โครงสร้าง

โปรแกรมภาษา C

2.1 โปรแกรมภาษา

- **โปรแกรมภาษา** ใช้ในการพัฒนาโปรแกรมสำหรับงานเฉพาะตามที่ผู้ใช้ต้องการ
- **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
- แต่ยุ่งยากต่อการพัฒนาโดยมนุษย์
- ตัวอย่างเช่น **Object Code**

```
00111100001111010000001000000011
10000011000000111000001000000010
00001010000010100100000001000010
10000100000000100101000100010011
00001100000011010000111000001111
10001001000010011000100000001000
00010100000101010001011000010111
00011000000110010001101000011011
00011100000111010001111000011111
00100100001001010010011000100111
00111100001111010011111000111111
00101000001010010010101000101011
00101100001011010010111000101111
00110000001100010011001000110011
10011011000110111001101000011010
00111000001110010011101000111011
```

ภาษาแอสเซมบลี

- **ภาษาแอสเซมบลี** (Assembly Language)

  เป็นภาษาที่เขียนโดยใช้คำสั่งเป็นคำเฉพาะใน ภาษาอังกฤษที่มนุษย์เข้าใจ แทนการใช้รหัสเลขฐาน 2

  - **แต่ออกแบบมาเฉพาะสำหรับคอมพิวเตอร์แต่ละรุ่น**
  - **ผู้เขียนโปรแกรมยังต้องทราบข้อมูลที่เกี่ยวข้องกับอุปกรณ์ของเครื่องคอมพิวเตอร์** (ยังยากต่อการพัฒนา)
  - **และต้องใช้ แอสเซมเบอร์ (Assembler) ในการแปล ภาษาแอสเซมบลี ให้เป็นภาษาเครื่อง**

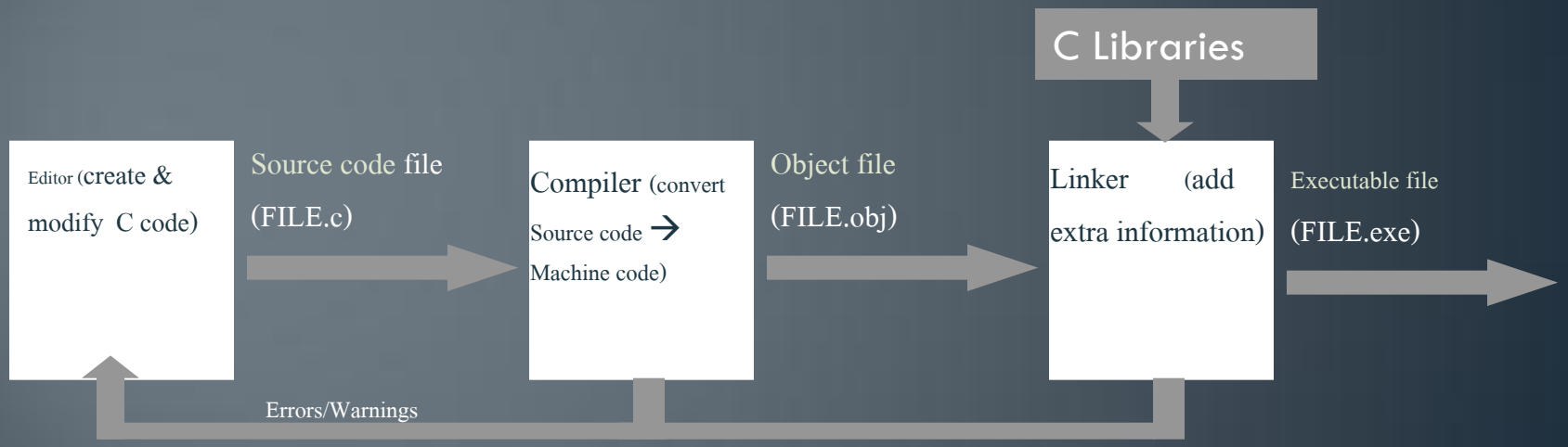
```
;CLEAR SCREEN USING BIOS
CLR:    MOV AX, 0600H
        MOV BH, 30
        MOV CX, 0000
        MOV DX, 184FH
        INT  10H
;INPUT A STRING
KEY:    MOV AH, 0AH
        LEA  DX, BUFFER
        INT  21H
        RET
```

