# บทที่ 3-1

## โครงสร้างภาษาซีเบื้องต้น

## Basic C Programming Language

# ประวัติภาษาซี

- ภาษาซีพัฒนาขึ้นมาในปี 1970 โดย Dennis Ritchie แห่ง Bell Telephone Laboratories, Inc. (ปัจจุบันคือ AT&T Bell Laboratories) ซึ่งภาษาซีนั้นมี ต้นกำเนิดมาจากภาษา 2 ภาษา คือ ภาษา BCPL และ ภาษา B
- ภาษาซีนั้นถูกใช้งานอยู่ เพียงใน Bell Laboratories จนกระทั่งปี 1978 Brian Kernighan และ Ritchie นั้นเป็นที่รู้จักกันในชื่อของ "K&R C"

# ประวัติภาษาซี

- หลังจากที่ตีพิมพ์ข้อกำหนดของ K&R นักคอมพิวเตอร์มืออาชีพรู้สึก ประทับใจกับคุณสมบัติที่น่าสนใจของภาษาซี และเริ่มส่งเสริมการใช้งานภาษาซีมากขึ้น
- ในกลางปี 1980 ภาษาซีก็กลายเป็นภาษาที่ได้รับความนิยม โดยทั่วไป มีการพัฒนาตัวแปลโปรแกรม และตัวแปลคำสั่งภาษาซีจำนวนมาก สำหรับคอมพิวเตอร์ทุกขนาด และภาษาซีก็ถูกนำมาไปใช้สำหรับพัฒนา โปรแกรมเชิงพาณิชย์เป็นจำนวนมาก ยิ่งไปกว่านั้นโปรแกรมเชิงพาณิชย์ที่เคยพัฒนาขึ้นมาโดยภาษาอื่น ก็ถูกเขียนขึ้นใหม่โดยใช้ภาษาซี เนื่องจากความ ต้องการใช้ความได้เปรียบทางด้านประสิทธิภาพ และความสามารถในการเคลื่อนย้ายได้ของภาษาซี

