Computer Programming I:
การเขียนโปรแกรมคอมพิวเตอร์ I

ระบบแถวลำดับ (Array)

อ.ดร.ปัญญนัท อ้นพงษ์

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

aonpong_p@su.ac.th

Outline

- **แถวลำดับ**
- การประยุกต์ใช้
- แถวลำดับหลายมิติ
- การประยุกต์ใช้แถวลำดับหลายมิติ

แถวลำดับ

- สมมติว่าอยากเขียนโปรแกรมรับค่าอายุของคน 5 คน แล้วพิมพ์ออกทางจอภาพแบบเอาอายุของคนสุดท้ายขึ้นก่อน จะเขียนโปรแกรมอย่างไร
  - ทางเลือกแรก เราอาจจะใช้ loop เพื่อรับค่า 5 ครั้ง แล้วพิมพ์ออกทางจอภาพ
    - ลองคิดให้ดีว่าทำได้จริงมั้ย
  - ทางเลือกที่สอง ประกาศตัวแปร 5 ตัว แล้วพิมพ์ตัวแปรทั้งหมดออกทางจอภาพแบบย้อนกลับ (เอาตัวที่รับตัวสุดท้ายขึ้นก่อน)

  - Loop ใช้ไม่ได้ ดังนั้นจึงต้องเลือกทางเลือกที่สอง

แถวลำดับ

```
void main() {
    int a1, a2, a3, a4, a5;
    scanf("%d %d %d %d %d", &a1, &a2, &a3, &a4, &a5);
    printf("%d %d %d %d %d", a5, a4, a3, a2, a1);
}
```

แถวลำดับ

- สมมติว่าอยากเขียนโปรแกรมรับค่าอายุของคน 100 คน แล้วพิมพ์ออกทางจอภาพแบบย้อนกลับ จะเขียนโปรแกรมอย่างไร
  - ทางเลือกแรก เราอาจจะใช้ loop เพื่อรับค่า 100 ครั้ง แล้วพิมพ์ออกทางจอภาพ
    - ตอนนี้เรารู้แล้วว่าทำไม่ได้
  - ทางเลือกที่สอง ประกาศตัวแปร 100 ตัว แล้วพิมพ์ตัวแปรทั้งหมดออกทางจอภาพ

  - Loop เป็นทางเลือกที่เป็นไปไม่ได้ จึงต้องเลือกทางเลือกที่ 2...
  - ...ก็แย่แล้ว ใครจะไปนั่งประกาศตัวแปร 100 ตัว

แถวลำดับ

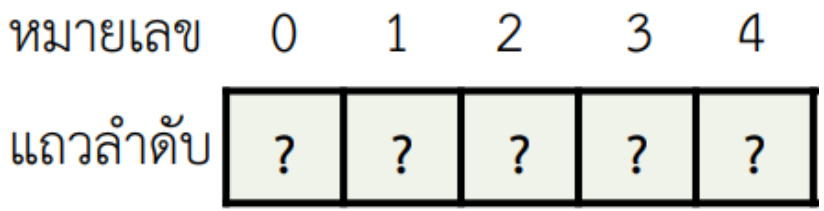
ขอนำเสนอตัวแปรแบบแถวลำดับ โดยจะแบ่งการอธิบายออกเป็น 2 หัวข้อย่อย

- การประกาศอาเรย์
- การอ้างถึงตัวแปรแต่ละตัวในอาเรย์

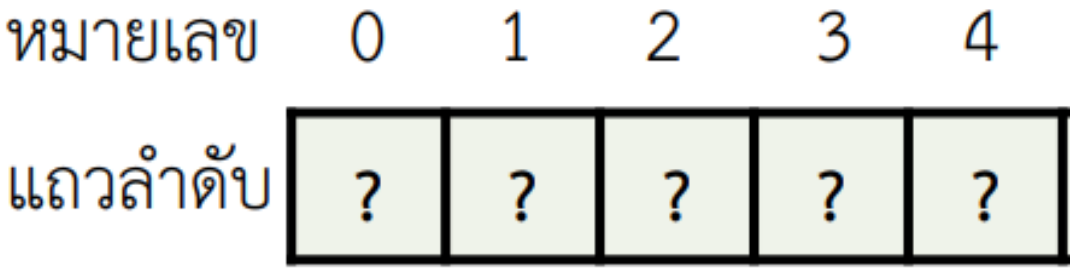
แถวลำดับ

- เพื่อให้อธิบายได้ง่าย เรากลับมาประกาศตัวแปรแค่ 5 ตัวก่อน
- เราสามารถประกาศตัวแปร 5 ตัวแบบนี้ได้

  int a[5]; <-- สิ่งนี้คือตัวแปรแบบอาเรย์
- อาเรย์ เป็นวิธีเก็บข้อมูลประเภทเดียวกันจำนวนมากไว้ด้วยกัน โดยจะเก็บเป็นแถวต่อเนื่องกันไป
- การประกาศตามตัวอย่างข้างบนจะเหมือนมีตัวแปร 5 ตัวเกิดขึ้น แบบนี้



แถวลำดับ



- สังเกตว่า เริ่มที่ 0 จบที่ 4 (ไม่ใช่ลำดับการนับเลขที่คุ้นเคยเท่าไหร่)
- แต่ก็มีช่องรวมกัน 5 ช่อง ตามที่ประกาศว่า int a[5];
- แถวลำดับเก็บข้อมูลได้หลายตัว แต่ทุกตัวต้องเป็นชนิดเดียวกัน เช่นเป็น int เหมือนกันหมด จะให้แถวลำดับเก็บค่า int ปน float ไม่ได้

Array: การประกาศ

- ถ้าไม่ใช้อาเรย์ จะได้ว่า

```
int a1, a2, a3, a4, a5;
```

- ถ้าใช้อาเรย์ จะได้ว่า

```
int a[5];
```

จะเห็นว่าการประกาศง่ายกว่ากัน ทีนี้ลองจินตนาการว่าถ้าอยากประกาศตัวแปร 100 ตัวจะเป็นไปได้มั้ย

