ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

Computer Programming II
การเขียนโปรแกรมคอมพิวเตอร์2
LECTURE#4 Control Statement

อ.สถิตย์ ประสมพันธ์

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

KMUTNB

คำสั่งควบคุม (Control Statement)

- โดยปกติการทำงานของคอมพิวเตอร์จะทำงานเรียงลำดับคำสั่งลงมาตั้งแต่ต้นโปรแกรมจนจบโปรแกรม
- ถ้าต้องการเปลี่ยนแปลงขั้นตอนการทำงานของคำสั่ง เช่น กระโดดข้ามไปที่ คำสั่งใดคำสั่งหนึ่ง หรือให้วนกลับมาทำคำสั่งที่เคยทำไปแล้ว ลักษณะการสั่งงานแบบนี้จะต้องใช้คำสั่งควบคุม
- คำสั่งควบคุมจึงเป็นคำสั่งที่ใช้เปลี่ยนแปลงลำดับขั้นตอนการทำงานของโปรแกรม

คำสั่งควบคุม

- คำสั่งควบคุมแบ่งออกเป็น 3 ชนิด คือ
  - 1. คำสั่งให้ไปทำงานโดยไม่มีเงื่อนไข (Unconditional branch Statement) ได้แก่คำสั่ง goto
  - 2. คำสั่งให้ไปทำงานโดยมีเงื่อนไข (Condition Statement) ได้แก่คำสั่ง if, switch
  - 3. 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

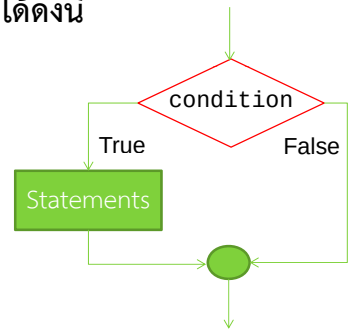
```
printf("Hello world");
```

คำสั่ง if

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

รูปแบบที่ง่ายที่สุดของประโยค if แสดงได้ดังนี้

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



โดยที่ condition คือเงื่อนไขสำหรับตัดสินใจ

Statements คือชุดคำสั่งที่จะถูกทำเมื่อเงื่อนไขใน condition เป็นจริง

คำสั่ง if

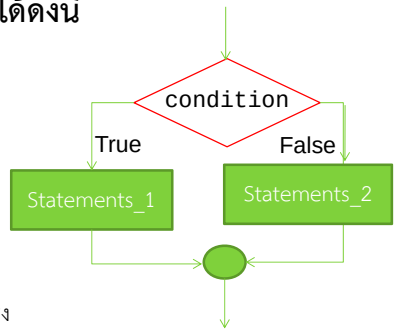
```
#include <stdio.h>
int main()
{
  int score;
  scanf("%d",&score);
  if (score >= 40)
     printf("Congratulations! You Passed \n");
  printf("See you again \n");
}
```


คำสั่ง if-else

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

รูปแบบที่ง่ายที่สุดของประโยค if แสดงได้ดังนี้

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



โดยที่

condition คือเงื่อนไขที่ให้ผลลัพธ์เป็นเลขจำนวนจริง

ค่าผลลัพธ์ที่เป็น ศูนย์ หรือ NULL หมายถึงเท็จ

ค่าผลลัพธ์ที่ไม่ใช่ ศูนย์ หรือไม่ใช่ NULL หมายถึงจริง

ส่วน statements_1 คือชุดคำสั่งที่จะทำกรณี เงื่อนไขเป็นจริง

ส่วน statements_2 คือชุดคำสั่งที่จะทำกรณีเงื่อนไขเป็นเท็จ

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)

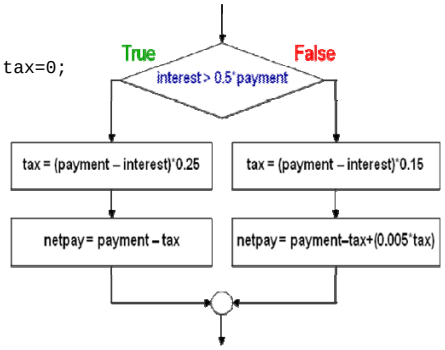
```c
#include <stdio.h>
int main (void)
{
   int month;
   scanf("%d",&month);
   if (month == 12||month == 1||month == 2)
         printf("Winter");
   else  if (month == 3||month == 4||month == 5)
         printf("Spring");
   else  if (month == 6||month == 7||month == 8)
         printf("Summer");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

   return  0;
}
```


