่อนจากจุดทศนิยม (floating-point precision error)

ตัวอย่าง

```c
float a = 0.1 * 3;
float b = 0.3;

if (a == b)
    printf("Equal\n");
else
    printf("Not equal\n");
```

ผลลัพธ์:
อาจแสดงว่า "Not equal" เนื่องจากจุดทศนิยมไม่ตรงเป๊ะ

Operator	ใช้กับ float	ความแม่นยำ / ข้อควรระวัง
==, !=	ได้	มีโอกาสคลาดเคลื่อน!!!
<, >, <=, >=	ได้	ใช้ได้ทั่วไป

การใช้ตัวดำเนินการเปรียบเทียบกับชนิดข้อมูลอื่น ๆ

2. ชนิด char (ตัวอักษร)

ในภาษา C ตัวแปรชนิด char แท้จริงแล้วคือ int ที่เก็บค่ารหัส ASCII ดังนั้นจึงสามารถใช้ตัวดำเนินการเปรียบเทียบได้เหมือนกับ int

ตัวอย่าง

```c
char x = 'A';
char y = 'B';

if (x < y)
    printf("A comes before B\n");

if (x == 65)
    printf("A is ASCII 65\n");
```

Operator	ใช้กับ char	หมายเหตุ
==, !=	ได้	เปรียบเทียบตัวอักษร
<, >, <=, >=	ได้	อิงตามลำดับ ASCII
เทียบกับ int	ได้	'A' == 65

เพิ่มเติมตัวอย่างรหัส ASCII

‘A’ = 65, ‘B’ = 66, ‘a’ = 97, ‘0’ = 48

ระวัง ความต่างระหว่างตัวพิมพ์ใหญ่-เล็ก เช่น **'A' != 'a'**

ตัวดำเนินการทางตรรกะ (Logical Operators)

ตัวดำเนินการทางตรรกะใช้ในการเชื่อมเงื่อนไข มากกว่า 1 เงื่อนไข เข้าด้วยกัน โดยค่าที่ได้จะเป็นเพียง จริง True (1) หรือ เท็จ False (0)

Operator	ชื่อเรียก	ความหมาย	ตัวอย่าง
&&	AND	จริงเมื่อทั้งสองเป็นจริง	**a > 5 && b < 10**
||	OR	จริงเมื่อเงื่อนไขใดเงื่อนไขหนึ่งเป็นจริง	**a > 5 || b < 10**
!	NOT	ตรงข้ามกับค่าปัจจุบัน	**!(a == 5)**

ตารางความจริง Truth table

Logical

not	x	!x
	FALSE	TRUE
	TRUE	FALSE

and	x	y	x && y
	FALSE	FALSE	FALSE
	FALSE	TRUE	FALSE
	TRUE	FALSE	FALSE
	TRUE	TRUE	TRUE

or	x	y	x || y
	FALSE	FALSE	FALSE
	FALSE	TRUE	TRUE
	TRUE	FALSE	TRUE
	TRUE	TRUE	TRUE

C Language

!	x	!x
	zero	1
	nonzero	0

and	x	y	x && y
	zero	zero	0
	zero	nonzero	0
	nonzero	zero	0
	nonzero	nonzero	1

or	x	y	x || y
	zero	zero	0
	zero	nonzero	1
	nonzero	zero	1
	nonzero	nonzero	1

ตารางความจริง Truth table

	x	!x
!	zero	1
	nonzero	0

	x	y	x && y
and	zero	zero	0
	zero	nonzero	0
	nonzero	zero	0
	nonzero	nonzero	1

	x	y	x || y
or	zero	zero	0
	zero	nonzero	1
	nonzero	zero	1
	nonzero	nonzero	1

C Language

ในภาษา C “non-zero” หมายถึง "จริง (true)" ทั้งหมด ไม่จำกัดว่าเป็น 1 เท่านั้น

```
if (5)      → ถือว่าเป็น true
if (-3)     → true เช่นกัน
if (1000)   → ก็ true
if (0)      → false
```

ดังนั้น ค่าทุกค่า ยกเว้น 0 ไม่ว่าจะเป็นบวกหรือลบ หมายถึง "จริง (true)" ทั้งหมด

>> floating point ก็เป็น true ยกเว้น 0.0 หรือ 0.00

>> char ก็เป็น true ยกเว้น '\0’ เพราะค่าในรหัส ASCII = 0

ตัวดำเนินการกำหนดค่า (Assignment Operators)

ในภาษา C การกำหนดค่า (Assignment) คือ การนำค่าหนึ่งไปเก็บในตัวแปร โดยใช้เครื่องหมาย = เป็นตัวดำเนินการหลัก เช่น

```
int a = 10;  // assign a value
```

ภาษา C มีตัวดำเนินการกำหนดค่าหลายรูปแบบที่ใช้ร่วมกับการคำนวณโดยตรงได้ เรียกว่า **Compound Assignment Operators** ช่วยลดรูปการเขียนให้กระชับขึ้น

