

พื้นฐานภาษา C

1. การแสดงข้อความ

```
#include<stdio.h>
int main()
{
    printf("message");
    return 0;
}
```

2. การแสดงข้อความจากตัวแปร

```
#include<stdio.h>
int main()
{
    int      a = 100;
    float    b = 2.2;
    char     c = 'C';
    printf("message %d %f %c",a,b,c);
    return 0;
}
```

3. การรับข้อความ

```
#include<stdio.h>
int main()
{
    int a;
    scanf("%d",&a);
    printf("%d",a);
    return 0;
}
```

4. ตัวแปร Local ใช้ได้เฉพาะภายในฟังก์ชัน

```
#include<stdio.h>
int main()
{
    int v = 10;
    printf("%d",v);
}
```

5. ตัวแปร Global ใช้ได้หมด

```
#include<stdio.h>
int v = 10;
int main()
{
    printf("%d",v);
}
```

6. เงื่อนไข

```
#include<stdio.h>
int main()
{
    int j=25;
    if(j>10 && j <= 20)
        { printf("1\n"); }
    else if(j>20)
        { printf("2\n "); }
    else
        { printf("3\n "); }
}
```

7. ວິຊາ

```
#include<stdio.h>
int main()
{
    int i;
    for(i=0;i<10;i++)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

T.O 70%

8. Comment

```
/* message */  
//message
```

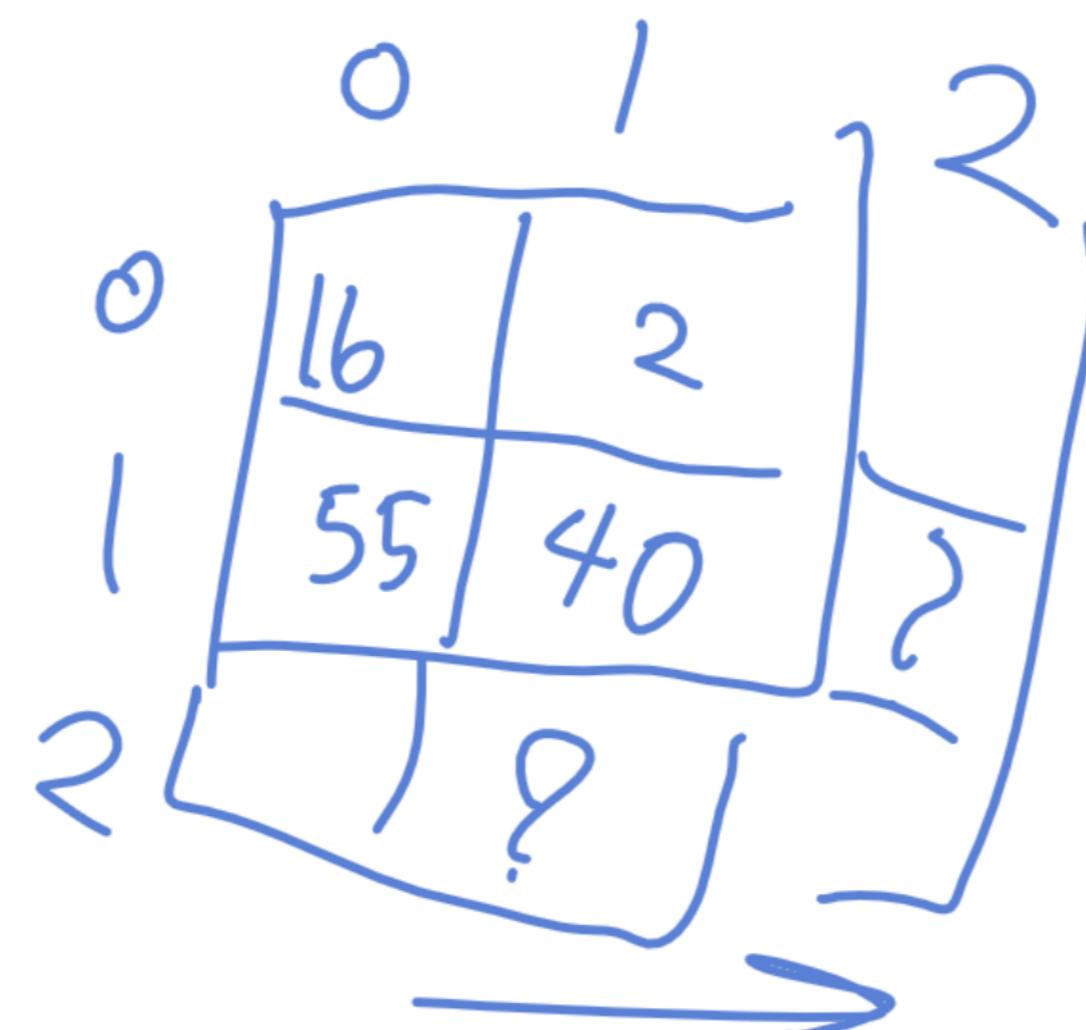
9. Array 1 dimension

```
#include<stdio.h>
int main()
{
    int a[5] = { 10, 20, 30, 11, 54 };
    int b[5];
    a[2] = 44;
    printf("%d\n", a[2]);
    return 0;
}
```

Index	0	1	2	3	4
Value					

10. Array 2 dimension

```
#include<stdio.h>
int main()
{
    int b[2][2] = { {16, 2}, {77, 40} };
    int b[2][2];
    b[1][0] = 55;
    printf("%d\n", b[1][0]);
    return 0;
}
```



Index	0	1
0		
1		

11 ขนาด Array

```
int arr[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
int size = sizeof(arr) / sizeof(arr[0]);
```

12. Write File

```
#include<stdio.h>
int main()
{
    int num;
    FILE *f = fopen("d:\\a.txt", "w");
    scanf("%d", &num);
    fprintf(f, "%d", num);
    fclose(f);

    return 0;
}
```

13. Read File

```
#include<stdio.h>
int main()
{
    int num;
    FILE *fptr = fopen("d:\\a.txt","r");
    fscanf(fptr,"%d", &num);
    printf("%d", num);
    fclose(fptr);

    return 0;
}
```

14. String

```
#include<stdio.h>
int main()
{
    char str[] = "Geeks";
    printf("%s\n", str);
    strcpy(str,"change");
    printf("%s\n", str);
    return 0;
}
```

15. แปลง STRING, INT, FLOAT

```
#include<stdio.h>
int main()
{
    int i = atoi("20");
    printf("%d\n",i);

    char buffer [50];
    itoa(250, buffer,10);
    printf("%s\n",buffer);
    return 0;
}
```

16. Pointer

name	ip	name	var
value	&var	address	&var
value of value	*ip	value	20
address	&ip		

name	ip	name	var
value	&var	address	&var
value of value	*ip = 30	value	30
address	&ip		

name	ip	name	var
value	&var	address	&var
value of value	*ip	value	50
address	&ip		

```
#include<stdio.h>
int main()
{
    int *ip;
    int var = 20;
    printf("%d\n",var);

    ip = &var;
    printf("%d\n",*ip);
    printf("%d\n",var);
    printf("%x\n",ip);
    printf("%x\n",&var);
    printf("%x\n",&ip);

    *ip = 30;
    printf("%d\n",*ip);
    printf("%d\n",var);

    var = 50;
    printf("%d\n",*ip);
    printf("%d\n",var);

    return 0;
}
```

17. Function

```
#include<stdio.h>
void f1()
{
    printf("me\n");
}
int f2()
{
    return 10;
}
int f3(float a, float b)
{
    float c = a * b;
    return c;
}
int main()
{
    f1();
    printf("%d\n",f2());
    printf("%d\n",f3(5,4));
    return 0;
}
```

18. Pass by Reference and Pass by Value

```
#include<stdio.h>
void f4(int *a)
{
    *a = 50;
    printf("%d\n",*a);
}
void f5(int a)
{
    a = 200;
    printf("%d\n",a);
}
int main()
{
    int b = 44;
    printf("%d\n",b);
    f4(&b);
    printf("%d\n",b);

    b = 44;
    printf("%d\n",b);
    f5(b);
    printf("%d\n",b);
    return 0;
}
```

19. STRUCT

```
#include<stdio.h>
typedef struct B
{
    int i;
    char m[10];
} tB;
void print(tB o)
{
    printf("%d %s \n",o.i,o.m);
}
int main()
{
    tB b1, *b5;

    b1.i = 10;
    strcpy(b1.m,"test");
    print(b1);

    b5 = &b1;
    (*b5).i = 10;
    strcpy( b5->m, "test111");
    print(b1);
    print( (*b5) );
}
```

Computer Programming II การเขียนโปรแกรมคอมพิวเตอร์ 2 LECTURE#1

อ.สุทธิ์ ประสมพันธ์

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

KMUTNB

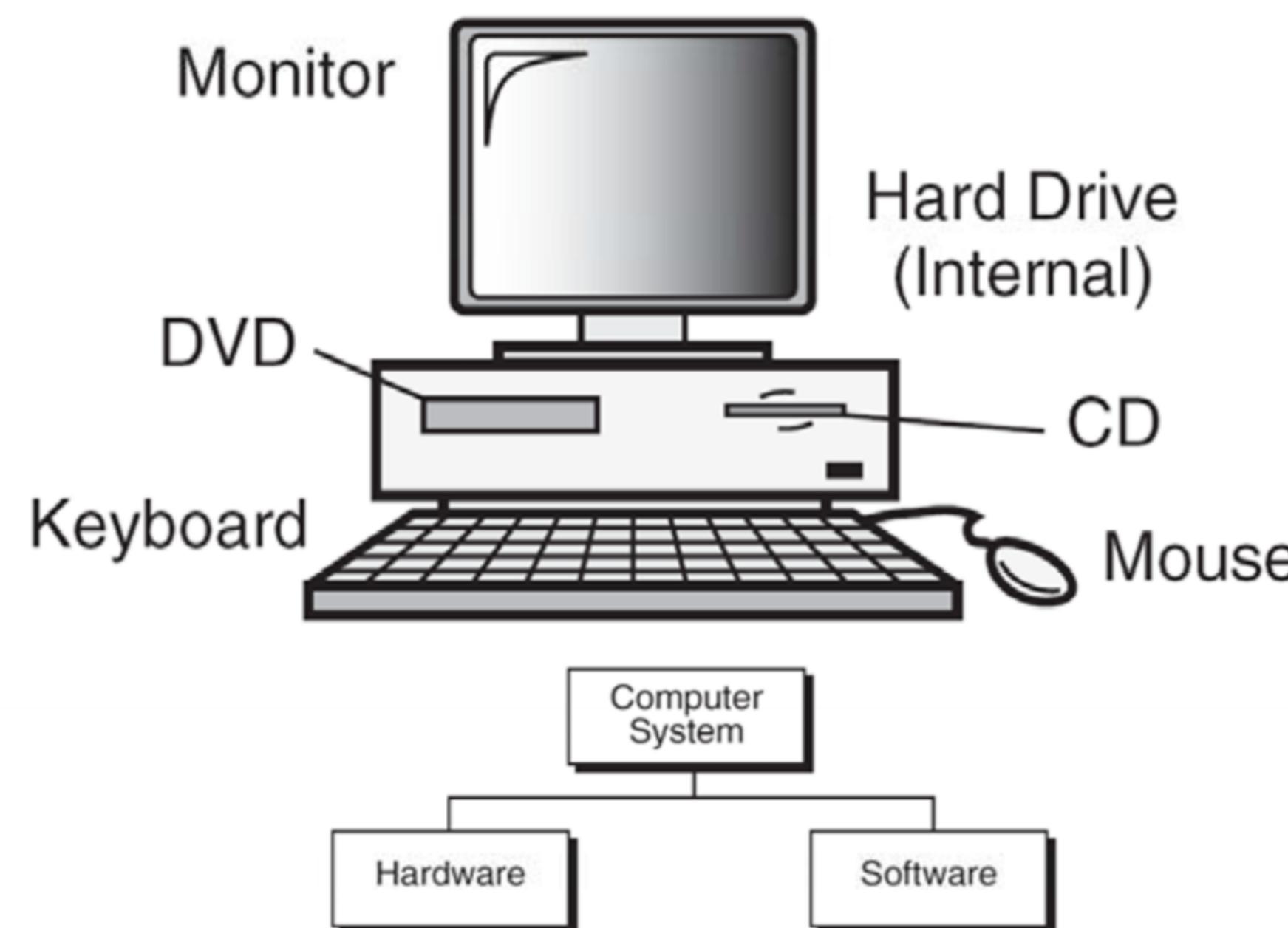
ความหมายของคอมพิวเตอร์

- คอมพิวเตอร์เป็นอุปกรณ์ทางไฟฟ้านิดหนึ่งที่สามารถประมวลผลและจำข้อมูลต่าง ๆ ได้ สามารถคำนวณตัวเลข สามารถตอบสนองต่อการกระทำของผู้ใช้ได้และมีความสามารถในการเชื่อมต่อกับอุปกรณ์ไฟฟ้าบางชนิด เพื่อส่งให้อุปกรณ์นั้นทำงานตามคำสั่งได้ ใช้สำหรับแก้ปัญหาต่าง ๆ ทั้งที่ง่ายและซับซ้อนโดยวิธีทางคณิตศาสตร์

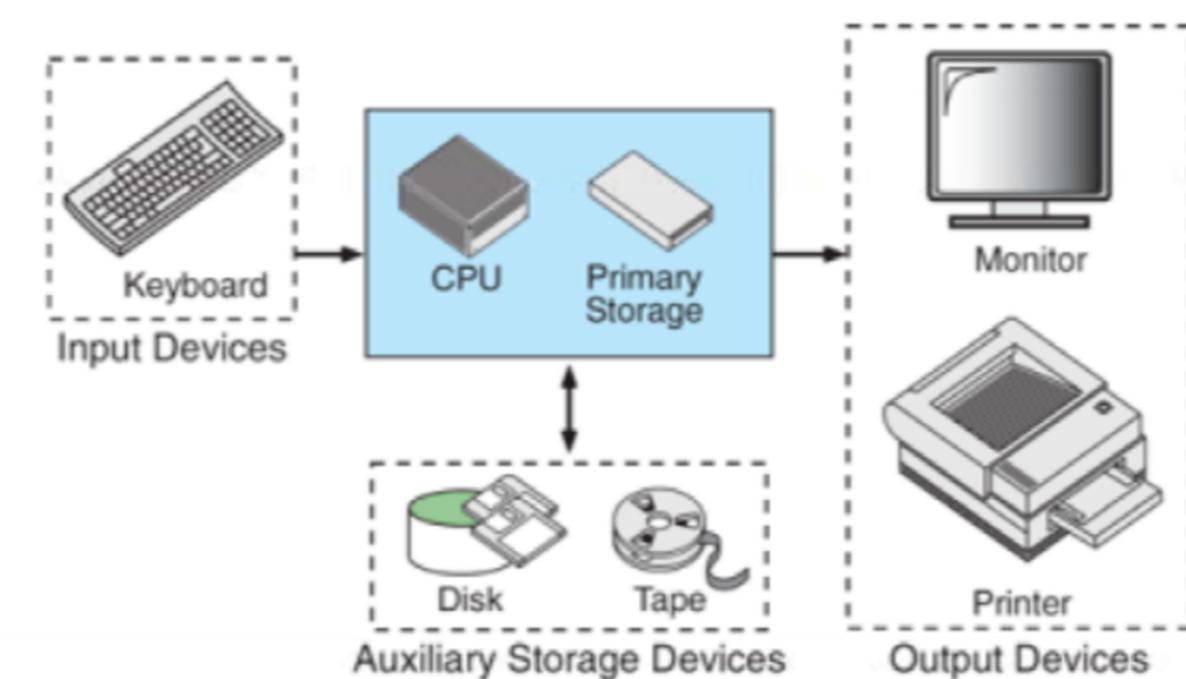
Computer Programming II

2

องค์ประกอบของระบบคอมพิวเตอร์



องค์ประกอบของระบบคอมพิวเตอร์



- คอมพิวเตอร์ฮาร์ดแวร์ (Hardware Computer)
- เป็นส่วนประกอบของคอมพิวเตอร์ที่จับต้องสัมผัสได้ ประกอบด้วยวงจรทางไฟฟ้า อิเล็กทรอนิกส์ที่เรียกว่าฮาร์ดแวร์ (Hardware)
- มีส่วนประกอบที่สำคัญได้แก่
 - หน่วยรับข้อมูล
 - หน่วยแสดงผลข้อมูล
 - ระบบประมวลผล
 - หน่วยเก็บข้อมูล

องค์ประกอบของระบบคอมพิวเตอร์

- 1. หน่วยรับข้อมูล(Input Unit)
 - เป็นส่วนที่ใช้รับข้อมูลและคำสั่งจากภายนอกเข้าสู่เครื่องคอมพิวเตอร์เพื่อนำไปประมวลผล
 - อุปกรณ์อินพุตจะเปลี่ยนข้อมูลที่มีนุชย์เข้าใจเปลี่ยนเป็นรหัสข้อมูลที่เครื่องคอมพิวเตอร์เข้าใจ
 - อุปกรณ์เหล่านี้จะทำงานได้ ข้อมูลคำสั่งจะต้องถูกเก็บไว้บนสื่อ(Input media) ที่อุปกรณ์นั้น ๆ รู้จักเรียกว่า Input Device
- 2. หน่วยแสดงผลหรือเอาต์พุต(Output Unit)
 - เป็นส่วนที่ใช้แสดงผลลัพธ์จากการประมวลผลออกมาในรูปแบบต่าง ๆ ที่มนุษย์เข้าใจ

องค์ประกอบของระบบคอมพิวเตอร์

- 3. หน่วยประมวลผลกลาง(Central Processing Unit)
 - มีหน้าที่เก็บข้อมูลคำสั่งทำการประมวลผลทางคณิตศาสตร์ เปรียบเทียบเทียบข้อมูล เมื่อข้อมูลเข้าสู่ระบบแล้วหน่วยประมวลผลจะทำหน้าที่ประมวลผลตามคำสั่ง หรือโปรแกรมที่กำหนดไว้ โดยโปรแกรมและข้อมูลต่าง ๆ จะถูกเก็บเอาไว้ในหน่วยความจำ เมื่อหน่วยประมวลผลทำงานสำเร็จแล้วจะเก็บข้อมูลลงหน่วยเก็บข้อมูลหรือส่งผลลัพธ์ที่ได้ออกทางหน่วยแสดงผลต่อไป
 - หน่วยประมวลผลกลาง มีหน้าที่ 2 อย่างคือ
 - 1. ทำหน้าที่ประสานการทำงานในระบบคอมพิวเตอร์
 - 2. ทำหน้าที่ประมวลผลทางคณิตศาสตร์และตรรกศาสตร์ของข้อมูล
 - หน่วยประมวลผลกลางแบ่งหน่วยการทำงานออกเป็น 3 หน่วยหลัก คือ
 - หน่วยควบคุม(Control Unit)
 - หน่วยคำนวณและตรรกศาสตร์(Arithmetic and Logic Unit)
 - หน่วยความจำหลัก(Main Memory หรือ Primary Storage)

องค์ประกอบของระบบคอมพิวเตอร์

- การวัดขนาดของหน่วยความจำหลัก จะวัดจากจำนวนข้อมูลที่เก็บโดยมีหน่วยของการวัดดังนี้
 - 1 KB(Kilo Byte)=1024 Bytes
 - 1 MB(Mega Byte) =1024 K Bytes
 - 1 GB(Giga Byte) = 1024 M Bytes
 - 1 TB(Tera Byte) = 1024 G Bytes

องค์ประกอบของระบบคอมพิวเตอร์

- 4. หน่วยความจำสำรอง (Auxiliary Memory หรือ Secondary storage)
 - เป็นหน่วยความจำที่อยู่นอกเครื่องคอมพิวเตอร์ มีหน้าที่ช่วยให้หน่วยความจำหลักทำงานได้มากขึ้น โดยจะเก็บข้อมูลที่ของการประมวลผล และข้อมูลที่ประมวลผลเสร็จแล้ว อุปกรณ์ที่นำมาใช้เป็นหน่วยความจำรอง เช่น Hard Disk Drive เป็นต้น

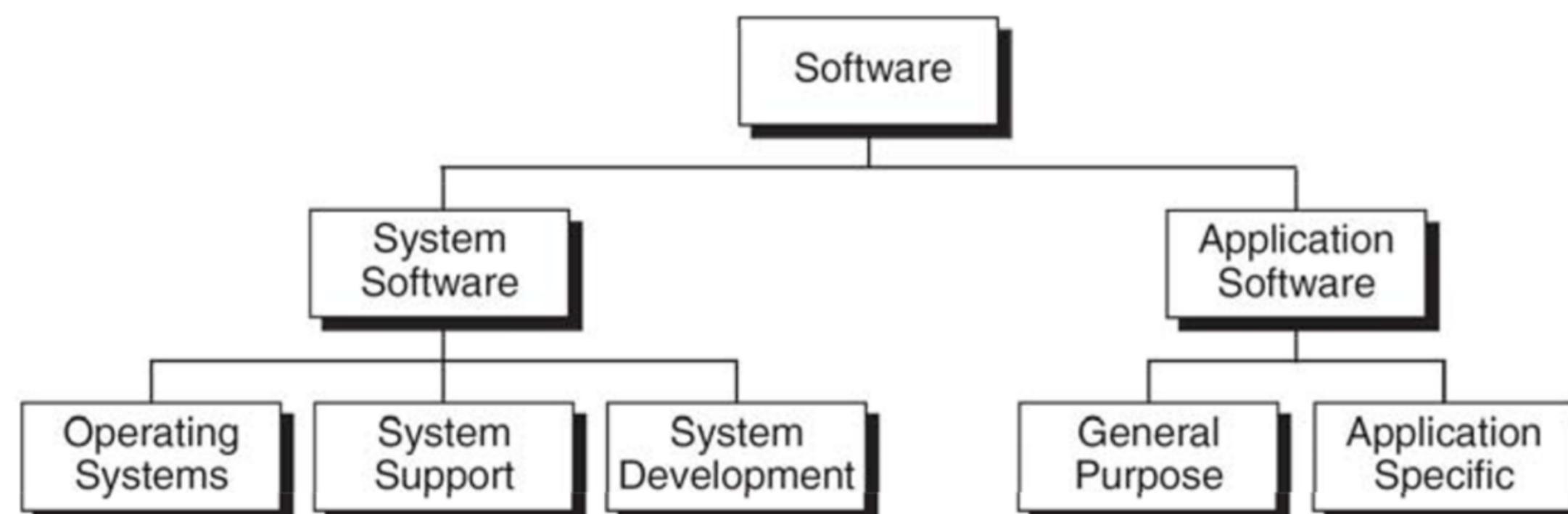
คอมพิวเตอร์ซอฟต์แวร์ (Software Computer)

- Software คือ ชุดคำสั่งที่มีไว้เพื่อทำงานอย่างโดยย่างหนึ่ง สามารถแบ่งซอฟต์แวร์ตามการทำงานได้ 2 ประเภท คือ
- ซอฟต์แวร์ระบบ (System Software)
 - เป็นซอฟต์แวร์ที่ทำหน้าที่ควบคุมการทำงานของเครื่องเพื่อให้สามารถทำงานต่าง ๆ ได้สะดวก แบ่งออกเป็น
 - โปรแกรมระบบปฏิบัติการ(OS: Operating System)
 - โปรแกรมอัตโนมัติ (Utilities Program)
 - โปรแกรมดีไวซ์ไดร์เวอร์ (Device Driver)

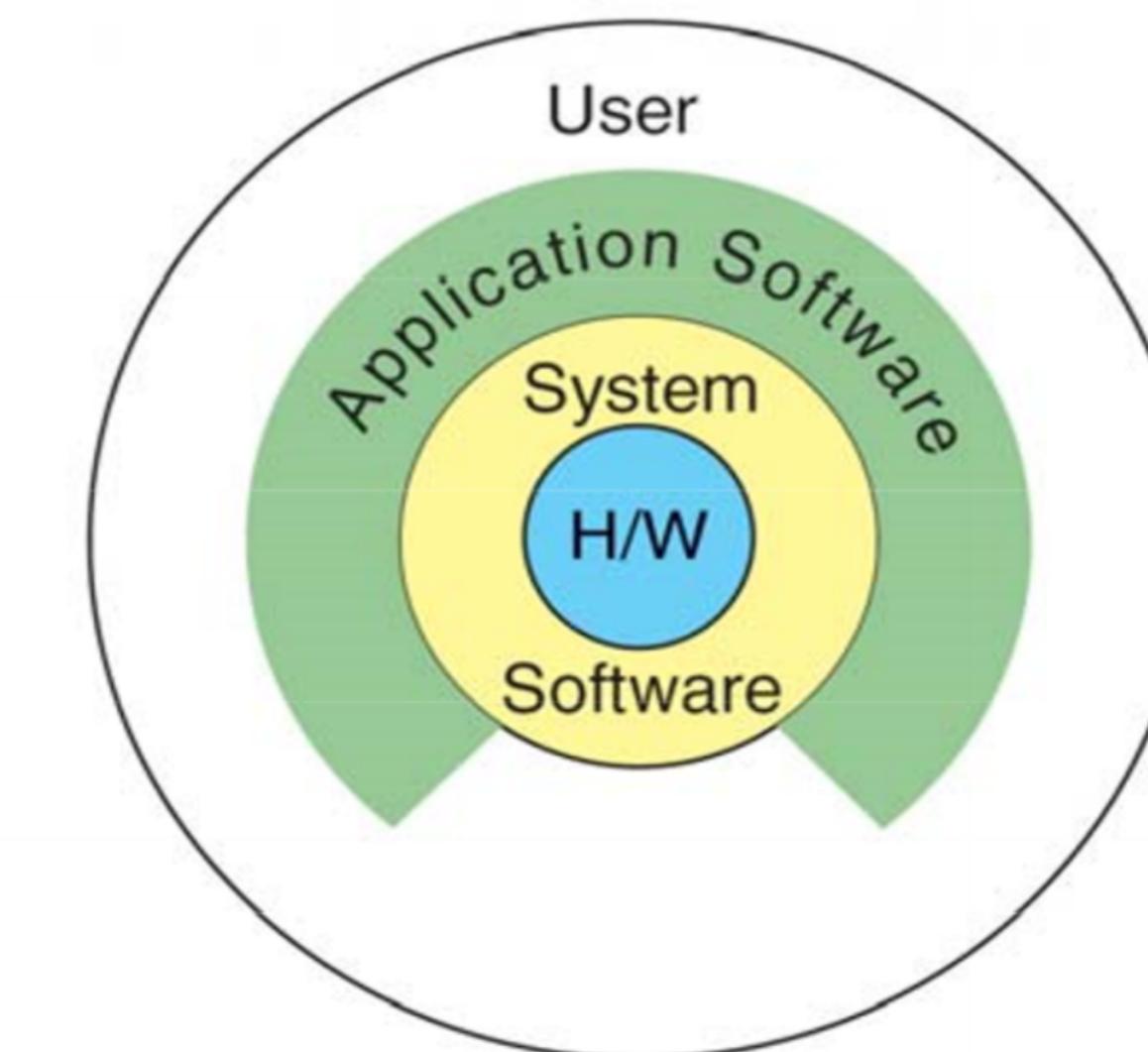
คอมพิวเตอร์ซอฟต์แวร์ (Software Computer)

- ซอฟต์แวร์ประยุกต์ (Application Software)
 - เป็นโปรแกรมที่พัฒนาขึ้นสำหรับงานเฉพาะต่างๆ อาจเป็นโปรแกรมที่เขียนขึ้นเองหรือโปรแกรมที่มีอยู่ทั่วไป Application Software ผลิตขึ้นมาเพื่อให้ผู้ใช้งานเฉพาะทาง เช่น ซอฟต์แวร์ประมวลผลคำ ซอฟต์แวร์ตารางจัดการ ประเภทของโปรแกรมประยุกต์ที่มองเห็นทั่ว ๆ ไปมีดังนี้
 - ซอฟต์แวร์สำเร็จรูป (Package Software)
 - ซอฟต์แวร์เฉพาะ(Custom Software)
 - ซอฟต์แวร์แบบเปิด (Open Source Software)
 - แชร์แวร์ (Shareware)
 - ซอฟต์แวร์ฟรี(Freeware)

คอมพิวเตอร์ซอฟต์แวร์ (Software Computer)



ความสัมพันธ์ระหว่างคอมพิวเตอร์ฮาร์ดแวร์และคอมพิวเตอร์ซอฟต์แวร์



ภาษาคอมพิวเตอร์(Computer Language)

- ภาษาเครื่อง (Machine Language)
- เป็นภาษาเครื่องเข้าใจคำสั่งได้โดย มีลักษณะคำสั่งเป็นตัวเลขล้วน เป็นภาษาคอมพิวเตอร์ระดับต่ำที่สุด เพราะเป็นตัวเลขฐานสอง แทนข้อมูล และคำสั่งต่างๆ ทั้งหมด

```
1 00000000 00000100 0000000000000000
2 01011110 00001100 11000010 0000000000000010
3 11101111 00010110 0000000000000010
4 11101111 10011110 000000000000001011
5 11110000 10101101 11011111 00000000000010010
6 01100010 11011111 00000000000010101
7 11101111 00000010 11110111 00000000000010111
8 11110100 10101101 11011111 00000000000011110
9 00000011 10100010 11011111 0000000000100001
10 11101111 00000010 11110101 0000000000100100
11 01111110 11110100 10101101
12 11110000 10101110 11000101 0000000000101011
13 00000110 10100010 11110101 00000000000110001
14 11101111 00000010 11110111 00000000000110100
15 01010000 11010100 00000000000111011
16 00000100 000000000000111101
```

ภาษาคอมพิวเตอร์(Computer Language)

- ภาษาระดับต่ำ(Low level Language)
- เป็นภาษาที่มีลักษณะใกล้เคียงกับภาษาเครื่อง เพียงแต่มีการใช้สัญลักษณ์ หรือตัวอักษรมาแทนคำสั่งในส่วนต่าง ๆ ตัวอย่างของภาษาที่ได้แก่ ภาษาแอสเซมบลี(Assembly) เป็นภาษาคอมพิวเตอร์ที่พัฒนาขึ้นมาเพื่อให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรม ติดต่อกับคอมพิวเตอร์ได้ง่ายกว่า ภาษาเครื่อง โดยใช้คำย่อภาษาอังกฤษใน การเขียนคำสั่ง

