บทที่ 3

โครงสร้างภาษาซีเบื้องต้น

Basic C Programming Language

ประวัติภาษาซี

1967

BCPL

Dennis Ritchie

1970

B

Ken Thompson

1972

C

Dennis Ritchie

https://en.wikipedia.org/wiki/Bell_Labs

https://www.computerhistory.org/tdih/february/4/

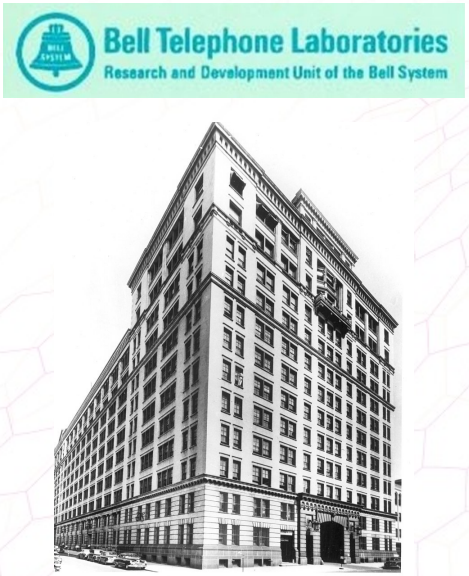
ประวัติภาษาซี

Dennis Ritchie
1941-2011

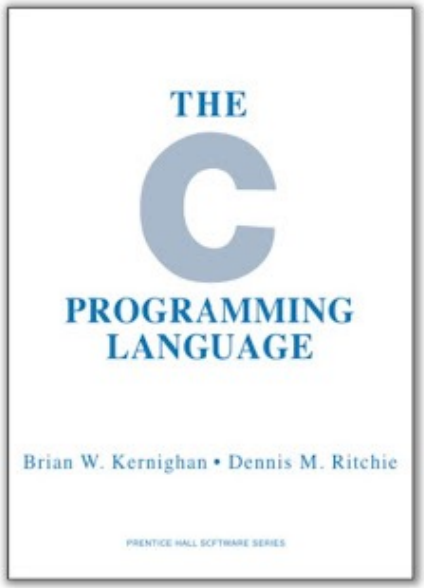
"C is quirky, flawed, and an enormous success."
— DENNIS RITCHIE

ประวัติภาษาซี

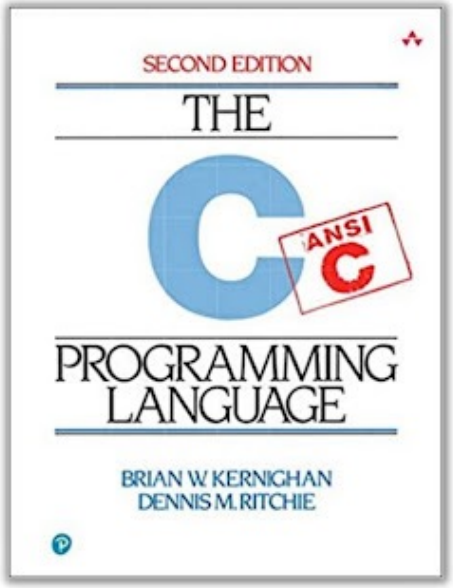
ในปี1978 หนังสือการเขียนภาษาซี



https://en.wikipedia.org/wiki/Bell_Labs



1st Edition



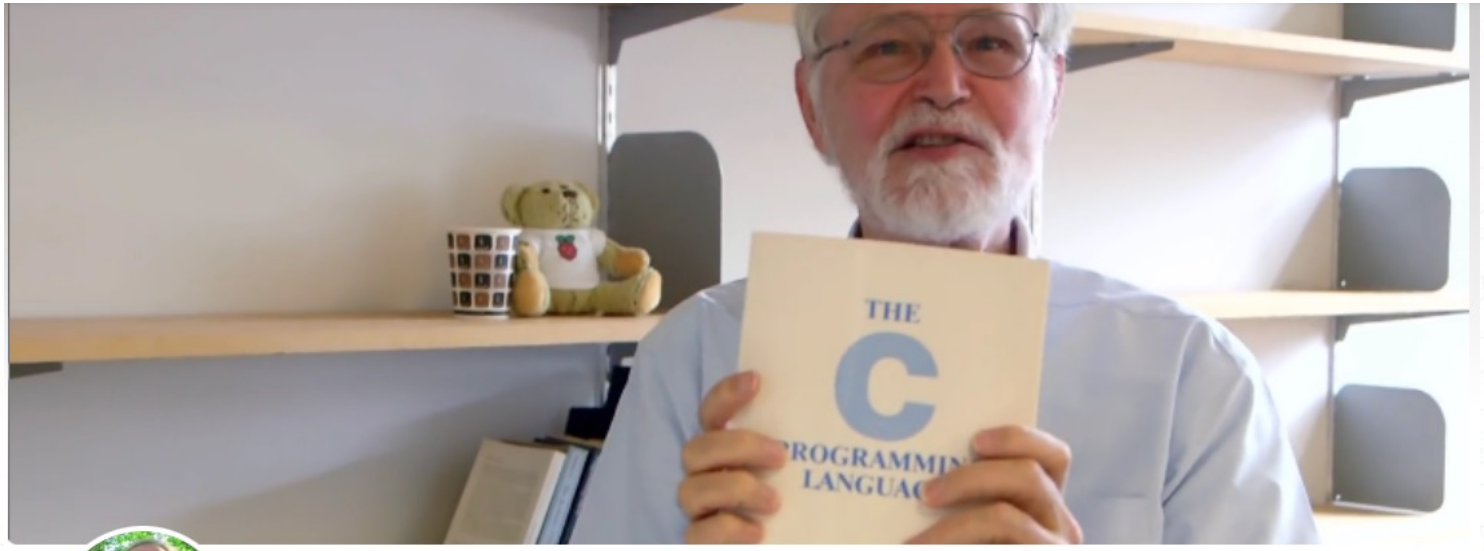
2nd Edition

https://en.wikipedia.org/wiki/The_C_Programming_Language

Brian Kernighan และ Ritchie
นั้นเป็นที่รู้จักกันในชื่อของ "K&R C"

