



การเขียนโปรแกรมคอมพิวเตอร์ 1

Computer Programming I

แถวลำดับ (Array)

ภิญโญ แท้ประสาทสิทธิ์

Emails : pinyotae+111 at gmail dot com, pinyo at su.ac.th

Web : <http://www.cs.su.ac.th/~pinyotae/compro1/>

Facebook Group : [ComputerProgramming@CPSU](https://www.facebook.com/ComputerProgramming@CPSU)

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

สัปดาห์ที่ 9



หัวข้อเนื้อหา

- ข้อจำกัดของโปรแกรมที่ผ่านมา
- แถวลำดับพื้นฐาน
- การประยุกต์ใช้แถวลำดับพื้นฐาน
- แถวลำดับหลายมิติ
- การประยุกต์ใช้แถวลำดับหลายมิติ

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

2

พิจารณาตัวอย่าง



ตัวอย่าง จงเขียนโปรแกรมที่รับจำนวนเต็มมา 3 จำนวน จากนั้นพิมพ์ตัวเลขดังกล่าวในลำดับจากหลังไปหน้า

วิเคราะห์ เราสามารถใช้ตัวแปร 3 ตัว x, y, z มาเก็บค่าไว้ จากนั้นสั่งพิมพ์ z, y และ x ตามลำดับ

```
void main() {
    int x, y, z;
    scanf("%d %d %d", &x, &y, &z);
    printf("%d %d %d", z, y, x);
}
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

3

พิจารณาตัวอย่างเพิ่มเติม



ตัวอย่าง จงเขียนโปรแกรมที่รับจำนวนเต็มมา 6 จำนวน จากนั้นพิมพ์ตัวเลขดังกล่าวในลำดับจากหลังไปหน้า

วิเคราะห์ เราสามารถใช้ตัวแปร 6 ตัว u, v, w, x, y, z มาเก็บค่าไว้ จากนั้นสั่งพิมพ์ z, y, x, w, v และ u ตามลำดับ

```
void main() {
    int u, v, w, x, y, z;
    scanf("%d %d %d %d %d %d", &u, &v, &w, &x, &y, &z);
    printf("%d %d %d %d %d %d", z, y, x, w, v, u);
}
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

4

แล้วถ้าแบบนี้ล่ะ



ตัวอย่าง จงเขียนโปรแกรมที่รับจำนวนเต็มมา 300 จำนวน จากนั้นพิมพ์ตัวเลขดังกล่าวในลำดับจากหลังไปหน้า

วิเคราะห์ เราสามารถใช้ตัวแปร 300 ตัว x_1, x_2, \dots, x_{300} มาเก็บค่าไว้ จากนั้นสั่งพิมพ์ $x_{300}, x_{299}, x_{298}, \dots, x_2$ และ x_1 ตามลำดับ

THIS IS MADNESS!

ต้องหาทางที่จะทำให้เราเก็บตัวเลขเป็นลำดับยาว ๆ ได้

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

5

ข้อจำกัดของโปรแกรมที่ผ่านมา



- งานที่ต้องจำข้อมูลแบบเดียวกันจำนวนมาก ๆ ไว้ทั้งหมดเป็นสิ่งที่ยากในวิธีเขียนโปรแกรมที่ผ่านมา
- สังเกตหรือไม่ว่าในงานที่ผ่านมา หากงานไหนที่รับข้อมูลเข้าแบบไม่จำกัดจำนวน เราจะพิจารณาค่าของข้อมูลนั้นแล้วก็ทิ้งข้อมูลนั้นไปได้
 - เช่น นับว่ามีเลขคู่กี่ตัว พอเราตรวจดูว่าหารด้วยสองลงตัวหรือเปล่าเสร็จแล้วเราก็ทิ้งตัวเลขที่เรารับมานั้นไปได้ เก็บแค่ตัวนับไว้ก็พอ
 - แต่ในตัวอย่างพิมพ์เลขย้อนหลัง เราจำเป็นที่จะต้องจำค่าทุกค่าเอาไว้จริง ๆ
- วิธีเขียนโปรแกรมแบบที่ผ่านมา ไม่เอื้ออำนวยให้เราจำค่าทุกค่าเป็นจำนวนมาก ๆ ไว้ เพราะวิธีดั้งเดิมบังคับให้เราต้องตั้งชื่อตัวแปรมาใหม่
 - เราต้องการวิธีเก็บข้อมูลมาก ๆ ไว้ภายใต้ชื่อเดียวกัน

