

# ToBeIT

@ K M I T L 5 8

## PROGRAMMING



## การเขียนโปรแกรม (Programming)

การเขียนโปรแกรมคอมพิวเตอร์ หรือเรียกง่ายๆ ว่าการเขียนโค้ด (Coding) คือขั้นตอนการเขียน ทดสอบ และดูแล Source Code ของโปรแกรมคอมพิวเตอร์ ซึ่ง Source Code นั้นจะเขียนด้วยภาษาโปรแกรม (Programming Language)

### ขั้นตอนการเขียนโปรแกรม

ขั้นตอนการเขียนโปรแกรมหรือพัฒนาโปรแกรม มีขั้นตอนโดยสังเขปดังต่อไปนี้

1. วิเคราะห์ปัญหาและความต้องการ (Problem Analysis and Requirement Analysis)
2. การออกแบบ (Design)
3. การเขียนโค้ด (Coding)
4. การทดสอบ (Testing)
5. การจัดทำเอกสาร (Documentation)
6. การบำรุงรักษา (Maintenance)

### ระดับของภาษาโปรแกรม

ภาษาในการเขียนโปรแกรม ถูกแบ่งตามความง่ายต่อการอ่าน ซึ่งแบ่งออกเป็น 3 ระดับหลักๆ คือ

#### 1. Machine Language (ภาษาเครื่อง)

มีการใช้เขียนโปรแกรมด้วยภาษาเดียวเท่านั้นในช่วงก่อนปี ค.ศ. 1952 คือ ภาษาเครื่อง ซึ่งเป็นภาษาระดับต่ำที่สุด เนื่องจากภาษาเครื่องคือการนำเลขฐานสองมาประกอบเป็นรูปแบบต่างๆ แทนคำสั่ง และข้อมูลที่จัดเก็บ และเนื่องจากคอมพิวเตอร์และหน่วยประมวลผลที่ต่างชนิดกันนั้นจะมีรูปแบบของคำสั่งที่แตกต่างกัน ทำให้การเขียนโปรแกรมในสมัยนั้นยุ่งยากมาก จึงทำให้เกิดการพัฒนาภาษาระดับต่ำขึ้นมาต่อไป

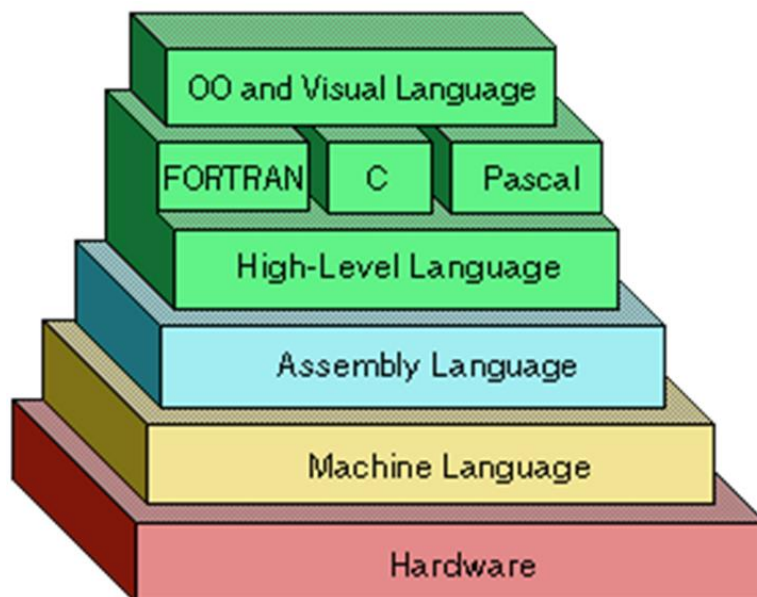
## 2. Low-level Language (ภาษาระดับต่ำ)

