



โครงสร้าง โปรแกรมภาษา C

หัวข้อ

- โปรแกรมภาษา
- การ Compile & Run โปรแกรม
- โครงสร้างโปรแกรมภาษา C
- คำสั่งแสดงผลลัพธ์ (Output)
- คำสั่งรับข้อมูล (Input)

โปรแกรมภาษา

- โปรแกรมภาษา ใช้ในการพัฒนาโปรแกรมสำหรับงานเฉพาะตามที่ใช้ต้องการ
- ประเภทของโปรแกรมภาษา
 1. ภาษาระดับต่ำ (Low-Level Language)
เช่น ภาษาเครื่อง (Machine language)
 2. ภาษาระดับกลาง (Middle-Level Language)
เช่น ภาษาแอสเซมบลี (Assembly Language)
 3. ภาษาระดับสูง (High-Level Language)
เช่น Pascal, Fortran, C, JAVA, ...

ภาษาเครื่อง

- ภาษาเครื่อง (Machine Language)
เป็นภาษาที่คอมพิวเตอร์เข้าใจ ซึ่งเขียนเป็นรหัสเลขฐาน 2 (0/1) และมีความเกี่ยวข้องกับอุปกรณ์ภายในเครื่องคอมพิวเตอร์
- แต่ยุ่งยากต่อการพัฒนาโดยมนุษย์เรา
- ตัวอย่างเช่น Object Code

```
00111100001111010000001000000011
10000011000000111000001000000010
00001010000010100100000001000010
10000100000001001010001000100011
00001100000011010000111000001111
10001001000010011000100000001000
00010100000101010001011000010111
00011000000110010001101000011011
00011100000111010001111000011111
00100100001001010010011000100111
00111100001111010011111000111111
00101000001010010010101000101011
00101100001011010010111000101111
00110000001100010011001000110011
10011011000110111001101000011010
00111000001110010011101000111011
```

ภาษาแอสเซมบลี

- ภาษาแอสเซมบลี (Assembly Language)
เป็นภาษาที่เขียนโดยใช้คำสั่งเป็นคำเฉพาะในภาษาอังกฤษที่
มนุษย์เข้าใจ แทนการใช้รหัสเลขฐาน 2
 - แต่ออกแบบมาเฉพาะสำหรับคอมพิวเตอร์แต่ละรุ่น
 - ผู้เขียนโปรแกรมยังต้องทราบข้อมูลที่เกี่ยวข้องกับอุปกรณ์ของเครื่อง
คอมพิวเตอร์ (ยังยากต่อการพัฒนา)
 - และต้องใช้ แอสเซมเบอ์ (Assembler) ในการแปล
ภาษาแอสเซมบลี ให้เป็นภาษาเครื่อง

;CLEAR SCREEN USING BIOS

CLR: MOV AX, 0600H

MOV BH, 30

MOV CX, 0000

MOV DX, 184FH

INT 10H

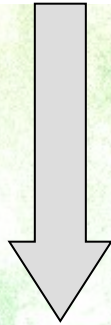
;INPUT A STRING

KEY: MOV AH, 0AH

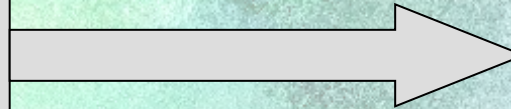
LEA DX, BUFFER

INT 21H

RET



Assembler



```
00111100001111010000001000000011
10000011000000111000001000000010
00001010000010100100000001000010
10000100000001001010001000100011
00001100000011010000111000001111
10001001000010011000100000001000
00010100000101010001011000010111
00011000000110010001101000011011
00011100000111010001111000011111
00100100001001010010011000100111
00111100001111010011111000111111
00101000001010010010101000101011
00101100001011010010111000101111
00110000001100010011001000110011
10011011000110111001101000011010
00111000001110010011101000111011
```