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

6

แถวลำดับ



- แถวลำดับหรือที่คนนิยมเรียกว่าอาเรย์ เป็นวิธีเก็บข้อมูลจำนวนมากไว้ภายใต้ชื่อเดียวกัน โดยจะเก็บข้อมูลไว้เป็นแถวต่อเนื่องกันไป
- แถวจะถูกแบ่งเป็นช่องเรียงตามหมายเลข

หมายเลข 0 1 2 3 4 5 6 7 8 9
แถวลำดับ

?	?	?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---	---

- จากตัวอย่างข้างบนแถวลำดับนี้เก็บข้อมูลได้สูงสุด 10 ตัว
 - แต่หมายเลขอ้างอิงจะเริ่มจาก 0 จึงสิ้นสุดที่เลข 9
- แถวลำดับเก็บข้อมูลได้หลายตัว แต่ทุกตัวต้องเป็นชนิดเดียวกัน เช่น เป็น int เหมือนกันหมด จะให้แถวลำดับเก็บค่า int ปน float ไม่ได้

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

7

การสร้างแถวลำดับ



- ถ้าหากเราต้องการเก็บเลขจำนวนเต็ม 10 ตัวไว้ด้วยกันภายใต้ชื่อ A เราเขียนว่า `int A[10];`
- ชนิดข้อมูลต้องนำหน้าชื่อเช่นเดียวกับการประกาศตัวแปรทั่วไป
- เราใช้วงเล็บเหลี่ยมหลังชื่อแถวลำดับ และเราใส่ตัวเลขเข้าไปเพื่อบอกว่าแถวลำดับนี้จะเก็บข้อมูลได้สูงสุดกี่ตัว ในที่นี้คือเก็บได้สูงสุด 10 ตัว
- สรุปลักษณะต่างในการสร้างแถวลำดับกับตัวแปรทั่วไปคือ แถวลำดับจะมีวงเล็บเหลี่ยม (square bracket) และจำนวนข้อมูลที่จะรับได้ตามมา
 - แต่ตัวแปรทั่วไปจะมีแค่ชนิดข้อมูลและชื่อ
 - เปรียบเทียบ `int A;` กับ `int A[10];` แบบแรกเป็นตัวแปร `int` ทั่วไป แต่แบบที่สองคือแถวลำดับที่เก็บ `int` ได้สูงสุด 10 ตัว

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

8

การอ้างถึงข้อมูลในแถวลำดับ



- การอ้างถึงข้อมูลในแถวลำดับมีหลากหลายมาก แต่ตอนนี้ขอพูดแต่วิธีพื้นฐาน
- ถ้าหากเราประกาศแถวลำดับ `int A[10];`
 - เราสามารถกำหนดค่าของข้อมูลตัวแรกในแถวลำดับให้เท่ากับ 3 เราเขียนว่า `A[0] = 3;`
 - การกำหนดค่าของข้อมูลในแถวลำดับแทบจะเหมือนกับตัวแปรทั่วไปต่างกันแค่เราต้องบอกหมายเลขลำดับด้วย
 - เราสามารถอ่านค่าของข้อมูลในแถวลำดับได้ในทำนองเดียวกับการกำหนดค่าคือต้องมีหมายเลขลำดับมากำกับด้วย
 - ถ้าต้องการอ่านค่าของช่องข้อมูลแรกมาเก็บไว้ในตัวแปร `x` เราเขียนว่า `x = A[0];`

