มหาวิทยาลัยราชภัฏนครปฐม

บทที่ 9
ความสัมพันธ์เวียนเกิดและการเรียงลำดับ

ผู้บรรยาย : ผศ.ดร.ณัฐชามญฑ์ ศรีจำเริญรัตนา
สาขาวิชาคอมพิวเตอร์ธุรกิจ คณะวิทยาการจัดการ
มหาวิทยาลัยราชภัฏนครปฐม

ความสัมพันธ์เวียนเกิดและการเรียงลำดับ

- ความสัมพันธ์เวียนเกิดเป็นลำดับความสัมพันธ์ของตัวแปรลำดับ a0, a1, a2, ... เป็นสมการที่แสดงความสัมพันธ์ของ an กับตัวที่มาก่อน คือ a0, a1, ..., an-1 ในรูปแบบที่แน่นอน
- ความสัมพันธ์เวียนเกิดชุดหนึ่งจะอธิบายด้วยเทอมที่ n ของลำดับชุดหนึ่งโดยทางอ้อม คือ

  จะคำนวณ an ต้องคำนวณเทอม a0 , ..., an-1 มาก่อน
- ความสัมพันธ์เวียนเกิดเป็นกระบวนการเขียนโปรแกรมเพื่อเรียกตัวเองทำงานแบบวนซ้ำ(recursion) หลาย ๆ รอบและมีการกำหนดการทำงานขั้นสุดท้ายไว้ จะเห็นได้ว่ากระบวนการเวียนเกิดเป็นกระบวนการแก้ปัญหาด้วยวิธีลดปัญหาให้มีขนาดเล็กลง อีกทั้งยังสามารถใช้แก้ปัญหาที่มีความซับซ้อนได้เป็นอย่างดี

9.1 ความสัมพันธ์เวียนเกิด

ตัวอย่างที่ 9.1 กำหนดให้ f(1) = 2 และ f(x + 1) = f(x) + 4 จงหาค่าของ f(2) และ f(5)

วิธีทำ สิ่งที่โจทย์กำหนดให้ f(1) = 2

สิ่งที่โจทย์ต้องการหาค่าของ f(2)

$f(1 + 1) = f(1) + 4$

$f(2) = 2 + 4 = 6$.....ตอบ

สิ่งที่โจทย์ต้องการหาค่าของ f(5) แต่ต้องหา f(3), f(4) ก่อน

$f(2 + 1) = f(2) + 4$

$f(3) = 6 + 4 = 10$

$f(3 + 1) = f(3) + 4$

$f(4) = 10 + 4 = 14$

ทำให้ได้ $f(4 + 1) = f(4) + 4$

$f(5) = 14 + 4 = 18$.....ตอบ

ดังนั้น f(2) เท่ากับ 6 และ f(5) เท่ากับ 18

9.2 แฟคทอเรียล

- แฟคทอเรียลของ n สามารถเขียนด้วย n! หรือ เอ็นแฟคทอเรียล
- โดยทั่วไป ถ้า n เป็นจำนวนเต็มบวก สามารถเขียนได้เป็น n! ให้อยู่ในรูปแบบแฟคทอเรียล ดังนี้

0! = 1

n! =n x (n-1)!

ดังนั้นอัลกอลิทึมแฟคทอเรียลสามารถอธิบายโดยใช้โปรแกรมแสดงได้ดังนี้

9.2 แฟคทอเรียล(ต่อ)

ตัวอย่างที่ 9.2 จงเขียนฟังก์ชั่นเพื่อคำนวณค่าแฟคทอเรียล

```
1  int factorial(int n)
2  {
3     if(n>1)
4      return n*factorial(n-1);
5  }
```

อธิบายการทำงาน

อัลกอริทึมกำหนดให้

- ถ้าตัวแปร n ไม่เท่ากับ 1 จะสามารถคำนวณหาค่า factorial

  จากค่าตัวแปร n คูณกับค่าผลคูณ factorial n-1
- แต่กรณีถ้า n มีค่าเท่ากับ 0 จะคืนค่า 0 มาให้

9.2 แฟคทอเรียล(ต่อ)

ตัวอย่างที่ 9.3 การเขียนกระบวนการเวียนเกิดเพื่อคำนวณหาค่าแฟคทอเรียล (factorial)

เช่น 5! สามารถหาค่าดังนี้

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

$4! = 4 \times 3 \times 2 \times 1 = 24$

9.2 แฟคทอเรียล(ต่อ)

แนวคิดเกี่ยวกับ

<u>อัลกอริทึมการเวียนเกิด</u> เป็นการค้นหาวิธีแก้ปัญหาที่กำหนดโดยตรงจากปัญหาที่มีให้ลดขนาดปัญหาของตัวเองลง