ในปี ค.ศ. 1952 มีการพัฒนาภาษาในการเขียนโปรแกรมขึ้นมาใหม่ นั่นคือ ภาษาแอสเซมบลี (Assembly language) โดยภาษานี้จะเป็นการใช้ตัวอักษรเป็นรหัสแทนชุดคำสั่งที่เป็นภาษาเครื่อง ทำให้การเขียนโปรแกรมเป็นไปได้ง่ายขึ้น (การใช้สัญลักษณ์แทนชุดคำสั่งภาษาเครื่อง เรียกว่า นิโมนิกโค้ด - Mnemonic code)

## 3. High-level Language (ภาษาระดับสูง)

ในปี ค.ศ. 1960 ได้มีการพัฒนาภาษาในการเขียนโปรแกรมที่มนุษย์เข้าใจง่ายมากขึ้น โดยใช้คำศัพท์ภาษาอังกฤษแทนชุดคำสั่งต่างๆ และยังสามารถใช้นิพจน์ทางคณิตศาสตร์ในการคำนวณได้อีกด้วย ทำให้การพัฒนาโปรแกรมเป็นไปอย่างมีประสิทธิภาพมากขึ้น เพราะนักพัฒนาจะได้ใช้เวลาในการคำนึงถึงวิธีที่จะใช้แก้ปัญหา มากกว่าความถูกต้องของชุดคำสั่งที่มีมากมาย และไม่ต้องกังวลว่าคอมพิวเตอร์จะทำงานอย่างไร

นอกจากสามระดับภาษาที่กล่าวมาแล้ว ยังมีการพัฒนาภาษาในการเขียนโปรแกรมให้มีความสมบูรณ์และเป็นมิตรกับนักพัฒนาอย่างต่อเนื่อง โดยมีการทำให้ใกล้เคียงกับลักษณะการพูดของมนุษย์ เช่น ภาษา SQL หรือ Delphi เป็นต้น และยังมีภาษาที่มีความสามารถสูงขึ้นไปอีก เช่น รองรับการทำงานร่วมกับเทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence : AI)



ลำดับชั้นของภาษาที่ใช้ในคอมพิวเตอร์

## การแปลภาษาในเขียนโปรแกรมเป็นภาษาเครื่อง

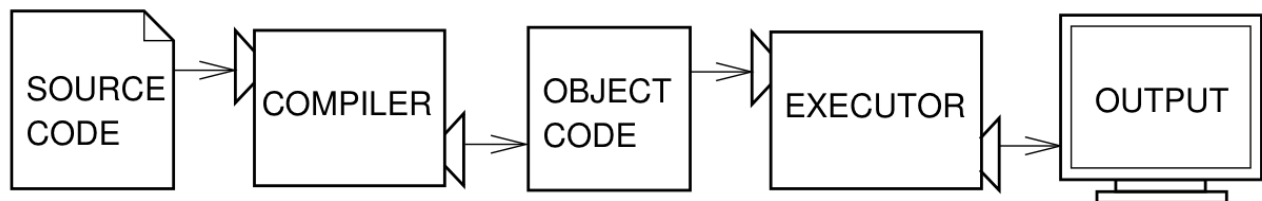
การเขียนโปรแกรมด้วยภาษาต่างๆ ล้วนต้องมีการแปลเป็นภาษาเครื่องเพื่อให้คอมพิวเตอร์เข้าใจได้ และทำการสั่งการอุปกรณ์ต่างๆ ให้ทำงานหรือคำนวณตามที่นักพัฒนาต้องการได้

- **Assembler**

ใช้แปลภาษา Assembly เป็นภาษาเครื่อง

- **Compiler**

แปลโค้ดของโปรแกรม (source code) ให้เป็นภาษาแอสเซมบลีหรือภาษาเครื่อง โดยแปลทั้งโปรแกรมในหนึ่งครั้ง และสร้าง object program ขึ้นมา เมื่อมีการเรียกใช้งานโปรแกรม จะสามารถเรียก object program ขึ้นมาใช้งานได้ทันที (คอมไพเลอร์ที่สมบูรณ์ตัวแรกคือภาษา FORTRAN โดย IBM ในปี ค.ศ.1957)



