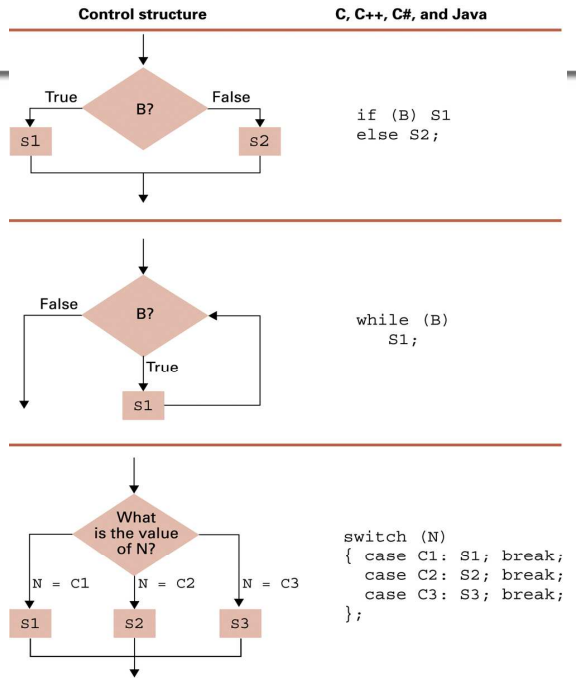
คำสั่งควบคุม
Control Statements

- เลือกดำเนินการโดยมีเงื่อนไข
- ทำซ้ำ
- กระโดดหรือควบคุมโดยไม่มีเงื่อนไข

Control structures and their representations in C, C++, C#, and Java

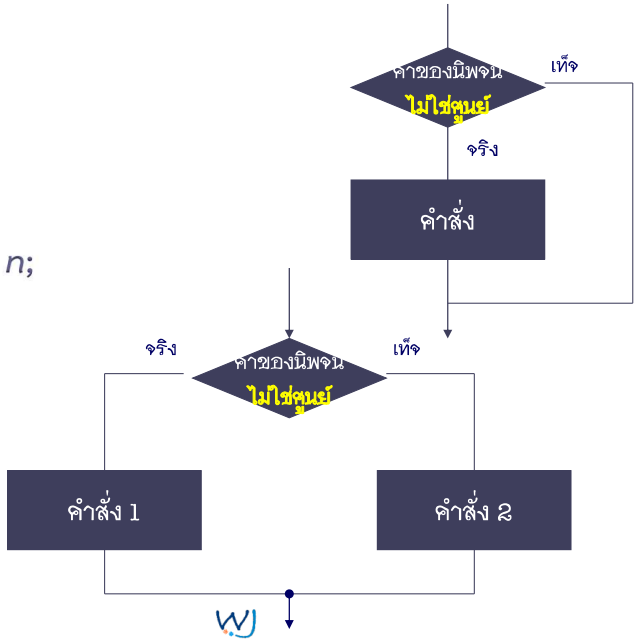


คำสั่งควบคุมขั้นตอน
Flow control statements

- คำสั่งที่จะส่งผลให้ลำดับขั้นตอนการทำงาน ไม่เป็นไปตามลำดับที่เขียนคำสั่ง เช่นทำให้เกิดการทำซ้ำ ทำให้เกิดการกระโดด หรือทำให้สามารถเลือกที่จะดำเนินการคำสั่งได้
- ตัวดำเนินการที่เกี่ยวข้อง
  - Relational operators
  - Equality operators
  - Logical operators
- คำสั่งควบคุม if-else, switch, while, do-while, for, break, continue

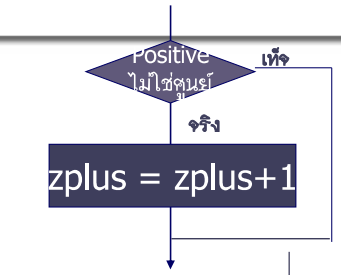
if - else statement

- if (*นิพจน์*)
  คำสั่ง;
- if (*นิพจน์*) {
  คำสั่ง1; ...คำสั่ง n;
  }
- if (*นิพจน์*)
  คำสั่ง1;
  else
  คำสั่ง2;

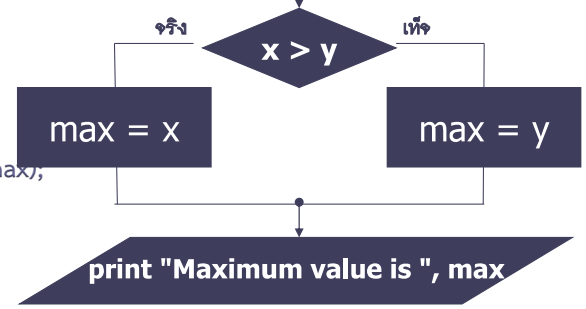


ตัวอย่าง

- if (positive)
  zplus = zplus + 1;



- if (x>y)
  max = x;
  else
  max = y;
  printf("Maximum value is %d\n", max);



- if (x > 0 && x < 100)
  printf ("Valid input ...\n");

Relational operators

- ตัวดำเนินการเปรียบเทียบ
- Binary operator
- กำหนดสัญลักษณ์แทนตัวดำเนินการดังนี้
  - < แทน น้อยกว่า
  - <= แทน น้อยกว่าหรือเท่ากับ
  - > แทน มากกว่า
  - >= แทน มากกว่าหรือเท่ากับ
- รูปแบบการใช้ตัวดำเนินการเปรียบเทียบ