ส่วน **<u>ฟังก์ชั่นการเวียนเกิด</u>** เป็นฟังก์ชั่นที่ใช้เรียกตัวเอง ซึ่งภายในฟังก์ชั่นประกอบด้วยฟังก์ชั่นตัวเองที่สามารถดำเนินประมวลผลการทำงานซ้ำ ๆ ก่อนสิ้นสุดกระบวนการทำงาน และจะพบว่าอัลกอลิทึมการเวียนเกิดจะดำเนินการโดยอาศัยฟังก์ชั่นการเวียนเกิด

9.2 แฟคทอเรียล(ต่อ)

ตัวอย่างที่ 9.4 ฟังก์ชั่นการเวียนเกิดจะดำเนินการผ่าน ฟังก์ชั่นแฟคทอเรียล

Int fact (int num)
{
If (num ==0)
return 1
else
return num * fact(num-1)
}
	ผลการทำงาน cout<< fact(3)

- การเรียกฟังก์ชั่นการเวียนเกิดโดยตรง (direct recursion) คือ ฟังก์ชั่นที่ทำการเรียกตัวเองโดยตรง
- การเรียกฟังก์ชั่นการเวียนเกิดโดยอ้อม (indirect recursion) คือ ฟังก์ชั่นที่ทำการเรียกฟังก์ชั่นอื่นเพื่อมาทำงาน

9.3 ลำดับการเวียนเกิด

- ลำดับการเวียนเกิด หรือจำนวนลำดับฟิโบนักซี (fibonacci) คิดค้นโดย Leonardo Fibonacci ในปี ค.ศ. 1202
- โดยจำนวน fibonacci เป็นกลุ่มจำนวนที่เกิดจากผลบวกของ 2 จำนวนก่อนหน้า และทำการบวกเรียงลำดับไปเรื่อย ๆ พิจารณาลำดับตัวเลขจำนวนเต็มตามลำดับของ fibonacci ต่อไปนี้

1, 1, 2, 3, 5, 8, 13, 21, 34,.....

- ตัวเลขที่ปรากฏจะอาศัยตัวเลขก่อนหน้ามาทำการคำนวณ
- โดยกำหนดให้ ข้อมูล 2 ลำดับแรกเท่ากับ 1 จากนั้นลำดับถัดไปจะกำหนด ดังต่อไปนี้

$F1=F2=1$

$Fi = F(i-1)+F(i-2)$ (สำหรับ $i > 2$)

9.3 ลำดับการเวียนเกิด(ต่อ)

ตัวอย่างที่ 9.5 จงเขียนฟังก์ชั่นเพื่อคำนวณหาค่าจากลำดับการเวียนเกิด

```
1  int fibonacci(int number)
2  {
3  if(number == 0 || number == 1)
4  return number;
5  else
6  return (fibonacci(number-1) + fibonacci(number-2));
7  }
```

9.3 ลำดับการเวียนเกิด(ต่อ)

ตัวอย่างความสัมพันธ์เวียนเกิดที่เก่าแก่ที่สุด คือ ลำดับการเวียนเกิดของฟิโบนักซีที่ได้ตั้งคำถามไว้ว่า “ถ้าเลี้ยงกระต่ายโตเต็มวัยและต่างเพศกันคู่หนึ่ง หลังจาก 1 ปี จะมีกระต่ายกี่คู่ ถ้าแต่ละคู่จะให้กำเนิดกระต่ายคู่ใหม่ (ต่างเพศ) ทุกเดือน กระต่ายคู่ใหม่จะกลายเป็นกระต่ายโตเต็มวัยใน 2 เดือน และจะให้กำเนิดกระต่ายคู่ใหม่ต่อไป สมมุติว่าจะไม่มีกระต่ายตายเกิดขึ้น”

โดยกำหนดให้ F_n แทนจำนวนคู่ของกระต่ายเมื่อตอนต้นเดือนที่ n จะได้ตารางแสดงผลผลิตของกระต่ายดังนี้

เดือนที่ n	จำนวนคู่กระต่ายเดือนที่ n			
	เดือนที่ n	อายุ 1 เดือน	เกิดใหม่	ทั้งหมด (F_n)
1	1	0	0	1
2	1	0	1	2
3	1	1	1	3
4	2	1	2	5
5	3	2	3	8
6	5	3	5	13

9.3 ลำดับการเวียนเกิด(ต่อ)

ดังนั้นจำนวนคู่กระต่ายในเดือนที่ n จะเท่ากับจำนวนคู่กระต่ายเดือนที่ n-1 บวกกับจำนวนคู่กระต่ายเกิดใหม่ในเดือนที่ n ดังนั้น สำหรับ $n \geq 3$ จะได้ว่า $F_n = F_{n-1} + F_{n-2}$

ถ้ากำหนดให้ $F_0 = 1$, จากสมการข้างต้นจะเป็นจริงถ้า $n \geq 2$

จากตาราง $F_1 = 1$ ดังนั้น

$F_2 = F_1 + F_0 = 2 = (1+1)$

$F_3 = F_2 + F_1 = 3 = (2+1)$

$F_4 = F_3 + F_2 = 5 = (3+2)$

$F_5 = F_4 + F_3 = 8 = (5+3)$

$F_6 = F_5 + F_4 = 13 = (8+5)$

$F_7 = F_6 + F_5 = 21 = (13+8)$

$F_8 = F_7 + F_6 = 34 = (21+13)$ เป็นต้น

9.3 ลำดับการเวียนเกิด(ต่อ)