Array: การประกาศ

สรุปวิธีการประกาศตัวแปรอาเรย์

- ถ้าหากเราต้องการเก็บเลขจำนวนเต็ม 5 ตัวไว้ด้วยกันภายใต้ชื่อ a เราเขียนว่า int a[5];
- ชนิดข้อมูลต้องนำหน้าชื่อเช่นเดียวกับการประกาศตัวแปรทั่วไป
- เราใช้วงเล็บเหลี่ยมหลังชื่อแถวลำดับ และเราใส่ตัวเลขเข้าไปเพื่อบอกว่าแถวลำดับนี้จะเก็บข้อมูลได้สูงสุดกี่ตัว ในที่นี้คือเก็บได้สูงสุด 5 ตัว
- สรุปความแตกต่างในการสร้างแถวลำดับกับตัวแปรทั่วไปคือ แถวลำดับจะมีวงเล็บเหลี่ยม (square bracket) และจำนวนข้อมูลที่จะรับได้ตามมา
  - แต่ตัวแปรทั่วไปจะมีแค่ชนิดข้อมูลและชื่อ
  - เปรียบเทียบ int a; กับ int a[5]; แบบแรกเป็นตัวแปร int ทั่วไป แต่แบบที่สองคือแถวลำดับที่เก็บ int ได้สูงสุด 5 ตัว

แถวลำดับ

- การประกาศอาเรย์
- การอ้างถึงตัวแปรแต่ละตัวในอาเรย์

Array: การอ้างถึง

ตอนนี้จึงเกิดคำถามว่าเมื่อเราประกาศอาเรย์ไปแล้ว จะเรียกใช้ยากมั้ย

ตอบ ไม่ยาก สมมติว่าเราไม่ได้ใช้อาเรย์ ก็จะเป็นทรงนี้ (ขอนำมาเฉพาะส่วนรับค่าก่อน)

```
void main() {
    int a1, a2, a3, a4, a5;
    scanf("%d", &a1);
    scanf("%d", &a2);
    scanf("%d", &a3);
    scanf("%d", &a4);
    scanf("%d", &a5);
}
```

Array: การอ้างถึง

แต่ถ้าเราใช้อาเรย์ จะเห็นว่าเวลาเรียกใช้ ก็แค่ใส่เครื่องหมาย [ ] พร้อมลำดับไว้ข้างในวงเล็บ [ ] ได้เลย เพียงแต่ลำดับจะเริ่มที่ 0 ไม่ใช่ 1

```c
void main() {
    int a[5];
    scanf("%d", &a[0]);
    scanf("%d", &a[1]);
    scanf("%d", &a[2]);
    scanf("%d", &a[3]);
    scanf("%d", &a[4]);
}
```

```c
void main() {
    int a1, a2, a3, a4, a5;
    scanf("%d", &a1);
    scanf("%d", &a2);
    scanf("%d", &a3);
    scanf("%d", &a4);
    scanf("%d", &a5);
}
```

Array: การอ้างถึง

ตอนนี้จะเห็นว่าตอนประกาศง่ายขึ้นก็จริง แต่ตอนใช้ก็ยังยากอยู่

แต่จริงๆแล้ว อาเรย์มันมีดีกว่านั้น

```
void main() {
    int a[5];
    scanf("%d", &a[0]);
    scanf("%d", &a[1]);
    scanf("%d", &a[2]);
    scanf("%d", &a[3]);
    scanf("%d", &a[4]);
}
```

```
void main() {
    int a1, a2, a3, a4, a5;
    scanf("%d", &a1);
    scanf("%d", &a2);
    scanf("%d", &a3);
    scanf("%d", &a4);
    scanf("%d", &a5);
}
```

Array: การอ้างถึง

ตัวเลขที่อยู่ในเครื่องหมายวงเล็บ [ ] มันไม่จำเป็นต้องเป็นตัวเลขก็ได้

อาจใส่เป็นตัวแปรแทนได้ เช่น

```
int a[5], i=3;
scanf("%d", &a[i]);    //บรรทัดนี้มีค่าเท่ากับ scanf("%d", &a[3]);
```

แล้วมันทำให้ง่ายขึ้นตรงไหน?

Array: การอ้างถึง

เพราะมันสามารถประยุกต์กับ loop ได้

```
void main() {
    int a[5], i;
    for(i=0; i<5; i++){
        scanf("%d", &a[i]);
    }
}
```

```
void main() {
    int a[5];
    scanf("%d", &a[0]);
    scanf("%d", &a[1]);
    scanf("%d", &a[2]);
    scanf("%d", &a[3]);
    scanf("%d", &a[4]);
}
```

Array: การอ้างถึง

เขียนโปรแกรมให้จบ เราต้องการพิมพ์อายุย้อนหลังจากคนสุดท้าย

```
void main() {
    int a[5], i;
    for(i=0; i<5; i++){
        scanf("%d", &a[i]);
    }
    for(i=4; i>=0; i--){
        printf("%d ", a[i]);
    }
}
```

```
void main() {
    int a1, a2, a3, a4, a5;
    scanf("%d", &a1);
    scanf("%d", &a2);
    scanf("%d", &a3);
    scanf("%d", &a4);
    scanf("%d", &a5);
    printf("%d ", a5);
    printf("%d ", a4);
    printf("%d ", a3);
    printf("%d ", a2);
    printf("%d ", a1);
}
```

แถวลำดับ

- ทีนี้กลับมาดูโจทย์ข้อนี้อีกครั้ง
- สมมติว่าอยากเขียนโปรแกรมรับค่าอายุของคน 100 คน แล้วพิมพ์ออกทางจอภาพแบบย้อนกลับ จะเขียนโปรแกรมอย่างไร
  - ประกาศตัวแปร 100 ตัว แล้วพิมพ์ตัวแปรทั้งหมดออกทางจอภาพ
  - ทำได้หรือยัง ถ้าให้แก้โค้ดหน้าที่แล้วต้องแก้ยังไง?