  นิพจน์ 1 **ตัวดำเนินการเปรียบเทียบ** *นิพจน์ 2*
- ผลของการดำเนินการให้ค่าเป็นจริงหรือเท็จ
- ตัวอย่าง

```
printf("True value in C is %d\n",  1 < 10);
/* 1 is generated.*/
printf("False value in C is %d\n", 'A' >= 'a');
/* 0 is generated.*/
```


ลำดับความสำคัญของตัวดำเนินการ

ตัวดำเนินการ	ลำดับดำเนินการ
( )	ซ้ายไปขวา
+ (unary) -(unary)	ขวาไปซ้าย
* / %	ซ้ายไปขวา
+ -	ซ้ายไปขวา
< <= > >=	ซ้ายไปขวา
= += -= *= /= %=	ขวาไปซ้าย

การประกาศตัวแปรและกำหนดค่าเริ่มต้น

```
int     a=1,    b=2,    c=3;
double  x = 5.5, y = 7.7;
```

นิพจน์	นิพจน์เทียบเท่า	ค่าของนิพจน์
a < b - c	a < (b - c)	0
-a + 5* b >= c + 1	((-a) + (5* b)) >= (c + 1)	1
x - y <= b - c - 1	(x - y) <= ((b - c) - 1)	1
x + c + 7 < y / c	((x + c) + 7) < (y / c)	0

Relational operators(cont.)

ค่าของ exp1 - exp2	exp1<exp2	exp1>exp2	exp1<=exp2	exp1>=exp2
Positive	0	1	0	1
Zero	0	0	1	1
Negative	1	0	1	0

Equality operators

- ตัวดำเนินการเปรียบเทียบความเท่าเทียม
- **Binary operator**
- กำหนดสัญลักษณ์แทนตัวดำเนินการดังนี้
  - == แทน เท่ากับ
  - != แทน ไม่เท่ากับ
- รูปแบบการใช้ตัวดำเนินการเปรียบเทียบ

  นิพจน์ 1 **==** / **!=** *นิพจน์ 2*
- ผลของการดำเนินการให้ค่าเป็นจริงหรือเท็จ
- ตัวอย่าง

```
printf("True value in C is %d\n", 'A' != 'a'); /*1 is generated.*/
printf("False value in C is %d\n", 1 == 1.0); /*0 is generated.*/
```

ค่าของ exp1 - exp2	exp1==exp2	exp1 != exp2
Zero	1	0
Nonzero	0	1

ลำดับความสำคัญของตัวดำเนินการ

ตัวดำเนินการ	ลำดับดำเนินการ
( )	ซ้ายไปขวา
+ (unary) -(unary)	ขวาไปซ้าย
* / %	ซ้ายไปขวา
+ -	ซ้ายไปขวา
< <= > >=	ซ้ายไปขวา
== !=	**ซ้ายไปขวา**
= += -= *= /= %=	ขวาไปซ้าย

Equality operators

การประกาศตัวแปรและกำหนดค่าเริ่มต้น

```
int a=1, b=2, c=3;
```

นิพจน์	นิพจน์เทียบเท่า	ค่าของนิพจน์
a == b	b == a	0
a != c	c != a	1
a+b+c == -2*-c	((a+b)+c) == ((-2)* (-c))	1
c += b != c/b	c = (c+(b != (c/b)))	4

Logical operators

- ตัวดำเนินการตรรกะ
- กำหนดสัญลักษณ์แทนตัวดำเนินการดังนี้
  - ! แทน นิเสธ (NOT)
  - && แทน และ (AND)
  - || แทน หรือ (OR)
- รูปแบบการใช้ตัวดำเนินการเปรียบเทียบ

  ! นิพจน์

  นิพจน์ 1 **&&** / **||** *นิพจน์ 2*
- ผลของการดำเนินการให้ค่าเป็นจริงหรือเท็จ

Logical operators(cont.)

- ตารางค่าความจริงของ นิเสธ (NOT)

	นิพจน์	NOT นิพจน์
ค่าความจริง	จริง	เท็จ
	เท็จ	จริง

ในภาษา C	นิพจน์	! นิพจน์
	zero	1
	nonzero	0

- ตัวอย่าง

```
!ok                 !(w + 9.5)
cont = !ok          !!ok
```


Logical operators(cont.)

- ตารางค่าความจริงของ และ (AND) และ หรือ (OR)

x	y	x AND y	x OR y
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

- ในภาษา C

zero	zero	0	0
nonzero	zero	0	1
zero	nonzero	0	1
nonzero	nonzero	1	1

- ตัวอย่าง

```
x >= 5 && x <= 8        a || b
ans == 'y' || ok        0.8  && (-2 * x + 7)
```