```
1 entry main,"m<r2>
2 subl2 #12,sp
3 jsb C$MAIN_ARGS
4 movab $CHAR_STRING_CON
5
6 pushal -8(fp)
7 pushal (r2)
8 calls #2,SCANF
9 pushal -12(fp)
10 pushal 3(r2)
11 calls #2,SCANF
12 mul13 -8(fp),-12(fp),-
13 pusha 6(r2)
14 calls #2,PRINTF
15 clrl r0
16 ret
```

ภาษาคอมพิวเตอร์(Computer Language)

- ภาษาระดับสูง(High level Language)
- เป็นภาษาที่มีลักษณะใกล้เคียงกับภาษาอังกฤษที่มนุษย์ใช้กันอยู่ เรียนรู้ง่าย เข้าใจได้ง่าย สะดวกในการใช้งาน และใช้ได้กับทุกเครื่อง ตัวอย่างของภาษา ระดับสูงได้แก่ ภาษาโคบอล(COBOL) ภาษาปาส卡ล (PASCAL) ภาษาซี(C) ภาษาจาวา (JAVA) เป็นต้น

```
1 /* This program reads two integers from the keyboard
2  and prints their product.
3  Written by:
4  Date:
5 */
6 #include <stdio.h>
7
8 int main (void)
9 {
10 // Local Definitions
11 int number1;
12 int number2;
13 int result;
14
15 // Statements
16 scanf ("%d", &number1);
17 scanf ("%d", &number2);
18 result = number1 * number2;
19 printf ("%d", result);
20 return 0;
21 } // main
```

ขั้นตอนการพัฒนาโปรแกรม

- 1. การวิเคราะห์ปัญหา (Problem Analysis)
- 2. เขียนผังงาน (Pseudo Coding)
- 3. เขียนโปรแกรม (Programming)
- 4. ทดสอบและแก้ไขโปรแกรม (Program testing and Debugging)
- 5. ทำเอกสารและบำรุงรักษาโปรแกรม (Program document and Maintenance)

การวิเคราะห์ปัญหา (Problem Analysis)

- เราสามารถวิเคราะห์ปัญหาโดยการวิเคราะห์ส่วนต่าง ๆ ดังนี้
 - วิเคราะห์ข้อมูลนำเข้า (Input Analysis)
 - วิเคราะห์ขั้นตอนการทำงาน (Process Analysis)
 - วิเคราะห์ผลลัพธ์ (Output Analysis)

ตัวอย่างการวิเคราะห์ปัญหา#1

- จะเขียนแนวทางการแก้ปัญหาด้วยคอมพิวเตอร์สำหรับให้คอมพิวเตอร์คำนวณค่าจ้างพนักงานเป็นรายชั่วโมง จากนั้นแสดงค่าจ้างที่คำนวณได้
- ต้องการอะไร?** ต้องการทราบค่าจ้างของพนักงานแต่ละคน
- ต้องการผลลัพธ์อย่างไร(Output)** ต้องการผลลัพธ์เป็นค่าจ้างสุทธิของพนักงานทางจอภาพ
- ข้อมูลเข้า(Input)** รหัสพนักงาน ชื่อพนักงาน จำนวนชั่วโมงทำงาน เก็บในตัวแปรชื่อ Hours ค่าจ้างรายชั่วโมงเก็บในตัวแปรชื่อ PayRate

ตัวอย่างการวิเคราะห์ปัญหา#1

- วิธีการประมวลผล(Process)**
- กำหนดวิธีการคำนวณ
 - ค่าจ้างสุทธิ = จำนวนชั่วโมง \times อัตราต่อชั่วโมง
- ขั้นตอนการประมวลผล
 - 1. เริ่มต้น
 - 2. รับรหัสพนักงาน ชื่อพนักงาน จำนวนชั่วโมงทำงานเก็บในตัวแปรชื่อ Hours ค่าจ้างรายชั่วโมงเก็บในตัวแปรชื่อ PayRate
 - 3. คำนวณ ค่าจ้างสุทธิ = Hours \times PayRate
 - 4. แสดงผลลัพธ์เป็นรหัสพนักงาน ชื่อ ค่าจ้างสุทธิของพนักงานทางจอภาพ
 - 5. จบการทำงาน

ตัวอย่างการวิเคราะห์ปัญหา#2

- จะเขียนโปรแกรมเพื่อรายงานผลสอบของนักศึกษาวิชาคอมพิวเตอร์ โดยให้แสดงคะแนนรวมและเกรดออกมา
- ต้องการอะไร?** ต้องการพิมพ์คะแนนผลสอบและเกรดของนักศึกษา
- ต้องการผลลัพธ์อย่างไร(Output)** ต้องการผลลัพธ์เป็นคะแนนรวม และเกรดของนักศึกษาแต่ละคน
- ข้อมูลเข้า(Input)** รหัสประจำตัวนักศึกษา(ID) ชื่อนักศึกษา(name) คะแนนสอบกลางภาค(mid) คะแนนสอบย่อย(test) คะแนนสอบปลายภาค(final)

ตัวอย่างการวิเคราะห์ปัญหา#2

- วิธีการประมวลผล(Process)
- กำหนดวิธีการคำนวณ
 - คะแนนรวม = คะแนนกลางภาค + คะแนนสอบย่อย + คะแนนปลายภาค
 - ถ้าคะแนนรวม ≥ 80 ได้เกรด “A”
 - ถ้าคะแนนรวม ≥ 70 และ < 80 ได้เกรด “B”
 - ถ้าคะแนนรวม ≥ 60 และ < 70 ได้เกรด “C”
 - ถ้าคะแนนรวม ≥ 50 และ < 60 ได้เกรด “D”
 - ถ้าคะแนนรวม < 50 ได้เกรด “F”

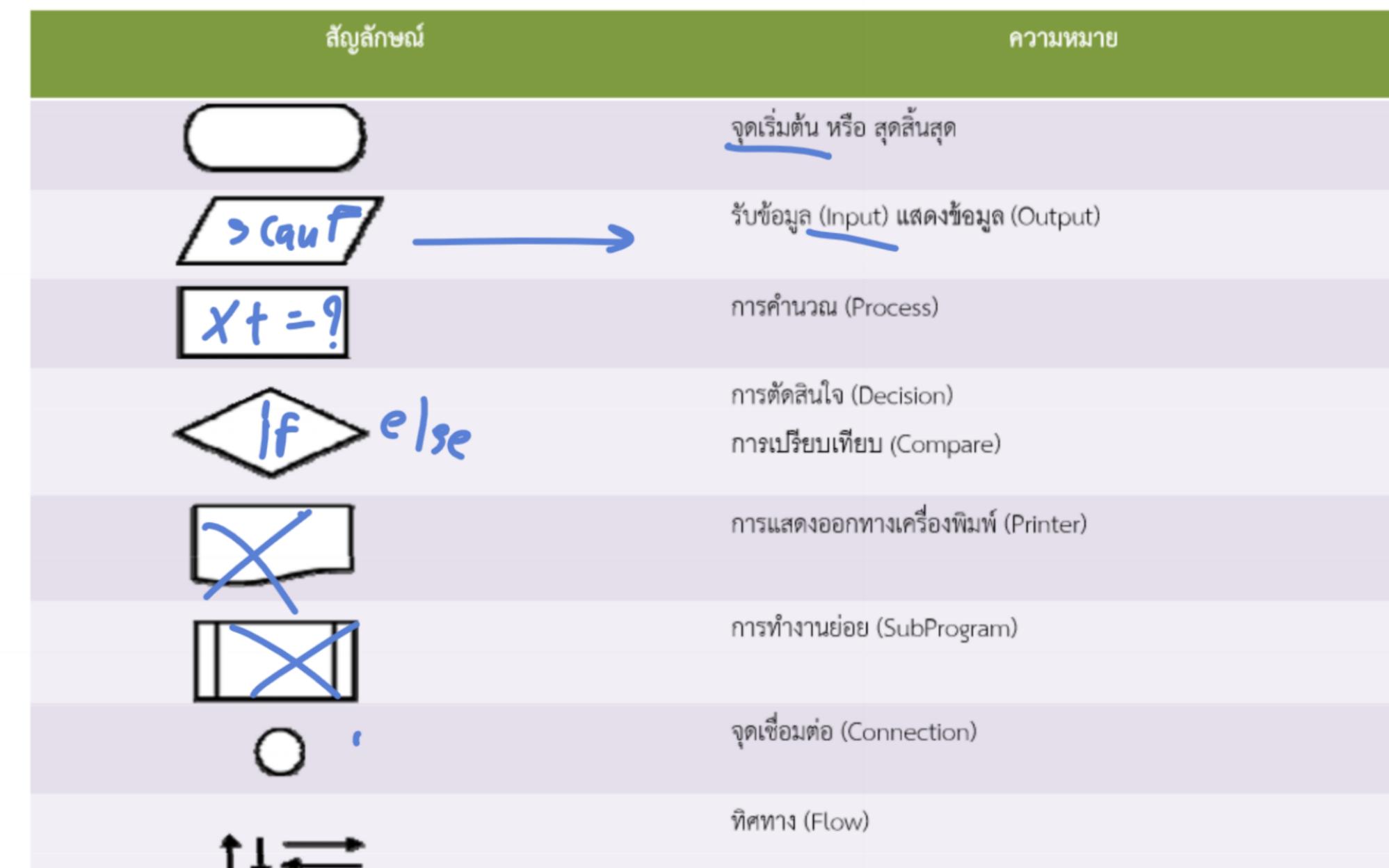
ตัวอย่างการวิเคราะห์ปัญหา#2

- ขั้นตอนการประมวลผล
 - เริ่มต้น
 - รับค่าตัวแปร ID name mid test final
 - คำนวนคะแนนรวมและเกรด
 - Total = mid + test + final
 - ถ้า Total ≥ 80 , Grade = “A”
 - ถ้า Total ≥ 70 และ < 80 , Grade = “B”
 - ถ้า Total ≥ 60 และ < 70 , Grade = “C”
 - ถ้า Total ≥ 50 และ < 60 , Grade = “D”
 - ถ้า Total < 50 , Grade = “F”
 - แสดง ID name Total Grade ของนักศึกษา
 - กลับไปข้อ 2 เพื่อรับจนครบทุกคน ถ้าครบแล้วไปข้อ 6
 - หยุดการทำงาน

การเขียนผังงานของโปรแกรม

- การเขียนผังงานที่ดี
 - เขียนตามสัญลักษณ์ที่กำหนด
 - ใช้ลูกศรแสดงทิศทางการทำงานจากบนลงล่าง
 - อธิบายสั้น ๆ ให้เข้าใจง่าย
 - ทุกแผนภาพต้องมีทิศทางเข้าออก
 - ไม่ควรโดยลูกศรไปที่ไกล ๆ มาก ถ้าต้องทำให้ใช้สัญลักษณ์ของการเขียนต่อแทน

การเขียนผังงานของโปรแกรม



การเขียนอัลกอริทึมของโปรแกรม

- อัลกอริทึม(Algorithms) หมายถึงลำดับขั้นตอนเชิงคำนวณที่แปลงข้อมูลด้านอินพุตของปัญหาไปเป็นผลลัพธ์ที่ต้องการ
- ขั้นตอนต่าง ๆ ในอัลกอริทึมสามารถเปลี่ยนไปเป็นคำสั่งที่ให้คอมพิวเตอร์ทำงานได้
- ถ้าหากทำตามอัลกอริทึมแล้ว ปัญหาจะต้องถูกแก้ได้สำเร็จและได้คำตอบที่ถูกต้องสำหรับทุกรูปแบบตามที่กำหนดในอัลกอริทึม
- ดังนั้น เราจะไม่ยอมรับอัลกอริทึมที่ทำงานติดอยู่ใน Loop ไม่มีที่สิ้นสุด
หรืออัลกอริทึมที่ทำงานแล้วได้คำตอบถูกบ้างผิดบ้าง

ตัวอย่างการเขียนอัลกอริทึมของโปรแกรม#1

- จงวิเคราะห์ปัญหาและเขียนอัลกอริทึมสำหรับหาค่าเฉลี่ยของอุณหภูมิประจำวันโดยรับค่าอุณหภูมิสูงสุดและอุณหภูมิต่ำสุดเป็นเลขจำนวนเต็มเข้าไปและให้แสดงค่าอุณหภูมิเฉลี่ยของทางจอภาพ
- ต้องการอะไร?** ต้องการทราบค่าเฉลี่ยของอุณหภูมิประจำวัน
- ต้องการผลลัพธ์อย่างไร(Output)** ต้องการผลลัพธ์เป็นค่าเฉลี่ยของอุณหภูมิประจำวันโดยใช้ตัวแปรชื่อ avg_temp
- ข้อมูลเข้า(Input)** รับค่าอุณหภูมิสูงสุดอยู่ในตัวแปรชื่อ max_temp อุณหภูมิต่ำสุดอยู่ในตัวแปรชื่อ min_temp

ตัวอย่างการเขียนอัลกอริทึมของโปรแกรม#1

- วิธีการประมวลผล(Process)**
 - รับค่าอุณหภูมิสูงสุดและอุณหภูมิต่ำสุด
 - หาค่าเฉลี่ยโดยใช้ $avg_temp = (max_temp+min_temp)/2$
 - แสดงค่า avg_temp ทางจอภาพ
- อัลกอริทึม(Algorithms)**

Find_avg_temperature

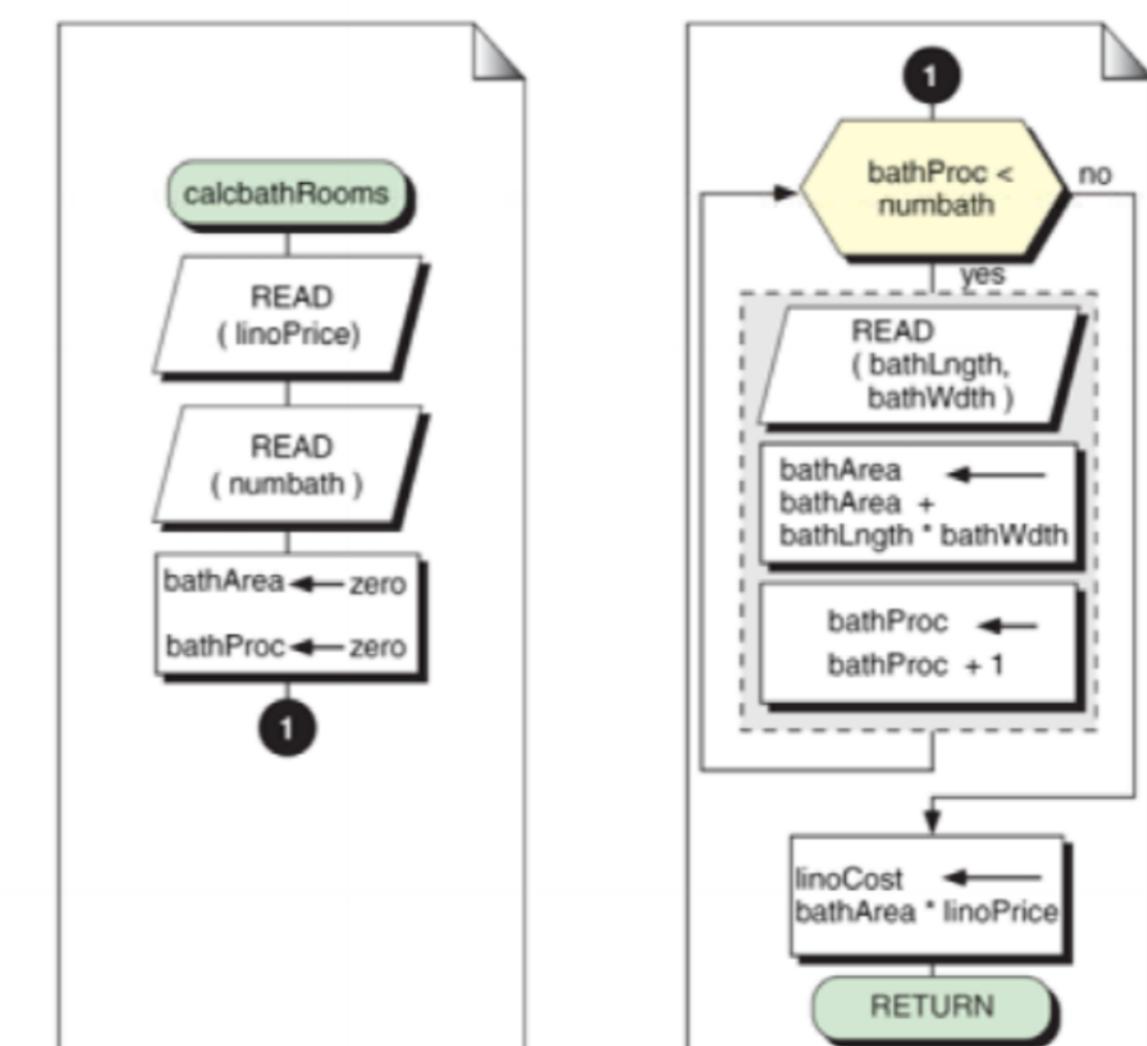
- READ max_temp, min_temp
- $avg_temp = (max_temp+min_temp)/2$
- Output avg_temp to the screen

End

การเขียนอัลกอริทึมของโปรแกรม

Pseudocode for Calculate Bathrooms

```
Algorithm Calculate BathRooms
1 prompt user and read linoleum price
2 prompt user and read number of bathrooms
3 set total bath area and baths processed to zero
4 while ( baths processed < number of bathrooms )
1   prompt user and read bath length and width
2   total bath area =
3   total bath area + bath length * bath width
4   add 1 to baths processed
5 bath cost = total bath area * linoleum price
6 return bath cost
end Algorithm Calculate BathRooms
```



ตัวอย่างการวิเคราะห์ปัญหาและการเขียนโปรแกรมเพื่อช่วยแก้ปัญหา

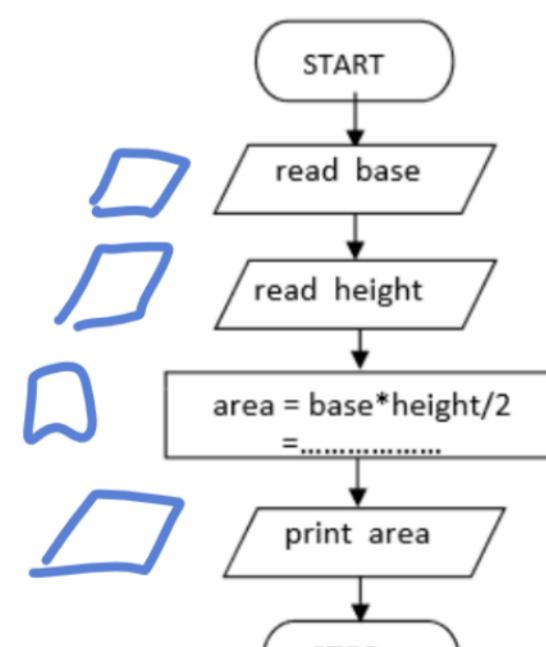
- จะเขียนผังงานและโปรแกรมเพื่อรับข้อมูลตัวเลขจำนวนจริงความยาวฐาน (base) และความสูง (height) ของรูปสามเหลี่ยม แล้วให้ทำการคำนวณพื้นที่และแสดงผลในรูปแบบต่อไปนี้
- Enter base value: 10 (กดแป้น Enter)
- Enter height value: 5 (กดแป้น Enter)
- Area is : 25.000

ตัวอย่างการวิเคราะห์ปัญหาและการเขียนโปรแกรมเพื่อช่วยแก้ปัญหา

- ข้อมูลนำเข้า ความยาวฐาน และความสูง
- แสดงผล พื้นที่
- กำหนดตัวแปร
 - ชื่อตัวแปร ความหมาย
 - base ความยาวฐานของรูปสามเหลี่ยม
 - height ความสูงของรูปสามเหลี่ยม
 - area พื้นที่ของรูปสามเหลี่ยม

ตัวอย่างการวิเคราะห์ปัญหาและการเขียนโปรแกรมเพื่อช่วยแก้ปัญหา

ขั้นตอนการทำงาน



เขียนโปรแกรม

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    float base, height, area;
    printf("Enter base value: "); /* prompt to input base */
    scanf("%f", &base); /* input base */
    printf("Enter height value: "); /* prompt to input height */
    scanf("%f", &height); /* input height */
    area = base * height / 2; /* compute area */
    printf("Area = %7.2f\n", area); /* display result */
    system("PAUSE");
    return 0;
}
```

การเขียนโปรแกรม

- เป็นขั้นตอนของการเขียนโปรแกรม เพื่อให้คอมพิวเตอร์สามารถประมวลผลได้
- การเขียนโปรแกรมจะต้องเขียนตามภาษาที่คอมพิวเตอร์เข้าใจ
- จะใช้ภาษาอะไรดีก็ได้ ซึ่งจะต้องเขียนให้ถูกตามหลักไวยากรณ์ (Syntax) ของภาษาหนึ่ง ๆ

การทดสอบและแก้ไขโปรแกรม

- หลังจากการเขียนโปรแกรมจะต้องทดสอบความถูกต้องของโปรแกรมที่เขียนขึ้นว่ามีข้อผิดพลาดหรือไม่ ซึ่งเรียกว่า ดีบัก (Debug) ซึ่งโดยทั่วไปข้อผิดพลาด (Bug) มี 2 ประเภท คือ
- 1. **Syntax error** คือ การเขียนคำสั่งไม่ถูกต้องตามหลักการเขียนโปรแกรมของภาษาหนึ่ง ๆ กรณีที่เกิด Syntax error โปรแกรมจะไม่สามารถทำงานได้
- 2. **Logic error** เป็นข้อผิดพลาดทางตรรกะ โปรแกรมสามารถทำงานได้แต่ผลลัพธ์จะไม่ถูกต้อง

การทำเอกสารและบำรุงรักษาโปรแกรม

- 1. **คู่มือการใช้** หรือ User Document หรือ User Guide ซึ่งจะเป็นส่วนที่อธิบายการใช้โปรแกรม
- 2. **คู่มือโปรแกรมเมอร์** หรือ Program Document หรือ Technical Document ซึ่งจะทำให้มีความสะดวกในการแก้ไข และพัฒนาโปรแกรมต่อไปในอนาคต

Computer Programming II

การเขียนโปรแกรมคอมพิวเตอร์ 2

LECTURE#2 Introduction to C Language

อ.สติตย์ ประสมพันธ์

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

KMUTNB

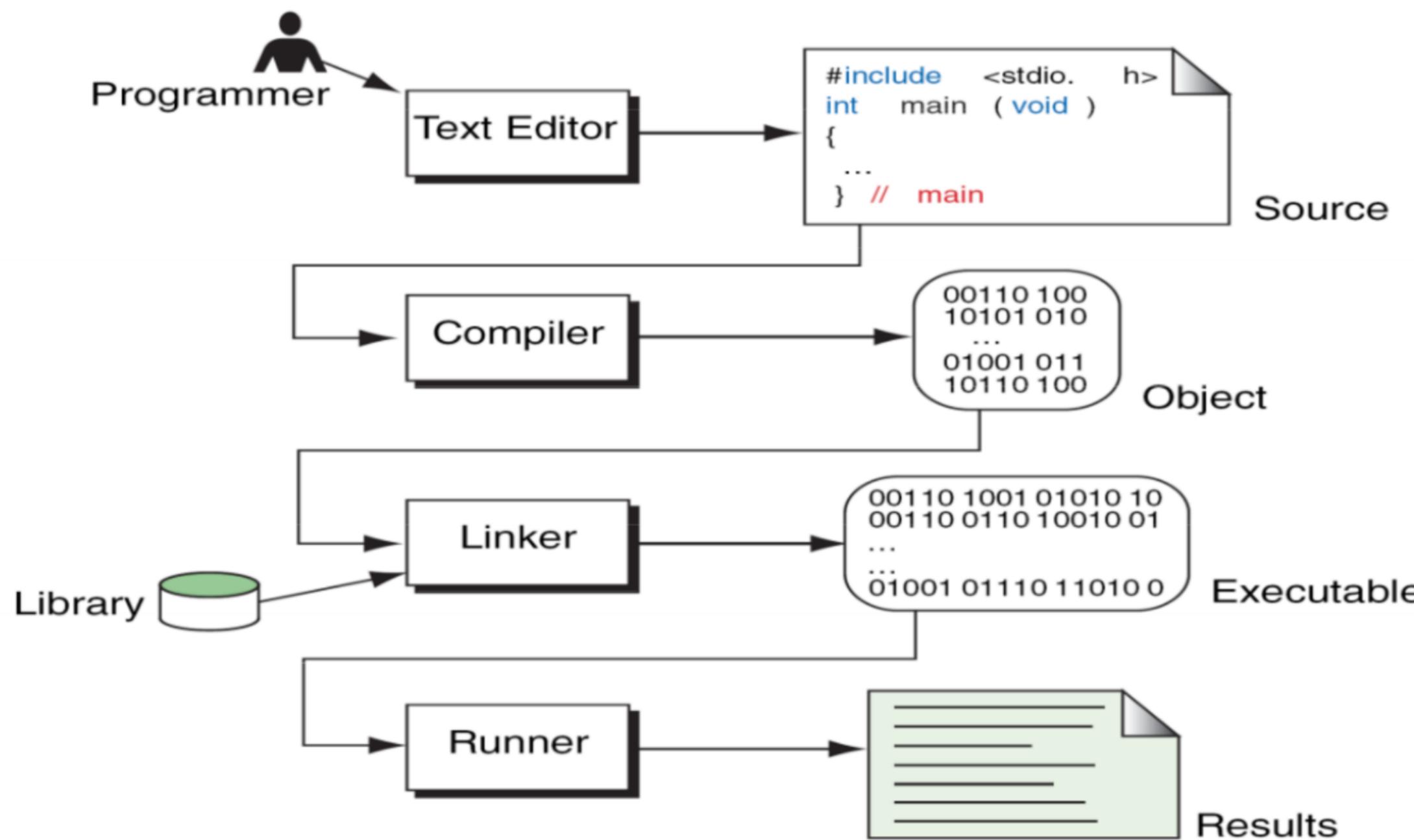
ประวัติความเป็นมาของภาษาซี

- ค.ศ. 1970 มีการพัฒนาภาษา B โดย Ken Thompson ซึ่งทำงานบนเครื่อง DEC PDP-7 ซึ่งทำงานบนเครื่องไมโครคอมพิวเตอร์ไม่ได้ และยังมีข้อจำกัดในการใช้งานอยู่ (ภาษา B สืบทอดมาจากภาษา BCPL ซึ่งเขียนโดย Martin Richards)
- ค.ศ. 1972 Dennis M. Ritchie และ Ken Thompson ได้สร้างภาษา C เพื่อเพิ่มประสิทธิภาพภาษา B ให้ดียิ่งขึ้น ในระยะแรกภาษา C ไม่เป็นที่นิยมแก่นักโปรแกรมเมอร์โดยทั่วไปนัก
- ค.ศ. 1978 Brian W. Kernighan และ Dennis M. Ritchie ได้เขียนหนังสือเล่มหนึ่งชื่อว่า The C Programming Language และหนังสือเล่มนี้ทำให้บุคคลทั่วไปรู้จักและนิยมใช้ภาษา C ใน การเขียนโปรแกรมมากขึ้น แต่เดิมภาษา C ใช้ Run บนเครื่องคอมพิวเตอร์ 8 bit ภายใต้ระบบปฏิบัติการ CP/M ของ IBM PC ซึ่งในช่วงปี ค.ศ. 1981 เป็นช่วงของการพัฒนาเครื่องไมโครคอมพิวเตอร์ ภาษา C จึงมีบทบาทสำคัญในการนำ มาใช้บนเครื่อง PC ตั้งแต่นั้นเป็นต้นมา และมีการพัฒนาต่อมาอีกหลาย ๆ ค่าย ดังนั้นเพื่อกำหนดที่ศูนย์กลางการใช้ภาษา C ให้เป็นไปแนวทางเดียวกัน ANSI (American National Standard Institute) ได้กำหนดข้อตกลงที่เรียกว่า 3J11 เพื่อสร้างภาษา C มาตรฐานขึ้นมา เรียนว่า ANSI C
- ค.ศ. 1983 Bjarne Stroustrup แห่งห้องปฏิบัติการเบล (Bell Laboratories) ได้พัฒนาภาษา C++ ขึ้นรายละเอียดและความสามารถของ C++ มีส่วนขยายเพิ่มจาก C ที่สำคัญ ๆ ได้แก่ แนวความคิดของการเขียนโปรแกรมแบบ OOP (Object Oriented Programming) ซึ่งเป็นแนวคิดการเขียนโปรแกรมที่หมายถึงการพัฒนาโปรแกรมขนาดใหญ่ที่มีความ слับซับขับข้องมาก มีข้อมูลที่ใช้ในโปรแกรมจำนวนมาก จึงนิยมใช้เทคนิคของการเขียนโปรแกรมแบบ OOP ในการพัฒนาโปรแกรมขนาดใหญ่ในปัจจุบันนี้