Assembler

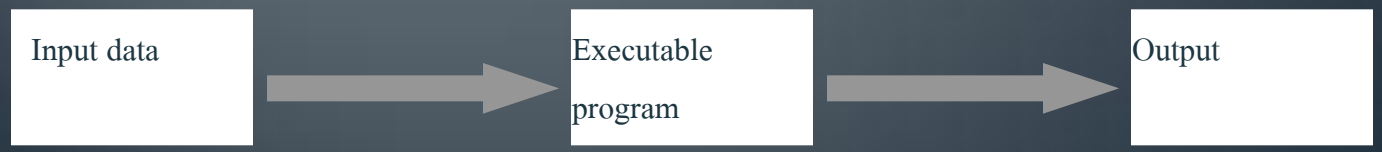
```
00111100001111010000001000000011
10000011000000111000001000000010
00001010000010100100000001000010
10000100000001001010001000100011
00001100000011010000111000001111
10001001000010011000100000001000
00010100000101010001011000010111
00011000000110010001101000011011
00011100000111010001111000011111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

2.2 การ Compile

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

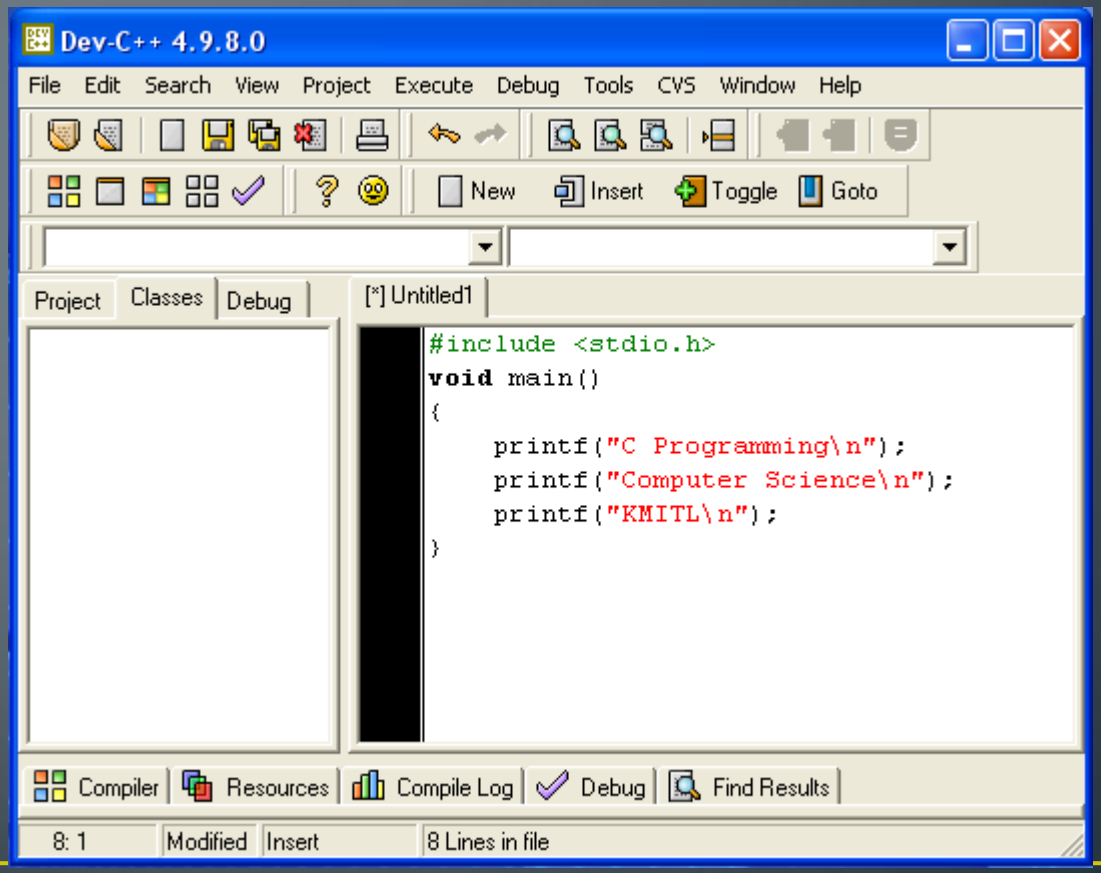


- ขั้นตอนการประมวลผลโปรแกรมภาษา C



Compile & Run

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



2.3 โครงสร้างโปรแกรม C

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

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

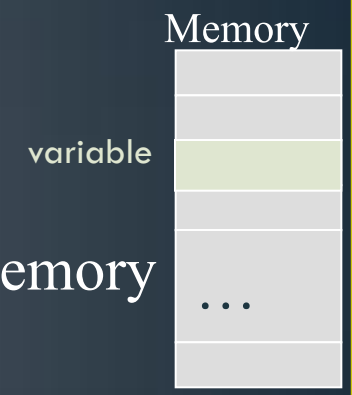
- **ตัวอย่างเช่น** ถ้าต้องการพิมพ์ C Programming
- **C Program** คือ

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

- **\n หมายถึง การย้าย cursor ไปบรรทัดใหม่ (newline)**

2.5 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), …

%d สำหรับ integer หรือ decimal

%f สำหรับ real หรือ floating point

ตัวอย่าง 2.1

- เขียนโปรแกรม พิมพ์คำว่า C Programming

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

F:\My cProgram\example.exe

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

Dev-C++ 4.9.8.0

File Edit Search View Project Execute Debug Tools CVS Window Help

New Insert Toggle Goto

Project | Classes | Debug | example.c