# แนะนำภาษาซี

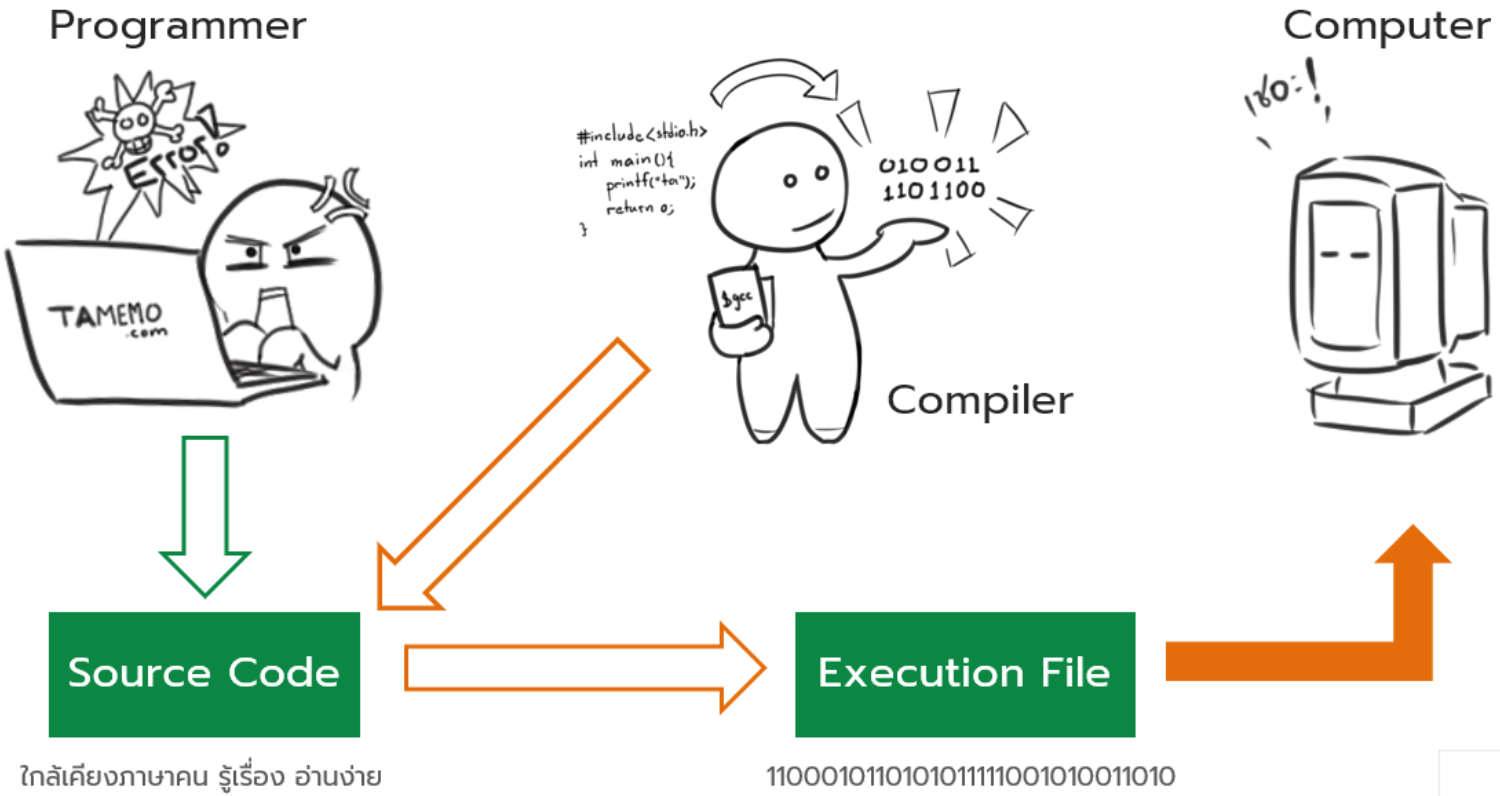
- ภาษาซีเป็นภาษาที่เป็นโครงสร้างและใช้ได้กับงานทั่วไป คำสั่งของภาษาซี จะประกอบด้วยพจน์ (term) ซึ่งจะมีลักษณะเหมือนกับนิพจน์ทางพีชคณิต และมีส่วนขยายเป็นคำหลัก (keyword) ในภาษาอังกฤษ เช่น if, else, for, do และ while
- ดังนั้นภาษาซีเป็นภาษาระดับสูง ภาษา อื่น ๆ เช่น ปาสคาล และ ฟอร์เทรน 77 มีลักษณะเป็นโครงสร้างเช่นกัน แต่ภาษาซีก็มี คุณสมบัติพิเศษเพิ่มขึ้น นั่นคือสามารถใช้งานในระดับต่ำ (low-level) ได้ ดังนั้นจึงเปรียบเหมือนสะพานเชื่อมภาษาเครื่องเข้ากับภาษาระดับสูง จากจุดนี้
  - ทำให้ภาษาซีสามารถใช้กับงานด้านโปรแกรมระบบ (system programming) เช่น เขียนโปรแกรมระบบปฏิบัติการ (operating system) หรือใช้กับงานทั่ว ๆ ไป เช่น เขียนโปรแกรมแก้ปัญหาทางคณิตศาสตร์ ที่ซับซ้อน หรือเขียนโปรแกรมเพื่อออกใบเสร็จให้กับลูกค้า เป็นต้น

# แนะนำภาษาซี

- คุณสมบัติที่สำคัญอีกประการหนึ่งของภาษาซี ก็คือ โปรแกรมภาษาซีสามารถย้ายไปทำงานในเครื่องอื่นได้ง่ายกว่าภาษาระดับสูงอื่น ๆ ที่เป็นเช่นนี้ เพราะภาษาซีได้แยกส่วนที่ขึ้นอยู่กับเครื่องคอมพิวเตอร์ไปเป็นไลบรารี ฟังก์ชัน ดังนั้น โปรแกรมภาษาซีทุก ๆ โปรแกรม ก็จะทำงานโดยเรียกฟังก์ชัน จากไลบรารี่ฟังก์ชันมาตราฐาน และมีวิธีการเขียนใช้งานแบบเดียวกัน ดังนั้น โปรแกรมภาษาซีทั้งหมดจึงสามารถนำมาใช้งานบนเครื่องคอมพิวเตอร์ที่ แตกต่างกันได้ โดยแก้ไขโปรแกรมเพียงเล็กน้อย หรืออาจจะไม่ต้องแก้ไข