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

9

ตัวอย่างการอ้างถึงข้อมูลในแถวลำดับ



การประกาศ `int A[10];` หมายความว่า `A` คือแถวลำดับที่เก็บ `int`

- ส่วน `A[ตัวเลขลำดับ]` คือข้อมูลในแถวลำดับ เช่น ถ้าตัวเลขลำดับคือ 1 หมายถึงข้อมูลตัวที่สอง ถ้าตัวเลขลำดับคือ 9 หมายถึงข้อมูลตัวที่ 10
- การเขียนว่า `A[ตัวเลขลำดับ]` จะให้ผลเหมือนตัวแปรทั่วไปแทบทุกอย่าง
- การรับค่าจาก `scanf` ทำได้เหมือนตัวแปรทั่วไป (ถ้ามีเลขลำดับประกอบ)
 - `scanf("%d", &A[0]);` เป็นการอ่านข้อมูลจากผู้ใช้งานเก็บไว้ที่ข้อมูลตัวแรก
 - `scanf("%d", &A[7]);` เป็นการอ่านข้อมูลจากผู้ใช้งานเก็บไว้ที่ข้อมูลตัวที่ 8
- การแสดงผลจาก `printf` ก็ทำได้เหมือนกับตัวแปรทั่วไปเช่นกัน
 - `printf("%d", A[0]);` เป็นการพิมพ์ค่าของแถวลำดับตัวแรก
 - `printf("%d", A[7]);` เป็นการพิมพ์ค่าของแถวลำดับตัวที่แปด

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

10

ลองประยุกต์ใช้แก้ปัญหาแสดงเลขถอยหลัง



ตัวอย่าง จงเขียนโปรแกรมที่รับค่าตัวเลขจำนวนเต็มจากผู้เข้ามา 10 ค่า จากนั้นให้พิมพ์ตัวเลขทั้งหมดออกมาเรียงลำดับจากหลังไปหน้า

วิเคราะห์

- ค่าที่รับเข้ามามีชนิดข้อมูลเหมือนกันหมด แบบนี้ใช้แถวลำดับได้
- เราสามารถเตรียมที่เก็บข้อมูลทั้ง 10 ค่าจากแถวลำดับได้ด้วยการประกาศ `int A[10];` เราไม่จำเป็นต้องประกาศตัวแปรออกมา 10 ชื่ออีกต่อไป
- เราต้องมีการ `scanf` และ `printf` เราอาจจะเลือกทางที่เร็วกว่าก็ได้ เช่น `scanf("%d %d %d %d %d %d %d %d %d %d", &A[0], &A[1], &A[2], &A[3], &A[4], &A[5], &A[6], &A[7], &A[8], &A[9]);`
- แต่งงานที่ทำซ้ำ ๆ แบบนี้เราใช้ลูปได้ และเราควรหัดใช้ลูปกับแถวลำดับ

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

11

โค้ดตัวอย่าง



```
void main() {
    int A[10];

    int i;
    for(i = 0; i < 10; ++i) {
        scanf("%d", &A[i]);
    }

    for(i = 9; i >= 0; --i) {
        printf("%d ", A[i]);
    }
}
```

สังเกตให้ดีถึงวิธีในการไล่รับลำดับในแถวลำดับจากหน้าไปหลังและหลังไปหน้า

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

12

เราควรใช้แถวลำดับเมื่อใด



1. เมื่อต้องการเก็บข้อมูลที่มีชนิดเดียวกันมาก ๆ
2. เมื่อข้อมูลชนิดเดียวกันนั้นมีการนำไปใช้ในลักษณะเดียวกัน
3. เมื่อข้อมูลแต่ละตัวอาจถูกอ้างถึงมากกว่าหนึ่งครั้ง
(จุดนี้แหละที่เป็นจุดเปลี่ยนที่แท้จริงของการเลือกใช้แถวลำดับ)
4. เมื่อเราคิดว่าการเตรียมที่เก็บข้อมูลทั้งหมดไว้ก่อนเป็นเรื่องที่สะดวกกว่า
(อันนี้เป็นเรื่องของธรรมชาติในการใช้ความคิดของแต่ละบุคคล)