ตัวอย่างที่ 9.6 จงเขียนโปรแกรมเพื่อคำนวณหาค่าฟิโบนักซี

```
1  #include <iostream>
2  using namespace std;
3  int fibonacci(int);
4  int main(void)
5  {
6  int p;
7  int count;
8  cout << "Enter numbers in the Fibonacci
9  sequence ? ";
10 cin >> count;
11 cout<<"----------------------\n"<<endl;
```

```
12 for(p = 0; p<=count; p=p+1)
13 cout<<"Fibonacci("<<p<<") = “
          <<fibonacci(p)<<"\n";
14 system("pause");
15   return 0;
16 }
17
18 int fibonacci(int number)
19 {
20 if(number == 0 || number == 1)
21 return number;
22 else
23 return (fibonacci(number-1) +
          fibonacci(number-2));
24 }
```

9.3 ลำดับการเวียนเกิด(ต่อ)

ผลการทำงานของโปรแกรม

```
Enter numbers in the Fibonacci sequence ? 10
----------------------------------------

Fibonacci(0) = 0
Fibonacci(1) = 1
Fibonacci(2) = 1
Fibonacci(3) = 2
Fibonacci(4) = 3
Fibonacci(5) = 5
Fibonacci(6) = 8
Fibonacci(7) = 13
Fibonacci(8) = 21
Fibonacci(9) = 34
Fibonacci(10) = 55
```

9.3 ลำดับการเวียนเกิด(ต่อ)

ตัวอย่างที่ 9.7 จงใช้ลูป for ในการแสดงผล ลำดับการเวียนเกิด (fibonacci number)

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6     int count, first = 0, second = 1, next, c;
7
8     cout<<"Enter numbers
              in the Fibonacci sequence:";
9     cin>>count;
10
11   cout<<"\n-------------------------------
12  "<<endl;
```

```
13    for ( c = 0 ; c < count ; c++ )
14   {
15       if ( c <= 1 )
16         next = c;
17       else
18       {
19         next = first + second;
20         first = second;
21         second = next;
22       }
23       cout<<"\n"<<next;
24     }
25   system("pause");
26   return 0;
27  }
```

9.3 ลำดับการเวียนเกิด(ต่อ)

ผลการทำงานของโปรแกรม

```
1   Enter numbers in the Fibonacci sequence:10
2
3   ----------------------------------------
4
5   0
6   1
7   1
8   2
9   3
10  5
11  8
12  13
13  21
14  34
```

9.4 ทาวเวอร์ออฟฮานอย

- ทาวเวอร์ออฟฮานอย (Tower of Hanoi) เป็นวิธีการแก้ปัญหาแบบรีเคอร์ซีฟ
- เนื่องจากปัญหามีความซับซ้อนในการแก้ปัญหาอย่างมาก ถ้าใช้วิธีการแก้ปัญหาแบบอื่นที่ไม่ใช่รีเคอร์ซีฟจะมีความยุ่งยากในการแก้ปัญหา
- ทาวเวอร์ออฟฮานอยเป็นเกมส์อย่างหนึ่งทางคณิตศาสตร์ ที่ประกอบด้วยเสาจำนวน 3 แท่งและจานขนาดที่แตกต่างกัน 3 ชิ้นและแต่ละชิ้นจะมีรูอยู่ตรงกลาง โดยให้ทำการย้ายแผ่นจากที่เรียงลำดับแผ่นใหญ่สุดอยู่ด้านล่างและแผ่นเล็กสุดอยู่ด้านบน ทั้ง 3 แผ่นวางไว้ที่เสาอันแรก จากนั้นให้ย้ายทั้ง 3 แผ่นมาเสาอันที่สาม
- ดังแสดงในตัวอย่างโปรแกรมการแก้ปัญหาทาวเวอร์ออฟฮานอย โดยให้มีวิธีย้ายแผ่นจานดังนี้

1) ในการย้ายจานแต่ละครั้งจะย้ายได้ทีละ 1 จาน

2) จานที่จะทำการย้าย ห้ามย้ายไว้บนจานที่มีขนาดเล็กกว่า

9.4 ทาวเวอร์ออฟฮานอย(ต่อ)

ตัวอย่างที่ 9.8 จงเขียนโปรแกรมเพื่อแก้ปัญหาทาวเวอร์ออฟฮานอย

```
1   #include <iostream>
2   using namespace std;
3
4   void towers(int, char, char, char);
5
6   int main()
7   {
8       int topN;
9
10      cout<<"Enter the number of disks : ";
11      cin>>topN;
12      cout<<"\nTower of Hanoi\n";
13      towers(topN, 'A', 'C', 'B');
14                  system("pause");
15                  return 0;
16  }
```

9.4 ทาวเวอร์ออฟฮานอย(ต่อ)

ตัวอย่างที่ 9.8 จงเขียนโปรแกรมเพื่อแก้ปัญหาทาวเวอร์ออฟฮานอย