# กระบวนการแปลภาษาคอมพิวเตอร์

คอมพิวเตอร์ทำงานได้ต้องมีการประมวลผลภาษาเครื่องเท่านั้น เพื่อให้การเขียนโปรแกรมได้ง่ายขึ้นจึงพัฒนาเป็นภาษาคอมพิวเตอร์ จึงต้องมีโปรแกรมแปลภาษาคอมพิวเตอร์เป็นภาษาเครื่อง

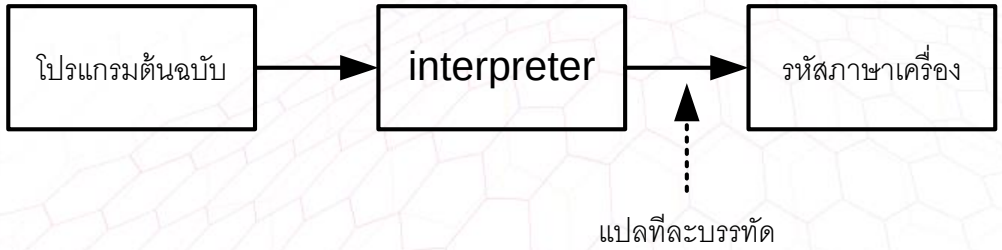


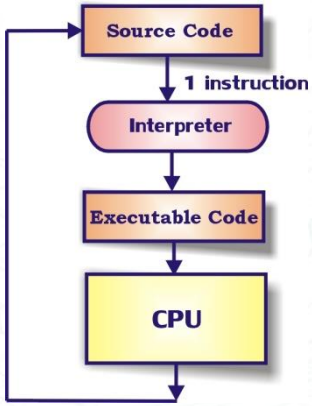



ที่มา: https://www.tamemo.com/post/141/learn-intro-compiler-interpreter/

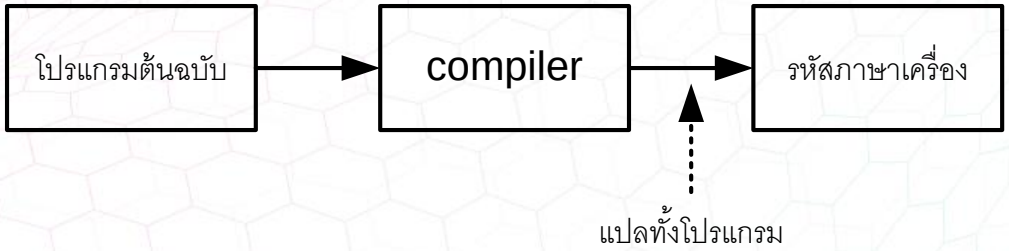
# กระบวนการแปลภาษาคอมพิวเตอร์

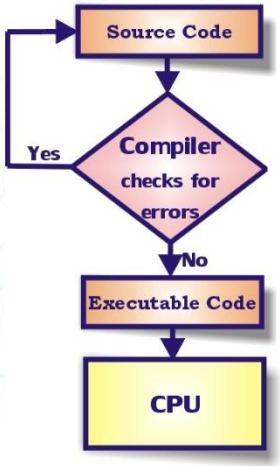
- โปรแกรมแปลภาษาเป็นภาษาเครื่อง มี 2 ประเภท
  - อินเทอร์พรีเตอร์ (Interpreter)




  - คอมไพเลอร์ (Compiler)