ในปี1980 ภาษาซีก็กลายเป็นภาษาที่ได้รับความนิยม

ประวัติภาษาซี



Brian Kernighan
コミュニティ

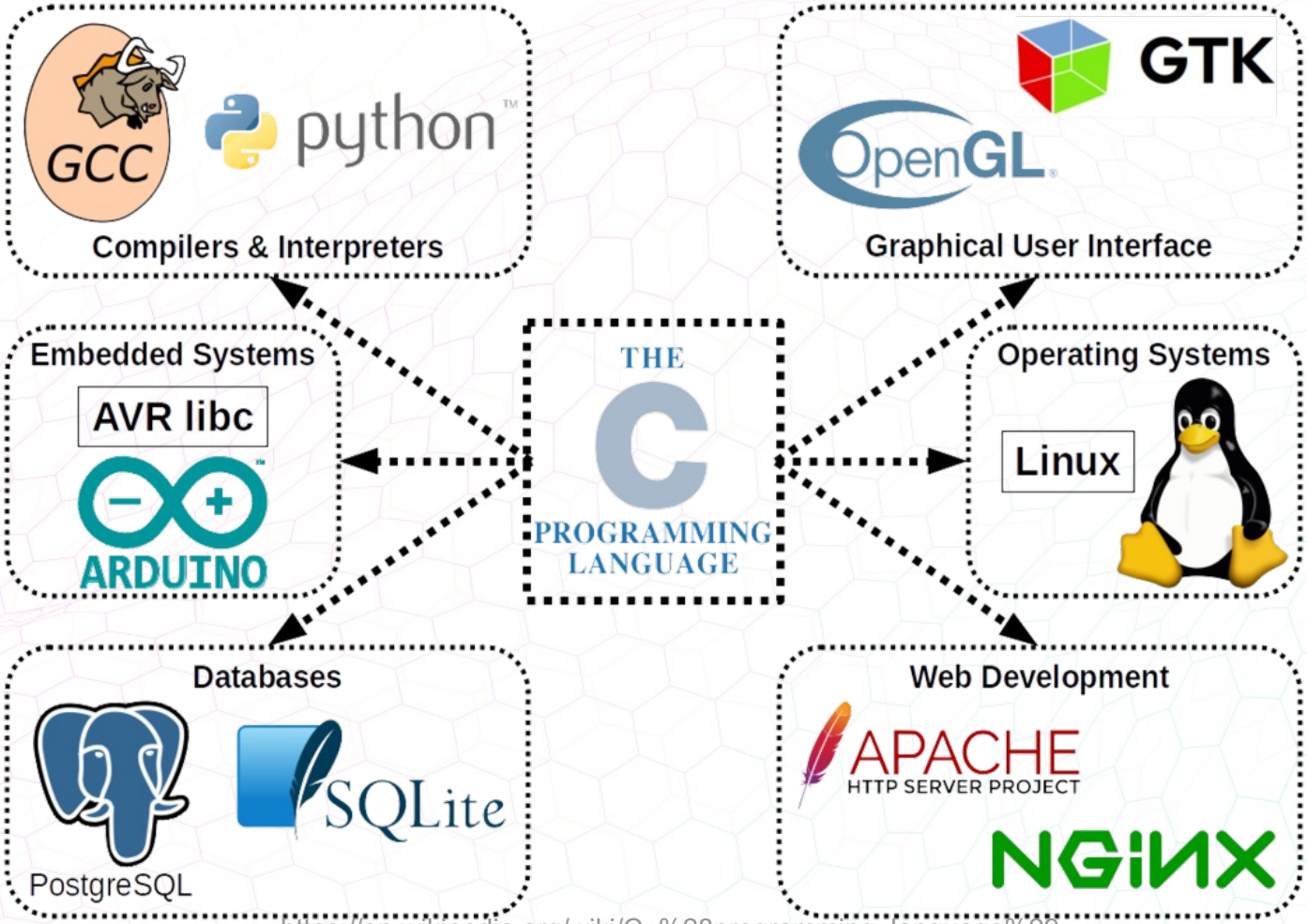
ホーム　基本データ　動画　写真　その他

```
main()
{
    printf("hello, world\n");
}
```

Brian Kernighan

ประวัติภาษาซี

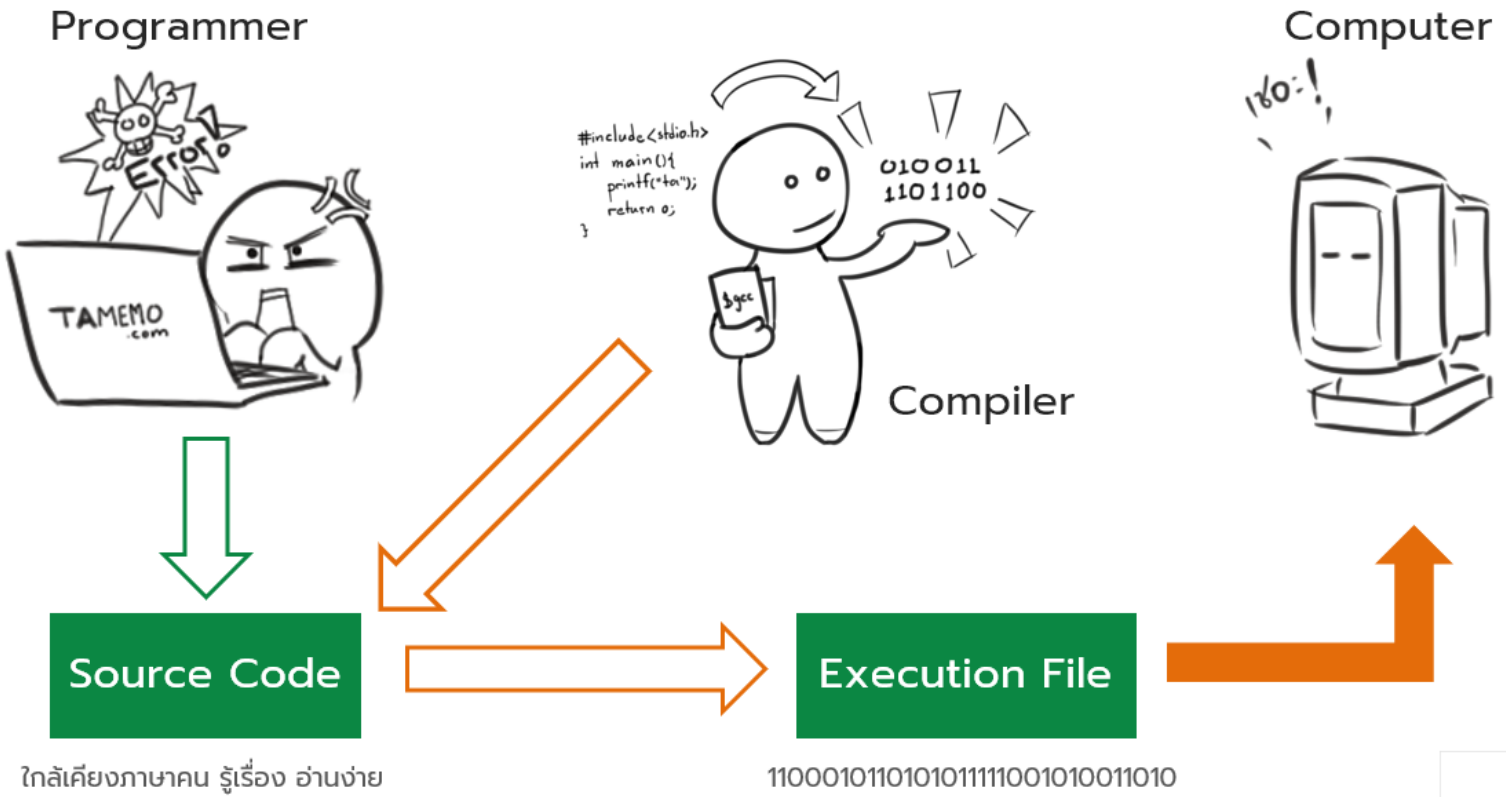
ภาษาซีเป็นภาษาระดับสูง ที่มีคุณสมบัติพิเศษที่สามารถใช้งานในระดับต่ำ (low-level) ได้ เปรียบเหมือนสะพานเชื่อมภาษาเครื่องเข้ากับภาษาระดับสูง



https://en.wikipedia.org/wiki/C_%28programming_language%29

กระบวนการแปลภาษาคอมพิวเตอร์

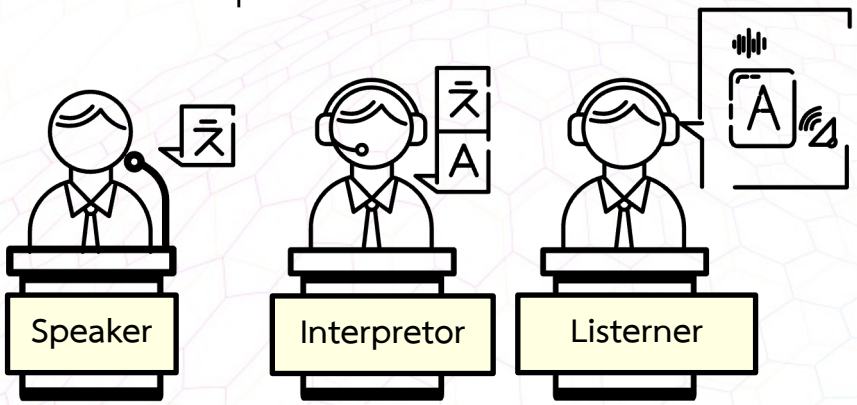
คอมพิวเตอร์ทำงานได้ต้องมีการประมวลผลภาษาเครื่องเท่านั้น เพื่อให้การเขียนโปรแกรมได้ง่ายขึ้นจึงพัฒนาเป็นภาษาคอมพิวเตอร์ จึงต้องมีโปรแกรมแปลภาษาคอมพิวเตอร์เป็นภาษาเครื่อง