ลักษณะการทำงานของ Compiler

- **Interpreter**

แปลโค้ดของโปรแกรมให้เป็นภาษาเครื่อง โดยการทำงานจะแตกต่างกับคอมไพเลอร์ตรงที่ interpreter จะแปลทีละบรรทัด แล้วทำงานทันที หากบรรทัดใดมีจุดผิดพลาด จะหยุดการทำงาน และแจ้งข้อผิดพลาดนั้น



ลักษณะการทำงานของ Interpreter

## ประเภทของข้อมูล (ตัวแปร) ในการเขียนโปรแกรม

- Integer (จำนวนเต็ม)
  - เป็นได้ทั้งจำนวนเต็มลบ จำนวนเต็มศูนย์ และจำนวนเต็มบวก
  - ไม่สนใจทศนิยม
- Float (ทศนิยม)
  - เป็นจำนวนจริง
  - มีทศนิยมตั้งแต่ 1 ตำแหน่งขึ้นไป
- Character (ตัวอักษร)
  - เป็นตัวอักษรเพียงตัวเดียว
  - ครอบด้วย ' ' (single quote)\*
- String (สายอักขระ)
  - ประกอบด้วยตัวอักษรตั้งแต่สองตัวอักษรเป็นต้นไป (รวมทั้ง Space และตัวอักขระพิเศษ)
  - ครอบด้วย " " (double quote)\*

\* อ้างอิงจากหลักการเขียนภาษาซี และภาษาจาวา

ข้อมูลตัวอย่าง	ชนิดของข้อมูล				Note
	Int	Float	Char	String	
"IT KMITL"					
19					
'T'					
58					
"100"					
"18.98"					
"1,123"					
-19					
-15					
' '					

## กระบวนการทำงานของโปรแกรม



Input	การนำข้อมูลเข้า หรือการป้อนข้อมูลเข้าสู่ระบบ
Process	กระบวนการต่างๆ ที่กระทำกับข้อมูล (การคำนวณ การเปรียบเทียบ ฯลฯ)
Output	การนำผลลัพธ์ที่คาดหวังจากการทำงานของโปรแกรมออกจากระบบ

## กระบวนการวิเคราะห์ปัญหา

- 1) ทำความเข้าใจสิ่งที่โจทย์ต้องการ (คำตอบที่ต้องการจากการแก้ปัญหา : Output)
- 2) วิเคราะห์ข้อมูลที่ต้องใช้ในการแก้ปัญหา (การหา Input)
- 3) เลือกขั้นตอน (อัลกอริทึม) ที่เหมาะสมในการแก้ปัญหา (การจัด Process)

## อัลกอริทึม (Algorithm)

อัลกอริทึม (Algorithm) หมายถึง ลำดับขั้นตอนวิธีการทำงานของโปรแกรมอย่างเป็นขั้นตอนและชัดเจน เพื่อนำไปสู่การแก้ปัญหาใดปัญหาหนึ่ง ซึ่งถ้าปฏิบัติตามขั้นตอนอย่างถูกต้องแล้ว จะต้องสามารถแก้ปัญหาหรือประมวลผลตามความต้องการได้สำเร็จ

การแก้ไขปัญหามักทำได้หลายวิธี นั่นหมายถึงเราสามารถใช้อัลกอริทึมที่หลากหลายมาแก้ไขปัญหานั้นได้ ซึ่งการเลือกอัลกอริทึมต่างๆ มาใช้ในการแก้ปัญหานั้นเป็นสิ่งสำคัญ

ตัวอย่าง “การส่งพัสดุ”

เป้าหมาย : พักส่งพัสดุถึงมือผู้รับปลายทาง

Algorithm 1 (วิธีที่ 1)	Algorithm 2 (วิธีที่ 2)	Algorithm 3 (วิธีที่ 3)
1) จัดเตรียมพัสดุ 2) ส่งผ่านไปรษณีย์ไทย 3) ปลายทางรอรับพัสดุ	1) จัดเตรียมพัสดุ 2) ส่งผ่านเครื่องบิน 3) ปลายทางรอรับพัสดุ	1) จัดเตรียมพัสดุ 2) เดินทางไปส่งด้วยตนเอง

