บทที่ 8 แถวลำดับ (Array)

วัตถุประสงค์

1. เข้าใจการเก็บข้อมูลแบบแถวลำดับ

2. เขียนโปรแกรมแก้ปัญหาทางคณิตศาสตร์เบื้องต้นโดยใช้แถวลำดับได้

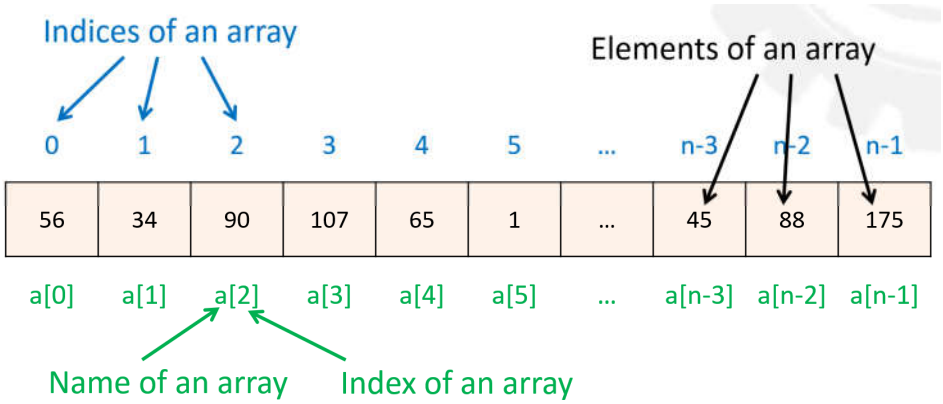
แถวลำดับ หรือ อะเรย์ (array) คือโครงสร้างของข้อมูลแบบหนึ่งที่เก็บข้อมูลหลายๆ ตัวไว้ด้วยกันโดยจัดเก็บให้**ต่อกัน**ตามลำดับในหน่วยความจำ (memory) และเราสามารถเข้าถึงข้อมูลแต่ละตัวได้ด้วยเลขดัชนี (index number) ซึ่งเริ่มจาก 0 ถึง n-1 เมื่อ n เป็นจำนวนของสมาชิก (element) ทั้งหมดในอะเรย์ ในภาษาซีพลัสพลัส อะเรย์หนึ่งจะสามารถเก็บข้อมูลได้เพียงชนิดเดียวเท่านั้น เราจะไม่สามารถเก็บข้อมูลต่างชนิดกันไว้ในอะเรย์เดียวกันได้ เช่น เราสามารถเก็บข้อมูลชนิด float 10 ตัวไว้ในอะเรย์ A ได้ แต่เราไม่สามารถเก็บข้อมูลชนิด float และ string เข้าไว้ในอะเรย์เดียวกันได้ ถ้าเราต้องการเก็บข้อมูลทั้ง float และ string หลายตัว เราต้องแยกเป็นอะเรย์สำหรับ float หนึ่งอะเรย์ และอะเรย์สำหรับ string หนึ่งอะเรย์

ในบทเรียนนี้เราจะเรียนอะเรย์ 3 แบบคือ อะเรย์ 1 มิติ 2 มิติ และ 3 มิติ นอกจากนี้เราจะเรียนวิธีการใช้อะเรย์ในฟังก์ชัน และการค้นหาและเรียงลำดับอะเรย์อีกด้วย

8.1 แถวลำดับหนึ่งมิติ (1D Array)

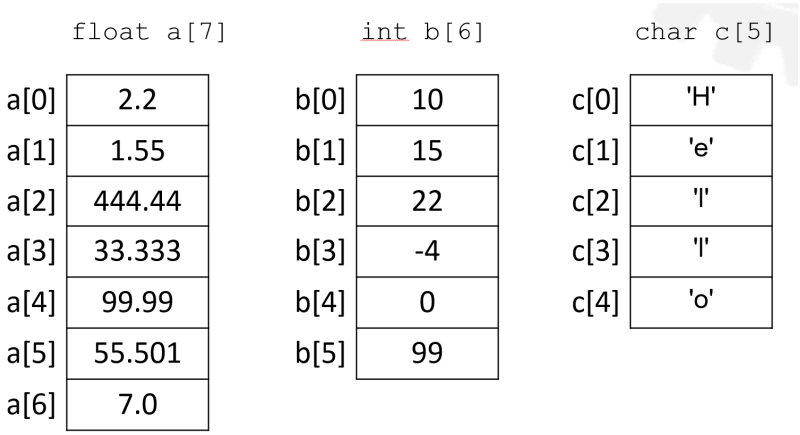
อะเรย์มีส่วนประกอบที่สำคัญ 2 อย่างคือ สมาชิกของอะเรย์ (element) และดัชนี (index) สมาชิกของอะเรย์คือวัตถุหรือข้อมูลที่เรียงกันอยู่ในอะเรย์ ส่วนดัชนีคือเบอร์อ้างอิงหรือหมายเลขตำแหน่งที่อยู่ของสมาชิกแต่ละตัว เราใช้ดัชนีนี้ในการเข้าถึงสมาชิกในตำแหน่งใดๆ ของอะเรย์ โดยดัชนีของอะเรย์จะเริ่มจาก 0 เสมอจนไปถึง n-1 เมื่อ n คือจำนวนสมาชิกของอะเรย์ นั่นคือเราจะต้องใช้ดัชนี i-1 เมื่อเราต้องการเข้าถึงสมาชิกของอะเรย์ตัวที่ i (สมาชิกตัวที่ i อยู่ที่ตำแหน่ง i-1) นอกจากนี้ ดัชนีของอะเรย์ยังหมายถึงจำนวนขั้น (step) ของกล่องหน่วยความจำที่สมาชิกตัวนั้นๆอยู่ห่างจากกล่องของสมาชิกตัวแรก เช่น สมาชิกของอะเรย์ที่ตำแหน่งที่ 3 (สมาชิกตัวที่ 4) จะอยู่ห่างจากสมาชิกตัวแรกเป็นจำนวน 3 ขั้นหรือ 3 ห้องของหน่วยความจำ นั่นคือ สมาชิกของอะเรย์ที่ตำแหน่งที่ k จะอยู่ห่างจากสมาชิกตัวแรกเป็นจำนวน k ขั้นหรือ k ห้องของหน่วยความจำ ดังแสดงในรูปที่ 77

เราสามารถมองอะเรย์1 มิติ เป็นห้องแถวที่เรียงต่อกันจำนวน n ห้องได้ และเบอร์ของห้องแถวนี้คือดัชนี โดยเริ่มจากห้องที่ 0 ไปจนถึงห้องที่ n-1 รูปที่ 77 แสดงให้เห็นถึงอะเรย์ 1 มิติของจำนวนเต็มซึ่งมีชื่อว่า a ที่มีขนาด n ตัว ในการเข้าถึงสมาชิกของอะเรย์ เราสามารถเข้าถึงด้วยการเขียนชื่ออะเรย์ตามด้วยวงเล็บสี่เหลี่ยม ([ ]) และเขียนดัชนีไว้ในวงเล็บนั้น เช่นถ้าต้องการเข้าถึงสมาชิกของอะเรย์ a ตัวแรก เราเขียนว่า a[0] อ่านว่าสมาชิกของอะเรย์ a ที่ตำแหน่งที่ 0 หรือถ้าต้องการเข้าถึงสมาชิกของอะเรย์ a ตัวที่ 5 เราเขียนว่า a[4] อ่านว่าสมาชิกของอะเรย์ a ที่ตำแหน่งที่ 4



รูปที่ 77 อะเรย์หนึ่งมิติ

ตัวอย่างของอะเรย์ชนิด float, int และ char แสดงตามลำดับในรูปที่ 78



รูปที่ 78 ตัวอย่างของอะเรย์ชนิด float, int และ char

หมายเหตุ เราไม่จำเป็นต้องเข้าถึงอะเรย์ตามลำดับไปทีละตัว เช่นถ้าเราต้องการเข้าถึงอะเรย์ที่ตำแหน่งที่ 7 เราไม่จำเป็นต้องเข้าถึงอะเรย์ตำแหน่งที่ 0-6 ก่อน เราสามารถเข้าถึงอะเรย์ตัวที่เราต้องการโดยการใช้ a[7] ได้เลย

การใช้อะเรย์ในภาษาซีพลัสพลัสนั้นเริ่มด้วยการประกาศตัวแปรที่ใช้เก็บอะเรย์เช่นเดียวกับข้อมูลพื้นฐานอย่างจำนวนจริงหรือจำนวนเต็ม เราใช้รูปแบบการประกาศอะเรย์โดยเริ่มด้วยการกำหนดชนิดของอะเรย์ ต่อด้วยชื่ออะเรย์และวงเล็บสี่เหลี่ยมซึ่งใส่ขนาดของอะเรย์ (จำนวนสมาชิกของอะเรย์) ตามตัวอย่างดังนี้

```
int a[100];
```

หมายความว่าเราประกาศอะเรย์ชนิดจำนวนเต็ม ชื่อ a และสามารถมีสมาชิกได้ 100 ตัวขึ้นมาใช้ โดยขอย้ำอีกทีว่าสมาชิกทุกตัวของอะเรย์นี้จะต้องมีชนิดเป็น int ด้วยเช่นกัน ถ้าเราพยายามเก็บข้อมูลชนิดอื่น เช่นจำนวนจริง เข้ามาในอะเรย์ โปรแกรมจะเปลี่ยนรูปแบบของข้อมูลให้เป็นจำนวนเต็ม ดังนั้นเราจึงต้องระมัดระวังการใช้ด้วย

ส่วนการให้ค่าหรืออ่านค่าของสมาชิกในอะเรย์นั้น เราทำได้คล้ายกับตัวแปรทั่วไป แต่แทนที่จะใช้ชื่อตัวแปรเฉยๆ เราต้องใช้ชื่ออะเรย์และตำแหน่งของสมาชิกที่เราต้องการทำงานด้วยกำกับไว้เสมอ เช่น a[2] และ a[3]

โค้ด:

```
1   int main()
2   {
3       double a[3];
4
5       a[2]= 55.55;
6       a[0]= 11.11;
7       a[1]= 33.33;
8
9       cout << "a[0] = " << a[0]<< endl;
10      cout << "a[1] = " << a[1]<< endl;
11      cout << "a[2] = " << a[2]<< endl;
12  }
```

โค้ดข้างต้นทำการประกาศอะเรย์ชนิดจำนวนจริง (double) ชื่อ a โดยกำหนดให้มีสมาชิก 3 ตัว จากนั้นก็ให้ค่าสมาชิกในตำแหน่งที่ 2 เป็น 55.55 สมาชิกในตำแหน่งที่ 0 เป็น 11.11 ในตำแหน่งที่ 1 เป็น 33.33 แล้วแสดงค่าสมาชิกในตำแหน่งที่ 0, 1 และ 2 ออกทางหน้าจอตามลำดับ

สังเกตว่าเราไม่จำเป็นจะต้องให้ค่าหรือเข้าถึงค่าสมาชิกแต่ละตัวตามลำดับ แต่เราสามารถใช้ชื่ออะเรย์และตำแหน่งของสมาชิกที่เราต้องการทำงานด้วยในการให้ค่าหรือเข้าถึงค่าได้เลย ตามที่ได้อธิบายไปก่อนหน้า

นี้แล้ว อย่างไรก็ตามโดยปกติแล้วเรานิยมใช้การวนซ้ำในการจัดการอะเรย์เนื่องจากคุณสมบัติที่มีค่าดัชนีเรียงต่อกันของอะเรย์ เพื่อทำให้การทำงานของโปรแกรมและการเขียนโปรแกรมนั้นมีประสิทธิภาพกว่าการเข้าถึงอะเรย์โดยไม่มีลำดับ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 63 การอ่านและเขียนอะเรย์
โจทย์ : จงเขียนโปรแกรมภาษาซีพลัสพลัสเพื่อรับจำนวนเต็ม 10 จำนวนผ่านทางแป้นพิมพ์เข้ามาใส่ในอะเรย์ จากนั้นให้พิมพ์ค่าสมาชิกของอะเรย์ออกทางหน้าจอตามลำดับจากหน้าไปหลัง (ตำแหน่งแรกไปตำแหน่งสุดท้าย) และจากหลังมาหน้า (ตำแหน่งสุดท้ายมาตำแหน่งแรก)

1) เอาต์พุต คือค่าสมาชิกของอะเรย์ออกทางหน้าจอตามลำดับจากหน้าไปหลังและจากหลังมาหน้า
2) อินพุต คือจำนวนเต็ม 10 จำนวน
3) โจทย์กำหนดให้รับค่าเข้ามาใส่ในอะเรย์ก่อน แล้วจึงพิมพ์ออก
4) เราทราบวิธีการวนลูปจากหน้าไปหลังและหลังมาหน้า

ในโจทย์ข้อนี้ เราต้องรับจำนวนเต็มเข้ามา 10 ตัวและใส่ในอะเรย์ จึงไม่มีประโยชน์ที่จะประกาศเป็นตัวแปรจำนวนเต็ม 10 ตัว แต่ให้ประกาศเป็นอะเรย์ชนิดจำนวนเต็มที่มีขนาดเป็น 10 แทน เราสามารถใช้รูปแบบการเข้าถึงอะเรย์โดยใช้ดัชนี (เบอร์ห้องแถว) ได้ โดนวนรับค่าจากแป้นพิมพ์เข้าไปยังสมาชิกของอะเรย์ตำแหน่งที่ 0, 1, 2 เพิ่มไปจนถึง 9 ตามลำดับ (บรรทัดที่ 6-10) จากนั้นเราก็ทำการวนอ่านค่าจากอะเรย์ตำแหน่งที่ 0, 1, 2 ไปจนถึง 9 ตามลำดับเช่นกัน (บรรทัดที่ 12-17) สำหรับการอ่านค่าจากหลังมาหน้า เราก็วนอ่านค่าเช่นกันแต่เริ่มจากตำแหน่งที่ 9, 8, 7 ลดลงมาเรื่อยๆ จนถึงตำแหน่งที่ 0 (บรรทัดที่ 20-25)

โค้ด:

```
1   int main()
2   {
3       const int SIZE = 10;
4       double a[SIZE];
5
6       cout << "Enter " << SIZE << " numbers: \t";
7       for (int i = 0; i < SIZE; i++)
8       {
9           cin >> a[i];
10      }
11
12      cout << "In order: ";
13      for (int i = 0; i < SIZE; i++)
14      {
```

```
15          cout << "\t" << a[i];
16      }
17      cout << endl;
18
19
20      cout << "In reverse order: ";
21      for (int i = SIZE - 1; i >= 0; i--)
22      {
23          cout << "\t" << a[i];
24      }
25      cout << endl;
26      return 0;
27  }
```

