การเขียนโปรแกรมคอมพิวเตอร์

การโปรแกรมโครงสร้างแบบเลือกทำ

อาจารย์ โอภาส วงษ์ทวีทรัพย์ (อ.โอ๊ต)
ศูนย์ปฏิบัติการวิจัย และพัฒนาระบบสารสนเทศอัจฉริยะ
ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร
E-Mail :: oatcomster@gmail.com

ขอบเขตของเนื้อหา

- ทบทวนการเขียนโปรแกรมแบบโครงสร้าง *(Structured Programming)*
- การเขียนโปรแกรมแบบเลือกทำ *(Selection Statement)*
- นิพจน์ทางตรรกศาสตร์ *(Boolean Expression)*
- ลำดับความสำคัญของตัวดำเนินการทางตรรกศาสตร์ *(Precedence of Boolean Operators)*
- ตัวอย่างคำสั่ง และการประยุกต์ใช้งานในโปรแกรม

การเขียนโปรแกรมแบบโครงสร้าง

- การเขียนโปรแกรมแบบโครงสร้าง หรือ *Structured Programming* นั้น เป็นแนวคิดและวิธีการที่ใช้ในการเขียนโปรแกรมคอมพิวเตอร์ ตั้งแต่แรกเริ่มที่มีคอมพิวเตอร์ขึ้นมา
- โดยการเขียนโปรแกรมในลักษณะนี้ จะมีการเขียนในลักษณะของโครงสร้างของคำสั่ง 3 รูปแบบ ด้วยกัน กล่าวคือ
- โครงสร้างแบบเรียงลำดับ *(Sequence Statement)* โครงสร้างแบบเลือก *(Selection Statement)* และโครงสร้างแบบทำซ้ำ *(Iteration Statement)*

การเขียนโปรแกรมแบบโครงสร้าง(2)

- สามารถมีโครงสร้างต่างๆ ซ้อนกันได้ *(Nested Structure)*
- การเขียนโปรแกรมนั้นจะต้องมีทางเข้าและทางออกของโปรแกรมเพียงทางเดียวเท่านั้น
- มีจุดเริ่มต้น และจุดสิ้นสุดของโปรแกรมอย่างชัดเจน
- ไม่มีการใช้คำสั่ง GOTO ถ้าไม่จำเป็น เพราะจะทำให้ขาดความเป็นโครงสร้าง *(UnStructure)* อีกทั้งอาจก็ให้เกิดปัญหา *Spaghetti Code* ขึ้นมาได้

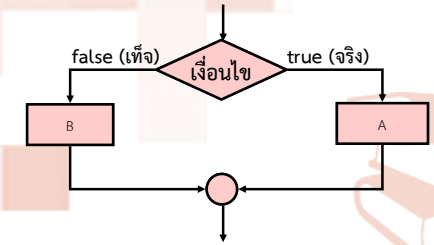
การเขียนโปรแกรมแบบโครงสร้าง(3)

- โดยโครงสร้างแบบเรียงลำดับ *(Sequence Statement)* ที่เราได้เรียนไปนั้น ประกอบไปด้วยคำสั่งดังนี้
- คำสั่งรับค่า และแสดงผลลัพธ์ของข้อมูล *(Input/Output Statement)*
  - ตัวอย่างของคำสั่งรับค่า คือ *scanf*
  - ตัวอย่างของคำสั่งแสดงผลลัพธ์ คือ *printf*
- นิพจน์ *(Expression)* ที่เราเขียนขึ้นเพื่อสร้างนิพจน์ในการคำนวณ โดยการใช้ตัวดำเนินการ *(Operators)* และตัวถูกดำเนินการ *(Operands)* เป็นองค์ประกอบหลัก

การเขียนโปรแกรมแบบโครงสร้าง(4)

- การเขียนโปรแกรมแบบเรียงลำดับนั้น โดยการเขียนปกติ เรามักจะพบว่า ก็จะใช้ในการแก้ปัญหาแบบ รับค่าเข้ามา ประมวลผล และก็แสดงผลลัพธ์ออกไป
- แต่ในแง่ความเป็นจริง เราพบว่าการแก้ไขปัญหานั้น ไม่ได้เป็นเส้นตรงที่แก้ไขได้แบบนั้น เพราะว่าบางครั้งต้องมีส่วนของการเลือกทำ ว่าจะทำหรือไม่ทำ
- โดยการใช้โครงสร้างคำสั่งแบบเลือกทำ *(Selection Statement)*
- เข้ามาช่วยในการตัดสินใจ ว่าจะดำเนินการหรือทำงานอย่างใดดี