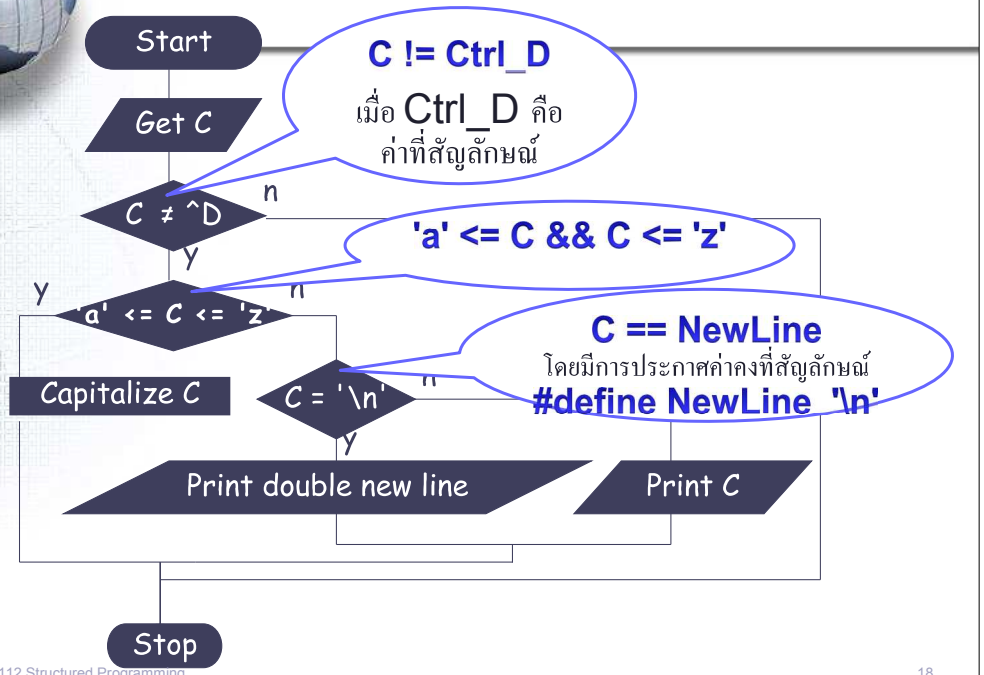

ลำดับความสำคัญของตัวดำเนินการ

ตัวดำเนินการ	ลำดับดำเนินการ
( )	ซ้ายไปขวา
+ (unary) -(unary) ! sizeof	ขวาไปซ้าย
* / %	ซ้ายไปขวา
+ -	ซ้ายไปขวา
< <= > >=	ซ้ายไปขวา
== !=	ซ้ายไปขวา
&&	ซ้ายไปขวา
||	ซ้ายไปขวา
= += -= *= /= %=	ขวาไปซ้าย

การประกาศตัวแปรและกำหนดค่าเริ่มต้น

```
int     a=1,    b=2,    c=3;
float   x=0.0,  y = 2.5;
```

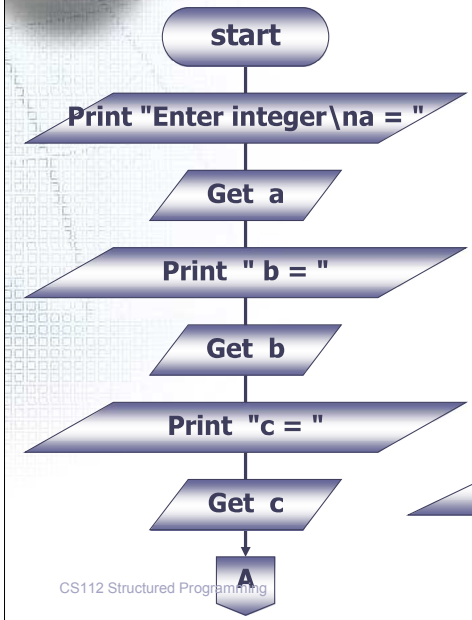
นิพจน์	นิพจน์เทียบเท่า	ค่าของนิพจน์
!(a-b) + 1	(!(a-b)) + 1	1
x || a && b - 2	x || (a && (b - 2))	0
a<b && !y	(a<b) && (!y)	0
a = 'A' && (c=!b)	(a = 'A') && (c=(!b))	0