ถ้าถามว่าเราสามารถเขียนโปรแกรมข้อข้างต้นนี้โดยไม่ใช้อะเรย์ได้หรือไม่ คำตอบคือ ได้ ถ้าขนาดของจำนวนที่รับเข้ามานั้นไม่ใหญ่มากและจะไม่ถูกเปลี่ยนแปลงได้ทีหลัง เช่น ในกรณีนี้โจทย์กำหนดให้รับจำนวนเต็มทั้งหมด 10 ตัว เราอาจจะกำหนดตัวแปรที่เป็นจำนวนเต็ม (int) ขึ้นมา 10 ตัว สั่ง cin 10 ครั้ง และ cout 10 ครั้งสำหรับการแสดงออกตามลำดับ และอีก 10 ครั้งสำหรับการแสดงออกแบบหลังมาหน้าก็ได้ แต่เราจะเห็นว่าเราต้องเขียนโค้ดเป็นจำนวนมาก และให้ลองคิดดูว่าหากโจทย์กำหนดให้รับจำนวนเต็มเข้ามา 100 ตัว หรือ 1000 ตัว เราก็ไม่ควรเขียนโค้ดโดยไม่ใช้อะเรย์ เพราะจะทำให้โค้ดยาวและเสียเวลามาก แต่ถ้าเราใช้อะเรย์ ไม่ว่าโจทย์จะให้รับและแสดงจำนวนเต็มเป็นเท่าใด เราก็สามารถเขียนโค้ดได้ในแบบเดิมนี้ แต่เปลี่ยนค่าของตัวแปร size เท่านั้น ส่วนโค้ดอื่นๆ ยังเหมือนเดิม นอกจากนี้ถ้าหากเราเขียนโปรแกรมโดยไม่ใช้อะเรย์เสร็จแล้ว มีคนบอกว่าให้เปลี่ยนขนาดของอะเรย์ เราก็จะต้องมาแก้ลดหรือเพิ่มโค้ดใหม่ แต่ถ้าเราใช้อะเรย์ในการเขียน เราไม่จำเป็นต้องแก้โค้ดเลยนอกจากเปลี่ยนค่าของตัวแปร size เท่านั้น ส่วนโค้ดอื่นๆ ยังเหมือนเดิมเช่นกัน จะเห็นได้ว่าในกรณีแบบนี้ การใช้อะเรย์นั้นทำให้เราเขียนโปรแกรมที่มีโครงสร้างลักษณะนี้ได้ง่ายขึ้นมาก

การให้ค่าเริ่มต้นกับอะเรย์

หากเราประกาศอะเรย์แล้วไม่ได้ให้ค่าสมาชิกของอะเรย์ ค่าเริ่มต้นของสมาชิกอะเรย์แต่ละตัวจะเป็นค่าขยะที่ค้างอยู่ในหน่วยความจำที่อะเรย์ไปใช้อยู่ ดังตัวอย่างโค้ดข้างล่างนี้

โค้ด:

```
1   int main()
2   {
3       const int SIZE = 4;
4       float a[SIZE];
5       for (int i = 0; i < SIZE; i++)
6       {
7           cout << "\ta[" << i << "] = " << a[i] << endl;
8       }
9       return 0;
10  }
```

ผลการทำงานบนเครื่องของอาจารย์:

```
1           a[0] = nan
2           a[1] = 1.18811e+032
3           a[2] = 1.18813e+032
4           a[3] = 6.03915e-039
```

เราจะเห็นได้ว่าถ้าเราไม่ให้ค่าเริ่มต้นของอะเรย์ก่อนนำไปใช้ เราจะได้ค่าขยะที่เราไม่สามารถรู้ได้ล่วงหน้าว่าคือค่าอะไร และเป็นค่าที่ไม่เท่ากันในการรันโปรแกรมแต่ละครั้ง ทำให้โปรแกรมของเราอาจจะทำงานผิดได้ ดังนั้นเราจึงควรให้ค่าของอะเรย์ก่อนนำไปใช้ ไม่ว่าจะให้ค่าเริ่มต้นตอนประกาศอะเรย์เลย หรือว่าให้ค่าหลังจากประกาศแล้วก็ได้ เราสามารถให้ค่าเริ่มต้นกับอะเรย์ในแบบต่างๆ ได้ดังนี้

91) ให้ค่าเริ่มต้นกับสมาชิกของอะเรย์แต่ละตัว

```
a[2]= 55.55;
a[0]= 11.11;
a[1]= 33.33;
```

เราสามารถให้ค่าเริ่มต้นกับสมาชิกของอะเรย์โดยการให้ค่าแต่ละตัวโดยตรงเลยตามที่ได้อธิบายไปแล้ว

92) ให้ค่าเริ่มต้นโดยไม่ได้กำหนดขนาดของอะเรย์

```
float a[] = {22.2, 44.4, 66.6};
```

จริงๆ แล้วเราสามารถประกาศอะเรย์โดยไม่กำหนดขนาดโดยตรงได้ แต่เราต้องกำหนดค่าเริ่มต้นให้กับสมาชิกทุกตัวของอะเรย์ เช่นตัวอย่างข้างบนนี้ เราประกาศอะเรย์ a โดยไม่กำหนดขนาด แต่ให้ค่าเริ่มต้น 3 ค่า การประกาศแบบนี้จะทำให้อะเรย์ a มีขนาดเท่ากับ 3 ตามจำนวนของค่าเริ่มต้นที่เรากำหนดให้ โดยค่าที่ใส่เข้าไปจะถูกนำไปเก็บให้สมาชิกแต่ละตัวของอะเรย์ตามลำดับ โดยเริ่มจากสมาชิกที่ตำแหน่งที่ 0 นั่นคือ สำหรับอะเรย์ a ในข้อนี้ a[0] จะมีค่า 22.2, a[1] จะมีค่า 44.4 และ a[2] จะมีค่า 66.6 ตามลำดับ ดังแสดงในรูปที่ 79

a[0]	22.2
a[1]	44.4
a[2]	66.6

รูปที่ 79 ค่าของ a[0] – a[2]

93) ให้ค่าเริ่มต้นโดยกำหนดขนาดของอะเรย์ แต่ให้ค่าไม่ครบทุกค่า

```
float a[7] = {22.2, 44.4, 66.6};
```

หากเรากำหนดขนาดของอะเรย์ แต่ให้ค่าเริ่มต้นตามรูปแบบข้างต้นไม่ครบทุกตัว ค่าจะใส่เข้าไปจะถูกนำไปเก็บให้สมาชิกแต่ละตัวของอะเรย์ตามลำดับ โดยเริ่มจากสมาชิกที่ตำแหน่งที่ 0 ส่วนสมาชิกตัวที่ไม่ได้ถูกให้ค่านั้นจะมีค่าเป็น 0 โดยอัตโนมัติ ดังแสดงในรูปที่ 80

a[0]	22.2
a[1]	44.4
a[2]	66.6
a[3]	0
a[4]	0
a[5]	0
a[6]	0

รูปที่ 80 ค่าของ a[0] - a[6]

94) ให้ค่าเริ่มต้นโดยกำหนดขนาดของอะเรย์ ให้ทุกค่าเป็น 0

```
float a[9] = {0, 0, 0, 0, 0, 0, 0, 0, 0};
```

การให้ค่าเริ่มต้นแบบนี้ไม่มีอะไรซับซ้อน ค่าทุกค่าของสมาชิกจะเป็น 0 โดยขนาดของอะเรย์จะเป็นไปตามที่เรากำหนดในการประกาศ

95) ให้ค่าเริ่มต้นเป็น 0 ทั้งหมดโดยไม่ได้กำหนดขนาดของอะเรย์

```
float a[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};
```

การประกาศเช่นนี้จะให้ผลเหมือนกับข้อที่ 2) โดยค่าทุกค่าจะเป็น 0 และอะเรย์จะมีขนาดเท่ากับจำนวนที่เราใส่ค่าเข้าไป

96) ให้ค่าเริ่มต้นโดยกำหนดขนาดของอะเรย์ ให้ทุกค่าเป็น 0 แต่ไม่ได้ครบทุกค่า

```
float a[9] = {0};
```

การประกาศเช่นนี้จะให้ผลเหมือนกับข้อที่ 3) โดยอะเรย์จะมีขนาดเท่าที่เรากำหนด และสมาชิกทุกตัวจะมีค่าเป็น 0 ดังนี้วิธีที่สั้นที่สุดที่จะประกาศและให้ค่าเริ่มต้นกับอะเรย์เป็น 0 คือการประกาศอะเรย์ในรูปแบบที่ 6) นี้

ข้อควรระวังในการทำงานกับอะเรย์คือ เราไม่สามารถให้ค่าอะเรย์จากอีกอะเรย์โดยใช้คำสั่งแบบในบรรทัดที่ 4 หรือ 6 ของโค้ดข้างล่างนี้ได้

โค้ด:

```
1      float a[7] = {22.2, 44.4, 66.6};
2      float b[7] = {33.3, 55.5, 77.7};
3
4      b = a;
5
6      float c[7] = a;
```

ถ้าหากเราต้องการจะคัดลอกค่าในอะเรย์ในภาษาซีพลัสพลัส เราต้องทำการวนลูปให้ค่าอะเรย์ไปทีละตัว ดังตัวอย่างในโค้ดข้างล่างนี้

โค้ด:

```
1      float a[7] = {22.2, 44.4, 66.6};
2      float b[7];
3
```

```
4       for(int i = 0; i < 7; i++)
5       {
6           b[i] = a[i];
7       }
```

การหาขนาดของอะเรย์ในหน่วยไบต์

ก่อนหน้านี้เราเข้าใจว่า “ขนาดของอะเรย์” คือจำนวนสมาชิกของอะเรย์นั้นๆ แต่ถ้าถามว่าขนาดของอะเรย์ในหน่วยไบต์ (byte) คือเท่าใด เราจะหาได้อย่างไร ขนาดของอะเรย์ในหน่วยไบต์นั้นคือขนาดของหน่วยความจำที่เก็บข้อมูลในอะเรย์หนึ่งๆ นั่นเอง ซึ่งขึ้นอยู่กับชนิดของข้อมูลที่อะเรย์นั้นเก็บอยู่ ขนาดของอะเรย์ในหน่วยไบต์จะเท่ากับขนาดของชนิดของข้อมูลคูณด้วยจำนวนสมาชิกของอะเรย์ เช่นถ้าเรามีอะเรย์ชนิด float ซึ่งมีสมาชิก 10 ตัว ขนาดของอะเรย์จะเป็น 4 x 10 = 40 ไบต์ เนื่องจาก float มีขนาด 4 ไบต์ เราสามารถใช้ฟังก์ชัน sizeof() ในการหาขนาดของตัวแปร ชนิดข้อมูล หรืออะเรย์ในหน่วยไบต์ได้

โค้ด:

```
1   int main()
2   {
3       float a[] = {22.2, 44.4, 66.6};
4       int size = sizeof(a) / sizeof(float);
5       for (int i = 0; i < size; i++)
6       {
7           cout << "\ta[" << i << "] = " << a[i] << endl;
8       }
9       return 0;
10  }
```

จากตัวอย่างในโค้ดข้างบน ตัวแปร size คือตัวแปรที่เก็บขนาดของอะเรย์ ในที่นี้เราต้องการหาขนาดของอะเรย์เพราะโค้ดไม่ได้กำหนดมาให้โดยตรง แต่ให้ค่าเริ่มต้นกับอะเรย์มาเฉยๆ ค่าของตัวแปร size จึงหาได้จาก ขนาดของอะเรย์ในหน่วยไบต์ หารด้วยขนาดของชนิดของอะเรย์ (float) ในหน่วยไบต์เช่นกัน

การใช้ดัชนีของอะเรย์เกินขอบเขตของอะเรย์ (Array Index out of Bound)

เมื่อเราใช้ดัชนีเกินขอบเขตของอะเรย์ เช่น อะเรย์ a มีสมาชิก 6 ตัว แต่เราพยายามจะอ่านหรือเขียนที่ตำแหน่งที่ 10 ในภาษาซีพลัสพลัสคอมไพเลอร์จะไม่แจ้งเตือน เพราะมันจะเข้าใจว่าให้อ่านหรือเขียนหน่วยความจำกล่องที่ 10 นับจากตำแหน่งเริ่มต้นของอะเรย์ ซึ่งในบางกรณีนั้นตำแหน่งที่เราต้องการเข้าถึงก็อยู่ในส่วนของหน่วยความจำที่โปรแกรมของเราสามารถใช้ได้ โปรแกรมก็จะไม่เกิดการผิดพลาดขณะคอมไพล์ (compilation error) แต่มันอาจจะทำให้ค่าตัวแปรตัวอื่นในโปรแกรมของเราผิดเพี้ยนได้ในบางครั้ง (ขึ้นอยู่กับการจัดการหน่วยความจำของคอมไพเลอร์ด้วย)

โค้ด:

```
1   int main()
2   {
3       const int SIZE = 4;
4       float a[SIZE] = {33.3, 44.4, 55.5, 66.6};
5       for(int i = 0; i < 7; i++)
6       {
7           cout << "\ta[" << i << "] = " << a[i] << endl;
8       }
9       return 0;
10  }
```

จากโค้ดตัวอย่างข้างบน อะเรย์ a มีขนาดแค่ 4 ห้อง แต่ในลูป for นั้นเราพยายามเข้าถึงตำแหน่งที่ 4, 5 และ 6 ด้วย สิ่งที่จะแสดงออกมาทางหน้าจออาจจะเป็นตามผลการทำงานข้างล่างนี้ได้ ซึ่งขอให้นักศึกษาเข้าใจว่าการรันแต่ละครั้งอาจจะได้ผลไม่เหมือนกัน เพราะค่าที่อยู่ในตำแหน่งที่ 4-6 นั้นจะเป็นค่าอะไรก็ตามที่ค้างอยู่ในหน่วยความจำกล่องที่ 4, 5, และ 6 โดยนับจากตำแหน่งแรกของอะเรย์นั่นเอง

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
33.3	44.4	55.5	66.6	?	?	?