```
17  void towers(int topN, char from, char to, char inter)
18  {
19     if (topN == 1)
20     {
21        cout<<"\n Move disk 1 from "<< from <<" to "<< to;
22        return;
23     }
24     towers(topN - 1, from, inter, to);
25     cout<<"\n Move disk "<< topN <<" from "<< from<<" to "<< to;
26     towers(topN - 1, inter, to, from);
27  }
```

9.4 ทาวเวอร์ออฟฮานอย(ต่อ)

ผลการทำงานของโปรแกรม

```
Enter the number of disks : 4

Tower of Hanoi

 Move disk 1 from A to B
 Move disk 2 from A to C
 Move disk 1 from B to C
 Move disk 3 from A to B
 Move disk 1 from C to A
 Move disk 2 from C to B
Move disk 1 from A to B
 Move disk 4 from A to C
 Move disk 1 from B to C
 Move disk 2 from B to A
 Move disk 1 from C to A
 Move disk 3 from B to C
 Move disk 1 from A to B
 Move disk 2 from A to C
 Move disk 1 from B to C
```

9.4 ทาวเวอร์ออฟฮานอย(ต่อ)

วิธีดำเนินการ

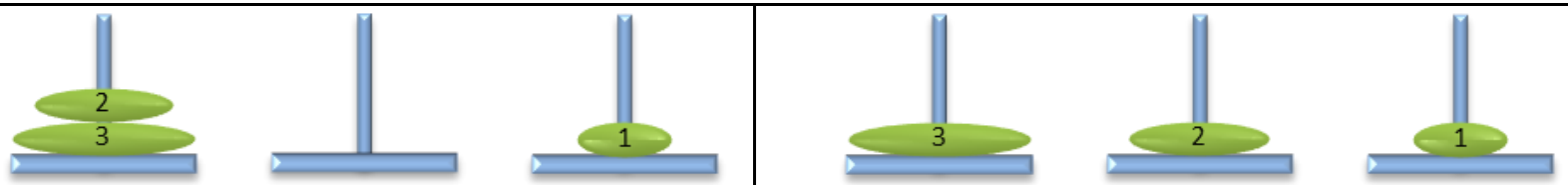
เริ่มต้นเกมส์จานทั้ง 3 ใบนี้จะวางที่เสาแรกเรียงลำดับขนาดใหญ่อยู่ล่างสุดจนถึงจานขนาดเล็กสุดให้อยู่ด้านบน เป้าหมายคือต้องการย้ายจานทั้งหมดไปอยู่เสาที่ 3 โดยมีเงื่อนไขดังต่อไปนี้

1) สามารถย้ายจานได้ครั้งละ 1 ใบและเป็นแผ่นที่อยู่บนสุดของเสา

2) จานที่มีขนาดใหญ่ไม่สามารถวางไว้บนจานที่มีขนาดเล็กกว่าได้

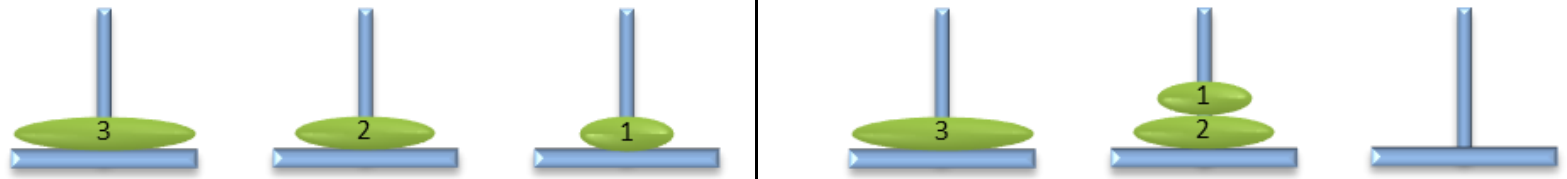
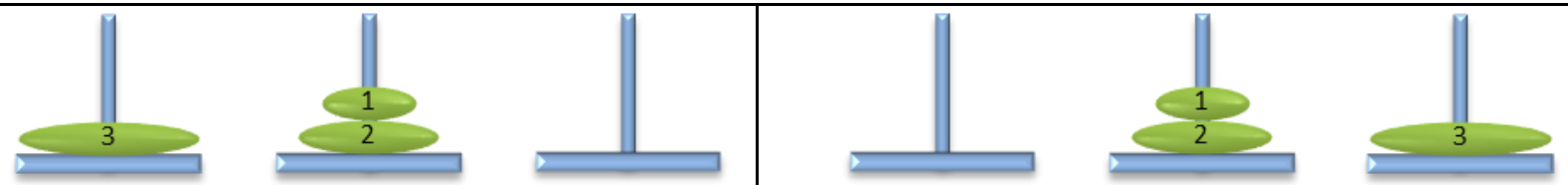
9.4 ทาวเวอร์ออฟฮานอย(ต่อ)

วิธีแก้ปัญหาทาวเวอร์ออฟฮานอยกับจาน 3 ใบ

	ก่อนเรียง	หลังเรียง
1		
	เริ่มต้น เสาแรกประกอบด้วยจาน 3 ใบ ให้ทำการย้ายจานใบที่ 1 ไปเสาที่ 3	
2		
	เสาแรกประกอบด้วยจาน 2 ใบ ให้ทำการย้ายจานใบที่อยู่บนสุดหมายเลข 2 มาไว้เสาที่ 2	

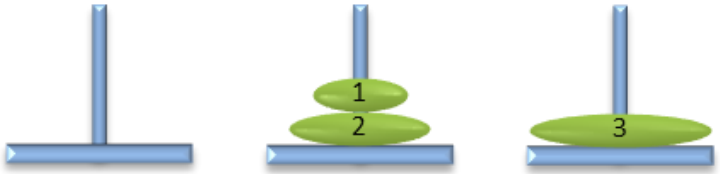
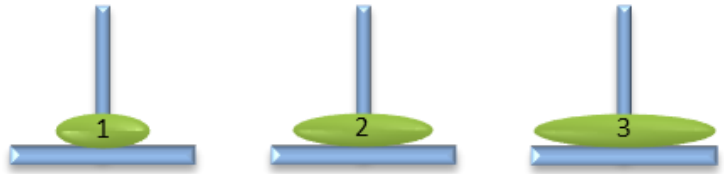
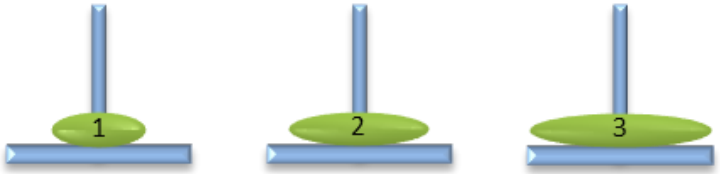
9.4 ทาวเวอร์ออฟฮานอย(ต่อ)

วิธีแก้ปัญหาทาวเวอร์ออฟฮานอยกับจาน 3 ใบ (1)

	ก่อนเรียง	หลังเรียง
3		
	ให้ทำการย้ายจานใบที 1 จากเสาที่ 3 มาไว้เสาที่ 2	