โครงสร้างแบบมีทางเลือก



นิพจน์ทางตรรกศาสตร์

- ในการเขียนโปรแกรมแบบเลือกทำ *(Selection Statement)* นั้น จะมีการตรวจสอบเงื่อนไข โดยเงื่อนไขนั้น จะเขียนในรูปของนิพจน์ทางตรรกศาสตร์ *(Boolean Expression)*
- นิพจน์ทางตรรกศาสตร์ คือ นิพจน์ที่เขียนขึ้น และพบปรากฏว่ามีตัวดำเนินการทางตรรกศาสตร์อยู่ *(Boolean Operator)*
- โดยตัวดำเนินการทางตรรกศาสตร์นั้น สามารถแบ่งได้เป็น
  - *Logical Operator* ได้แก่ ! (NOT) , && (AND) , || (OR)
  - *Relational Operator* ได้แก่ = = , != , < , > , <= , >=

นิพจน์ทางตรรกศาสตร์ (2)

- ข้อควรระวัง ในการเปรียบเทียบเงื่อนไข ถ้าเราต้องการเปรียบเทียบการเท่ากัน เราจะใช้ตัวดำเนินการ == (เขียน 2 อันติดกัน)
- เราจะไม่ใช้ตัวดำเนินการ = เพราะเนื่องจาก = (เท่ากับ) นั้นจะถูกใช้งานในนิพจน์ทางคณิตศาสตร์แล้ว (เช่น *sum = a + b;*)
- ในภาษาซีนั้นไม่มีการประกาศตัวแปรชนิด *boolean* ขึ้นเพื่อใช้งาน แต่ในการทำงานจะใช้เลขจำนวนเต็ม *(Integer)* แทนค่าความจริง
  - ถ้าค่าของตัวเลขมีค่าเท่ากับ 0 แสดงว่าเป็นเท็จ *(False or Zero)*
  - ถ้าค่าของตัวเลขมีค่าไม่เท่ากับ 0 แสดงว่าเป็นจริง *(True or Non-Zero)*

ตัวอย่างของนิพจน์ทางตรรกศาสตร์

ความหมาย	สัญลักษณ์	ตัวอย่าง	ค่าของนิพจน์
เท่ากับ	==	2 == 3	false
มากกว่า	>	20 > 5	true
น้อยกว่า	<	7 < 2	false
มากกว่าหรือเท่ากับ	>=	4 >= 4	true
น้อยกว่าหรือเท่ากับ	<=	7 <= 6	true
ไม่เท่ากับ	!=	5 != 5	false

ตัวอย่างของนิพจน์ทางตรรกศาสตร์ (2)

- การเปรียบเทียบจะนำตัวแปรที่ต้องการเปรียบเทียบไว้ทางด้านซ้ายมือของตัวดำเนินการเปรียบเทียบ ส่วนค่าที่ต้องการจะนำมาเปรียบเทียบนั้นจะไว้ทางด้านขวามือของตัวดำเนินการ เช่น

ตัวที่นำมาเปรียบเทียบด้วยซึ่งอาจเป็นค่าคงที่หรือตัวแปรก็ได้

X >100

ตัวที่ต้องการเปรียบเทียบ

ตัวอย่างของนิพจน์ทางตรรกศาสตร์ (3)

X = 25

Y = -85

นิพจน์ทางตรรกศาสตร์	ค่าของนิพจน์
X == Y	False
X > 0	True (check ว่า X เป็นเลขบวก?)
Y < 0	True (check ว่า Y เป็นเลขลบ?)
X != Y	True
(X > 0) && (Y > 0)	False (check ว่าเป็นเลขบวกทั้งคู่?)