รูปที่ 81 การใช้ดัชนีของอะเรย์เกินขอบเขตของอะเรย์

ผลการทำงานบนเครื่องของอาจารย์:

```
1           a[0] = 33.3
2           a[1] = 44.4
3           a[2] = 55.5
```

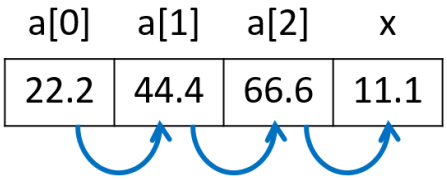
```
4           a[3] = 66.6
5           a[4] = 5.60519e-045
6           a[5] = 7.00649e-045
7           a[6] = 3.85678e-039
```

ตัวอย่างในโค้ดข้างบนนี้อาจจะไม่ทำให้เกิดปัญหามากนักกับโปรแกรมของเรา แต่ให้ลองพิจารณาโค้ดด้านล่างนี้

โค้ด:

```
1   int main()
2   {
3       const int SIZE = 4;
4       float a[] = {22.2, 44.4, 66.6};
5       float x = 11.1;
6       cout << "x = " << x << endl;
7       a[3] = 88.8;
8       cout << "x = " << x << endl;
9       return 0;
10  }
```

ในโค้ดนี้ อะเรย์ a มีขนาดเท่ากับ 3 (มีตำแหน่งสูงสุดคือ 2) และเราประกาศตัวแปร x ต่อท้ายอะเรย์ไป ดังนั้นจึงมีความเป็นไปได้ที่โปรแกรมจะเก็บค่าของตัวแปร x ต่อจากอะเรย์ a เลยในหน่วยความจำ ดังแสดงในรูปที่ 82 ดังนั้นเมื่อเราให้ค่า a[3] = 88.8 ที่บรรทัดที่ 7 โปรแกรมจะเข้าใจว่าให้เอาค่า 88.8 ไปเก็บไว้ที่กล่องในตำแหน่งที่ 3 นับจากตำแหน่งแรกของอะเรย์ ซึ่งก็คือตำแหน่งที่ x อยู่ ทำให้ค่าของตัวแปร x ถูกเปลี่ยนเป็น 88.8 โดยไม่ได้ตั้งใจได้



รูปที่ 82 การเปลี่ยนค่าตัวแปร x โดยไม่ได้ตั้งใจ

หมายเหตุ ที่ใช้คำว่า “เป็นไปได้” ที่จะเกิดกรณีแบบนี้เพราะการจัดการหน่วยความจำสำหรับตัวแปรต่างๆ นั้นขึ้นอยู่กับคอมไพเลอร์ ในบางแวดล้อมตัวแปร x จะไม่ถูกเก็บต่อจากอะเรย์ a แต่อาจจะเก็บไว้

ก่อนอะเรย์ a หรือที่อื่น ซึ่งจะทำให้ไม่เกิดกรณีผิดพลาดแบบข้างต้นได้ อย่างไรก็ตาม เวลาเขียนโปรแกรมควรตรวจสอบให้ดีว่ามีการเข้าถึงตำแหน่งที่เกินขนาดของอะเรย์หรือไม่

อีกกรณีหนึ่งที่อาจจะเป็นปัญหาของการใช้ดัชนีของอะเรย์เกินขอบเขตของอะเรย์ คือการใช้ดัชนีที่เกินขอบเขตของอะเรย์ไปมากๆ เช่น ในบรรทัดที่ 7 ของโค้ดด้านล่างนี้ โปรแกรมจะเข้าใจว่าให้เอาค่า 88.8 ไปเก็บไว้ที่กล่องในตำแหน่งที่ 3333 นับจากตำแหน่งแรกของอะเรย์ ซึ่งถือว่าไกลมากๆ ในกรณีนี้อาจจะเกิดการผิดพลาดขณะรัน (run-time error) ได้ เพราะโปรแกรมอาจจะพยายามเข้าถึงส่วนของหน่วยความจำที่ไม่ได้ถูกกำหนดให้ใช้ได้โดยโปรแกรม ในการรันโปรแกรมนี้ระบบปฏิบัติการจะแจ้งว่า name.exe has stopped working หมายความว่าโปรแกรมหยุดทำงานโดยไม่ปกติ

โค้ด:

```
1   int main()
2   {
3       const int SIZE = 4;
4       float a[] = {22.2, 44.4, 66.6};
5       float x = 11.1;
6       cout << "x = " << x << endl;
7       a[3333] = 88.8;
8       cout << "x = " << x << endl;
9       return 0;
10  }
```

8.2 การค้นหาและเรียงสมาชิกในแถวลำดับ (Searching and Sorting Arrays)

หากเราต้องการหาว่าในอะเรย์หนึ่งมีค่าที่เราต้องการหรือไม่ เราสามารถใช้วิธีการค้นหาได้หลายแบบ แบบที่ง่ายที่สุดคือการค้นหาแบบเส้นตรง (linear search) คือการหาค่าโดยการเปรียบเทียบค่าที่เราต้องการหากับค่าของสมาชิกแต่ละตัวของอะเรย์ไปตามลำดับ โดยเริ่มจากตำแหน่งที่ 0 โดยปกติแล้วการหาค่าในอะเรย์จะให้ผลลัพธ์เป็นดัชนี (index) ของสมาชิกที่มีค่าที่เราต้องการหา (ตำแหน่งที่หาค่าเจอนั่นเอง) ดังตัวอย่างโค้ดต่อไปนี้

โค้ด:

```
1   int main()
2   {
3       int a[6] = {11, 55, 33, 77, 99, 77};
4       int target = 77;
```

```
5       for(int i = 0; i < 6; i++)
6       {
7           if (a[i] == target)
8           {
9               cout << "found at a[" << i << ']' << endl;
10              return 0;
11          }
12      }
13      cout<<"not found"<<endl;
14      return 0;
15  }
```

ในโค้ดนี้ เราต้องการหาค่า 77 ในอะเรย์ a ซึ่งมีอยู่ 2 ค่าที่ตำแหน่งที่ 3 และ 5 เราจะเห็นได้ว่ามีการวนซ้ำจาก i เป็น 0 ถึง 5 ซึ่ง i แทนตำแหน่งหรือค่าดัชนีของสมาชิกแต่ละตัว ในการหาค่านั้นโปรแกรมเปรียบเทียบค่าของสมาชิกในตำแหน่งปัจจุบัน (a[i]) กับค่าที่ต้องการหา (target) โดยเรียงตามลำดับในอะเรย์ ถ้าหากเจอครั้งแรกก็จะแสดงตำแหน่งที่เจอออกมาและจบการทำงาน หากไม่เจอก็จะแสดงข้อความ not found ออกมาแล้วจบการทำงาน

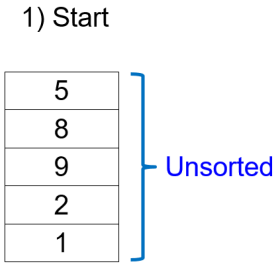
การค้นหาค่าแบบเส้นตรงมีข้อดีตรงที่ง่ายและตรงไปตรงมา แต่มีข้อเสียคืออาจจะใช้เวลานานมากในการหาข้อมูลที่เราต้องการหากข้อมูลนั้นอยู่ลึกไปในอะเรย์ ดังนั้นเราอาจจะใช้อัลกอริทึมในการค้นหาแบบอื่นที่มีประสิทธิภาพมากกว่า เช่น การค้นหาแบบไบนารี (binary search) ซึ่งจะหาข้อมูลจากอะเรย์ที่เรียงค่าสมาชิกไว้แล้วโดยเริ่มหาจากกึ่งกลางของอะเรย์ แล้วตัดพื้นที่ค้นหาออกทีละครึ่งหนึ่ง ทำให้การหาข้อมูลนั้นเร็วกว่าการค้นหาแบบเส้นตรงอยู่มาก แต่ในบทนี้เราจะไม่อธิบายเรื่องการค้นหาแบบไบนารี แต่เราจะอธิบายเรื่องการเรียงลำดับสมาชิกของอะเรย์ เพราะเราสามารถใช้อะเรย์ที่เรียงแล้วในการค้นหาแบบอื่นๆ นอกจากการค้นหาแบบไบนารีได้อีกด้วย

การเรียงข้อมูลแบบเลือกสรร (Selection Sort)

การเรียงข้อมูลแบบเลือกสรรนั้นสามารถทำได้ง่ายแต่อาจจะไม่มีประสิทธิภาพเท่ากับการเรียงข้อมูลแบบอื่น หลักการของการเรียงแบบนี้คือ ในแต่ละรอบของการเรียงเราจะต้องหาค่าที่น้อยที่สุด (ในกรณีที่เรียงข้อมูลจากน้อยไปหามาก) หรือมากที่สุด (ในกรณีที่เรียงข้อมูลจากมากไปหาน้อย) ในขอบเขตของการเรียงในรอบนั้นๆ แล้วเอาค่าที่ได้ไปไว้ที่ตำแหน่งแรกในขอบเขตของการเรียง (ในการอธิบายนี้เราจะเรียงข้อมูลจากน้อยไปหามาก) จากนั้นในรอบถัดไปให้เราทำเหมือนเดิมแต่ลดขอบเขตของการเรียงลง 1 ตัวนั่นคือตัดตัวที่

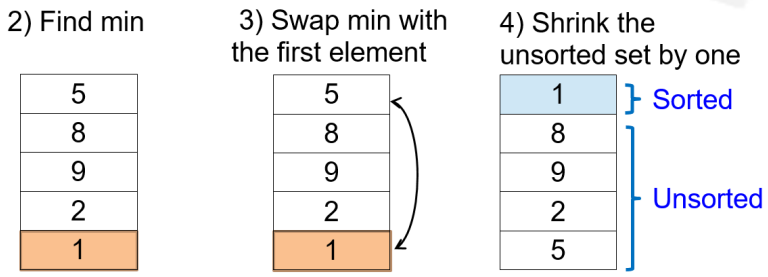
น้อยที่สุดหรือมากที่สุดที่เราเอาไปไว้ข้างหน้าแล้วออกไป ไม่ต้องนำมาคิดด้วยเพราะเราถือว่าเราเรียงค่านั้นไปแล้ว ให้เราวนทำซ้ำดังนี้ไปเรื่อยๆ จนเหลือแค่ค่าเดียวในขอบเขตของการเรียง การทำเช่นนี้จะทำให้ในแต่ละรอบเราจะได้ค่าที่น้อยที่สุดที่เหลืออยู่ไว้ด้านหน้าของอะเรย์ต่อกันไปเรื่อยๆ เมื่อทำครบทุกตัวเราก็จะได้อะเรย์ที่เรียงจากน้อยไปหามากตามที่เราต้องการ

เราจะอธิบายการเรียงอะเรย์แบบเลือกสรรโดยใช้ตัวอย่างดังต่อไปนี้ เริ่มด้วยอะเรย์ที่ยังไม่ได้เรียงข้างล่างนี้ ซึ่งมีสมาชิกเป็น 5 8 9 2 1 ตามลำดับ



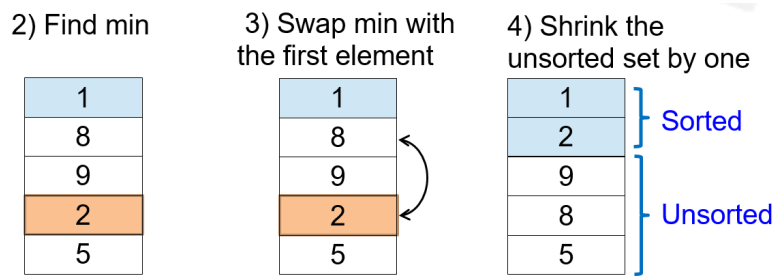
รูปที่ 83 อะเรย์เริ่มต้นที่ยังไม่ได้ถูกเรียง

เราต้องการเรียงอะเรย์ให้เป็น 1 2 5 8 9 ตามลำดับโดยใช้การเรียงข้อมูลแบบเลือกสรร ตอนนี้ขอบเขตของอะเรย์ที่เราต้องการเรียงเริ่มจาก 5 และสิ้นสุดที่ 1 มีสมาชิก 5 ตัว เราต้องหาค่าที่น้อยที่สุดในขอบเขตนี้ ซึ่งคือค่า 1 จากนั้นก็สลับค่า 1 กับค่าในตำแหน่งแรกของขอบเขตนี้ซึ่งคือค่า 5 แล้วเราก็จะได้ส่วนที่เรียงไว้แล้วเป็นค่า 1 ซึ่งเป็นส่วนที่เราจะไม่แตะในรอบถัดๆ ไป



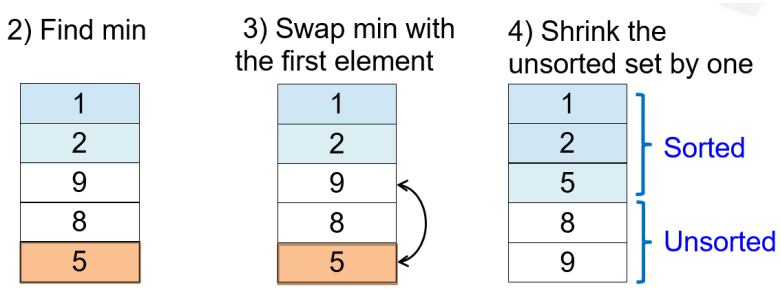
รูปที่ 84 รอบแรกของการเรียงอะเรย์

ในรอบต่อมา ขอบเขตของอะเรย์ที่เราต้องการเรียงเริ่มจาก 8 และสิ้นสุดที่ 5 มีสมาชิก 4 ตัว เราต้องหาค่าที่น้อยที่สุดในขอบเขตนี้ ซึ่งคือค่า 2 จากนั้นก็สลับค่า 2 กับค่าในตำแหน่งแรกของขอบเขตนี้ซึ่งคือค่า 8 แล้วเราก็จะได้ส่วนที่เรียงไว้แล้วเป็นค่า 1 2 ตามลำดับ ซึ่งเป็นส่วนที่เราจะไม่แตะในรอบถัดๆ ไป



รูปที่ 85 รอบที่ 2 ของการเรียงอะเรย์

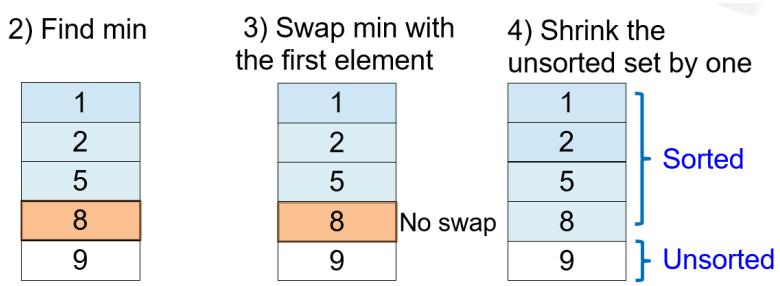
ในรอบที่ 3 นี้ ขอบเขตของอะเรย์ที่เราต้องการเรียงเริ่มจาก 9 และสิ้นสุดที่ 5 มีสมาชิก 3 ตัว เราต้องหาค่าที่น้อยที่สุดในขอบเขตนี้ ซึ่งคือค่า 5 จากนั้นก็สลับค่า 5 กับค่าในตำแหน่งแรกของขอบเขตนี้ซึ่งคือค่า 9 แล้วเราก็จะได้ส่วนที่เรียงไว้แล้วเป็นค่า 1 2 5 ตามลำดับ ซึ่งเป็นส่วนที่เราจะไม่แตะในรอบถัดๆ ไป



รูปที่ 86 รอบที่ 3 ของการเรียงอะเรย์

เราจะเห็นได้ว่าจริงๆ แล้วหลังจากขั้นตอนนี้อะเรย์นั้นถูกเรียงเรียบร้อยแล้ว แต่โปรแกรมจะไม่รู้ได้ว่าค่าที่เหลือคือค่าอะไรบ้างและถ้าเราใช้อะเรย์อื่นในรอบนี้ค่าอาจจะยังไม่ได้เรียงครบหมด โปรแกรมจึงต้องทำซ้ำไปอีกจนกว่าจะเหลือสมาชิกแค่ 1 ตัวในขอบเขตที่เราต้องการหา