ประเภทของ Assignment Operators

ตัวดำเนินการ	ความหมาย	ตัวอย่าง int x;	ความหมาย
=	กำหนดค่าโดยตรง	x = 5;	x มีค่าเป็น 5
+=	บวกค่าเดิมกับค่าที่ระบุ แล้วเก็บผลลัพธ์กลับไปในตัวแปร	x += 3;	เทียบเท่ากับ x = x + 3;
-=	ลบค่าที่ระบุออกจากค่าปัจจุบัน แล้วเก็บผลลัพธ์กลับไป	x -= 2;	เทียบเท่ากับ x = x - 2;
*=	คูณค่าปัจจุบันด้วยค่าที่ระบุ แล้วเก็บผลลัพธ์กลับไป	x *= 4;	เทียบเท่ากับ x = x * 4;
/=	หารค่าปัจจุบันด้วยค่าที่ระบุ แล้วเก็บผลลัพธ์กลับไป	x /= 2;	เทียบเท่ากับ x = x / 2;
%=	หารเอาเศษ แล้วเก็บผลลัพธ์กลับไปในตัวแปร	x %= 3;	เทียบเท่ากับ x = x % 3;

Practice: cp_pt_06-3

```c
#include <stdio.h>
int main() {
    int a = 10; // Declare and initialize an integer variable a

    printf("Initial value >> a: %d\n\n", a);

    a += 5;
    printf("a += 5 result a: %d\n", a);

    a -= 3;
    printf("a -= 3 result a: %d\n", a);

    a *= 2;
    printf("a *= 2 result a: %d\n", a);

    a /= 4;
    printf("a /= 4 result a: %d\n", a);

    a %= 5;
    printf("a %%= 5 result a: %d\n\n", a);

    return 0;
}
```

Output

หมายเหตุเพิ่มเติม:

- ตัวดำเนินการกำหนดค่าทั้งหมดสามารถใช้กับ int, float, char ได้เหมือนกัน
- ระวังการหารด้วยศูนย์ (/= หรือ %=) เพราะจะทำให้โปรแกรมผิดพลาดหรือหยุดทำงานทันที

Practice: cp_pt_06-4

จงหาค่าของตัวแปร x, y ตามคำสั่งต่อไปนี้ตามลำดับ (คำนวณมือ)

บรรทัดที่	นิพจน์	x	y
1			
2			
3			
4			
5			

ตัวดำเนินการเพิ่ม/ลดค่า (Increment/Decrement Operators)

ในภาษา C ตัวดำเนินการ เพิ่มค่า (++) และ ลดค่า (--) คือ Unary Operators ที่ใช้ในการเพิ่มหรือลดค่าของตัวแปรจำนวนเต็มทีละ 1 หน่วย เหมาะกับการใช้งานในลูป การนับรอบ หรือปรับค่าแบบ step สั้น ๆ

<u>รูปแบบของตัวดำเนินการ</u>

ประเภท	ตัวดำเนินการ	ความหมาย	ผลลัพธ์หลังใช้
เพิ่มค่า	++x (prefix)	เพิ่มค่า x ก่อน แล้วจึงใช้	ใช้ค่าใหม่ทันที
เพิ่มค่า	x++ (postfix)	ใช้ค่า x ก่อน แล้วจึงเพิ่ม	ใช้ค่าก่อนเพิ่ม
ลดค่า	--x (prefix)	ลดค่า x ก่อน แล้วจึงใช้	ใช้ค่าใหม่ทันที
ลดค่า	x-- (postfix)	ใช้ค่า x ก่อน แล้วจึงลด	ใช้ค่าก่อนลด

ตัวดำเนินการเพิ่มค่า (Increment Operator)

```
int a = 5, b;
```

Prefix `++a`

เพิ่มค่าก่อน แล้วจึงใช้ค่า

```
b = ++a
1  a = a + 1
2  b = a
```

Postfix `a++`

ใช้ค่าก่อน แล้วจึงเพิ่ม

```
b = a++
1  b = a
2  a = a + 1
```

ตัวดำเนินการเพิ่มค่า (Increment Operator)

int a = 5, b;

Prefix **++a**

เพิ่มค่าก่อน แล้วจึงใช้ค่า

b = ++a

1 a = a + 1 ⇒ a = a + 1 = 5 + 1 = 6

2 b = a ⇒ b = a = 6

Final Output: a = 6, b = 6

a++ **Postfix**

ใช้ค่าก่อน แล้วจึงเพิ่ม

b = a++

1 b = a ⇒ b = a = 5

2 a = a + 1 ⇒ a = a + 1 = 6

Final Output: a = 6, b = 5

ตัวดำเนินการลดค่า (Decrement Operator)

int a = 5, b;

Prefix **--a**

ลดค่าก่อน แล้วจึงใช้ค่า

b = --a

1 a = a - 1 → a = a - 1 = 5 - 1 = 4

2 b = a → b = a = 4

Final Output: a = 4, b = 4

a-- **Postfix**

ใช้ค่าก่อน แล้วจึงลด

b = a--

1 b = a → b = a = 5

2 a = a - 1 → a = 5 - 1 = 4

Final Output: a = 4, b = 5

Practice: cp_pt_06-5

จงหาค่าของตัวแปร i, j, k ตามคำสั่งต่อไปนี้ตามลำดับ (คำนวณมือ)

บรรทัดที่	นิพจน์	i	j	k
1				
2				
3				
4				
5				
6				

นิพจน์ในภาษา C (Expression in C)