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

13

ตัวอย่างการเก็บและเรียกดูข้อมูล



ตัวอย่าง การเก็บและเรียกดูข้อมูล

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

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

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

14

วิเคราะห์ปัญหาการเก็บและเรียกดูข้อมูล



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

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

15

โค้ดสำหรับเก็บและเรียกดูข้อมูล



```
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");
}
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

16

แถวลำดับกับจำนวนข้อมูลที่ไม่แน่นอน



- ตัวอย่างที่ให้มาก่อนหน้าเราทราบจำนวนข้อมูลล่วงหน้ามาก่อน
→ เราสร้างแถวลำดับที่เก็บข้อมูลได้แบบพอดี
- จะเกิดอะไรขึ้นถ้าเราไม่ทราบจำนวนข้อมูลที่ตายตัวล่วงหน้า
 - ถ้าเรารู้จำนวนข้อมูลสูงสุดที่เป็นไปได้ก่อน เราเตรียมแถวลำดับไว้เผื่อได้
 - ทำแบบนี้จะทำให้มีบางตำแหน่งในแถวลำดับที่ไม่มีข้อมูลอยู่
 - ถ้าผู้ใช้จะเป็นคนระบุจำนวนข้อมูลมาตอนโปรแกรมทำงาน
→ เราใช้แถวลำดับพลวัต (dynamic array) เข้ามาจัดการได้
 - ถ้าจำนวนข้อมูลเพิ่มขึ้นได้เรื่อย ๆ ไม่จำกัด เราต้องใช้วิธีที่ซับซ้อนขึ้น
- หมายเหตุ แถวลำดับแบบที่เราใช้มาก่อนหน้าจะมีจำนวนข้อมูลที่เก็บได้สูงสุดแน่นอนตายตัวตั้งแต่ตอนประกาศ

ตัวอย่างการพิมพ์เลขย้อนลำดับ



ตัวอย่าง จงเขียนโปรแกรมที่รับข้อมูลเป็นเลขจำนวนเต็มบวกมา N ค่าโดยที่ N มีค่าไม่เกิน 1000 ตัว โปรแกรมจะหยุดรับค่าจากผู้ใช้เมื่อผู้ใช้ใส่เลขศูนย์หรือติดลบเข้ามา เมื่อหยุดรับค่าแล้วโปรแกรมจะพิมพ์ตัวเลขทั้งหมดที่ผู้ใช้ใส่เข้ามาย้อนลำดับจากหลังมาหน้า

กำหนดให้ผู้ใช้จะไม่ใส่เลขเกิน 1000 ตัว ดังนั้นโปรแกรมไม่ต้องคอยตรวจว่าผู้ใช้ใส่มาเกิน 1000 หรือเปล่า และกำหนดให้ผู้ใช้ใส่เลขบวกอย่างน้อยหนึ่งค่า

วิเคราะห์

1. เราไม่รู้ค่า N ที่ตายตัวล่วงหน้า แต่รู้อย่างไร N ก็ไม่เกิน 1,000 ดังนั้นเราประกาศ `int A[1000];` ไว้ได้
2. ต้องมีตัวแปรคอยนับค่า N เพื่อติดตามว่าผู้ใช้ใส่เลขบวกเข้ามาก็ตัวกันแน่

โค้ดสำหรับพิมพ์เลขย้อนลำดับมากที่สุด 1,000 ตัว



```
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]);
}
```

เปลี่ยนโจทย์การพิมพ์เลขย้อนลำดับเล็กน้อย



ตัวอย่าง จงเขียนโปรแกรมที่รับข้อมูลเป็นเลขจำนวนเต็มบวกมา N ค่าโดยที่ N มีค่าไม่เกิน 1000 ตัว โปรแกรมจะหยุดรับค่าจากผู้ใช้เมื่อผู้ใช้ใส่เลขศูนย์หรือติดลบเข้ามา เมื่อหยุดรับค่าแล้วโปรแกรมจะพิมพ์ตัวเลขทั้งหมดที่ผู้ใช้ใส่เข้ามาย้อนลำดับจากหลังมาหน้า

กำหนดให้ผู้ใช้จะไม่ใส่เลขเกิน 1000 ตัว ดังนั้นโปรแกรมไม่ต้องคอยตรวจว่าผู้ใช้ใส่มาเกิน 1000 หรือเปล่า และหากผู้ใช้ไม่ได้ใส่เลขบวกเข้ามาเลยให้โปรแกรมพิมพ์ว่า No input และจบการทำงาน

วิเคราะห์ เราทำเหมือนเดิม เพียงแต่ให้ตรวจเพิ่มเติมว่า N ที่ได้เป็นศูนย์หรือเปล่า ถ้าเป็นศูนย์ก็ให้พิมพ์คำว่า No input ออกมาแทน

ตรรกะในการจัดการกรณีที่ไม่มีค่าบวก (1)



มีมากกว่าหนึ่งแบบ มาดูแบบชัดเจนกันก่อน

```
while(1) {
    .....
}

if(N > 0) {
    int i;
    for(i = N-1; i >= 0; --i) {
        printf("%d\n", A[i]);
    }
}
else {
    printf("No input\n");
}
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

21

ตรรกะในการจัดการกรณีที่ไม่มีค่าบวก (2)



ที่นี้มาดูแบบที่ดูยากขึ้น แบบนี้จะไม่มี if-else มีแค่ if ตัวเดียว

```
while(1) {
    .....
}

int i;
for(i = N-1; i >= 0; --i) {
    printf("%d\n", A[i]);
}

if(N == 0)
    printf("No input\n");
```

ดูออกหรือไม่ว่าทำไมโปรแกรมถึงทำงานถูกต้อง ?