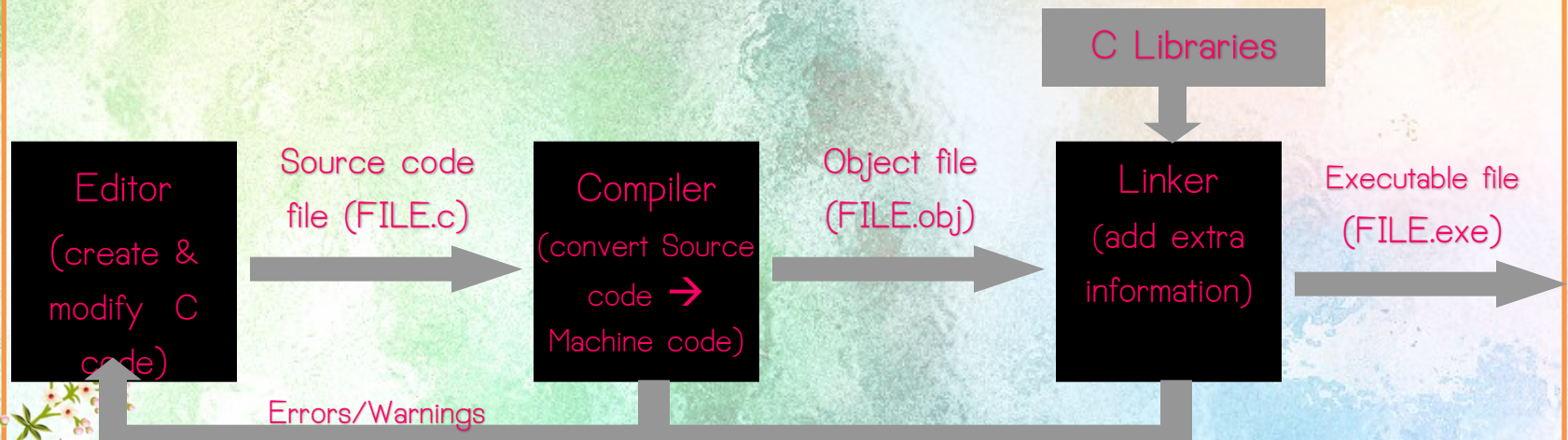
ภาษาระดับสูง

- ภาษาระดับสูง (High-Level Languages) เช่น C, JAVA,
เป็นภาษาที่พัฒนาได้ง่าย เพราะใช้คำสั่งเป็นภาษาอังกฤษ
—ใช้ คอมไพเลอร์ (Compiler) หรือ
Interpreter ในการแปลภาษาระดับสูงให้เป็น
ภาษาเครื่อง
 - Compiler แปลทั้งโปรแกรม (เช่น Pascal, C, ...)
 - Interpreter แปลทีละบรรทัด
- ตัวอย่าง โปรแกรมภาษา C

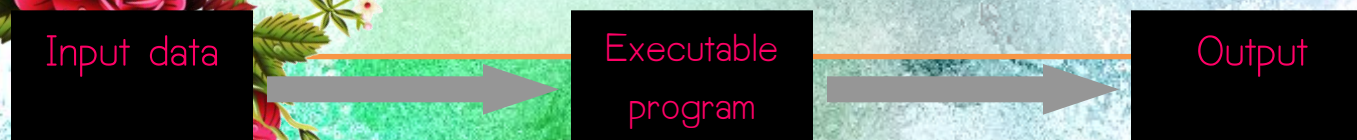
```
#include <stdio.h>
void main()
{
    int feet, inches;
    printf("Enter number (in feet) ");
    scanf("%d", &feet);
    inches = feet*12;
    printf("= %d inches\n", inches);
}
```

การ Compile

- ขั้นตอนการแปล Source code (file) ของโปรแกรมภาษา C ให้เป็น Machine code (Object file)