ลำดับความสำคัญของตัวดำเนินการ

1.คำนวณค่าของนิพจน์ทางคณิตศาสตร์ก่อน และถ้ามีวงเล็บต้องทำในวงเล็บก่อน
2.ในกรณีที่มีวงเล็บซ้อนกัน ให้ทำเครื่องหมายวงเล็บในสุดก่อน
3.คำนวณตามลำดับ แต่ถ้ามีลำดับเดียวกันหลายตัวให้ดำเนินการ ซ้าย --> ขวา

ลำดับการคำนวณ	เครื่องหมาย
1.	! (NOT)
2.	> , < , >= , <= , != , ==
3.	&& (AND) , || (OR)

ตัวอย่าง ((X>5)&&(Y<7))||(Z<=20)

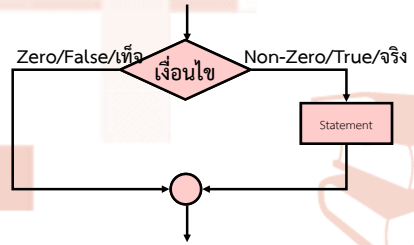
การเขียนโปรแกรมแบบเลือกทำ

- การเขียนโปรแกรมแบบเลือกทำ *(Selection Statement)* เป็นโครงสร้างที่มีการทดสอบเงื่อนไขทางตรรกศาสตร์ *(Logical Decision)*
- เพื่อตัดสินใจทำงานอย่างใดอย่างหนึ่ง โดยจะมีการข้ามไม่ทำงานบางประโยคคำสั่งไป ซึ่งโดยทั่วไปนั้นมีโครงสร้างอยู่ 2 รูปแบบคือ โครงสร้างแบบ *if* และ *switch-case*
- โดยโครงสร้างแบบ *if Statement* จะมีรูปแบบการใช้งานโดยทั่วไปจำนวน 3 รูปแบบ ได้แก่ *if, if – else* และ *Nested - if*

การเขียนโปรแกรมแบบเลือกทำ (2)

- ประเภทของการเขียนโปรแกรมแบบเลือกทำ *(Selection Statement)* โดยทั่วๆ ไปนั้น จะแบ่งเป็น 4 รูปแบบหลักๆ
  1.รูปแบบของคำสั่ง *if*
  2.รูปแบบของคำสั่ง *if – else*
  3.รูปแบบของคำสั่ง *Nested - if*
  4.รูปแบบของคำสั่ง *Switch - Case*

โครงสร้างของคำสั่ง if



รูปแบบของคำสั่ง if

```
if (boolean expression) statement;
```

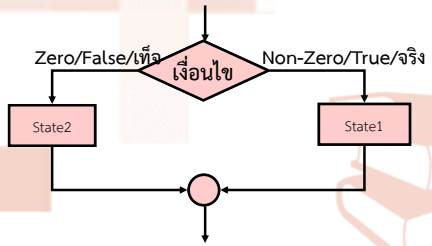
- *Boolean expression* หมายถึง นิพจน์ทางตรรกศาสตร์ ที่จะมีการเปรียบเทียบเงื่อนไขว่ามีค่าเป็นเท่าไหร่ ไม่ใช่ศูนย์ หรือศูนย์
- *Statement* ที่อยู่หลัง *Boolean expression* หมายถึง คำสั่งหรือกลุ่มของคำสั่งที่จะต้องทำถ้านิพจน์เป็นจริง และถ้าเป็นเท็จก็จะมีการทำคำสั่งถัดไป

ตัวอย่างการใช้งานคำสั่ง if

- วิเคราะห์การทำงาน คิดว่าเขียนแบบนี้ดีไหม???
- แล้วมีวิธีการอื่นหรือไม่ ที่จะให้ทำงานได้เร็วขึ้น ไม่เสียเวลากับการตรวจสอบเงื่อนไข

```
if(num ==0) printf("Zero Number");
if(num > 0) printf("Positive Number");
if(num < 0) printf("Negative Number");
```

โครงสร้างของคำสั่ง if - else



รูปแบบของคำสั่ง if - else

```
if (boolean expression) statement1;
else   statement2;
```

- *Statement1* ที่อยู่หลัง *Boolean expression* หมายถึง คำสั่งหรือกลุ่มของคำสั่งที่จะต้องทำถ้านิพจน์การเปรียบเทียบเงื่อนไขเป็นจริง (มีค่าไม่เป็น 0)
- *Statement2* ที่อยู่หลัง *else* หมายถึง คำสั่งหรือกลุ่มของคำสั่งที่จะต้องทำถ้านิพจน์การเปรียบเทียบเงื่อนไขเป็นเท็จ (มีค่าเป็น 0)