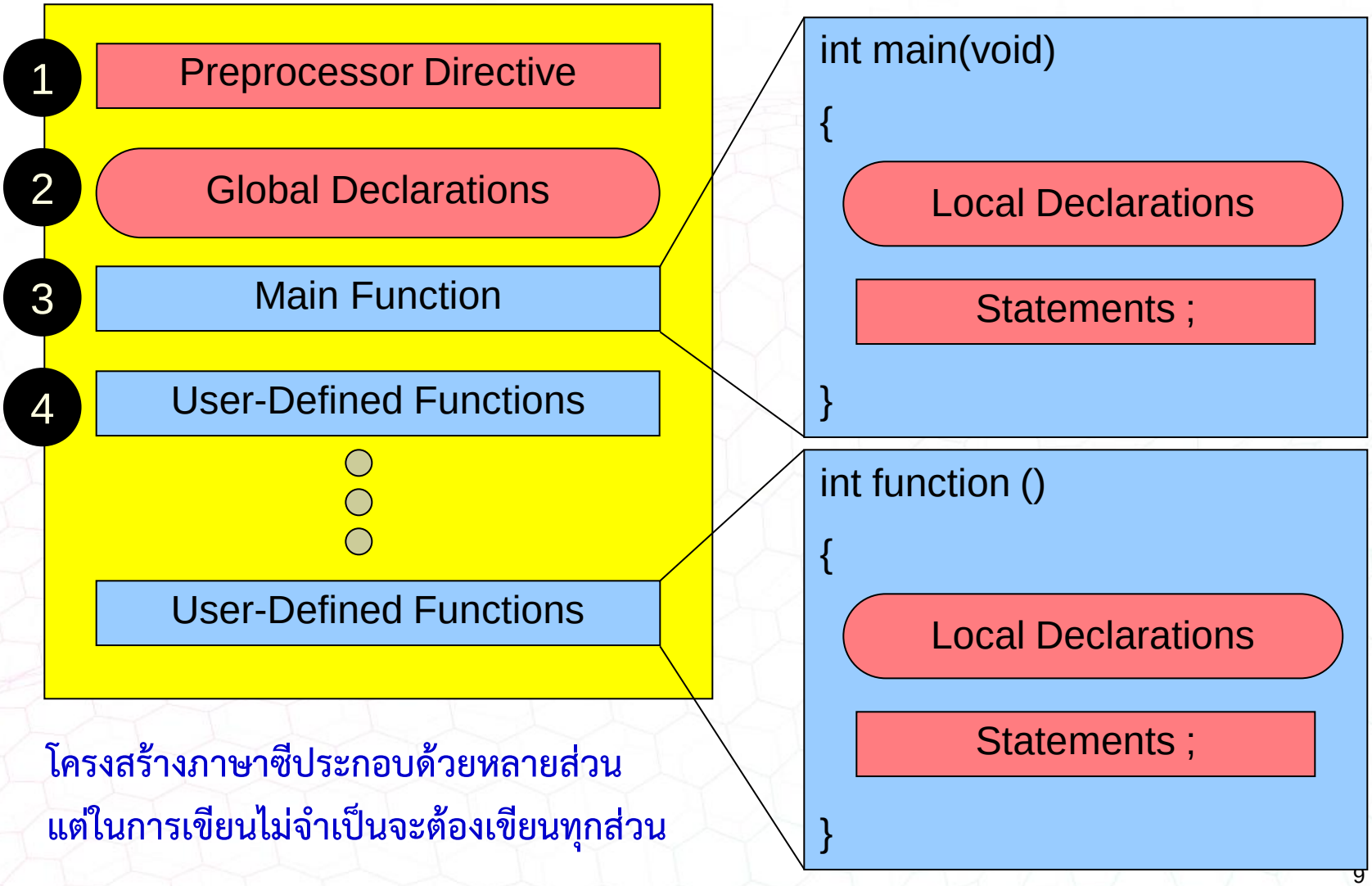




ขั้นตอนการแปลภาษาโปรแกรม

# กระบวนการแปลภาษาคอมพิวเตอร์

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

# โครงสร้างโปรแกรมภาษาซี




โครงสร้างภาษาซีประกอบด้วยหลายส่วน
แต่ในการเขียนไม่จำเป็นจะต้องเขียนทุกส่วน

# โครงสร้างโปรแกรมภาษาซี

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

Header, Body

# Preprocessor Directive

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

| | | | |
|---|---|---|---|
| **#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 ‘}’
- ระหว่างปีกกาจะประกอบไปด้วยคำสั่ง(Statement) ต่างๆ ที่จะให้โปรแกรมทำงาน
- แต่ละคำสั่งจะต้องจบด้วยเซมิโคลอน ‘;’ (Semicolon)
- ต้องมี return(); เสมอ และต้องใส่เลขจำนวนเต็ม เช่น 0 = Success , 1 = Failure

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

# Preprocessor Directive

ตัวอย่าง

```
1   #include <stdio.h>
2   int feet,inches;
3   int main(void)
4   {
5     feet = 6;
6     inches = feet * 12;
7     printf("Height in inches is %d",inches);
8     return(0);
9   }
10
11
```

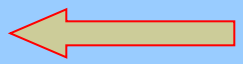
ผลการทำงาน

```
Height in inches is 72
```

# ส่วนประกาศ (Global Declarations)

ตัวอย่าง

```
1  #include <stdio.h>
2  int feet,inches;
3  int main(void)
4  {
5
6    feet = 6;
7    inches = feet * 12;
8    printf("Height in inches is %d",inches);
9    return(0);
   }
```



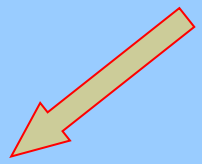
ผลการทำงาน

```
Height in inches is 72
```

# ฟังก์ชันหลักของโปรแกรม (Main Function)

ตัวอย่าง

```
1  #include <stdio.h>
2  int feet,inches;
3  int main(void)
4  {
5
6    feet = 6;
7    inches = feet * 12;