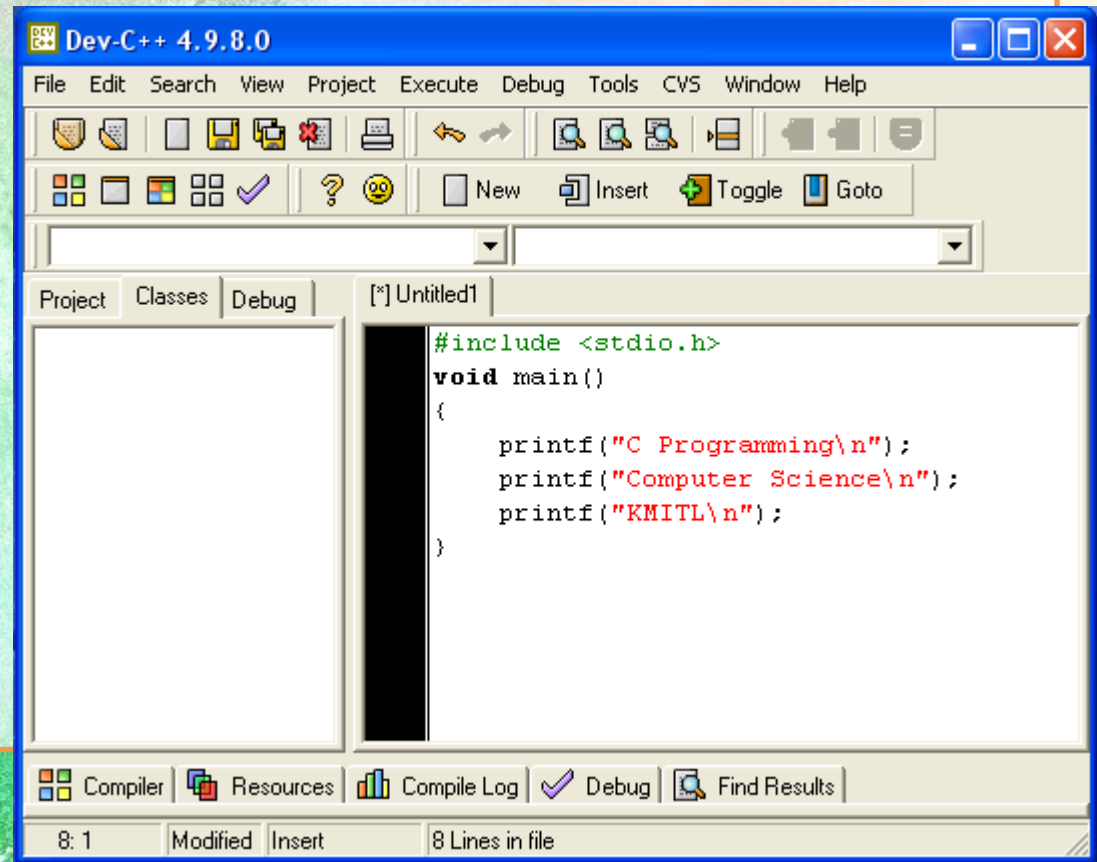


ขั้นตอนการประมวลผลโปรแกรมภาษา C



Compile & Run

- โปรแกรมภาษา C จะอยู่ในรูปแบบของฟังก์ชัน ซึ่งมีอย่างน้อยหนึ่งฟังก์ชัน (คือ main)
- ตัวอย่างการ Edit โปรแกรมภาษา C



โครงสร้างโปรแกรม C

- Preprocessor Directive (ข้อความสั่งตัวประมวลผลก่อน)
- Main Function (ฟังก์ชันหลัก)

```
main()                                /* main function */
{                                     /* Begin (เริ่มต้น) */
    variable declaration;           /* ประกาศตัวแปรที่เก็บข้อมูล */
    statements;                      /* คำสั่งประมวลผล */
}                                     /* End (จบ) */
```

- ตัวอย่างเช่น ถ้าต้องการพิมพ์ C Programming
- C Program คือ

```
#include <stdio.h> /* Preprocessor directive */
void main()
{
    printf("C Programming\n");
}
```

