THIS LECTURE

- **Introduction to Control Structures** โครงสร้างโปรแกรม
  - Sequential Structure การทำงานแบบตามลำดับ
  - Decision Structure การตัดสินใจทำตามเงื่อนไข
    - Statement and Logical Expression
    - If Statement
    - If-Else Statement
    - Cascaded if statement and Switch (ทบทวน)
    - Nested If
  - Loop Structure การทำซ้ำ
  - Case Structure การทำงานตามกรณี

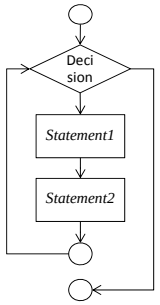
THIS LECTURE

- **Introduction to Control Structures** โครงสร้างโปรแกรม
  - Loop Structure การทำซ้ำ
    - โครงสร้างการวนลูป และ ลักษณะทั่วไปของการแก้ปัญหาด้วยการทำซ้ำ
    - While statement
    - ลักษณะการควบคุมลูป
    - การหยุดวนรอบ

LOOP STRUCTURE โครงสร้างการวนลูป

Loop Structure การทำซ้ำโดยจำนวนรอบของการทำซ้ำถูกกำหนดโดยใช้เงื่อนไข

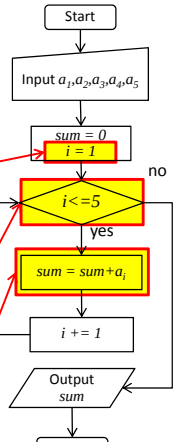
มีประโยชน์มากในกรณีที่ต้องทำงานหนึ่งซ้ำ ๆ กันหลาย ๆ ครั้ง



Loop Structure

ส่วนประกอบสำคัญ

1. การกำหนดค่าเริ่มต้นให้กับตัวแปรที่ควบคุมจำนวนการทำซ้ำ
2. เงื่อนไขที่ต้องการตรวจสอบ
3. การปรับค่าตัวแปรที่ใช้ในกำควมคุมการทำซ้ำ

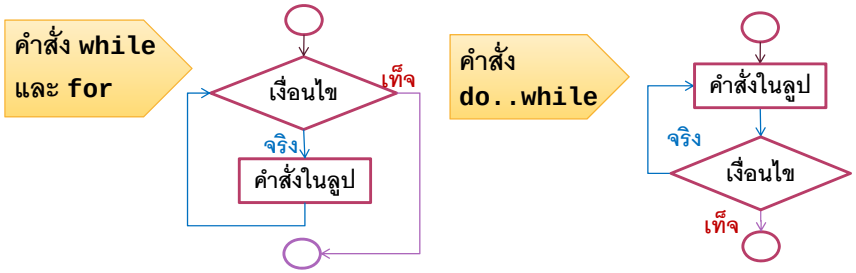


คำนวณผลรวมของเลขจำนวนเต็ม 5จำนวน

รูปแบบของโครงสร้างตรรกะเพื่อการวนรอบ

โครงสร้างตรรกะเพื่อการวนรอบ (Loop logic structure)

- แบบตรวจเงื่อนไขก่อนเข้าสู่การวนรอบ (คำสั่ง `while`)
- แบบนับรอบเพิ่มอัตโนมัติ (คำสั่ง `for`)
- แบบทำหนึ่งรอบแล้ววนกลับ (คำสั่ง `do...while`, repeat..until)



ลักษณะทั่วไปของการแก้ปัญหาด้วยการทำซ้ำ

คำสั่งในลูป คือคำสั่งที่มีการทำซ้ำหลายครั้งนั้น ใช้เพื่อ

- **เพื่อสะสมค่า (Accumulating)** หรือ หาผลรวมของจำนวน (Summation)
  - ลักษณะคำสั่ง จะเป็น **`sum = sum + newValue;`**
  - ต้องตั้งค่าเริ่มต้น**`sum`**ให้เป็น **`0`** ก่อน
- **เพื่อนับค่า (Counting)**
  แบบนับขึ้น (Incrementing) หรือ แบบนับลง (Decrementing)
  - ลักษณะคำสั่งเป็นนับขึ้น **`noOfStudents = noOfStudents + 1;`**
    - ตัวแปรที่ใช้นับค่า เช่น **`noOfStudents`** จะต้องตั้งค่าเริ่มต้นก่อน
    - ส่วนใหญ่จะตั้งตัวนับค่าเริ่มต้นเป็น **`0`**
  - ลักษณะคำสั่งแบบเลื่อนลง เช่น **`totalDone = totalDone - 1;`**
    - ส่วนใหญ่จะตั้งตัวนับลง ให้เป็นจำนวนครั้งที่ต้องการทำซ้ำ