Outline

- แถวลำดับ
- **การประยุกต์ใช้**
- แถวลำดับหลายมิติ
- การประยุกต์ใช้แถวลำดับหลายมิติ

เราควรใช้อาเรย์เมื่อไร

- เมื่อต้องการเก็บข้อมูลที่มีชนิดเดียวกันมาก ๆ
- เมื่อข้อมูลชนิดเดียวกันนั้นมีการนำไปใช้ในลักษณะเดียวกัน
- เมื่อข้อมูลแต่ละตัวอาจถูกอ้างถึงมากกว่าหนึ่งครั้ง เช่นต้องให้โปรแกรมจดจำเลขที่ป้อนเข้ามาทุกตัว
  - (จุดนี้เป็นจุดที่แทบจะบังคับเลือกใช้แถวลำดับ)
- เมื่อเราคิดว่าการเตรียมที่เก็บข้อมูลทั้งหมดไว้ก่อนเป็นเรื่องที่สะดวกกว่า
  - เช่นโจทย์เก่า ๆ บางข้อรับค่าแล้วทิ้งค่าไปเลยก็ได้ แต่ถ้าเราจะเก็บค่าไว้ก็ไม่ได้ทำให้โปรแกรมเสียหายแต่อย่างใด ตราบใดที่หน่วยความจำยังไม่เต็ม
  - (อันนี้เป็นเรื่องของธรรมชาติในการใช้ความคิดของแต่ละบุคคล)
- แน่นอนว่าการประกาศตัวแปรก็ต้องใช้พื้นที่ในหน่วยความจำด้วย การประกาศตัวแปรทีละตัวมักไม่ค่อยมีปัญหาอะไร แต่เมื่อเรารู้จักอาเรย์ นั่นคือเราอาจมีการประกาศตัวแปรเป็นพัน ๆ ตัว ซึ่งทำให้หน่วยความจำเต็มได้ไม่ยาก

ตัวอย่างโจทย์ (1)

การเก็บและเรียกดูข้อมูล

จงเขียนโปรแกรมที่รับค่าเป็นส่วนสูงของนักศึกษาในคลาสซึ่งมีทั้งหมด 15 คน โดยส่วนสูงนี้เป็นจำนวนเต็มมีหน่วยเป็นเซนติเมตร เมื่อใส่ข้อมูลจนครบ 15 คนแล้ว ผู้ใช้จะสามารถเรียกดูส่วนสูงของนักศึกษาคนใดก็ได้ด้วยการอ้างถึงลำดับที่จาก 1 ถึง 15 ซึ่งเรียงตามลำดับการป้อนข้อมูลเข้ามาในตอนแรก

หากผู้ใช้ถามถึงส่วนสูงของนักศึกษาคนแรกก็จะใส่เลข 1 เข้ามา และหากต้องการถามถึงคนที่ 2 ก็จะใส่เลข 2 เข้ามาอย่างนี้เป็นต้น หากผู้ใช้ใส่เลขที่อยู่นอกเหนือจากเลข 1 ถึง 15 โปรแกรมจะพิมพ์คำว่า Good bye และจบการทำงาน

ตัวอย่างโจทย์ (1)

วิเคราะห์ปัญหา

1. ในปัญหานี้ข้อมูลตัวหนึ่งสามารถถูกอ้างอิงได้หลายครั้ง
2. ข้อมูลเชื่อมโยงกับลำดับ ดังนั้นการใช้แถวลำดับมาเก็บข้อมูลจะทำให้การแก้ปัญหาเป็นไปโดยสะดวก
3. เพราะต้องการเก็บข้อมูลนักเรียนไว้ 15 คน จึงควรเตรียมแถวลำดับที่เก็บข้อมูลได้ 15 ตัว ด้วยการวนลูปอ่านค่าส่วนสูงจากผู้ใช้ทั้งหมด 15 รอบ
4. ในการถามถึงข้อมูล ลำดับการนับของผู้ใช้เริ่มจาก 1 แต่ภาษาซีเริ่มจาก 0
5. ผู้ใช้จะถามถึงข้อมูลกี่ครั้งก็ได้ไม่จำกัด จะถามคนเดิมซ้ำก็ได้ โปรแกรมจึงควรวนลูปรับคำถามจากผู้ใช้ไม่จำกัดจำนวนครั้งเช่นกัน
6. เพราะอย่างนี้การใช้ลูปที่มีคำสั่ง break; อยู่ด้วยจึงเป็นวิธีการที่เหมาะสม
7. เราควร break; เมื่อผู้ใช้ถามถึงข้อมูลลำดับที่ 0 หรือติดลบหรือเกิน 15

ตัวอย่างโจทย์ (1)

```c
#include <stdio.h>

void main() {
    int A[15];
    int i;
    for(i = 0; i < 15; ++i) {
        scanf("%d", &A[i]);
    }
    while(1) {
        scanf("%d", &i);
        if(i <= 0 || i > 15)
            break;
        printf("%d\n", A[i-1]);
    }
    printf("Good bye\n");
}
```