\n หมายถึง การย้าย cursor ไปบรรทัดใหม่ (newline)

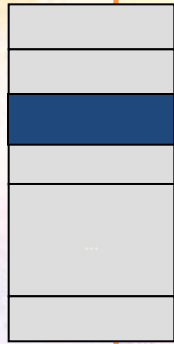
คำสั่งในภาษา C

- ภาษา C ประกอบด้วยคำสั่งพื้นฐาน 4 ชนิด คือ
 1. คำสั่งรับและแสดงผล (Input / Output Statements)
 2. คำสั่งคำนวณนิพจน์ (Expression Statements)
 3. คำสั่งทดสอบเงื่อนไข (Conditioning Statements)
 4. คำสั่งทำงานซ้ำ (Looping Statements)
- คำสั่งแต่ละชนิด อาจต้องใช้ตัวแปร (Variables) ในการอ้างอิงข้อมูลที่เก็บอยู่ใน Memory ในบทนี้จะกล่าวถึงตัวแปร 2 ชนิด คือ
 - ตัวแปร Integer เช่น int x, y
 - ตัวแปร Floating-Point เช่น float area

คำสั่งแสดงผล

- **printf** (เป็นฟังก์ชันมาตรฐานของ C ใน stdio)
 - ประกาศ stdio ในส่วน Preprocessor Directive
 - ก่อนเรียกใช้ฟังก์ชัน printf ใน main
- รูปแบบ **printf**("control string", variable,...);
 - variable เป็นตัวแปรใช้เก็บค่า (ที่เปลี่ยนแปลงได้) ใน Memory ในขณะประมวลผล
- **control string** ประกอบด้วย
 - ข้อความอธิบาย เช่น printf("C Programming");
 - %format เช่น %d, %f, %c, %s
 - อักษรควบคุม เช่น \n (new line), ...