WHILE STATEMENT

In Writing

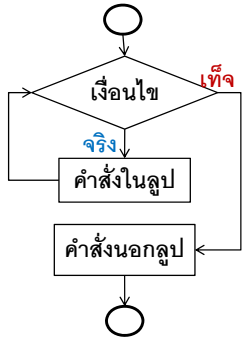
while (นิพจน์เปรียบเทียบเงื่อนไข **)**
do คำสั่งในลูปที่ทำเมื่อผลการเปรียบเทียบเป็นจริง

In C

```
while (expression )
{
   statement;
}
```

หลักการทำงานของ `while` คือ

- ตรวจสอบเงื่อไขก่อนการทำงานทุกครั้ง
  - หาก**เงื่อนไขเป็นจริง** จะเข้าไปทำงานในblock การทำงานของ while loop
  - แต่หาก**เงื่อนไขเป็นเท็จ** จะไม่เข้าสู่การทำงานของ while loop
    - จะไปทำงานคำสั่งถัดไปที่อยู่นอก while loop ทันที



ตัวอย่าง: หาผลรวมของหลายจำนวน

Find sum and average of N numbers.
ข้อมูล
sum, float (output)
count, int (iterator)
totalNum, nextNum, float (input)
วิธีทำ
sum = 0
count = 0
input totalNumbers

while (count < totalNum)
do {
** input nextNum**
** sum = sum + nextNum**
** count = count + 1**
}

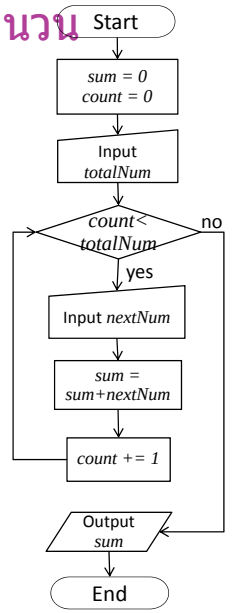
output "Sum was" sum

ส่วนประกอบสำคัญสำหรับควบคุมการทำซ้ำ

กำหนดค่าเริ่มต้น

เงื่อนไขที่ต้องการตรวจสอบ

การปรับค่าตัวแปรที่ควบคุม



คำนวณผลรวมของเลขที่รับเข้ามา

Cited from slide CS102 Week 5 by wdc

Example: addnum.c

Find sum and average of N numbers.
ข้อมูล
sum, float (output)
count, int (iterator)
totalNum, nextNum, float (input)
วิธีทำ
sum = 0
count = 0
input totalNum

while (count < totalNum)
do {
** input nextNum**
** sum = sum + nextNum**
** count = count + 1**
}

output "Sum was" sum

```
#include <stdio.h>
int main()
{




  return 0;
}
```

Cited from slide CS102 Week 5 by wdc

COMMON MISTAKES IN **WHILE**

```
while (num < minimum)
  scanf("%d", &num);
  printf("Number must be greater than %d.\n",
  minimum);
  printf("Please try again.\n");
```

ทำซ้ำแค่บรรทัดนี้

```
while (num < minimum)
{
  scanf("%d", &num);
  printf("Number must be greater than %d.\n",
  minimum);
  printf("Please try again.\n");
}
```

COMMON MISTAKES IN **WHILE**

ทำให้การทำงานไม่สิ้นสุด

```
while (num < minimum);
{
  scanf("%d", &num);
  printf("Number must be greater
  than %d.\n", minimum);
  printf("Please try again.\n");
}
```