ในรอบที่ 4 นี้ ขอบเขตของอะเรย์ที่เราต้องการเรียงเริ่มจาก 8 และสิ้นสุดที่ 9 มีสมาชิก 2 ตัว เราต้องหาค่าที่น้อยที่สุดในขอบเขตนี้ ซึ่งคือค่า 8 แล้วไม่ต้องสลับค่าใดๆ เพราะ 8 อยู่ในตำแหน่งแรกอยู่แล้ว เราก็จะได้ส่วนที่เรียงไว้แล้วเป็นค่า 1 2 5 8 ตามลำดับ รอบนี้จะเป็นรอบสุดท้าย เพราะว่าสมาชิกที่เหลือในขอบเขตที่เรายังไม่ได้เรียงมีแค่ 1 ตัวซึ่งควรจะเป็นตัวที่มีค่ามากที่สุด (ในที่นี้คือ 9) เราจึงสามารถหยุดการทำงานของโปรแกรมไว้เท่านี้ได้ และจะได้อะเรย์ที่เรียงค่าเป็นผลลัพธ์ออกมาคือ 1 2 5 8 9 ตามลำดับ



รูปที่ 87 รอบที่ 4 ของการเรียงอะเรย์

โค้ดที่ใช้ในการเรียงแบบเลือกสรรมีดังนี้ บรรทัดที่ 9-15 เป็นการหาค่าสมาชิกที่น้อยที่สุดในแต่ละรอบ บรรทัดที่ 17-22 เป็นการสลับค่าในตำแหน่งที่มีค่าน้อยที่สุดอยู่ (a[iMin]) กับตำแหน่งแรกของขอบเขตการค้นหา (a[j]) ถ้าหากมันไม่ใช่ตำแหน่งเดียวกัน ส่วนบรรทัดที่ 25-28 เป็นการแสดงค่าในอะเรย์ที่เรียงแล้วออกมา

โค้ด:

```
1   int main()
2   {
3       int n = 5;
4       int a[] = {5, 8, 9, 2, 1};
5       int i, j, iMin, temp;
6       for (j = 0; j < n-1; j++)
7       {
8           iMin = j;
9           for (i = j + 1; i < n; i++)
10          {
11              if (a[i] < a[iMin])
12              {
13                  iMin = i;
14              }
15          }//end loop i (find min)
16
17          if(iMin != j)
18          {
19              temp = a[j];
20              a[j] = a[iMin];
21              a[iMin] = temp;
22          }
23      }//end loop j
24
25      for (int i = 0; i < n; i++)
26      {
27          cout << a[i] << ' ';
28      }
29      return 0;
30  }
```

ตัวอย่างที่ 64 การหาค่ากลางของข้อมูล (median)
โจทย์ : จงเขียนโปรแกรมภาษาซีพลัสพลัสเพื่อรับจำนวนเต็ม N เข้ามาแล้วสร้างอะเรย์ขนาด N จากนั้นให้รับข้อมูลเป็นจำนวน N ตัว (ไม่จำเป็นต้องเป็นข้อมูที่ถูกเรียงไว้แล้ว) แล้วมาเก็บไว้ที่อะเรย์ที่สร้างขึ้นมา จากนั้นให้เรียงข้อมูลในอะเรย์ พิมพ์อะเรย์ที่เรียงแล้วออกมาพร้อมกับหาค่ากลางของข้อมูลในอะเรย์

1) เอาต์พุต คืออะเรย์ที่เรียงแล้ว และค่ากลางของข้อมูลในอะเรย์
2) อินพุต คือจำนวนเต็ม N และจำนวนเต็มที่ใส่เข้ามาจำนวน N ตัว
3) โจทย์กำหนดให้รับค่าเข้ามาใส่ในอะเรย์ก่อน แล้วจึงเรียงและพิมพ์ออก
4) เราทราบว่าค่ากลางของข้อมูลคือค่าที่อยู่ตรงกลางของกลุ่มเลขที่เรียงกันแล้ว ถ้าเลขกลุ่มนั้นมีจำนวนเป็นเลขคู่ ให้เอาค่ากลาง 2 ค่ามาหาค่าเฉลี่ย

ในโจทย์ข้อนี้เราไม่ทราบขนาดของอะเรย์ก่อนที่ผู้ใช้จะกำหนดค่า ดังนั้นเราไม่สามารถสร้างอะเรย์ก่อนรับค่า N เข้ามาได้ เมื่อรับค่า N เข้ามาแล้วเราจึงจะสร้างอะเรย์ขนาด N และวนรับข้อมูลเข้ามาจากผู้ใช้เป็นจำนวน N ครั้ง ก่อนที่จะใช้โค้ดการเรียงแบบเลือกสรรเพื่อเรียงข้อมูลในอะเรย์ ในการหาค่ากลางของสมาชิกในอะเรย์ เมื่อเราเรียงค่าในอะเรย์แล้ว ในกรณีที่ N เป็นจำนวนคี่ ค่ากลางก็จะอยู่ห้องตรงกลางหรือห้องที่เป็นครึ่งหนึ่งของขนาดของอะเรย์ (N) คือห้องเบอร์ N/2 เช่น สำหรับอะเรย์ที่มีขนาด 11 ห้อง ห้องตรงกลางควรจะเป็นตัวที่ 6 หรือห้องเบอร์ 5 ซึ่ง 11/2 = 5 (จำนวนเต็มหารด้วยจำนวนเต็มจะได้ผลลัพธ์เป็นจำนวนเต็ม) ส่วนในกรณีที่ N เป็นจำนวนคู่ เราต้องเอาสมาชิก 2 ตัวตรงกลางมาหาค่าเฉลี่ย ซึ่งเบอร์ห้องของสองค่าที่อยู่ตรงกลางจะเป็น N/2 - 1 และ N/2 เสมอ เช่น สำหรับอะเรย์ที่มีขนาด 10 ห้อง ห้องตรงกลางควรจะเป็นตัวที่ 5 กับตัวที่ 6 นั่นคือห้องเบอร์ 4 กับห้องเบอร์ 5 ซึ่งคือ 10/2 - 1 = 4 และ 10/2 = 5

โค้ด:

```
1   int main()
2   {
3       int n, median;
4       cout << "How many integers? ";
5       cin >> n;
6       int a[n];
7
8       // read array
9       cout << "Enter " << n << " integers: ";
10      for(int i = 0; i < n; i++)
11      {
12          cin >> a[i];
13      }
14
```

```
15      // sort array
16      int i, j, iMin, temp;
17      for (j = 0; j < n-1; j++)
18      {
19          iMin = j;
20          for (i = j+1; i < n; i++)
21          {
22              if (a[i] < a[iMin])
23              {
24                  iMin = i;
25              }
26          }//end loop i (find min)
27
28          if(iMin != j)
29          {
30              temp= a[j];
31              a[j] = a[iMin];
32              a[iMin] = temp;
33          }
34      }//end loop j
35
36      // find median!
37      if(n%2)
38      {
39          median = a[n/2];
40      }
41      else
42      {
43          median = (a[n/2 - 1] + a[n/2]) / 2.0;
44      }
45
46      // print the sorted array and median
47      for (int i = 0; i < n; i++)
48      {
49          cout << a[i] << ' ';
50      }
51      cout << endl;
52      cout << "Median = " << median << endl;
53      return 0;
54  }
```

ผลการทำงานเมื่อใส่ 3 98 45 21 3 8 4 5 13 87 ตามลำดับผ่านทางแป้นพิมพ์:

```
1  How many integers? 10
2  Enter 10 integers: 3 98 45 21 3 8 4 5 13 87
3  3 3 4 5 8 13 21 45 87 98
4  Median = 10
5
```

8.3 การใช้แถวลำดับกับฟังก์ชัน

การใช้แถวลำดับกับฟังก์ชันจะคล้ายกับการใช้ข้อมูลชนิดพื้นฐานกับฟังก์ชันแบบที่เราได้เรียนมาในเรื่องฟังก์ชัน นั่นคือ

1) เราสามารถใช้อะเรย์เป็นพารามิเตอร์เพื่อใช้ผ่านค่าเข้าฟังก์ชันได้ แต่อะเรย์ที่ถูกใช้เป็นพารามิเตอร์นั้นจะถูกผ่านแบบอ้างอิง (reference) เสมอ โดยไม่ต้องมีเครื่องหมาย & กำกับ เพราะจริงๆ แล้วค่าที่ถูกผ่านเข้าไปจะเป็นที่อยู่ของสมาชิกตัวแรกของอะเรย์อยู่แล้ว จึงไม่ต้องมี & เพื่อระบุว่าให้ผ่านที่อยู่ของตัวแปรเข้าไป ดังนั้นเมื่อผ่านอะเรย์เข้าไปในฟังก์ชัน อะเรย์นั้นจะสามารถถูกเปลี่ยนค่าได้ จึงต้องระวังเวลาใช้
2) เวลาเรียกฟังก์ชันที่มีอะเรย์เป็นพารามิเตอร์ เราเขียนแค่ชื่อของอะเรย์เท่านั้น ไม่ต้องมีเครื่องหมาย [ ] กำกับไปด้วย เช่น เราเรียกใช้ฟังก์ชัน print_array() โดยการผ่านอะเรย์ a เข้าไปได้ดังนี้

   ```
   print_array(a);
   ```

3) โดยปกติแล้วเวลาผ่านอะเรย์เข้าฟังก์ชัน เราควรเขียนขนาดของอะเรย์กำกับไว้ที่นิยามของฟังก์ชันและการประกาศฟังก์ชันด้วย แต่ว่าเราสามารถจะละขนาดของอะเรย์ในมิติแรกได้ เช่น เราสามารถเขียนนิยามของฟังก์ชันที่มีอะเรย์เป็นพารามิเตอร์ได้แบบใดก็ได้ต่อไปนี้

```
97) void print_array(int a[10]){ }
98) void print_array(int a[]){ }
99) หรือ
100)        void print_array(int b[10][5]){ }
101)        void print_array(int b[][5]){ }
```

4) ถ้าเราต้องการผ่านค่าสมาชิกตัวใดตัวหนึ่งของอะเรย์ เราทำการผ่านโดยใช้รูปแบบเหมือนการเข้าถึงสมาชิกตัวนั้นๆ เช่น a[2] หรือ b[10] ค่าที่ถูกส่งผ่านเข้าไปจะเป็นค่าของสมาชิกตัวนั้นๆ ไม่ใช่ทั้งอะเรย์ เช่นถ้าอะเรย์ a เป็นอะเรย์ของจำนวนเต็ม ค่า a[2] ที่ถูกส่งผ่านเข้าฟังก์ชันจะเป็นจำนวนเต็มหนึ่งค่าเท่านั้น

ตัวอย่างโค้ดข้างล่างนี้แสดงให้เห็นถึงการเรียกใช้ฟังก์ชันและการนิยามฟังก์ชัน 2 แบบ สำหรับการเรียกใช้ฟังก์ชันในบรรทัดที่ 11 ของโค้ดทั้งสองแบบนั้นเราจะเห็นได้ว่าเราส่งผ่านอะเรย์เข้าไปในฟังก์ชันโดยใช้การเรียกชื่อเท่านั้น โดยไม่มีเครื่องหมาย [ ] ซึ่งเหมือนกับการส่งผ่านตัวแปรทั่วไปเข้าไปในฟังก์ชัน ส่วนการนิยาม

ฟังก์ชัน เนื่องจากเราจะทำการส่งผ่านอะเรย์ 1 มิติ เราจึงสามารถละขนาดของอะเรย์ที่เป็นพารามิเตอร์ในนิยามฟังก์ชันได้ คือเราจะใช้ int a[] หรือ int a[3] ก็ได้

โค้ด 1:

```
1   void print(int a[], int n)
2   {
3       for (int i=0; i<n ;i++)
4           cout << a[i] << " ";
5       cout << endl;
6   }
7
8   int main( )
9   {
10      int a[]={1,2,3};
11      print(a,3);
12      return 0;
13  }
```

โค้ด 2:

```
1   void print(int a[3], int n)
2   {
3       for (int i=0; i<n ;i++)
4           cout << a[i] << " ";
5       cout << endl;
6   }
7
8   int main( )
9   {
10      int a[]={1,2,3};
11      print(a,3);
12      return 0;
13  }
```

ส่วนการประกาศฟังก์ชันที่มีอะเรย์เป็นพารามิเตอร์นั้น เราต้องเขียนชนิดของข้อมูลที่เป็นพารามิเตอร์ลงไปเช่นเดียวกับการผ่านข้อมูลพื้นฐานแบบอื่น นั่นคือสำหรับอะเรย์แล้วเราจำเป็นต้องใส่เครื่องหมาย [] เข้าไปหลังชนิดของอะเรย์เพื่อกำกับว่าค่าที่เราจะผ่านเข้าไปในฟังก์ชันเป็นอะเรย์นะ ไม่ใช่ตัวแปรพื้นฐานทั่วไป ดังตัวอย่างในโค้ดบรรทัดที่ 1 ข้างล่างนี้ นั่นคือเราใส่ int [ ] ตรงตำแหน่งที่จะผ่านอะเรย์เข้าไปในฟังก์ชัน

โค้ด:

```
int sum(int [], int);

int main() {
  int a[] = {11, 33, 55, 77};
  int size = sizeof(a)/sizeof(int);
  cout << "sum(a, size) = " << sum(a, size)
       << endl;
}

int sum(int a[], int n) {
  int sum = 0;
  for (int i = 0; i < n; i++)
    sum += a[i];
  return sum;
}
```

การใช้ฟังก์ชันสำหรับอ่านข้อมูลและเขียนข้อมูลสมาชิกของอะเรย์ (I/O Functions for Array)

เนื่องจากโค้ดที่ใช้ในการอ่านข้อมูลและเขียนข้อมูลให้กับสมาชิกของอะเรย์นั้นมีรูปแบบที่สามารถนำมาใช้ซ้ำได้และส่วนใหญ่เป็นการวนลูปซึ่งต้องมีหลายบรรทัด การแยกการอ่านข้อมูลและเขียนข้อมูลสมาชิกของอะเรย์ออกเป็นฟังก์ชันนั้นก็จะทำให้เขียนโปรแกรมได้ง่ายขึ้น

โค้ด:

```
const int MAXSIZE = 100;
void read(int[], int&);
void print(int[], int);

int main()
{
    int a[MAXSIZE] = {0}, size;
    read(a, size);
    cout << "The array has " << size << " elements: ";

    print(a, size);
    return 0;
}

void read(int a[], int& n)
{
    cout << "Enter integers. Terminate with 0:\n";
    n = 0;
    do
```

```
20      {
21          cout << "a[" << n << "]: ";
22          cin >> a[n];
23      }
24      while (a[n++] != 0 && n < MAXSIZE);
25      --n;
26  }
27
28  void print(int a[], int n)
29  {
30      for (int i = 0; i < n; i++)
31      {
32        cout << a[i] << " ";
33      }
34      cout << endl;
35  }
```