Computer Programming II

2

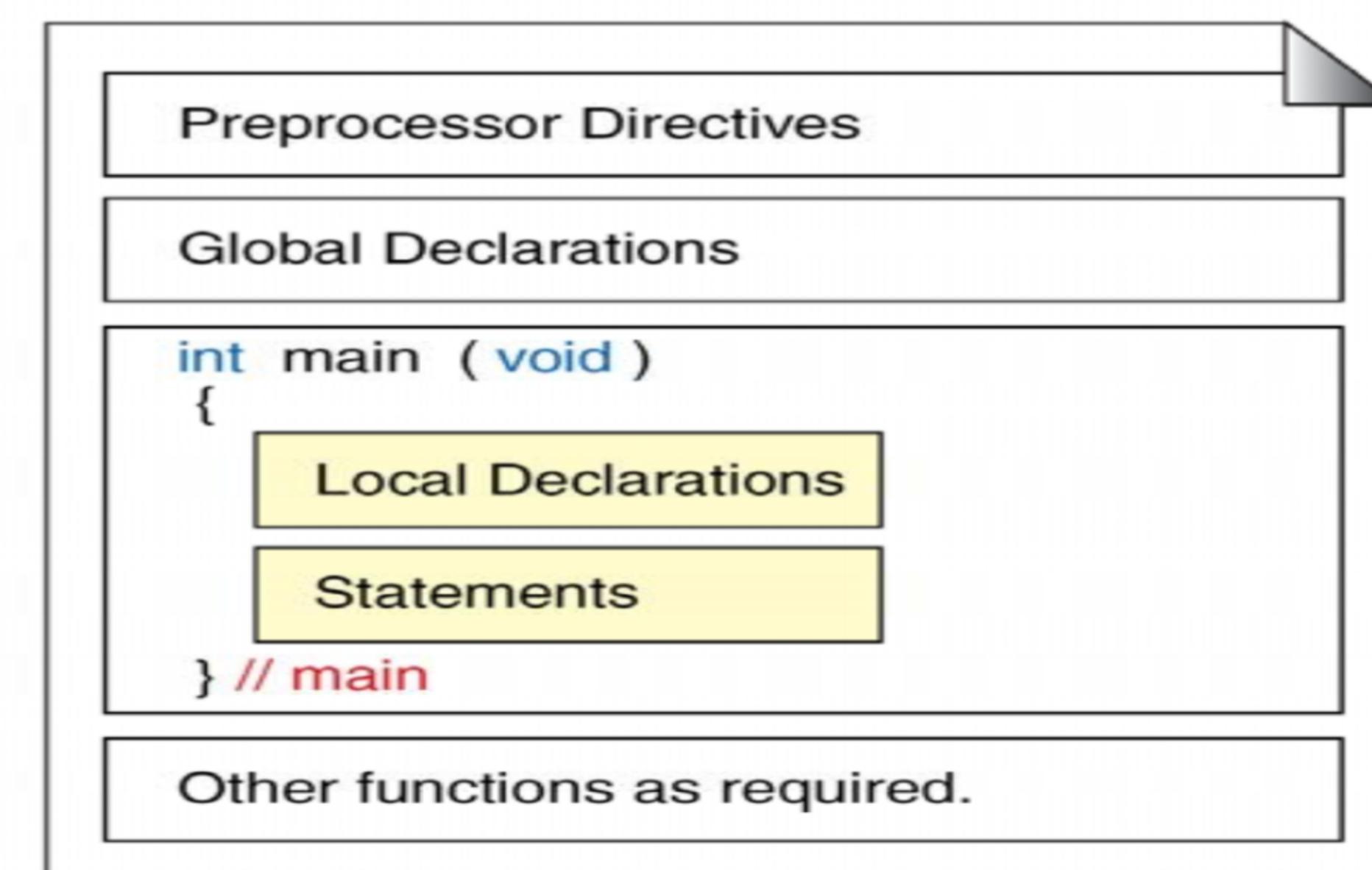
การคอมไพล์และลิงค์โปรแกรมในภาษาซี



Computer Programming II

3

โครงสร้างโปรแกรมภาษาซี



Computer Programming II

4

โครงสร้างและลำดับการเขียนภาษาซี

- คำสั่งตัวประมวลผลก่อน(Preprocessor statement)
- รหัสต้นฉบับ (Source code) มีลำดับการเขียนดังนี้
 - คำสั่งประกาศครอบคลุม (Global declaration statements)
 - ต้นแบบฟังก์ชัน(function prototypes)
 - ฟังก์ชันหลัก(main function) มีฟังก์ชันเดียว
 - ฟังก์ชัน(functions) มีได้หลายฟังก์ชัน
 - คำสั่งประกาศตัวแปรเฉพาะที่(Local declaration statements)
- หมายเหตุ(Comment) สามารถแทรกไว้ที่ใดก็ได้ภายในโปรแกรม

Computer Programming II

5

รหัสต้นฉบับ (Source code)

```
/* This program demonstrates function calls by
calling a small function to multiply two numbers.
Written by:
Date:
*/
#include <stdio.h>
int multiply (int num1, int num2);
int main (void)
{
    int multiplier;
    int multiplicand;
    int product;
    printf("Enter two integers: ");
    scanf ("%d%d", &multiplier, &multiplicand);
    product = multiply (multiplier, multiplicand);
    printf("Product of %d & %d is %d\n",
           multiplier, multiplicand, product);
    return 0;
} // main
/*
=====
Multiples two numbers and returns product.
Pre num1 & num2 are values to be multiplied
Post product returned
*/
int multiply (int num1, int num2)
{
// Statements
    return (num1 * num2);
} // multiply
```

ผลลัพธ์จากการประมวลผลโปรแกรม
Enter two integers: 17 21
Product of 17 & 21 is 357

Computer Programming II

6

โครงสร้างและลำดับการเขียนภาษาซี

```
/* The greeting program. This program demonstrates
some of the components of a simple C program.
Written by: your name here
Date: date program written
*/
#include <stdio.h>

int main (void)
{
// Local Declarations
// Statements

    printf("Hello World!\n");

    return 0;
} // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม
Hello World!

Computer Programming II

7

โครงสร้างและลำดับการเขียนภาษาซี

```
/* This program reads two integers from the keyboard
and prints their product.
Written by:
Date:
*/
#include <stdio.h>

int main (void)
{
// Local Definitions
    int number1;
    int number2;
    int result;

// Statements
    scanf ("%d", &number1);
    scanf ("%d", &number2);
    result = number1 * number2;
    printf ("%d", result);
    return 0;
} // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม
5 25
125

Computer Programming II

8

คำสั่งตัวประมวลผลก่อน (Preprocessor statement)

- ตัวประมวลผลก่อน (Preprocessor) คือ ส่วนที่คอมไพล์จะต้องทำก่อนทำการแปลงโปรแกรม คำสั่งของตัวประมวลผลก่อนจะนำหน้าด้วยเครื่องหมาย # มีคำสั่งต่างๆ ต่อไปนี้

```
#include    #define    #if        #program  
#endif      #error     #ifndef    #undef  
#elif      #else      #ifdef
```

คำสั่งตัวประมวลผลก่อน (Preprocessor statement)

- #include ทำหน้าที่แจ้งให้คอมไпал์ร์อ่านไฟล์อื่นเข้ามาเพลาร่วมด้วย มีรูปแบบดังนี้
`#include <filename>` หรือ `#include "filename"`
เช่น`#include <dos.h>` อ่านไฟล์ dos.h จากไดเร็คทรอรีที่กำหนด`#include "sample.h"` อ่านไฟล์ sample.h จากไดเร็คทรอรีปัจจุบันหรือที่กำหนด`#include "stdio.h"` อ่านไฟล์ stdio.h จากไดเร็คทรอรีปัจจุบันหรือที่กำหนด

คำสั่งตัวประมวลผลก่อน (Preprocessor statement)

- #define ทำหน้าที่ใช้กำหนดค่าคงที่ ที่เป็นชื่อแทน คำนิพจน์ คำสั่ง หรือคำสั่งหลายคำสั่ง มีรูปแบบการใช้งานดังนี้
`#define ชื่อตัวแปร (ชื่อที่ใช้แทน) ค่าที่ต้องการกำหนด`

เช่น

```
#define TWO 2          กำหนดตัวแปร TWO แทนค่า 2  
#define PI 3.141592654 กำหนดตัวแปร PI แทนค่า 3.141592654
```

หมายเหตุ (Comment)

- สามารถแทรกไว้ที่ได้ก็ได้ภายในโปรแกรม ภาษาซึ่นิยมการเขียนข้อความอธิบายการทำงานในส่วนต่างๆ ของโปรแกรมเพื่อให้เข้าใจและอ่านโปรแกรมง่ายขึ้น การเขียนอธิบายจะใช้เครื่องหมาย /* และ */ คร่อมข้อความที่ต้องการอธิบาย
 -ข้อความที่ต้องการอธิบาย.....*/
- แต่ถ้าต้องการเขียนอธิบายหลายบรรทัด จะเขียนได้ดังนี้
`/*
..... ข้อความที่ต้องการอธิบาย.....
.....*/`

```
/* This program demonstrates three ways to use  
constants.  
Written by:  
Date:  
*/  
#include <stdio.h>  
#define PI 3.1415926536  
  
int main (void)  
{  
// Local Declarations  
const double cPi = PI;  
  
// Statements  
printf("Defined constant PI: %f\n", PI);  
printf("Memory constant cPi: %f\n", cPi);  
printf("Literal constant: %f\n", 3.1415926536);  
return 0;  
} // main
```

รูปแบบคำสั่งในภาษาซี

- รูปแบบคำสั่งในภาษาซี มีกฎเกณฑ์ในการเขียนคำสั่ง ดังนี้
 - คำสั่งทุกคำสั่งต้องเขียนด้วยอักษรตัวเล็กเสมอ เช่นคำสั่ง printf , scanf , for
 - ทุกคำสั่งจะใช้เครื่องหมาย ; และการจบของคำสั่ง เช่น printf("Hello");
 - การเขียนคำสั่ง จะเขียนได้แบบอิสระ (Free Format) คือ สามารถเขียนหลายๆคำสั่ง ต่อ กันได้ เช่น
printf("Hello"); printf("C Programming"); f = 3.414;

แต่เพื่อความเป็นระเบียบและอ่านง่าย ควรจะเขียน 1 คำสั่งต่อ 1 บรรทัด

ตัวแปร (Variable)

- ชื่อเรียกแทนพื้นที่เก็บข้อมูลในหน่วยความจำ มีชนิดของข้อมูล หรือแบบของตัวแปรคือ char, int, long, float, double, unsigned int, unsigned long int
- การกำหนดตัวแปร ทำได้ 2 แบบ คือ
 - 1. กำหนดไว้นอกกลุ่มคำสั่ง หรือฟังก์ชัน เรียกตัวแปรนี้ว่า Global Variable กำหนดไว้นอกฟังก์ชัน ใช้งานได้ทั้งโปรแกรม มีค่าเริ่มต้นเป็น 0 (กรณีไม่ได้กำหนดค่าเริ่มต้น)
 - 2. กำหนดไว้ในกลุ่มคำสั่ง หรือฟังก์ชัน เรียกตัวแปรนี้ว่า Local Variable กำหนดไว้ภายในฟังก์ชัน ใช้งานได้ภายในฟังก์ชันนั้น และไม่ถูกกำหนดค่าเริ่มต้นโดยอัตโนมัติ

ชนิดตัวแปร ชื่อตัวแปร , ชื่อตัวแปร, ชื่อตัวแปร,.....;

กฎการตั้งชื่อตัวแปร การตั้งชื่อตัวแปร

- มีข้อกำหนดดังนี้
 - ประกอบด้วย a ถึง z, 0 ถึง 9 และ _ เท่านั้น
 - อักษรตัวแรกต้องเป็น a ถึง z และ _
 - ห้ามใช้ชื่อเฉพาะ
 - ตัวพิมพ์ใหญ่ ตัวพิมพ์เล็ก มีความหมายที่แตกต่างกัน (Case sensitive)
 - ยาวสูงสุดไม่เกิน 31 ตัวอักษร

ตัวอย่างของการตั้งชื่อตัวแปรที่ถูกต้องและไม่ถูกต้อง

Valid Names		Invalid Name	
a	// Valid but poor style	\$sum	// \$ is illegal
student_name		2names	// First char digit
_aSystemName		sum-salary	// Contains hyphen
_Bool	// Boolean System id	stdnt Nmbr	// Contains spaces
INT_MIN	// System Defined Value	int	// Keyword

กลุ่มคำในภาษาซี

คำส่วน (Keywords)

คำที่ภาษาซีกำหนดไว้ก่อนแล้ว เพื่อใช้งาน ได้แก่

auto	default	float	register	struct	volatile	break	void
do	for	extern	const	long	return	static	enum
goto	short	char	int	sizeof	case	while	continue
union	unsigned	typedef	if		else		

กลุ่มคำในภาษาซี

- คำที่ผู้ใช้ตั้งขึ้นใหม่ (User Defines words)
- กลุ่มอักษรที่นิยามขึ้นใช้ในโปรแกรม โดยผู้เขียนโปรแกรมกำหนดขึ้นเอง มีข้อกำหนดดังนี้
 - ตัวอักษรภาษาอังกฤษตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ภาษาซีถือว่าเป็นคนละตัวกัน เช่น Area และ area เป็นตัวแปรคนละตัวกัน
 - ตัวอักษรตัวแรกต้องเป็นตัวอักษรหรือ _ จะเป็นตัวเลขไม่ได้
 - ตัวอักษรที่ไม่ใช่ตัวแรกจะเป็นตัวอักษรหรือ _ หรือตัวเลขก็ได้
 - ก่อนการใช้ชื่อใดๆ ต้องนิยามก่อนเสมอ
 - ห้ามตั้งชื่อซ้ำกับคำส่วน
 - ภายในกลุ่มคำสั่ง สามารถกำหนดชื่อขึ้นใหม่ได้ ซึ่งนั้นจะถูกใช้งานภายในกลุ่มคำสั่ง แลกกลุ่มคำสั่งที่อยู่ลงไปเท่านั้น หากชื่อในกลุ่มคำสั่งไปซ้ำกับที่นิยามไว้ภายนอก จะถือเอาชื่อที่นิยามใหม่เป็นหลัก
 - ความยาวชื่อจะขึ้นอยู่กับตัวแปรในภาษาซี สำหรับโปรแกรม Borland C ได้ 32 ตัวอักษร

ชนิดของข้อมูลในภาษาซี

ชนิด	ขนาดความกว้าง	ช่วงของค่า
char	8 บิต	ASCII character (-128 ถึง 127)
unsigned char	8 บิต	0-255
int	16 บิต	-32768 ถึง 32767
long int	32 บิต	-2147483648 ถึง 2147483649
float	32 บิต	3.4E-38 ถึง 3.4E+38 หรือ ทศนิยม 6 ตำแหน่ง
double	64 บิต	1.7E-308 ถึง 1.7E+308 หรือ ทศนิยม 12 ตำแหน่ง
unsigned int	16 บิต	0 ถึง 65535
unsigned long int	32 บิต	0 ถึง 4294967296

การหาขนาดของชนิดตัวแปรต่าง ๆ จะใช้คำสั่ง sizeof(ประเภทข้อมูล) โดยลำดับขนาดของประเภทข้อมูลเรียงลำดับจากน้อยไปมากมีดังนี้

sizeof (short) ≤ sizeof (int) ≤ sizeof (long) ≤ sizeof (long long)

ค่าคงที่ constant

ตัวอย่างค่าคงที่	ชนิด	ความหมาย
255	decimal int	จำนวนเต็มฐานสิบ
0xFF	hexadecimal int	จำนวนเต็มฐานสิบหก
0377	octal int	จำนวนเต็มฐานแปด
255L หรือ 255I	long int	จำนวนเต็มฐานสิบแบบยาว
255U หรือ	unsigned int	จำนวนเต็มฐานสิบไม่ติดลบ
255u	unsigned long int	เครื่องหมาย
0xFFUL	unsigned long int	เดชฐานสิบหกแบบยาวไม่ติดลบ
		เครื่องหมาย
15.75E2	floating point	เดชทศนิยมแบบยกกำลัง
-1.23	floating point	เดชทศนิยมแบบค่าติดลบ
.123	floating point	เดชทศนิยม
123F	floating point	เดชทศนิยม
'a'	character	ตัวอักษร
""	string	ประโยค ข้อความ

คือ ค่าของข้อมูลที่มีจำนวนแน่นอน
เขียนได้ 3 ลักษณะ คือ
1 ค่าคงที่จำนวนเต็ม เขียนอยู่ในรูป
ตัวอักษร อาจมีเครื่องหมายลบบวกหน้า
2 ค่าคงที่จำนวนจริง เขียนในรูป
ตัวเลขมีทศนิยม
3 ค่าคงที่ที่หมายถึงรหัสตัวอักษร
(ตัวอักษรฐานจำนวนรูปแบบตัวเลข ตาม
รหัส ASCII)

การแสดงผลข้อมูล

- ฟังก์ชัน `printf()` เป็นฟังก์ชันจากคลังที่มาพร้อมกับตัวแปลโปรแกรมภาษาซี ใช้สำหรับการแสดงผล มีรูปแบบดังนี้

```
printf("สายอักขระควบคุม", ตัวแปร);
```

- 1. สายอักขระควบคุมประกอบด้วย 3 ส่วนคือ
 - ตัวอักขระที่จะแสดง
 - รูปแบบการแสดงผล ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์(%)
 - ลำดับหลีก (escape sequence)
- 2. ตัวแปร คือชื่อของตัวแปรที่จะแสดงผล (format specifies)

การกำหนดรูปแบบการแสดงผล

ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์(%) ตามด้วยอักขระ 1 ตัว หรือหลายตัวโดยที่อักขระมีความหมายดังนี้

อักขระ	ชนิดข้อมูล	รูปแบบการแสดงผล
c	char	อักขระเดียว
d	int	จำนวนเต็มฐานสิบ
o	int	จำนวนเต็มฐานแปด
x	int	จำนวนเต็มฐานสิบหก
f	float	จำนวนที่มีทศนิยมในรูปฐานสิบ

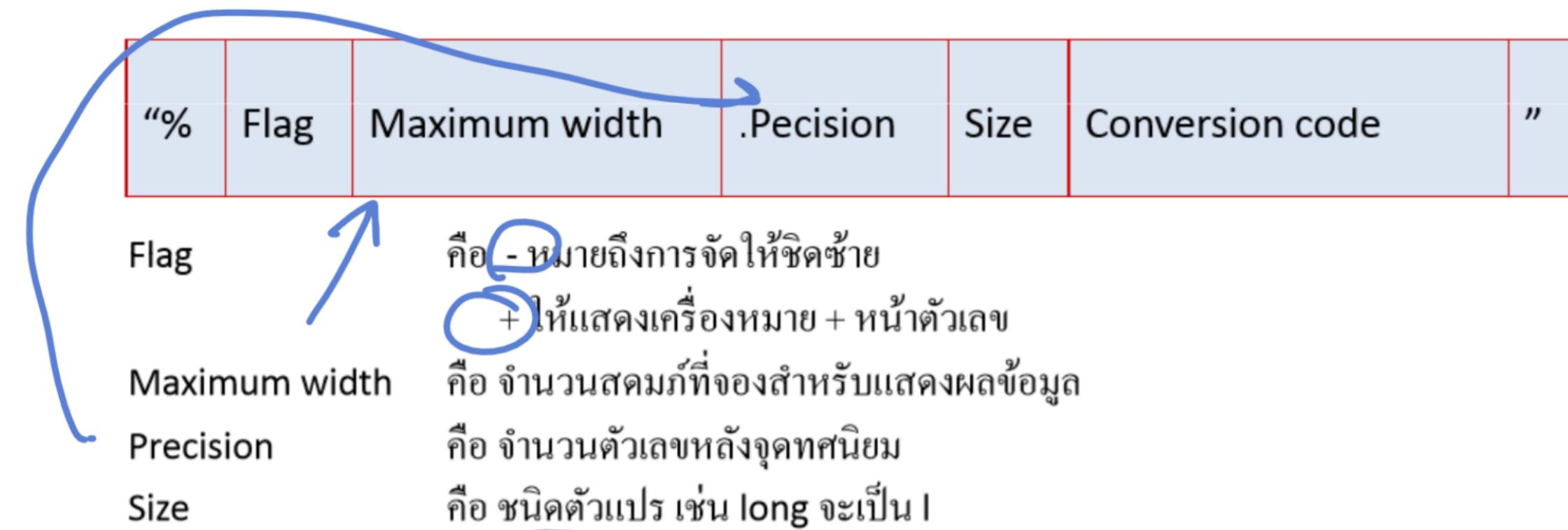
ลำดับหลีก(Escape sequence)

ลำดับหลีก	ผลการกระทำ
\n	ขึ้นบรรทัดใหม่ (new line)
\t	เลื่อนไปยังจุดตั้งระยะ tab ต่อไป
\a	เสียงกระดิ่ง (bell)
\b	ถอยไปหนึ่งที่ว่าง (backspace)
\f	ขึ้นหน้าใหม่ (form feed)
\	แสดงเครื่องหมายทับกลับหลัง (backslash)
'	แสดงเครื่องหมายผนಥง (single quote)
"	แสดงเครื่องหมายพื้นหนู (double quote)

การแสดงค่าของตัวแปร

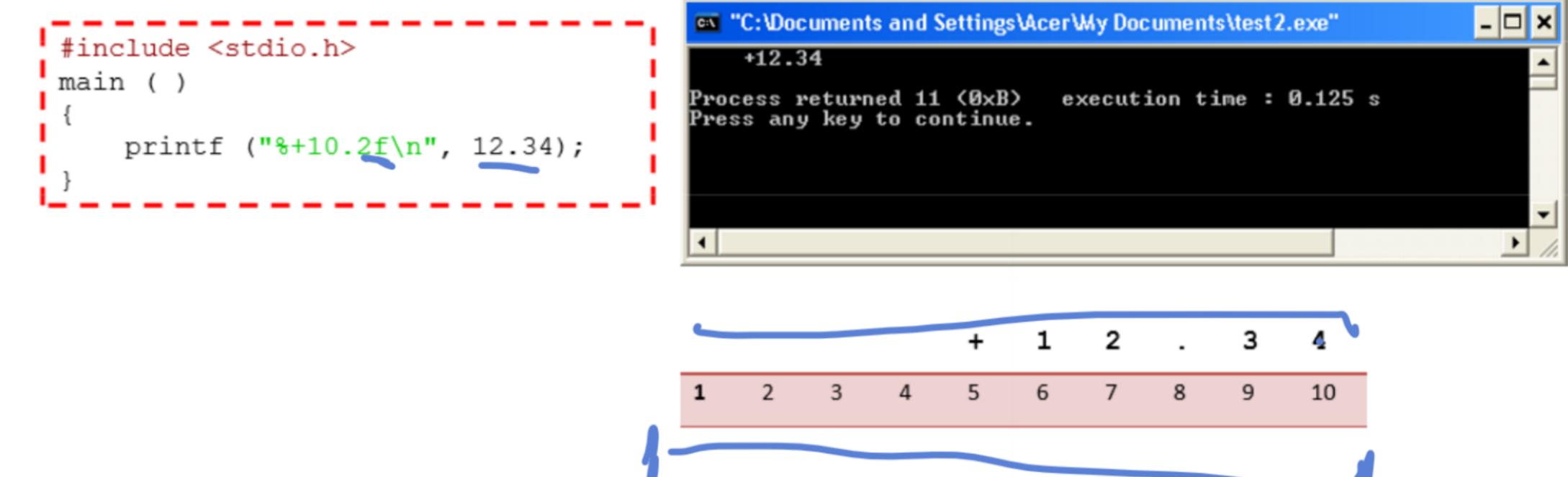
ชนิดข้อมูล	ชนิดตัวแปร	รูปแบบสานหับ printf
จำนวนเต็ม	short integer long unsigned short unsigned int unsigned long	%hd หรือ %hi %d หรือ %i %ld หรือ %li %hu %u %lu
จำนวนจริง	float double	%f %lf
อักขระ	char	%c
สายอักขระ	char s[]	%s
เลขฐาน	จำนวนเต็มฐาน 10 จำนวนเต็มฐาน 8 จำนวนเต็มฐาน 16	%d %o %x
	แสดงเครื่องหมาย %	%%

การจัดรูปแบบผลลัพธ์ printf()



การจัดรูปแบบผลลัพธ์ printf()

"	%	+	10	.	2		f	"
---	---	---	----	---	---	--	---	---



การจัดรูปแบบผลลัพธ์ printf()

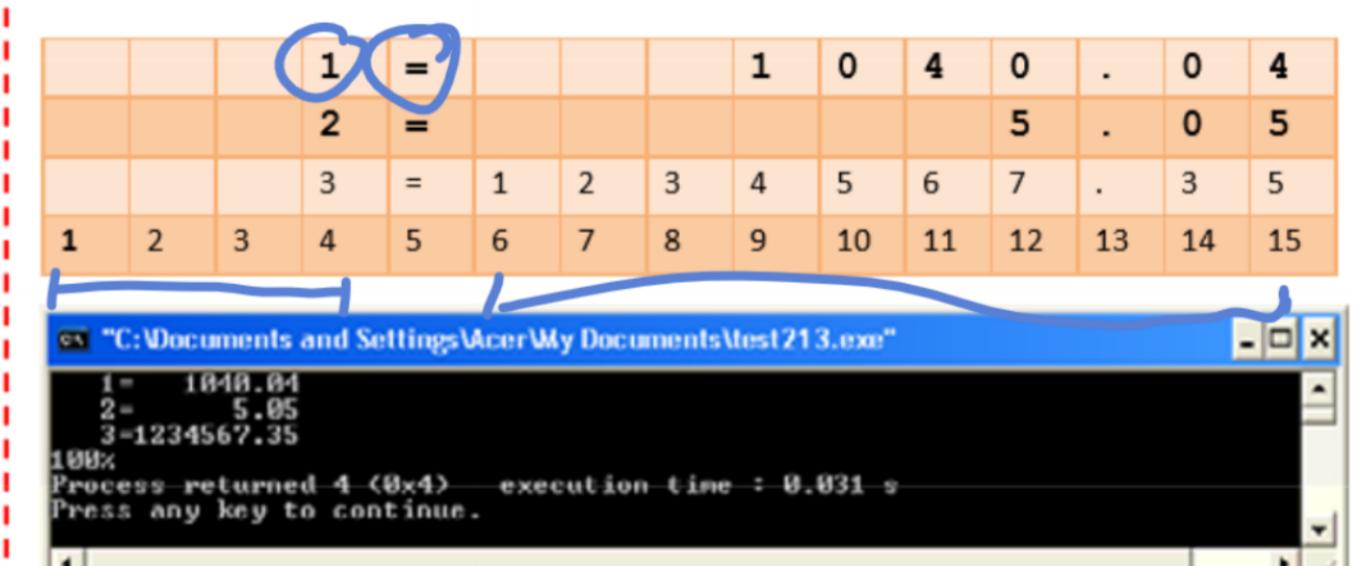
Conversion code คือ การกำหนดชนิดข้อมูลในการแสดงผล

- %d : พิมพ์ int ด้วยตัวเลขฐานสิบ
- %o : พิมพ์ int ด้วยตัวเลขฐานแปด
- %x : พิมพ์ int ด้วยตัวเลขฐานสิบหก
- %f : พิมพ์ float, double แบบจุดทศนิยม (หากตำแหน่ง)
- %e : พิมพ์ float, double แบบวิทยาศาสตร์ เช่น 1.23e+23
- %c : พิมพ์ char ตัวอักษร 1 ตัว
- %s : พิมพ์ข้อความ เช่น "Hello"

การจัดรูปแบบผลลัพธ์ printf()

```
#include <stdio.h>
main ()
{
  int a;
  double b;

  a=1;
  b=1040.041;
  printf ("%4i=%10.2f\n", a, b);
  a=2;
  b=5.05;
  printf ("%4i=%10.2f\n", a, b);
  a=3;
  b=1234567.351;
  printf ("%4i=%10.2f\n", a, b);
  printf("100%%");
}
```



การรับค่าข้อมูล

- ฟังก์ชัน `scanf()`
- เป็นฟังก์ชันจากคลังใช้ในการรับข้อมูลจากแป้นพิมพ์ โดยจะบอกเลขที่อยู่ของตัวแปรในหน่วยความจำ และจึงนำค่าที่รับมาไปเก็บไว้ตามที่อยู่นั้น ฟังก์ชัน `scanf()` มีรูปแบบดังนี้
`scanf("%รูปแบบ", &ตัวแปร);`
- โดยที่ `&` ตัวแปร หมายถึงเลขที่อยู่ (address) ของตัวแปรที่จะรับค่ามาเก็บในหน่วยความจำ