```
#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();
}
```

Compiler | Resources | Compile Log | Debug | Find Results

17: 2 | Insert | 18 Lines in file

2.6 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);

  รับข้อมูลชนิด integer จากคีย์บอร์ด

  แล้วนำไปเก็บไว้ใน ตัวแปร X ที่ตำแหน่ง &X

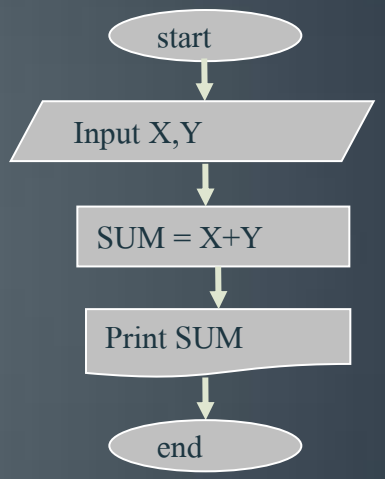
%d สำหรับ integer หรือ decimal

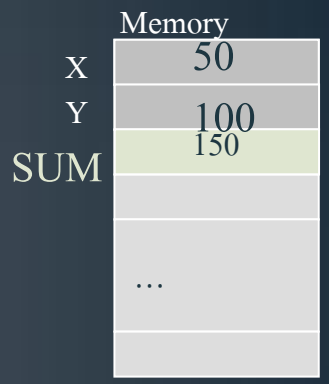
%f สำหรับ real หรือ floating point

ตัวอย่าง 2.2

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

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

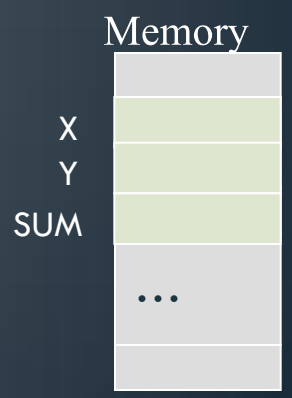
ตัวแปรและชนิดของข้อมูล
(Variables & Data Types)

%format

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

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