โค้ดข้างบนนี้แสดงให้เห็นถึงการใช้ฟังก์ชันเพื่ออ่านและเขียนอะเรย์ ฟังก์ชัน read() เป็นฟังก์ชันที่เอาไว้รับค่าสมาชิกของอะเรย์ โดยมีพารามิเตอร์ 2 ตัว คืออะเรย์ชนิดจำนวนเต็มและจำนวนเต็มที่ถูกผ่านแบบอ้างอิง จำนวนเต็ม n นี้คือจำนวนสมาชิกของอะเรย์ที่ผู้ใช้ใส่เข้ามา (สูงสุดคือ 100 ตัว) โค้ดนี้ไม่ได้กำหนดค่า n แบบตายตัวตั้งแต่แรก แต่ให้โปรแกรมนับว่าผู้ใช้ใส่จำนวนเต็มในการวนรับค่าเข้ามากี่ตัว ในแต่ละรอบของการวนซ้ำจำนวนเต็มที่ผู้ใช้ใส่เข้ามาก็จะถูกเก็บไว้ในห้องแถว (อะเรย์) เรียงกันจาก 0 ถึง n-1 ตามลำดับ

ฟังก์ชัน print() เป็นฟังก์ชันที่เอาไว้แสดงค่าของอะเรย์ ซึ่งจะถูกเรียกใช้หลังจากฟังก์ชัน read() ฟังก์ชัน print() มีพารามิเตอร์ 2 ตัวคืออะเรย์ชนิดจำนวนเต็มและจำนวนเต็มที่ถูกผ่านแบบใช้ค่า (by value) ในฟังก์ชัน print() เราต้องการจำนวนเต็ม n เพื่อมาวนลูปแสดงค่าสมาชิกเท่านั้น โดยไม่ได้มีการเปลี่ยนแปลงค่า n ดังนั้นเราจึงไม่ผ่านค่า n เข้ามาแบบอ้างอิง สังเกตว่าฟังก์ชัน print() นี้เป็นมาตรฐานการพิมพ์อะเรย์แบบเรียงลำดับจากหน้าไปหลัง เราสามารถใช้รูปแบบโค้ดนี้ในโปรแกรมอื่นๆ ได้ด้วย

8.4 แถวลำดับหลายมิติ (Multidimensional Array)

จากหัวข้อที่ผ่านมา เราจะเห็นได้ว่าชนิดของข้อมูลในอะเรย์นั้นสามารถเป็นข้อมูลพื้นฐานชนิดใดก็ได้ ซึ่งรวมทั้งชนิดของข้อมูลที่เป็นอะเรย์ด้วย นั่นคือ แทนที่สมาชิกของอะเรย์ห้องหนึ่งจะเป็นแค่ข้อมูลพื้นฐาน เช่น จำนวนเต็ม หรือจำนวนจริง สมาชิกของอะเรย์ห้องหนึ่งสามารถเป็นอะเรย์ได้ด้วย! เราเรียกอะเรย์ที่มีสมาชิกเป็นอะเรย์ว่าเป็น แถวลำดับหลายมิติ หรืออะเรย์หลายมิติ เช่นถ้าสมาชิกของอะเรย์มีชนิดของข้อมูลเป็นอะเรย์ 1 มิติ อะเรย์นั้นจะเป็นอะเรย์ 2 มิติ (แต่ละห้องมีอะเรย์ 1 มิติถูกเก็บอยู่) แต่ถ้าสมาชิกของอะเรย์มีชนิดของข้อมูลเป็นอะเรย์ 2 มิติ อะเรย์นั้นจะเป็นอะเรย์ 3 มิติ (แต่ละห้องมีอะเรย์ 2 มิติถูกเก็บอยู่)

8.4.1 แถวลำดับ 2 มิติ (2D Array)

ในการทำงานกับอะเรย์ 1 มิติ เรามองมันเป็นห้องแถวที่เรียงกันอยู่ ส่วนในการทำงานกับอะเรย์ 2 มิติ ก็จะคล้ายๆ กัน แต่เราจะมองมันเป็นตารางแทน ซึ่งจะมีเบอร์แถวและเบอร์หลัก (คอลัมน์) แทนเบอร์ห้องที่เรียงกันเป็นเส้นตรง ดังแสดงในรูปที่ 88

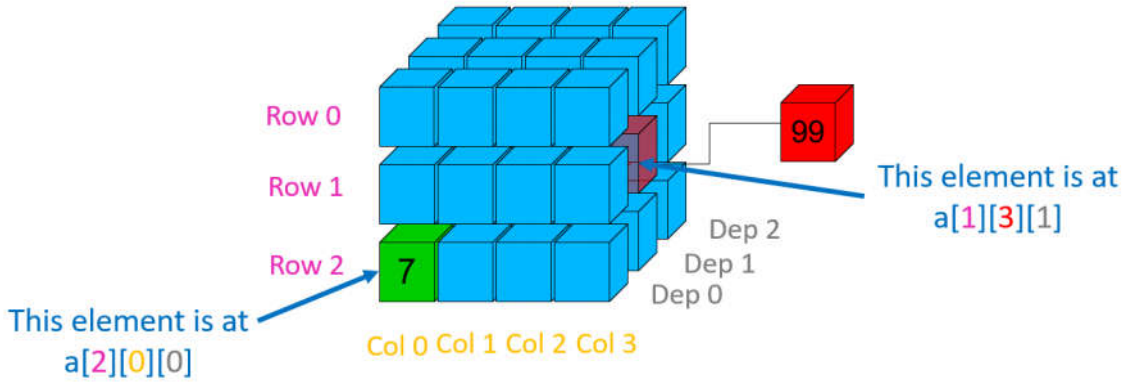
	Col 0	Col 1	Col 2	Col 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

รูปที่ 88 ลักษณะของอะเรย์ 2 มิติ

การเข้าถึงอะเรย์แบบ 2 มิติ เราจะเขียนคล้ายๆ อะเรย์ 1 มิติ แต่เราต้องเพิ่มอีก “มิติ” เข้าไปในการเขียนด้วย นั่นคือการใส่เครื่องหมาย [ ] เข้าไปอีกคู่หนึ่ง โดยตัวเลขที่อยู่ในวงเล็บสี่เหลี่ยมคู่แรกจะหมายถึง **แถว** ของอะเรย์เสมอ ส่วนตัวเลขที่อยู่ในวงเล็บสี่เหลี่ยมคู่หลังจะหมายถึง **หลัก** ของอะเรย์เสมอ เช่น ถ้าเราเขียนว่า a[3][2] จะหมายถึงสมาชิกของอะเรย์ 2 มิติในแถวที่ 3 (นับจาก 0) หลักที่ 2 (นับจาก 0 เช่นกัน)

8.4.2 แถวลำดับ 3 มิติ (3D Array)

ในการทำงานกับอะเรย์ 3 มิติ เรามองตัวอะเรย์เป็นกล่องหรือคิวบิก (cubic) นั่นคือนอกจากจะมีแถวและหลักในอะเรย์แล้ว เรายังต้องมีความลึกของอะเรย์เพิ่มเข้ามาอีกด้วย ดังแสดงในรูปที่ 89



รูปที่ 89 ลักษณะของอะเรย์ 3 มิติ

ในการเข้าถึงอะเรย์ 3 มิติ เราจะต้องเพิ่ม “มิติ” นั่นคือเพิ่มความลึกเข้าไปยังการเขียนอะเรย์ 2 มิติ นั่นคือเพิ่มเครื่องหมาย [ ] เข้าไปอีก 1 คู่ ซึ่งจะทำให้มีเครื่องหมาย [ ] จำนวน 3 คู่ โดยตัวเลขที่อยู่ในวงเล็บสี่เหลี่ยมคู่แรกจะหมายถึง **แถว** ของอะเรย์เสมอ ส่วนตัวเลขที่อยู่ในวงเล็บสี่เหลี่ยมคู่กลางจะหมายถึง **หลัก** ของอะเรย์เสมอ และตัวเลขที่อยู่ในวงเล็บสี่เหลี่ยมคู่หลังสุดจะหมายถึง **ความลึก** ของอะเรย์เสมอ เช่น ถ้าเราเขียนว่า a[1][3][2] เราหมายถึงสมาชิกที่อยู่ในอะเรย์ 3 มิติในแถวที่ 1 หลักที่ 3 และความลึกที่ 2 โดยเลขดัชนีทั้งหมดเริ่มจาก 0 เช่นเดียวกัน

8.4.2 การทำงานกับแถวลำดับหลายมิติ

เราจะสังเกตได้ว่าจำนวนของมิติของอะเรย์กับจำนวนของคู่วงเล็บสี่เหลี่ยม [ ] ในการเขียนถึงอะเรย์นั้นจะเท่ากัน นั่นคือว่าถ้าเราต้องการเข้าถึงอะเรย์ 1 มิติ เราต้องมีวงเล็บสี่เหลี่ยมหลังชื่ออะเรย์จำนวน 1 คู่ เช่นเดียวกันกับถ้าเราต้องการเข้าถึงอะเรย์ 2 มิติ เราต้องมีวงเล็บสี่เหลี่ยมหลังชื่ออะเรย์จำนวน 2 คู่ และถ้าเราต้องการเข้าถึงอะเรย์ 3 มิติ เราต้องมีวงเล็บสี่เหลี่ยมหลังชื่ออะเรย์จำนวน 3 คู่ ตารางที่ 13 แสดงรูปแบบการประกาศและการให้ค่าอะเรย์ในมิติต่างๆ ส่วนตารางที่ 14 แสดงการให้ค่า การพิมพ์ค่าและการรับค่าของสมาชิกอะเรย์ในมิติต่างๆ ในตารางที่ 13 ให้สังเกตการให้ค่าเริ่มต้นของอะเรย์ การให้ค่าเริ่มต้นของอะเรย์ 1 มิตินั้นเราจะใช้วงเล็บปีกกา ({ }) คู่เดียวหรือกลุ่มเดียว โดยสมาชิกแต่ละตัวจะอยู่ในวงเล็บคู่นี้ และคั่นด้วยเครื่องหมายลูกน้ำ (,) ส่วนการให้ค่าเริ่มต้นของอะเรย์ 2 มิตินั้นเราจะใช้วงเล็บปีกกา ({ }) 2 ระดับชั้น ให้จำง่ายๆ ว่า วงเล็บปีกกาชั้นนอกสุดเอาไว้สำหรับครอบอะเรย์ทั้งก้อน ส่วนวงเล็บกลุ่มชั้นในนั้น เอาไว้สำหรับครอบกลุ่มก้อนของสมาชิกในแต่ละแถว (ถ้าอะเรย์มี 3 แถวก็จะมีก้อน { } 3 ก้อนอยู่ข้างใน) ซึ่งแต่ละก้อนหรือแต่ละแถวเราจะใช้เครื่องหมายลูกน้ำ (,) คั่นระหว่างก้อน และใน 1 ก้อนจะมีสมาชิกของอะเรย์ในแถวนั้นๆ บรรจุอยู่ โดยเริ่มจากสมาชิกในหลักที่ 0, 1, 2 ไปเรื่อยๆ จนครบจำนวนหลัก

ตารางที่ 13 การประกาศและให้ค่าตั้งต้นกับอะเรย์ในมิติต่างๆ

มิติ	การประกาศและให้ค่าตั้งต้นกับอะเรย์

1D	`int a[size];`
	`int a[4];`
	`int a[] = {1,2,3,4};`

2D	`int a[row_size][col_size];`
	`int a[2][3];`
	`int a[2][3] =`
	`{`
	`  {1,2,3},`
	`  {4,5,6}`
	`};`

3D	`int a[row_size][col_size][dep_size];`
	`int a[2][3][4];`
	`int a[2][3][4] =`
	`{`
	`{ {1,2,3,4}, {1,2,3,4}, {9,10,11,12} },`
	`{ {13,14,15,16}, {17,18,19,20}, {21,22,23,24} }`
	`};`

และด้วยการเขียนคล้ายๆ กัน การให้ค่าเริ่มต้นของอะเรย์ 3 มิตินั้นเราจะใช้วงเล็บปีกกา ({ }) 3 ระดับชั้น ให้จำง่ายๆ เช่นกันว่า วงเล็บปีกกาชั้นนอกสุดเอาไว้สำหรับครอบอะเรย์ทั้งก้อน ส่วนชั้นต่อมาคือชั้นกลางจะเอาไว้สำหรับครอบกลุ่มสมาชิกในหนึ่งแถว และกลุ่มชั้นในสุดจะเอาไว้สำหรับครอบกลุ่มสมาชิกในหนึ่งหลัก จากตัวอย่างในตารางที่ 13 อะเรย์ 3 มิติมี 2 แถว จึงมีก้อนวงเล็บปีกกาชั้นกลางจำนวน 2 ก้อน (ได้เขียนแยกบรรทัดไว้ให้แล้ว) แล้วในแต่ละแถว ก็จะมีกลุ่มก้อนที่ครอบด้วยวงเล็บปีกกาอีกชั้นหนึ่งอีก ซึ่งหมายถึงสมาชิกของอะเรย์ในแต่ละหลัก เช่นถ้าอะเรย์มี 3 หลักก็จะมีก้อนวงเล็บปีกกาชั้นในสุดนี้อยู่ 3 ก้อนในแต่ละแถว เรียงตามหลักที่ 0, 1, 2 ตามลำดับ ส่วนสมาชิกที่อยู่ในก้อนปีกกาในสุดนี้คือสมาชิกในแต่ละความลึกของอะเรย์ ถ้าอะเรย์มีความลึกเท่ากับ 4 เราก็จะมีสมาชิกจำนวน 4 ตัวอยู่ในก้อนในสุดนี้ เป็นต้น

ตารางที่ 14 การให้ค่า การพิมพ์ค่าและการรับค่าของสมาชิกอะเรย์ในมิติต่างๆ

มิติ	การให้ค่า การพิมพ์ค่าและการรับค่าของสมาชิกอะเรย์

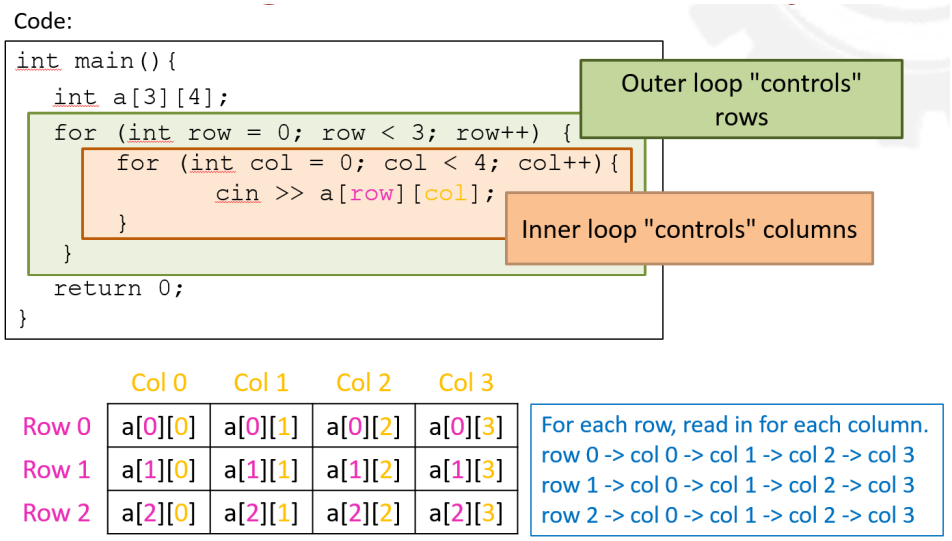
1D	`a[1] = 6;`
	`cout << a[1];`
	`cin >> a[1];`