Memory

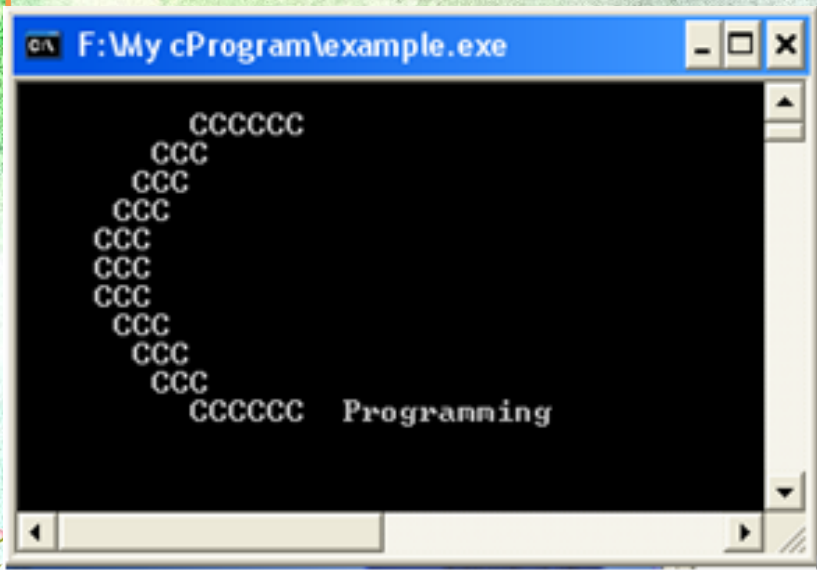


%d สำหรับ integer หรือ decimal

%f สำหรับ real หรือ floating point

ตัวอย่าง 2.1

- เขียนโปรแกรม พิมพ์คำว่า C Programming



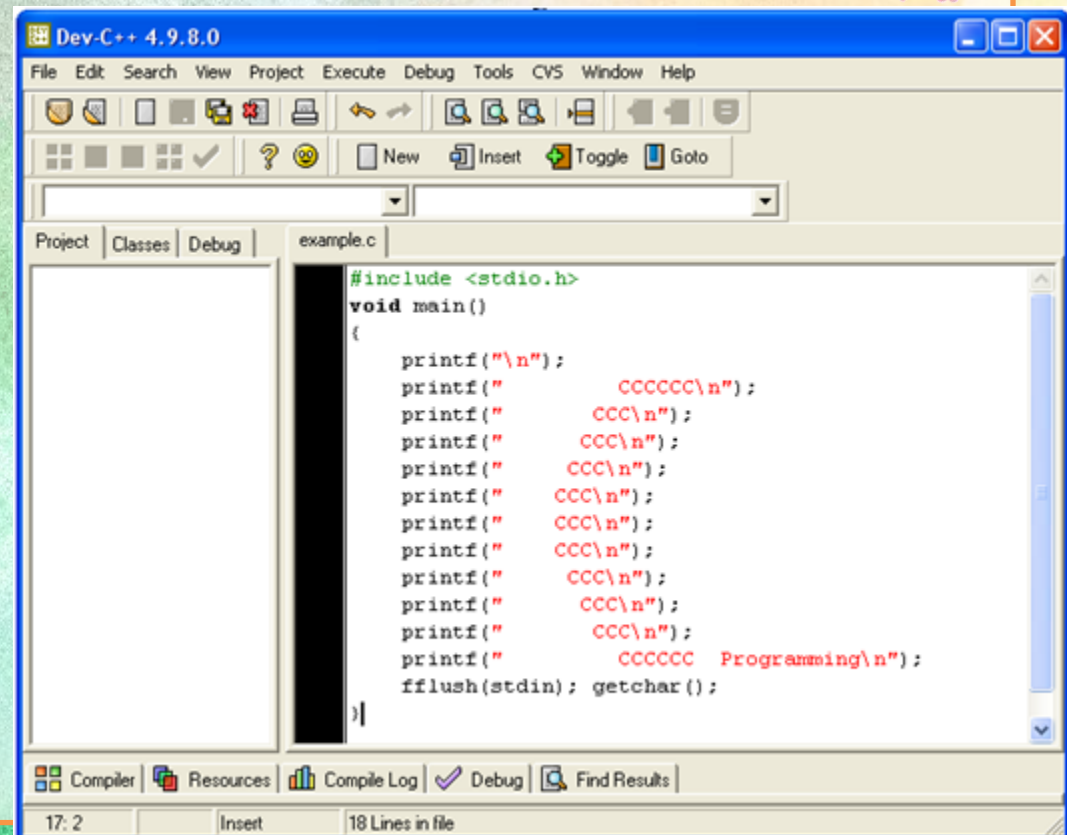
```
CCCCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCCCC Programming
```

```
CCCCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCC  
CCCCC Programming
```

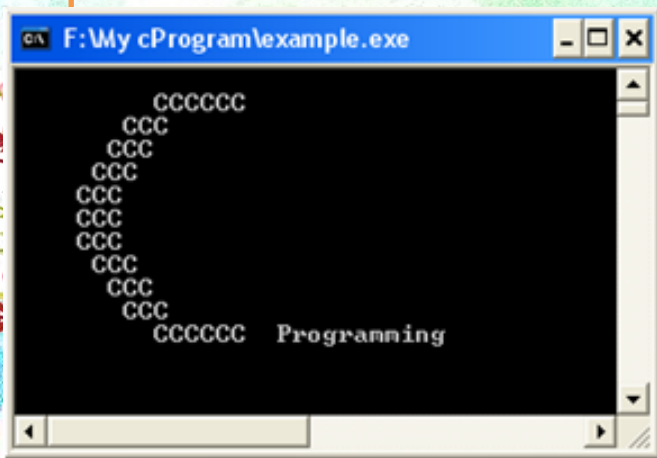
ตัวอย่าง 2.1

- เขียนโปรแกรม พิมพ์คำว่า C Programming

```
      CCCCCC
     CCC
    CCC
   CCC
  CCC
 CCC
CCC
 CCC
  CCC
   CCC
    CCCCCC Programming
```



```
#include <stdio.h>
void main()
{
    printf("\n");
    printf("      CCCCCC\n");
    printf("     CCC\n");
    printf("    CCC\n");
    printf("   CCC\n");
    printf("  CCC\n");
    printf(" CCC\n");
    printf("CCC\n");
    printf(" CCC\n");
    printf("  CCC\n");
    printf("   CCC\n");
    printf("    CCCCCC Programming\n");
    fflush(stdin); getchar();
}
```