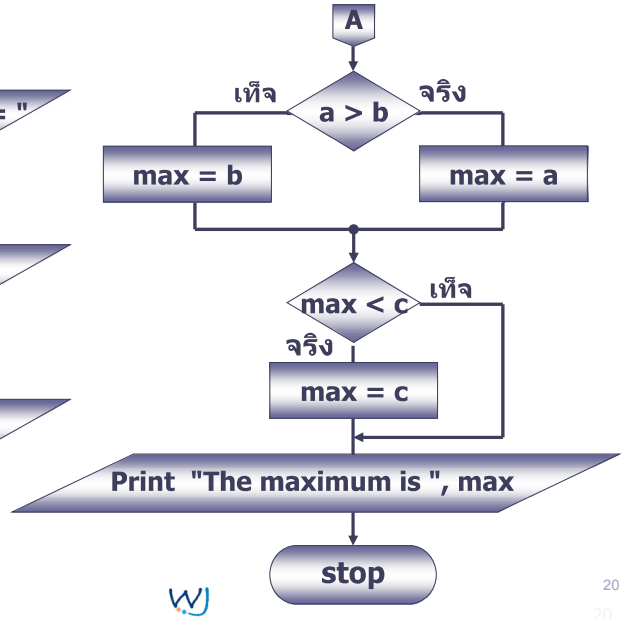


ตัวอย่าง

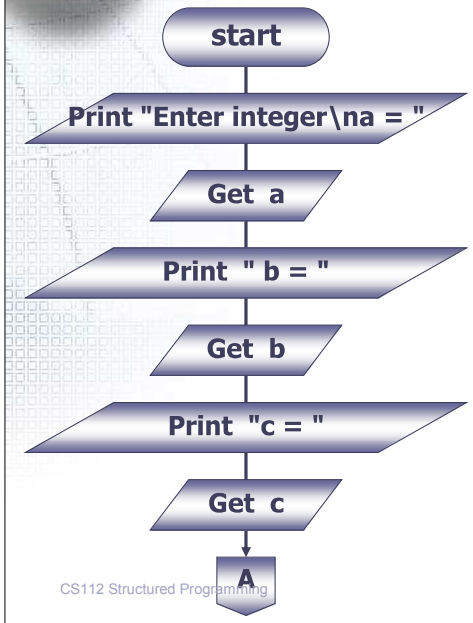
- จงเขียนโปรแกรมภาษาซี เพื่อหาค่ามากที่สุดของเลขจำนวนเต็ม 3 จำนวน
- วิเคราะห์
  - ผลลัพธ์ ค่าที่มากที่สุด (**max**)
  - ข้อมูลนำเข้า เลขจำนวนเต็ม 3 จำนวน (**a, b, c**)
  - วิธีประมวล
    - รับข้อมูล
    - เปรียบเทียบหาค่าที่มากที่สุด
    - แสดงผล

ตัวอย่าง (ต่อ)



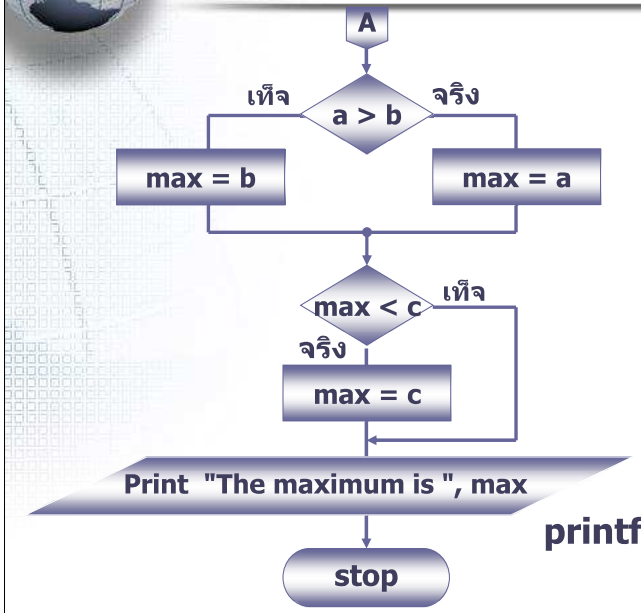


ตัวอย่าง (ต่อ)




```
int   a,  b,  c,  max;

printf("Enter integer \na = ");

scanf("%d", &a);

printf("b = ");

scanf("%d", &b);

printf("c = ");

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


ตัวอย่าง (ต่อ)




```
if (a > b)
        max = a;
else
        max = b;

if (max < c)
        max = c;
```

```
printf("The maximum is %d",
                        max);
```


ตัวอย่าง (ต่อ)

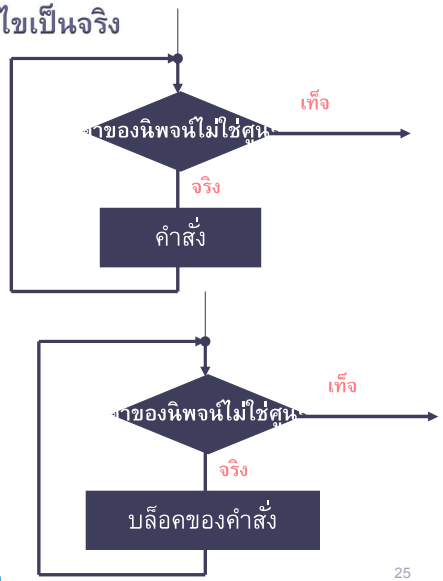
```
#include <stdio.h>
void main( ) {
        int   a,  b,  c,  max;
        printf("Enter integer \na = ");
        scanf("%d", &a);
        printf("b = ");
        scanf("%d", &b);
        printf("c = ");
        scanf("%d", &c);
        if (a > b)
                max = a;
        else
                max = b;
        if (max < c)
                max = c;
        printf("The maximum is %d", max);
}
```


nested if statement

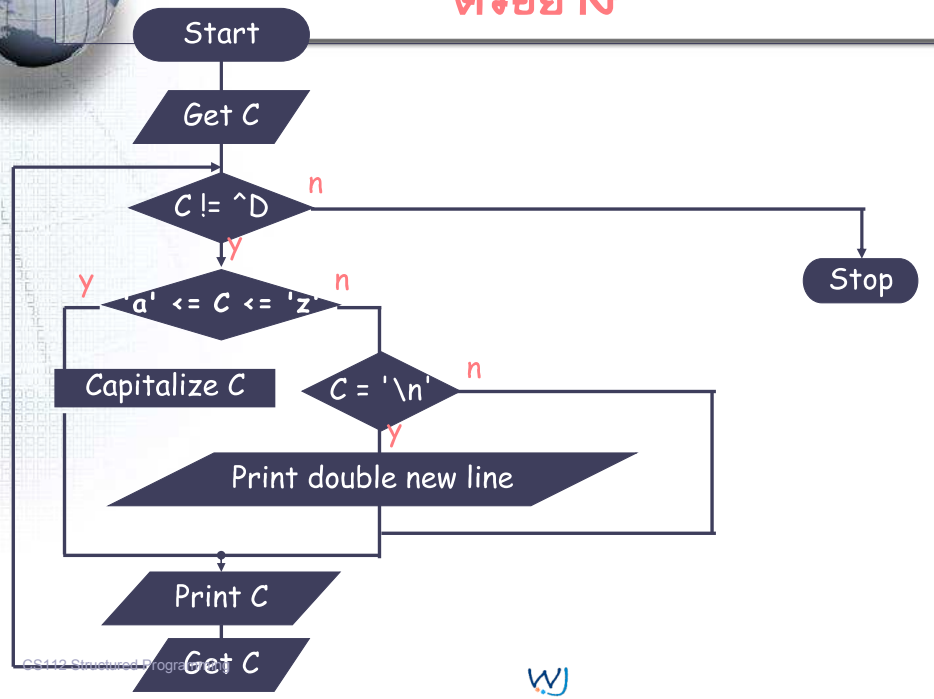
```
/* Another kind of nested if statement.*/
#define TRUE   1
:
printf("Enter the time as HH : MM  ");
scanf("%d : %d", &hour, &minute);
if (hour > 23)
    printf("Hour is too large.\n");
else if (hour < 0)
        printf("Hour is too small.\n");
     else if (minute > 59)
                printf("Minute is too large.\n");
          else if (minute < 0)
                    printf("Minute is too small.\n");
               else
                        inputok = TRUE;
/* inputok is a boolean variable which would be tested elsewhere in the program.  *
 * Note that no more than one message will be displayed                          */
```

while statements

- คำสั่งที่ทำให้เกิดการทำซ้ำในขณะที่เงื่อนไขเป็นจริง
- **while (นิพจน์)**
  คำสั่ง;
- **while (นิพจน์) {**
  คำสั่ง1; คำสั่ง2;
  ...คำสั่งn;
  }



ตัวอย่าง



ตัวอย่าง (ต่อ)