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

22

เปลี่ยนโจทย์การพิมพ์เลขย้อนลำดับอีกที



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

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

23

โค้ดสำหรับพิมพ์เลขย้อนลำดับ N ตัว



เมื่อไม่ต้องติดตามนับค่า N และ break; ลูป โปรแกรมจะง่ายขึ้นมาก

```
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]);
}
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

24

หัวข้อเนื้อหา



- ข้อจำกัดของโปรแกรมที่ผ่านมา
- แกลวลำดับพื้นฐาน
- การประยุกต์ใช้แกลวลำดับพื้นฐาน
- **แกลวลำดับหลายมิติ**
- การประยุกต์ใช้แกลวลำดับหลายมิติ

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

25

แกลวลำดับสองมิติ



- แกลวลำดับพื้นฐาน หรือแกลวลำดับหนึ่งมิติเป็นเหมือนคนเข้าคิวยืนต่อกันไป

หมายเลข 0 1 2 3 4 5 6 7 8 9

แกลวลำดับ

?	?	?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---	---

- แกลวลำดับสองมิติเหมือนลูกเสือ เนตรนารี หรือทหารเข้าแถว

หมายเลข 0 1 2 3 4 5 6 7 8 9 (คอลัมน์)

0

?	?	?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---	---

1

?	?	?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---	---

2

?	?	?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---	---

(แถว)

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

26

เราใช้แกลวลำดับสองมิติทำอะไรบ้าง



- ส่วนมากจะใช้เป็นตารางเก็บข้อมูล เช่น
 - หากมีนักเรียน 150 คนและมีคะแนนสอบย่อย 5 ครั้ง
→ สร้างแกลวลำดับที่มี 150 แถว 5 คอลัมน์
 - ในการแข่งขันโอลิมปิกที่มี 227 ประเทศเข้าแข่งขันและมีการบันทึกจำนวนเหรียญทอง เงิน และทองแดงเก็บไว้
→ สร้างแกลวลำดับที่มี 227 แถว 3 คอลัมน์ (ไม่จำเป็นต้องเก็บเหรียญรวม)
- ใช้เก็บข้อมูลเมตริกซ์ เช่น ถ้าต้องการเก็บข้อมูลของเมตริกซ์ขนาด 3×3
→ สร้างแกลวลำดับที่มี 3 แถว 3 คอลัมน์
- ใช้เก็บรูปภาพ เช่นรูปขนาด 1600×1200 พิกเซล (กว้าง \times สูง)
→ สร้างแกลวลำดับที่มี 1200 แถว (แถวคือความสูง)
1600 คอลัมน์ (คอลัมน์คือความกว้าง)

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

27

การประกาศแกลวลำดับสองมิติ



- แกลวลำดับหนึ่งมิติมิวงเล็บสี่เหลี่ยมตามหลังชื่อแกลวลำดับเป็นจำนวน 1 คู่
เช่น `int A[10];`
- แกลวลำดับสองมิติมิวงเล็บสี่เหลี่ยมตามหลังชื่อแกลวลำดับเป็นจำนวน 2 คู่
เช่น `int S[150][5];` (เลขตัวแรกคือจำนวนแถว ตัวที่สองคือจำนวนคอลัมน์)
`int Medal[227][3];`
`char Image[1200][1600];`
`double Matrix[3][3];` (ข้อมูลในเมตริกซ์มักเป็นเลขทศนิยม)

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

28

การอ้างอิงข้อมูลในแถวลำดับสองมิติ



- คล้ายกับกรณีแถวลำดับหนึ่งมิติ เพียงแต่ครั้งนี้เราต้องใส่ทั้งลำดับแถวและลำดับคอลัมน์ (ทั้งแถวและคอลัมน์ ลำดับเริ่มจากศูนย์)
 - กำหนดค่า $S[5][1] = 5;$
(นักศึกษาคนที่หก สอบย่อยครั้งที่สองได้ 5 คะแนน)
 - อ่านค่า $\text{int } x = S[3][2];$ (อ่านค่าคะแนนจากนักเรียนคนที่ 4 การสอบย่อยครั้งที่ 3 มาเก็บไว้ที่ตัวแปร x)
- การ scanf และ printf ก็ทำในลักษณะเดิม
 - `scanf("%d", &S[5][1]);`
 - `printf("%d %d\n", S[0][3], S[1][3]);`

ตัวอย่าง โปรแกรมบันทึกคะแนนสอบย่อย



ตัวอย่าง กำหนดให้ชั้นเรียนมีนักศึกษาทั้งหมด 150 คน และมีการสอบย่อยทั้งหมด 5 ครั้ง คะแนนการสอบแต่ละครั้งมีชนิดข้อมูลเป็นเลขทศนิยมแบบ float จงเขียนโปรแกรมที่ให้ผู้ใช้นำข้อมูลคะแนนนักศึกษาเข้าไปทีละคน การบันทึกคะแนนนี้จะบันทึกคะแนนทั้ง 5 ครั้งเข้าไปด้วยกันแล้วจึงรับคะแนนของนักศึกษาค้นถัดไป

หมายเหตุ โปรแกรมนี้ต้องการเพียงแสดงให้เห็นถึงวิธีเขียนข้อมูลเข้าไปในแถวลำดับสองมิติ โปรแกรมยังไม่ได้ทำอะไรที่เป็นประโยชน์ในทางปฏิบัติ

โค้ดโปรแกรมบันทึกคะแนนสอบย่อย