คำสั่งรับข้อมูล

- **scanf** (เป็นฟังก์ชันมาตรฐานใน `stdio.h`)
- รูปแบบ `scanf("%format,...", &variable,...);`
 - **&variable** หมายถึง **ตำแหน่งที่อยู่** (Address) ของตัวแปร **variable** ที่เก็บในหน่วยความจำ

● **%format** เช่น **%d, %f, %c, %s**

● ตัวอย่างเช่น `scanf("%d", &X);`

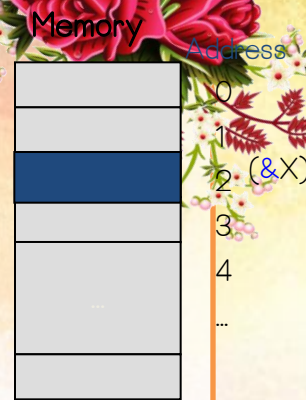
%d สำหรับ integer หรือ decimal

%f สำหรับ real หรือ floating point

รับข้อมูลชนิด integer จากคีย์บอร์ด

แล้วนำไปเก็บไว้ใน ตัวแปร **X** ที่ตำแหน่ง **&X**

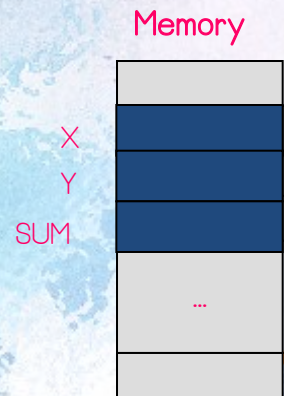
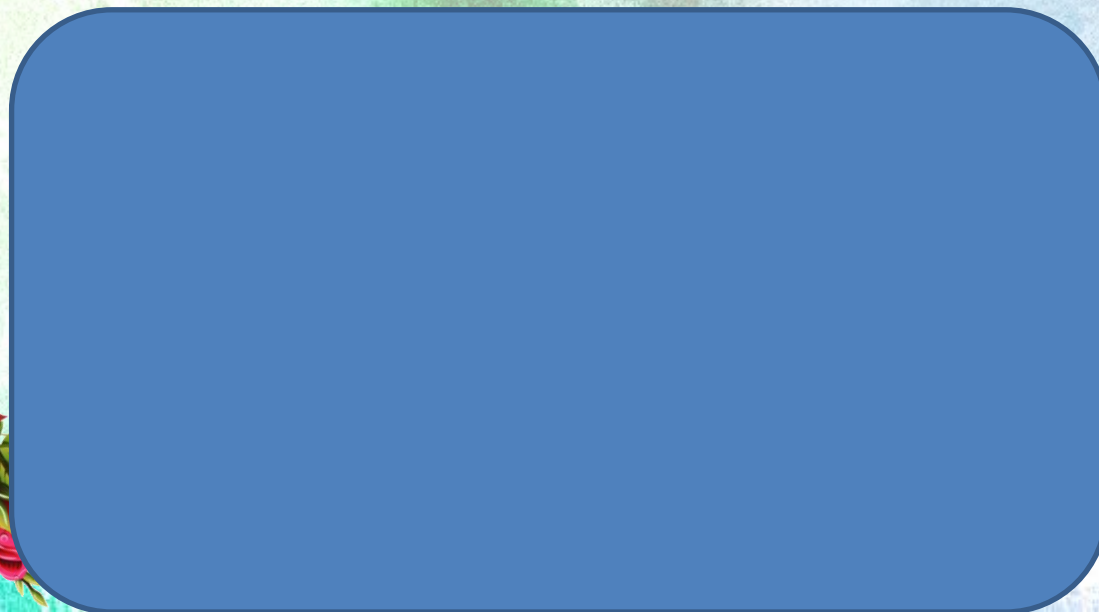
variable
(X)