4		
	ให้ทำการย้ายจานใบที 3 จากเสาที่ 1 มาไว้เสาที่ 3	

9.4 ทาวเวอร์ออฟฮานอย(ต่อ)

วิธีแก้ปัญหาทาวเวอร์ออฟฮานอยกับจาน 3 ใบ (2)

	ก่อนเรียง	หลังเรียง
5		
	ให้ทำการย้ายจานใบที 1 จากเสาที่ 2 มาไว้เสาที่ 1	
6		
	ให้ทำการย้ายจานใบที 2 จากเสาที่ 2 มาไว้เสาที่ 3	

9.4 ทาวเวอร์ออฟฮานอย(ต่อ)

วิธีแก้ปัญหาทาวเวอร์ออฟฮานอยกับจาน 3 ใบ (3)

	ก่อนเรียง	หลังเรียง
7		
ให้ทำการย้ายจานใบที 1 จากเสาที่ 1 มาไว้เสาที่ 3		

จากการทำงานของฟังก์ชั่นพบว่า

จาน 1 ใบจะทำการย้าย 1 ครั้ง

จาน 2 ใบ จะทำการย้าย 3 ครั้ง

จาน 3 ใบ จะทำการย้าย 7 ครั้ง

ดังนั้นรูปแบบสมการจะเป็น 2^n-1

9.5 การเรียงสับเปลี่ยน Permutations

วิธีการเรียงสับเปลี่ยน

เป็นวิธีจัดเรียงสิ่งของ n สิ่ง โดยให้ความสำคัญกับลำดับหรือตำแหน่งสิ่งของที่นำมาจัดเรียงสับเปลี่ยนเป็นสิ่งที่ต้องคำนึงถึง เช่น การเรียงสับเปลี่ยนของ AB และ BA จะถือว่าแตกต่างกัน

วิธีเรียงสับเปลี่ยนสิ่งของที่แตกต่างกัน n สิ่ง

- ถ้ามีสิ่งของที่แตกต่างกันทั้งหมด n สิ่ง นำมาเรียงสับเปลี่ยนในแนวเส้นตรงทำได้ n! วิธี

9.5 การเรียงสับเปลี่ยน Permutations(ต่อ)

ตัวอย่างที่ 9.9 กำหนดให้ set { a,b,c } สามารถทำการเรียงสับเปลี่ยนสิ่งของได้กี่วิธี

วิธีคิด

จำนวนตัวอักษร 3 หลัก = 3! = 3 x2 x1 = 6 วิธี

9.5 การเรียงสับเปลี่ยน Permutations(ต่อ)

ตัวอย่างที่ 9.11 จงเขียนโปรแกรมให้เรียงสับเปลี่ยนสิ่งของที่แตกต่างกัน n สิ่ง

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <iostream>
4  using namespace std;
5
6  int z=0;
7  void swap(char *x, char *y)
8  {
9     char temp;
10    temp = *x;
11    *x = *y;
12    *y = temp;
13 }
14
```

```
15 void permute(char *a, int l, int r)
16 {
17   int i ;
18   if (l == r)
19   {
20        z=z+1;
21
22    cout<<z<<" : "<<a<<"\n";
23 }  else   {
24      for (i = l; i <= r; i++)
25      {
26        swap((a+l), (a+i));
27        permute(a, l+1, r);
28        swap((a+l), (a+i)); //backtrack
29      }
30   }
31 }
```