8    printf("Height in inches is %d",inches);
9    return(0);
   }
```



ผลการทำงาน

```
Height in inches is 72
```

# การสร้างฟังก์ชันใช้งานเอง (User-Defined Function)

ตัวอย่าง

```
1   #include <stdio.h>
2   int FtoI(int);
3   int feet,inches;
4   int main(void)
5   {
6     feet = 6;
7     inches = FtoI(feet);
8     printf("Height in inches is %d",inches);
9     return(0);
10  }
11  int FtoI(int f)
12  {
13    return f*12;
14  }
```

ผลการทำงาน

```
Height in inches is 72
```

# 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 |

# คำอธิบายของโปรแกรม (Program Comments)

- ใช้เขียนส่วนอธิบายโปรแกรม (คอมเมนต์)
- ช่วยให้ผู้ศึกษาโปรแกรมภายหลังเข้าใจการทำงานของโปรแกรม
- ส่วนของคำอธิบายจะถูกข้ามเมื่อคอมไพล์โปรแกรม

การเขียนส่วนอธิบายโปรแกรม (comments)ทำได้ 2 วิธีคือ

// สำหรับคำอธิบายไปจนถึงท้ายบรรทัด

และ

/* คำอธิบาย */ ลักษณะการใช้เหมือนวงเล็บนั้นเอง

# คำอธิบายของโปรแกรม (Program Comments)

ตัวอย่าง

```
#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
```

# คำสั่ง printf( )

เป็นคำสั่งที่ใช้ในการแสดงผลออกทางจอภาพ โดยมีรูปแบบการใช้งานดังนี้

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

**control หรือ format string**

เป็นส่วนที่ใส่ข้อความที่จะแสดงผล และส่วนควบคุมลักษณะการแสดงผล รวมทั้งบอกตำแหน่งที่ตัวแปรจะแสดงผล

**variable list**

เป็นตัวแปรที่ต้องการจะแสดงผล ในกรณีที่ต้องการแสดงข้อความ ไม่จำเป็นต้องมีส่วนนี้

# ตัวอย่างโปรแกรม การใช้คำสั่ง printf( )

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

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

โปรแกรม

```
C:\Users\patrapee.s\Documents\Visual Studio 2010\Projects\Sample5\Debug\
Welcome to RMUTI_
```