%format

- **%format** ระบุชนิดของตัวแปร ในคำสั่ง scanf, printf

ชนิดข้อมูล	ชนิดตัวแปร	%format
จำนวนเต็ม (integer)	int	%d
จำนวนจริง (real)	float	%f
ตัวอักขระ (char)	char	%s



ตัวอย่าง 2.2

- เขียนโปรแกรม แสดงการบวกค่าเลข 2 จำนวน (X, Y) และแสดง ผลบวก (Sum)

```
#include <stdio.h>
```

```
main()
```

```
{   int  X, Y, SUM;
```

```
    printf("Enter X: "); scanf("%d", &X);
```

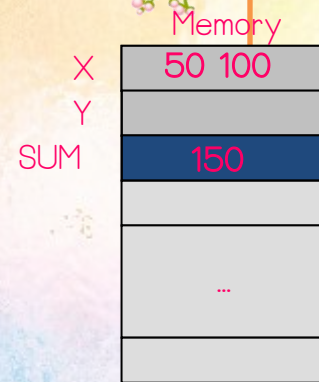
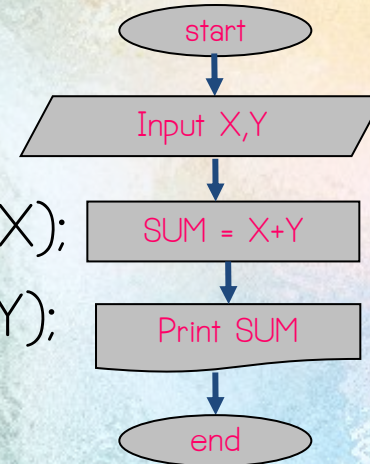
```
    printf("Enter Y: "); scanf("%d", &Y);
```

```
    SUM = X + Y;
```

```
    printf("Sum = %d\n", SUM);
```

```
}
```

- printf หมายถึง ฟังก์ชันแสดงผลลัพธ์ (Output) ในภาษา C
- scanf หมายถึง ฟังก์ชันรับข้อมูล (Input) ในภาษา C
- %d หมายถึง ชนิดของข้อมูลแบบ Integer (หรือ Decimal)



ตัวอย่าง

Input Number : 30

Input Number : 20

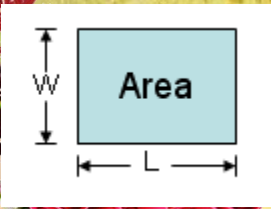
Sum = 50

Minus = 10

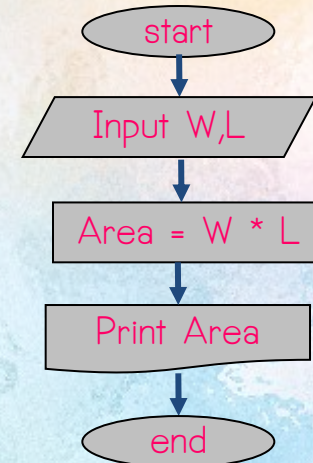
Multiple = 600

Divide = 1

ตัวอย่าง 2.3



- เขียนโปรแกรมคำนวณพื้นที่ของสี่เหลี่ยม (Area) เมื่อ $\text{Area} = \text{Width} \times \text{Length}$ (กว้าง x ยาว) และแสดงผลจากการคำนวณ