9.5 การเรียงสับเปลี่ยน Permutations(ต่อ)

ตัวอย่างที่ 9.11 จงเขียนโปรแกรมให้เรียงสับเปลี่ยนสิ่งของที่แตกต่างกัน n สิ่ง (ต่อ 1)

```
32   int main(void)
33   {
34      char str[] = "ABCDEF";
35      int n = strlen(str);
36      permute(str, 0, n-1);
37      system("pause");
38      return 0;
39   }
```

ผลการทำงานของโปรแกรม

```
ABC
ACB
BAC
BCA
CBA
CAB
```

9.6 การจัดกลุ่ม Combinations

วิธีการจัดกลุ่มสิ่งของ n สิ่ง โดยที่มีบางสิ่งเหมือนกันจำนวน m กลุ่ม

ถ้ามีสิ่งของ n สิ่ง โดยที่มีบางสิ่งเหมือนกันจำนวน m กลุ่ม สามารถจัดกลุ่มได้ดังนี้

กลุ่มที่ 1 มีสิ่งของซ้ำกัน n1 สิ่ง

กลุ่มที่ 2 มีสิ่งของซ้ำกัน n2 สิ่ง

กลุ่มที่ 3 มีสิ่งของซ้ำกัน n3 สิ่ง

:

กลุ่มที่ m มีสิ่งของซ้ำกัน nm สิ่ง

วิธีการจัดกลุ่ม จะเท่ากับ $(n!)/(n1!n2!n3!...nm!)$ ดังแสดงในตัวอย่างดังนี้

9.6 การจัดกลุ่ม Combinations(ต่อ)

ตัวอย่างที่ 9.12 จงหาจำนวนการจัดกลุ่มลำดับสิ่งของที่แตกต่างกัน 3 กลุ่มมาเรียงตามแนวเส้นตรง กำหนดให้มีลูกแก้วสีแดง 5 ลูก ลูกแก้วสีขาว 2 ลูก ลูกแก้วสีน้ำเงิน 3 ลูก และลูกแก้วสีเดียวกันไม่แตกต่างกัน ได้กี่วิธี

วิธีทำ

$$\frac{10!}{5!\,2!\,3!} = \frac{10 \times 9 \times 8 \times 7 \times 6 \times \cancel{5!}}{\cancel{5!} \times 2! \times 3!}$$

$$= \frac{10\,x\,9\,x\,8\,x7\,x\,6}{2!\,x\,3!}$$

$$= \frac{10\,x\,9\,x\,8\,x7}{2!}$$

$$= 10 \times 9 \times 4 \times 7$$

$$= 2520 \text{ วิธี}$$

9.7 การเรียงลำดับ

- การเรียงข้อมูล คือ การจัดลำดับของข้อมูลในชุดเดียวกันให้เรียงลำดับจากค่าน้อยไปมาก หรือจากค่ามากไปหาค่าน้อย โดยการเปรียบเทียบข้อมูลในชุดข้อมูลนั้น ๆ
- การเรียงข้อมูลในโครงสร้างข้อมูลเป็นส่วนที่ใช้บ่อยและมีหลายวิธีการในการเรียงข้อมูล
- ข้อมูลที่มีการเรียงลำดับอย่างเป็นระเบียบจะทำให้หาความสัมพันธ์ของข้อมูลต่าง ๆ กระทำได้สะดวกมากขึ้น

9.7 การเรียงลำดับ(ต่อ)

9.7.1 การเรียงลำดับแบบเลือก

- การเรียงลำดับแบบเลือก (selection sort) เป็นวิธีที่ง่ายต่อการทำความเข้าใจ
- จุดเด่นคือหาค่าที่น้อยที่สุดมาทำการพิจารณา ในการเรียงข้อมูลแบบอาร์เรย์ ข้อมูลจะถูกแบ่งเป็นสองชุด คือ
  - ชุดที่ทำการเรียงลำดับ
  - ชุดข้อมูลที่ยังไม่เรียงลำดับ

- การเรียงลำดับมีวิธีการดังต่อไปนี้

1) ข้อมูลตัวแรกของชุดข้อมูลที่ยังไม่เรียงจะถูกกำหนดให้เป็นข้อมูลชุดแรกที่เรียงลำดับแล้ว