```
#include <stdio.h>
void 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);
}
```



3.1 ตัวแปร

- ตัวแปร (Variable) เป็นชื่อที่ใช้แทนการอ้างอิงข้อมูลที่เก็บอยู่ใน Memory
  - ตัวแปรที่เก็บค่าต่างๆ ในโปรแกรม (เช่น X, Y, SUM, …) จะต้องถูกจองเนื้อที่ไว้ที่ Address หนึ่งๆ ใน Memory (เช่น 2293604, 2293605, 2293606, …)
  - ค่าของตัวแปร สามารถเปลี่ยนค่าได้ ในขณะประมวลผล

การจองเนื้อที่ให้ตัวแปร

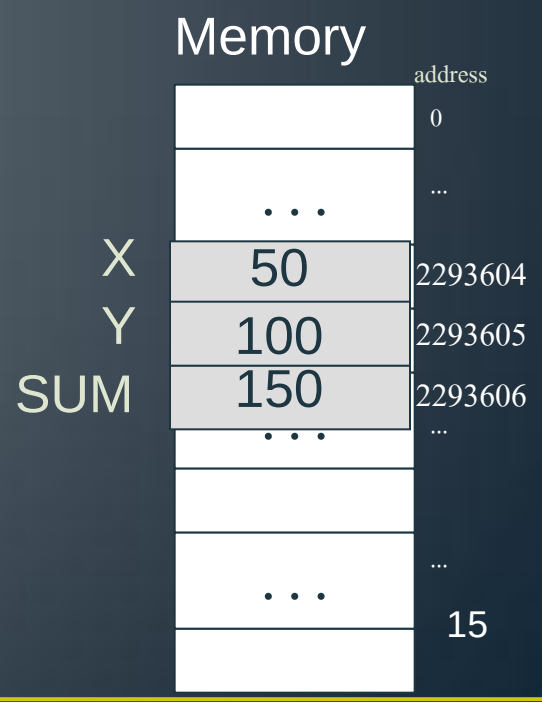
```
int  X, Y, SUM;
```

การกำหนดค่าตัวแปร