ผลการทำงาน

# 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

```
C:\Users\patrapee.s\Documents\Visual Studio 2010\Projects\Sample5\Debug\S
Welcome to RMUTT Department of Computer Engineering
```

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)

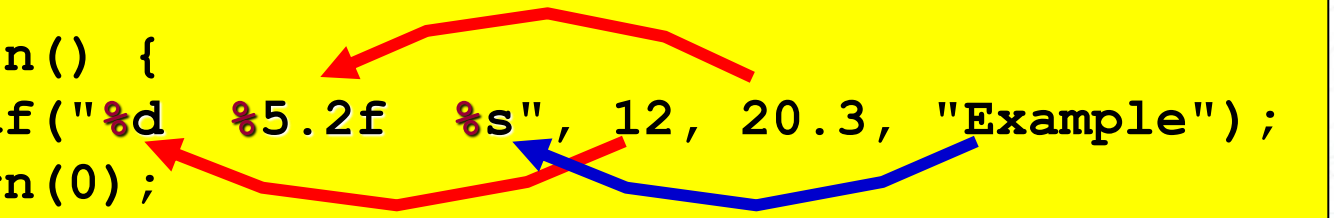
| | |
|---|---|
| **%d** | พิมพ์จำนวนเต็มฐานสิบ |
| **%u** | พิมพ์เลขไม่มีเครื่องหมาย |
| **%f** | พิมพ์เลขทศนิยม |
| **%e** | พิมพ์ในรูปจำนวนจริงยกกำลัง |
| **%c** | พิมพ์ตัวอักขระตัวเดียว (Character) |
| **%s** | พิมพ์ชุดตัวอักขระ (String) |
| **%%** | พิมพ์เครื่องหมาย % |
| **%o** | พิมพ์เลขฐานแปด |
| **%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()
{
  int x ;
  x=65 ;
  printf("%d  %c  %o  %x", x, x, x, x);
  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 |

# คำถามเกี่ยวกับ printf( )

จากส่วนของโปรแกรม

```
yards = 8;
feet = yards * 3;
printf("%d yards is", yards);
printf("%d feets \n", feet);
```

ผลการทำงาน คือ ?

```
8 yards is24 feets
_
```

# คำถามเกี่ยวกับ printf( )

จากส่วนของโปรแกรม

```
yards = 8;
feet = yards * 3;
printf("%d yards is \n", yards);
printf("%d feets", feet);
```

ผลการทำงาน คือ ?

```
8 yards is
24 feets
```

# การใช้ scanf( )

เป็นคำสั่งที่ใช้ในการรับค่า โดยมีรูปแบบการใช้งานดังนี้

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

**format string**

เป็นส่วนที่ใช้ในการใส่รูปแบบของการรับข้อมูล

**address list**

เป็นตำแหน่งของตัวแปรที่ต้องการเก็บข้อมูล

# ตัวอย่างโปรแกรม การใช้ scanf( )

โปรแกรม

```
#include <stdio.h>

int main() {
   int x ;

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

ผลการทำงาน

```
2563
2563
```

```
-1
-1
```

# ตัวอย่างโปรแกรม ที่มีการโต้ตอบกับผู้ใช้

```
1   #include <stdio.h>
2
3   int main() {
4     float b,h,area ;
5     printf("Input Base :> ");
6     scanf("%f",&b);
7     printf("Input Height :> ");
8     scanf("%f",&h);
9     area = 0.5*b*h ;
10    printf("Area of triangle is %5.2f",area);
11    return(0);
12  }
```

```
Input Base :> 12.0
Input Height :> 6.0
Area of triangle is 36.00
```

```
Input Base :> 3.2
Input Height :> 1.2
Area of triangle is 1.92
```

# จบบทที่ 3-1

## โครงสร้างภาษาซีเบื้องต้น

## Basic C Programming Language