```
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]);
        }
    }
}
```

ตัวอย่าง โปรแกรมบันทึกและคำนวณคะแนนสอบย่อย



ตัวอย่าง กำหนดให้ชั้นเรียนมีนักศึกษาทั้งหมด 150 คน และมีการสอบย่อยทั้งหมด 5 ครั้ง คะแนนการสอบแต่ละครั้งมีชนิดข้อมูลเป็นเลขทศนิยมแบบ float จงเขียนโปรแกรมที่ให้ผู้ใช้นำข้อมูลคะแนนนักศึกษาเข้าไปทีละคน การบันทึกคะแนนนี้จะบันทึกคะแนนทั้ง 5 ครั้งเข้าไปด้วยกันแล้วจึงรับคะแนนของนักศึกษาค้นถัดไป

เมื่อได้ข้อมูลมาครบแล้ว โปรแกรมจะพิมพ์ผลรวมคะแนนสอบของแต่ละคนออกมาตามลำดับ (อย่าพิมพ์จนกว่าโปรแกรมจะรับข้อมูลมาจนหมด)

โค้ดโปรแกรมบันทึกและหาผลรวมคะแนนสอบย่อย



```
void main() {
    float S[150][5];

    int 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);
    }
}
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

33

ทิปในการเขียนโปรแกรมที่รับข้อมูลมาก ๆ



1. ที่จริงเราควรเริ่มจากการเขียนโปรแกรมที่รับข้อมูลมาแค่ชนิดเดียวกัน เช่น รับมาแค่ 2 คนและให้มีการสอบย่อยแค่ 3 ครั้ง
2. จากข้อมูลเล็กน้อยลองทดสอบดูก่อนว่าโปรแกรมทำงานถูกหรือเปล่า
3. เมื่อแน่ใจแล้วว่าโปรแกรมทำงานกับข้อมูลขนาดเล็กสำหรับแล้วจึงเขียนโปรแกรมกับข้อมูลขนาดใหญ่ขึ้น
4. ทำแบบนี้เวลาทดสอบโปรแกรมจะทดสอบได้ง่ายขึ้นเพราะไม่ต้องคอยพิมพ์ข้อมูลมาก ๆ เข้าไป เราเริ่มจากส่วนเล็ก ๆ ก่อนเวลาทดสอบโปรแกรมจะได้ทดสอบได้เร็ว ๆ
5. การรู้จักพิมพ์ข้อมูลที่เกี่ยวข้องออกมาทางหน้าจอช่วยให้เราหาที่ผิดได้ง่ายขึ้น (ข้อมูลที่ว่าอาจจะไม่ต้องเป็นผลลัพธ์ก็ได้)