ตัวอย่างการใช้งานคำสั่ง if - else

- วิเคราะห์การทำงาน คิดว่าเขียนแบบนี้ดีไหม???
- แล้วมีวิธีการอื่นหรือไม่ ที่จะให้ทำงานได้เร็วขึ้น ไม่เสียเวลากับการตรวจสอบเงื่อนไข

```
if(num ==0) printf("Zero Number");
else if(num > 0) printf("Positive Number");
     else if(num < 0) printf("Negative Number");
```


ตัวอย่างการใช้งานคำสั่ง if – else (2)

- เนื่องจากเราต้องการตรวจสอบเลขจำนวนเต็มที่รับเข้ามา ซึ่งเลขจำนวนเต็มจะมีอยู่ 3 แบบ คือ จำนวนเต็มบวก เต็มลบ และเต็มศูนย์ ดังนั้น เราไม่จำเป็นต้องตรวจสอบเงื่อนไขสุดท้าย

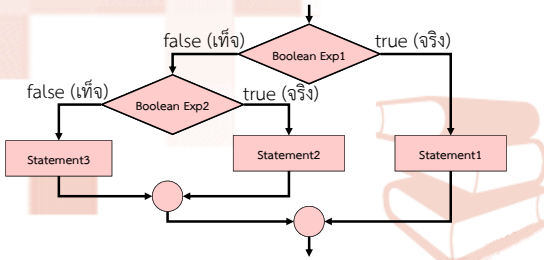
```
if(num ==0) printf("Zero Number");
else if(num > 0) printf("Positive Number");
     else if(num < 0) printf("Negative Number");
```


ตัวอย่างการใช้งานคำสั่ง if – else (2)

- หลังจากตัดเงื่อนไขสุดท้ายออก เพราะว่ายังไงก็ต้องเป็นกรณีสุดท้ายแน่ๆ ไม่ต้องตรวจสอบแล้ว ลำดับของการเขียนเงื่อนไขก็มีผลกับประสิทธิภาพของการเขียนโปรแกรมเหมือนกัน

```
if(num > 0) printf("Positive Number");
else if(num < 0) printf("Negative Number");
     else printf("Zero Number");
```

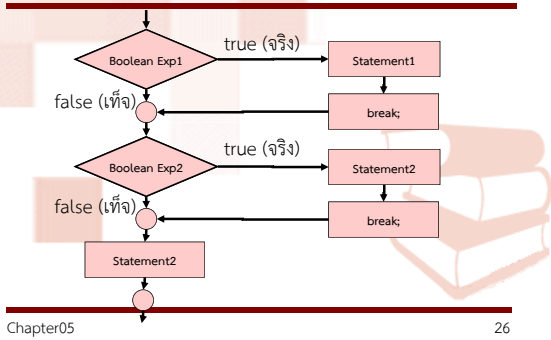

โครงสร้างของคำสั่ง Nested - if



รูปแบบของคำสั่ง Switch - case

```
switch(expression)
{
    case list value 1  :  Statement 1;break;
    case list value 2  :  Statement 2;break;
    ...          ...
    case list value N  :  Statement N;break;
    [default : StatementElse;break; ]
}
```


โครงสร้างของคำสั่ง Switch - case



ตัวอย่างการใช้งานคำสั่ง Switch - case

```
char color,r;
scanf("%c",&color);
switch(color) {
    case 'r'  :  printf("Red");
    case 'b'  :  printf("Blue");
    case 'v'  :  printf("Violet");
    default   : printf("Unknow");
}
```

ตัวอย่างการใช้งานคำสั่ง Switch – case (2)

```
scanf("%c",&color);
switch(color)
{
    case 'r'  :  printf("Red");
    case 'b'  :  printf("Blue");
    case 'v'  :  printf("Violet");
    case 'r'  :  printf("Red");
    default   : printf("Unknow");
}
```


ตัวอย่างการใช้งานคำสั่ง Switch – case (3)

```
scanf("%c",&color);
switch(color)
{
    case 'r'  :  printf("Red");break;
    case 'b'  :  printf("Blue");break;
    case 'v'  :  printf("Violet");break;
    default   : printf("Unknow");
}
```