```
#include <stdio.h>
#define  ControlD    4
#define  NEWLINE   '\n'
void main ( ) {
        char  c;
        /* get characters, stop at end of file */
        scanf("%c", &c);
        while (c != ControlD) {
          if ('a' <= c && c <= 'z')
                c +=  'A' - 'a'; /*Capitalize a lowercase character */
          else
               if (c == NEWLINE)
                  printf("%c%c", NEWLINE, NEWLINE);
/*write two newline character to double line space */
          printf("%c", c);
          scanf("%c", &c);
        }
}
```


ตัวอย่าง

- จงเขียนโปรแกรมเพื่อทำการนับจำนวนหลักของเลขจำนวนเต็มที่ป้อน
- วิเคราะห์
  - ผลลัพธ์ จำนวนหลัก (digits)
  - ข้อมูลนำเข้า เลขจำนวนเต็ม 1 จำนวน (number)
  - วิธีประมวล
    - รับข้อมูล
    - จำนวนหลักของเลขจำนวนเต็มใดๆ จะเท่ากับจำนวนรอบของการหารด้วย 10 จนกระทั่งตัวตั้งของการหารเป็น 0
    - แสดงผล

ทดสอบผลการเรียนรู้ (ใช้เวลา 10 นาที)
• เตรียมกระดาษคนละ 1 แผ่น แล้วเขียนโปรแกรมภาษาซี โดยใช้ผลการวิเคราะห์ข้างต้นนี้

ตัวอย่าง (ต่อ)

```
#include <stdio.h>
void main ( ) {
        int    digits = 0, number;

        printf("Enter any integer number : ");
        scanf("%d", &number);
        while (number > 0) {
                number /= 10;
                digits += 1;
         }
         printf("No. of digits of your input is %d\n", digits);
}
```


do - while statements

- คำสั่งที่ทำให้เกิดการทำซ้ำถ้าเงื่อนไขยังคงเป็นจริง
- do

  คำสั่ง;

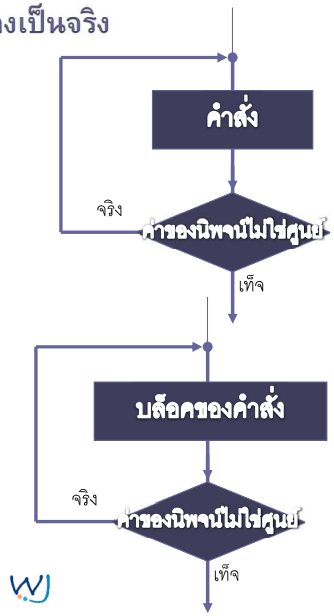
  while (นิพจน์);

- do {

  คำสั่ง1; คำสั่ง2;

  ...คำสั่งn;

  } while (นิพจน์);



ตัวอย่าง

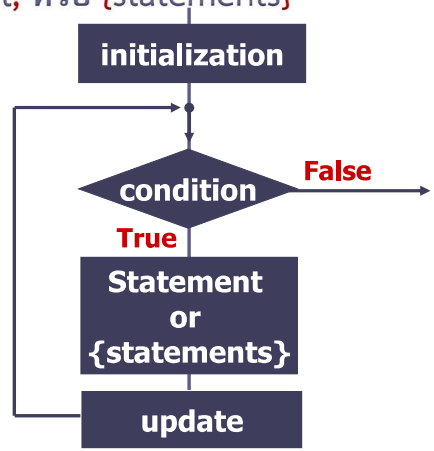
- จงเขียนโปรแกรมเพื่อทำการนับจำนวนหลักของเลขจำนวนเต็มที่ป้อน จนกว่าไม่ต้องการทำต่อ
- วิเคราะห์
  - ผลลัพธ์ จำนวนหลัก (digits)
  - ข้อมูลนำเข้า เลขจำนวนเต็ม 1 จำนวน (number)
    ทำงานต่อหรือไม่ (ans)
  - วิธีประมวล
    - รับข้อมูล
    - จำนวนหลักของเลขจำนวนเต็มใดๆ จะเท่ากับจำนวนรอบของการหารด้วย 10 จนกระทั่งตัวตั้งของการหารเป็น 0
    - แสดงผล
    - ถามความต้องการทำงานต่อหรือไม่?

ตัวอย่าง (ต่อ)