อาเรย์

แถวลำดับกับจำนวนข้อมูลที่ไม่แน่นอน

ตัวอย่างที่ให้มาก่อนหน้าเราทราบจำนวนข้อมูลล่วงหน้ามาก่อน

- เราจึงสามารถสร้างแถวลำดับที่เก็บข้อมูลได้แบบพอดี (โจทย์กำหนด 15 ตัว เราจึงประกาศ a[15])
- จะเกิดอะไรขึ้นถ้าเราไม่ทราบจำนวนข้อมูลที่ตายตัวล่วงหน้า
  - ถ้าเรารู้จำนวนข้อมูลสูงสุดที่เป็นไปได้ก่อน เราเตรียมแถวลำดับไว้เผื่อได้
  - เช่นถ้าโจทย์บอกว่ารับค่าไม่เกิน 100 ตัว ก็ประกาศอาเรย์ 100 ช่องไปเลย (แม้จะใช้จริงไม่ถึงก็ตาม)
  - ทำแบบนี้จะทำให้มีบางตำแหน่งในแถวลำดับที่ไม่มีข้อมูลอยู่
- ถ้าผู้ใช้จะเป็นคนระบุจำนวนข้อมูลมาตอนโปรแกรมทำงาน
  - เราอาจใช้แถวลำดับพลวัต (dynamic array) เข้ามาจัดการได้
- ถ้าจำนวนข้อมูลเพิ่มขึ้นได้เรื่อย ๆ ไม่จำกัด เราต้องใช้วิธีที่ซับซ้อนขึ้น

ตัวอย่างโจทย์ (2)

การพิมพ์เลขย้อนลำดับแบบไม่กำหนดจำนวนตัวตายตัว

จงเขียนโปรแกรมที่รับข้อมูลเป็นเลขจำนวนเต็มบวกมา N ค่า โดยที่ N มีค่าไม่เกิน 1000 ตัว โปรแกรมจะหยุดรับค่าจากผู้ใช้เมื่อผู้ใช้ใส่เลขศูนย์หรือติดลบเข้ามา เมื่อหยุดรับค่าแล้ว โปรแกรมจะพิมพ์ตัวเลขทั้งหมดที่ผู้ใช้ใส่เข้ามาย้อนลำดับจากหลังมาหน้า

กำหนดให้ผู้ใช้จะไม่ใส่เลขเกิน 1000 ตัว ดังนั้นโปรแกรมไม่ต้องคอยตรวจว่าผู้ใช้ใส่มาเกิน 1000 หรือไม่ และกำหนดให้ผู้ใช้ใส่เลขบวกอย่างน้อยหนึ่งค่า

วิเคราะห์

1. เราไม่รู้ค่า N ที่ตายตัวล่วงหน้า แต่รู้ว่ายังไง N ก็ไม่เกิน 1,000 ดังนั้นเราประกาศ int A[1000]; ไว้ได้

2. ต้องมีตัวแปรคอยนับค่า N เพื่อติดตามว่าผู้ใช้ใส่เลขบวกเข้ามากี่ตัวกันแน่

ตัวอย่างโจทย์ (2)

การพิมพ์เลขย้อนลำดับแบบไม่กำหนดจำนวนตัวตายตัว

```c
#include <stdio.h>

void main() {
    int A[1000];
    int N = 0;
    int num;
    while(1) {
        scanf("%d", &num);
        if(num <= 0)
            break;
        A[N] = num;
        ++N;
    }
    int i;
    for(i = N-1; i >= 0; --i) {
        printf("%d\n", A[i]);
    }
}
```

ตัวอย่างโจทย์ (3)

เปลี่ยนโจทย์การพิมพ์เลขย้อนลำดับข้อที่แล้วเล็กน้อย

จงเขียนโปรแกรมที่รับข้อมูลเป็นเลขจำนวนเต็มบวกมา N ค่า โดยที่ N มีค่าไม่เกิน 1000 ตัว โปรแกรมจะหยุดรับค่าจากผู้ใช้เมื่อผู้ใช้ใส่เลขศูนย์หรือติดลบเข้ามา เมื่อหยุดรับค่าแล้ว โปรแกรมจะพิมพ์ตัวเลขทั้งหมดที่ผู้ใช้ใส่เข้ามาย้อนลำดับจากหลังมาหน้า

โจทย์กำหนดให้ผู้ใช้จะไม่ใส่เลขเกิน 1000 ตัว ดังนั้นโปรแกรมไม่ต้องคอยตรวจว่าผู้ใช้ใส่มาเกิน 1000 หรือเปล่า และหากผู้ใช้ไม่ได้ใส่เลขบวกเข้ามาเลยให้โปรแกรมพิมพ์ว่า No input และจบการทำงาน

วิเคราะห์ เราทำเหมือนเดิม เพียงแต่ให้ตรวจเพิ่มเติมว่า N ที่ได้เป็นศูนย์หรือเปล่า ถ้าเป็นศูนย์ก็ให้พิมพ์คำว่า No input ออกมาแทน โจทย์ข้อนี้มีวิธีแก้หลากหลาย

ตัวอย่างโจทย์ (3)

เปลี่ยนโจทย์การพิมพ์เลขย้อนลำดับข้อที่แล้วเล็กน้อย

วิธีที่ 1

```
void main() {
    int A[1000];
    int N = 0;
    int num;
    while(1) {
        scanf("%d", &num);
        if(num <= 0)
            break;
        A[N] = num;
        ++N;
    }
    if(N > 0) {
        int i;
        for(i = N-1; i >= 0; --i) {
            printf("%d\n", A[i]);
        }
        }
        else {
            printf("No input\n");
        }
}
```

ตัวอย่างโจทย์ (3)

เปลี่ยนโจทย์การพิมพ์เลขย้อนลำดับข้อที่แล้วเล็กน้อย

วิธีที่ 2 แบบนี้จะดูยากกว่าวิธีแรก

ไม่มี if-else แต่มีแค่ if ตัวเดียว

```
void main() {
    int A[1000];
    int N = 0;
    int num;
    while(1) {
        scanf("%d", &num);
        if(num <= 0)
            break;
        A[N] = num;
        ++N;
    }
    int i;
    for(i = N-1; i >= 0; --i) {
        printf("%d\n", A[i]);
    }
    if(N == 0)
        printf("No input\n");
}
```

ตัวอย่างโจทย์ (4)