2D	`a[1][2] = 6;`
	`cout << a[1][2];`
	`cin >> a[1][2];`

3D	`a[1][2][3] = 6;`
	`cout << a[1][2][3];`
	`cin >> a[1][2][3];`

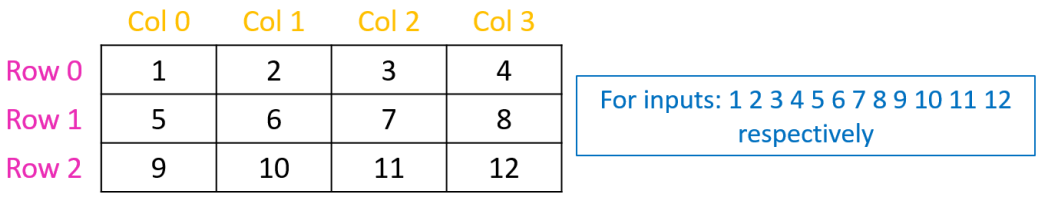
โดยทั่วไปแล้วการทำงานกับอะเรย์ เช่น การอ่านและพิมพ์สมาชิกของอะเรย์ทุกตัวเรียงตามลำดับ เราจะใช้จำนวนชั้นของลูปตามจำนวนมิติ นั่นคือถ้าเราต้องการอ่านและพิมพ์สมาชิกของอะเรย์ 1 มิติเรียง

ตามลำดับ เราต้องใช้ลูป 1 ชั้นในการทำงานนี้ ถ้าเราต้องการอ่านและพิมพ์สมาชิกของอะเรย์ 2 มิติเรียงตามลำดับ เราต้องใช้ลูป 2 ชั้นในการทำงานนี้ และถ้าเราต้องการอ่านและพิมพ์สมาชิกของอะเรย์ 3 มิติเรียงตามลำดับ เราต้องใช้ลูป 3 ชั้นในการทำงานนี้ ในการวนลูป 2 และ 3 ชั้นเราจะต้องกำหนดว่าลูปชั้นนอกจะควบคุมการเข้าถึงสมาชิกในแถวหรือหลัก (หรือความลึก) ซึ่งโดยปกติแล้วลูปชั้นนอกจะควบคุมการเข้าถึงสมาชิกในแถวก่อน แล้วลูปชั้นต่อมาจะควบคุมหลักและความลึกตามลำดับ ดังแสดงในรูปที่ 90 นั่นคือเริ่มเข้าถึงจากแถวที่ 0 หลักที่ 0 ก่อน จากนั้นเข้าถึง หลักที่ 1 (ยังคงเป็นแถวที่ 0 อยู่), 2, 3, 4 ไปเรื่อยๆ จนหมดหลักในแถวนั้น แล้วจึงย้ายไปทำงานในแถวที่ 1 หลักที่ 0, 1, 2 ไปเรื่อยๆ จนหมดหลักในแถวนั้น แล้วเลื่อนแถวไปเรื่อยๆ จนหมดทุกแถวในอะเรย์



รูปที่ 90 โค้ดการใส่ข้อมูลเข้าอะเรย์ 2 มิติโดยใช้โค้ดที่เข้าถึงอะเรย์แบบเรียงแถว

ตัวอย่างของการใส่ข้อมูลเข้าอะเรย์ 2 มิติโดยใช้โค้ดที่เข้าถึงอะเรย์แบบเรียงแถว (โค้ดเดียวกับในรูปที่ 90) แสดงในรูปที่ 91 เมื่อเราใส่อินพุตเป็น 1-12 ตามลำดับ เราจะได้ค่าที่ถูกใส่เข้ามาเอาไว้ในห้องของแถวแรกให้ครบทุกหลักโดยเริ่มจากหลักที่ 0 ก่อน จากนั้นถึงเอาค่าไปเป็นสมาชิกของแถวที่ 1 และ 2 ตามลำดับ



รูปที่ 91 การใส่ข้อมูลเข้าอะเรย์ 2 มิติโดยใช้โค้ดที่เข้าถึงอะเรย์แบบเรียงแถว

จริงๆ แล้วก็มีบางกรณีเช่นกันที่เราต้องการให้โปรแกรมของเราเข้าถึงอะเรย์แบบเรียงหลักก่อน ดังที่เราจะได้เห็นในตัวอย่างต่อไป การเข้าถึงสมาชิกของแบบเรียงหลักก่อนนั้นเราจะต้องกำหนดให้ลูปชั้นนอกสุดควบคุมหลัก ไม่ใช่ควบคุมแถว ดังแสดงในรูปที่ 92 การเข้าถึงอะเรย์ก็จะเริ่มจากหลักที่ 0 แถวที่ 0 ก่อน จากนั้นเลื่อนไปเป็นแถวที่ 1 (ยังหลักที่ 0 เหมือนเดิม) , 2, 3, 4 จนหมดทุกแถวในหลักนั้น แล้วจึงย้ายไปทำงานที่หลักที่ 1 ในแถวที่ 0, 1, 2 ไปเรื่อยๆ จนหมดทุกแถวในหลักนั้น แล้วย้ายไปทำหลักที่ 2, 3, 4 ในลักษณะเดียวกันจนครบทุกหลักในอะเรย์นั้น

Code:

```
int main(){
   int a[3][4];
   for (int col = 0; col < 3; col++) {
      for (int row = 0; row < 4; row++){
            cin >> a[row][col];
      }
   }
   return 0;
}
```

Outer loop "controls" columns

Inner loop "controls" rows

	Col 0	Col 1	Col 2	Col 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

For each col, read in for each row.
col 0 -> row 0 -> row 1 -> row 2
col 1 -> row 0 -> row 1 -> row 2
col 2 -> row 0 -> row 1 -> row 2
col 3 -> row 0 -> row 1 -> row 2

รูปที่ 92 โค้ดการใส่ข้อมูลเข้าอะเรย์ 2 มิติโดยใช้โค้ดที่เข้าถึงอะเรย์แบบเรียงหลัก

ตัวอย่างของการใส่ข้อมูลเข้าอะเรย์ 2 มิติโดยใช้โค้ดที่เข้าถึงอะเรย์แบบเรียงหลัก (โค้ดเดียวกับในรูปที่ 92) แสดงในรูปที่ 93 เมื่อเราใส่อินพุตเป็น 1-12 ตามลำดับ เราจะได้ค่าที่ถูกใส่เข้ามาเอาไว้ในห้องของหลักแรกให้ครบหมดทุกแถวก่อน จากนั้นถึงเอาค่าไปเป็นสมาชิกของหลักที่ 1, 2 และ 3 ตามลำดับ

	Col 0	Col 1	Col 2	Col 3
Row 0	1	4	7	10
Row 1	2	5	8	11
Row 2	3	6	9	12

For inputs: 1 2 3 4 5 6 7 8 9 10 11 12 respectively

รูปที่ 93 การใส่ข้อมูลเข้าอะเรย์ 2 มิติโดยใช้โค้ดที่เข้าถึงอะเรย์แบบเรียงหลัก

ตัวอย่างที่ 65

โจทย์: จงหาผลการทำงานของโค้ดต่อไปนี้ เมื่อใส่ค่าต่อไปนี้ตามลำดับผ่านทางแป้นพิมพ์

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

โค้ด:

```
1   int main()
2   {
3       int a[3][5];
4       cout << "Enter 15 integers, 5 per row:\n";
5       for (int i = 0; i < 3; i++)
6       {
7           cout << "Row " << i << ": ";
8           for (int j = 0; j < 5; j++)
9           {
10              cin >> a[i][j];
11          }
12      }
13      cout << "Print array 3x5" << endl;
14      for (int i = 0; i < 3; i++)
15      {
16          for (int j = 0; j < 5; j++)
17          {
18              cout << " " << a[i][j];
19          }
20          cout << endl;
21      }
22      return 0;
23  }
```

ในโค้ดนี้มีอะเรย์หนึ่งตัวซึ่งเป็น 2 มิติ มีขนาด 3 แถว แถวละ 5 หลัก (หรือเขียนว่าอะเรย์ขนาด 3x5) โค้ดในบรรทัดที่ 4-12 เป็นโค้ดสำหรับรับค่าเข้ามาในอะเรย์ เราสังเกตได้ว่ามีคำสั่ง cin อยู่ในลูป 2 ชั้น โดยลูปชั้นนอกควบคุมแถว ซึ่งสังเกตได้จาก i < 3 และลูปชั้นในควบคุมหลัก ซึ่งสังเกตได้จาก j < 5 เพราะฉะนั้นโค้ดในส่วนนี้เป็นการกรอกข้อมูลลงอะเรย์แบบเรียงตามแถว นั่นคือใส่ข้อมูลในแถวแรกให้ครบทุกหลักก่อน แล้วจึงย้ายไปใส่ข้อมูลในแถวถัดๆ ไป และจะทำให้ได้อะเรย์ที่มีหน้าตาเช่นนี้

15	14	13	12	11
10	9	8	7	6
5	4	3	2	1

โค้ดในบรรทัดที่ 13-21 เป็นโค้ดสำหรับแสดงผลของสมาชิกในอะเรย์ออกทางหน้าจอ ซึ่งเราสังเกตได้จากคำสั่ง cout ในลูป 2 ชั้น เราจะสังเกตได้ว่าลูป 2 ชั้นนี้มีลักษณะเหมือนลูปของการรับค่าเข้ามาในอะเรย์เลย นั่นคือการแสดงค่าก็จะแสดงออกในลำดับเดียวกัน คือแสดงค่าของสมาชิกในแต่ละแถวให้ครบทุกหลักก่อน แล้วจึงย้ายไปแสดงค่าของสมาชิกในแถวถัดๆ ไป ผลการทำงานทั้งหมดของโปรแกรมนี้แสดงได้ดังข้างล่างนี้

ผลการทำงาน:

```
Enter 15 integers, 5 per row:
Row 0: 15 14 13 12 11
Row 1: 10 9 8 7 6
Row 2: 5 4 3 2 1
Print array 3x5
 15 14 13 12 11
 10 9 8 7 6
 5 4 3 2 1
```

ตัวอย่างที่ 66 การคำนวณคะแนนเฉลี่ยของนักศึกษา

โจทย์ : จงเขียนโปรแกรมภาษาซีพลัสพลัสโดยใช้ฟังก์ชันเพื่อ 1) รับค่าคะแนนสอบของนักเรียน 3 คน ซึ่งแต่ละคนมีคะแนนสอบ 5 ครั้งเข้ามาเก็บไว้ใน “ฐานข้อมูล” ดังแสดงตัวอย่างในรูปที่ 94 2) หาค่าเฉลี่ยของคะแนนของนักเรียนแต่ละคนสำหรับการสอบทั้งหมด และ 3) หาค่าเฉลี่ยของคะแนนของนักเรียนในชั้นเรียนของการสอบแต่ละครั้ง

S\Q	q1	q2	q3	q4	q5
s1	7	7	9	8	5
s2	7	6	5	4	3
s3	7	10	10	8	9

รูปที่ 94 ตัวอย่างฐานข้อมูลของคะแนนเก็บของนักเรียน

โดยให้ใช้ฟังก์ชันและตัวแปรต่างๆ ตามที่กำหนดให้ดังนี้

```
const int NUM_STUDENTS = 3;
const int NUM_QUIZZES = 5;
```

```
4  typedef int Score[NUM_STUDENTS][NUM_QUIZZES];
5
6  void read(Score);
7  void printQuizAverages(Score);
8  void printClassAverages(Score);
```

1) เอาต์พุต มี 2 อย่างคือค่าเฉลี่ยของคะแนนของนักเรียนแต่ละคนสำหรับการสอบทั้งหมด และค่าเฉลี่ยของคะแนนของนักเรียนในชั้นเรียนของการสอบแต่ละครั้ง
2) อินพุต คือค่าคะแนนสอบของนักเรียน 3 คน ซึ่งแต่ละคนมีคะแนนสอบ 5 ครั้ง
3) โจทย์กำหนดให้ใช้ตัวแปรโกลบอล 2 ตัว เป็นจำนวนเต็มที่เป็นค่าคงที่สำหรับจำนวนนักเรียน 3 คน และจำนวนการสอบ 5 ครั้ง โจทย์ยังกำหนด typedef คือชนิดของข้อมูลที่ผู้เขียนโปรแกรมสามารถกำหนดขึ้นมาเอง ชื่อว่า Score มีชนิดพื้นฐานเป็นอะเรย์ของจำนวนเต็มขนาด 3 แถว 5 หลัก และโจทย์กำหนดการประกาศฟังก์ชันเพื่ออ่านและหาค่าเฉลี่ยที่โจทย์ถามอีกด้วย
4) ไม่มีข้อมูลอะไรเพิ่มเติม

จากข้อมูลที่โจทย์กำหนดให้ในข้อนี้ เราสามารถเขียนโปรแกรมได้หลายแบบ สิ่งที่จะอธิบายต่อไปนี้เป็นเพียงแบบหนึ่งเท่านั้น เราจะเห็นได้ว่าเราจะต้องทำงาน 3 อย่างหลักๆ คือ 1) อ่านข้อมูลคะแนนจากผู้ใช้ 2) หาค่าเฉลี่ยของคะแนนของนักศึกษาแต่ละคน และ 3) หาค่าเฉลี่ยของคะแนนในแต่ละการสอบ ซึ่งโจทย์กำหนดให้แยกฟังก์ชันออกมาอย่างชัดเจนแล้ว ในแต่ละฟังก์ชันโจทย์กำหนดให้เป็นฟังก์ชันชนิด void ทั้งหมด นั่นคือจะไม่มีการคืนค่ากลับมาที่ฟังก์ชันหลัก (main) เพราะฉะนั้นเราสามารถบรรจุโค้ดทุกอย่างที่เกี่ยวกับการทำงานนั้นๆ ในฟังก์ชันที่เกี่ยวข้องเลยก็ได้

ก่อนอื่นจะขออธิบายเกี่ยวกับคำสั่ง typedef เพิ่มเติม คำสั่ง typedef คือคำสั่งที่ใช้ประกาศชนิดของข้อมูลที่ผู้เขียนโปรแกรมสามารถกำหนดขึ้นมาเอง นั่นคือ เราใช้ typedef ในการประกาศว่า ต่อไปนี้ฉันจะใช้ชื่อนี้ ในการเรียกชนิดข้อมูลต่อไปนี้นะ เช่นในโจทย์ข้อนี้ เราสามารถใช้ชื่อ Score ในการเรียกแทนข้อมูลชนิดอะเรย์ที่เป็นจำนวนเต็มที่มีขนาด 3x5 ดังนั้นแทนที่เราจะประกาศตัวแปรว่า

```
int myArray[NUM_STUDENTS][NUM_QUIZZES]
```