### ความแตกต่างของแต่ละอัลกอริทึม

- ความรวดเร็วที่แตกต่างกัน
- ค่าใช้จ่ายของแต่ละช่องทางการส่ง
- ระยะเวลาที่ใช้ตั้งแต่เริ่มต้น จนถึงปลายทาง

### ประสิทธิภาพของอัลกอริทึม

พิจารณาจาก 2 ส่วนหลักๆ ดังนี้

1. หน่วยความจำ (Memory) ที่จะต้องใช้ในการประมวลผล
2. เวลา (Time) ที่ใช้ในการประมวลผล

### คุณสมบัติของอัลกอริทึมที่ดี

1. อัลกอริทึมที่ดีต้องมีความถูกต้อง (Correctness)
2. อัลกอริทึมที่ดีต้องง่ายต่อการอ่าน (Readability)
3. อัลกอริทึมที่ดีต้องสามารถปรับปรุงได้ง่ายในภายหลัง (Ease of modification)
4. อัลกอริทึมที่ดีสามารถนำกลับมาใช้ใหม่ได้ (Reusability)
5. อัลกอริทึมที่ดีต้องมีประสิทธิภาพ (Efficiency)

## การอธิบายการทำงานของโปรแกรม

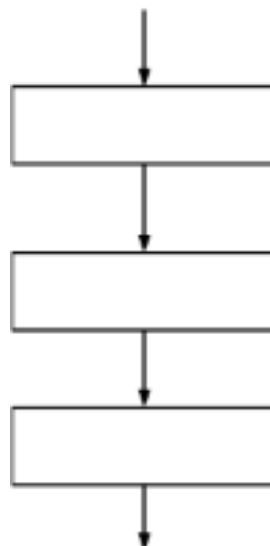
การอธิบายการทำงานของโปรแกรมหรืออัลกอริทึม นั้นทำขึ้นเพื่อแสดงลำดับขั้นตอนการทำงานของโปรแกรม เพื่อให้เป็นระบบและเข้าใจได้ง่าย โดยมีโครงสร้างและวิธีการที่นิยมใช้อยู่สองแบบคือ

1. **ผังงาน (Flowchart)** เป็นแผนผังแสดงขั้นตอนการทำงานต่างๆ ด้วยสัญลักษณ์ต่างๆ และลูกศรกำกับทิศทางการทำงาน
2. **ชุดคำสั่งเทียม (Pseudo code)** เป็นการแสดงขั้นตอนการทำงานของโปรแกรมด้วยศัพท์ภาษาอังกฤษ โดยมีรูปแบบคล้ายๆ กับการเขียนโปรแกรมจริงๆ แต่จะเข้าใจได้ง่ายกว่า

## รูปแบบการทำงานของโปรแกรม

- **การทำงานตามลำดับ (Sequence)**

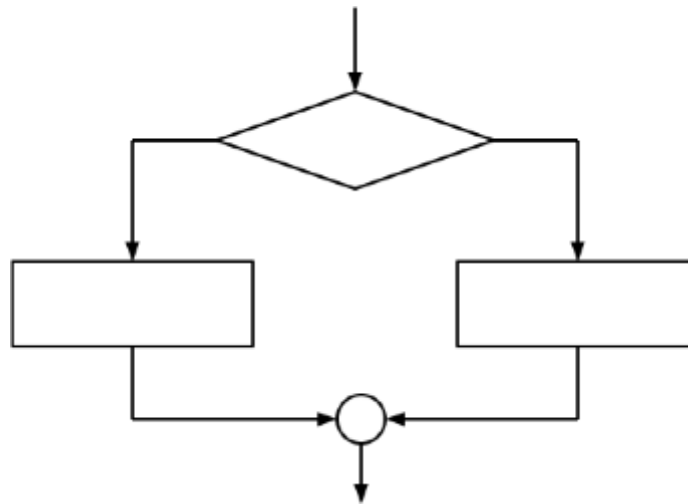
รูปแบบการทำงานที่เป็นพื้นฐานและเข้าใจได้ง่ายที่สุด คือการทำตามขั้นตอนไปเรื่อยๆ ทีละขั้น หรือการทำตามขั้นตอนจากบนลงล่าง