การรับค่าข้อมูล

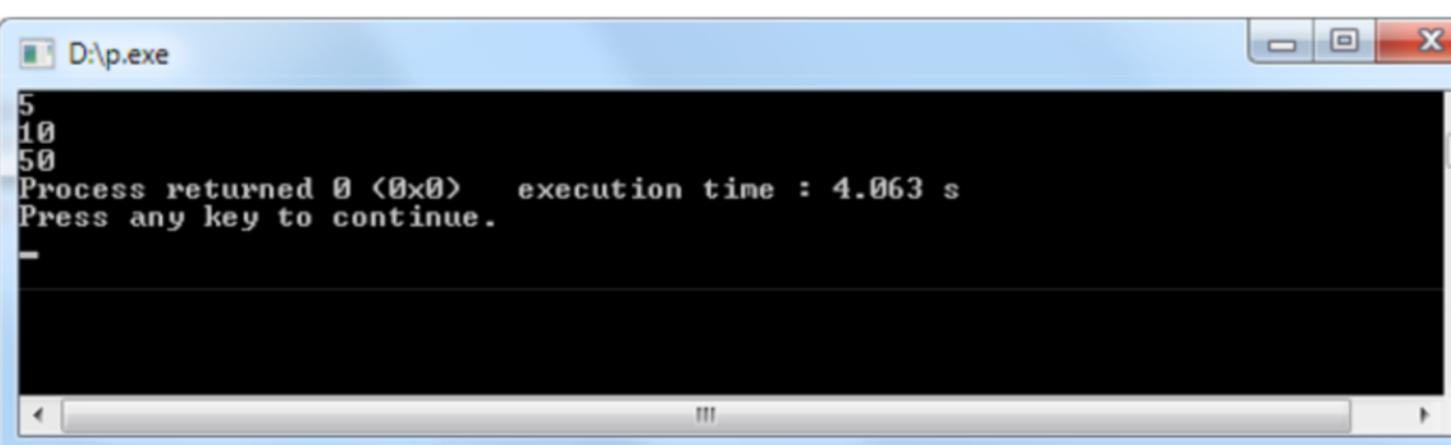
- การรับค่า `scanf()`
ฟังก์ชัน `scanf` เป็นการรับข้อมูลจากแป้นพิมพ์ ต้อง `#include <stdio.h>` จึงจะสามารถใช้คำสั่ง `scanf` ได้
 - รูปแบบ
`scanf("%รูปแบบ", &ตัวแปร);`
 - เช่น การรับข้อมูลชนิด `int` และไปเก็บไว้ในตัวแปร `age`
 - รูปแบบ
`scanf("%d", &age);`
 - สามารถรับข้อมูลที่ละหลายตัวแปรได้
 - รูปแบบ
`scanf("%os %of", name, &GPAX);`
- การรับข้อมูล `int float char` ต้องมีเครื่องหมาย `&` เติม `string` ไม่ต้องมี `&`

การรับค่าข้อมูล

```
/* This program reads two integers from the keyboard
and prints their product.
Written by:
Date:
*/
#include <stdio.h>

int main (void)
{
// Local Definitions
    int number1;
    int number2;
    int result;

// Statements
    scanf ("%d", &number1);
    scanf ("%d", &number2);
    result = number1 * number2;
    printf ("%d", result);
    return 0;
} // main
```



คำอธิบาย
#include <stdio.h> คือการบอกคอมไพล์เลอร์ให้นำไฟล์ stdio.h มารวมด้วย
main คือชื่อของฟังก์ชัน โปรแกรมจะเริ่มทำงานที่นี่ และเมื่อจบฟังก์ชัน main หมายถึงจบโปรแกรมด้วย
scanf ("%d", &number1); คือการใช้ฟังก์ชัน scanf รับค่าจากผู้ใช้ที่ป้อนในตัวแปร number1
result = number1 * number2; คือ ผลคูณของตัวเลขที่อยู่ในตัวแปร number1 และ number2 และ
เก็บลงในตัวแปร result
printf ("%d", result); คือการใช้ฟังก์ชัน printf พิมพ์พื้นความที่อยู่ในเครื่องหมาย "" ออกทางอุปกรณ์
เอาท์พุตมาตรฐาน
return 0 คือ การคืนค่าไปยังโปรแกรมที่เรียกว่าฟังก์ชัน main() ซึ่งจากตัวอย่างเป็นฟังก์ชัน main() ดังนั้น
return 0 จึงหมายถึงการจบการทำงานของโปรแกรม

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- ฟังก์ชัน `getchar()` เป็นฟังก์ชันที่ทำงานกับตัวอักษรโดยเฉพาะ
- getchar() เป็นฟังก์ชันที่ใช้รับข้อมูลเข้ามาทางแป้นพิมพ์ที่ละ 1 ตัวอักษร โดยต้องกด enter ทุกครั้งเมื่อสิ้นสุดข้อมูล และข้อมูลที่ป้อนจะปรากฏให้เห็นบนหน้าจอภาพด้วย
- รูปแบบ

ชื่อตัวแปร =getchar();

```
#include<stdio.h>
main ( )
{
    char ch;
    ch=getchar ( );
}
```

- ตัวอย่าง เครื่องจะรอรับข้อมูลจากแป้นพิมพ์ที่ผู้ใช้ป้อน จำนวน 1 ตัวอักษรเก็บไว้ในตัวแปร ch หลังจากที่ผู้ใช้ต้องกดปุ่ม enter เพื่อให้ฟังก์ชันรับค่าข้อมูล

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- ฟังก์ชัน getch()

getch() เป็นฟังก์ชันที่ใช้รับข้อมูลที่เป็นตัวอักษร 1 ตัว เข้ามาทางแป้นพิมพ์ โดยเมื่อป้อน ข้อมูลเสร็จ ไม่ต้องกดปุ่ม enter และอักษรที่ป้อนเข้ามาจะไม่ปรากฏบนจอภาพ ต้อง #include <conio.h> จึงจะสามารถใช้ฟังก์ชัน getch() ได้

- รูปแบบ

ชื่อตัวแปร =getch();

```
#include <stdio.h>
main ( )
{
    char x;
    x = getch( );
}
```

- ตัวอย่าง เครื่องจะรอรับข้อมูลจากแป้นพิมพ์เข้ามา 1 ตัวเพื่อนำมาเก็บไว้ในตัวแปร x โดยผู้ใช้เมื่อต้องกด enter หลังจากที่ป้อนข้อมูลเสร็จแล้ว

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- ฟังก์ชัน gets()

เป็นฟังก์ชันที่ใช้รับข้อมูลที่เป็นชื่อความ (ตัวอักษรจำนวนหนึ่ง) จากแป้นพิมพ์เข้ามาเก็บไว้ในตัวแปร gets มาจากคำว่า get string

- รูปแบบ

gets (ชื่อตัวแปร);

```
#include <stdio.h>
main ( )
{
    char name[10];
    gets(name);
}
```

โดยรับค่าชื่อความจากแป้นพิมพ์ ฟังก์ชันจะทำการใส่ '\0' เอาไว้ที่ตัวสุดท้ายของชื่อความ เพื่อแสดงการสิ้นสุดของชื่อความที่รับเข้ามาเมื่อผู้ใช้กดปุ่ม enter

เครื่องจะจดที่ตัวแปรชุดที่ชื่อ name ซึ่งเป็นอักษรละ 1 ตัว และรอรับค่าที่เป็นชื่อความเข้ามาเก็บไว้ในตัวแปรชุดที่ชื่อ name ได้ยากไม่เกิน 9 ตัวอักษรเพื่อให้ name ตัวที่ 10 (ตัวสุดท้าย) เก็บ \0 เอาไว้

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- ฟังก์ชัน putchar()

เป็นฟังก์ชันที่ใช้ให้คอมพิวเตอร์ แสดงผลบนจอภาพที่ละ 1 ตัวอักษร

- รูปแบบ

putchar (ชื่อตัวแปร);

```
#include<stdio.h>
main ( )
{
    char x;
    x=getch();
    printf ("The result is \n");
    putchar ( x );
}
```

ลักษณะข้อมูลที่ป้อน

A .

ผลลัพธ์

The result is

A

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- ฟังก์ชัน puts()

เป็นฟังก์ชันที่ใช้แสดงผลข้อมูลที่เป็น ชื่อความที่เก็บไว้ในตัวแปรชุดอักษรบน จอภาพ puts() มาจากคำว่า put string ใช้สำหรับพิมพ์สตริงอักขระจากภาพ

- รูปแบบ

puts (ชื่อตัวแปร);

```
#include <stdio.h>
main ( )
{
    char name [10];
    gets(name);
    puts(name);
}
```

ตัวอย่าง เครื่องจะนำค่าที่เก็บในตัวแปรชุด name มาแสดงผลบนจอภาพ

ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

- ฟังก์ชัน **getche()**
- getche()** เป็นฟังก์ชันในการรับข้อมูล 1 ตัวอักษรโดยจะปรากฏตัวอักษรให้เห็นในการป้อนข้อมูล โดยไม่ต้องกด Enter การใช้ **getche()** จะต้องทำการ `#include <conio.h>` จึงจะสามารถใช้ **getche()** ได้
- รูปแบบ
ชื่อตัวแปร = **getche();**

รหัส Ascii

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000	000	NUL (null)	32	20 040	040	 	Space	64	40 100	100	@	Ø	96	60 140	140	`	`
1	1 001	001	SOH (start of heading)	33	21 041	041	!	!	65	41 101	101	A	A	97	61 141	141	a	a
2	2 002	002	STX (start of text)	34	22 042	042	"	"	66	42 102	102	B	B	98	62 142	142	b	b
3	3 003	003	ETX (end of text)	35	23 043	043	#	#	67	43 103	103	C	C	99	63 143	143	c	c
4	4 004	004	EOT (end of transmission)	36	24 044	044	$	\$	68	44 104	104	D	D	100	64 144	144	d	d
5	5 005	005	ENQ (enquiry)	37	25 045	045	%	%	69	45 105	105	E	E	101	65 145	145	e	e
6	6 006	006	ACK (acknowledge)	38	26 046	046	&	&	70	46 106	106	F	F	102	66 146	146	f	f
7	7 007	007	BEL (bell)	39	27 047	047	'	'	71	47 107	107	G	G	103	67 147	147	g	g
8	8 010	010	BS (backspace)	40	28 050	050	((72	48 110	110	H	H	104	68 150	150	h	h
9	9 011	011	TAB (horizontal tab)	41	29 051	051))	73	49 111	111	I	I	105	69 151	151	i	i
10	A 012	012	LF (NL line feed, new line)	42	2A 052	052	*	*	74	4A 112	112	J	J	106	6A 152	152	j	j
11	B 013	013	VT (vertical tab)	43	2B 053	053	+	+	75	4B 113	113	K	K	107	6B 153	153	k	k
12	C 014	014	FF (NP form feed, new page)	44	2C 054	054	,	,	76	4C 114	114	L	L	108	6C 154	154	l	l
13	D 015	015	CR (carriage return)	45	2D 055	055	-	-	77	4D 115	115	M	M	109	6D 155	155	m	m
14	E 016	016	SO (shift out)	46	2E 056	056	.	.	78	4E 116	116	N	N	110	6E 156	156	n	n
15	F 017	017	SI (shift in)	47	2F 057	057	/	/	79	4F 117	117	O	O	111	6F 157	157	o	o
16	10 020	020	DLE (data link escape)	48	30 060	060	0	0	80	50 120	120	P	P	112	70 160	160	p	p
17	11 021	021	DC1 (device control 1)	49	31 061	061	1	1	81	51 121	121	Q	Q	113	71 161	161	q	q
18	12 022	022	DC2 (device control 2)	50	32 062	062	2	2	82	52 122	122	R	R	114	72 162	162	r	r
19	13 023	023	DC3 (device control 3)	51	33 063	063	3	3	83	53 123	123	S	S	115	73 163	163	s	s
20	14 024	024	DC4 (device control 4)	52	34 064	064	4	4	84	54 124	124	T	T	116	74 164	164	t	t
21	15 025	025	NAK (negative acknowledgement)	53	35 065	065	5	5	85	55 125	125	U	U	117	75 165	165	u	u
22	16 026	026	SYN (synchronous idle)	54	36 066	066	6	6	86	56 126	126	V	V	118	76 166	166	v	v
23	17 027	027	ETB (end of trans. block)	55	37 067	067	7	7	87	57 127	127	W	W	119	77 167	167	w	w
24	18 030	030	CAN (cancel)	56	38 070	070	8	8	88	58 130	130	X	X	120	78 170	170	x	x
25	19 031	031	EM (end of medium)	57	39 071	071	9	9	89	59 131	131	Y	Y	121	79 171	171	y	y
26	1A 032	032	SUB (substitute)	58	3A 072	072	:	:	90	5A 132	132	Z	Z	122	7A 172	172	z	z
27	1B 033	033	ESC (escape)	59	3B 073	073	;	;	91	5B 133	133	[[123	7B 173	173	{	{
28	1C 034	034	FS (file separator)	60	3C 074	074	<	<	92	5C 134	134	\	\	124	7C 174	174	|	
29	1D 035	035	GS (group separator)	61	3D 075	075	=	=	93	5D 135	135]]	125	7D 175	175	})
30	1E 036	036	RS (record separator)	62	3E 076	076	>	>	94	5E 136	136	^	^	126	7E 176	176	~	~
31	1F 037	037	US (unit separator)	63	3F 077	077	?	?	95	5F 137	137	_	_	127	7F 177	177		DEL

Source: www.LookupTables.com

Computer Programming II การเขียนโปรแกรมคอมพิวเตอร์ 2 LECTURE#3 Operator & Expression

อ.สิติย์ ประสมพันธ์

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

KMUTNB

คำสั่งและนิพจน์ (Statement and Expression)

คำสั่ง, ข้อคำสั่ง (Statement) คือขั้นตอนในการทำงานหนึ่งขั้นตอนทุกคำสั่งต้องจบด้วยเครื่องหมาย ;

กลุ่มคำสั่ง คือคำสั่งที่อยู่ในวงเล็บปีกๆ { }

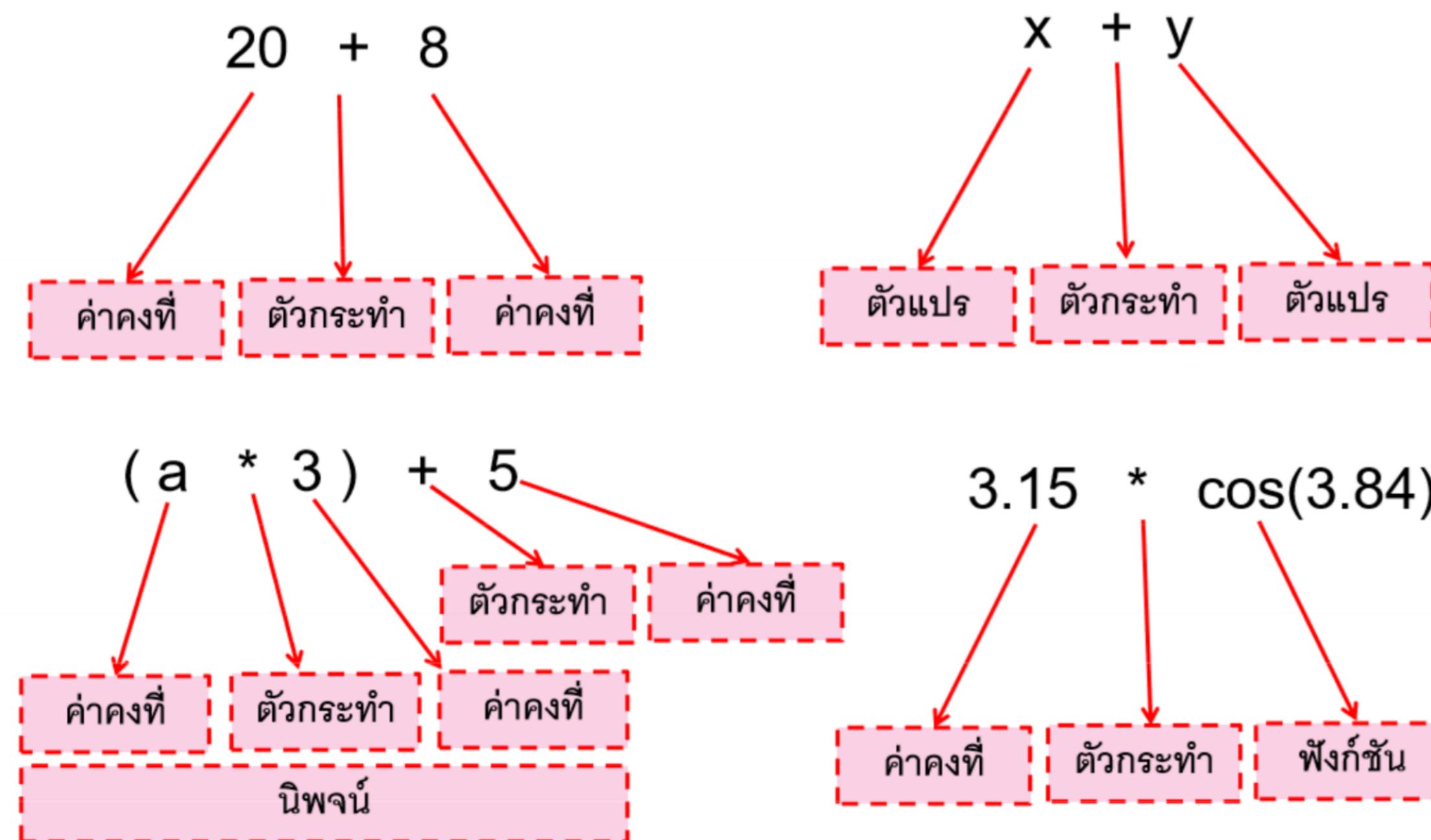
นิพจน์ (Expression) คือการกระทำเพื่อให้ได้ผลลัพธ์ค่าหนึ่งค่าประกอบไปด้วยตัวถูกกระทำ (Operands) และตัวกระทำ (Operators) เขียนเรียงกันไป

เช่น $3 * 2 - 1 + 7$ หรือ $a * 5$ เป็นต้น

Computer Programming II

2

ตัวอย่างการเขียนนิพจน์คณิตศาสตร์และนิพจน์ภาษา C



นิพจน์

- นิพจน์คณิตศาสตร์ (Arithmetic Expression) หมายถึง การนำตัวแปรค่าคงที่มาสัมพันธ์กันโดยใช้เครื่องหมายคณิตศาสตร์เป็นตัวเชื่อมผลที่ได้จากนิพจน์แบบนี้จะเป็นตัวเลข
- นิพจน์ตรรกะ (Logical Expression) หมายถึงการนำตัวแปรค่าคงที่หรือนิพจน์มาสัมพันธ์กันโดยใช้เครื่องหมายเปรียบเทียบและเครื่องหมายตรรกะเป็นตัวเชื่อม ผลที่ได้จะออกมาระบุเป็นจริงหรือเท็จ

Computer Programming II

3

Computer Programming II

4

กฎเกณฑ์การเขียนนิพจน์

- 1. ห้ามเขียนตัวแปร 2 ตัวติดกันโดยไม่มีเครื่องหมายเช่น ab
- 2. ถ้าเขียน นิพจน์โดยมีค่าของตัวแปร หรือค่าคงที่ต่างชนิดกันในนิพจน์เดียวกันภาษาซีจะเปลี่ยนชนิดของข้อมูลที่มีขนาดเล็กให้เป็นชนิดข้อมูลที่มีขนาดใหญ่ขึ้น

Computer Programming II

5

เครื่องหมายดำเนินการ (Operators)

- กำหนดการกระทำที่เกิดขึ้นกับตัวแปรและค่าคงที่ โดยที่นิพจน์ประกอบด้วยตัวแปร และค่าคงที่ และใช้ตัวดำเนินการคำนวณเพื่อให้ได้ค่า ได้แก่
 - ตัวดำเนินการทางคณิตศาสตร์
 - ตัวดำเนินการสัมพันธ์และตัวดำเนินการตรรกะ
 - ตัวดำเนินการประกอบ
 - ตัวดำเนินการระดับบิต
 - ดำเนินการบอกชนิดตัวแปร (type) และตัวดำเนินการบอกขนาดหน่วยความจำของตัวแปร (size of) เป็นต้น

Computer Programming II

6

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

เครื่องหมาย	ความหมาย	ตัวอย่าง
+	การบวก	$A+B$
-	การลบ	$A-B$
*	การคูณ	$A*B$
/	การหาร	A/B
%	การหารเอาแต่เศษไว้ (Modulo)	5%3=1 เศษ 2 จะเก็บแต่เศษ 2 เอาไว้
--	การลดค่าลงครั้งละ 1	$A--$ จะเหมือนกับ $A=A-1$
++	การเพิ่มค่าขึ้นครั้งละ 1	$A++$ จะเหมือนกับ $A=A+1$

Computer Programming II

7

การดำเนินการแบบ Unary

เครื่องหมาย	ความหมาย	ตัวอย่าง
+	เป็นตัวดำเนินการที่ระบุค่าบวกสำหรับ Operand	$c=+a$
-	เป็นตัวดำเนินการที่ระบุค่าลบสำหรับ Operand	$c=-a$
++	การเพิ่มค่าให้ Operand 1 ค่า	$++a$
--	การลดค่าให้ Operand 1 ค่า	$--a$

- Prefix unary operator จะคำนวน unary operator ก่อน binary operator
- Postfix unary operator จะคำนวน binary operator ก่อน

Computer Programming II

8

การดำเนินการแบบ Unary

```
#include <stdio.h>
void main()
{
    int i, j, N;
    i=10; j= 2; N=i*++j;           3
    printf("N=i* ++j=%d, i=%d, j=%d\n", N, i, j);
    i=10; j= 2; N=i* j++;         20 30 10 3
    printf("N=i* j++=%d, i=%d, j=%d\n", N, i, j);
    i=10; j= 2; N=i* --j;          10 10 1
    printf("N=i* --j=%d, i=%d, j=%d\n", N, i, j);
    i=10; j= 2; N=i* j--;          20 10 1
    printf("N=i* j--=%d, i=%d, j=%d\n", N, i, j);
}
```

การทำงานของตัวดำเนินการ

- การโอเปอเรชัน (Operation) หรือการทำงานของตัวดำเนินการต้องเป็นไปตามลำดับของการวางแผนค่าคงที่หรือตัวแปรที่จะกระทำต่อกัน ซึ่งเรียกว่า ลำดับของการโอเปอเรชัน ซึ่งลำดับการทำงานของโอเปอเรเตอร์โดยคอมไพเลอร์ภาษา C ต้องเป็นไปตามกฎการกำหนดลำดับการทำงานของตัวดำเนินการ (Precedence)

② ลำดับการทำงานก่อน-หลังของตัวดำเนินการ

ตัวดำเนินการ	ทิศทางการดำเนินการ
0, [], ->	ซ้ายไปขวา
, ~, ++, --, +(ค่าบวก), -(ค่าลบ), *, &(type), sizeof	ขวาไปซ้าย
*, /, %	ซ้ายไปขวา
+, - (ตัวกระทำการทางคณิตศาสตร์)	ซ้ายไปขวา
<<, >>	ซ้ายไปขวา
<, <=, >, >=	ซ้ายไปขวา
==, !=	ซ้ายไปขวา
&	ซ้ายไปขวา
^	ซ้ายไปขวา
	ซ้ายไปขวา
&&	ซ้ายไปขวา
	ซ้ายไปขวา
? :	ขวาไปซ้าย
=, +=, -=, /=, %=, ^=, =, <<=, >>=	ขวาไปซ้าย
,	ซ้ายไปขวา

กฎการกำหนดลำดับการทำงานของตัวดำเนินการ (Precedence)

โอเปอเรเตอร์	ลำดับของการโอเปอเรชัน
()	ซ้ายไปขวา
-	ซ้ายไปขวา
แสดงความเป็น ลบของตัวเลข	
* /	ซ้ายไปขวา
+ -	ซ้ายไปขวา

$$z = p * r \% q + w / x - y$$

6 1 2 4 3 5

$$z = a * b + c / d - e$$

5 1 3 2 4

การกำหนดลำดับการทำงานของตัวดำเนินการ (Precedence)

$$y = a * x * x + b * x + c;$$

Step 1. $y = 2 * 5 * 5 + 3 * 5 + 7;$ (Leftmost multiplication)
2 * 5 is 10

Step 2. $y = 10 * 5 + 3 * 5 + 7;$ (Leftmost multiplication)
10 * 5 is 50

Step 3. $y = 50 + 3 * 5 + 7;$ (Multiplication before addition)
3 * 5 is 15

Step 4. $y = 50 + 15 + 7;$ (Leftmost addition)
50 + 15 is 65

Step 5. $y = 65 + 7;$ (Last addition)
65 + 7 is 72

Step 6. $y = 72$ (Last operation—place 72 in y)

การทำงานของตัวดำเนินการ

- การคำนวณ $a + b * c$

ภาษา C จะกระทำการคูณก่อน加 แต่ถ้าเครื่องหมายมีความเท่าเทียมกันจะกระทำการบวกก่อน เช่น $a+b+c$ จะเริ่มทำการบวก a กับ b แล้วจึงนำค่าที่ได้มาบวก c ถ้าต้องการกำหนดลำดับให้ทำการใส่เครื่องหมายวงเล็บในส่วนที่ต้องการให้ทำงานก่อน

- ตัวดำเนินการ %

$$\% \quad 10.5$$

เป็นตัวดำเนินการทางคณิตศาสตร์เพียงตัวเดียวที่กำหนดให้ใช้กับค่าจำนวนเต็ม เท่านั้น นอกนั้นสามารถใช้กับค่าอื่นๆ ได้

- เครื่องหมาย /

$$/ \quad 10.5$$

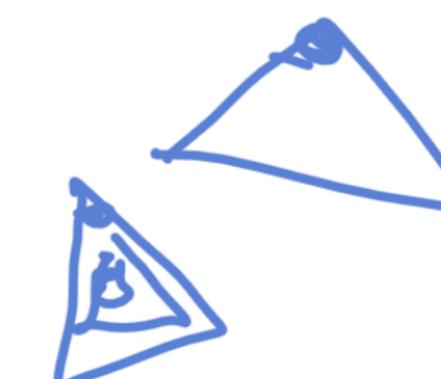
a / b ถ้าทั้งตัวแปร a และ b เป็น integer ค่าที่ได้จากการดำเนินการของหารจะมีชนิดเป็น integer ถ้า a หรือ b ตัวแปรใดตัวแปรหนึ่งเป็นตัวแปรชนิด float จะได้คำตอบเป็นชนิด float เช่น

$$39 / 5 \text{ จะมีค่าเท่ากับ } 7$$

$$39.0 / 5 \text{ จะมีค่าเท่ากับ } 7.8$$

ตัวดำเนินการเพิ่ม และลดค่า (Increment & Decrement)

- ตัวดำเนินการเพิ่ม และลดค่า เป็นตัวดำเนินการในการเพิ่มหรือลดค่าของตัวแปร โดย
 - ตัวดำเนินการเพิ่มค่า $++$ จะบวกหนึ่งเข้ากับตัวถูกดำเนินการ
 - ตัวดำเนินการลดค่า $--$ จะลบหนึ่งออกจากตัวถูกดำเนินการ
- ตัวดำเนินการเพิ่ม และลดค่ามีวิธีใช้งาน 2 แบบคือ
 - ใช้เป็นตัวดำเนินการแบบมาก่อน (prefix) เช่น $++n$
 - ใช้ตัวดำเนินการแบบตามหลัง (postfix) ก็ไดเช่น $n++$
- ในทั้งสองกรณี ผลลัพธ์คือการเพิ่มค่า 1 ให้กับ n
 - แต่ถ้า $++n$ จะเพิ่มค่าก่อนที่จะนำค่าไปใช้
 - ขณะที่ $n++$ จะเพิ่มค่าหลังจากนำค่าไปใช้ ซึ่งหมายความว่าถ้ามีการนำค่าไปใช้ (ไม่เพียงหัวและท้ายผลลัพธ์) $++n$ และ $n++$ จะแตกต่างกัน



ตัวดำเนินการเพิ่ม และลดค่า (Increment & Decrement)

ตัวอย่าง

ถ้า n มีค่า 5

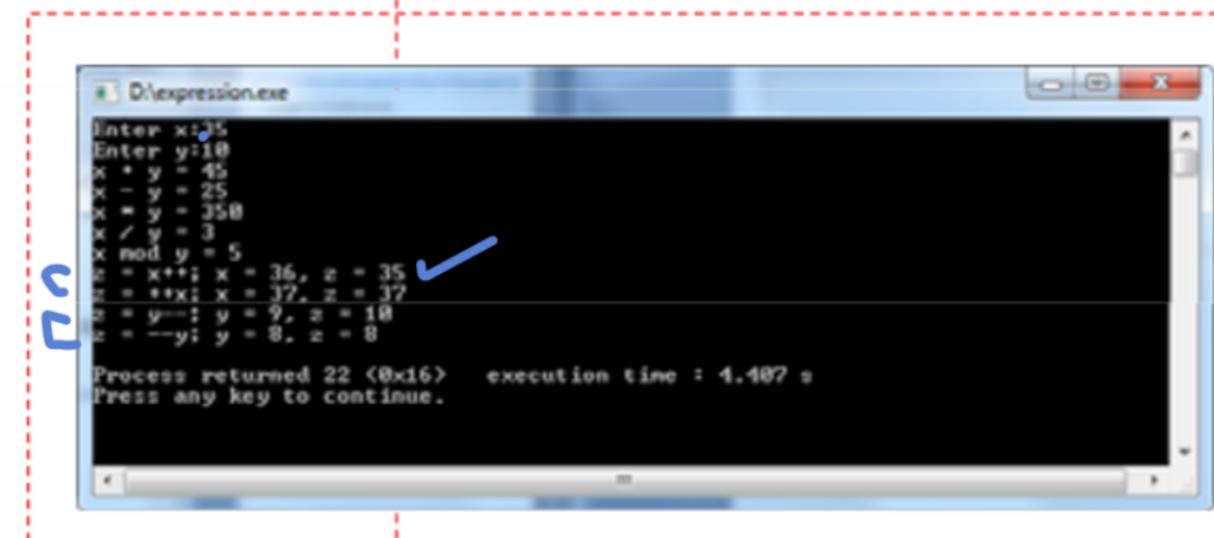
5 $x = n++;$ จะเป็นการกำหนดค่า 5 ให้กับ x

6 $x = ++n;$ จะเป็นการกำหนดค่า 6 ให้กับ x

ตัวดำเนินการนี้จะใช้ได้กับเฉพาะตัวแปรเท่านั้น จะใช้กับนิพจน์อื่นๆ ไม่ไดเช่น $(i+j)++$