เราสามารถประกาศได้หลังจากใช้คำสั่ง typedef เช่นในบรรทัดที่ 4 ของโค้ดว่า

```
Score myArray;
```

ซึ่งจะทำให้การเขียนโปรแกรมที่ต้องใช้ข้อมูลชนิดนี้เป็นจำนวนมากนั้นทำให้ง่ายขึ้นและสั้นลง

เราจะอธิบายการเขียนโค้ดไปทีละฟังก์ชัน ฟังก์ชันแรก read() เป็นฟังก์ชันที่รับข้อมูลคะแนนของนักเรียนเข้ามาจากผู้ใช้ เราผ่านตัวแปรชนิด Score (ซึ่งจริงๆ เป็นอะเรย์) เข้าไปเพื่อรับข้อมูลลงอะเรย์ ขอทบทวนอีกทีว่าการผ่านข้อมูลชนิดอะเรย์เข้าไปในฟังก์ชันนั้นจะเป็นการผ่านแบบอ้างอิง (by reference) เสมอ จึงไม่ต้องใส่เครื่องหมาย & ในการนิยามฟังก์ชัน ฟังก์ชัน read() จะทำการวนลูป 2 ชั้นเพื่อรับค่าคะแนนของนักเรียนโดยเรียงตามคนก่อน นั่นคือจะกรอกคะแนนของนักเรียนคนแรกให้ครบทุกการสอบ ซึ่งมี 5 การสอบก่อน แล้วจึงเลื่อนไปกรอกคะแนนของคนถัดไป ดังนั้นลูปชั้นนอกจะต้องเป็นการควบคุมจำนวนนักเรียน (s ตั้งแต่ 0-2) และลูปชั้นในจะเป็นการควบคุมจำนวนการสอบ (q ตั้งแต่ 0-4)

เมื่อเราได้อะเรย์ที่กรอกคะแนนครบมาแล้ว เราก็ผ่านเข้าไปในฟังก์ชัน printQuizAverages() เพื่อหาคะแนนเฉลี่ยในการสอบของนักเรียนแต่ละคน นั่นคือเราเอาคะแนนสอบทั้ง 5 การสอบของนักเรียนคนหนึ่งมาหาค่าเฉลี่ย เพราะฉะนั้นในการวนลูปเราจะต้องวนลูปเพื่อรวมคะแนนของนักเรียนทีละคนก่อน (เรียงตามคน) ดังนั้นลักษณะการวนลูปจึงเหมือนการวนลูปเพื่อรับคะแนนของนักเรียนเข้ามาจากผู้ใช้ นั่นคือลูปชั้นนอกควบคุมการเข้าถึงนักเรียนแต่ละคน สำหรับแต่ละคนจะมีลูปชั้นในควบคุมการเข้าถึงคะแนนสอบแต่ละครั้ง ซึ่งหลังจากรวมคะแนนจากทั้ง 5 การสอบของนักเรียนแต่ละคนแล้วก็จะหาค่าเฉลี่ยของคะแนนทั้ง 5 นั้นสำหรับนักเรียนคนนั้น และแสดงค่าเฉลี่ยออกทางหน้าจอ แล้วย้ายไปทำอย่างเดียวกันกับนักเรียนคนถัดไป

ฟังก์ชันสุดท้ายที่โปรแกรมนี้จะทำคือฟังก์ชัน printClassAverages() ซึ่งจะรับอะเรย์คะแนนของนักเรียนเข้าไปแล้วทำการหาค่าเฉลี่ยของคะแนนของนักเรียนในชั้นเรียนสำหรับการสอบแต่ละครั้ง นั่นคือเราจะต้องวนลูปเรียงตามการสอบแต่ละครั้งก่อน ไม่ใช่เรียงตามนักเรียนแต่ละคนก่อนเหมือนที่ผ่านมา ดังนั้นในการวนลูปเราจะต้องเอาการควบคุมการสอบแต่ละครั้ง (q) มาไว้ที่ลูปชั้นนอก แล้วในแต่ละรอบของการลูปชั้นนอก (การสอบแต่ละครั้ง) โปรแกรมก็จะรวมคะแนนของนักเรียนทุกคนในการสอบครั้งนั้น แล้วคำนวณและแสดงค่าเฉลี่ยของการสอบครั้งนั้นออกทางหน้าจอ แล้วย้ายไปทำอย่างเดียวกันกับการสอบครั้งถัดไป

โค้ด:

```
1   const int NUM_STUDENTS = 3;
2   const int NUM_QUIZZES = 5;
3
4   typedef int Score[NUM_STUDENTS][NUM_QUIZZES];
5
6   void read(Score);
7   void printQuizAverages(Score);
8   void printClassAverages(Score);
9
10  int main()
```

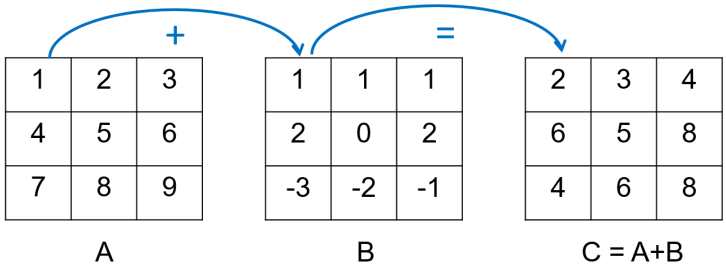
```
11 {
12     Score score;
13
14     read(score);
15
16     cout << "The quiz averages are:\n";
17     printQuizAverages(score);
18
19     cout << "The class averages are:\n";
20     printClassAverages(score);
21
22     return 0;
23 }
24
25 void read(Score score)
26 {
27     cout << "Enter " << NUM_QUIZZES << " scores for each
28 student:\n";
29     for (int s = 0; s < NUM_STUDENTS; s++)
30     {
31         cout << "Student " << s << ": ";
32         for (int q = 0; q < NUM_QUIZZES; q++)
33             cin >> score[s][q];
34     }
35 }
36
37 void printQuizAverages(Score score)
38 {
39     for (int s = 0; s < NUM_STUDENTS; s++)
40     {
41         float sum = 0.0;
42         for (int q = 0; q < NUM_QUIZZES; q++)
43             sum += score[s][q];
44         cout << "\tStudent " << s << ": "
45              << sum/NUM_QUIZZES << endl;
46     }
47 }
48
49 void printClassAverages(Score score)
50 {
51     for (int q = 0; q < NUM_QUIZZES; q++)
52     {
53         float sum = 0.0;
54         for (int s = 0; s < NUM_STUDENTS; s++)
55             sum += score[s][q];
56         cout << "\tQuiz " << q << ": "
57              << sum/NUM_STUDENTS << endl;
58     }
59 }
```

ผลการทำงาน เมื่อใส่อินพุตเป็น 9 8 7 6 5 7 6 5 7 6 8 4 5 2 9 ตามลำดับผ่านทางแป้นพิมพ์ (ตัวหนาคืออินพุต):

```
Enter 5 scores for each student:
Student 0: 9 8 7 6 5
Student 1: 7 6 5 7 6
Student 2: 8 4 5 2 9
The quiz averages are:
        Student 0: 7
        Student 1: 6.2
        Student 2: 5.6
The class averages are:
        Quiz 0: 8
        Quiz 1: 6
        Quiz 2: 5.66667
        Quiz 3: 5
        Quiz 4: 6.66667
```

ตัวอย่างที่ 67 การบวกเมทริกซ์

โจทย์ : จงเขียนโปรแกรมภาษาซีพลัสพลัสเพื่อบวกเมทริกซ์ A และ B ที่กำหนดให้โดยใช้อะเรย์ นำผลการบวกไปไว้ในเมทริกซ์ C แล้วแสดงผลออกทางหน้าจอ



รูปที่ 95 การบวกเมทริกซ์

1) เอาต์พุต คืออะเรย์ C ที่เป็นผลบวกของอะเรย์ A และ B
2) ไม่มีอินพุต
3) โจทย์กำหนดให้ใช้อะเรย์ในการบวก
4) เราทราบว่าการบวกเมทริกซ์คือการบวกค่าในเมทริกซ์ 2 เมทริกซ์ที่ตำแหน่งเดียวกัน (แถวและหลักเดียวกัน) แล้วเอาไปไว้ในเมทริกซ์ใหม่ที่ตำแหน่งเดียวกันนั้น ดังนั้นเมทริกซ์ที่เอามาบวกกันและเมทริกซ์ผลลัพธ์จะต้องมีขนาดเท่ากัน

โจทย์ข้อนี้กำหนดให้ใช้อะเรย์ในการบวก นั่นคือเราจะต้องแทนเมทริกซ์ A, B และ C ด้วยอะเรย์ 2 มิติ โดยโจทย์ได้กำหนดค่าของ A และ B มาแล้ว เราก็ให้ค่าเริ่มต้นกับ A และ B ได้เลยโดยไม่ต้องมีการรับค่าเข้ามาจากผู้ใช้ เรารู้ว่าการบวกเมทริกซ์คือการบวกค่าในตำแหน่งเดียวกัน ดังนั้นเราจึงวนลูป 2 ชั้นเพื่อเข้าถึงค่าให้ครบทุกตำแหน่ง แล้วในแต่ละตำแหน่งให้เอาค่าสมาชิกใน A และ B ณ ตำแหน่งนั้น (แถวที่ i หลักที่ j) มาบวกกันแล้วเอาไปเก็บไว้ใน C[i][j] ตามโค้ดในบรรทัดที่ 12

โค้ด:

```
1   int main()
2   {
3       int A[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};
4       int B[3][3] = {{1,1,1}, {2,0,2}, {-3,-2,-1}};
5       int C[3][3];
6
7       // Add: C = A+B
8       for (int i = 0; i < 3; i++)
9       {
10          for (int j = 0; j < 3; j++)
11          {
12              C[i][j] = A[i][j] + B[i][j];
13          }
14      }
15
16      // Print A and B
17      cout << "Matrix A: " << endl;
18      for (int i = 0; i < 3; i++)
19      {
20          for (int j = 0; j < 3; j++)
21          {
22              cout << A[i][j] << " ";
23          }
24          cout << endl;
25      }
26
27      cout << "Matrix B: " << endl;
28      for (int i = 0; i < 3; i++)
29      {
30          for (int j = 0; j < 3; j++)
31          {
32              cout << B[i][j] << " ";
33          }
34          cout << endl;
35      }
36
37      // Print C
38      cout << "Matrix C: " << endl;
39      for (int i = 0; i < 3; i++)
40      {
41          for (int j = 0; j < 3; j++)
```

```
42          {
43              cout << C[i][j] << " ";
44          }
45          cout << endl;
46      }
47      return 0;
48  }
```

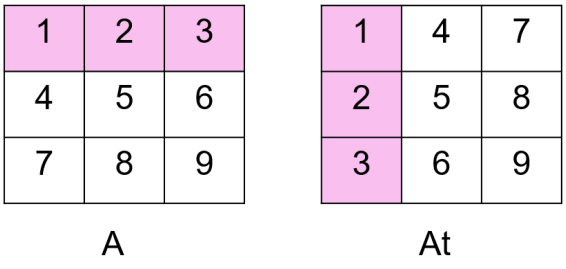
ส่วนโค้ดที่เหลือเป็นโค้ดการพิมพ์อะเรย์ A, B และ C ออกทางหน้าจอ ซึ่งเราสามารถเขียนอีกแบบเป็นฟังก์ชันก็ได้ (ให้นักศึกษาลองทำดูเอง) เมื่อบวกเมทริกซ์เสร็จแล้วเราจะได้ผลการทำงานของโปรแกรมดังนี้

ผลการทำงาน:

```
1   Matrix A:
2   1 2 3
3   4 5 6
4   7 8 9
5   Matrix B:
6   1 1 1
7   2 0 2
8   -3 -2 -1
9   Matrix C:
10  2 3 4
11  6 5 8
12  4 6 8
```

ตัวอย่างที่ 68 การสลับตำแหน่งแถวกับหลักในเมทริกซ์ (matrix transpose)

โจทย์ : จงเขียนโปรแกรมภาษาซีพลัสพลัสเพื่อสลับตำแหน่งแถวกับหลัก (matrix transpose) ของเมทริกซ์ A ที่กำหนดให้โดยใช้อะเรย์ แล้วนำผลที่ได้ไปไว้ในเมทริกซ์ At แล้วแสดงผลออกทางหน้าจอ



รูปที่ 96 การสลับตำแหน่งแถวกับหลักของเมทริกซ์ (matrix transpose)

1) เอาต์พุต คืออะเรย์ At ที่สลับแถวกับหลักของอะเรย์ A

2) ไม่มีอินพุต
3) โจทย์กำหนดให้ใช้อะเรย์ในการเก็บเมทริกซ์
4) เราทราบว่าการสลับแถวกับหลักของเมทริกซ์ (matrix transpose) คือการนำสมาชิกของแถวที่ 1 ทั้งหมดมาเรียงเป็นหลักที่ 1 และนำสมาชิกของแถวที่ 2 ทั้งหมดมาเรียงเป็นหลักที่ 2 แบบนี้จนครบทุกแถว ดังแสดงในรูปที่ 96

โจทย์ข้อนี้กำหนดให้ใช้อะเรย์ในการเก็บเมทริกซ์ โดยโจทย์ได้กำหนดค่าของ A มาแล้วเช่นเดียวกับตัวอย่างที่แล้ว ในการสลับแถวกับหลักของเมทริกซ์ (matrix transpose) ให้เราสังเกตผลลัพธ์ของ At เทียบกับ A เช่นค่าในตำแหน่งเส้นทะแยงมุมของเมทริกซ์จะไม่มีการเปลี่ยนแปลง นั่นคือค่าของ At[0][0], At[1][1] และ At[2][2] จะเป็นค่าเดียวกับสมาชิกของ A ในตำแหน่งเดียวกัน ส่วนค่าในตำแหน่งอื่นๆ เช่น At[0][1] ซึ่งมีค่า 4 จะเอามาจาก A[1][0] ส่วน At[0][2] ซึ่งมีค่า 7 จะเอามาจาก A[2][0] ส่วน At[2][1] ซึ่งมีค่า 6 จะเอามาจาก A[1][2] เราจะเห็นได้ว่า ค่าในตำแหน่งหนึ่งๆ ใน At จะเอาค่ามาจากการสลับดัชนี (เบอร์ห้อง) ของแถวกับหลักในเมทริกซ์ A นั่นคือ At[i][j] จะเท่ากับ A[j][i] เมื่อกำหนดให้ i และ j เป็นดัชนีของเมทริกซ์นั่นเอง ดังนั้นเราจึงได้โค้ดสำหรับแก้ปัญหานี้ดังนี้ โดยโค้ดสำหรับการสลับแถวและหลักจะอยู่ในบรรทัดที่ 6-13 ส่วนโค้ดหลังจากนั้นเป็นการแสดงเมทริกซ์ A และ At ออกทางหน้าจอ