นิพจน์ (Expression) คือ การรวมกันของค่าคงที่ ตัวแปร และตัวดำเนินการ (Operators) ที่มีผลลัพธ์เป็นค่าเดียว

ตัวอย่าง

```
x + y
a * (b + c)
x = y + 3
```

องค์ประกอบของนิพจน์

- ตัวดำเนินการ (Operators) เช่น +, -, *, /, %
- ตัวแปร (Operands) เช่น a, b, x, num
- เครื่องหมายวงเล็บ (Parentheses) เพื่อควบคุมลำดับ

ประเภทของนิพจน์ในภาษา C (Expression in C)

ประเภทนิพจน์	ตัวอย่าง	ความหมาย
นิพจน์คำนวณ (Arithmetic)	a + b * c	คำนวณค่าตัวเลข
นิพจน์เปรียบเทียบ (Relational)	a > b	ให้ค่าผลลัพธ์เป็น 1 (จริง) หรือ 0 (เท็จ)
นิพจน์ลอจิก (Logical)	a > b && b < c	เชื่อมเงื่อนไขหลายตัว
นิพจน์กำหนดค่า (Assignment)	x = y + 3	กำหนดค่าให้ตัวแปร
นิพจน์แบบฟังก์ชัน	printf("Hi")	เรียกใช้ฟังก์ชัน

การเขียนนิพจน์ในรูปย่อ (Short-hand Expression)

เพื่อให้โค้ดกระชับขึ้น ภาษา C จึงมีรูปแบบ ย่อของนิพจน์การกำหนดค่า ที่ใช้ตัวดำเนินการร่วมกับ =

เช่น +=, -=, *=, /=, %=

<u>ตารางรูปแบบย่อ</u>

แบบเต็ม	แบบย่อ	ความหมาย
a = a + 1;	a += 1;	เพิ่มค่า 1 เข้ากับ a
a = a - b;	a -= b;	ลบ b ออกจาก a
a = a * 2;	a *= 2;	คูณ a ด้วย 2
a = a / b;	a /= b;	หาร a ด้วย b
a = a % 3;	a %= 3;	a เท่ากับเศษจาก a หาร 3

ลำดับการประเมินผลของนิพจน์ (Order of Evaluation in C Expressions)

เมื่อในนิพจน์ (Expression) มี ตัวดำเนินการหลายตัว (Operators) ปรากฏร่วมกัน เช่น

```
int result = a + b * c - d;
```

ภาษา C จะ ไม่ประมวลผลจากซ้ายไปขวาเสมอไป แต่จะใช้กฎ ลำดับความสำคัญ (Precedence) และ การจัดกลุ่ม (Associativity) เพื่อตัดสินว่าตัวดำเนินการใดควรถูกประมวลผลก่อน

ลำดับการประเมินผลของนิพจน์ (Order of Evaluation in C Expressions)

ลำดับ	กลุ่ม Operator	ตัวอย่าง	ลำดับการประเมิน
1	วงเล็บ	(…)	ชั้นในสุด ⟶ ชั้นนอก
2	Unary Operators	!a, -a, ++a, --a	ขวา ⟶ ซ้าย
3	Multiplicative	*, /, %	ซ้าย ⟶ ขวา
4	Additive	+, -	ซ้าย ⟶ ขวา
5	Relational	<, <=, >, >=	ซ้าย ⟶ ขวา
6	Equality	==, !=	ซ้าย ⟶ ขวา
7	Logical AND	&&	ซ้าย ⟶ ขวา
8	Logical OR	||	ซ้าย ⟶ ขวา
9	Assignment	=, +=, -=, *=, /=	ขวา ⟶ ซ้าย

1. หาก () มีหลายชั้น () ชั้นในสุดจะถูกประเมินก่อน เช่น

   int result = 2 * (3 + (4 - 1)); // เริ่มจากในสุด

2. Unary Operators >> !a, -a, ++a, --a

   หากมีหลาย operator ในบรรทัดเดียวกัน กลุ่มนี้จะเริ่มประเมินจากด้านขวาก่อน แล้วไล่กลับมาซ้าย เช่น

   int a = 0;

   int b = !++a;

ลำดับการประเมินผล:

1. ++a ⟶ เป็น prefix increment ⟶ เพิ่ม a ก่อน ⟶ a = 1
2. !1 ⟶ เป็น false ⟶ ได้ค่า 0
3. **ดังนั้น b = 0**

ลำดับการประเมินผลของนิพจน์ (Order of Evaluation in C Expressions)

ลำดับ	กลุ่ม Operator	ตัวอย่าง	ลำดับการประเมิน
1	วงเล็บ	(…)	ชั้นในสุด ⟶ ชั้นนอก
2	Unary Operators	!a, -a, ++a, --a	ขวา ⟶ ซ้าย
3	Multiplicative	*, /, %	ซ้าย ⟶ ขวา
4	Additive	+, -	ซ้าย ⟶ ขวา
5	Relational	<, <=, >, >=	ซ้าย ⟶ ขวา
6	Equality	==, !=	ซ้าย ⟶ ขวา
7	Logical AND	&&	ซ้าย ⟶ ขวา
8	Logical OR	||	ซ้าย ⟶ ขวา
9	Assignment	=, +=, -=, *=, /=	ขวา ⟶ ซ้าย