2) นำข้อมูลตัวที่ทำการเรียงแล้วในชุดแรกมาทำการเปรียบเทียบกับข้อมูลชุดที่ยังไม่เรียงลำดับทุกตัวเมื่อพบตัวใดที่มีค่าน้อยกว่าข้อมูลชุดแรกที่เปรียบเทียบให้ทำการสลับตำแหน่งกัน

3) ทำซ้ำข้อสอง จนกว่าจะได้ครบทุกตำแหน่ง

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.13 โปรแกรมการเรียงลำดับแบบเลือก

```
1  #include <iostream>
2  using namespace std;
3
4  void display (int [], int);
5  void selection_sort (int [], int);
6  int min_item;
7  int size=10;
8
```

```
9  int main ()
10 {
11 int data[] = {12, 48, 7,65, 59,61, 41,93,
12 74,63 };
13 cout << "Data Before Sorting: ";
14 display (data, size);
15 selection_sort (data, size);
16 cout << "Data After Sorting: ";
17 display (data, size);
18 system("pause");
19 return 0;
20 }
```

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.13 โปรแกรมการเรียงลำดับแบบเลือก (ต่อ)

```
21  void selection_sort (int data[], int size)
22  {
23     for (int i = 0; i < size-1; ++i)
24     {
25          min_item = i;
26          for (int j = i + 1; j < size; ++j)
27          {
28               if (data[j] < data[min_item])
29               {
30                  min_item = j;
31               }
32          }
33          swap (data[i], data[min_item]);
34     }
35  }
36
```

```
37  void display (int data[], int size)
38  {
39       for (int i = 0; i < size; ++i)
40       {
41          cout << data[i] << " ";
42       }
43       cout << endl;
44  }
```

ผลการทำงานของโปรแกรม

Data Before Sorting: 12 48 7 65 59 61 41 93 74 63

Data After Sorting: 7 12 41 48 59 61 63 65 74 93

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.14 จงแสดงขั้นตอนการเรียงลำดับข้อมูลจากน้อยไปหามากด้วยวิธี selection sort

ดังข้อมูลต่อไปนี้

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
12	48	7	65	59	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 1 เริ่มต้นด้วยอาร์เรย์ 10 ข้อมูล คือ [0]...[9]

นำค่าในอาร์เรย์ตำแหน่ง [0] กับ ตำแหน่ง[1] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[2] น้อยกว่าอาร์เรย์ตำแหน่ง[0] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
12	48	7	65	59	61	41	93	74	63

ข้อมูลหลังการจัดเรีย

Data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	48	12	65	59	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 2 เริ่มต้นด้วยอาร์เรย์ตำแหน่ง คือ [1]...[9]

นำค่าในอาร์เรย์ตำแหน่ง [1] กับ ตำแหน่ง[2] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[2] น้อยกว่าอาร์เรย์ตำแหน่ง[1] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

Data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	48	12	65	59	61	41	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	65	59	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 3 เริ่มต้นด้วยอาร์เรย์ตำแหน่ง คือ [2]...[9]

นำค่าในอาร์เรย์ตำแหน่ง [2] กับ ตำแหน่ง[3] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[6] น้อยกว่าอาร์เรย์ตำแหน่ง[2] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	65	59	61	41	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	65	59	61	48	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 4 เริ่มต้นด้วยอาร์เรย์ตำแหน่ง คือ [3]...[9]

นำค่าในอาร์เรย์ตำแหน่ง [3] กับ ตำแหน่ง[4] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[4] น้อยกว่าอาร์เรย์ตำแหน่ง[3] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	65	59	61	48	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	59	65	61	48	93	74	63

9.7 การเรียงลำดับ(ต่อ)

พบว่าอาร์เรย์ตำแหน่ง[6] น้อยกว่าอาร์เรย์ตำแหน่ง[3] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	59	65	61	48	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	65	61	59	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 5 เริ่มต้นด้วยอาร์เรย์ตำแหน่ง คือ [4]...[9]

นำค่าในอาร์เรย์ตำแหน่ง [4] กับ ตำแหน่ง[5] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[5] น้อยกว่าอาร์เรย์ตำแหน่ง[4] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	65	61	59	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	61	65	59	93	74	63

9.7 การเรียงลำดับ(ต่อ)

นำค่าในอาร์เรย์ตำแหน่ง [4] กับ ตำแหน่ง[5] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[6] น้อยกว่าอาร์เรย์ตำแหน่ง[4] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	61	65	59	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	65	61	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 6 เริ่มต้นด้วยอาร์เรย์ตำแหน่ง คือ [5]...[9]

นำค่าในอาร์เรย์ตำแหน่ง [5] กับ ตำแหน่ง[6] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[6] น้อยกว่าอาร์เรย์ตำแหน่ง[5] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	65	61	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 7 เริ่มต้นด้วยอาร์เรย์ตำแหน่ง คือ [6]...[9]