ที่มา: https://www.tamemo.com/post/141/learn-intro-compiler-interpreter/

กระบวนการแปลภาษาคอมพิวเตอร์

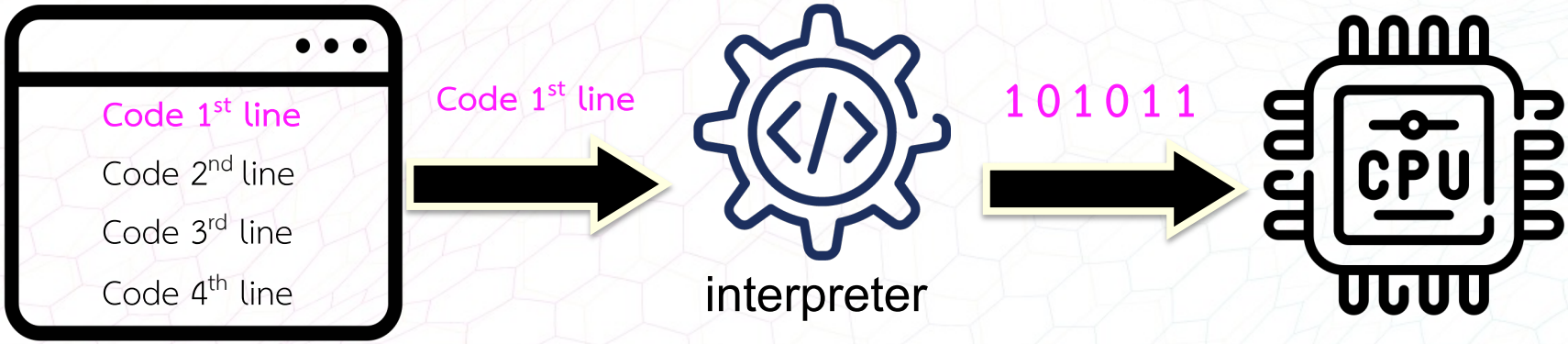
- โปรแกรมแปลภาษาเป็นภาษาเครื่อง มี 2 ประเภท
  - อินเทอร์พรีเตอร์ (Interpreter)



  - คอมไพเลอร์ (Compiler)

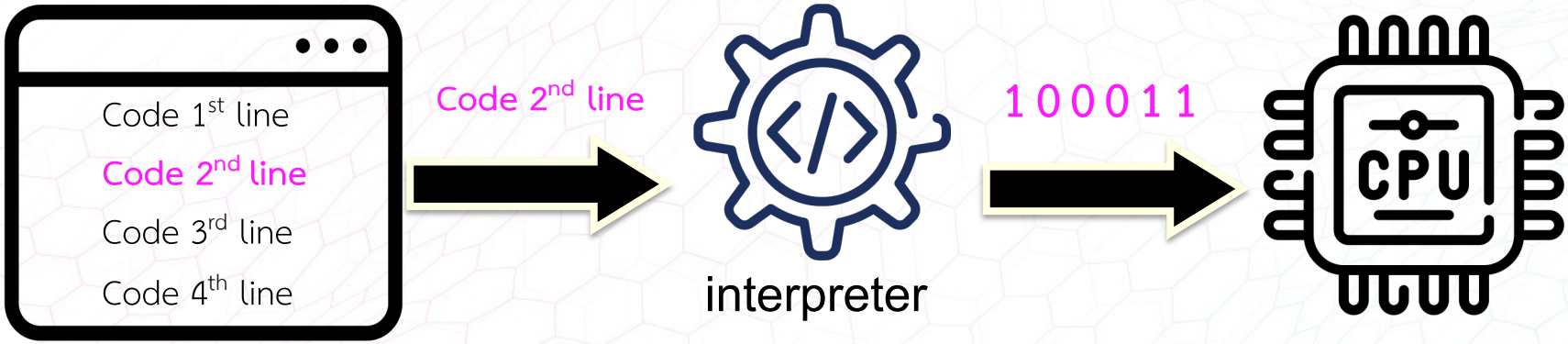
กระบวนการแปลภาษาคอมพิวเตอร์

- โปรแกรมแปลภาษาเป็นภาษาเครื่อง มี 2 ประเภท
  - อินเทอร์พรีเตอร์ (Interpreter)



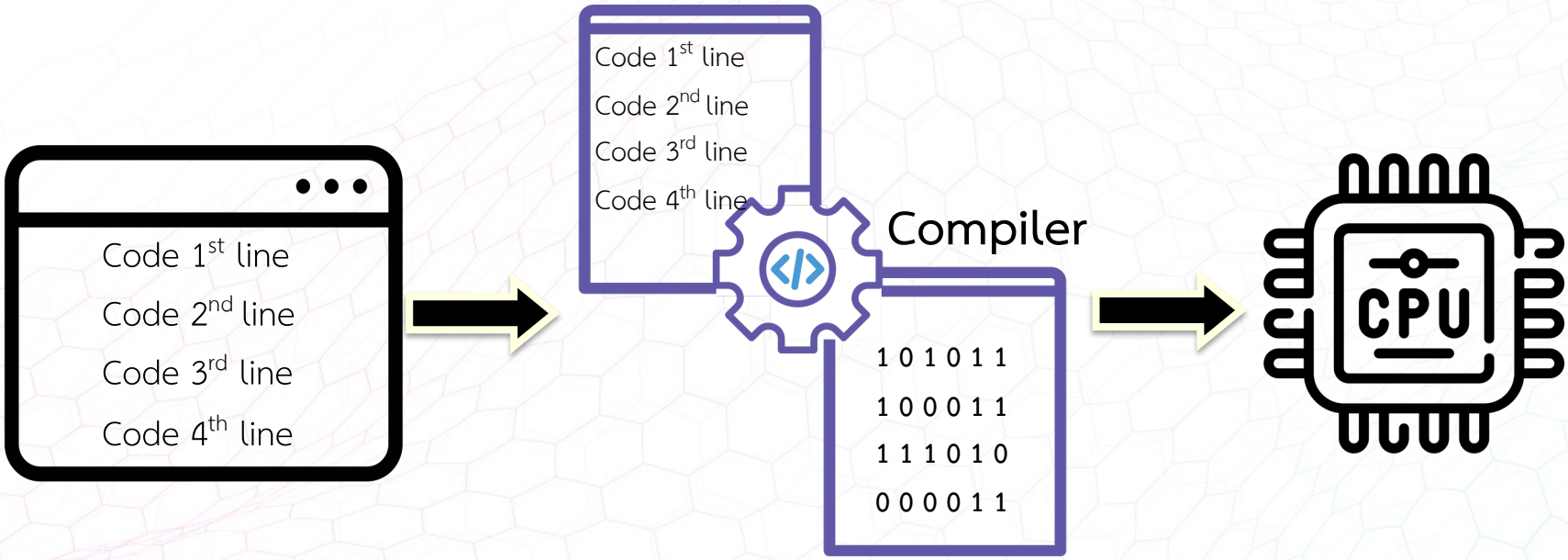
กระบวนการแปลภาษาคอมพิวเตอร์

- โปรแกรมแปลภาษาเป็นภาษาเครื่อง มี 2 ประเภท
  - อินเทอร์พรีเตอร์ (Interpreter)