9. Assignment >> =, +=, -=, *=, /=
   จะประเมินค่าจากขวาไปซ้าย เมื่อมีหลายตัวในระดับเดียวกัน เช่น

```
int a, b, c;
a = b = c = 5;
```

<u>การประเมินค่าจะเกิดจาก ขวา ⟶ ซ้าย:</u>

1. c = 5; ⟶ c ได้ค่า 5
2. b = c; ⟶ b ได้ค่า 5
3. a = b; ⟶ a ได้ค่า 5
4. สุดท้าย a, b, c จะมีค่าเท่ากันคือ 5

เนื่องจากสิ่งที่ C มองเห็นคือ a = (b = (c = 5)) จึง ขวา ไป ซ้าย

<u>หมายเหตุสำคัญ</u>

- หลีกเลี่ยงนิพจน์ที่มี side effect ซ้อนกัน เช่น a++ + ++a
- ใช้วงเล็บ () เพื่อควบคุมลำดับการประเมิน

การเปลี่ยนประเภทของข้อมูล (Type Conversion/Type casting)

1. การเปลี่ยนประเภทของข้อมูล (Type Conversion)

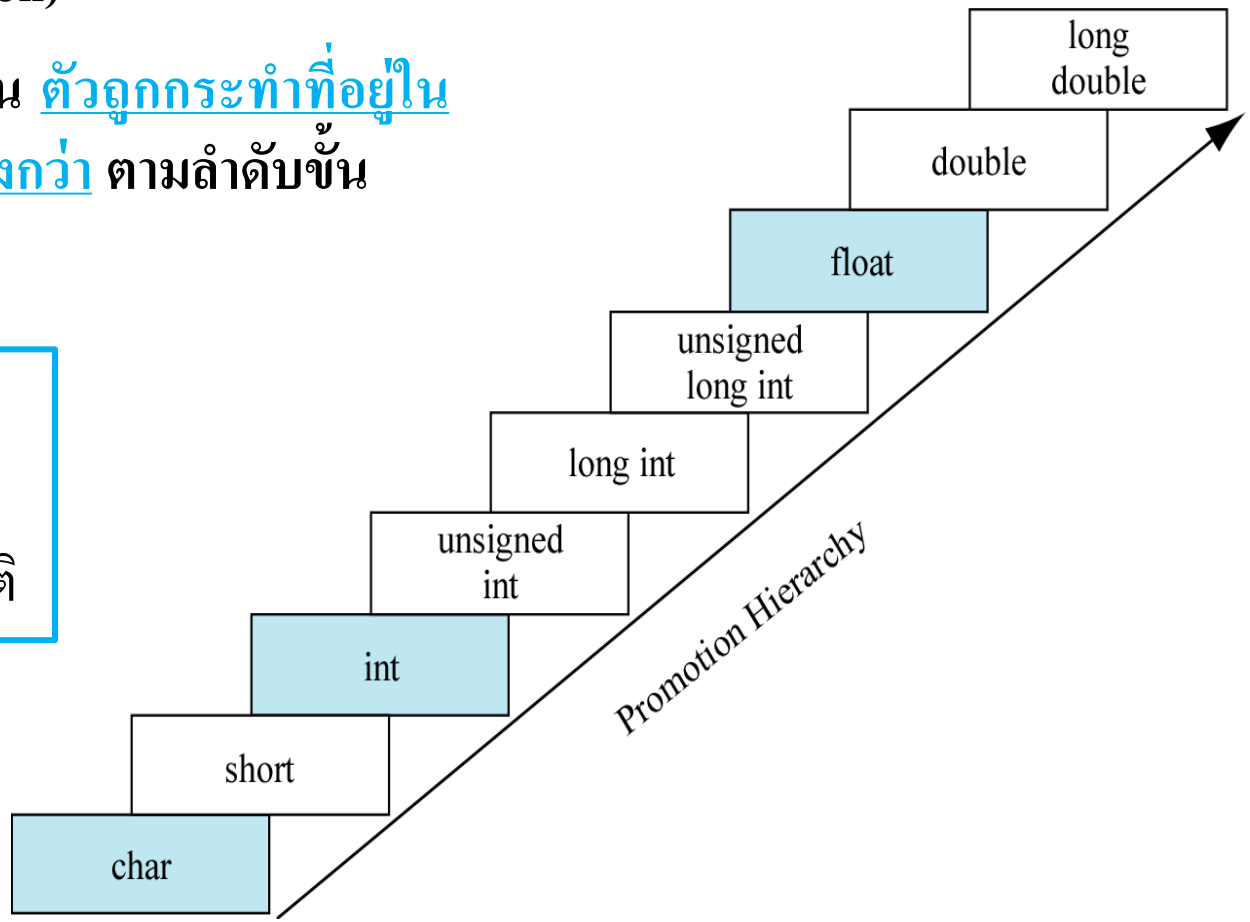
เมื่อมีการคำนวณทางคณิตศาสตร์ ในนิพจน์เดียวกัน ตัวถูกกระทำที่อยู่ในระดับต่ำกว่าจะถูกเปลี่ยนชนิดให้เหมือนกับตัวที่อยู่สูงกว่า ตามลำดับขั้น