Infinite Loop

THIS LECTURE

- **Introduction to Control Structures** โครงสร้างโปรแกรม
  - Loop Structure การทำซ้ำ
    - โครงสร้างการวนลูป และ ลักษณะทั่วไปของการแก้ปัญหาด้วยการทำซ้ำ
    - While statement
    - ลักษณะการควบคุมลูป
    - การหยุดวนรอบ

ลักษณะการควบคุมลูป

เรามีแนวทางใช้ลูปเพื่อควบคุมการจบลูป อยู่ 2 วิธีคือ

1. **เมื่อทราบจำนวนรอบ**
   - ควบคุมด้วยจำนวนรอบ (Counter-Controlled Loop)
2. **เมื่อไม่ทราบจำนวนรอบ**
   - ควบคุมด้วยข้อมูลปิด (sentinel-controlled)
   - ใช้ลูปที่ควบคุมด้วย flag (flag-controlled loops)

การควบคุมด้วยจำนวนรอบ (COUNTER-CONTROLLED LOOP)โดยใช้ WHILE

การควบคุมด้วยจำนวนรอบ:

- ต้องกำหนดตัวแปรและค่าเริ่มต้นสำหรับนับจำนวนครั้งในการทำซ้ำ
- ต้องมีการกำหนดการเพิ่มค่า หรือ ลดลงของค่าในตัวแปร
- เงื่อนไขในการทำซ้ำถูกกำหนดโดยใช้ตัวแปรดังกล่าว

ตัวอย่าง : โปรแกรมแสดงค่าของตัวแปรสำหรับนับรอบการทำซ้ำ

คำตอบ

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

```
#include <stdio.h>
int main ()
{
    /* counter */
    int a = 10;        // ค่าเริ่มต้นสำหรับนับ
    /* while loop execution */
    while( a < 20 )    // เงื่อนไข
    {
      printf("value of a: %d\n", a);
      a++;             // เพิ่มค่าของตัวนับ
    }
  return 0;
}
```

counterControl.c

การควบคุมด้วยจำนวนรอบ (COUNTER-CONTROLLED LOOP)โดยใช้ WHILE

```
#include <stdio.h>
int main ()
{
    /* counter */
    int a = 10;
    /* while loop execution */
    while( a < 20 )
    {
      printf("value of a: %d\n", a);
      a++;
    }
  return 0;
}
```

จงแก้โค้ดเพื่อให้ได้ผลลัพธ์ดังต่อไปนี้

```
value of a: 20
value of a: 19
value of a: 18
value of a: 17
value of a: 16
value of a: 15
value of a: 14
value of a: 13
value of a: 12
value of a: 11
value of a: 10
```


ลักษณะการควบคุมลูป

เรามีแนวทางใช้ลูปเพื่อควบคุมการจบลูป อยู่ 2 วิธีคือ

1. เมื่อทราบจำนวนรอบ
   - ควบคุมด้วยจำนวนรอบ (Counter-Controlled Loop)
2. **เมื่อไม่ทราบจำนวนรอบ**
   - ควบคุมด้วยข้อมูลปิด (sentinel-controlled)
   - ใช้ลูปที่ควบคุมด้วย flag (flag-controlled loops)

ควบคุมด้วยข้อมูลปิด (sentinel-controlled) โดยใช้ while

การควบคุมด้วยข้อมูลปิด:

- ค่าของตัวแปรที่ควบคุมลูปได้รับภายในลูป
- ลูปจะทำงานจนกระทั่งเงื่อนไขที่กำหนดไว้เป็นเท็จ

ตัวอย่าง : รับตัวเลขจากผู้ใช้และแสดงตัวเลขนั้น หากผู้ใช้ใส่ค่า -1 ให้แสดงตัวเลข -1 และหยุดการทำงาน

```
#include <stdio.h>
int main ()
{
    /* initialize input value */
    int value = 0;
    /* while loop execution */
    while( value != -1)
    {
      printf("Enter an integer value:");
      scanf("%d",&value);
      printf("value entered : %d\n", value);
}
  return 0;
}
```

sentinelControl.c

ควบคุมด้วย flag (flag-controlled loops)