กระบวนการแปลภาษาคอมพิวเตอร์

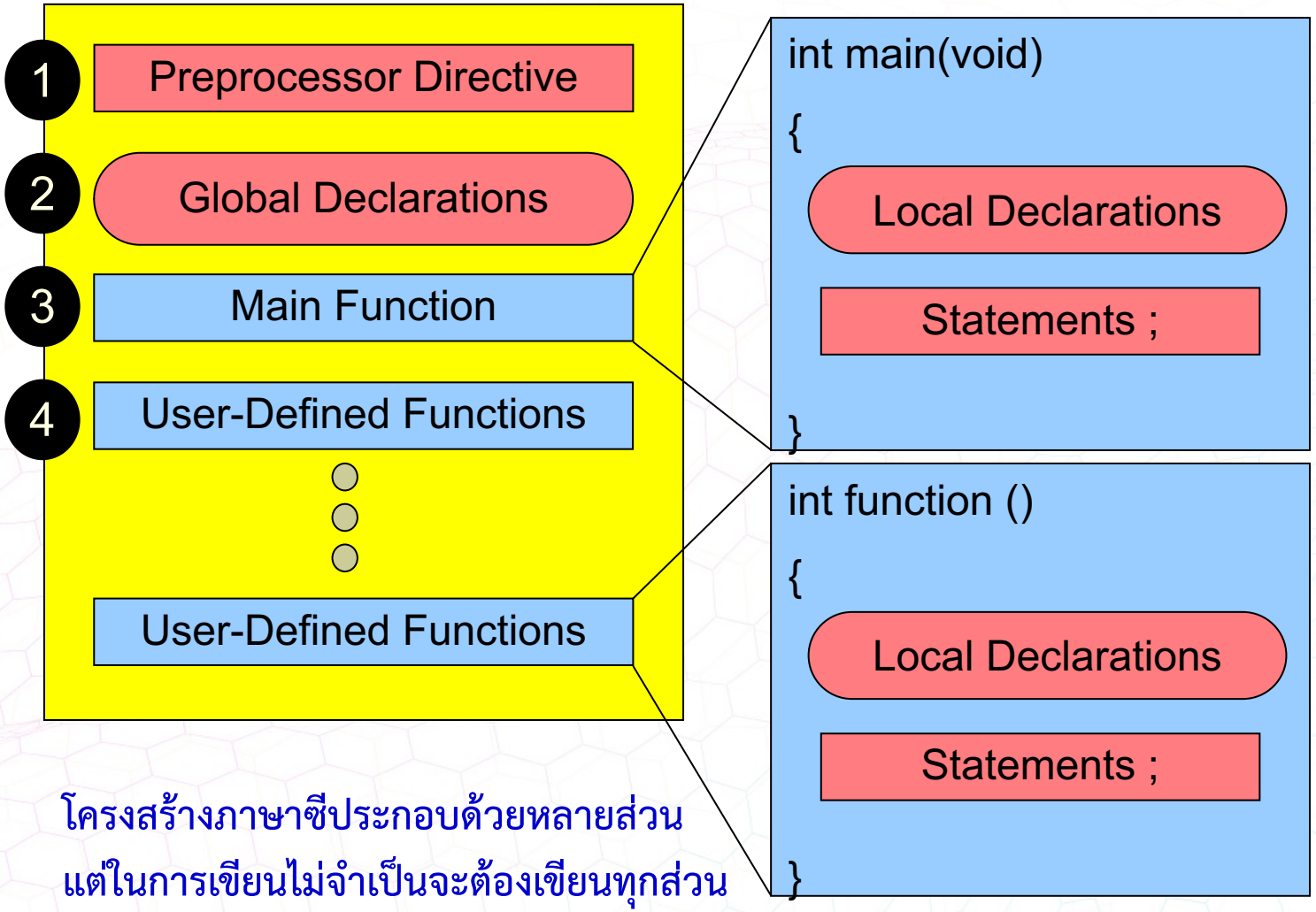
- โปรแกรมแปลภาษาเป็นภาษาเครื่อง มี 2 ประเภท
  - คอมไพเลอร์ (Compiler)



กระบวนการแปลภาษาคอมพิวเตอร์

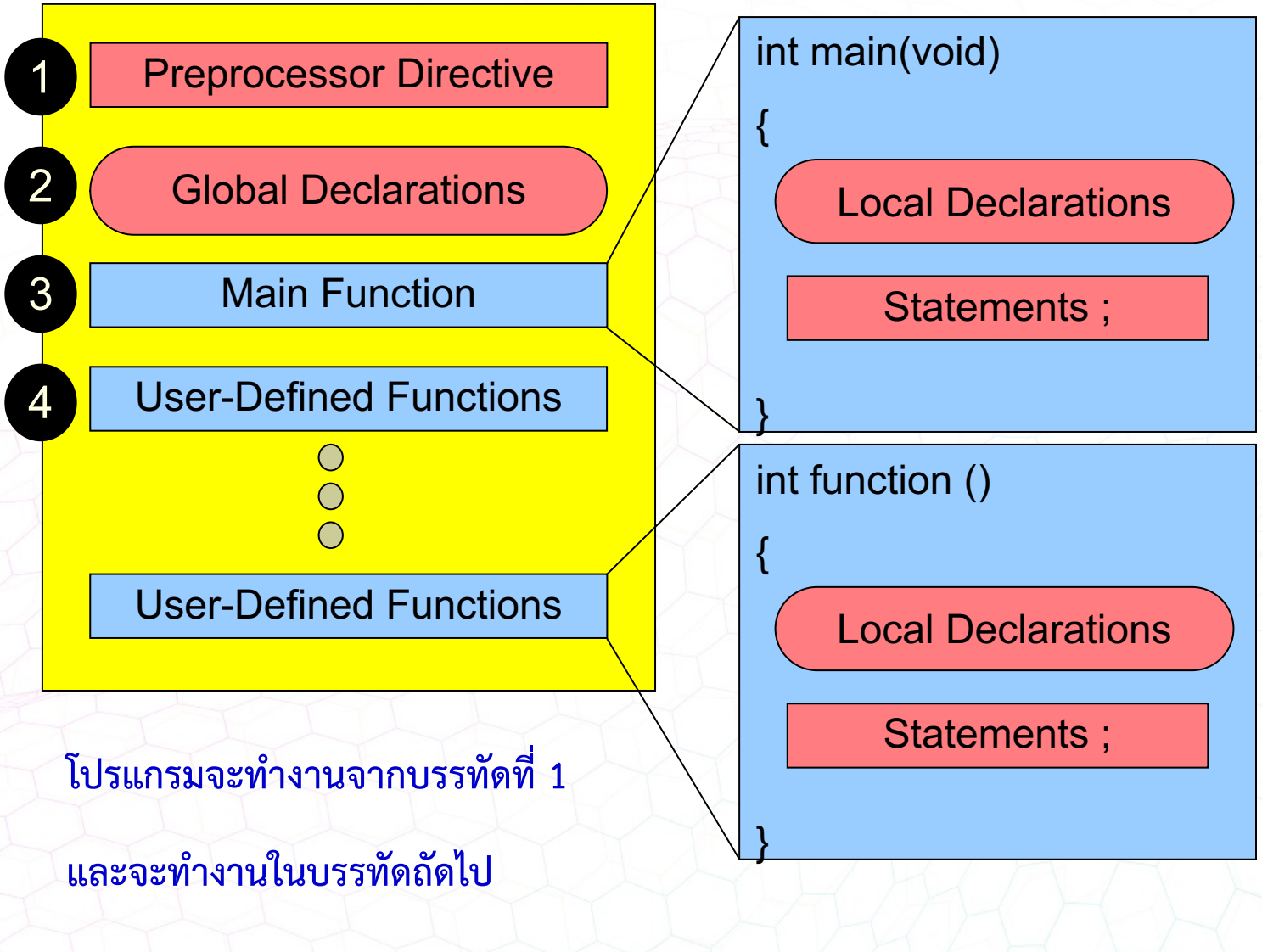
- **อินเตอร์พรีเตอร์**
  - **ข้อดี**
    - อินเตอร์พรีเตอร์ถูกสร้างขึ้นได้ง่ายกว่าและมีขนาดเล็ก ทำให้ภาษาที่ใช้อินเตอร์พรีเตอร์สามารถทำงานข้ามแพลตฟอร์มได้
  - **ข้อเสีย**
    - ทำงานได้ช้ากว่าคอมไพเลอร์
- **คอมไพเลอร์**
  - **ข้อดี**
    - ทำงานได้เร็ว
    - ตรวจสอบข้อผิดพลาดของโปรแกรมซอร์ดโค้ดในขั้นตอนของการคอมไพล์
  - **ข้อเสีย**
    - ต้องนำโปรแกรมซอร์ดโค้ดมาแปลใหม่เมื่อเปลี่ยนระบบปฏิบัติการ เนื่องจากคอมไพเลอร์เป็นตัวแปลภาษาที่ขึ้นอยู่กับแพลตฟอร์ม (Platform Specific)