```
#include <stdio.h>
void main ( ) {
        int    digits = 0, number;
        char  ans;

        do {
          printf("Enter any integer number : ");
          scanf("%d", &number);
          while (number > 0) {
                number /= 10;
                digits += 1;
          }
          printf("No. of digits of your input is %d\n", digits);
          printf("Do you want to continue (y/Y)? ");
          scanf("%c", &ans);
        } while ( scanf("%[yY]", &ans) );
}
```

- คำสั่งที่ทำให้เกิดการทำซ้ำที่ทราบจำนวนรอบ
- for (initialization;condition;update)

  statement; หรือ {statements}




```
        :
int    digits, number;
        :
for(digits = 0; number > 0; digits += 1)
        number /= 10;

for (digits=0; number>0; number /= 10, digits+=1);

digits = 0;
for (; number > 0; ) {
   number /= 10;
   digits += 1;
}
```


34


```
/* A nested for-loop to print a pyramid of asterisks */
for (row = 1; row<=height; row++) {
  /* Draw space on left of pyramid  */
    for (col=1; col<=height; col++)
        putchar(' ');
  /* Draw body of pyramid */
    for (col = 1; col<=row; col++)
        putchar('*');
   putchar('\n');
}
```


35


```
int      i,  n = 100;

for (i=1; n != 1; i += 1) {
  n = (n % 2 == 0) ? n / 2  :  3 * n + 1;
  if (i % 6 == 0)
     printf("\n");
  printf("%d  ", n);
}
printf("No. of hailstones generated : %d\n", i);
```


36

break statement

- ผลของคำสั่งทำให้*หยุด* ลำดับการทำงานตามขั้นตอนของโครงสร้างที่กำลังดำเนินการอยู่ ณ ขณะนั้น แล้วไปทำยังคำสั่งโครงสร้างที่อยู่ถัดไป
- หากคำสั่งนี้อยู่ภายในโครงสร้างแบบทำซ้ำ (Repetition or Loop) ผลของคำสั่งคือ กระโดดออกจากลูป ไปทำยังคำสั่งถัดจากโครงสร้างลูป
- ใช้กับ คำสั่งลูป และ switch

ตัวอย่าง

```
double  x;
:
while (1) {  /* infinite loop*/
  scanf("%lf", &x);
  if (x < 0.0)
    break;          /* exit loop if x is negative */
  printf("%f\n", sqrt(x));
}
/*break jumps to here */
printf("Loop is terminated by negative input\a\a");
```


continue statement

- ผลของคำสั่งทำให้*หยุด* ลำดับการทำงานตามขั้นตอนของโครงสร้างที่กำลังดำเนินการอยู่ ณ ขณะนั้น แล้วกลับไปทำยังตอนต้นของลูป
- ใช้กับ คำสั่งลูป
  - for
  - do-while
  - while

ตัวอย่าง

```
double  x;
:
while (1) {
  scanf("%lf", &x);
  if (x < 0.0)
    continue;          /* disregard a negative value*/
  printf("%f\n", sqrt(x));
/*continue transfers control to begin next iteration */
}
printf("Loop is terminated by negative input\a\a");
```

switch statement

- เป็นคำสั่งเพื่อเลือกการทำงานตามค่าของนิพจน์หนึ่ง ในกรณีที่ทางเลือกมากกว่า 2 ทาง
- รูปแบบ

```
switch (expression ) {
  case integer-const-1 : statement-1;
                                    break;
  case integer-const-2 : statement-2;
                                    break;
  case integer-const-n : statement-n;
                                    break;
   :                                  :
  default                : statement;
}
```


ตัวอย่าง

```
:
char    choice;
                    :
scanf("%c", &choice);
switch (choice) {
      case  '1'  : printf("You get 1\n");
      case  '2'  : printf("You get 2\n");
                   break;
      default  : printf("You get neither 1 nor 2\n");
}
```


ตัวอย่าง

```
#include <stdio.h>
void main() {
        int   n;
        while (scanf("%[0123]", &n)) {
        switch (n -= '0') {
            case 0 : printf("I get 0.\n");
                     break; /*jump to statement after block of switch*/
            case 1 : printf("I get 1.\n");
                     continue;/*return to start of loop to get new value*/
            default : printf("I get more than 1.\n");
                     if (n == 2) {
                       pritnf("I get 2.\n"); break;
                     }
                     else {
                        pritnf("I get 3.\n"); fflush(stdin); continue;
                     }
                     printf("Statement after if…\n");
        } /*end of switch*/
        printf("Statement after switch…\n");  fflush(stdin);
    } /*end of while*/
  printf("It\'sthe last statement.\a\a");
} /* end of main */
```


ตัวอย่าง

- จงเขียนโปรแกรมเพื่อจำลองการทำงานของเครื่องคิดเลข ที่สามารถ บวก ลบ คูณ และหาร จนกว่าผู้ใช้ไม่ต้องการทำงานต่อ
- โดยมีรูปแบบการแสดงผลดังนี้

operand-1 operator operand-2 = output

operand-1, operator และ *operand-2* ได้จากการป้อนเข้า

ส่วน *output* จะเป็นส่วนที่โปรแกรม ต้องคำนวณและแสดงผล

- เช่น สมมติให้

83 <Enter>

\+ <Enter>

0.5 <Enter>

83 + 0.5 = 83.5

ตัวอย่าง(ต่อ)

- วิเคราะห์
  - **ผลลัพธ์ที่ต้องการ** ผลการคำนวณตามระบุ (result)
  - **ข้อมูลนำเข้า** เลขจำนวนจริง 2 จำนวน (operand1 และ operand2)
    และ ตัวดำเนินการ (operator)
    ทำงานต่อหรือไม่ (ans)
  - **วิธีประมวล**
    - รับข้อมูล ได้แก่ operand1, operator และ operand2
    - เลือกดำเนินการคำนวณที่สอดคล้องกับ ตัวดำเนินการที่รับเข้า
    - แสดงผล
    - ถามความต้องการทำงานต่อหรือไม่?

ตัวอย่าง