ตัวอย่าง

```
int a = 5;
float b = 2.0;
float result = a + b;  // a ถูกแปลงเป็น float อัตโนมัติ
```

int a = 5

float b = 2.0

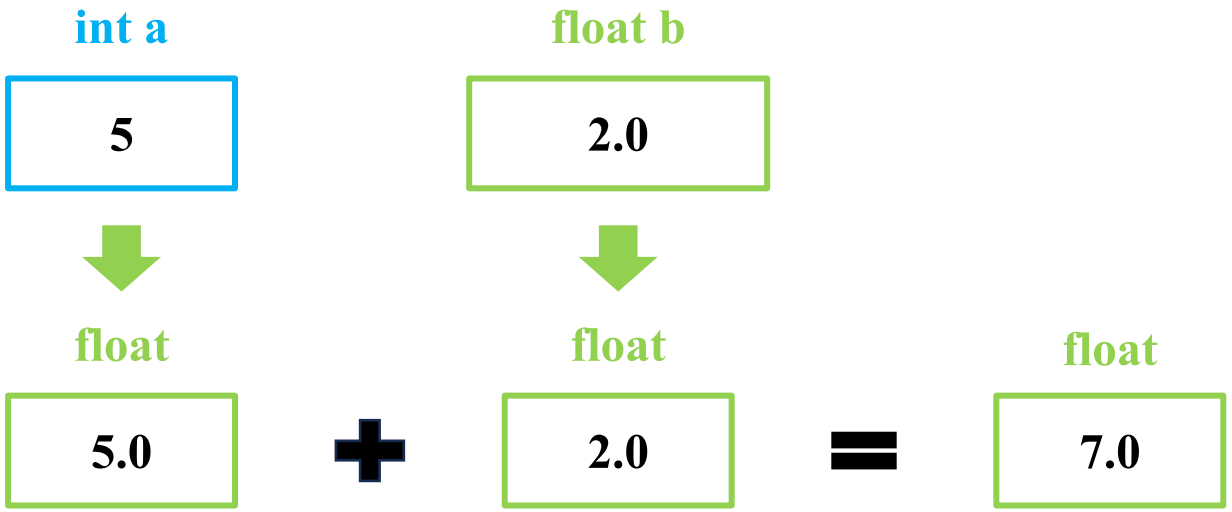


การเปลี่ยนประเภทของข้อมูล (Type Conversion/Type casting)

ตัวอย่าง

```
int a = 5;
float b = 2.0;
```

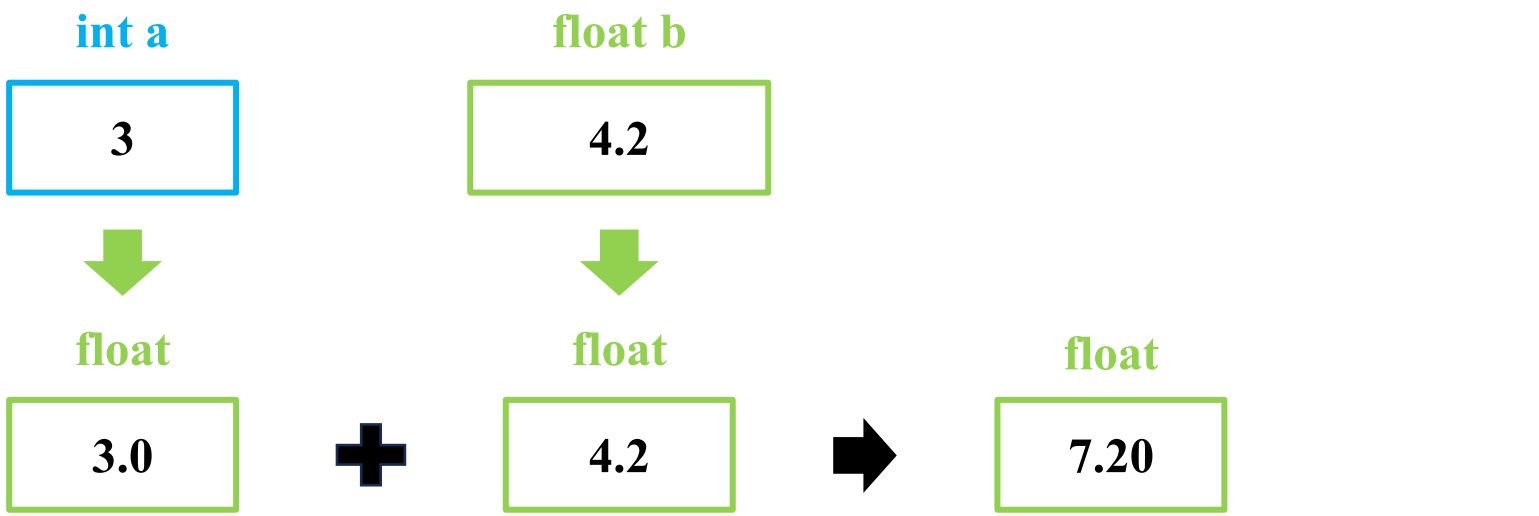
a + b =?



การเปลี่ยนประเภทของข้อมูล (Type Conversion/Type casting)

ตัวอย่าง

```
int a = 3;
float b = 4.2;
printf("%.2f\n", a + b); // Output: 7.20
```



Practice: cp_pt_06-6

ตัวอย่าง

```
int x = 10;
float y = 20.5, z;
z = x + y;
```

z =?