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

34

การใช้แถวลำดับสองมิติแทนภาพ



- ภาพที่เราเห็นเป็นสองมิติ เราสามารถใช้แถวลำดับสองมิติแทนภาพได้
- เราสามารถเขียนภาพลงในแถวลำดับจนเสร็จแล้วพิมพ์ภาพออกมาทีเดียว
- ข้อดีของการใช้แถวลำดับก็คือ ตอนเราแก้ไขภาพเราไม่ต้องแก้ไปที่ละแถวก็ได้เราแก้ตำแหน่งไหนก่อนก็ได้

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

```
*****
*           *
*           *
*           *
*           *
*****
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

35

แก่นของการใส่ดอกจัน



```
void main() {
    char A[5][7];
    .....
    ..... (1) ส่วนเตรียมค่า
    for(col = 0; col < 7; ++col) {
        A[0][col] = '*';
        A[4][col] = '*';
    }
    for(row = 0; row < 5; ++row) {
        A[row][0] = '*';
        A[row][6] = '*';
    }
    .....
    ..... (2) ส่วนพิมพ์ผลลัพธ์
}
```

ใส่ดอกจันในแถวแรกและแถวสุดท้าย

ใส่ดอกจันในคอลัมน์แรกและสุดท้าย

ในตอนที่เรายังไม่พิมพ์ เราจะใส่ดอกจันตามแถวหรือคอลัมน์ก็ได้ทั้งนั้น

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

36

ตอนพิมพ์ก็วิ่งลูปพิมพ์ออกมาทีละแถว



```
char A[5][7];
```

(2) ส่วนพิมพ์ผลลัพธ์

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

ดึงตัวอักษรที่เก็บไว้ใน A ออกมาพิมพ์ทีละตัวไปเรื่อย ๆ จนหมด

สังเกตด้วยว่าส่วนของการคิดตำแหน่งตอกจันกับการพิมพ์จะแยกออกจากกัน

→ ช่วยให้เราแบ่งงานเขียนโปรแกรมออกเป็นหลายส่วนที่ไม่ซับซ้อนได้

อย่าลืมเตรียมค่าเริ่มต้นในภาพ



ค่าในแถวลำดับตอนแรกจะมีอะไรก็ไม่อาจทราบได้ ดังนั้นต้องเตรียมค่าไว้ก่อน

```
void main() {  
    char A[5][7];  
  
    int row, col;  
    for(row = 0; row < 5; ++row) {  
        for(col = 0; col < 7; ++col) {  
            A[row][col] = ' '  
        }  
        .....  
    }  
}
```

เริ่มแรกกำหนดให้ตัวอักษรเป็นช่องว่างให้หมดจากนั้นค่อยใส่ตอกจันทีละลงไป

การเตรียมค่าเริ่มต้นแล้วเขียนทับค่าเดิมเป็นเทคนิคที่พบบ่อยในการจัดการภาพ

ก่อนไปสู่เรื่องต่อไป



- แถวลำดับเป็นสิ่งที่สำคัญมาก คาดว่าโปรเจกต์จบการศึกษาทุกคนต้องใช้แถวลำดับ เพราะเราต้องจัดการกับข้อมูลชนิดเดียวกันเป็นปริมาณมาก
- เวลาทำงานกับแถวลำดับสองมิติ ไม่ควรใช้ตัวแปรชื่อ i, j ในการอ้างถึงข้อมูล แต่ควรใช้คำว่า row และ col หรืออื่น ๆ ที่สื่อความหมายที่ดี
- ถ้าใช้ i กับ j คนจำนวนมากจะหลงและมักนำไปสู่โปรแกรมที่ผิด (ซำร้ายยังหาเจอยากเพราะ i กับ j มันดูคล้ายกัน)
- การเขียนหรืออ่านค่าในแถวลำดับไม่จำเป็นต้องเรียงในทิศใดทิศหนึ่ง เราอยากอ้างถึงข้อมูลตรงไหนก็ได้ตามใจชอบทันที
 - การอ้างถึงข้อมูลจุดใดก็ได้ทันทีแบบนี้เรียกว่า การเข้าถึงแบบสุ่ม (random access)
 - ส่วนการอ้างถึงข้อมูลแบบที่ต้องผ่านตัวแรกก่อนที่จะค่อย ๆ ไล่ไปตัวที่สอง สาม สี่เรื่อย ๆ จะเรียกว่าการเข้าถึงโดยลำดับ (sequential access)