เปลี่ยนโจทย์การพิมพ์เลขย้อนลำดับอีกครั้ง

ตัวอย่าง จงเขียนโปรแกรมที่รับข้อมูลเป็นเลขจำนวนเต็มมา N ค่าโดยที่ N มีค่าไม่เกิน 1000 ตัว โปรแกรมจะรับค่า N มาจากผู้ใช้ก่อน จากนั้นจะวนรับค่าจำนวนเต็มจากผู้ใช้จนครบ N จำนวนและหยุดรับค่าหลังจากนั้น เมื่อหยุดรับค่าแล้วโปรแกรมจะพิมพ์ตัวเลขทั้งหมดที่ผู้ใช้ใส่เข้ามาย้อนลำดับจากหลังมาหน้า กำหนดให้ผู้ใช้จะไม่ใส่ค่า N มาเกิน 1000 ดังนั้นโปรแกรมไม่ต้องคอยตรวจว่า N มีค่าเกิน 1000 หรือเปล่า หากผู้ใช้ใส่ค่า N มาเป็นศูนย์หรือน้อยกว่า โปรแกรมจะหยุดทำงานโดยไม่พิมพ์ค่าใด ๆ ออกมา

วิเคราะห์

1. คราวนี้มีการระบุค่า N มาแต่เริ่ม ดังนั้นเราสั่งวนลูป N รอบได้เลย
2. แบบนี้แสดงว่าคำสั่ง break; ไม่เป็นสิ่งจำเป็นอีกต่อไป

ตัวอย่างโจทย์ (4)

เมื่อไม่ต้องติดตามค่า N และคอย break ทำให้โปรแกรมมีความซับซ้อนน้อยลงมาก (ข้อนี้ง่ายกว่าข้อก่อนหน้า)

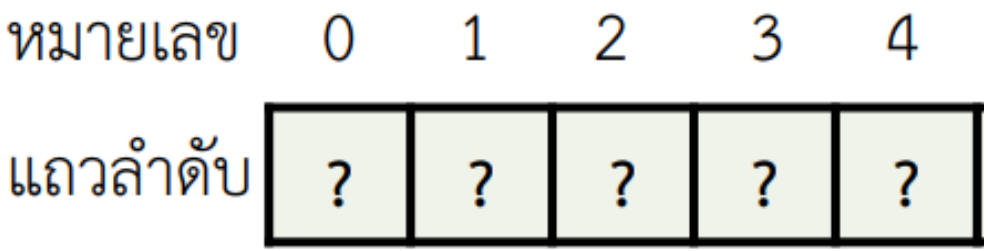
```
void main() {
    int A[1000];
    int N, num;
    scanf("%d", &N);
    int i;
    for(i = 0; i < N; ++i) {
        scanf("%d", &num);
        A[i] = num;
    }
    for(i = N-1; i >= 0; --i) {
        printf("%d\n", A[i]);
    }
}
```

Outline

- แถวลำดับ
- การประยุกต์ใช้
- **แถวลำดับหลายมิติ**
- การประยุกต์ใช้แถวลำดับหลายมิติ

แถวลำดับหลายมิติ

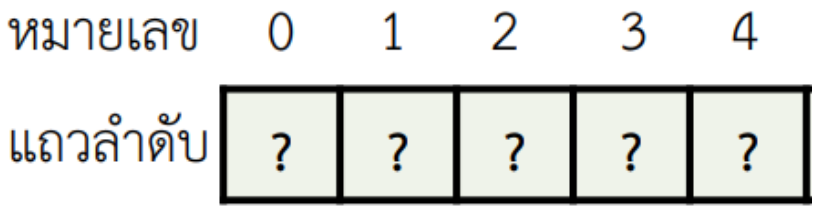
- ยกตัวอย่างแถวลำดับแบบที่เพิ่งได้เรียนมาเป็นพื้นฐานก่อน
  - เมื่อเราประกาศ int a[5]; สิ่งที่เราจะได้คือ ตัวแปร int ที่ต่อกัน 5 ตัว



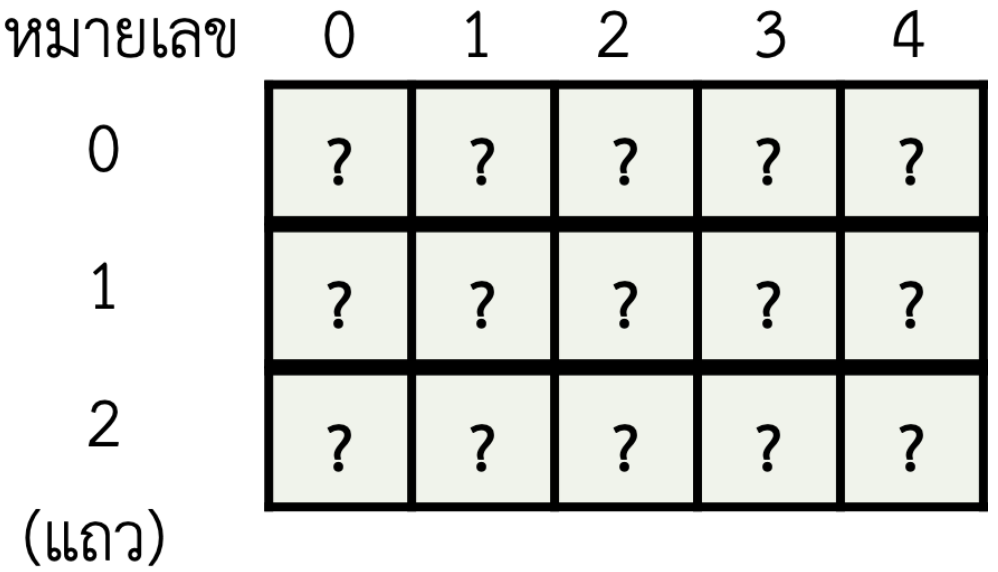
- พูดให้ง่ายก็คือ จับตัวแปร int มาเรียงต่อกันให้ได้จำนวนตัวตามที่กำหนด