โครงสร้างโปรแกรมภาษาซี



โครงสร้างภาษาซีประกอบด้วยหลายส่วน
แต่ในการเขียนไม่จำเป็นจะต้องเขียนทุกส่วน

โครงสร้างโปรแกรมภาษาซี



โปรแกรมจะทำงานจากบรรทัดที่ 1

และจะทำงานในบรรทัดถัดไป

โครงสร้างโปรแกรมภาษาซี

```
1  #include<file.h>                    Preprocessor Directive
   type function_name(type);           Prototype Functions
2  type variable;                      Global Declarations
   int main()
   {
       type variable;
3      statement-1;
       ...
       statement-n;
       return 0;
   }
   type function_name(type variable)
   {
4      statement-1;
       ...
       statement-n;
       return(var);
   }
```

Header

Main Function

User-Defined Functions

Body

Preprocessor Directive

ทุกโปรแกรมต้องมี

- ใช้เรียกไฟล์ที่โปรแกรมใช้ในการทำงานร่วมกัน
- ใช้กำหนดเงื่อนไขในการคอมไพล์ให้กับโปรแกรม
- Preprocessor Directives พื้นฐานทั่วไปที่นิยมใช้มีดังนี้
  - #include ใช้สำหรับเรียกไฟล์ที่โปรแกรมใช้ในการทำงาน
  - #define ใช้สำหรับกำหนดมาโครที่ให้กับโปรแกรม ใช้กำหนดค่าคงที่ให้กับโปรแกรม

#include	**#define**	#undef	#if
#ifdef	#ifndef	#else	#elif
#endif	#line	#error	#pragma

การใช้ #include

วิธีการใช้งาน

#include <ชื่อไฟล์> หรือ #include “ชื่อไฟล์”

ตัวอย่าง

```
#include <stdio.h>
```
(เป็นการเรียกใช้ไฟล์ stdio.h เข้ามาในโปรแกรม)

```
#include <mypro.h>
```
(เป็นการเรียกใช้ไฟล์ mypro.h เข้ามาในโปรแกรม)

< > จะเรียกไฟล์ใน directory ที่กำหนดโดยตัวคอมไพล์เลอร์

“ ” จะเรียกไฟล์ใน directory ที่ทำงานอยู่ในปัจจุบัน

การใช้ #define

วิธีการใช้งาน

#define ชื่อ ค่าที่ต้องการ

ตัวอย่าง

```
#define START 10        (กำหนดค่า START = 10)
#define A 3*5/4          (กำหนดค่า A=3*5/4)
#define pi 3.14159       (กำหนดค่า pi = 3.14159)
#define sum(a,b) a+b
        (กำหนดค่า sum(ตัวแปรที่1, ตัวแปรที่2) = ตัวแปรที่1+ตัวแปรที่2
```

Global Declarations

- เป็นการประกาศตัวแปรเพื่อใช้งานในโปรแกรม โดยตัวแปรนั้นสามารถใช้ได้ในทุกที่ในโปรแกรม
- เป็นส่วนที่ใช้ในการประกาศฟังก์ชันที่ผู้ใช้งานสร้างขึ้น (Function Prototype) ของโปรแกรม

⚠ ส่วนนี้ในบางโปรแกรมอาจไม่มีก็ได้

```
1  #include <stdio.h>
2  int x;
3  int main()
4  {
5          x = 5;
6          ...
7          Statement ;
8          return(int value);
9  }
```

ฟังก์ชันหลักของโปรแกรม (Main Function)

ส่วนนี้ทุกโปรแกรมจะต้องมี โดยโปรแกรมหลักจะเริ่มต้นด้วย main() และตามด้วยเครื่องหมายปีกกาเปิด ‘{’ และปีกกาปิด ‘}’

- แต่ละคำสั่งจะต้องจบด้วยเซมิโคลอน ‘;’ (Semicolon)
- ต้องมี return(); เสมอ และต้องใส่เลขจำนวนเต็ม เช่น 0 = Success , 1 = Failure

```
#include <stdio.h>
int main(void)
{
        ...
        Statement (คำสั่ง) ;
        return(int value);
}
```

การสร้างฟังก์ชันใช้งานเอง (User Define Function)

- สร้างฟังก์ชันหรือคำใหม่ ขึ้นมาใช้งานตามที่เราต้องการ
- ระหว่างปีกกาจะประกอบด้วยคำสั่ง(Statement) ต่างๆ ที่จะให้ฟังก์ชันทำงาน
- สามารถเรียกใช้ภายในโปรแกรมได้ทุกที่

```
#include <stdio.h>
int function();
int main(void)
{
        ...
        Statement ;
        return(int value);
}
int function()
{
        Statement ;
        ...
        return (int value);
}
```

คำอธิบายของโปรแกรม (Program Comments)

- ใช้เขียนส่วนอธิบายโปรแกรม (คอมเมนต์)
- ช่วยให้ผู้ศึกษาโปรแกรมภายหลังเข้าใจการทำงานของโปรแกรม
- ส่วนของคำอธิบายจะถูกข้ามเมื่อคอมไพล์โปรแกรม

การเขียนส่วนอธิบายโปรแกรม (comments)ทำได้ 2 วิธีคือ

// สำหรับคำอธิบายไปจนถึงท้ายบรรทัด

และ

/* คำอธิบาย */ ลักษณะการใช้เหมือนวงเล็บนั้นเอง

ตัวอย่าง :คำอธิบายของโปรแกรม (Program Comments)

Comment ส่วนที่
คอมไพเลอร์จะไม่นำไปแปล

```
#include <stdio.h>          // Change Feet to Inches
int main()                  // by CPE RMUTT
{                           // Start
  int feet,inches;
  feet = 6;                 // feet ← 6
  inches = feet * 12;       // inches ← feet * 12
  printf("Height in inches is %d", inches);
  return(0);                // write inches
}                           // Stop
```