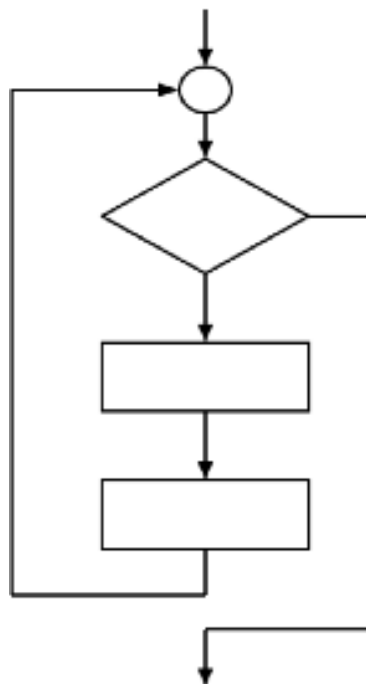
- การเลือกทำตามเงื่อนไข (Decision-based)

รูปแบบการทำงานโดยยึดตามเงื่อนไขที่กำหนด เช่น ถ้าเงื่อนไขที่กำหนดเป็นจริงจะทำงานแบบหนึ่ง หากเงื่อนไขเป็นเท็จจะทำงานอีกแบบหนึ่ง



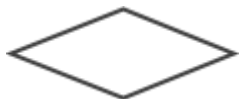








- การทำซ้ำ (Looping or Repeating)

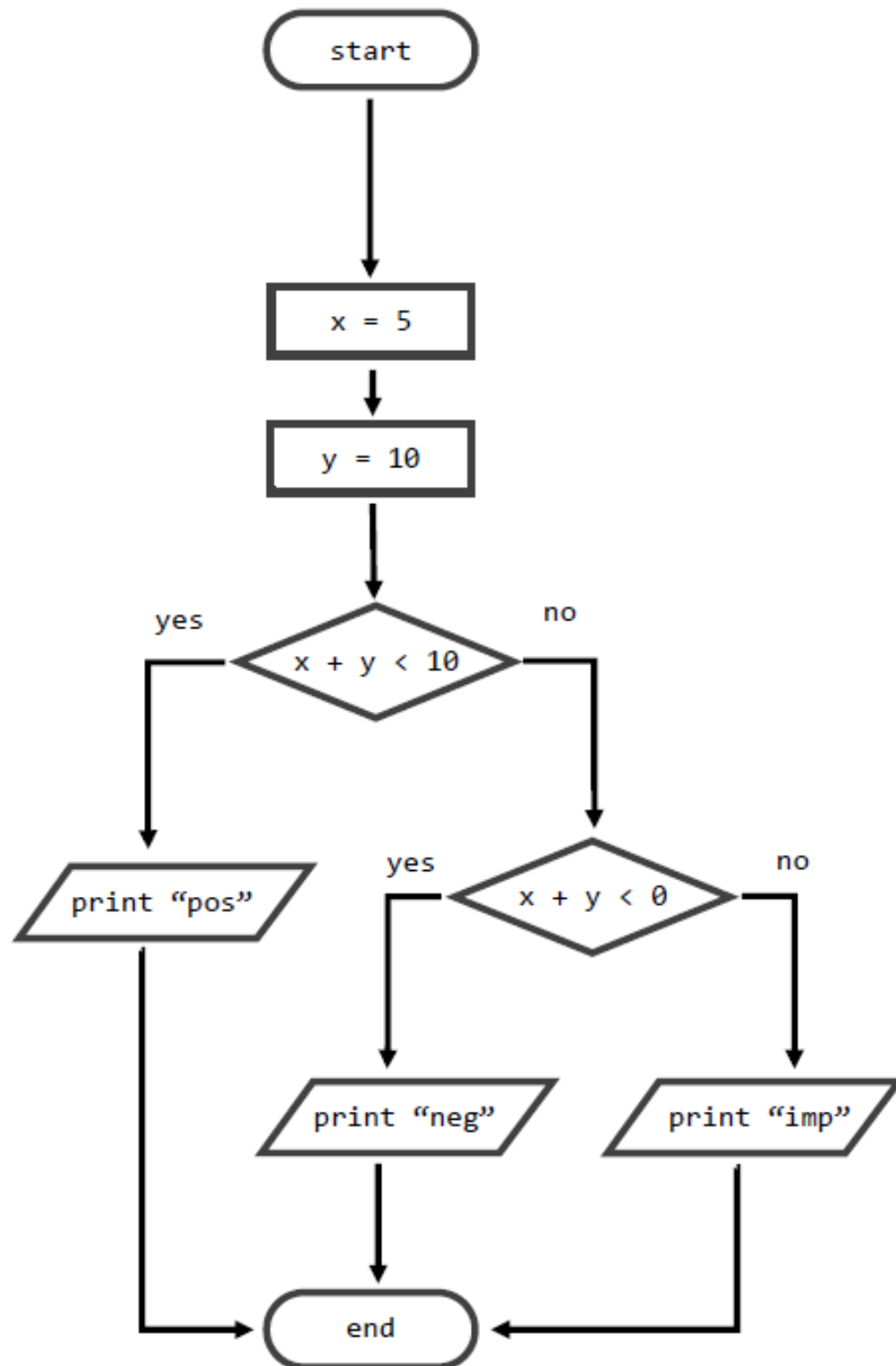
การทำงานที่มีการวนซ้ำขั้นตอนเดิมจนกว่าเงื่อนไขที่กำหนดจะเป็นจริง ซึ่งจะสังเกตว่าเป็นการผสมรูปแบบการทำงานแบบลำดับและแบบมีเงื่อนไขเข้าด้วยกัน