แถวลำดับหลายมิติ

- อาเรย์ 2 มิติ ก็คล้ายกัน เพียงแต่แทนที่จะจับ int มาเรียงต่อกัน เราก็จับอาเรย์มาทั้งก้อน แล้วเรียงกันแทน



แบบ 1 มิติ



แบบ 2 มิติ

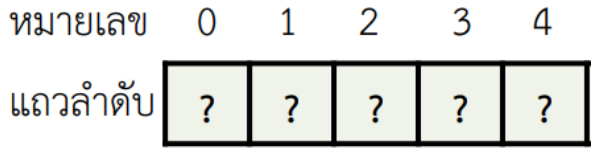
แถวลำดับหลายมิติ

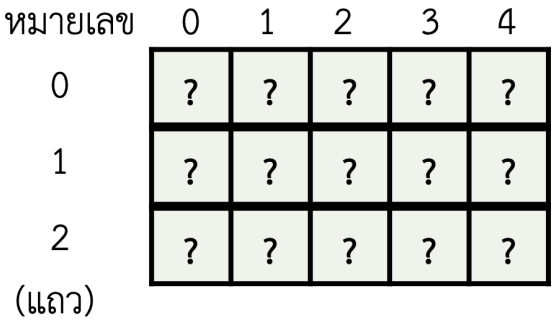
การอธิบายจะแบ่งเป็น 2 ส่วน เหมือนเดิม

- การประกาศ
- การเรียกใช้

แถวลำดับหลายมิติ: การประกาศ

- เปรียบเทียบกับอาเรย์ 1 มิติ





อาเรย์ 1 มิติ	อาเรย์ 2 มิติ
ประกาศโดยการกำหนด ประเภทตัวแปร ชื่อตัวแปร [จำนวนตัวที่ต้องการเก็บ]	ประกาศโดยกำหนด ประเภทตัวแปร ชื่อตัวแปร [row][col];
เช่น int a[10]; float weight[100]; double score[305];	เช่น int a[10][5]; float weight[100][2]; double score[305][7];

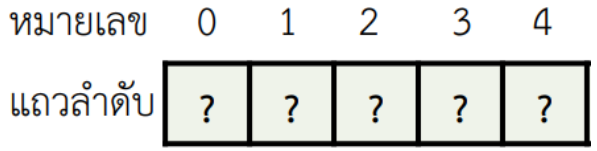
แถวลำดับหลายมิติ

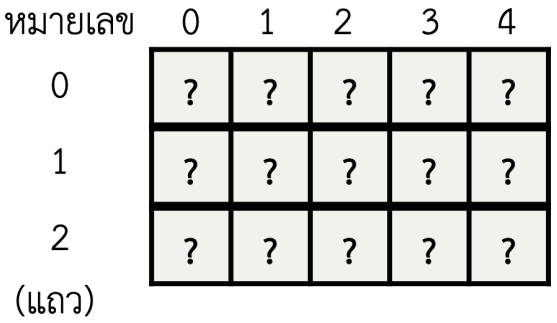
การอธิบายจะแบ่งเป็น 2 ส่วน เหมือนเดิม

- การประกาศ
- การเรียกใช้

แถวลำดับหลายมิติ: การเรียกใช้

- เปรียบเทียบกับอาเรย์ 1 มิติ





อาเรย์ 1 มิติ	อาเรย์ 2 มิติ
เรียกใช้ตัวแปรโดย ชื่อตัวแปร [ตำแหน่ง]	เรียกใช้ตัวแปรโดย ชื่อตัวแปร [ตำแหน่งrow][ตำแหน่งcol]

เช่น
a[2] = 3; //กำหนดตัวแปร a ตำแหน่งที่ 2 เป็น 3
scanf("%f", &weight[0]);
printf("%lf", score[101]);
	เช่น
	a[4][2] = 15;
	scanf("%f", &weight[100][2]);
	printf("%lf", score[0][0]);

Outline

- แถวลำดับ
- การประยุกต์ใช้
- แถวลำดับหลายมิติ
- **การประยุกต์ใช้แถวลำดับหลายมิติ**

ตัวอย่างโจทย์ (5)

โปรแกรมบันทึกคะแนนสอบย่อย

กำหนดให้ชั้นเรียนมีนักศึกษาทั้งหมด 150 คน และมีการสอบย่อยทั้งหมด 5 ครั้ง คะแนนการสอบแต่ละครั้งมีชนิดข้อมูลเป็นเลขทศนิยมแบบ float จงเขียนโปรแกรมที่ให้ผู้ใช้บันทึกข้อมูลคะแนนนักศึกษาเข้าไปทีละคน การบันทึกคะแนนจะบันทึกคะแนนทั้ง 5 ครั้งเข้าไปด้วยกัน แล้วจึงรับคะแนนของนักศึกษาคนถัดไป

ตัวอย่างโจทย์ (5)

โปรแกรมบันทึกคะแนนสอบย่อย

```c
#include <stdio.h>

void main() {
    float S[150][5];
    int row, col;
    for(row = 0; row < 150; ++row) {
        for(col = 0; col < 5; ++col) {
            scanf("%f", &S[row][col]);
        }
    }
}
```

ตัวอย่างโจทย์ (6)

โปรแกรมบันทึกและคำนวณคะแนนสอบย่อย

กำหนดให้ชั้นเรียนมีนักศึกษาทั้งหมด 150 คน และมีการสอบย่อยทั้งหมด 5 ครั้ง คะแนนการสอบแต่ละครั้งมีชนิดข้อมูลเป็นเลขทศนิยมแบบ float จงเขียนโปรแกรมที่ให้ผู้ใช้บันทึกข้อมูลคะแนนนักศึกษาเข้าไปทีละคน การบันทึกคะแนนจะบันทึกคะแนนทั้ง 5 ครั้งเข้าไปด้วยกัน แล้วจึงรับคะแนนของนักศึกษาคนถัดไป