```
X = 50;
Y = 100;
```

การคำนวณค่าตัวแปร

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



3.1.1 การตั้งชื่อตัวแปร

- **กฎการตั้งชื่อตัวแปร**
  1. ประกอบด้วย a - z, A - Z, 0 - 9 หรือ _
  2. อักษรตัวแรกต้องเป็น a - z, A - Z, หรือ _ เท่านั้น
  3. ห้ามใช้ชื่อเฉพาะ (Reserved Words) เช่น int, if, float,...
  4. ยาวสูงสุด 31 ตัวอักษร
- **ตัวอย่าง:**
  - ชื่อที่ใช้ได้ เช่น i, n, _sys, K, SUM, …
  - แต่ชื่อที่ไม่อนุญาต เช่น int, 5j, sum 2
  - **ชื่อที่มีตัวพิมพ์ใหญ่ / เล็ก** มีความหมายต่างกัน

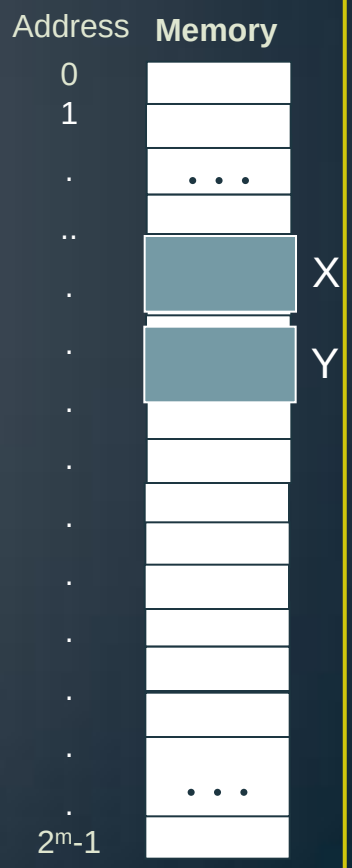
    เช่น ชื่อตัวแปร sum, Sum, SUM ต่างกัน

3.1.2 การประกาศตัวแปร

- การประกาศตัวแปร (Variable declaration)

  data_type variable1, variable2, ... ;

  - เช่น int X, Y;
  - เพื่อเป็นการ**จองเนื้อที่ใน Memory** เพื่อเก็บข้อมูล และเรียกใช้ในโปรแกรม (ในรูปแบบของตัวแปร)



- ชนิดของตัวแปร มี 3 ชนิด (ตามชนิดข้อมูลที่เก็บ)

  1. ตัวแปรชนิด Integer (2 bytes)
  2. ตัวแปรชนิด Floating Point (4 bytes)
  3. ตัวแปรชนิด Character (1 byte)

3.2 ชนิดข้อมูล

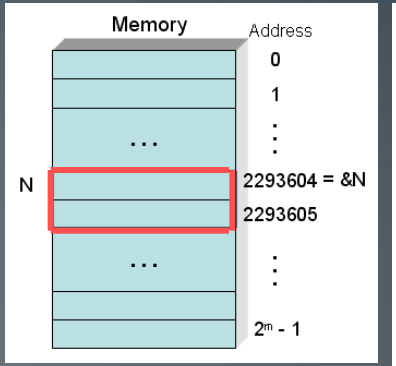
- **ชนิดข้อมูล (Data Types)** ในภาษา C มีหลายชนิด (int, float, char,…) ที่มีขนาด (byte) ต่างกัน
- ผู้ใช้ต้องเลือกใช้ให้เหมาะสมกับการใช้งาน (ใช้เนื้อที่น้อยๆ โดยไม่เกิดค่า Overflow)

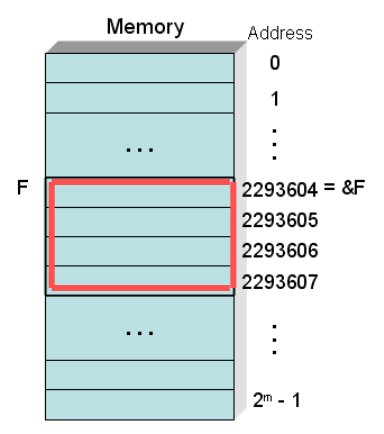
-2^{15} $+2^{15}-1$

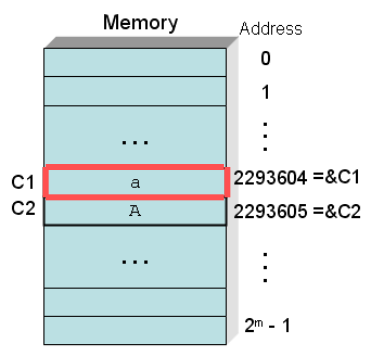
	ชนิดข้อมูล	ขนาด (bits)	ค่าต่ำสุด	ค่าสูงสุด
จำนวนเต็ม (Integer)	int	16	-32768	+32767
จำนวนจริง(Real) แบบ Floating Point	float	32	-3.402823×10^{38}	$+3.402823 \times 10^{38}$
ตัวอักษร(Character)	char	8	-128	+127

3.3 ชนิดตัวแปร

- ตัวแปรในภาษา C มีหลายชนิด ที่กำหนดตามชนิดของข้อมูล
- ตัวแปรพื้นฐาน 3 ชนิด



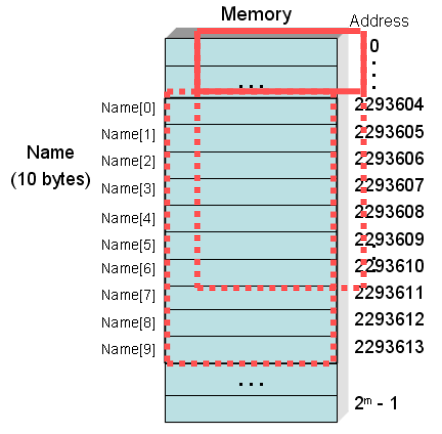




- ตัวแปร Integer (2 bytes) เก็บค่าเลขจำนวนเต็ม
- ตัวแปร Floating Point (4 bytes) เก็บค่าเลขจำนวนจริง
- ตัวแปร Character (1 byte) เก็บตัวอักษรรหัส ASCII

ตัวแปรขั้นสูง

- ตัวแปร String (m bytes) เก็บหลายตัวอักษร
- ตัวแปร Array (n bytes) เก็บค่าเลขหลายสมาชิก



3.3.1 ตัวแปร Integer

- **ตัวแปรชนิด Integer** (เลขจำนวนเต็ม) ที่สามารถแสดงได้ 3 เลขฐาน
  - **เลขฐาน 10** (0,1,2,3,4,5,6,7,8,9)
  - **เลขฐาน 8** (0,1,2,3,4,5,6,7)
  - **เลขฐาน 16** (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)
- ตัวแปร Integer (ขนาด 16, 32 บิต)
  - int (16 bits): -32768 - +32767
  - unsigned int (16 bits): 0 - 65535
  - long (32 bits): -2147483648 - +2147483547
  - unsigned long (32 bits): 0 - 4294967296
- ตัวอย่างเช่น