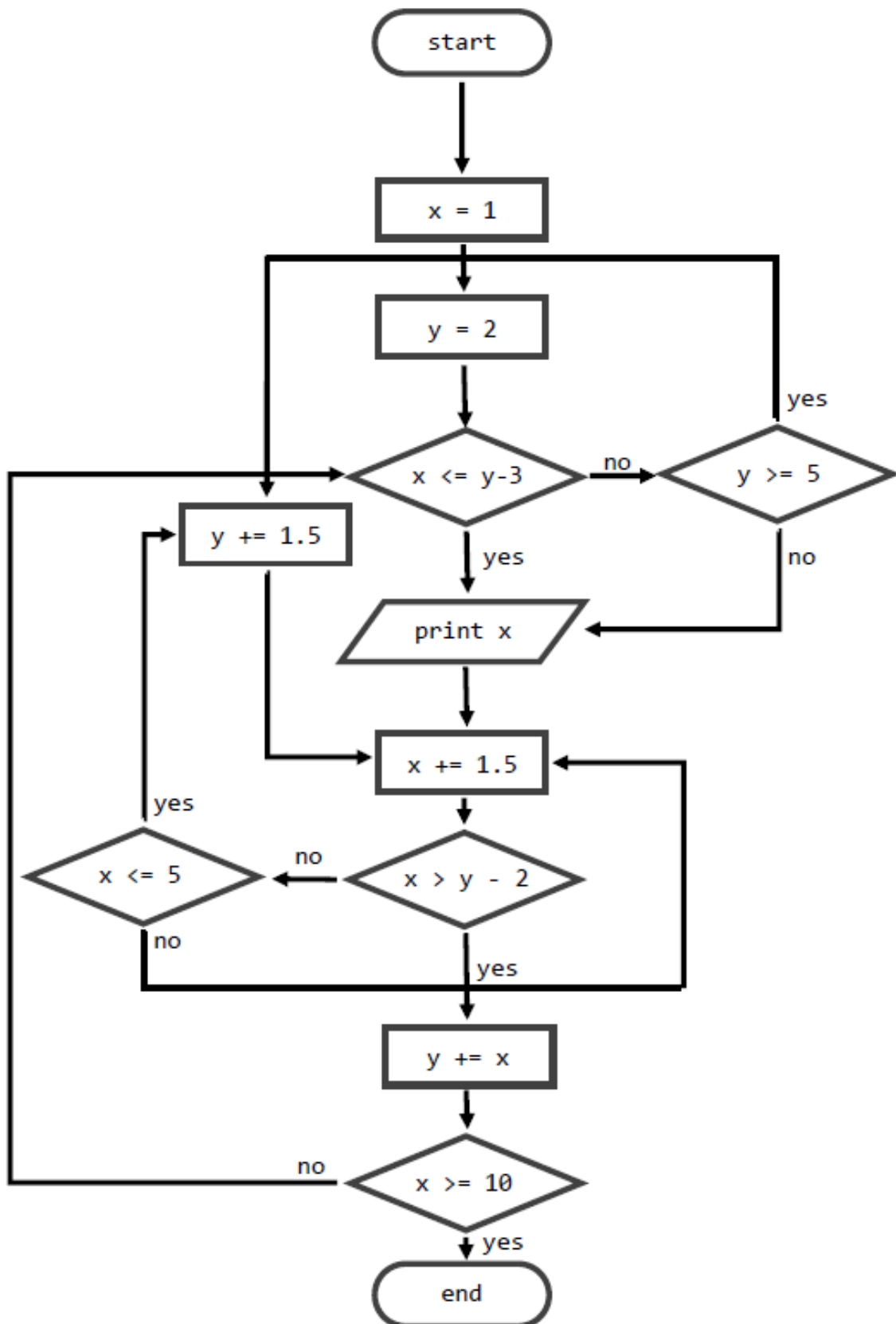
## สัญลักษณ์ของผังงาน (Flowchart)