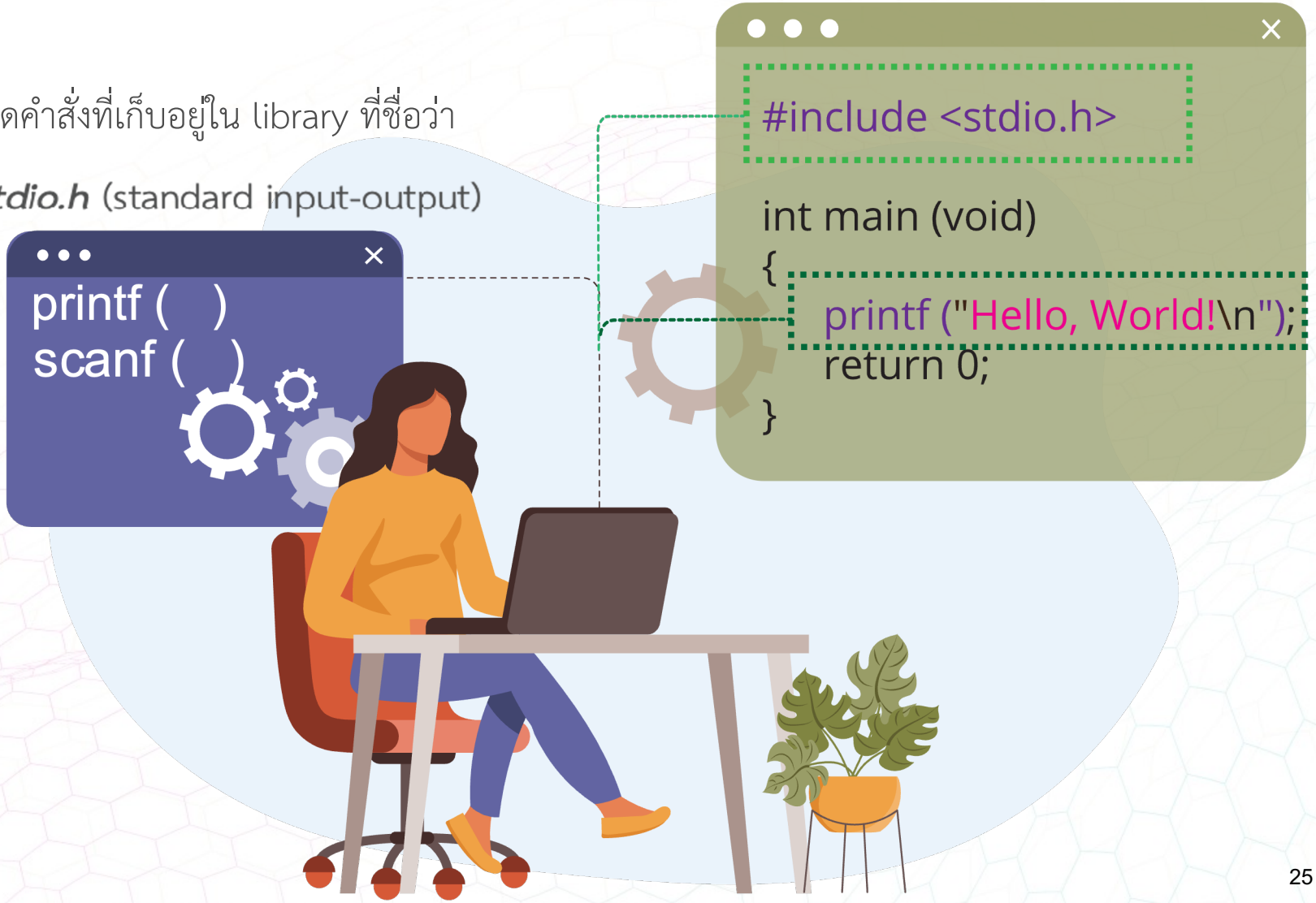
ผลการทำงาน

```
Height in inches is 72
```

ประโยคคำสั่งเบื้องต้น การรับข้อมูล และ การแสดงข้อมูล

ชุดคำสั่งที่เก็บอยู่ใน library ที่ชื่อว่า

stdio.h (standard input-output)



การใช้คำสั่ง printf( )

เป็นคำสั่งที่ใช้ในการแสดงผลออกทางจอภาพ โดยมีรูปแบบการใช้งานดังนี้

printf(“ข้อความ หรือ control หรือ format string”,variable list);

control หรือ format string

เป็นส่วนที่ใส่ข้อความที่จะแสดงผล และส่วนควบคุมลักษณะการแสดงผล รวมทั้งบอกตำแหน่งที่ตัวแปรจะแสดงผล

variable list

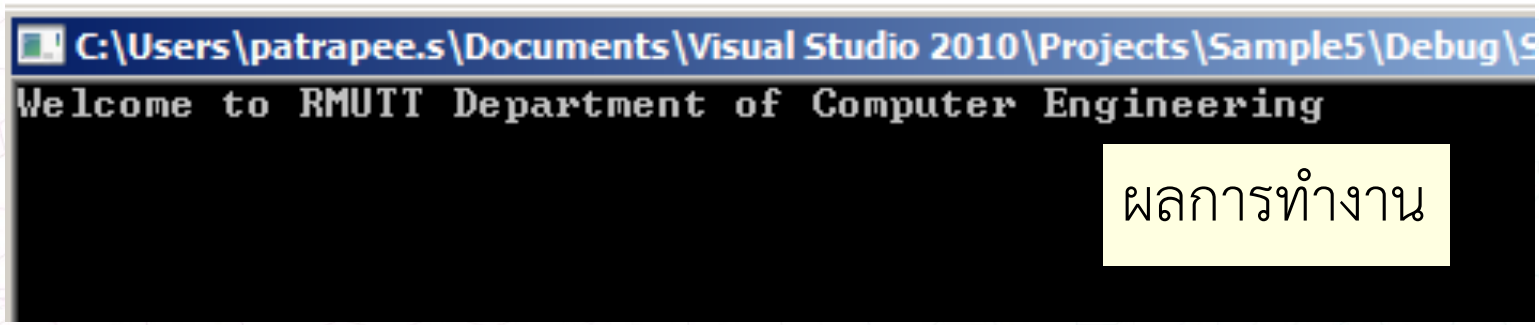
เป็นตัวแปรที่ต้องการจะแสดงผล ในกรณีที่ต้องการแสดงข้อความไม่จำเป็นต้องมีส่วนนี้

ตัวอย่างโปรแกรม การใช้คำสั่ง printf( )

ชุดคำสั่งที่เก็บอยู่ใน library ที่ชื่อว่า *stdio.h* (standard input-output)

```
#include <stdio.h>
int main()
{
    printf("Welcome to RMUTT");
    printf(" Department of Computer Engineering");
    return(0);
}
```

โปรแกรม



ผลการทำงาน

การใช้อักขระควบคุมการแสดงผล

คำสั่ง printf( ) สามารถควบคุมการแสดงผล ด้วยอักขระที่มี backslash นำหน้า

`\n`	ขึ้นบรรทัดใหม่
`\t`	เว้นระยะ 1 tab
`\a`	ส่งเสียงปี้บ
`\\`	แสดง \
`\"`	แสดง “

ตัวอย่างโปรแกรม การใช้อักขระควบคุมการแสดงผล

โปรแกรม

Backslash n ขึ้นบรรทัดใหม่

```
#include <stdio.h>

int main() {
  printf("Welcome to RMUTT\n");
  printf(" Department of Computer Engineering");
  return(0);
}
```

ผลการทำงาน

```
Welcome to RMUTT
 Department of Computer Engineering
```

รหัสควบคุมลักษณะ (Format String)

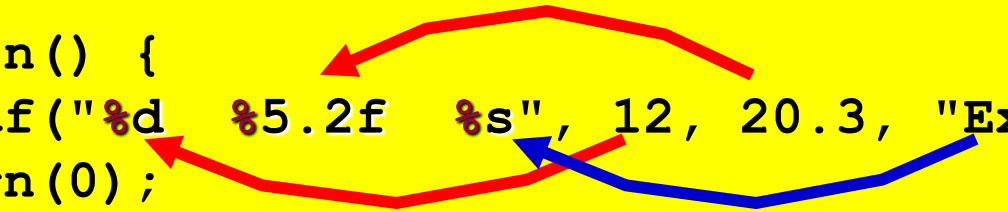
%d	พิมพ์จำนวนเต็มฐานสิบ
%u	พิมพ์เลขไม่มีเครื่องหมาย
%f	พิมพ์เลขทศนิยม
%e	พิมพ์ในรูปจำนวนจริงยกกำลัง
%c	พิมพ์ตัวอักขระตัวเดียว (Character)
%s	พิมพ์ชุดตัวอักขระ (String)
%%	พิมพ์เครื่องหมาย %
%o	พิมพ์เลขฐานแปด
%x	พิมพ์เลขฐานสิบหก

ตัวอย่างโปรแกรม การใช้รหัสควบคุมลักษณะ

โปรแกรม

```
#include <stdio.h>

int main() {
  printf("%d  %5.2f  %s", 12, 20.3, "Example");
  return(0);
}
```



ผลการทำงาน

```
12   20.30   Example
```

%d %5.2f %s คือ รหัสควบคุม

ตัวอย่างโปรแกรม การใช้รหัสควบคุมลักษณะ

โปรแกรม

```
#include <stdio.h>
int main()
{
  printf("%d  %c  %o  %x", 65, 65, 65, 65);
  return(0);
}
```

ผลการทำงาน

```
65   A   101   41
```

ผลจากการใช้รหัสควบคุมลักษณะด้วย %c จะได้ค่าผลลัพธ์เป็น A
ซึ่งเป็นอักขระลำดับที่ 65 ของตาราง ASCII