Practice: cp_pt_06-7

ตัวอย่าง

```
float x = 7.3, y = 3.5;
int z;
z = x + y;
```

z =?

การเปลี่ยนประเภทของข้อมูล (Type Conversion/Type casting)

2. การเปลี่ยนแบบระบุ (Explicit Type Casting)

นักเขียนโปรแกรมสามารถ บังคับแปลงชนิดข้อมูล ได้ด้วย (type) ข้างหน้า:

(Data type) expression

ตัวอย่าง

- **(float)a**
- **(float)(x+y)**
- **(int)(a/10)** มีค่าไม่เท่ากับ (int)a/10

การเปลี่ยนประเภทของข้อมูล (Type Conversion/Type casting)

2. การเปลี่ยนแบบระบุ (Explicit Type Casting)

นักเขียนโปรแกรมสามารถ บังคับแปลงชนิดข้อมูล ได้ด้วย (type) ข้างหน้า:

ตัวอย่าง

```
float pi = 3.14159;
int x = (int) pi;    // x จะมีค่า 3
```

ตัวแปร pi มีประเภทของข้อมูลเป็น int ไม่สามารถเก็บตัวเลขที่มีจุดทศนิยมได้ ดังนั้น ค่าที่คำนวณได้จะถูกตัดจุดทศยมทิ้ง เพื่อแปลงเป็นจำนวนเต็ม (ไม่มีการปัดจุด)

ใช้เมื่อ:

- ต้องการ ตัดทศนิยม
- ควบคุมชนิดข้อมูลให้นิ่ง เช่น บังคับไม่ให้ int กลายเป็น float
- แก้ปัญหาผลลัพธ์ผิดเพี้ยนจากการหาร

การเปลี่ยนประเภทของข้อมูล (Type Conversion/Type casting)

ตัวอย่างการใช้ Type Casting เพื่อควบคุมผลลัพธ์

```
int a = 5, b = 2;
float result = (float) a / b;
printf("%.2f\n", result); // Output: 2.50
```

หากไม่แปลง:

```
int a = 5, b = 2;
float result = a / b;
printf("%.2f\n", result); // Output: 2.00 (เพราะ a/b = int)
```

การเปลี่ยนประเภทของข้อมูล (Type Conversion/Type casting)

ตารางเปรียบเทียบการเปลี่ยนประเภทอัตโนมัติ

ตัวอย่างนิพจน์	ประเภทของผลลัพธ์	คำอธิบาย
3 + 2.5	double	3 ถูกแปลงเป็น double
5 / 2	int	ผลลัพธ์คือ 2 (ปัดทศ)
(float)5 / 2	float	ได้ 2.5
7 / (float)3	float	ได้ 2.333…

Practice: cp_pt_06-8

จงหาค่าของ a, b, c, d, และ e

1. float a;
2. int b;
3. int c;
4. int d;
5. float e;

Practice: cp_pt_06-9

```c
#include <stdio.h>

int main() {
    int a = 7;
    float b = 2.5;

    printf("a / b = %.2f\n", a / b);
    printf("(int)b = %d\n", (int)b);
    printf("(float)a / (int)b = %.2f\n", (float)a / (int)b);

    return 0;
}
```

Output

Coding Time

Coding_week_06

1. สร้าง Repository ใน Github โดยตั้งชื่อ compro_w06_รหัสนศ. เช่น compro_w06_4810160269
2. Clone Repository >> compro_w06_รหัสนศ. ไว้ที่ vs code
3. จากนั้นสร้างไฟล์ .c ของแต่ละข้อ โดยตั้งชื่อตามลำดับ เช่น Coding_w06-01.c
4. เขียนโค้ด รันเพื่อแสดงผลลัพธ์ อธิบายโค้ดแต่ละบรรทัดด้วยการคอมเม้น รวมถึงตอบคำถามในแต่ละข้อ โดยใช้การคอมเม้น ในตอนล่างสุดของโค้ด
5. Capture รูปภาพที่เห็นทั้งส่วนของโค้ด และผลลัพธ์ และตั้งชื่อตามลำดับ เช่น Coding_w06-01
6. Push ไฟล์ .c ทั้งหมด เข้าไปใน Repo >> compro_w06_รหัสนศ.
7. Upload ไฟล์รูปที่ Capture ของทุกข้อ เข้าไปใน Repo >> Repo >> compro_w06_รหัสนศ.
8. Copy url ของ Repo >> compro_w06_รหัสนศ. และส่งผ่าน Google Form (ประกาศใน Google Classroom)

Coding_w06-01

หัวข้อ: Arithmetic Operators

<u>โจทย์</u>: กำหนดให้ a =5, b = 17, c = 8.5, d = 4.0

1. เขียนโค้ด (พร้อมคอมเม้นอธิบายการทำงานของแต่ละบรรทัด/ส่วน) เพื่อคำนวณนิพจน์ และแสดงผลลัพธ์ ดังนี้

บรรทัดที่	ผลลัพธ์
1	แสดงค่าเริ่มต้นของตัวแปรทั้งหมด
2	ผลการคำนวณของ d + a
3	ผลการคำนวณของ a - b
4	ผลการคำนวณของ c * d
5	ผลการคำนวณของ a * c