Flag เป็นตัวแปรชนิด boolean โดยมีค่าเป็นจริง หรือ เท็จ

- ในภาษาซีไม่มีชนิด boolean ดังนั้น จึงต้องใช้ตัวแปรชนิด **`int`**, **`char`**, หรือ **`unsigned char`** เพื่อเก็บค่า boolean
- ลูปจะทำงานซ้ำจนกระทั่งค่าของ flag ในเงื่อนไขเป็นเท็จ

```
#include <stdio.h>

int main()
{
   int flag = 1;        // ค่าเริ่มต้นของflag
   int value;
   while (flag)         // เงื่อนไข
   {
      scanf("%d",&value);
      if (value != -1)
      {
         printf("Input: %d\n", value);
      }
      else
      {
         flag = 0;      // เปลี่ยนค่าflag
      }
   }
   printf("Loop ends here\n");
   return 0;
}
```

flagControl.c

ตัวอย่าง :

เขียนโปรแกรมที่รับเลขจำนวนเต็มจากผู้ใช้ หากผู้ใช้ใส่ค่า -999 ให้หยุดการทำงาน และแสดงผลรวมและค่าเฉลี่ย

THIS LECTURE

- **Introduction to Control Structures โครงสร้างโปรแกรม**
  - Loop Structure การทำซ้ำ
    - โครงสร้างการวนลูป และ ลักษณะทั่วไปของการแก้ปัญหาด้วยการทำซ้ำ
    - While statement
    - ลักษณะการวนลูป
    - การหยุดวนรอบ

การหยุดวนรอบ

หยุดอย่างปกติ

- คือการวนรอบหยุดเมื่อเงื่อนไขให้วนรอบ (loop condition) เป็นเท็จ (false) นั่นคือ expression ที่ควบคุมลูปให้ค่าเป็น 0

หยุดกระทันหัน ด้วยคำสั่งในโปรแกรม

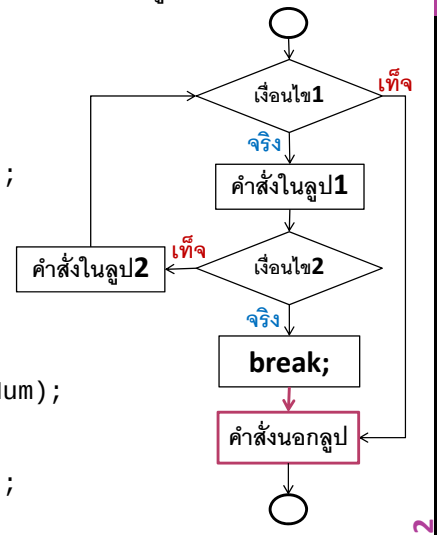
- คำสั่ง break บังคับออกจากลูปเมื่อทำคำสั่ง break เสร็จ
- คำสั่ง exit บังคับให้ออกจากโปรแกรม
- คำสั่ง return เป็นการบังคับให้จบฟังก์ชันและส่งผลลัพธ์กลับทันที

หยุดเมื่อโปรแกรมทำงานผิดพลาด หรือผู้ใช้กดปุ่ม Ctrl-C (cancel), หรือ Ctrl-Break บนคีย์บอร์ด

THE **BREAK** STATEMENT

คำสั่ง **break** บังคับโปรแกรมออกจากลูป ถึงแม้ว่าเงื่อนไขของลูปยังไม่เป็นเท็จก็ตาม

```
int main()
{
  float nextNum;
  while (1)
  {
    scanf("%f", &nextNum);
    if (nextNum==0.0)
    {
      break;
    }
    else
    {
      printf("%f\n",nextNum);
    }
  }
  printf("Out of loop\n");
  return 0;
}
```