Standard ASCII Characters

Dec	Hex	Oct	Char	Description
64	40	100	@	Commercial at/At sign
65	41	101	A	Latin capital letter A
66	42	102	B	Latin capital letter B
67	43	103	C	Latin capital letter C
68	44	104	D	Latin capital letter D
69	45	105	E	Latin capital letter E
70	46	106	F	Latin capital letter F
71	47	107	G	Latin capital letter G
72	48	110	H	Latin capital letter H
73	49	111	I	Latin capital letter I

อ้างอิง https://www.techonthenet.com/ascii/chart.php

การจัดการหน้าจอด้วยรหัสควบคุมลักษณะ

ในกรณีที่ต้องการจัดการหน้าจอแสดงผลสามารถใช้ตัวเลขร่วมกันกับรหัสควบคุมได้ เช่น

%5d หมายถึง แสดงตัวเลขจำนวนเต็ม 5 หลักอย่างต่ำ

%5.2f หมายถึง แสดงตัวเลขจำนวนจำนวน 5 หลักอย่างต่ำ และทศนิยม 2 ตำแหน่ง

ค่า	%d	%5d
12	12	___12
123	123	__123
1234	1234	_1234
12345	12345	12345

ค่า	%f	%5.2f
1.2	1.200000	_1.20
1.234	1.234000	_1.23
12.345	12.345000	12.35
123.456	123.456000	123.46

การใช้ scanf( )

เป็นคำสั่งที่ใช้ในการรับค่า โดยมีรูปแบบการใช้งานดังนี้

```
scanf("format string", address list …);
```

format string (**รหัสควบคุมลักษณะ**)

เป็นส่วนที่ใช้ในการใส่รูปแบบของการรับข้อมูล

address list

เป็นตำแหน่งของตัวแปรที่ต้องการเก็บข้อมูล

ตัวอย่างโปรแกรม การใช้ scanf( )

โปรแกรม

```
#include <stdio.h>

int main() {
   int x ;
      รหัสควบคุมลักษณะ     ตัวแปรที่ต้องการเก็บข้อมูล
   scanf("%d",&x );
   printf("%d",x);
   return(0);
}
```


```
printf("%d", x);
```

```
scanf("%d",&x);
```

ตัวอย่างโปรแกรม การใช้ scanf( )

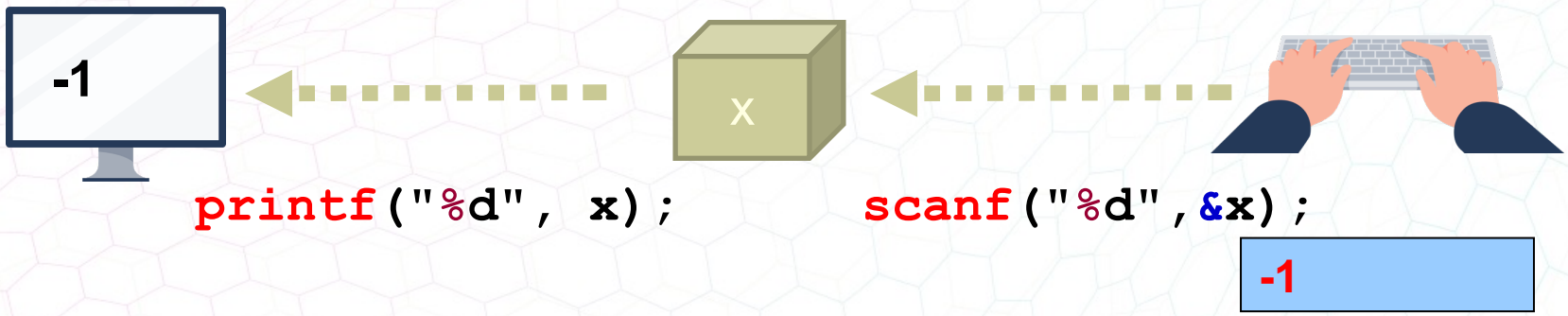
โปรแกรม

```
#include <stdio.h>

int main() {
   int x ;

   scanf("%d",&x);
   printf("%d",x);
   return(0);
}
```

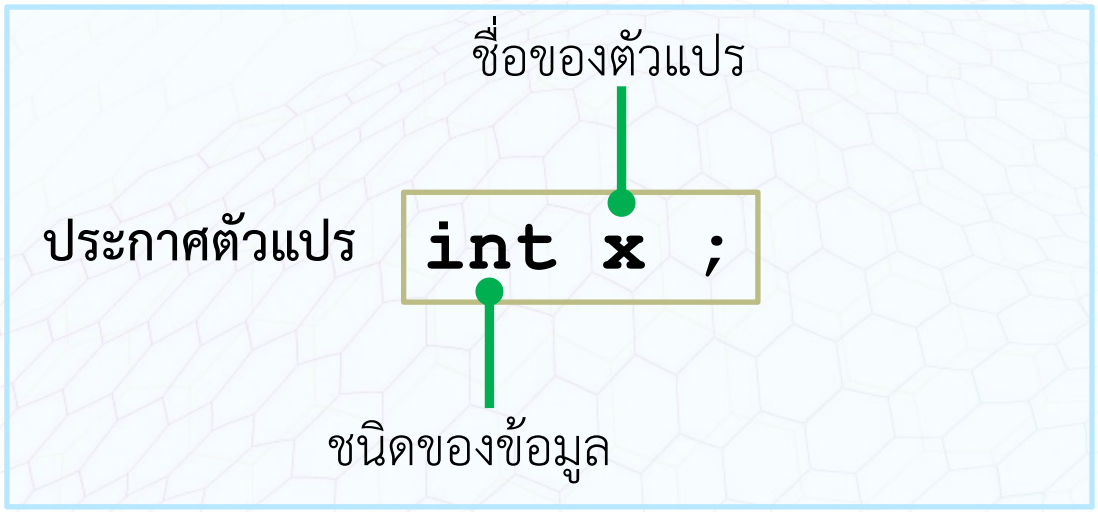
ผลการทำงาน



ตัวแปร variable

การประกาศตัวแปรสำหรับการเขียนโปรแกรม เปรียบเสมือนการสร้างกล่องเพื่อเก็บค่าลงในกล่องนั้นโดยที่กล่องแต่ละกล่องต้องมีชื่อที่แตกต่างกัน

ตัวอย่าง



สร้างกล่องที่มีขนาดสำหรับ

เก็บข้อมูลชนิด integer -> int

หลักการตั้งชื่อตัวแปร ค่าคงที่ และ ฟังก์ชัน

- ต้องขึ้นต้นด้วยตัวอักษรภาษาอังกฤษ (ตัวใหญ่หรือเล็กก็ได้) หรือขีดล่าง ‘_’
- ตามด้วยตัวอักษรภาษาอังกฤษ ตัวเลข หรือขีดล่าง (Underscore) ‘_’
- **ไม่มีช่องว่างหรือตัวอักษรพิเศษอื่นๆ** เช่น ‘!’, ‘@’, ‘#’, ‘$’, ‘%’, ‘^’ เป็นต้น
- ตัวพิมพ์ใหญ่และเล็กจะเป็นคนละตัวกันเช่น NAME, name, Name, NamE
- ห้ามซ้ำกับคำสงวน Reserved Words ของภาษา C เช่น char, do, const, break,...
- ห้ามตั้งชื่อซ้ำกับ Function ที่อยู่ใน Library ของภาษา C เช่น printf, scanf, ...