บรรทัดที่	ผลลัพธ์
6	ผลการคำนวณของ c / d
7	ผลการคำนวณของ b / c
8	ผลการคำนวณของ a % b
9	ผลการคำนวณของ c % a
10	ผลการคำนวณของ c % d

Coding_w06-01 (ต่อ)

หัวข้อ: Arithmetic Operators

<u>โจทย์</u>: กำหนดให้ a =5, b = 17, c = 8.5, d = 4.0

2. อธิบายผลลัพธ์ที่เกิดขึ้นในแต่ละบรรทัด และอภิปรายผลจากการทดลองแยกเป็นข้อๆ (ด้วยการเขียนคอมเม้นไว้ส่วนล่างสุดของโค้ด)

หมายเหตุเพิ่มเติม:

- ในกรณีที่ run แล้ว เกิด error เนื่องจากไม่เป็นไปตามข้อกำหนดของภาษา C ให้ คอมเม้น โค้ด บรรทัดนั้นไว้ แล้วอธิบายว่าเกิดอะไร? ทำไม? อย่างไร?
- หรือหากใครมีแนวทางในการแก้ไข หรือปรับเปลี่ยนโค้ดเพื่อให้สามารถดำเนินการได้ ก็เขียนโค้ดที่แก้ไข ปรับแต่ง พร้อมอธิบายเพิ่มเติม ว่าเกิดอะไร? ทำไม? อย่างไร?

Coding_w06-02

หัวข้อ: Arithmetic Operators

1. จงหาค่าของตัวแปร i, j, k ตามคำสั่งต่อไปนี้ตามลำดับ (คำนวณมือ)

w06-02-01

บรรทัดที่	นิพจน์	i	j	k
1	int i = 1, j = 2 , k;			
2	k = i + j;			
3	i = i + (k * j);			
4	j = i / 2;			
5	k = i % 2;			
6	i = (j + k) * 3;			

w06-02-02

บรรทัดที่	นิพจน์	x	y	z
1	double x=1.0, y=2.0;			
2	x = y + 5.0;			
3	y = x / 2.0;			
4	y = (x * 3.0) + 4.0;			
5	x = -0.5 - y;			
6	z = x + y			

Coding_w06-02 (ต่อ)

หัวข้อ: Arithmetic Operators

2. เขียนโค้ด เพื่อคำนวณค่าต่างๆ ทั้ง 2 โปรแกรมย่อย และแสดงผลลัพธ์ แต่ละบรรทัด (พร้อมเขียนคอมเม้นอธิบาย)

w06-02-01

บรรทัดที่	นิพจน์	i	j	k
1	int i = 1, j = 2 , k;			
2	k = i + j;			
3	i = i + (k * j);			
4	j = i / 2;			
5	k = i % 2;			
6	i = (j + k) * 3;			

w06-02-02

บรรทัดที่	นิพจน์	x	y	z
1	double x=1.0, y=2.0;			
2	x = y + 5.0;			
3	y = x / 2.0;			
4	y = (x * 3.0) + 4.0;			
5	x = -0.5 - y;			
6	z = x + y			

Coding_w06-03

หัวข้อ: Relational & Logical Operators

<u>โจทย์</u>: กำหนดให้ x = 12, y = 7, z = 12;

ลำดับ	เงื่อนไขเปรียบเทียบ (Expression)	ผลลัพธ์ + วิธีคิด
1	x > y	
2	x < z	
3	x == z	
4	x != y	
5	!(2*5 >= y) || (5 != (5/3))	
6	!(x < y)	
7	(x + y) > (z * 2)	
8	(x % 2 == 0) || (y % 2 == 1)	
9	(x > y) && (z < y)	

Coding_w06-03 (ต่อ)

หัวข้อ: Relational & Logical Operators

<u>โจทย์</u>: กำหนดให้ x = 12, y = 7, z = 12;

1. จงหาผลลัพธ์ของนิพจน์ทั้ง 9 ข้อ (ด้วยการคำนวณมือ) พร้อมวิธีคิด/คำอธิบายประกอบ

2. ให้เขียนโปรแกรมเพื่อตรวจสอบเงื่อนไขเปรียบเทียบต่อไปนี้ และแสดงผลลัพธ์ว่าแต่ละเงื่อนไขเป็น จริง (1) หรือ เท็จ (0)

Coding_w06-04

หัวข้อ: การเขียนนิพจน์ในรูปย่อ (Short-hand Expression)

1. จงเขียนนิพจน์ต่อไปนี้ เป็นแบบย่อ

แบบเต็ม	แบบย่อ
x = x – 4.0;	
x = 6.5 * x;	
x = x % (y + z * a)	
x = x / (2.0 * x);	
total = total + (price * quantity - discount);	
x = x * (1 + rate / 100);	
score = score - (penalty * (mistake + 1));	

Coding_w06-04 (ต่อ)

หัวข้อ: การเขียนนิพจน์ในรูปย่อ (Short-hand Expression)

2. เขียนโค้ด เพื่อเปรียบเทียบผลลัพธ์ระหว่างแบบเต็ม และแบบย่อ พร้อมทั้งแสดงผลลัพธ์ (เขียนคอมเม้นอธิบาย)