การทำงานของตัวดำเนินการ

```
#include<stdio.h>
main ( )
{
    int x,y,z;
    printf("Enter x:");
    scanf("%d", &x); 35
    printf("Enter y:");
    scanf("%d", &y); 10
    printf("x + y = %d\n", x+y);
    printf("x - y = %d\n", x-y);
    printf("x * y = %d\n", x*y);
    printf("x / y = %d\n", x/y);
    printf("x mod y = %d\n", x%y);
    z=x++;
    printf("z = x++; x = %d, z = %d\n", x,z);
    z=++x;
    printf("z = ++x; x = %d, z = %d\n", x,z);
    z=y--;
    printf("z = y--; y = %d, z = %d\n", y,z);
    z=--y;
    printf("z = --y; y = %d, z = %d\n", y,z);
}
```



ตัวดำเนินการสัมพันธ์และตัวดำเนินการตรรกะ (Relational, Equality, and Logical Operators)

Operator	เครื่องหมาย	ความหมาย
ตัวดำเนินการสัมพันธ์ (Relational Operator)	<	น้อยกว่า
	>	มากกว่า
	<=	น้อยกว่า หรือ เท่ากับ
	>=	มากกว่า หรือ เท่ากับ
ตัวดำเนินการเท่ากับ (Equality Operator)	==	เท่ากับ
	!=	ไม่เท่ากับ
ตัวดำเนินการตรรกะ (Logical Operator)	!	นิเสธ
	&&	และ
		หรือ

ตัวดำเนินการสัมพันธ์ (Relational Operator)

- ตัวดำเนินการสัมพันธ์ เป็นเครื่องหมายที่ใช้ในการเปรียบเทียบและตัดสินใจ ซึ่งผลของการเปรียบเทียบจะเป็นได้ 2 กรณีเท่านั้นคือ จริง และเท็จ

- ค่าทางตรรกะจริงและเท็จมี ชนิดเป็น int ดังนั้นผลการกระทำทางตรรกะจึงมีค่าเป็นจำนวนเต็ม และมีค่าได้เพียงสองค่าคือ 1 หรือตัวเลขใดๆ แทนค่าความจริงเป็นจริง และ 0 แทนค่าความจริงเป็นเท็จ (0 เป็นเท็จ ส่วนตัวเลขอื่นๆ ทั้งหมดมีค่าเป็นจริง) เครื่องหมายที่เป็นตัวดำเนินการสัมพันธ้มี 4 ตัวคือ

< (น้อยกว่า) > (มากกว่า) <= (น้อยกว่าเท่ากับ) และ >= (มากกว่าเท่ากับ)

ตัวดำเนินการเท่ากับ (Equality Operator)

ใช้ในการเปรียบเทียบค่า 2 ค่า ว่ามีค่าเท่าหรือไม่เท่ากัน ผลลัพธ์ที่ได้ จะมีเพียง 2 ค่าคือ จริง และ เท็จ เครื่องหมายที่เป็นตัวดำเนินการเท่ากับมี 2 ตัวคือ

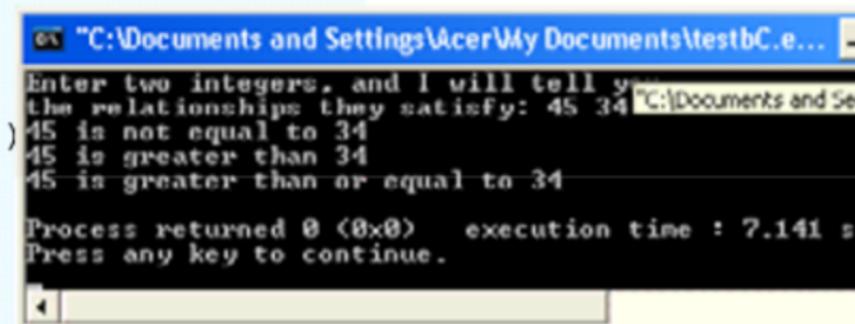
== (เท่ากับ) และ != (ไม่เท่ากับ)

ตัวอย่างการใช้งาน	คำอธิบาย
c == 'A'	/* ถ้าตัวแปร c เก็บค่าข้อ Raz A ผลลัพธ์จะได้ค่าจริง */
k != -2	/* ถ้าตัวแปร k เก็บค่าตัวเลข -2 ผลลัพธ์จะได้ค่าเท็จ */
x+y == 2 * z - 5	
a=b ✗	/* การเปรียบเทียบค่าตัวแปร a กับ b ว่ามีค่าเท่ากันหรือไม่ ถูกจัดการให้ค่าตัวแปร a ด้วยค่าตัวแปร b */
a= =b -1 ✗	/* ใช้งานผิดรูปแบบ โดยมีช่องว่างระหว่างเครื่องหมาย = ทั้ง 2 ตัว */

ตัวดำเนินการสัมพันธ์ (Relational Operator)

```
#include <stdio.h>
int main( void )
{
    int num1; /* first number to be read from user */
    int num2; /* second number to be read from user */
    printf( "Enter two integers, and I will tell you\n" );
    printf( "the relationships they satisfy: " );
    scanf( "%d %d", &num1, &num2 ); /* read two integers */
    if ( num1 == num2 ) {
        printf( "%d is equal to %d\n", num1, num2 );
    } /* end if */
    if ( num1 != num2 ) {
        printf( "%d is not equal to %d\n", num1, num2 );
    } /* end if */
    if ( num1 < num2 ) {
        printf( "%d is less than %d\n", num1, num2 );
    } /* end if */
    if ( num1 > num2 ) {
        printf( "%d is greater than %d\n", num1, num2 );
    } /* end if */
    if ( num1 <= num2 ) {
        printf( "%d is less than or equal to %d\n", num1, num2 );
    } /* end if */
    if ( num1 >= num2 ) {
        printf( "%d is greater than or equal to %d\n", num1, num2 );
    } /* end if */
    return 0; /* indicate that program ended successfully */
} /* end function main */

```



ตัวดำเนินการตรรกะ (Logical Operator)

ใช้ในการเปรียบเทียบ และกระทำการตรรกะกับค่าตัวเลข หรือค่าที่อยู่ในตัวแปร ผลลัพธ์ที่ได้ จะมีเพียง 2 ค่าคือ จริง และ เท็จ

เครื่องหมาย		ตัวถูกกระทำ	ตัวอย่างการใช้งานพิດแบบ	ตัวอย่างการใช้งานที่ถูกต้อง
! (นิสัย)	การกลับค่า ความจริง ของค่า ตัวเลข หรือตัวแปร	ต้องการตัวถูก กระทำการเพียง 1 ตัว	a! a = b	!a !(x+7.7) !(a < b c < d)
&& (และ)		ต้องการตัวถูก กระทำ 2 ตัว	a && a & b &b	a && b !(a < b) && C /* นำ a มาเปรียบเทียบกับ b แล้วนำ ผลลัพธ์มาอินเสอร์ จากนั้นมา && กับ C */b
(หรือ)		ต้องการตัวถูก กระทำ 2 ตัว	a b	a b

ตัวดำเนินการตรรกะ (Logical Operator)

11.2

ตัวดำเนินการ &&

P	Q	P && Q
0	0	0
0	1	0
1	0	0
1	1	1

11.3

ตัวดำเนินการ ||

P	Q	P Q
0	0	0
0	1	1
1	0	1
1	1	1

11.4

ตัวดำเนินการ !

P	!P
0	1
1	0

ตัวดำเนินการประกอบ (Compound Operator)

ตัวดำเนินการประกอบ คือ ตัวดำเนินการที่เป็นรูปแบบย่อของตัวดำเนินการ+ตัวแปรที่ถูกดำเนินการ ดังนี้ += -= /= %= <=> |= ^=

ตัวอย่าง

- | | | |
|---------------------|-----------------------|----------------|
| i = i + 1; | ตัวดำเนินการประกอบคือ | i += 1; |
| i = i - a; | ตัวดำเนินการประกอบคือ | i -= a; |
| i = 1 * (a + 1); | ตัวดำเนินการประกอบคือ | i *= a+1; |
| i = i / (a-b); | ตัวดำเนินการประกอบคือ | i /= a-b; |
| i = i % 101; | ตัวดำเนินการประกอบคือ | i %= 101; |
| i = i << 1; | ตัวดำเนินการประกอบคือ | i <<= 1; |
| i = i >> i; | ตัวดำเนินการประกอบคือ | i >>= i; |
| i = i & 01; | ตัวดำเนินการประกอบคือ | i = 0xf; |
| i = i & 01; | ตัวดำเนินการประกอบคือ | i = 0xf; |
| i = i ^ (07 0xb); | ตัวดำเนินการประกอบคือ | i ^= 07 0xb; |

ตัวกระทำอ กขนาด (Sizeof Operator)

- เป็นตัวกระทำใช้รายงานขนาดของหน่วยความจำที่ใช้ในการเก็บค่า โดยมีรูปแบบดังนี้

sizeof ค่าคงที่ หรือ sizeof (แบบของตัวแปร)

ตัวอย่าง

```
int ABC;  
printf("%d", sizeof(ABC));
```

ผลลัพธ์

2
จะเห็นว่ามีการพิมพ์ค่าขนาดของตัวแปรชื่อ ABC ที่มีชนิดเป็น integer ออกรหานาจว่าค่า 2 คือขนาดของ integer ทั่วไปนั่นเอง

ตัวดำเนินการแบบมีเงื่อนไข (Conditional Operator)

- ตัวกระทำสำหรับเลือกค่า เป็นตัวกระทำใช้ในการเลือกการให้ค่าของนิพจน์ ตัวกระทำนินี้ประกอบไปด้วยเครื่องหมาย ? และ นิพจน์เลือกค่า (Condition Expressions) จะเขียนอยู่ในรูป

นิพจน์1 ? นิพจน์2 : นิพจน์3

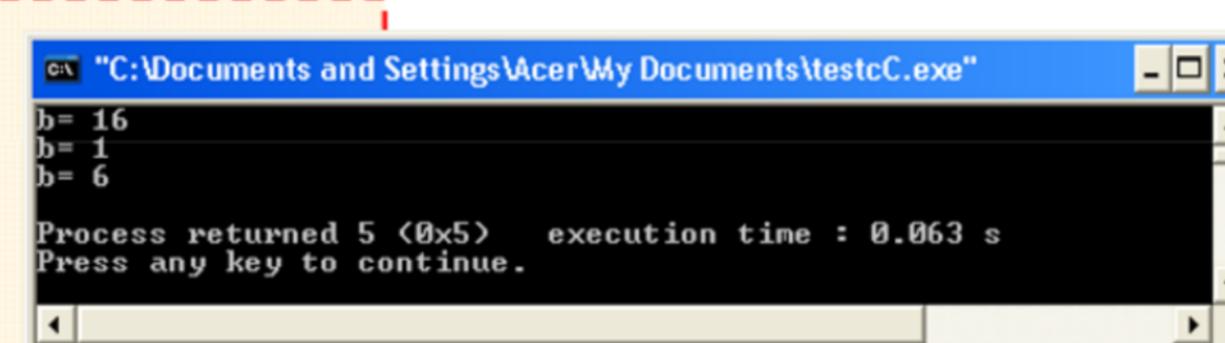
นิพจน์เลือกค่ามีการให้ค่าดังนี้คือ หากค่าในนิพจน์1 มีค่าไม่เท่ากับ 0 (เป็นจริง) จะให้ค่านิพจน์เป็นไปตามนิพจน์ที่ 2 แต่ถ้าค่าในนิพจน์1 เป็น 0 จะให้ค่านิพจน์เป็นไปตามนิพจน์ที่ 3 เช่น

$c = ((a+5)? (a+1) : 0);$

สมมติให้ค่า a เป็น 1 ซึ่งทำให้นิพจน์ a+5 เป็นจริง ก็จะให้ค่า c เป็น 2 ซึ่งได้จาก a+1 และถ้าสมมติให้ a เป็น -5 ก็จะทำให้นิพจน์ a+5 เป็นเท็จ c ก็จะได้รับค่า 0 ดังตัวอย่างต่อไปนี้

ตัวดำเนินการแบบมีเงื่อนไข (Conditional Operator)

```
#include<stdio.h>  
int main(void)  
{  
    int a,b;  
    a = 15;  
    b = ((a+5)? (a+1) : 0);  
    printf("b= %d\n",b);  
    a=0;  
    b=((a+5)?(a+1) : 0);  
    printf("b= %d\n",b);  
    a=5;  
    b=((a+5)?(a+1) : 0);  
    printf("b= %d\n",b);  
}
```



การเปลี่ยนแปลงค่าผลลัพธ์เป็นตัวแปรชนิดใหม่ (Casting)

- ผลของการกระทำของนิพจน์จะให้ค่าอ กมาค่าหนึ่งเสมอ ค่าที่ได้จะมีชนิดสอดคล้อง กับตัวกระทำ และตัวถูกกระทำภายในนิพจน์นั้น ๆ เช่น ตัวถูกกระทำเป็น int ค่าที่ได้จะเป็นชนิด int ด้วย
- เรารอจะเปลี่ยนชนิดของค่า ให้มีชนิดตามที่เราต้องการได้โดยการเขียนชนิดของข้อมูลแบบใหม่ ภายในวงเล็บ นำหน้านิพจน์นั้น ๆ

(แบบข้อมูลแบบใหม่) นิพจน์

การเปลี่ยนแปลงค่าผลลัพธ์เป็นตัวแปรชนิดใหม่ (Casting)

- ตัวอย่าง

(int)(a*5.2)

เป็นการเปลี่ยนค่า output เป็นข้อมูลชนิด int

(float)(b+5)

เป็นการเปลี่ยนค่า output เป็นข้อมูลชนิด float

การเปลี่ยนแปลงค่าผลลัพธ์เป็นตัวแปรชนิดใหม่ (Casting)

- ตัวอย่าง เขียนโปรแกรมสำหรับการคำนวนเงินฝากพร้อมดอกเบี้ยเมื่อเวลาผ่านไป 1 ปี ที่คำนวนเงินได้แบบดอกเบี้ยทบต้น และแสดงผลลัพธ์จากการคำนวนด้วยเลขทศนิยม 2 ตำแหน่ง เมื่อ

รายได้จากเงินฝาก = เงินต้น * $(1 + อัตราดอกเบี้ย)^{\text{ปี}}$

#include <stdio.h>

void main() {

int iR;

float M, R, Income;

printf("Enter Invested Money(M) : ");

scanf("%f", &M); *10000*

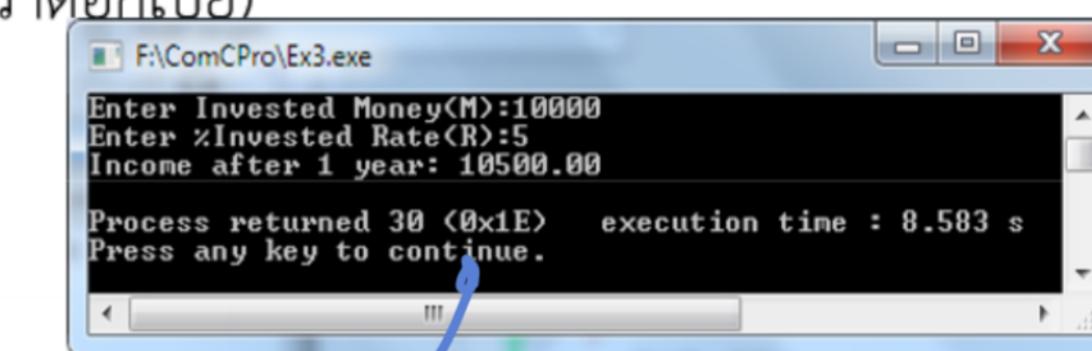
printf("Enter %Invested Rate(R) : ");

scanf("%d", &iR); *5*

R=(float)iR/100; *5/100*

Income=M*(1+R); *10000 * 1.05*

printf("Income after 1 year: %.2f\n", Income);



ตัวดำเนินการระดับบิต(bitwise operator)

- โอเปอเรเตอร์ระดับบิต(bitwise operator) คือ โอเปอเรเตอร์ที่นำค่าแต่ละบิตของโอเปอเรนด์ 2 ตัวมากระทำกันหรืออาจเป็นโอเปอเรเตอร์ที่กระทำการบวกค่าในระดับบิตของโอเปอเรนด์เดียวกันได้

ตัวดำเนินการ	ความหมาย
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
>>	Shift Right
<<	Shift Left

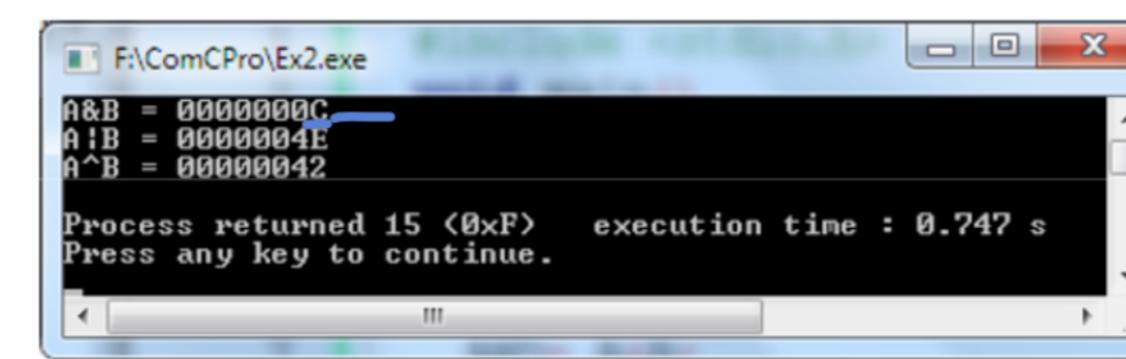
ตัวดำเนินการระดับบิต(bitwise operator)

A	B	A&B	A B	$\neg A$	$A \wedge B$
1	1	1	1	0	0
1	0	0	1	0	1
0	1	0	1	1	1
0	0	0	0	1	0

ตัวดำเนินการระดับบิต(bitwise operator)

- ตัวอย่าง

```
#include <stdio.h>
void main()
{
    unsigned long A, B, AND, OR, XOR;
    A=0X0E; .
    B=0X4C; .
    AND= A&B;
    printf("A&B = %.8X\n",AND);
    OR= A|B;
    printf("A|B = %.8X\n",OR);
    XOR= A^B;
    printf("A^B = %.8X\n",XOR);
}
```



$$16 \rightarrow 2 \\ 2 \& 2 \rightarrow 16$$

การดำเนินการแบบ Bit Shift

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
<<	การดำเนินการเลื่อนค่าของตัวแปรไปทางซ้าย	V>>1
>>	การดำเนินการเลื่อนค่าของตัวแปรไปทางขวา	V<<4

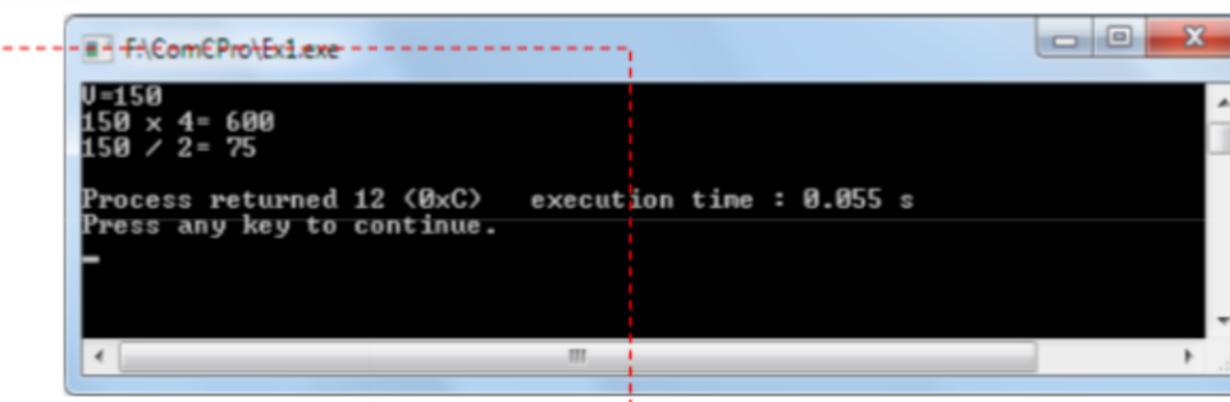
```
#include <stdio.h>
void main()
```

```
{
```

```
int V=150;
printf("V=%d\n", V);
printf("%d x 4= %d\n", V, V<<2);
printf("%d / 2= %d\n", V, V>>1);
```

$$150 \times 4 = 600 \\ 1001011000$$

$$600 \div 2 = 300 \\ 1001011000 \downarrow \\ 10010110$$



Computer Programming II การเขียนโปรแกรมคอมพิวเตอร์ 2 LECTURE#4 Control Statement

อ.สติตย์ ประสมพันธ์
ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ
KMUTNB

คำสั่งควบคุม (Control Statement)

- โดยปกติการทำงานของคอมพิวเตอร์จะทำงานเรียงลำดับคำสั่งลงมาตั้งแต่ต้นโปรแกรมจนจบโปรแกรม
- ถ้าต้องการเปลี่ยนแปลงขั้นตอนการทำงานของคำสั่ง เช่น กระโดดข้ามไปที่ คำสั่งใดคำสั่งหนึ่ง หรือให้วนกลับมาทำคำสั่งที่เคยทำไปแล้ว ลักษณะการสั่งงานแบบนี้จะต้องใช้คำสั่งควบคุม
- คำสั่งควบคุมจึงเป็นคำสั่งที่ใช้เปลี่ยนแปลงลำดับขั้นตอนการทำงานของโปรแกรม

Computer Programming II

2

คำสั่งควบคุม

- คำสั่งควบคุมแบ่งออกเป็น 3 ชนิด คือ
 - คำสั่งให้ไปทำงานโดยไม่มีเงื่อนไข (Unconditional branch Statement) ได้แก่ คำสั่ง goto
 - คำสั่งให้ไปทำงานโดยมีเงื่อนไข (Condition Statement) ได้แก่ คำสั่ง if, switch
 - คำสั่งให้ไปทำงานแบบเป็นวงจร (Loop Control Statement) ได้แก่ คำสั่ง while, do while, for

รูปแบบของคำสั่งให้ไปทำงานโดยมีเงื่อนไข

คำสั่ง if ใช้สำหรับการตัดสินใจเลือกทำงานอย่างใดอย่างหนึ่งโดยใช้เงื่อนไขเป็นส่วนช่วยในการตัดสินใจเลือก

ไวยากรณ์
If (expression) {
Statement;
}
หรือ
If (expression) {
Statement;
}
else{
Statement;
}

โดยที่

Expression เป็นนิพจน์ที่จะต้องให้ผลลัพธ์เป็น true หรือ false เท่านั้น

Statement อาจเป็นคำสั่งเพียงคำสั่งเดียว เช่น

printf("Hello world");

คำสั่ง if

การใช้คำสั่ง if โดยไม่มี else (*If statements without ELSE*) คือ การตัดสินใจว่าจะทำคำสั่นถ้าเงื่อนไขเป็นจริง ถ้าไม่เป็นจริงก็ไม่ทำ

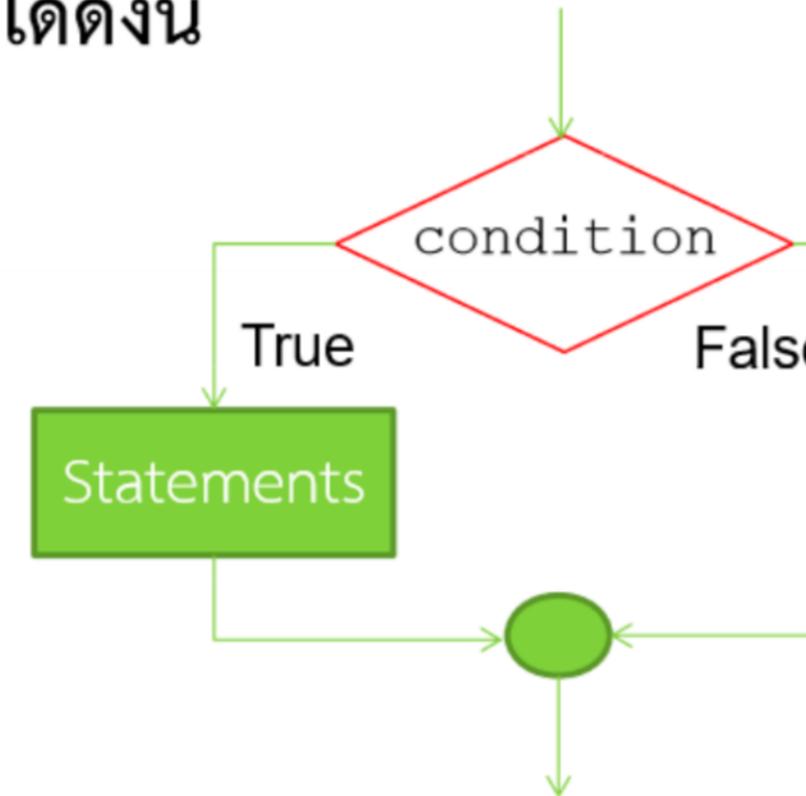
รูปแบบที่ง่ายที่สุดของประโยค if แสดงได้ดังนี้

```
if ( condition )
    statements;
```

โดยที่ condition คือเงื่อนไขสำหรับตัดสินใจ

Statements คือชุดคำสั่งที่จะถูกทำเมื่อ

เงื่อนไขใน condition เป็นจริง



คำสั่ง if

```
#include <stdio.h>
int main()
{
    int score;
    scanf("%d", &score);
    if (score >= 40)
        printf("Congratulations! You Passed \n");
    printf("See you again \n");
}
```

คำสั่ง if-else

เมื่อมีทางเลือก 2 ทางเลือก ใช้คำสั่ง if-else ตัดสินใจเพื่อเลือกการทำงานอย่างใดอย่างหนึ่ง โดยคำสั่ง if-else จะทำการทดสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงให้ทำงานอย่างหนึ่ง แต่ถ้าเงื่อนไขเป็นเท็จจะให้เลือกทำงานอีกอย่างหนึ่ง

รูปแบบที่ง่ายที่สุดของประโยค if แสดงได้ดังนี้

```
if (condition)
    statements_1;
else
    statements_2;
```

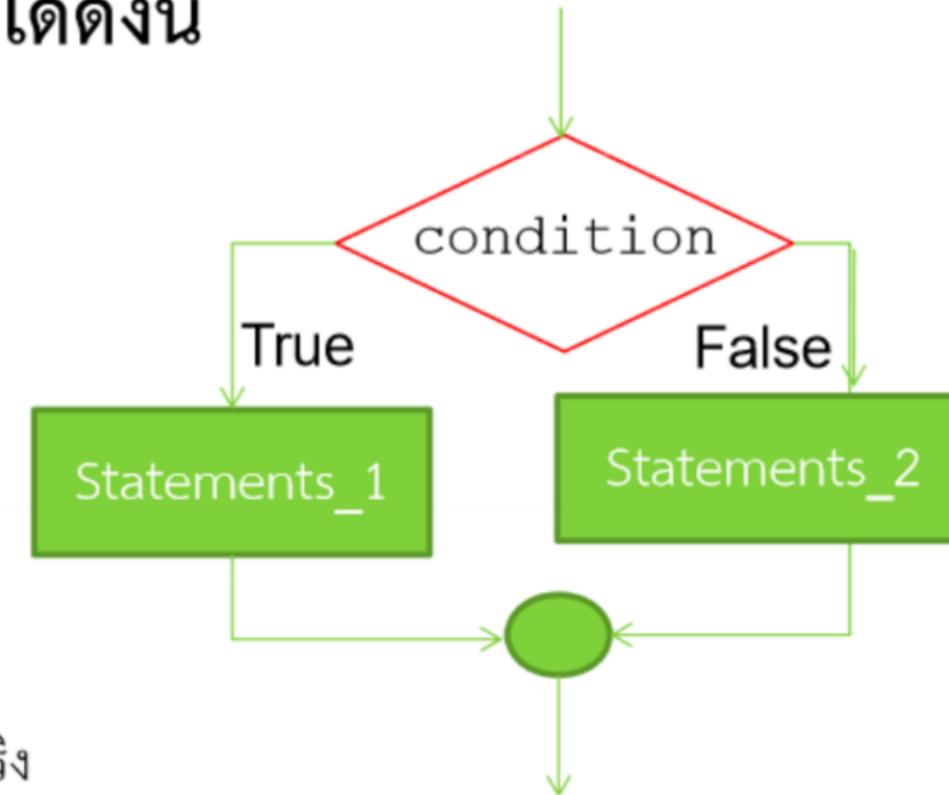
โดยที่

condition คือเงื่อนไขที่ให้ผลลัพธ์เป็นเลขจำนวนจริง ค่าผลลัพธ์ที่เป็นศูนย์ หรือ NULL หมายถึงเท็จ

ค่าผลลัพธ์ที่ไม่ใช่ ศูนย์ หรือไม่ใช่ NULL หมายถึงจริง

ส่วน statements_1 คือชุดคำสั่งที่จะทำการณ์ เมื่อเงื่อนไขเป็นจริง

ส่วน statements_2 คือชุดคำสั่งที่จะทำการณ์เมื่อเงื่อนไขเป็นเท็จ



การทดสอบเงื่อนไข

สัญลักษณ์	ความหมาย	สัญลักษณ์	ความหมาย
= =	เท่ากับ	!=	ไม่เท่ากับ
<	น้อยกว่า	<=	น้อยกว่าหรือเท่ากับ
>	มากกว่า	>=	มากกว่าหรือเท่ากับ

การทดสอบเงื่อนไขที่มี && (and), || (or) และ !(not)

A	B	A&&B	A B	!A
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

การทดสอบเงื่อนไขที่มี && (and), || (or) และ !(not)