```
#include <stdio.h>
void main() {
        float  operand1, operand2, result;
       char  operator, ans;

      printf("*****Calculator simulation*****\n");
      do {
          fflush(stdin);  scanf("%f", &operand1);
          fflush(stdin);  scanf("%c", &operator);
           fflush(stdin);  scanf("%f", &operand2);
           switch (operator) {
             case '+' : result = operand1 + operand2; break;
              case '-'  : result = operand1 - operand2; break;
             case '*'  : result = operand1 * operand2; break;
             case '/'  : result = operand1 / operand2; break;
             default   : printf("Invalid operator"); result = 0.0;
           } /*end of switch*/
           printf("%f %c %f = %f\n", operand1, operator, operand2, result);
           printf("Do you want to continue or not(Y/N)? ");
           fflush(stdin);  scanf("%c", &ans);
         } while (ans == 'Y' || ans == 'y');
} /* end of main */
```


Special Operators

ตัวดำเนินการพิเศษ

Increment and decrement operators

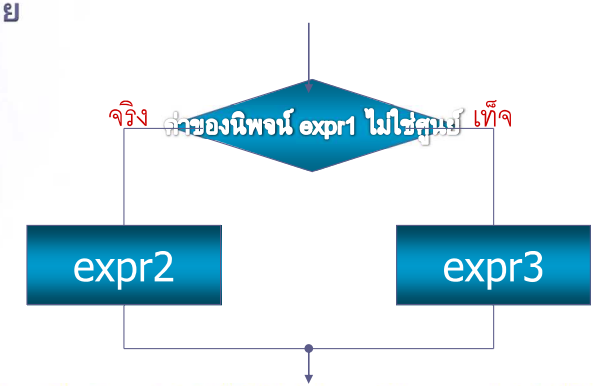
Conditional Operator

Conditional operators

- รูปแบบ

  expr1 ? expr2 : expr3;

- ความหมาย

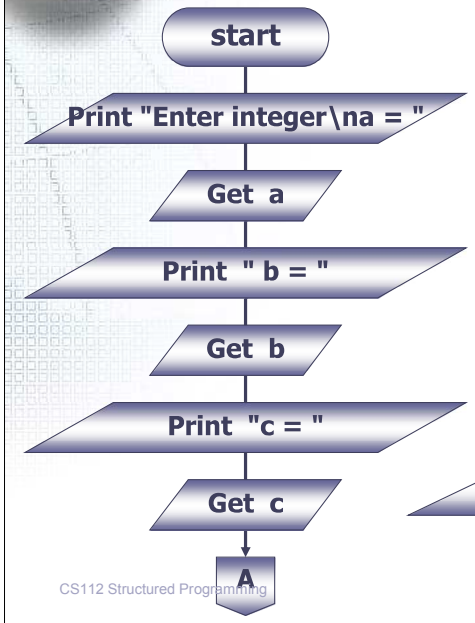


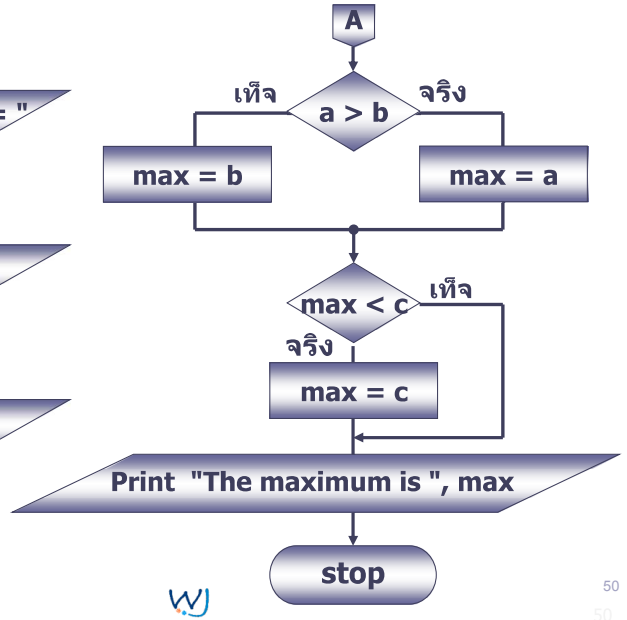