โค้ด:

```
1   int main()
2   {
3       int A[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};
4       int At[3][3];
5
6       // Transpose A
7       for (int i = 0; i < 3; i++)
8       {
9           for (int j = 0; j < 3; j++)
10          {
11              At[i][j] = A[j][i];
12          }
13      }
14
15      // Print A
16      cout << "Matrix A: " << endl;
17      for (int i = 0; i < 3; i++)
18      {
19          for (int j = 0; j < 3; j++)
20          {
21              cout << A[i][j] << " ";
22          }
```

```
23          cout << endl;
24      }
25
26      // Print At
27      cout << "Matrix At: " << endl;
28      for (int i = 0; i < 3; i++)
29      {
30          for (int j = 0; j < 3; j++)
31          {
32              cout << At[i][j] << " ";
33          }
34          cout << endl;
35      }
36      return 0;
37  }
```

ผลการทำงาน:

```
1   Matrix A:
2   1 2 3
3   4 5 6
4   7 8 9
5   Matrix At:
6   1 4 7
7   2 5 8
8   3 6 9
```

ตัวอย่างที่ 69 การนับค่า 0 ในอะเรย์ 3 มิติ

โจทย์ : กำหนดการประกาศฟังก์ชันและฟังก์ชันหลัก (main) ให้ดังนี้

```
1   int numZeros(int[2][4][3], int, int, int);
2
3   int main()
4   {
5       int a[2][4][3] =
6       {
7           {{5,0,2}, {0,0,9}, {4,1,0}, {7,7,7}},
8           {{3,0,0}, {8,5,0}, {0,0,0}, {2,0,9}}
9       };
10
11      cout << "This array has " << numZeros(a, 2, 4, 3) << "
12  zeros.\n";
13
14      return 0;
15  }
```

จงเขียนนิยามของฟังก์ชัน numZeros() ในภาษาซีพลัสพลัสเพื่อนับจำนวนสมาชิกของอะเรย์ a ที่มีค่าเป็น 0

1) เอาต์พุต คือจำนวนของค่า 0 ในอะเรย์ 3 มิติ a ซึ่งเป็นค่าที่ส่งกลับ (return) มาจากฟังก์ชัน numZeros()
2) ไม่มีอินพุตในฟังก์ชันหลัก
3) โจทย์กำหนดการประกาศฟังก์ชันและการเรียกฟังก์ชันมา ทำให้เรารู้ว่าค่าพารามิเตอร์ของฟังก์ชัน numZeros() คืออะเรย์ 3 มิติขนาด 2x4x3 และจำนวนเต็ม 3 จำนวนคือขนาดของอะเรย์ที่ส่งผ่านเข้าไป ซึ่งได้แก่จำนวนแถว หลัก และความลึกตามลำดับ และฟังก์ชัน numZeros() จะต้องคืนค่ามาเป็นจำนวนเต็ม
4) เราทราบวิธีการค้นหาอะเรย์ 1 มิติ เราก็จะนำมาประยุกต์กับอะเรย์ 3 มิติ โดยการใช้การค้นหาแบบเส้นตรง (linear search) คือหาทั่วทุกสมาชิกในอะเรย์และนับค่า 0

โจทย์ข้อนี้กำหนดอะเรย์ 3 มิติและสมาชิกทุกตัวมาให้ และให้หาว่าสมาชิกของอะเรย์ที่มีค่าเป็น 0 นั้นมีกี่ตัว โดยเขียนในฟังก์ชัน numZeros() เราสามารถใช้การค้นหาแบบเส้นตรง (linear search) ในการหาได้ โดยเราจะต้องวนลูป 3 ชั้น (แต่ละชั้นควบคุมแถว หลักและความลึกตามลำดับ) เพื่อเข้าถึงสมาชิกของอะเรย์แต่ละตัว จากนั้นก็เปรียบเทียบว่าค่าสมาชิกในตำแหน่งนั้นๆ (แถวที่ i หลักที่ j ความลึกที่ k) เป็น 0 หรือไม่ ถ้าเป็น 0 ก็ให้เพิ่มค่าเข้าตัวนับ (count) เมื่อเราวนลูปไปตรวจสอบค่าของสมาชิกทุกตัวแล้ว เราก็คืนค่า (return) ตัวนับ count เพื่อกลับมาแสดงผลที่ฟังก์ชันหลัก (main)

โค้ด:

```
1   int numZeros(int[][4][3], int, int, int);
2
3   int main()
4   {
5       int a[2][4][3] =
6       {
7           {{5,0,2}, {0,0,9}, {4,1,0}, {7,7,7}},
8           {{3,0,0}, {8,5,0}, {0,0,0}, {2,0,9}}
9       };
10
11      cout << "This array has " << numZeros(a, 2, 4, 3) << "
12  zeros.\n";
13
14      return 0;
```

```
15 }
16
17 int numZeros(int a[][4][3], int n1, int n2, int n3)
18 {
19     int count = 0;
20     for (int i = 0; i < n1; i++)
21         for (int j = 0; j < n2; j++)
22             for (int k = 0; k < n3; k++)
23                 if (a[i][j][k] == 0) count++;
24     return count;
25 }
```

ผลการทำงาน:

```
1  This array has 11 zeros.
```

ตัวอย่างที่ 70 การคูณเมทริกซ์

โจทย์ : จงเขียนโปรแกรมภาษาซีพลัสพลัส เพื่อคำนวณหาผลลัพธ์การคูณเมทริกซ์ A และ B (dot product) แล้วแสดงผลการคูณออกทางหน้าจอ โดยกำหนดให้ A และ B มีสมาชิกดังนี้

```
1     int A[4][2] = {{1,2}, {3,4}, {5,6}, {7,8}};
2     int B[2][3] = {{2,2,2}, {3,3,3}};
```

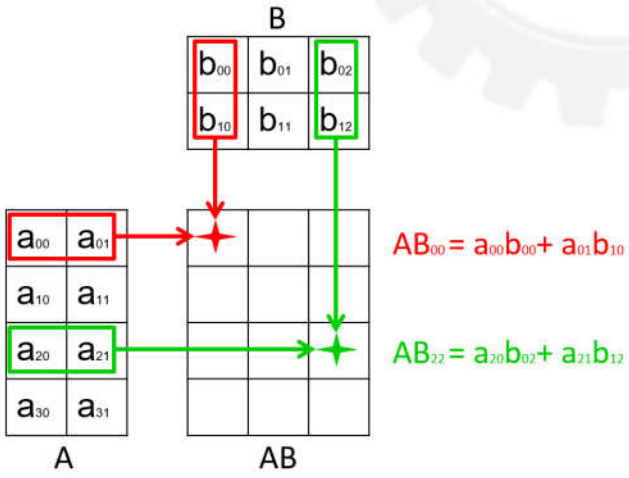
1) เอาต์พุต คือผลการคูณเมทริกซ์ A และ B ที่กำหนดให้
2) ไม่มีอินพุต
3) โจทย์กำหนดเมทริกซ์ A และ B มา โดย A มีขนาด 4x2 และ B มีขนาด 2x3 ตามลำดับ
4) เราทราบวิธีการคูณเมทริกซ์ อย่างแรกเลยคือเมทริกซ์ผลลัพธ์จะมีจำนวนแถวเท่ากับจำนวนแถวของ A และจำนวนหลักเท่ากับจำนวนหลักของ B ดังนั้นเมทริกซ์ผลลัพธ์จะมีขนาดเท่ากับ 4x3 ส่วนผลคูณในแต่ละตำแหน่งนั้น เราสามารถใช้สมการต่อไปนี้ในการหาค่าได้

102) $$(AB)_{ij} = \sum_{k=1}^{m} A_{ik}\, B_{kj}$$

103) นั่นคือค่าของผลลัพธ์ของการคูณ ณ ตำแหน่ง (i,j) โดย i เป็นแถว j เป็นหลักจะเท่ากับผลรวมของการคูณสมาชิกของ A ในแถวที่ i (A[i][k]) กับสมาชิกของ B ในหลักที่ j (B[k][j]) ทุกตัวตามลำดับ

จากขนาดของเมทริกซ์ที่สามารถคูณกันได้ จำนวนหลักของ A จะต้องเท่ากับจำนวนแถวของ B (คือค่า m ในสมการข้างต้น) เพราะฉะนั้นจำนวนสมาชิกในแถวหนึ่งๆ ของ A (จำนวนหลักของ A) จะเท่ากับจำนวนสมาชิกในหลักหนึ่งๆ ของ B (จำนวนแถวของ B) เวลาคูณเราก็จับคู่สมาชิกเรียงตามลำดับ หาผลคูณของแต่ละคู่ แล้วเอาผลคูณทั้งหมดมาบวกกัน เราก็จะได้ผลลัพธ์ของการคูณเมทริกซ์ที่ตำแหน่ง (i,j) ดังแสดงในรูปที่ 97 เช่น สำหรับตำแหน่ง (0,0) ในเมทริกซ์ผลลัพธ์ (เราจะเรียกว่า เมทริกซ์ AB) เราจะเอาค่า (a00 * b00) + (a01 * b10) ไปใส่ ส่วนในตำแหน่ง (2,2) ในเมทริกซ์ AB เราจะเอาค่า (a20 * b02) + (a21 * b12) ไปใส่ ส่วนในตำแหน่ง (3,1) ในเมทริกซ์ AB เราจะเอาค่า (a30 * b01) + (a31 * b11) ไปใส่

104)



105)

รูปที่ 97 แผนภาพการคูณเมทริกซ์

ในโจทย์ข้อนี้ เราต้องทำการคูณเมทริกซ์ ซึ่งเป็นอะเรย์ 2 มิติ ทำให้เราต้องใช้การวนลูป 2 ชั้นในการเข้าถึงแต่ละตำแหน่งในเมทริกซ์ AB ดังนั้นเราต้องวน 4 รอบสำหรับลูปชั้นนอกและ 3 รอบสำหรับลูปชั้นใน ซึ่งในแต่ละตำแหน่ง (i,j) ของเมทริกซ์ผลลัพธ์นั้น เราต้องมีลูปอีกชั้นหนึ่งข้างในลูปชั้นใน (โค้ดบรรทัดที่ 15-20) เพื่อจะวนรวมผลการคูณสมาชิกของแถว i ของเมทริกซ์ A และแถว j ของเมทริกซ์ B ดังที่ได้อธิบายไปแล้ว โค้ดการคูณเมทริกซ์แสดงในบรรทัดที่ 10-22 ส่วนหลังจากนั้นเป็นเพียงการพิมพ์เมทริกซ์ A, B และ AB ออกทางหน้าจอ

โค้ด:

```
1   #include <iomanip>
2
3   int main()
```

```
4   {
5       int A[4][2] = {{1,2}, {3,4}, {5,6}, {7,8}};
6       int B[2][3] = {{2,2,2}, {3,3,3}};
7       int AB[4][3];
8       int sum;
9
10      // AB = A x B (dot product)
11      for(int i = 0; i < 4; i++) // each row
12      {
13          for(int j = 0; j < 3; j++) // each col
14          {
15              sum = 0;
16              for(int k = 0; k < 2; k++) // sum of mult
17              {
18                  sum += A[i][k] * B[k][j];
19              }
20              AB[i][j] = sum;
21          }
22      }
23
24      // print A and B
25      cout << "A: " << endl;
26      for(int i = 0; i < 4; i++)
27      {
28          for(int j = 0; j < 2; j++)
29          {
30              cout << std::setw(4) << A[i][j];
31          }
32          cout << endl;
33      }
34
35      cout << "B: " << endl;
36      for(int i = 0; i < 2; i++)
37      {
38          for(int j = 0; j < 3; j++)
39          {
40              cout << std::setw(4) << B[i][j];
41          }
42          cout << endl;
43      }
44
45      // Print AB
46      cout << "AB: " << endl;
47      for(int i = 0; i < 4; i++)
48      {
49          for(int j = 0; j < 3; j++)
50          {
51              cout << std::setw(4) << AB[i][j];
52          }
53          cout << endl;
54      }
55
56      return 0;
57  }
```

ผลการทำงาน:

```
A:
   1   2
   3   4
   5   6
   7   8
B:
   2   2   2
   3   3   3
AB:
   8   8   8
  18  18  18
  28  28  28
  38  38  38
```

จากโค้ดนี้ เราจะเห็นได้ว่าสำหรับการใช้อะเรย์ 2 มิตินั้น จริงๆ แล้วการเข้าถึงอะเรย์ในแต่ละตำแหน่งนั้น เราใช้แค่ลูป 2 ชั้นก็เพียงพอ แต่เราไม่สามารถสรุปแบบตายตัวได้ว่าถ้าเป็นโจทย์อะเรย์ 2 มิติเราจะต้องใช้ลูปแค่ 2 ชั้นเท่านั้น เพราะในแต่ละตำแหน่งของอะเรย์ที่เราเข้าถึงเราอาจจะต้องมีการวนลูปเพื่อทำสิ่งอื่นอีกอย่างน้อย 1 ชั้น ดังนั้นเราอาจจะใช้ลูปมากกว่า 2 ชั้นได้ดังตัวอย่างในโจทย์ข้อนี้ เป็นต้น

สรุปสิ่งที่ควรได้จากบทเรียน

1) การทำงานกับอะเรย์ 1 มิติ
    a. ให้คิดว่าเหมือนเป็นห้องแถว โดยเบอร์ห้อง (ดัชนี) เริ่มจาก 0 เสมอ
    b. การพิมพ์อะเรย์จากหน้าไปหลังและหลังมาหน้า
2) หลักการทำงานของการค้นหาค่าในอะเรย์และการเรียงอะเรย์
3) การใช้อะเรย์ในฟังก์ชัน
    a. การผ่านอะเรย์เข้าฟังก์ชันมีคุณสมบัติเดียวกับการผ่านตัวแปรแบบอ้างอิง (by reference) แต่ไม่ต้องใส่เครื่องหมาย & ในนิยามของฟังก์ชัน
    b. ต้องใส่ขนาดของอะเรย์ในทุกมิติ ยกเว้นมิติแรกสามารถละได้
4) การทำงานกับอะเรย์ 2 มิติ

a. ให้คิดว่าเหมือนเป็นตารางที่มีแถว (row) และหลัก (column)

b. เข้าถึงสมาชิกทุกตัวโดยใช้ลูป 2 ชั้นเพื่อควบคุมแถวและหลักตามลำดับที่ต้องการ

5) การทำงานกับอะเรย์ 3 มิติ

a. ให้คิดว่าเหมือนเป็นกล่องที่มีแถว (row) หลัก (column) และความลึก (depth)

b. เข้าถึงสมาชิกทุกตัวโดยใช้ลูป 3 ชั้นเพื่อควบคุมแถว หลักและความลึกตามลำดับที่ต้องการ