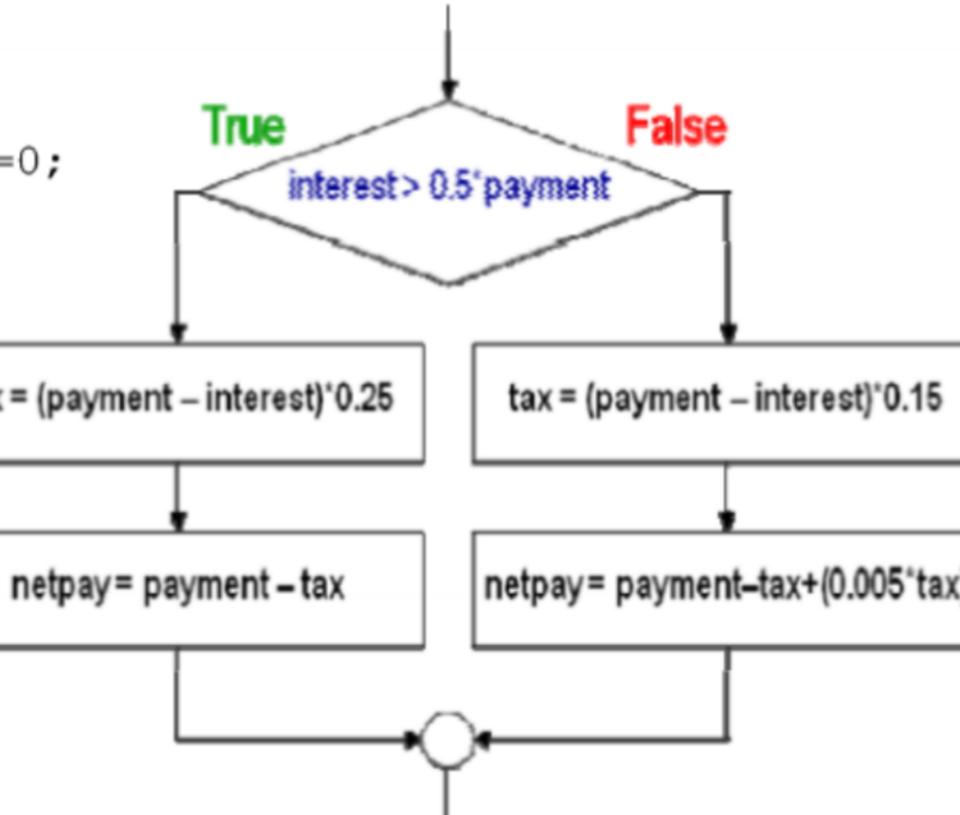
```
#include <stdio.h>
int main (void)
{
    int month;
    scanf("%d",&month);
    if (month == 12 || month == 1 || month == 2)
        printf("Winter");
    else if (month == 3 || month == 4 || month == 5)
        printf("Spring");
    else if (month == 6 || month == 7 || month == 8)
        printf("Summer");
    else if (month == 9 || month == 10 || month == 11)
        printf("Autumn");
    else
        printf(" No season");

    return 0;
}
```

Block

ส่วน statements ที่มีคำสั่งที่ต้องการทำมากกว่าหนึ่งคำสั่ง ต้องใช้เครื่องหมาย { ... }
เพื่อรวมให้เป็นชุดของคำสั่ง ซึ่งเรียกว่า block

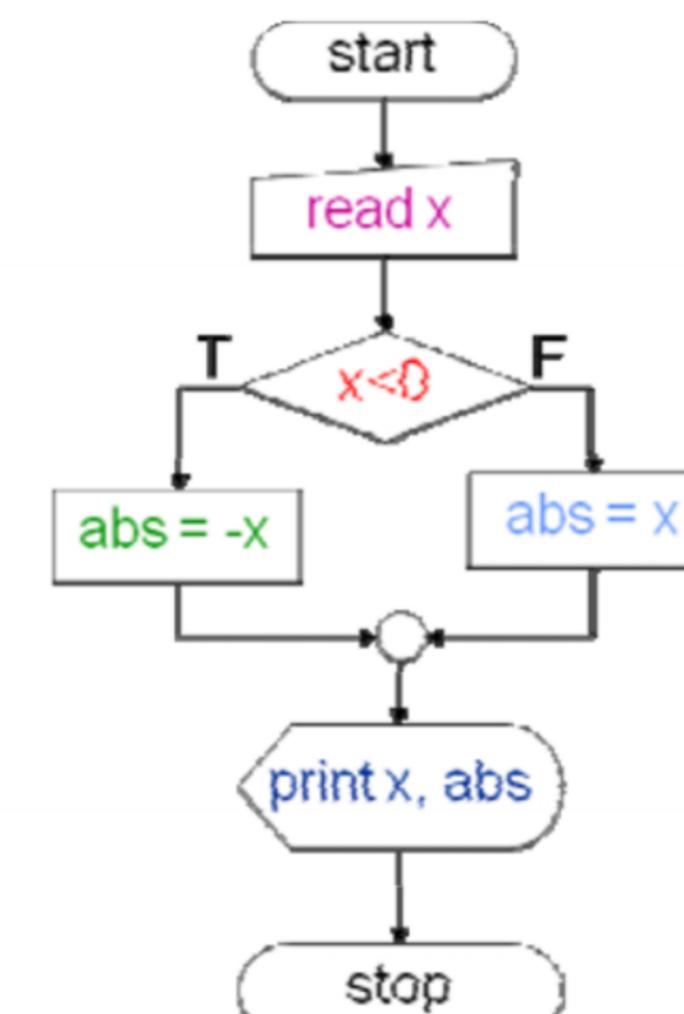
```
#include <stdio.h>
int main()
{
    float payment=0, interest=0, netpay=0, tax=0;
    printf("Input interest, payment:");
    scanf("%f %f", &payment, &interest);
    if (interest>0.5*payment){
        1 tax= (payment -interest)*0.25;
        2 netpay= payment - tax;
    }
    else{
        tax= (payment -interest)*0.15;
        netpay= payment - tax+(0.005*tax);
    }
    printf("Netpay=%2f", netpay);
}
```



Block

```
#include <stdio.h>
int main()
{
    int abs, x;
    printf ("Input an integer : ");
    scanf ("%d", &x);
    if (x<0)
        abs = -x;
    else
        abs = x;
    printf ("abs(%d) = %d\n", x, abs);
    return 0;
} // %

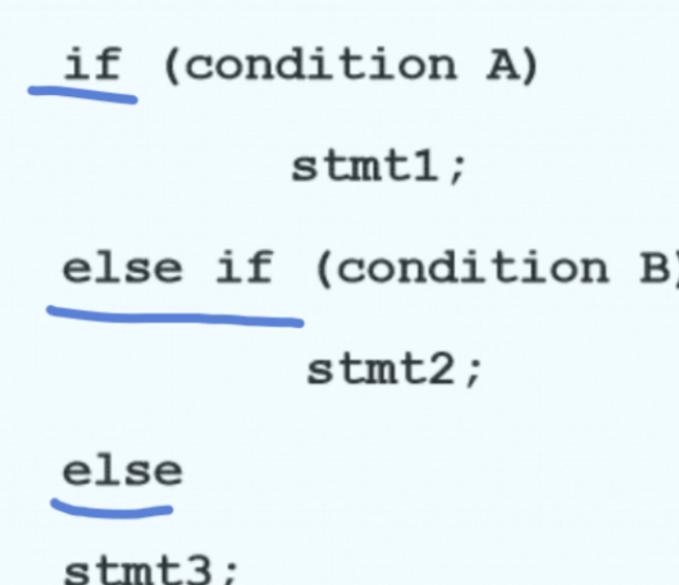
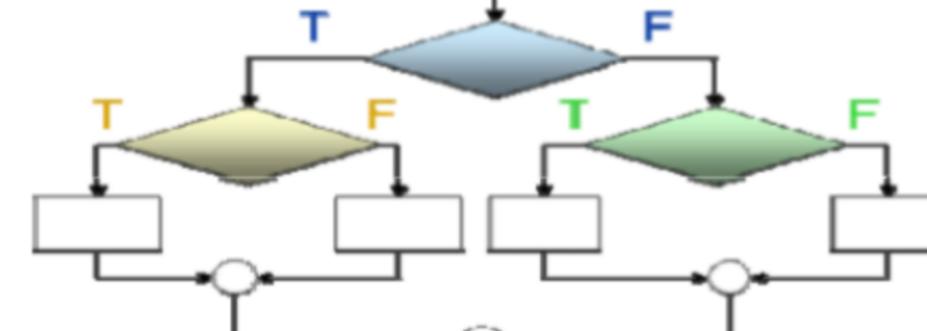
```



Nested if statements

Nested if คือการที่มีคำสั่ง if ซ้อนอยู่ใน block ของ if หรือ block ของ else

if-else-if Ladder หลังจากผ่านการเปรียบเทียบมาแล้วขั้นหนึ่ง อาจมีการเปรียบเทียบอีกต่อไป จึงต้องมีการใช้คำสั่ง if-else หลายครั้งซ้อนกันเรียกว่า if-else-if ladder



Computer Programming II

13

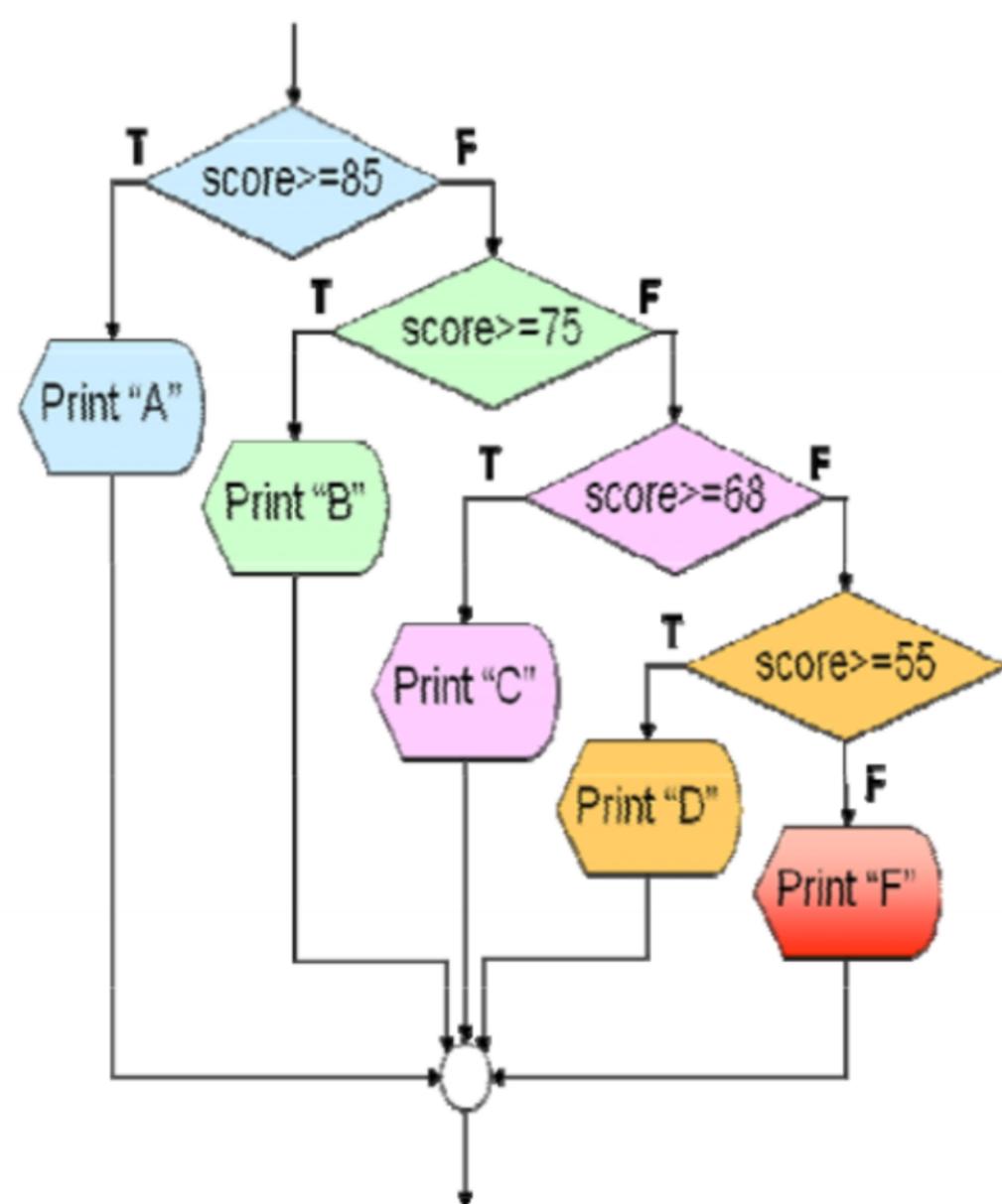
Nested if statements

```
if(test_condition1)
{
    if(test_condition2)
    {
        -----
        // Statements1 Or True Block;
    }
    else
    {
        -----
        // Statements2 Or False Block;
    }
}
else {
    -----
    // Statement 3 or false block
}
```

Computer Programming II

14

if-else-if Ladder



```
#include <stdio.h>
int main()
{
    int score;
    scanf("%d", &score);
    if (score >= 85)
        printf("A");
    else if (score >= 75)
        printf("B");
    else if (score >= 68)
        printf("C");
    else if (score >= 55)
        printf("D");
    else
        printf("F");
}
```

Computer Programming II

15

if-else-if Ladder

- **Program:** Write a program to enter the temperature and print the following message according to the given temperature by using if else ladder statement.

1. $T \leq 0$ "It's very very cold".
2. $0 < T < 0$ "It's cold".
3. $10 < T \leq 20$ "It's cool out".
4. $20 < T \leq 30$ "It's warm".
5. $T > 30$ "It's hot".

Computer Programming II

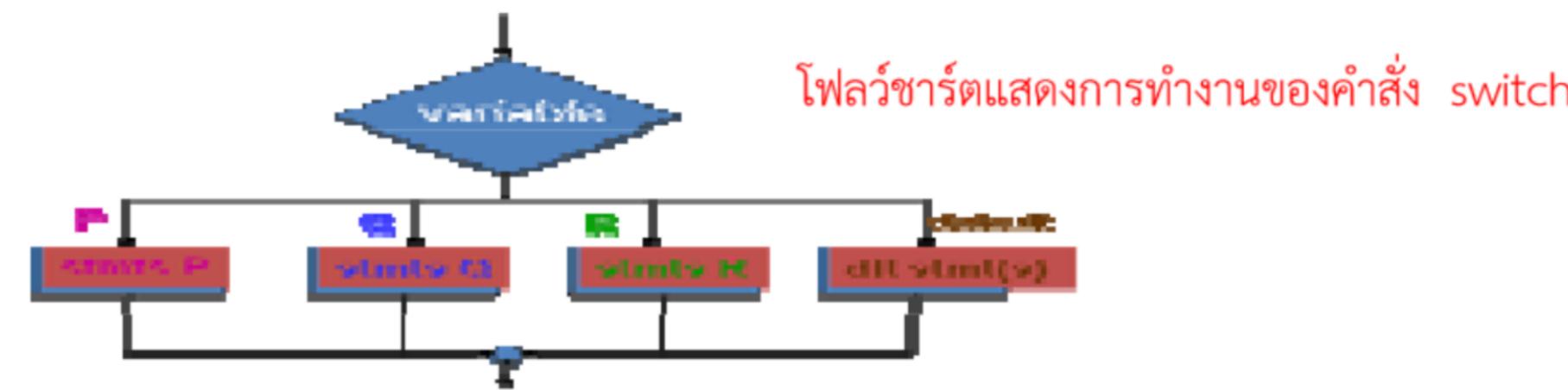
16

if-else-if Ladder

```
#include<stdio.h>
#include<conio.h>
void main()
{
    float T;
    printf("Enter The Temperature");
    scanf("%f",&T);
    if(T<=0)
    {
        printf("It's very very cold");
    }
    else if(T>0 && T<=10)
    {
        printf("It's cold");
    }
    else if(T>10 && T<=20)
    {
        printf("It's cool out");
    }
    else if(T<=30 && T>20)
    {
        printf("It's warm");
    }
    else
    {
        printf("It's hot");
    }
    getch();
}
```

การใช้คำสั่ง switch

คำสั่ง switch เป็นคำสั่งสำหรับการเลือกใช้คำสั่งหรือกลุ่มของคำสั่งจากหลาย ๆ กลุ่ม



variable คือ ตัวแปรที่จะใช้ในการทดสอบ
เงื่อนไข

P Q R คือ case หรือตัวอย่างของค่าที่จะถูก
นำมาทดสอบกับค่าที่อยู่ภายใต้ตัวแปร(variable)

กรณีที่ค่าในตัวแปรมีค่าตรงกับ case ได้ก็จะทำงานตามคำสั่งที่อยู่
ภายใต้ case นั้นๆ

หากไม่ตรงกับ case ใด ๆ เลย ก็จะเข้าสู่การทำงานในส่วนของ
default

ถ้ามี default เมื่อตรวจสอบค่าของตัวแปรหรือนิพจน์แล้วไม่ตรง
กับ case ใด ๆ ก็จะเข้ามาทำในส่วนของ default

ถ้าไม่มี default เมื่อตรวจสอบค่าของตัวแปรหรือนิพจน์แล้วไม่
ตรงกับ case ใด ๆ ก็จะไม่ทำงานตามคำสั่งภายใน switch-case

การใช้คำสั่ง switch

รูปแบบ

```
switch (expression) {
    case value_1: statements_1; break;
    ...
    case value_n: statements_n; break;
    default: statements_default;
}
```

โดยที่

expression ต้องให้ผลเป็น int หรือ char เท่านั้น

ค่าของ value_i ต้องเป็นค่าคงที่ชนิดเดียวกับ expression

value_i เป็นตัวแปรไม่ได้

การใช้คำสั่ง switch

```
#include <stdio.h>

int main (void)
{
// Local Declarations
int printFlag = 2;

// Program fragment 2 to demonstrate switch
switch (printFlag)
{
    case 1: printf("This is case 1\n");
    case 2: printf("This is case 2\n");
    default: printf("This is default\n");
} // switch
return 0;
} // main
```

การใช้คำสั่ง switch

```
#include <stdio.h>
int main()
{
    switch (grade)
    {
        case 'A': printf("Excellent"); break;
        case 'B': printf("Good"); break;
        case 'C': printf("So So"); break;
        case 'D': printf("Fails"); break;
        case 'F': printf("Get lost"); break;
        default : printf("Invalid");
    }
    return 0;
}
```

คำสั่ง break

คำสั่ง break ถูกใช้เพื่อให้โปรแกรมกระโดดข้ามการทำงานในคำสั่ง switch และไปทำคำสั่งแรกที่ตามหลังคำสั่ง switch

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    switch (grade)
    {
        case 'A': printf("Excellent\n");
        case 'B': printf("Good\n");
        case 'C': printf("So So\n");
        case 'D': printf("Fails\n");
        case 'F': printf("Get lost\n");
        default : printf("Invalid");
    }
    return 0;
}
```

ถ้าไม่ใส่ คำสั่ง break จะเกิดอะไรขึ้น?

ถ้า grade มีค่าเท่ากับ 'D' ผลลัพธ์คือ
Fails
Get lost
Invalid

Computer Programming II
การเขียนโปรแกรมคอมพิวเตอร์ 2
LECTURE#5 คำสั่งควบคุมการวนซ้ำ (Loop Statement)

อ.สติตย์ ประสมพันธ์

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

KMUTNB

คำสั่งควบคุม (Control Statement)

- การวนซ้ำ หมายถึง การควบคุมให้การกระทำการบางคำสั่งซ้ำหลายรอบ ซึ่งจะช่วยให้การเขียนโปรแกรมทำได้ง่ายสะดวก ไม่ต้องเขียนคำสั่งเดิมหลายครั้ง ทำให้โปรแกรมมีความกระชับ สามารถตรวจสอบความผิดพลาดได้ง่าย
- คำสั่งการวนรอบมี 3 ชนิดหลัก ๆ คือ
 - คำสั่ง for
 - คำสั่ง do-while
 - คำสั่ง while

Computer Programming II

2

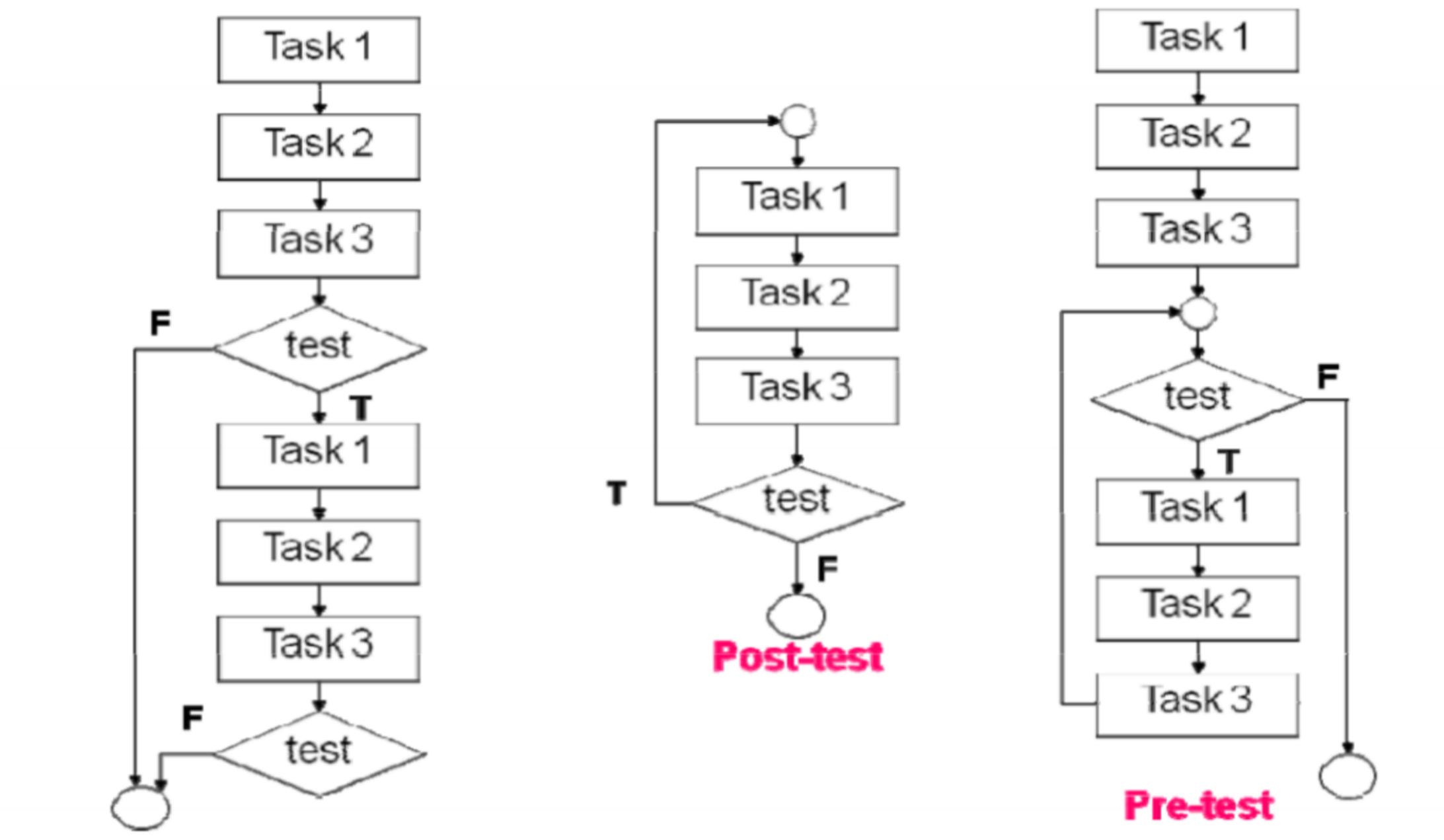
คำสั่งควบคุม (Control Statement)

- แต่ละคำสั่งมีรูปแบบและวิธีการใช้งานที่แตกต่างกัน สามารถเลือกใช้ตามความเหมาะสมของการใช้งานในโปรแกรม
- ส่วนประกอบของคำสั่งแบบวนซ้ำ ประกอบด้วย
 - ส่วนของการตรวจสอบ (loop test) เป็นเงื่อนไขเพื่อทดสอบว่าจะทำงานซ้ำ อีกหรือไม่
 - ส่วนของการทำงานซ้ำ (loop body) เป็นชุดคำสั่งที่จะถูกดำเนินการ

รายละเอียดของการสร้าง loop

- รายละเอียดของการสร้าง loop มีขั้นตอนการทำงานดังนี้
 - ระบุส่วนของการทำงานที่ต้องทำซ้ำ (loop body) ในส่วนนี้จะเป็นการระบุเงื่อนไข (loop test) ที่จะทำซ้ำ หรือเลิกทำซ้ำ
 - ระบุชนิดของ loop ที่จะใช้ ซึ่งชนิดของการทำซ้ำจะมีประเภทที่แตกต่างกัน หลัก ๆ 2 ประเภทคือ
 - Pre-test loop
 - Post-test loop

รายละเอียดของการสร้าง loop



Computer Programming II

5

รูปแบบการใช้ for แบบต่างๆ

```
#include<stdio.h>
int main()
{
    int i=0;
    float sum=0, score=0, sc=0;
    // ตัวอย่างการใช้คำสั่ง for แบบที่ 1
    for (i=1, sum=0; i<=10; i++)
        sum += i;
    // ตัวอย่างการใช้คำสั่ง for แบบที่ 2
    for (i=1; i<=25; i++)
    {
        scanf("%f", &score);
        printf("%f\n", score);
    }
    // ตัวอย่างการใช้คำสั่ง for แบบที่ 3
    for ③scanf("%f", &sc), sum=0; sc>0; sc=scanf("%f", &sc))
        sum += sc;
    return 0;
}
```

Handwritten annotations:

- For loop 1: $i < 25$
- For loop 2: $i \leq 25$
- For loop 3: $sc = 0; sc > 0;$

$sc \rightarrow 10$
 $sum = 0 + 10$

7

คำสั่ง for

คำสั่ง for เป็นคำสั่งที่สั่งให้ทำคำสั่ง หรือกลุ่มของคำสั่งวนซ้ำหลายรอบ โดยมีจำนวนรอบในการวนซ้ำที่แน่นอน



รูปแบบ

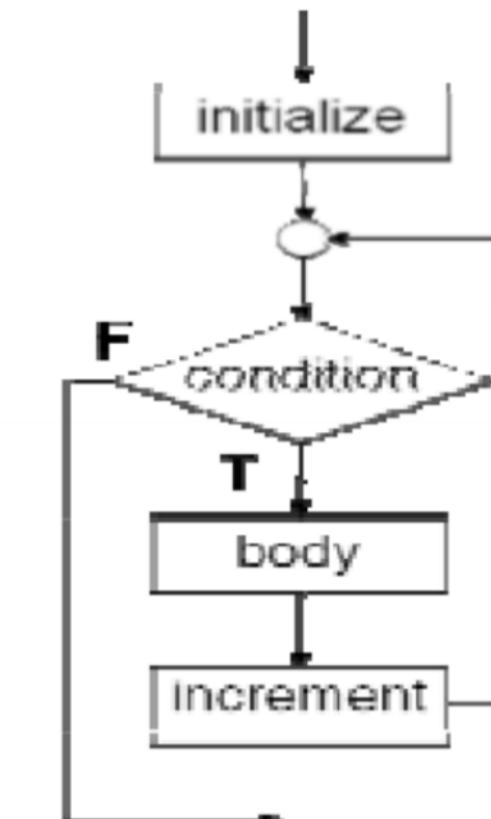
for (**ค่าเริ่มต้น**, **เงื่อนไข**; **ค่าเพิ่มหรือค่าลด**)

{

คำสั่ง

}

โดยที่ ค่าเริ่มต้น, เงื่อนไข, ค่าเพิ่มหรือค่าลด เป็นนิพจน์คำสั่ง หมายถึง คำสั่งที่จะถูกกระทำซ้ำ ซึ่งอาจจะมีเพียงคำสั่งเดียว หรือหลายคำสั่งก็ได้

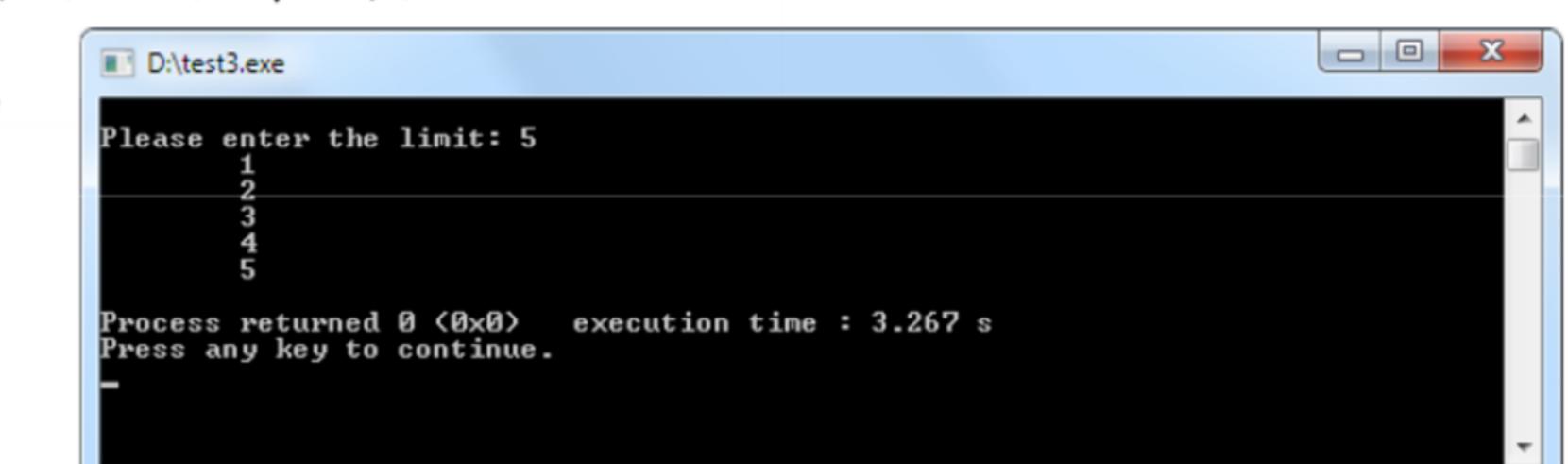


Computer Programming II

6

ตัวอย่างการพิมพ์ตัวเลข 1 ถึง จำนวนที่ผู้ใช้ระบุ