สัญลักษณ์	ความหมาย
	จุดเริ่มต้นหรือสิ้นสุดการทำงาน (ใช้ข้อความ Start และ Stop กำกับ)
	การประมวลผล หรือการคำนวณ
	เงื่อนไขต่างๆ (จะมีสองเส้นทางออกจากสัญลักษณ์นี้ คือที่เป็นจริงหรือที่เป็นเท็จ)
	การรับหรือแสดงข้อมูล (ไม่ระบุชนิดอุปกรณ์)
	แสดงข้อมูลหรือผลลัพธ์ทางจอภาพ
	แสดงข้อมูลหรือผลลัพธ์ทางเครื่องพิมพ์
	จุดเชื่อมต่อผังงาน (ใช้เชื่อมลูกศรหลายทิศทางเข้าที่จุดเดียวกัน)
	จุดเชื่อมต่อระหว่างหน้า (เมื่อจบหน้ากระดาษ จะมีตัวเลขกำกับบ่งบอกจุดเชื่อมต่อ)
	ลูกศรแสดงทิศทางการทำงานของโปรแกรม

ตัวอย่างผังงาน (Flowchart)



## ตัวอย่างผังงาน (Flowchart)



## รูปแบบของชุดคำสั่ง

- การตัดสินใจสองทางเลือก

```
1 if (x > 0){
2     printf("X is Positive");
3 }
4 else{
5     printf("X is Negative");
6 }
```

- การตัดสินใจหลายทางเลือก

```
1 if (x > 0){
2     printf("X is Positive");
3 }
4 else{
5     if (x < 0){
6         printf("X is Negative");
7     }
8     printf("X is 0");
9 }
```

```
1 switch (x){
2     case 1: printf("This is case 1");
3             break;
4     case 2: printf("This is case 2");
5             break;
6     case 3: printf("This is case 3");
7             break;
8     default: printf("Something went wrong!!!");
9 }
```

- การกระทำวนซ้ำ (ทำก่อนตรวจเงื่อนไข)

```
1 repeat
2     sum := sum + number;
3     number := number - 2;
4 until number = 0;
```

```
1 while number < 6 do
2 begin
3     sum := sum + number;
4     number := number - 2;
5 end;
```