```
int   X;        // 2 bytes
long N;   // 4 bytes
```

ตัวอย่าง 3.1

- เขียนโปรแกรม รับค่าตัวแปร N เป็นเลขฐาน 10 (%d) (Decimal Number) แล้วแสดงผลลัพธ์ เป็นเลขฐาน 8, 10, 16 ด้วย %o, %d, %X

```
#include <stdio.h>
void main()
{
        unsigned int  N;
        printf("Enter N (base 10): ");
        scanf("%d", &N);
            printf("N = %o (base 8)\n", N);
            printf("N = %d (base 10)\n", N);
            printf("N = %X (base 16)\n", N);
    }
```

ผลลัพธ์

```
Enter N (base 10): 10
N = 12 (base 8)
N = 10 (base 10)
N = A (base 16)
```

ตัวอย่าง 3.2

- เขียนโปรแกรม รับค่าตัวแปร N เป็นเลขฐาน 16 (%X) (Hexadecimal Number) แล้วแสดงผลลัพธ์เป็นเลขฐาน 8, 10, 16 ด้วย %o, %d, %X

```
#include <stdio.h>
void main()
{
        unsigned int  N;
        printf("Enter N (base 16): ");
        scanf("%X", &N);
        printf("N = %o (base 8)\n", N);
        printf("N = %d (base 10)\n", N);
        printf("N = %X (base 16)\n", N);
 }
```

ผลลัพธ์

```
Enter N (base 16): 20
N = 40 (base 8)
N = 32 (base 10)
N = 20 (base 16)
```

%format ใน scanf, printf

%format	ในคำสั่ง scanf("%format", &variable);
%d	สำหรับรับข้อมูลของตัวแปร int
%u	สำหรับรับข้อมูลของตัวแปร unsigned int
%ld	สำหรับรับข้อมูลของตัวแปร long
%lu	สำหรับรับข้อมูลของตัวแปร unsigned long
%o, %lo	สำหรับรับข้อมูลของตัวแปร int, long, unsigned .. (ฐาน 8)
%X, %lX	สำหรับรับข้อมูลของตัวแปร int, long, unsigned .. (ฐาน 16)

%format	ในคำสั่ง printf("%format", variable);
%d,%u,%ld,%lu	กำหนดใช้สำหรับข้อมูลเช่นเดียวกับใน scanf
%o, %#o	สำหรับพิมพ์ข้อมูลเลขฐาน 8 (กรณี # จะมี 0 นำหน้าค่าเลขฐาน 8)
%X, %#X	สำหรับพิมพ์ข้อมูลเลขฐาน 16 (กรณี # จะมี 0X นำหน้าค่าเลขฐาน 16)
%.2d	สำหรับพิมพ์ข้อมูล 2 ตำแหน่งรวมเลข 0 นำหน้า
%10d	สำหรับพิมพ์ข้อมูล integer ชิดขวาภายใน 10 ตำแหน่ง