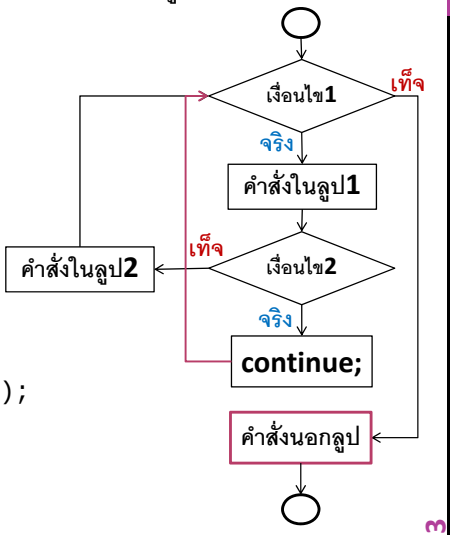


breakControl.c

THE **CONTINUE** STATEMENT

คำสั่ง **break** บังคับโปรแกรมออกจากลูป ถึงแม้ว่าเงื่อนไขของลูปยังไม่เป็นเท็จก็ตาม

```
int main()
{
  int x = 0;
  while (x<10)
  {
    x = x+1;
    if (x==5)
    {
      continue;
    }
    printf("%d ",x);
  }
  printf("Out of loop\n");
  return 0;
}
```



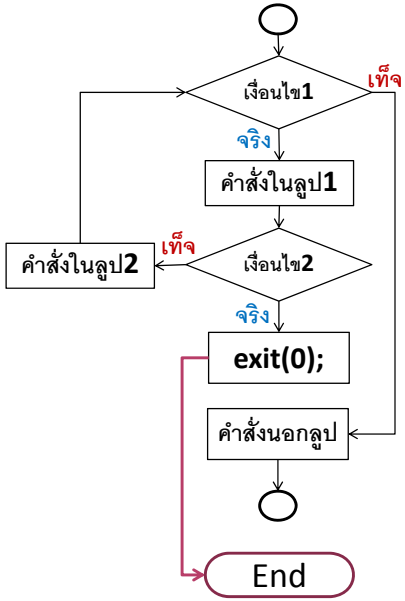
continueControl.c

THE **EXIT** STATEMENT

คำสั่ง **exit** บังคับให้ออกจากโปรแกรม

ต้องใส่ **#include <stdlib.h>** ด้วย

```
int main()
{
  float nextNum;
  while (1)
  {
    scanf("%f", &nextNum);
    if (nextNum==0.0)
    {
      exit(0);
    }
    else
    {
      printf("%f\n",nextNum);
    }
  }
  printf("Out of loop\n");
  return 0;
}
```

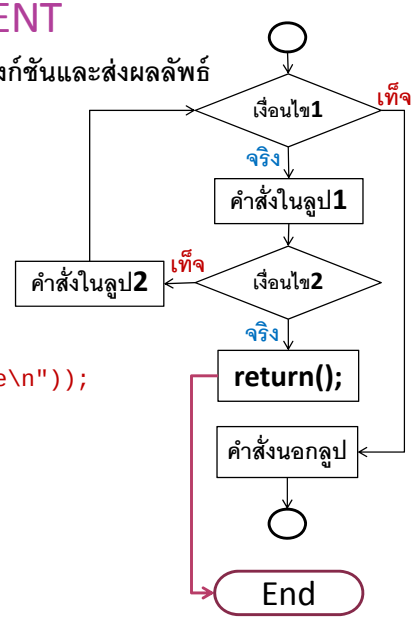


exitControl.c

THE **RETURN** STATEMENT

คำสั่ง **return** เป็นการบังคับให้จบฟังก์ชันและส่งผลลัพธ์กลับทันที

```
int main()
{
  float nextNum;
  while (1)
  {
    scanf("%f", &nextNum);
    if (nextNum==0.0)
    {
      return(printf("Exit Here\n"));
    }
    else
    {
      printf("%f\n",nextNum);
    }
  }
  printf("Out of loop\n");
  return 0;
}
```



returnControl.c

CONCLUSION

- ทบทวนโครงสร้าง **Cascaded If Statement** และ **Switch.**
  - ในบางปัญหาสามารถใช้แทนกันได้
- โครงสร้างการทำซ้ำ **Loop Structure**
  - While Statement
  - ลักษณะการควบคุมจำนวนครั้งของการทำซ้ำ
    - รู้จำนวนรอบ
    - ทำซ้ำจนกระทั่งเงื่อนไขการทำซ้ำเป็นเท็จ
  - การหยุดวนรอบ
    - Break, Continue, Exit, และ Return