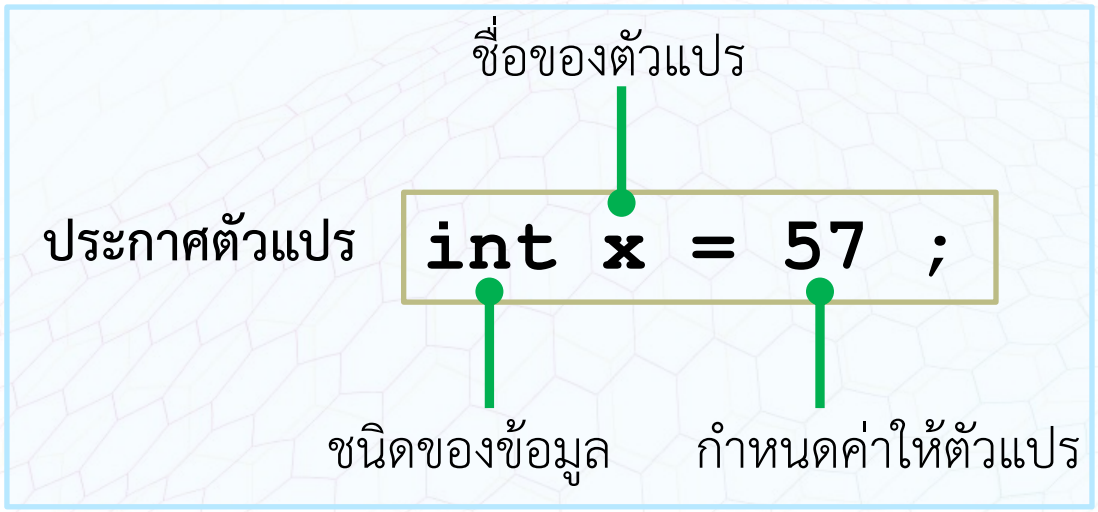
คำสงวน (Reserved Words) ของภาษา C

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while
asm	_cs	_ds	_es
_ss	cdecl	far	huge
interrupt	near	pascal	_export

ตัวแปร variable (2)

การประกาศตัวแปรสำหรับการเขียนโปรแกรม เปรียบเสมือนการสร้างกล่องเพื่อเก็บค่าลงในกล่องนั้นโดยที่กล่องแต่ละกล่องต้องมีชื่อที่แตกต่างกัน

ตัวอย่าง



สร้างกล่องที่มีขนาดสำหรับเก็บข้อมูลชนิด integer -> int

ตัวแปร variable (3)

การประกาศตัวแปรสำหรับการเขียนโปรแกรม เปรียบเสมือนการสร้างกล่องเพื่อเก็บค่าลงในกล่องนั้นโดยที่กล่องแต่ละกล่องต้องมีชื่อที่แตกต่างกัน

ตัวอย่าง

ประกาศตัวแปร

```
1  int x;
2  x = 57;
```

เก็บข้อมูลชนิด integer -> int

ตัวแปร variable (4)

ขั้นตอนวิธีการแก้ปัญหาสำหรับการรับข้อมูลตัวเลข 2 ค่า แล้วทำการสลับค่าตัวเลขระหว่างกัน

ประกาศตัวแปร

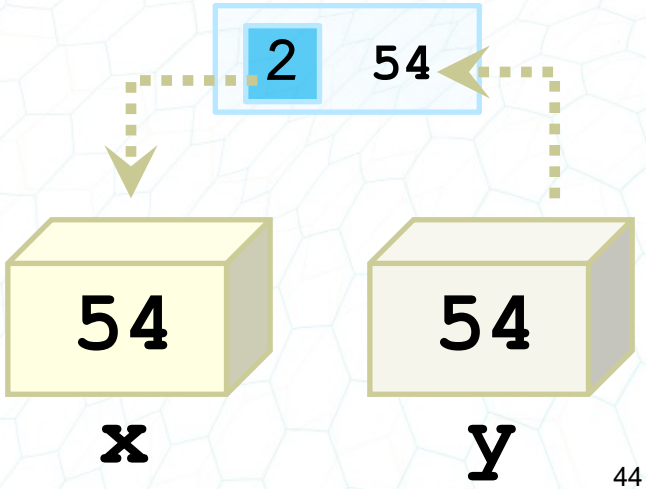
1

```
int x,y;
x = 57;
y = 54;
```

กำหนดค่าตัวแปร

2

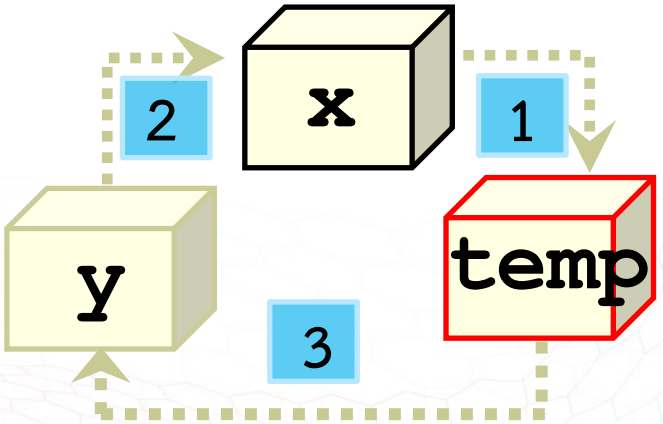
```
x = y;
```

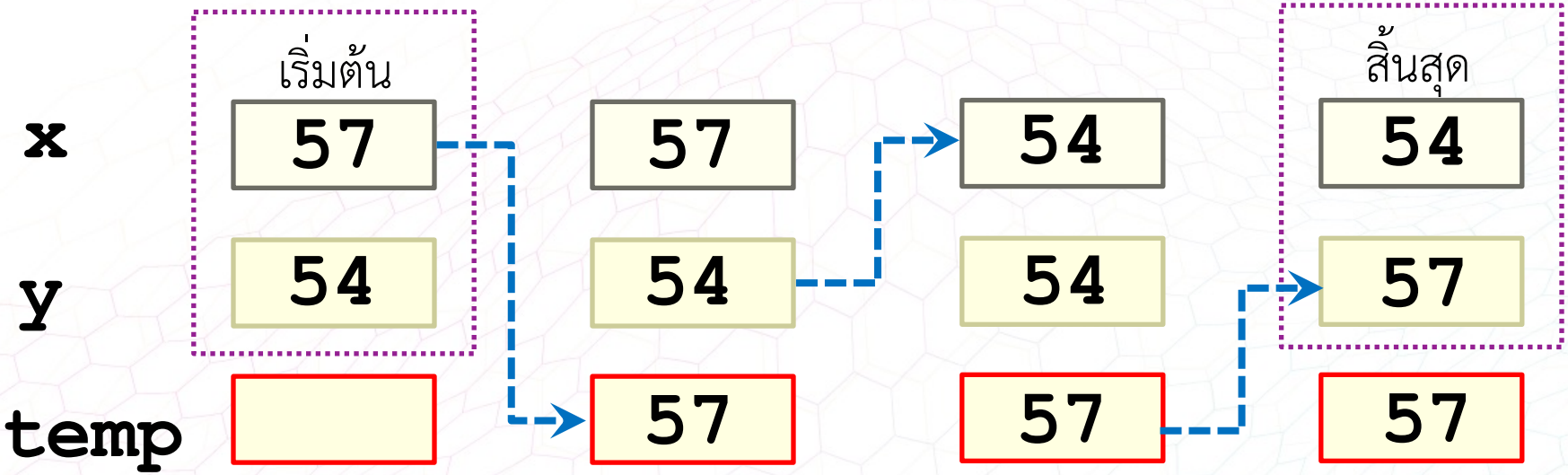


หมายเหตุ ที่เก็บข้อมูลทางคอมพิวเตอร์ (ตัวแปร) เก็บค่าได้เพียงค่าเดียวเท่านั้น

ตัวแปร variable (4)

ขั้นตอนวิธีการแก้ปัญหาสำหรับการรับข้อมูลตัวเลข 2 ค่า แล้วทำการสลับค่าตัวเลขระหว่างกัน






```
1 temp = x;
```

```
2 x = y;
```

```
3 y = temp;
```

หมายเหตุ ที่เก็บข้อมูลทางคอมพิวเตอร์ (ตัวแปร) เก็บค่าได้เพียงค่าเดียวเท่านั้น

จบบทที่ 3