เมื่อได้ข้อมูลมาครบแล้ว โปรแกรมจะพิมพ์ผลรวมคะแนนสอบของแต่ละคนออกมาตามลำดับ (อย่าพิมพ์จนกว่าโปรแกรมจะรับข้อมูลมาจนหมด)

ตัวอย่างโจทย์ (6)

โปรแกรมบันทึกและคำนวณคะแนนสอบย่อย

- นักศึกษาอาจสังเกตว่าโปรแกรมนี้มีการรับค่าจำนวนมากถึง 750 ตัว ทำให้ตรวจสอบการทำงานของโปรแกรมได้ยาก
- ในการพัฒนาโปรแกรมจึงมีเทคนิคเบาๆ

```c
#include <stdio.h>

void main() {
    float S[150][5];
    int row, col;
    for(row = 0; row < 150; ++row) {
        for(col = 0; col < 5; ++col) {
            scanf("%f", &S[row][col]);
        }
    }

    for(row = 0; row < 150; ++row) {
        float sum = 0;
        for(col = 0; col < 5; ++col) {
            sum += S[row][col];
        }
        printf("%.2f\n", sum);
    }
}
```

ตัวอย่างโจทย์ (6)

โปรแกรมบันทึกและคำนวณคะแนนสอบย่อย

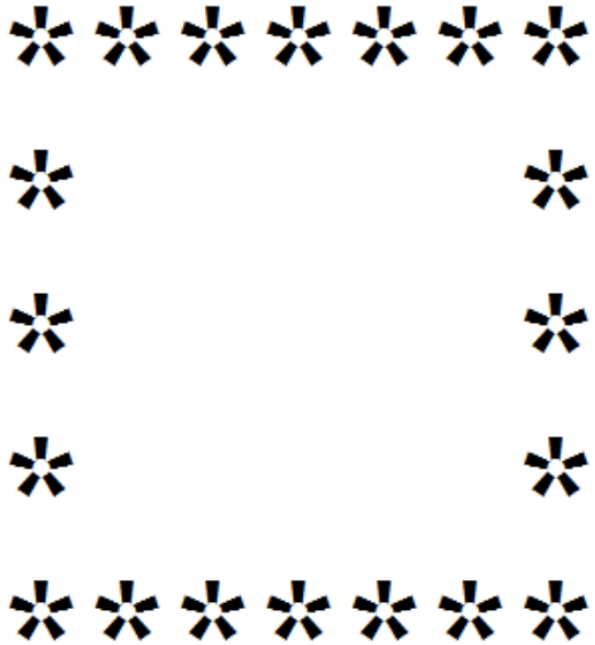
1. ที่จริงเราควรเริ่มจากการเขียนโปรแกรมที่รับข้อมูลมาแค่นิดเดียวกัน เช่น อาจรับมาแค่ 2 คนและกำหนดให้มีการสอบย่อยแค่ 3 ครั้งก่อน
2. จากข้อมูลเล็กน้อย ลองทดสอบดูก่อนว่าโปรแกรมทำงานถูกหรือเปล่า
3. เมื่อแน่ใจแล้วว่าโปรแกรมทำงานกับข้อมูลขนาดเล็กได้สำเร็จแล้ว จึงเขียนโปรแกรมกับข้อมูลขนาดใหญ่ขึ้น (เปลี่ยนค่าตัวแปรใน loop for/while/do..while)
4. ทำแบบนี้เวลาทดสอบโปรแกรมจะทดสอบได้ง่ายขึ้นเพราะไม่ต้องคอยพิมพ์ข้อมูลจำนวนมาก เราเริ่มจากส่วนเล็ก ๆ ก่อนเวลาทดสอบ โปรแกรมจะได้ทดสอบได้เร็ว ๆ
5. การรู้จักพิมพ์ข้อมูลที่เกี่ยวข้องออกมาทางหน้าจอช่วยให้เราหาที่ผิดได้ง่ายขึ้น (ข้อมูลที่ว่าอาจจะไม่ต้องเป็นผลลัพธ์ก็ได้ เทคนิคนี้ผมใช้ตลอดเวลาแม้กระทั่งเวลาสอนแล็บ)

ตัวอย่างโจทย์ (7)

การใช้แถวลำดับสองมิติแทนภาพ

ภาพที่เราเห็นเป็นสองมิติ เราสามารถใช้อาเรย์สองมิติแทนภาพได้

- เราสามารถเขียนภาพลงในอาเรย์จนเสร็จแล้วพิมพ์ภาพออกมาทีเดียว
- ข้อดีของการใช้แถวลำดับก็คือ ตอนเราแก้ไขภาพเราไม่ต้องแก้ไปทีละแถวก็ได้ เราแก้ตำแหน่งไหนก่อนก็ได้



เช่น จงเขียนภาพกรอบขนาด 5 x 7 (สูง x กว้าง) โดยให้ขอบเป็นเครื่องหมาย *

ตัวอย่างโจทย์ (7)

การใช้แถวลำดับสองมิติแทนภาพ

โจทย์ข้อนี้ต้องระมัดระวังเรื่องค่าขยะ เพราะเป็นการประกาศตัวแปรและแสดงผลเลย (ไม่ได้มีการรับค่าให้ตัวแปรแต่ละตัว)

เราอาจเคยได้ยินคำว่าค่าขยะมาบ้างแล้ว วันนี้เราจะมาลองดูกัน

ตัวอย่างโจทย์ (7)

การใช้แถวลำดับสองมิติแทนภาพ: กำจัดค่าขยะ

นักศึกษาคิดว่า โปรแกรมต่อไปนี้จะพิมพ์ค่าอะไรออกทางจอภาพ

```c
#include <stdio.h>

void main() {
    char a[5][7];
    int row, col;

    for(row=0; row<5; row++){
        for(col=0; col<7; col++){
            printf("%c", a[row][col]);
        }
        printf("\n");
    }
}
```