```
#include <stdio.h>
int main (void)
{
    // Local Declarations
    int limit;
    int i;
    // Statements
    printf ("\nPlease enter the limit: ");
    scanf ("%d", &limit);
    for (i = 1; i <= limit; i++) +i
    {
        printf ("\t%d\n", i);
    }
    return 0;
} // main
```



Computer Programming II

8

การวนซ้ำเพื่อพิมพ์ตัวเลขในแต่ละบรรทัด

```
#include <stdio.h>
int main (void)
{
// Statements
    for (int i = 1; i <= 3; i++)
    {
        printf("Row %d: ", i);
        for (int j = 1; j <= 5; j++)
            printf("%3d", j);
        printf("\n");
    } // for i
    return 0;
} // main
```

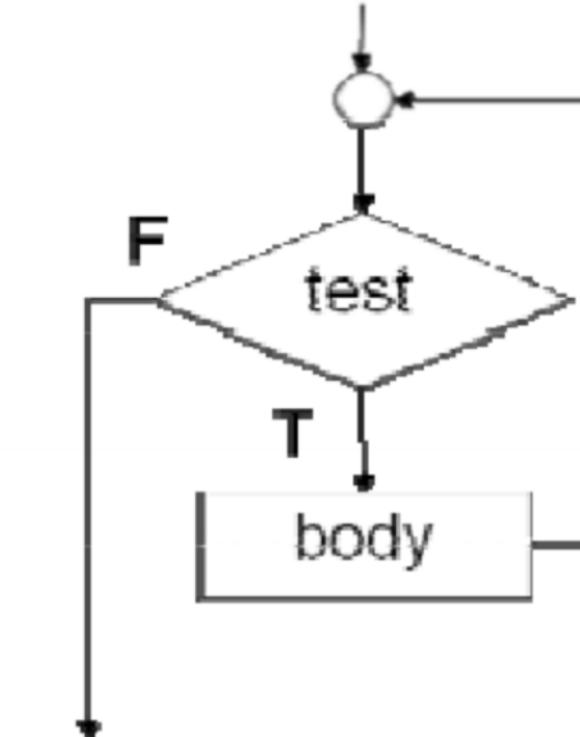
ผลลัพธ์จากการประมวลผลโปรแกรม

Row 1:	1	2	3	4	5
Row 2:	1	2	3	4	5
Row 3:	1	2	3	4	5

คำสั่ง while

คำสั่ง while เป็นคำสั่งวนซ้ำ ที่สั่งให้ทำคำสั่งที่อยู่ภายใต้คำสั่ง while หลายรอบจนกระทั่งเงื่อนไขเป็นเท็จ หรือ 0 จึงจะจบการวนซ้ำ

รูปแบบ
while (เงื่อนไข)
{
 คำสั่ง
}



โดยที่ เงื่อนไข เป็นนิพจน์ที่ให้ผลลัพธ์เป็นจริงหรือเท็จ คำสั่งอยู่ภายใต้คำสั่ง while อาจมีเพียงคำสั่งเดียว หรือหลายคำสั่ง

ตัวอย่างของการใช้คำสั่ง while ในรูปแบบต่าง ๆ

```
#include <stdio.h>
int main (void)
{
    int x;
    float sum=0;
    float vat=0;
    int prc=0;
    scanf("%d", &x);
    while (x<0) |
        x++;
    while (sum<20)
    {
        scanf("%d", &prc);
        if (prc<20)
        {
            vat = .07*prc;
            sum += prc + vat;
        }
    }
    return 0;
} // main
```

0 < 20
10
10
0 + 10 + vat

ตัวอย่างการพิมพ์ตัวเลขที่ผู้ใช้ระบุ โดยพิมพ์จากตัวนั้น ๆ ถึง 1 บรรทัดละ 10 ตัวเลข

```
// Test number
if (num > 100) |
    num = 100;
lineCount = 0;
while (num > 0)
{
    if (lineCount < 10)
        lineCount++;
    else
    {
        printf("\n");
        lineCount = 1;
    } // else
    printf("%4d", num--);
} // while
return 0;
} // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Enter an integer between 1 and 100: 15
15 14 13 12 11 10 9 8 7 6
5 4 3 2 1

ตัวอย่างการรับค่าตัวเลขและแสดงผลตัวเลข หาจำนวน digits และผลรวมของทุก ๆ digits

```
#include <stdio.h>
int main (void)
{
    int number;
    int count = 0;
    int sum = 0;
    printf("Enter an integer: ");
    scanf ("%d", &number);
    printf("Your number is: %d\n\n", number);
    while (number != 0)
    {
        count++;
        sum += number % 10; //-->5
        number /= 10; //-->1234
    } // while
    printf("The number of digits is: %3d\n", count);
    printf("The sum of the digits is: %3d\n", sum);
    return 0;
} // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม
Enter an integer: 12345
Your number is: 12345
The number of digits is: 5
The sum of the digits is: 15

Computer Programming II

13

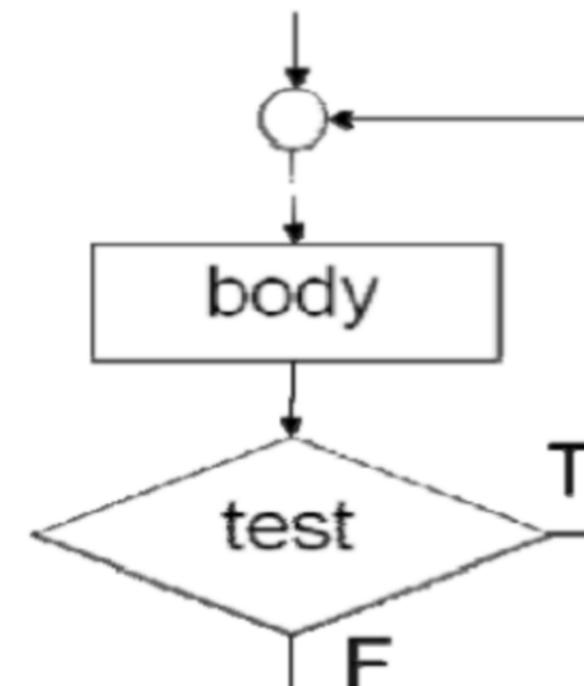
คำสั่ง do-while

คำสั่ง do-while เป็นคำสั่งงานซ้ำ ที่สั่งให้ทำคำสั่งที่อยู่ภายใต้คำสั่ง do-while หนึ่งรอบ แล้วจึงจะตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นเท็จจะจบการทำงานทันที

รูปแบบ

```
do
{
    คำสั่ง;
} while (เงื่อนไข);
```

โดยที่ เงื่อนไข เป็นนิพจน์ที่ให้ผลลัพธ์เป็นจริงหรือเท็จ และคำสั่ง อยู่ภายใต้คำสั่ง do-while อาจมีเพียงคำสั่งเดียว หรือหลายคำสั่ง



ตัวอย่างการรับค่าตัวเลขแล้วพิมพ์ตัวเลขนั้น ๆ จากหลังมาหน้า

```
#include <stdio.h>
int main (void)
{
// Local Declarations
long num;
int digit;
// Statements
printf("Enter a number and I'll print it backward: ");
scanf ("%d", &num);

while (num > 0)
{
    digit = num % 10;
    printf("%d", digit);
    num = num / 10;
} // while
printf("\nHave a good day.\n");
return 0;
} // main
```

Computer Programming II

14

ตัวอย่างการใช้คำสั่ง while และ do...while เพื่อพิมพ์ตัวเลขจาก 5 ถึง 1

```
#include <stdio.h>
int main (void)
{
// Local Declarations
int loopCount;
// Statements
loopCount = 5;
printf("while loop : ");
while (loopCount > 0)
    printf ("%3d", loopCount--);
printf("\n\n");
loopCount = 5;
printf("do...while loop: ");
do
    printf ("%3d", loopCount--);
while (loopCount > 0);
printf("\n");
return 0;
} // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม
while loop : 5 4 3 2 1
do...while loop: 5 4 3 2 1

Computer Programming II

15

Computer Programming II

16

การวนซ้ำซ้อน

การวนซ้ำซ้อน หมายถึง การควบคุมให้กระทำบางคำสั่งหลายรอบ และในการทำงานแต่ละรอบก็จะควบคุมให้ทำการคำสั่งที่อยู่ภายในนั้นอีกหลายรอบ การวนซ้ำซ้อน อาจจะเขียนได้โดยการใช้คำสั่งวนซ้ำได ๆ ซ้อนกัน 2 คำสั่ง เช่น

คำสั่ง for ซ้อนกับคำสั่ง for

คำสั่ง while ซ้อนกับคำสั่ง while

คำสั่ง while-do ซ้อนกับคำสั่ง while-do

การวนซ้ำซ้อน

```
#include <stdio.h>
int main (void)
{
// Local Declarations
    int limit;
// Read limit
    printf("\nPlease enter a number between 1 and 9: ");
    scanf("%d", &limit);
    for (int lineCtrl = 1; lineCtrl <= limit; lineCtrl++)
    {
        for (int numCtrl = 1;
            numCtrl <= lineCtrl;
            numCtrl++)
        {
            printf("%1d", numCtrl);
        }
        printf("\n");
    } // for lineCtrl
    return 0;
} // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Please enter a number between 1 and 9: 6

.1
.12
.123
.1234
.12345
.123456

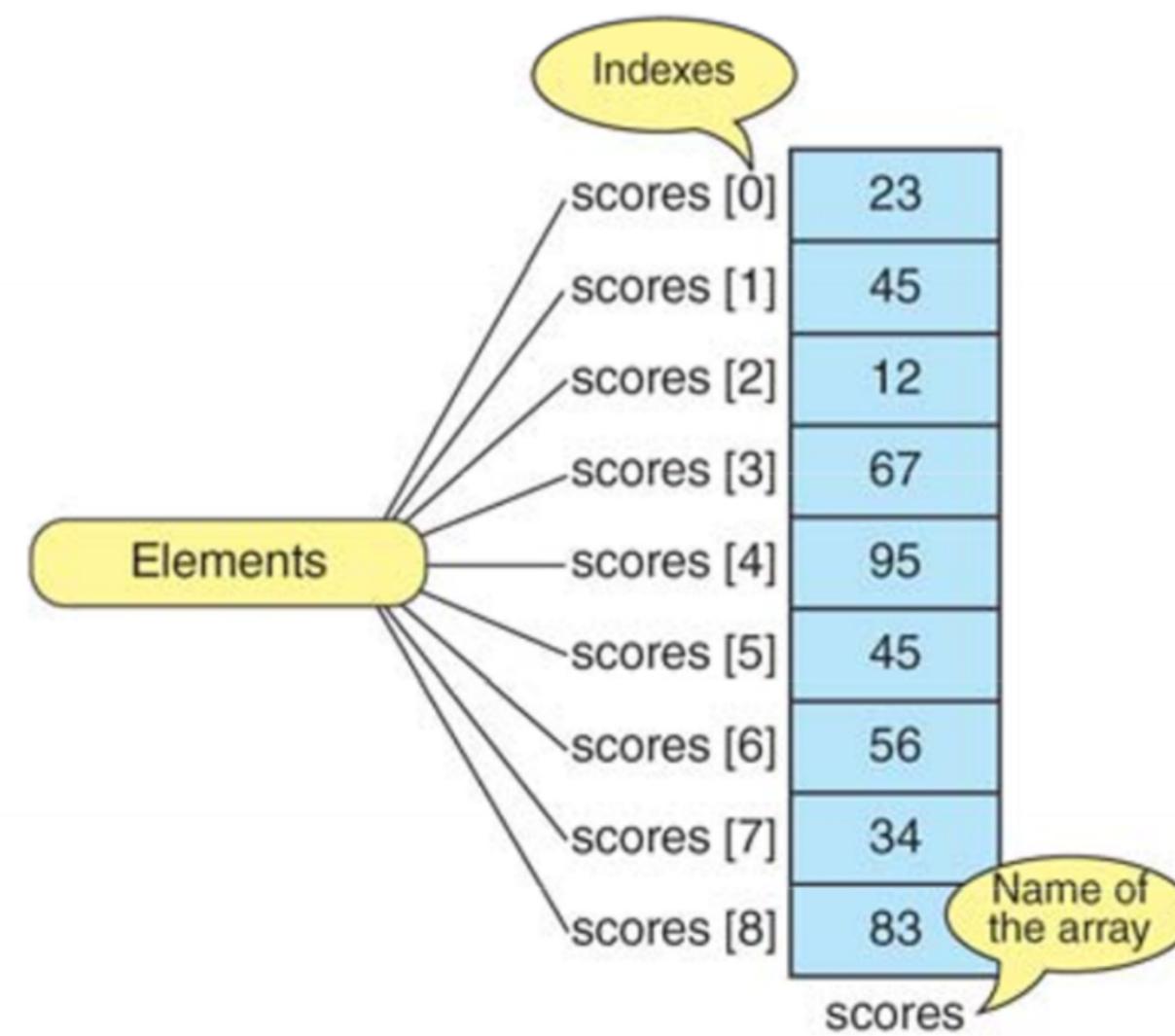
Computer Programming II การเขียนโปรแกรมคอมพิวเตอร์ 2 LECTURE#6 โครงสร้างข้อมูลแบบอาร์เรย์ (Arrays)

อ.สติตย์ ประสมพันธ์
ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ
KMUTNB

อาร์เรย์

- ตัวแปรชุด (array) array เป็นชนิดข้อมูลประเภทหนึ่งที่นำเอาชนิดข้อมูลข้อมูลพื้นฐานมาประยุกต์เป็นชนิดข้อมูลประเภทนี้ เช่น
 - ตัวอักษร(char)
 - ชนิดข้อมูลแบบเลขจำนวนเต็ม(int)
 - ชนิดข้อมูลแบบเลขจำนวนจริง(float)
- เมื่อประกาศโครงสร้างข้อมูลแบบอาร์เรย์(Array) จะเก็บข้อมูลต่างจากชนิดข้อมูลพื้นฐานทั่วไป คือ สามารถเก็บค่าภายในตัวแปรชนิดนี้ได้มากกว่า 1 ค่าซึ่งจำนวนค่าที่เก็บนั้นขึ้นอยู่กับขนาดของอาร์เรย์ที่ได้กำหนดไว้

ตัวอย่างของอาร์เรย์ scores



ประเภทของตัวแปรชุด

- อาจแบ่งตามลักษณะของจำนวนตัวเลขของดัชนี คือ
 - 1. ตัวแปรชุด 1 มิติ (one dimension arrays หรือ single dimension arrays) เป็นตัวแปรชุดที่มีตัวเลขแสดงขนาดเป็นเลขตัวเดียว เช่น word[20] , num[25] , x[15]
 - 2. ตัวแปรชุดหลายมิติ (multi-dimension arrays) เป็นตัวแปรชุดที่ซึ่งมีตัวเลขแสดงขนาดเป็นตัวเลขหลายตัว ที่นิยมใช้กันมี 2 มิติ กับ 3 มิติ
 - 2.1 ตัวแปรชุด 2 มิติ มีเลขแสดงขนาด 2 ตัว เช่น a[3][5] , name[5][6]
 - 2.2 ตัวแปรชุด 3 มิติ มีเลขแสดงขนาด 3 ตัว เช่น a[3][5][6] , name[5][6][8]

การประกาศและกำหนดค่าตัวแปรชุด 1 มิติ

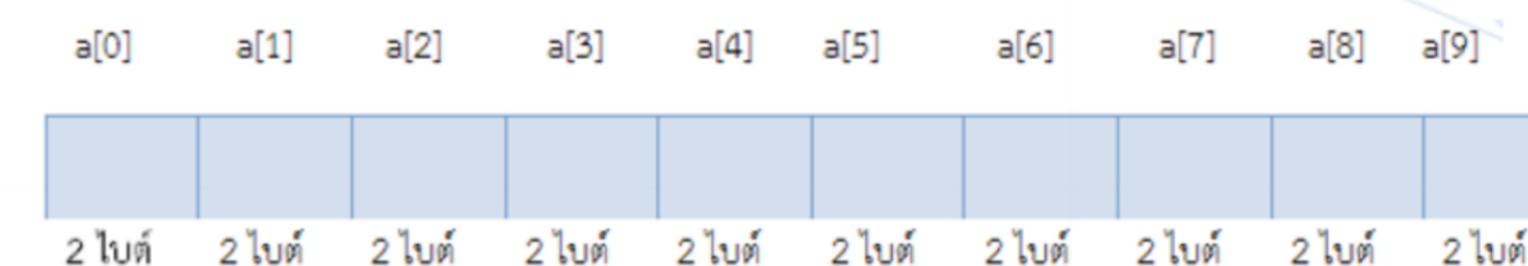
- การประกาศตัวแปรชุด 1 มิติ เพื่อใช้งาน ใช้คำสั่ง ดังนี้
type arrayname[size];
- โดย
 - type คือ ชนิดของตัวแปร เช่น int char float
 - arrayname คือชื่อของตัวแปร array
 - size คือ ขนาดของตัวแปร

Computer Programming II

5

การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท integer

- int a[10]; เป็นการประกาศตัวแปร array ชื่อ a เป็น array ของข้อมูลประเภท integer มีสมาชิกได้จำนวน 10 ตัว คือ a[0] a[1] a[2] a[3] ... a[9] โดยมีการจองเนื้อที่ในหน่วยความจำเปรียบเทียบได้ดังรูป



- โดยสมาชิกแต่ละตัวจะใช้เนื้อที่เท่ากับตัวแปรประเภท integer ที่ไม่ได้อยู่ใน array คือ 2 ไบต์ ต่อ ตัวแปร 1 ตัวดังนั้นเนื้อที่หน่วยความจำที่ใช้ทั้งหมดจึงเท่ากับจำนวนสมาชิก คูณ ด้วย 2 ไบต์

Computer Programming II

6

การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท integer

- การกำหนดค่าให้แก่ตัวแปร array อาจกำหนดพร้อมกับการประกาศ เช่น
int num1[3] = {56,25,89};
- เป็นการประกาศว่าตัวแปร num1 เป็น array ประเภท integer มีสมาชิก 3 ตัว โดย
 - num1[0] = 56;
 - num1[1]=25;
 - num1[2]=89;
- ประกาศว่า a เป็นตัวแปร array ประเภท integer ที่มีสมาชิก 2 ตัว โดย
 - a[0] มีค่า เป็น 200
 - a[1] มีค่าเป็น 230

Computer Programming II

7

การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท integer

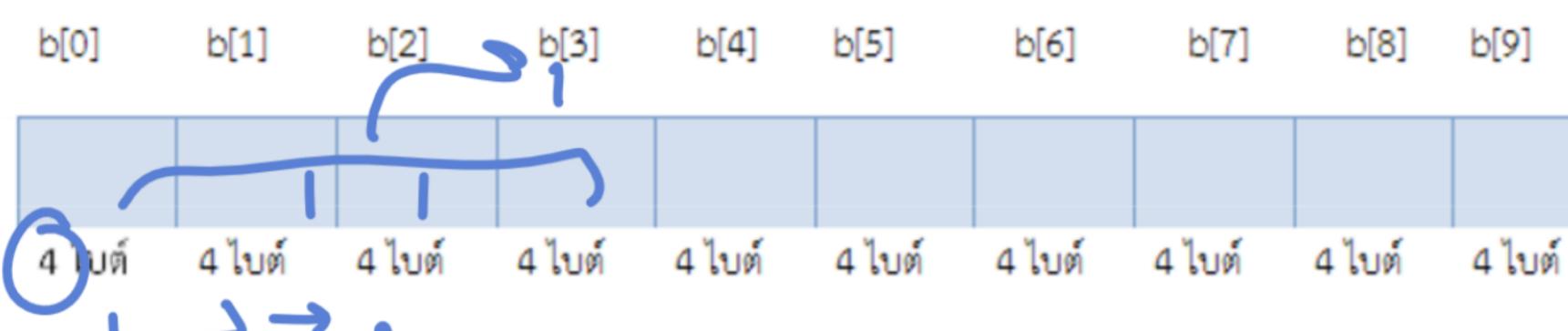
- แต่ไม่สามารถประกาศว่า int value[] ;
- โดยถ้าจะไม่ระบุจำนวนสมาชิก ต้องระบุค่าของแต่ละสมาชิกที่ถูกล้อมรอบด้วย {}
โดยระหว่างสมาชิกคั่นด้วยเครื่องหมาย , (คอมมา) ดังตัวอย่าง
int a[]={200,230};
- หรือ ประกาศตัวแปร โดยยังไม่กำหนดค่า เช่น
int money[5];
แล้วไปกำหนดค่าให้สมาชิกแต่ละตัวในภายหลัง เช่น
money[0] = 250; money[4] = 500;

Computer Programming II

8

การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท float

- float b[10]; เป็นการประกาศตัวแปร array ของ ตัวแปรจำนวนที่มีทศนิยมได้ คือ float ในชื่อ b ซึ่งมีสมาชิกได้ 5 ตัว คือ b[0] b[1]... b[9] มีการจองเนื้อที่ในหน่วยความจำเปรียบเทียบได้ ดังรูป



- โดยสมาชิกแต่ละตัวใช้หน่วยความจำ 4 ไบต์ ดังนั้นทั้งหมดจะใช้หน่วยความจำ 4 คูณ 5 คือ 20 ไบต์

20

Computer Programming II

9

การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท float

- การกำหนดค่าของตัวแปร array ประเภท float เป็นไปในลักษณะเดียวกับ array ประเภท integer ประกาศพร้อมกับกำหนดค่าให้โดยโดยล้อมรอบด้วย {} และค่าของสมาชิกแต่ละตัวคั่นด้วย , เช่น

float num[5] = {2.00, 1.25, 5.36, 6.32, 246.10};
num[0] = 2.00

num[1] = 1.25

num[2] = 5.36

num[3] = 6.32

num[4] = 246.10

- หรือประกาศตัวแปรก่อนแล้วไปกำหนด ค่าภายหลัง เช่น float salary[10];

— salary[0] = 25000.00;

— salary[9] = 55600.00;

Computer Programming II

10

ตัวอย่าง

- เขียนโปรแกรมเพื่อรับข้อมูล N จำนวนเก็บในตัวแปรอาร์เรย์ x ขนาด N และคำนวณค่าเฉลี่ยของคะแนน N จำนวนที่รับค่าทางคีย์บอร์ดเป็นรูปของคะแนนเฉลี่ยที่คำนวณได้

```

1 #include <stdio.h>
2 int main(){
3     int i, n, x[100];
4     float mean=0, sum=0;
5     printf("Enter N:");
6     scanf("%d", &n);
7     for(i=0; i<n, i++){
8         printf("x[%d]=", i);
9         scanf("%d", &x[i]);
10        sum=sum+x[i];
11    }
12    mean=sum/n;
13    printf("Mean=%f\n", mean);
14    return 0;
15 }
16 }
```

Computer Programming II

11

ตัวอย่างการประกาศตัวแปรชุด 1 มิติ

```

• #include <stdio.h>
• #define MAX_SIZE 25
int main (void)
{
    int list [MAX_SIZE] =
    {
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
        21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
        41, 42, 43, 44, 45
    };
    int numPrinted;
    numPrinted = 0;
    for (int i = 0; i < MAX_SIZE; i++)
    {
        printf("%3d", list[i]);
        if (numPrinted < 9)
            numPrinted++;
        else
            printf("\n");
        numPrinted = 0;
    }
}
/* Results:
   1 2 3 4 5 6 7 8 9 10
   21 22 23 24 25 26 27 28 29 30
   41 42 43 44 45
*/
```

Computer Programming II

12

ตัวอย่างการใช้งานตัวแปรชุด 1 มิติ

```

• #include <stdio.h>
#define ARY_SIZE 5

int main (void)
{
    // Local Declarations
    int sqrArry[ARY_SIZE];
    // Statements
    for (int i = 0; i < ARY_SIZE; i++)
        sqrArry[i] = i * i;

    printf("Element\tSquare\n");
    printf("-----\t-----\n");
    for (int i = 0; i < ARY_SIZE; i++)
        printf("%5d\t%4d\n", i, sqrArry[i]);
    return 0;
} // main

```

/* Results:
Element Square
----- -----
0 0
1 1
2 4
3 9
4 16

*/

Computer Programming II

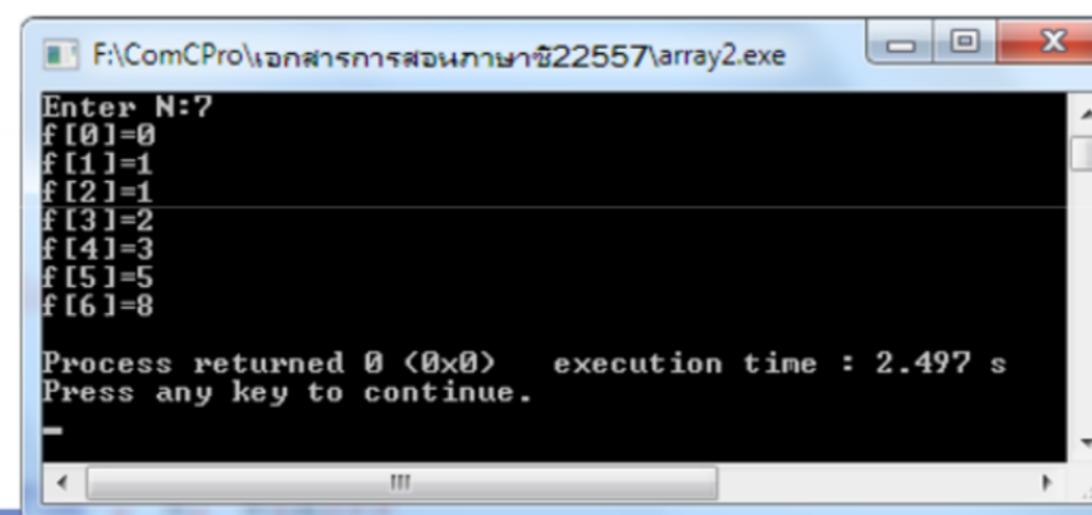
13

ตัวอย่างการใช้งานตัวแปรชุด 1 มิติ

```

1 #include <stdio.h>
2 int main(){
3     int i, n;
4     float f[40]={0,1};
5     printf("Enter N:");
6     scanf("%d", &n);
7     for(i=2;i<n;i++){
8         f[i]=f[i-1]+f[i-2];
9     }
10    for(i=0;i<n;i++){
11        printf("f[%d]=%.0f\n", i, f[i]);
12    }
13    return 0;
14 }

```



Computer Programming II

15

ตัวอย่างการใช้งานตัวแปรชุด 1 มิติ

```

#include <stdio.h>
int main (void)
{
    int readNum;
    int numbers[50];
    printf("You may enter up to 50 integers:\n");
    printf("How many would you like to enter? ");
    scanf ("%d", &readNum);
    if (readNum > 50)
        readNum = 50;
    printf("\nEnter your numbers: \n");
    for (int i = 0; i < readNum; i++)
        scanf ("%d", &numbers[i]);
    printf("\nYour numbers reversed are: \n");
    for (int i = readNum - 1, numPrinted = 0;
         i >= 0; i--)
    {
        printf("%3d", numbers[i]);
        if (numPrinted < 9)
            numPrinted++;
        else
        {
            printf("\n");
            numPrinted = 0;
        }
    }
    return 0;
}

```

/* Results:
You may enter up to 50 integers:
How many would you like to enter? 12
Enter your numbers:
1 2 3 4 5 6 7 8 9 10 11 12
Your numbers reversed are:
12 11 10 9 8 7 6 5 4 3
2 1
*/

Computer Programming II

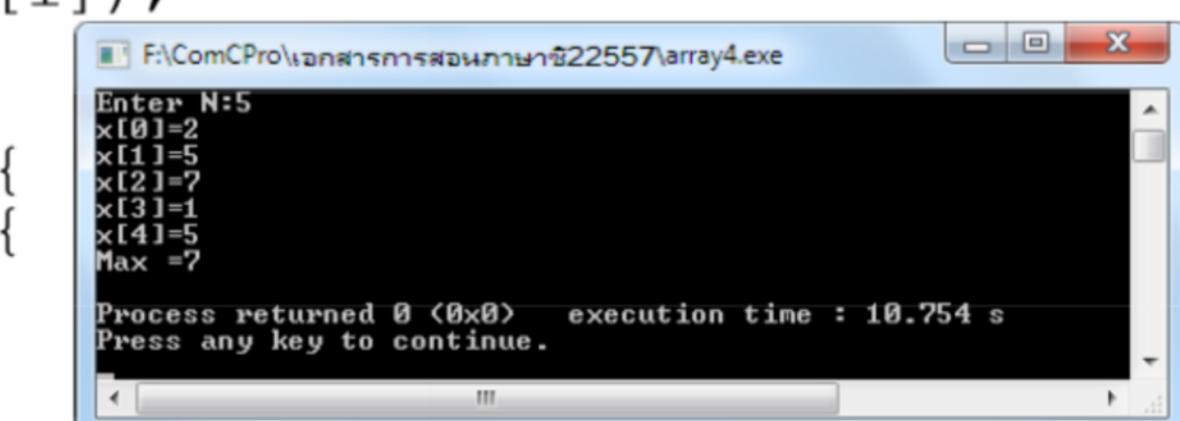
14

ตัวอย่างการใช้งานตัวแปรชุด 1 มิติ