- การกระทำวนซ้ำ (ตรวจเงื่อนไขก่อนทำ)

```
1 while number <= -2:
2     print number
3     number += 1
```

- ทำงานตามจำนวนรอบที่แน่นอน

```
1 for x := 1 to 10 do
2 begin
3     writeIn(x);
4 end;
```

## ชุดคำสั่งเทียม (Pseudo Code)

เป็นคำอธิบายขั้นตอนการทำงานของโปรแกรม โดยใช้ถ้อยคำที่ไม่มีรูปแบบการเขียนตายตัว อาจจะเขียนเป็นภาษาไทยก็ยังได้ แต่ยิ่งภาษาอังกฤษมีความเข้าใจที่ดีกว่า โดยปกติมักเขียนให้สั้น และได้ใจความ

ผู้เขียนโปรแกรมสามารถใช้ Pseudo Code ในการหาช่องโหว่หรือพัฒนาโปรแกรมของตนเองให้เป็นไปตามที่ผู้เขียนโปรแกรมตั้งใจไว้ได้

## ตัวอย่าง Pseudo code

ข้อ	Pseudo code	Output
1	<pre> 1 x = 5 2 while x &lt;= 5: 3     print "^" 4     x -= 1 </pre>	
2	<pre> 1 i = 4 2 j = 5 3 if i + j &lt;= 6: 4     print ((i + j)/2)*(3 * i) 5 else: 6     print ((i - j)/3)*(4 * i) </pre>	
3	<pre> 1 num1 := 3 2 num2 := 5 3 repeat 4     writeIn("IT KMITL"); 5     num1 := num1 + 1 6 until num1 + num2 := 10 </pre>	
4	<pre> 1 for number := 1 to 5 do 2 begin 3     number := (number + 5 * (number % 2)); 4     writeIn(number); 5 end; </pre>	
5	<pre> 1 """This is not pseudo code""" 2 def function(number, count = 0, temp = 0): 3     """This is program to practice your algorithm""" 4 5     for i in xrange(2, number + 1): 6         temp = 0 7         for answer in xrange(2, 20): 8             if i % answer == 0: 9                 temp += 1 10                break 11            if temp == 0: 12                count += 1 13 14     return count 15 print function(20) </pre>	

## ประเภทของข้อผิดพลาด (Error)

- Syntax Error

ข้อผิดพลาดที่เกิดจากเขียนโปรแกรมผิดเงื่อนไขของภาษานั้นๆ เช่น ผิดรูปแบบ การลืมเครื่องหมายสิ้นสุดคำสั่ง การเรียกตัวแปรที่ยังไม่ได้ประกาศ ฯลฯ

```
1 if (x > 0){  
2     printf("X is Positive");  
3 }  
4 else{  
5     printf("X is negative");  
6 }
```

- Runtime Error

ข้อผิดพลาดที่เกิดขึ้นขณะรันโปรแกรม เช่น โปรแกรมหารเลขที่มีการรับตัวตั้งและตัวหารจากผู้ใช้ แต่ผู้ใช้กำหนดตัวหารเป็น 0 ทำให้เกิดข้อผิดพลาดขึ้น ฯลฯ

```
1 for x := 5 downto -5 do  
2 begin  
3     x := 5 / x  
4     writeIn(x);  
5 end;
```

- Semantic Error

ข้อผิดพลาดที่เกิดขึ้นจากอัลกอริทึม (ขั้นตอนการทำงาน) เช่นการเรียกใช้ค่าในตัวแปรผิดตัว หรือข้อผิดพลาดจากการเขียนลำดับการคำนวณทางคณิตศาสตร์ ฯลฯ สรุปคือการรันโปรแกรมจะรันได้เสร็จสมบูรณ์ แต่คำตอบที่ได้ออกมาจะผิดพลาด

**\*\*\* Error ประเภทนี้พบบ่อยที่สุด \*\*\***