ตัวอย่างโจทย์ (7)

การใช้แถวลำดับสองมิติแทนภาพ: กำจัดค่าขยะ

นักศึกษาคิดว่า โปรแกรมต่อไปนี้จะพิมพ์ค่าอะไรออกทางจอภาพ

ตอบ บอกไม่ได้เลย เพราะรันกี่ครั้ง

ผลก็ไม่เหมือนกันเลย (ผลออกแบบสุ่ม

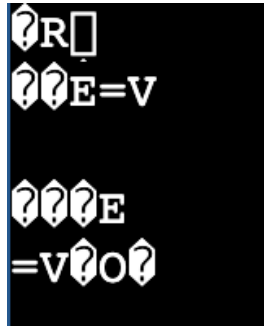
ขึ้นอยู่กับพื้นที่ในเมมโมรี่ที่ไปจอง

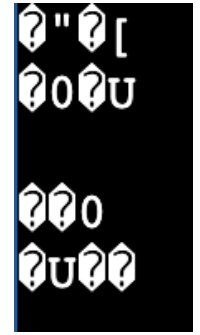
ซึ่งไม่รู้ว่าคอมไพล์เลอร์ไปจองที่ไหนให้)

```c
#include <stdio.h>

void main() {
    char a[5][7];
    int row, col;

    for(row=0; row<5; row++){
        for(col=0; col<7; col++){
            printf("%c", a[row][col]);
        }
        printf("\n");
    }
}
```





ตัวอย่างโจทย์ (7)

การใช้แถวลำดับสองมิติแทนภาพ: กำจัดค่าขยะ

ถ้าเราต้องการกำจัดค่าขยะ ก็จะต้องใส่ค่าที่เราต้องการเข้าไปแทน

(ในที่นี้ใส่เว้นวรรค ‘ ’)

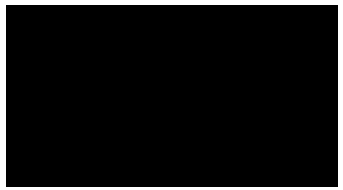
หลังจากนี้ถ้าลอง printf ออกมา จะไม่มีค่าขยะออกมาแล้ว (เป็นเว้นวรรคออกมาแทน)

```c
#include <stdio.h>

void main() {
    char a[5][7];
    int row, col;

    for(row=0; row<5; row++){
        for(col=0; col<7; col++){
            a[row][col] = ' ';
        }
    }

    for(row=0; row<5; row++){
        for(col=0; col<7; col++){
            printf("%c", a[row][col]);
        }
        printf("\n");
    }
}
```



ตัวอย่างโจทย์ (7)

การใช้แถวลำดับสองมิติแทนภาพ: ปรับแต่งอาเรย์

แต่โจทย์ต้องการให้ตรงขอบเป็นเครื่องหมาย ดอกจัน * ซึ่งหมายความว่าเราต้องปรับแต่งข้อมูลในอาเรย์อีกนิด

```
for(col = 0; col < 7; ++col) {
    a[0][col] = '*';
    a[4][col] = '*';
}
for(row = 0; row < 5; ++row) {
    a[row][0] = '*';
    a[row][6] = '*';
}
```

R\C	0	1	2	3	4	5	6
0	*	*	*	*	*	*	*
1	*						*
2	*						*
3	*						*
4	*	*	*	*	*	*	*

ตัวอย่างโจทย์ (7)

การใช้แถวลำดับสองมิติแทนภาพ: โค้ดรวม

ในเมื่อหลักการคือ เคลียร์ขยะ ปรับแต่งค่า พิมพ์ออกทางจอภาพ เมื่อนำโค้ดมาประกอบกันก็จะได้โค้ดหน้าตาดังนี้

```c
#include <stdio.h>

void main() {
    char a[5][7];
    int row, col;

    for(row=0; row<5; row++){
        for(col=0; col<7; col++){
            a[row][col] = ' ';
        }
    }
    for(col = 0; col < 7; col++) {
        a[0][col] = '*';
        a[4][col] = '*';
    }
    for(row = 0; row < 5; row++) {
        a[row][0] = '*';
        a[row][6] = '*';
    }
    for(row=0; row<5; row++){
        for(col=0; col<7; col++){
            printf("%c", a[row][col]);
        }
        printf("\n");
    }
}
```

สรุป + ทิป

- แถวลำดับเป็นสิ่งที่สำคัญมาก คาดว่างานส่วนใหญ่ในโลกล้วนต้องใช้แถวลำดับ เพราะเราต้องจัดการกับข้อมูลชนิดเดียวกันเป็นปริมาณมาก
- เวลาทำงานกับแถวลำดับสองมิติ ไม่ควรใช้ตัวแปรชื่อ i, j ในการอ้างถึงข้อมูล การเลือกใช้คำว่า row และ col หรือคำที่สื่อความหมายมักจะดีกว่า
- ถ้าใช้ i กับ j คนจำนวนมากจะหลงและมักนำไปสู่โปรแกรมที่ผิด (ฐ้าร้ายยังหาเจอยากเพราะ i กับ j มันดูคล้ายกัน)
- การเขียนหรืออ่านค่าในแถวลำดับไม่จำเป็นต้องเรียงในทิศใดทิศหนึ่ง
- เราอยากอ้างถึงข้อมูลตรงไหนก็ได้ตามใจชอบทันที
  - การอ้างถึงข้อมูลจุดใดก็ได้ทันทีแบบนี้เรียกว่า การเข้าถึงแบบสุ่ม (random access)
  - ส่วนการอ้างถึงข้อมูลแบบที่ต้องผ่านตัวแรกก่อนที่จะค่อย ๆ ไล่ไปตัวที่สอง สาม สี่ ไปเรื่อย ๆ จะเรียกว่าการเข้าถึงโดยลำดับ (sequential access)