```

• 1 #include <stdio.h>
2 int main(){
3     int i, max, n, x[100];
4     printf("Enter N:");
5     scanf("%d", &n);
6     for(i=0;i<n;i++){
7         printf("x[%d]=",i);
8         scanf("%d", &x[i]);
9     }
10    max=x[0];
11    for(i=1;i<n;i++){
12        if (max<x[i]){
13            max=x[i];
14        }
15    }
16    printf("Max=%d\n", max);
17    return 0;
18 }

```

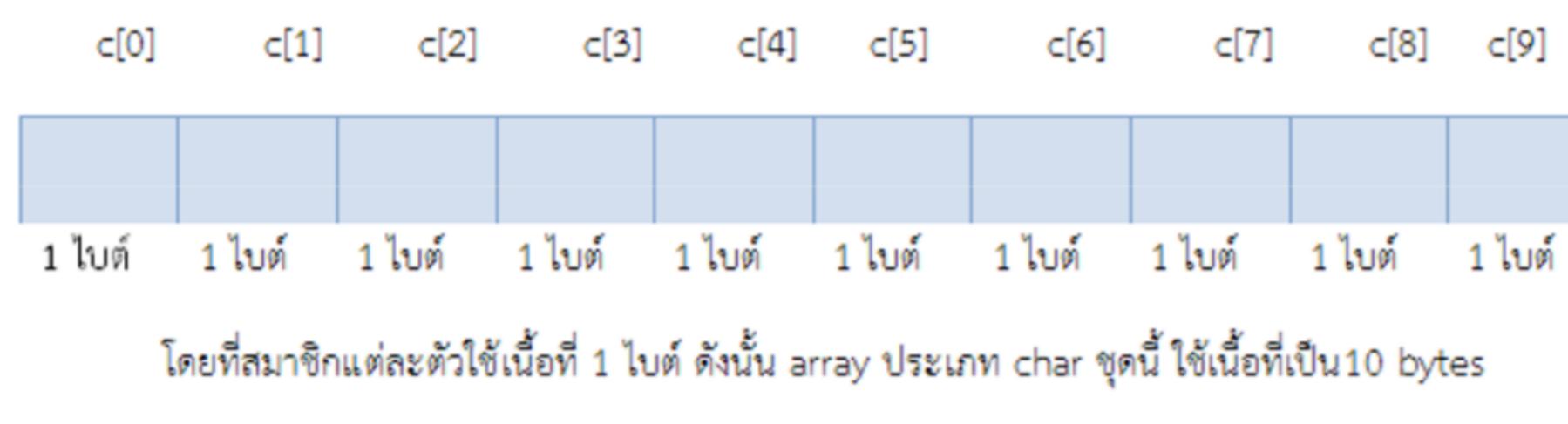


Computer Programming II

16

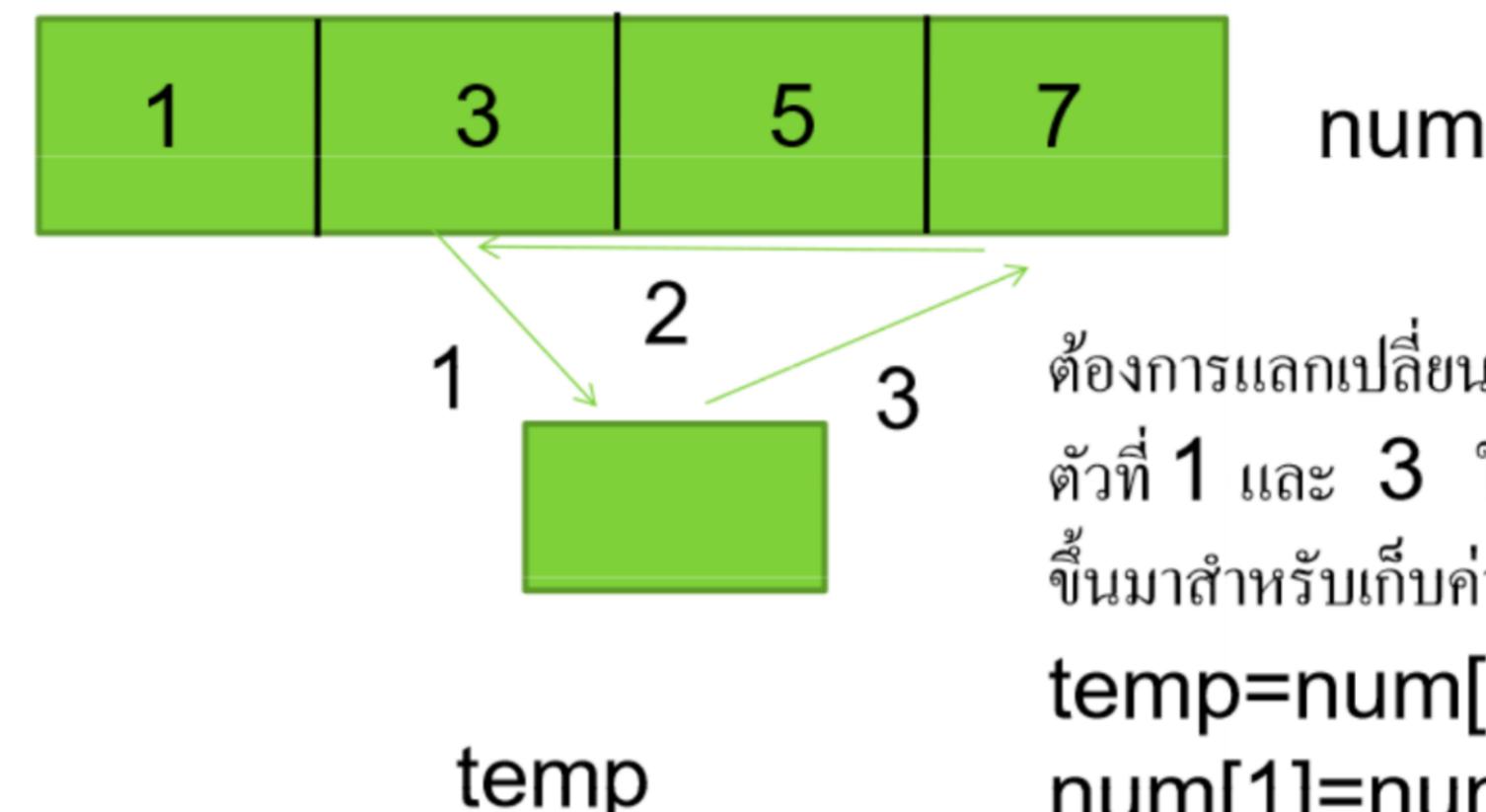
การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท char

- char c[10]; เป็นการประกาศตัวแปร array ชื่อ c เป็น array ของ ตัวแปรอักขระ char มีสมาชิกได้ 10 ตัว คือ a[0] a[1] ... a[9] โดยการใช้เนื้อที่ในหน่วยความจำเปรียบเทียบได้ ดังรูป

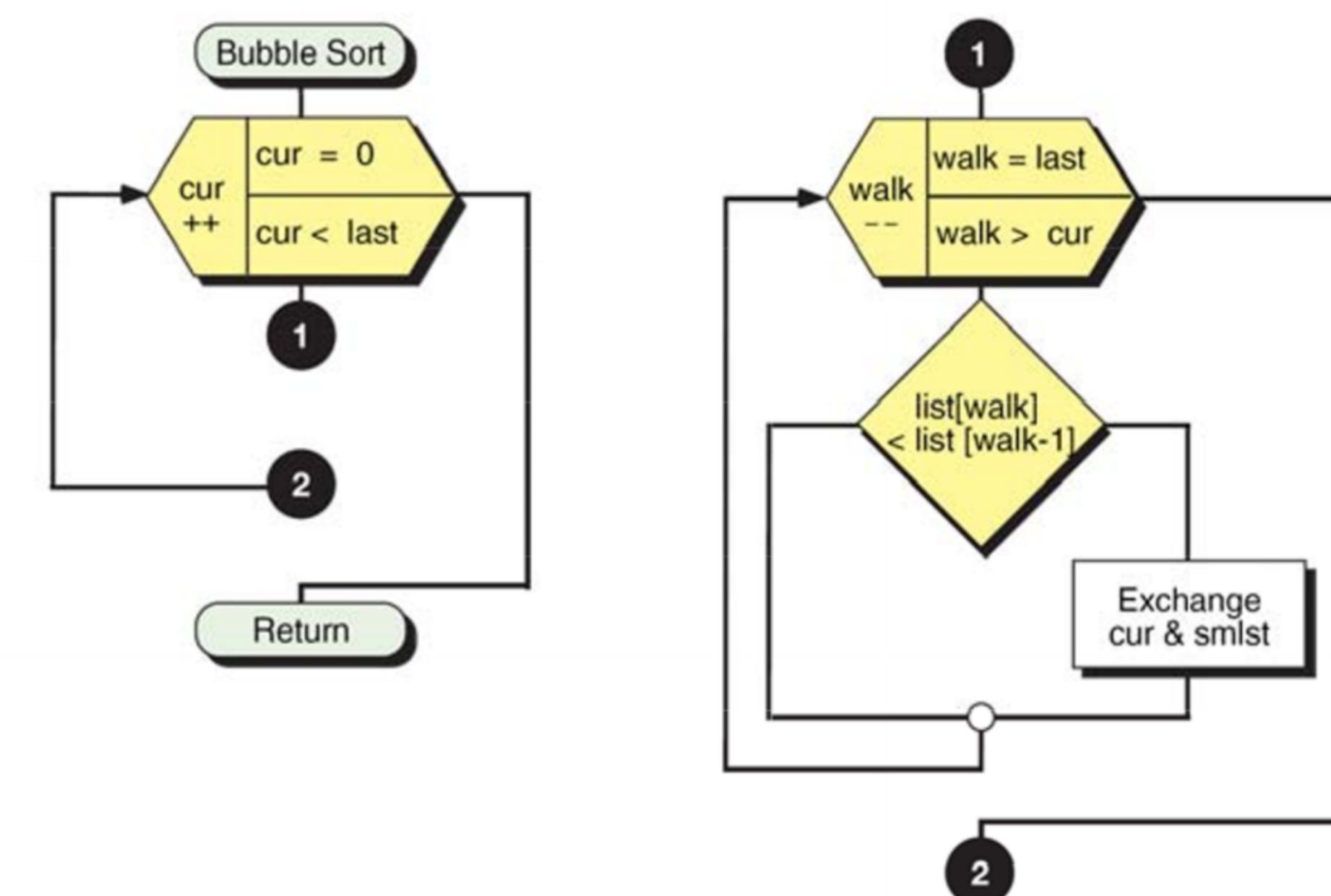
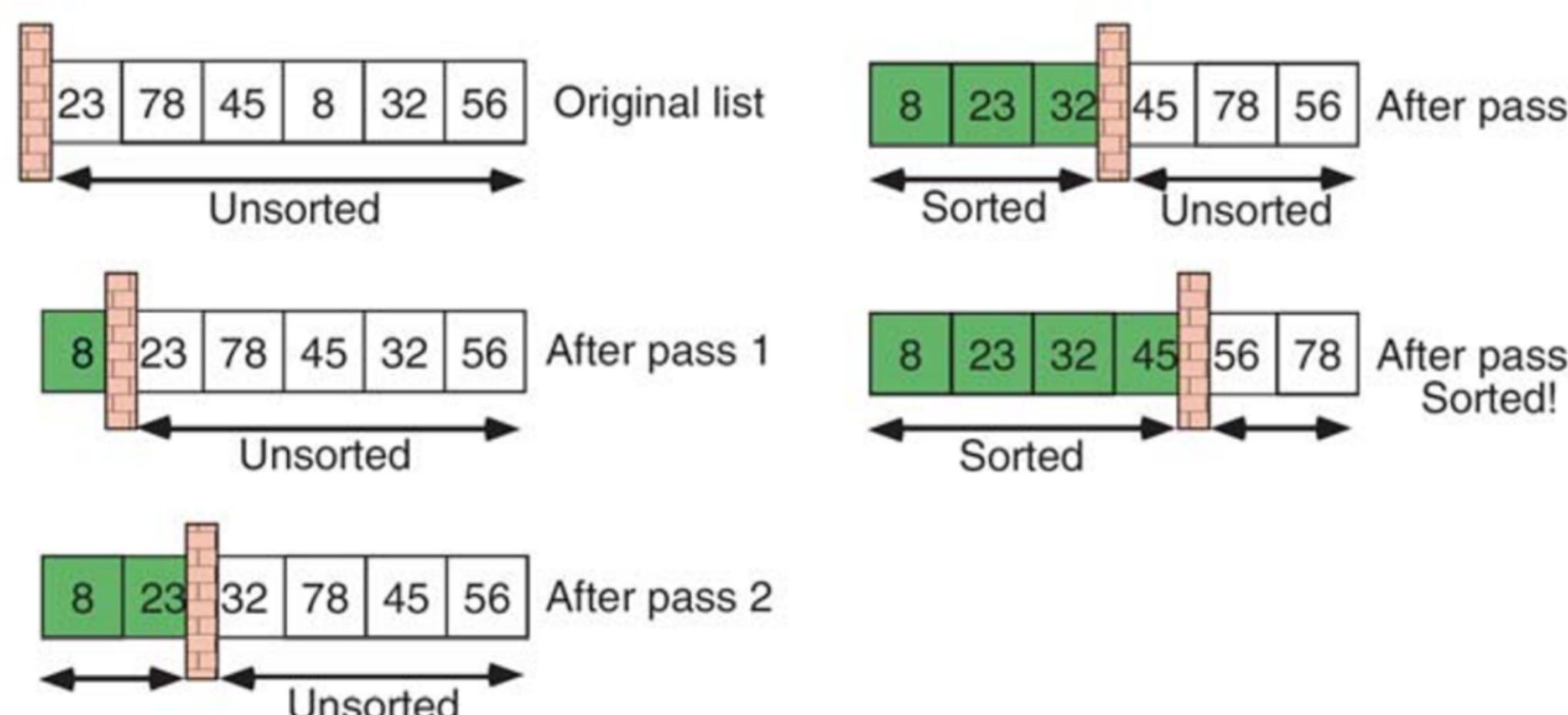


การแลกเปลี่ยนข้อมูลระหว่างกันของอาร์เรย์

- int num[4] = {1,3,5,7}



ตัวอย่างการประยุกต์ใช้งานอาร์เรย์กับการเรียงลำดับ



```

1  /* ===== bubbleSort ===== */
2  Sort list using bubble sort. Adjacent elements are
3  compared and exchanged until list is ordered.
4  Pre the list must contain at least one item
5      last contains index to last element in list
6  Post list rearranged in sequence low to high
7 */
8 void bubbleSort (int list [], int last)
9 {
10 // Local Declarations
11     int temp;
12
13 // Statements
14 // Outer loop
15     for(int current = 0; current < last; current++)
16     {
17         // Inner loop: Bubble up one element each pass

```

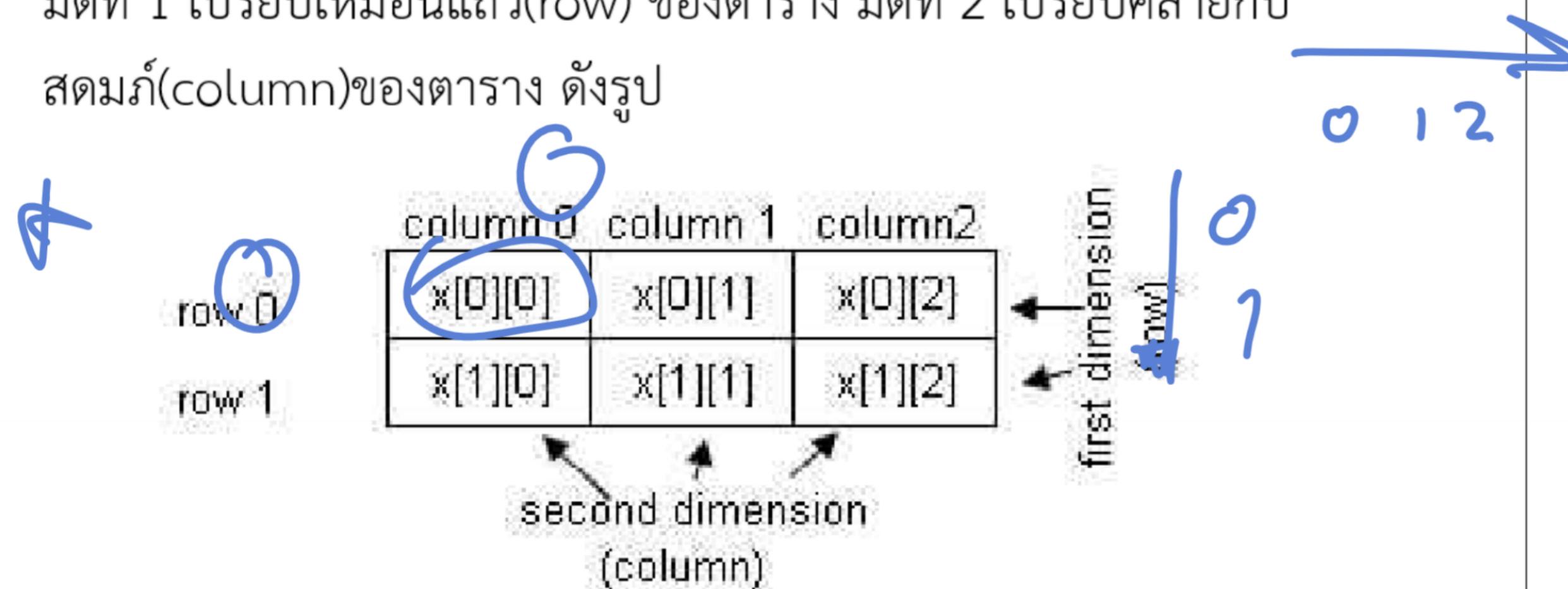
```

18     for (int walker = last;
19             walker > current;
20             walker--)
21     if (list[walker] < list[walker - 1])
22     {
23         temp          = list[walker];
24         list[walker] = list[walker - 1];
25         list[walker - 1] = temp;
26     } // if
27 } // for current
28 return;
29 } // bubbleSort

```

การประมวลและกำหนดค่าตัวแปรชุด 2 มิติ(two dimension arrays)

- array 2 มิติ มีการจัดการจัดเก็บเบรียบเทียบคล้ายกับ ตาราง 2 มิติ มิติที่ 1 เปรียบเหมือนแนว(row) ของตาราง มิติที่ 2 เปรียบคล้ายกับ สมณฑล(column)ของตาราง ดังรูป



การประมวลและกำหนดค่าตัวแปรชุด 2 มิติ(two dimension arrays)

```

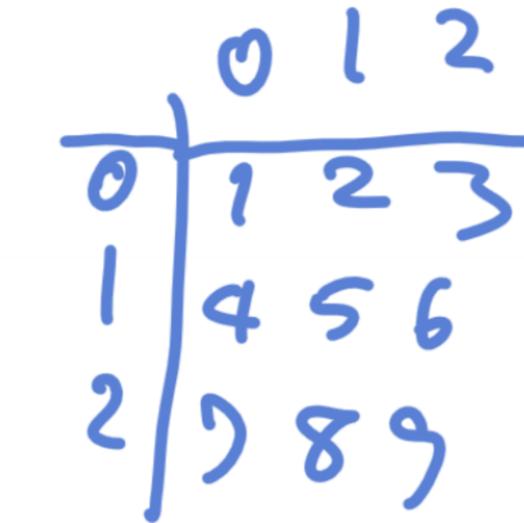
1 #include <stdio.h>
2 #include <conio.h>
3 int main()
4 {   int table[12][11], row, col ;
5     printf(" *** multiplication table ***");
6     for (row=1; row <= 12; row++) {
7         printf("\n");
8         for (col=2; col<=12; col++) {
9             table[row-1][col-2]=row*col;
10            printf(" %3d ",table[row-1][col-2]);
11        }
12    }
13 }

```

การประกาศและกำหนดค่าตัวแปรชุด 2 มิติ(two dimension arrays)

```

1 #include <stdio.h>
2 int main()
3 { int i, j, n=3, C[3][3];
4   int A[3][3]={ {1,2,3}, {4,5,6}, {7,8,9} };
5   int B[3][3]={ {5,6,7}, {8,9,10}, {11,12,13} };
6   for(i=0;i<n;i++) {
7     for(j=0;j<n;j++) {
8       C[i][j]= A[i][j]+B[i][j];
9     }
10  }
11  printf("Result:\n");
12  for(i=0;i<n;i++) {
13    for(j=0;j<n;j++) {
14      printf("%d ",C[i][j]);
15    }
16  }
17 }
18 }
```



การประกาศและกำหนดค่าตัวแปรชุด 3 มิติ(Three dimension arrays)

- ตัวแปร array 3 มิติ มีการประกาศ ดังนี้

type arrayname[p][r][c];

— type คือ ชนิดของตัวแปร เช่น int ,float,char

— arrayname คือชื่อของตัวแปร

— r,c,p คือตัวเลขแสดงจำนวนในมิติที่ 1 มิติที่ 2 และมิติที่ 3 ของ array ตามลำดับ

— โดยตัวเลขกำกับตำแหน่ง(ด้านใน)เป็นดังนี้

- p เป็น 0,1,2 .. , p-1

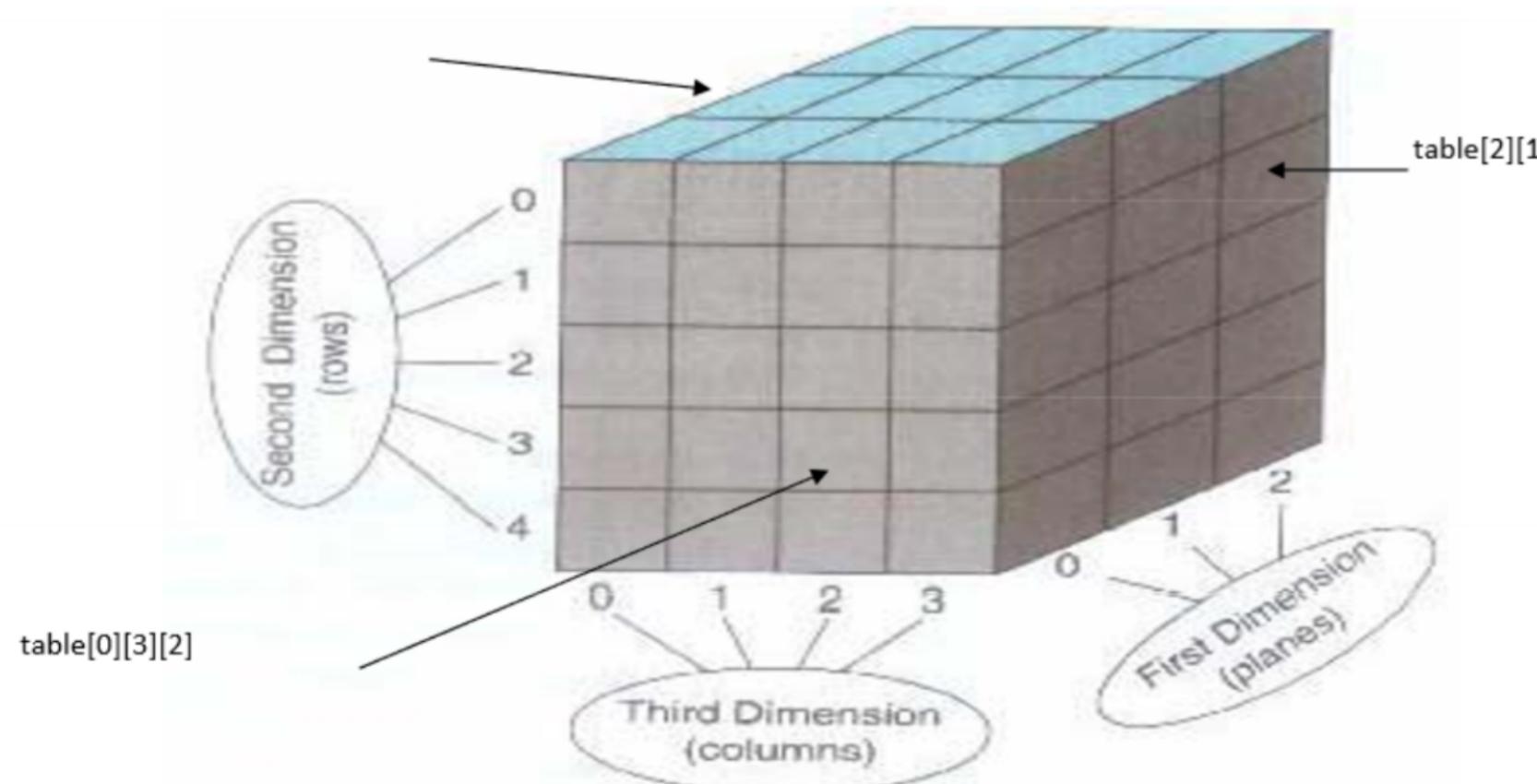
- r เป็น 0,1,2, ... ,r-1

- c เป็น 0,1,2 ... ,c-1

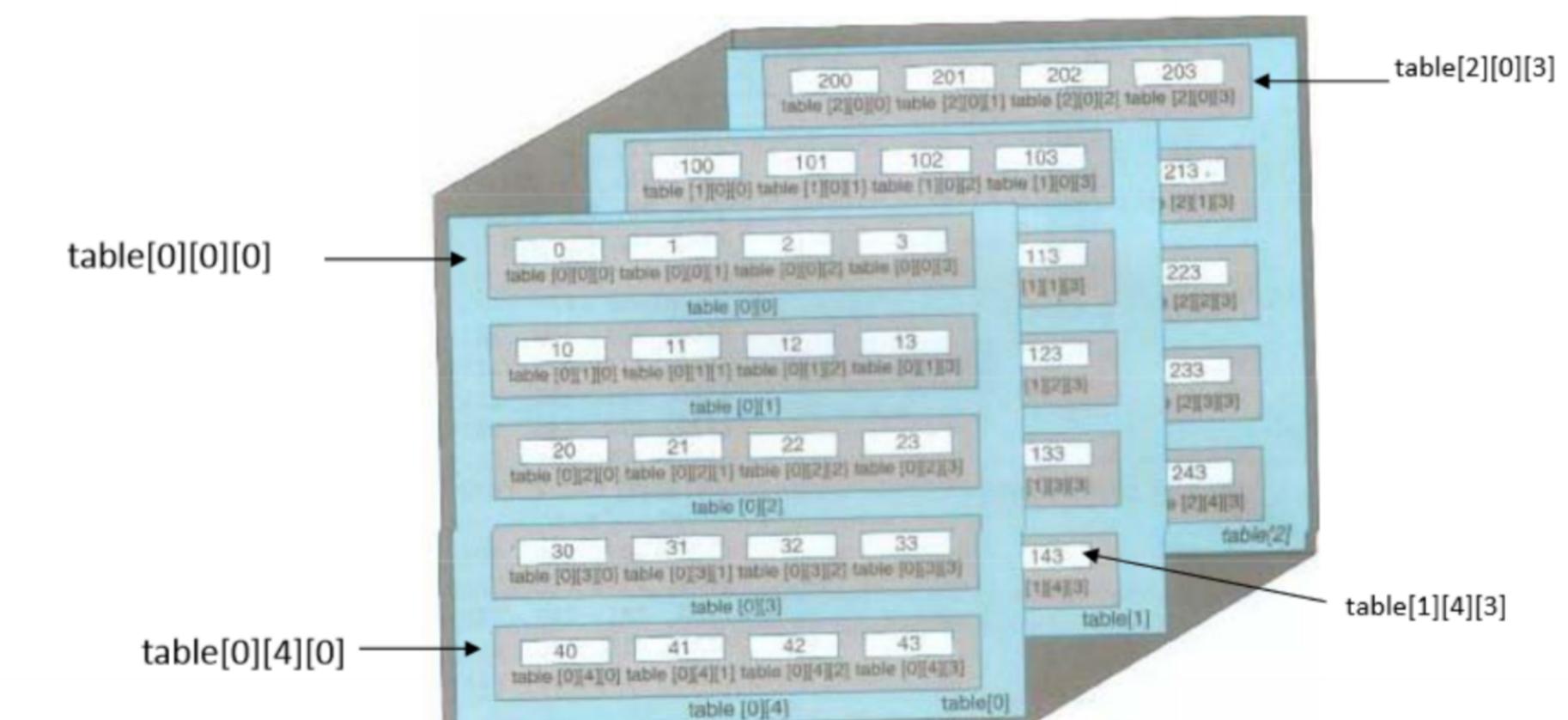
การประกาศและกำหนดค่าตัวแปรชุด 3 มิติ(Three dimension arrays)

- ลักษณะของ array 3 มิติ อาจเปรียบเทียบว่าเป็น arrays of arrays

ดังรูป



การประกาศและกำหนดค่าตัวแปรชุด 3 มิติ(Three dimension arrays)



การประมวลและกำหนดค่าตัวแปรชุด 3 มิติ(Three dimension arrays)

```
• #include <stdio.h>
int main()
{
    int arr[3][4][5];
    int i, j, k, sum = 0;
    for(i = 0; i < 3; i++)
        for(j = 0; j < 4; j++)
            for(k = 0; k < 5; k++)
            {
                scanf("%d", &arr[i][j][k]);
                sum = sum + arr[i][j][k];
            }
    printf("sum is %d", sum);
}
```