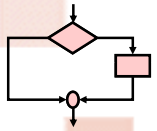

ตัวอย่างการใช้งานคำสั่ง Switch – case (4)

```
scanf("%c",&color);
switch(color)
{
    case 'r'  :  printf("Red");break;
    case 'b'  :  printf("Blue");break;
    case 'v'  :  printf("Violet");break;
    default   : printf("Unknow");break;
}
```

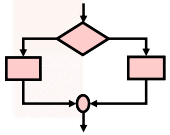

ตัวอย่างการใช้งานคำสั่ง Switch – case (5)

```
scanf("%c",&color);
switch(color)
{
    case 'r' : case 'R' :  printf("Red");break;
    case 'b' : case 'B' : printf("Blue");break;
    case 'v' : case 'V' : printf("Violet");break;
    default  : printf("Unknow");break;
}
```

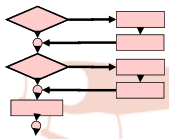

สรุปโครงสร้างแบบเลือกทำ



แบบเลือกทำหรือไม่ทำ
if - statement



แบบเลือกทำอย่างใดอย่างหนึ่ง
if - else



แบบหลายทางเลือกทำ
switch - case

ตัวอย่างการประยุกต์ใช้งาน

- การตรวจสอบว่าจำนวนที่รับมานั้นเป็นเลขคี่หรือเลขคู่

```
if (NUM % 2 == 0) printf("EVEN  NUMBER");
else printf("ODD NUMBER");
```

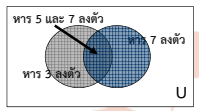
- การตรวจสอบว่ามีการหารลงตัวด้วยเลขจำนวนนั้นหรือไม่ เช่น ต้องการตรวจสอบว่าหารด้วย 5 ลงตัวหรือไม่

```
if (NUM % 5 == 0) printf("YES");
else printf("NO");
```


ตัวอย่างการประยุกต์ใช้งาน (2)

- การตรวจสอบเลขที่รับเข้ามาหารลงตัวด้วย 3 และ 7 หรือไม่

```
if (NUM%3 == 0 && NUM%7 == 0) printf("YES");
else printf("NO");
```




```
if (NUM%21== 0) printf("YES");
else printf("NO");
```

- การตรวจสอบเลขที่รับเข้ามาหารลงตัวด้วย 3 หรือ 7 หรือไม่

```
if  (NUM%3 == 0 || NUM%7 == 0) printf("YES");
else printf("NO");
```


ตัวอย่างการประยุกต์ใช้งาน (3)

การตรวจสอบว่าคะแนนของนักเรียนนั้นอยู่ในช่วง 80-100 หรือไม่ เราจะใช้ Logical Operator ตัวใดดีระหว่าง && (AND) กับ || (OR)

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


```
if (score >= 80 && score <= 100) printf("Grade A!!!");
else printf("Other Grade!!!");

if (score >= 80 || score <= 100) printf("Grade A!!!");
else printf("Other Grade!!!");
```


ตัวอย่างการประยุกต์ใช้งาน (4)

```
#include <stdio.h>
int main() {
    int score; char grade;
    printf("Input Score : ");scanf("%d",&score);
    if  (score >= 80) grade = 'A';
    else if  (score >= 70) grade = 'B';
      else if  (score >= 60) grade = 'C';
        else if  (score >= 50) grade = 'D';
          else grade= 'F';
    printf("Your Grade = %c",grade);
    return(0);
}
```

Examination

1.เขียนโปรแกรมรับค่าตัวแปร X จากนั้นทำการคำนวณค่าของ Y

X = 1,2	Y = 2X + 1
X = 4,6 - 8	Y = 4X
X = 9,10,13,15 - 20	Y = 20 - X
X = Other	Y = X

Input Number : 20

Y = 20 – 20 = 0

Examination

2. จงเขียนโปรแกรมรับคะแนน midterm และ final ของนักศึกษา(คะแนนเต็ม 100 คะแนน) หลังจากนั้นทำการคำนวณเกรดที่ได้ของนักศึกษา ซึ่งคะแนนสุดท้ายที่นำมาคิดเกรดนั้นคือ 100 คะแนนซึ่งคิดจากคะแนน midterm 40 % และคะแนน final 60 %