แบบเต็ม	แบบย่อ
x = x – 4.0;	
x = 6.5 * x;	
x = x % (y + z * a)	
x = x / (2.0 * x);	
total = total + (price * quantity - discount);	
x = x * (1 + rate / 100);	
score = score - (penalty * (mistake + 1));	

Coding_w06-05

หัวข้อ: ลำดับการประเมินผลของนิพจน์ (Order of Evaluation in C Expressions)

1. จงคำนวณเพื่อหาผลลัพธ์ พร้อมวิธีคิด/คำอธิบายประกอบ ของนิพจน์ต่อไปนี้ (คำนวณมือ)

นิพจน์	ผลลัพธ์ + วิธีคิด
A = -2 + 5 * 2;	
B = 10/2 * 3;	
C = 6 / 2 + 3 * (4 % 2);	
D = (5+2) * 15 % 4;	
E = 6 + 2 * 2 – 6 / 2	
F = 5 + 3 * 2 - 8 / 4 + (6 % 5);	
G = (4 + 3) * 2 - 10 / (2 + 3);	

Coding_w06-05 (ต่อ)

หัวข้อ: ลำดับการประเมินผลของนิพจน์ (Order of Evaluation in C Expressions)

2. เขียนโค้ด เพื่อคำนวณค่าของนิพจน์ต่อไปนี้ และแสดงผลลัพธ์ แต่ละบรรทัด (พร้อมเขียนคอมเม้นอธิบาย)

นิพจน์	ผลลัพธ์
A = -2 + 5 * 2;	
B = 10/2 * 3;	
C = 6 / 2 + 3 * (4 % 2);	
D = (5+2) * 15 % 4;	
E = 6 + 2 * 2 – 6 / 2	
F = 5 + 3 * 2 - 8 / 4 + (6 % 5);	
G = (4 + 3) * 2 - 10 / (2 + 3);	

Coding_w06-06

หัวข้อ: การใช้ตัวดำเนินการหลายชนิดร่วมกัน + วิเคราะห์ผล (Advanced Practice with Discussion)

<u>โจทย์:</u> กำหนดค่าเริ่มต้นตัวแปรดังนี้: a = 5, b = 2, x = 3.0, y = 4.5

1. จงหาผลลัพธ์ พร้อมแสดงวิธีคิด ของนิพจ์ ดังนี้ (คำนวณมือ):

นิพจน์	ผลลัพธ์ + วิธีคิด
int r1 = a++ * b + (int)y % 3;	
int r2 = (a > b) && ((int)x / b < 2);	
float r3 = ++x * y - a / 2;	
float r4 = ((x += 1.5) > y) || (b-- > 0);	

Coding_w06-06 (ต่อ)

หัวข้อ: การใช้ตัวดำเนินการหลายชนิดร่วมกัน + วิเคราะห์ผล (Advanced Practice with Discussion)

โจทย์: กำหนดค่าเริ่มต้นตัวแปรดังนี้: a = 5, b = 2, x = 3.0, y = 4.5

2. เขียนโปรแกรมประเมินนิพจน์ แสดงผลลัพธ์ พร้อมอภิปรายผลจากการ Coding:

นิพจน์	ผลลัพธ์
int r1 = a++ * b + (int)y % 3;	
int r2 = (a > b) && ((int)x / b < 2);	
float r3 = ++x * y - a / 2;	
float r4 = ((x += 1.5) > y) || (b-- > 0);	

สำหรับค่า float ให้แสดงด้วยทศนิยม 2 ตำแหน่ง

Learning Content (Weekly Schedule)

Status	Week	Topic/Content	Deliverables
✔	1	Course Introduction	Practice #1 Coding: Hello World" in VS Code + GitHub
✔	2	Fundamentals of Computer System	Practice #2 Group activity: variety of computer language and its application.
✔	3	Algorithmic Thinking	Practice #3 Flowchart from case study problem
✔	4	C Basics I: Program structure, Data types & variables	Practice #4 Coding: main, headers, variables
✔	5	C Basics II: Console I/O Functions, Input/Output	Practice #5 Coding: printf, scanf
✔	6	Operators and Expressions	Practice #6 Coding: operators
	7	Selection statements: If & Switch Statement	Practice #7 Coding: if, switch
Mid-Term Examination			
	8	Iteration / Loop Statements: while & for loops	Practice #8 Coding: while, for loops
	9	Iteration / Loop Statements: do-while	Practice #9 Coding: do-while loops and complex loops
	10	Arrays & strings	Practice #10 Coding: Arrays & strings
	11	Functions & modular code	Practice #11 Coding: Functions
	12	Pointers & memory model	Practice #11 Coding: Pointer
	13	Mini-Project planning	Project proposal
	14	Mini-Project Sprint #1: Program structure & Coding	Prototype code compiles & passes baseline tests
	15	Mini-Project Sprint #2: Testing & refinement + Finalize	Unit-test coverage report ≥ 70 % pushed + Final code (v1.0) tagged
	16	Project demo & final project presentation	Live demo (slide deck submitted) + Assessment (Self & peer assessment)
Final Examination			

Computer Programming

(120213600)

Week 06 : Operators and Expressions

Panadda Kongsilp

Electrical and Automation Engineering Technology

Faculty of Engineering and Technology

King Mongkut's University of Technology North Bangkok, Rayong Campus