Block

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

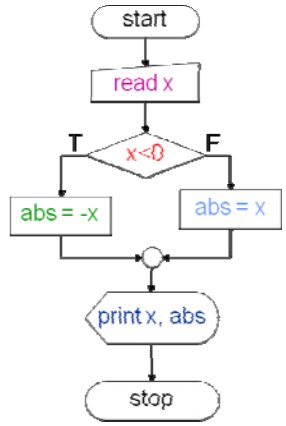
```c
#include <stdio.h>
int main()
{
    float payment=0,interest=0, netpay=0, tax=0;
    printf("Input interest, payment:");
    scanf("%f %f",&payment,&interest);
    if (interest>0.5*payment){
        tax= (payment -interest)*0.25;
        netpay= payment - tax;
    }
    else{
        tax= (payment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



Block

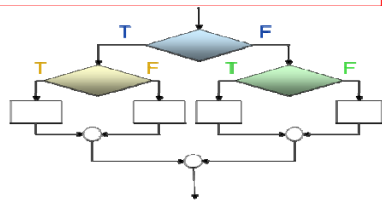
```c
#include <stdio.h>
int main()
{
  int abs, x;
  printf ("Input an integer : ");
  scanf ("%d", &x);
   if (x<0)
      abs = -x;
   else
      abs = x;
   printf ("abs(%d) = %d\n", x, abs);

   return 0;
}
```

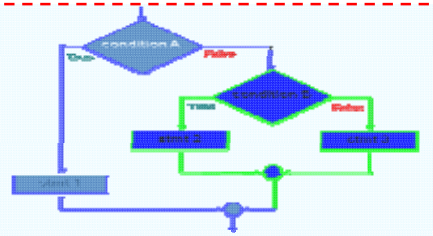


Nested if statements

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



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



```
if (condition A)
        stmt1;
else if (condition B)
        stmt2;
else
stmt3;
```

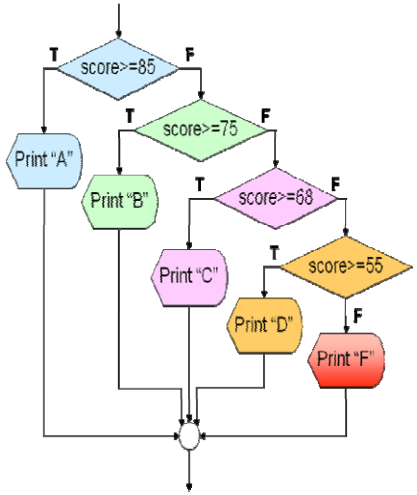

Nested if statements

```
if(test_condition1)
  {
     if(test_condition2)
      {
             ----------------------
             ----------------------  // Statements1 Or True Block;

      }
      else
      {
             ----------------------
             ----------------------  // Statements2 Or False Block;

      }
  }
  else {
          -------------------------
          ------------------------- // Statement 3 or false block
  code for test_condition1.
        }
```


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");

   return 0;
}
```


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.

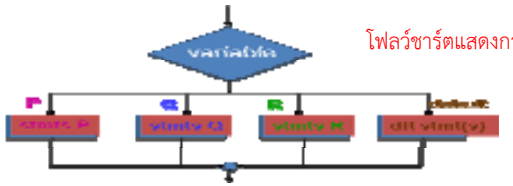
1. T<=0 "It's very very cold".
2. 0 < T < 0 "It's cold".
3. 10 < T < =20 "It's cool out".
4. 20 < T < =30 "It's warm".
5. T>30 "It's hot".

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

คำสัง switch เป็นคำสั่งสำหรับการเลือกใช้คำสั่งหรือกลุ่มของคำสั่งจากหลาย ๆ กลุ่ม



โฟลว์ชาร์ตแสดงการทำงานของคำสั่ง switch

variable คือ ตัวแปรที่จะใช้ในการทดสอบเงื่อนไข

P Q R คือ case หรือตัวอย่างของค่าที่จะถูกนำมาทดสอบกับค่าที่อยู่ภายในตัวแปร(variable)

กรณีที่ค่าในตัวแปรมีค่าตรงกับ case ใดก็จะทำงานตามคำสั่งที่อยู่ภายใต้ case นั้นๆ

หากไม่ตรงกับ case ใด ๆ เลยก็จะเข้าสู่การทำงานในส่วนของ default

ถ้ามี default เมื่อตรวจสอบค่าของตัวแปรหรือนิพจน์แล้วไม่ตรงกับ case ใดๆ ก็จะเข้ามาทำในส่วนของ default

ถ้าไม่มี default เมื่อตรวจสอบค่าของตัวแปรหรือนิพจน์แล้วไม่ตรงกับ case ใดๆ 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 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

```c
#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 แล้วไปทำคำสั่งแรกที่ตามหลังคำสั่ง switch

```c
#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
```