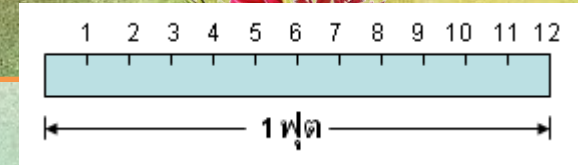
ผลลัพธ์

Enter width: _ 5

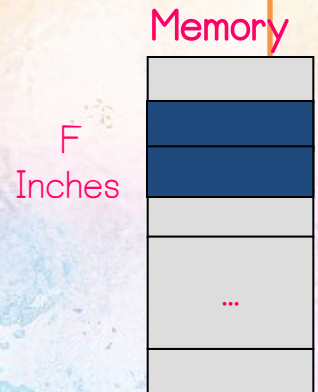
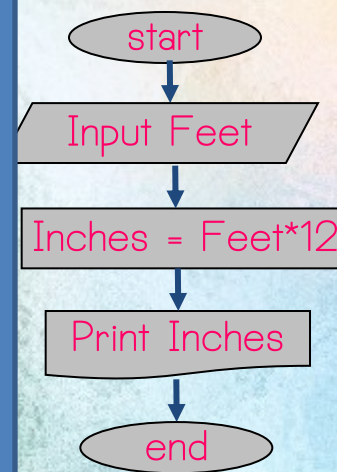
Enter length: _ 10

Area = 50.0

ตัวอย่าง 2.4



- เขียนโปรแกรม แสดงการเปลี่ยนค่าข้อมูล (ที่รับมามีหน่วยเป็นฟุต) ให้เป็นนิ้ว



ผลลัพธ์

Enter feet: 2

= 24 inches

ฟังก์ชัน getch, gets

- **scanf** เป็นฟังก์ชันที่รับข้อมูลได้ทุกชนิด
 - จึงต้องระบุ “**%format**” สำหรับข้อมูลแต่ละชนิด
- แต่สำหรับข้อมูลหรือตัวแปร Character สามารถเลือกใช้ฟังก์ชัน
 - **getchar()** รับหนึ่งตัวอักษร และต้องกด ENTER
เป็นฟังก์ชันใน Library “**stdio**”
 - **getch()** รับหนึ่งตัวอักษร แต่ไม่ต้องกด ENTER
เป็นฟังก์ชันใน Library “**conio**”
 - **gets()** รับหลายตัวอักษรเป็น String และต้องกด ENTER
เป็นฟังก์ชันใน Library “**stdio**”

ตัวอย่าง 2.5

- เขียนโปรแกรม รับข้อมูล วัน (Day), เดือน (Month), ปี (Year)
เช่น 1, November, 2023 และแสดงผลพัธ์เป็นการพิมพ์วันที่
ในรูปแบบ Month day, year (เช่น November 1, 2023)

ผลลัพธ์

Enter Day : 1
Enter Month : November
Enter Year : 2023
Date is November 1, 2023

Memory

Day
Year

Month

1
2015
N
o
v
e
m
b
e
r

ตัวอย่าง 2.5

- เขียนโปรแกรม รับข้อมูล วัน (Day), เดือน (Month), ปี (Year) เช่น 1, November, 2015 และแสดงผลลัพธ์เป็นการพิมพ์วันที่ในรูปแบบ Month day, year (เช่น November 1, 2015)

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int    Day, Year;
```

```
    char Month[20]; // string
```

```
    printf("Enter Day: "); scanf("%d", &Day);
```

```
    printf("Enter Month: "); scanf("%s", &Month);
```

```
    printf("Enter Year: "); scanf("%d", &Year);
```

```
    printf("Date is %s %d, %d\n", Month, Day, Year);
```

```
}
```

```
    fflush(stdin); getchar();
```

ผลลัพธ์

Enter Day: _ 1

Enter Month: _November

Enter Year: _ 2015

Date is November 1, 2015

Memory

Day
Year

Month



↑
รับข้อมูลที่ตั้งอยู่ใน Keyboard Buffer

↑
รอรับหนึ่งตัวอักษรจาก Keyboard