```
scanf("%[01]", &i) ? printf("Valid input") : printf("Invalid input") ;
```

ตัวอย่าง

- จงเขียนโปรแกรมภาษาซี เพื่อหาค่ามากที่สุดของเลขจำนวนเต็ม 3 จำนวน
- วิเคราะห์
  - ผลลัพธ์ ค่าที่มากที่สุด (max)
  - ข้อมูลนำเข้า เลขจำนวนเต็ม 3 จำนวน (a, b, c)
  - วิธีประมวล
    - รับข้อมูล
    - เปรียบเทียบหาค่าที่มากที่สุด
    - แสดงผล

ตัวอย่าง (ต่อ)

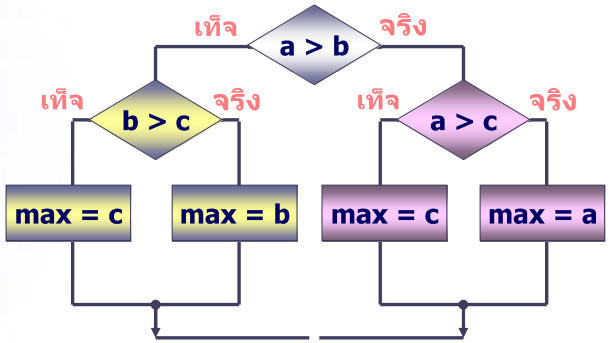




ตัวอย่าง (ต่อ)

```
#include <stdio.h>
void main( ) {
        int   a,   b,   c,   max;
        printf("Enter integer \na = ");
        scanf("%d", &a);
        printf("b = ");
        scanf("%d", &b);
        printf("c = ");
        scanf("%d", &c);
        max =  (a>b) ? a : b;
        (max < c) ? max=c : ;
        printf("The maximum is %d", max);
}
```


ตัวอย่าง (ต่อ)



max = (a>b) ? (a> c ? a : c) : (b> c ? b : c);

Increment & decrement operators

- ใช้ได้กับตัวแปรที่มีชนิดเป็น Integral types เท่านั้น
- การเปลี่ยนแปลงค่าของตัวแปรนั้นทีละ 1 หน่วย
- ++ (Increment) เพิ่มค่าของตัวแปรขึ้น 1 หน่วย
  - n = n + 1;
  - n += 1;
  - n++;
  - ++n;
- -- (Decrement) ลดค่าของตัวแปรลง 1 หน่วย
  - n = n - 1;
  - n -= 1;
  - n--;
  - --n;
- Postfix operator ตัวดำเนินการที่วางไว้หลังตัวถูกดำเนินการ
- Prefix operator ตัวดำเนินการที่วางไว้หน้าตัวถูกดำเนินการ

ตัวอย่าง

```
/* Demonstrate prefix and postfix increment operator */
#include  <stdio.h>
void main() {
  int     x ,   post = 1,  pre = 1;
  x = post++;
  printf("x = %d\tpost = %d\n", x, post);
  x = ++pre;
  printf ("x = %d\tpre = %d\n", x, pre);
}
```

ผลลัพธ์จากการ run โปรแกรมนี้คือ

```
x = 1        post = 2
x = 2        pre  = 2
```


ลำดับความสำคัญของตัวดำเนินการ

ตัวดำเนินการ	ลำดับดำเนินการ
( ) *++(postfix)* *--(postix)*	ซ้ายไปขวา
+(unary) -(unary) ! sizeof *++(prefix)* *--(prefix)*	ขวาไปซ้าย
* / %	ซ้ายไปขวา
+ -	ซ้ายไปขวา
< <= > >=	ซ้ายไปขวา
== !=	ซ้ายไปขวา
&&	ซ้ายไปขวา
||	ซ้ายไปขวา
= += -= *= /= %=	ขวาไปซ้าย

Arithmetic operators

การประกาศตัวแปรและกำหนดค่าเริ่มต้น

```
int    a=1,      b=2,      c=3,      d = 4;
```

นิพจน์	นิพจน์เทียบเท่า	ค่าของนิพจน์
++a * b – c--	((++a) * b) – (c--)	1
7 - -b * d++	7 – ((-b) * (d++))	15

ตัวอย่าง

```
        :
int    digits, number;
        :
for(digits = 0; number > 0; digits += 1)
        number /= 10;
```

```
for (digits=0; number>0; number /= 10, digits++);

for (digits=0; number>0; digits++)
    number /= 10;
for (digits=0; number>0; number /= 10, ++digits);
```


แบบฝึกคิด

จงเขียนโปรแกรม เพื่อเป็นเล่นเกมทายใจระหว่าง ADA กับ Richie (ผู้เล่น 2 คน) โดยผู้เล่นทั้งสองจะมองไม่เห็นว่าอีกฝ่ายป้อนตัวเลขอะไร โดยมีข้อกำหนดดังต่อไปนี้

2.1 เลขที่ป้อน เป็นเลขจำนวนเต็มไม่มีเครื่องหมาย มีความยาวไม่เกิน 4 หลัก

2.2 ใครเริ่มก่อนก็ได้ แต่ละครั้งที่ป้อนตัวเลขทายใจ ต้องขึ้นบรรทัดใหม่

2.3 แต่ละครั้งที่ทาย

หากทายถูกให้แสดงข้อความ "Congratulation, you guess right." พร้อมเสียงบี๊บยาวๆ

หากทายผิดให้บอก "Sorry, try again..." แล้วรับการทายใหม่

2.4 ลองทายได้ไม่เกิน 5 ครั้ง

2.5 จบเกมแต่ละครั้ง ให้ถามความสมัครใจว่าจะเล่นต่อหรือไม่

บทส่งท้าย

จงเขียนชุดคำสั่งภาษาซีเพื่อแสดงเลขในลักษณะตามที่แสดงเป็นตัวอย่างข้างล่างนี้ ซึ่งขึ้นกับค่า n ที่ได้รับ