นำค่าในอาร์เรย์ตำแหน่ง [6] กับ ตำแหน่ง[7] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[9] น้อยกว่าอาร์เรย์ตำแหน่ง[6] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	63	93	74	65

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 8 เริ่มต้นด้วยอาร์เรย์ตำแหน่ง คือ [7]...[9]

นำค่าในอาร์เรย์ตำแหน่ง [7] กับ ตำแหน่ง[8] ถึงตำแหน่ง[9] เปรียบเทียบค่าทีละลำดับ

พบว่าอาร์เรย์ตำแหน่ง[9] น้อยกว่าอาร์เรย์ตำแหน่ง[7] ให้ทำการสลับตำแหน่งกัน

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	63	93	74	65

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	63	65	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 9 ค่าในอาร์เรย์ลำดับที่ 8 และ 9 เรียงถูกต้องอยู่แล้วจึงจบการเรียงข้อมูล

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	63	65	74	93

9.7 การเรียงลำดับ(ต่อ)

9.7.2 การเรียงลำดับแบบแทรก

- การเรียงลำดับแบบแทรก (insertion sort) เป็นการเรียงลำดับข้อมูลอีกวิธีที่ง่ายที่สุด โดยข้อมูลที่นำมาเรียงลำดับด้วยวิธี insertion sort จะถูกแบ่งเป็น 2 ส่วน คือ

  - ข้อมูลส่วนที่เรียงลำดับ
  - ข้อมูลส่วนที่ยังไม่เรียงลำดับ

- ขั้นตอนการเรียงลำดับ

  1. ข้อมูลส่วนที่ยังไม่ได้เรียงลำดับลำดับที่ 2 เป็นข้อมูลส่วนที่เรียงลำดับแล้ว
  2. นำข้อมูลที่ไม่ได้เรียงลำดับตัวถัดไปเปรียบเทียบกับข้อมูลในส่วนข้อมูลที่เรียงลำดับจะแสดงทีละลำดับ ถ้าลำดับที่กำลังเปรียบเทียบมากกว่าให้ทำการแทรกลำดับก่อนหน้า แต่ถ้าน้อยกว่าให้ทำการเปรียบเทียบหาตำแหน่งที่เหมาะสมต่อไป
  3. วนการทำงานข้อ 2 เพื่อเรียงลำดับข้อมูลส่วนที่ยังไม่ได้เรียงจนครบ

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.15 จงเขียนโปรแกรมเพื่อทำการเรียงลำดับข้อมูลโดยใช้วิธีการเรียงลำดับแบบแทรก

```
1  #include <iostream>
2  using namespace std;
3
4  void display (int [], int);
5  void insertion_sort (int [], int);
6  int min_item;
7  int size=10;
8
```

```
9  int main()
10 {
11 int data[] = {12, 48, 7, 65, 59, 61,
   41, 93, 74, 63 };
12 cout << "Data Before Sorting: ";
13 display (data, size);
14 insertion_sort(data, size);
15 cout << "Data After Sorting: ";
16 display (data, 10);
17 system("pause");
18 return 0;
19 }
20
```

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.15 จงเขียนโปรแกรมเพื่อทำการเรียงลำดับข้อมูลโดยใช้วิธีการเรียงลำดับแบบแทรก

```
21  void insertion_sort(int data[], int size)
22  {
23  for (int i = 1 ; i <= size - 1; i++)
24  {
25  min_item = i;
26  while ( min_item > 0 &&
    data[min_item] < data[min_item-1])
27  {
28      swap (data[min_item-1],data[min_item]);
29       min_item--;
30  }
31  }
32  }
33
```

```
34  void display (int data[], int size)
35  {
36  for (int i = 0; i < size; ++i)
37  {
38          cout << data[i] << " ";
39  }
40  cout << endl;
41  }
```

9.7 การเรียงลำดับ(ต่อ)

ผลการทำงานของโปรแกรม

Data Before Sorting: 12 48 7 65 59 61 41 93 74 63

Data After Sorting: 7 12 41 48 59 61 63 65 74 93

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.16 จงแสดงขั้นตอนการเรียงลำดับข้อมูลจากน้อยไปหามาก

ด้วยวิธี เรียงลำดับแบบแทรก กำหนดให้ข้อมูลตัวแรกเป็นข้อมูลที่เรียงลำดับแล้ว

รอบที่ 1 12 ถูกเรียงลำดับแล้ว

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
12	48	7	65	59	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 2 กำหนดให้ 48 เป็นข้อมูลที่นำมาจัดเรียง

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
12	48	7	65	59	61	41	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
12	48	7	65	59	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 3 กำหนดให้ 7 เป็นข้อมูลที่ต้องการนำมาจัดเรียง

นำ 7 มาเปรียบเทียบกับข้อมูลชุดที่เรียงแล้วคือ 12, 48 ทีละลำดับ

พบว่า 7 น้อยกว่า 12 ให้ทำการแทรก 7 ไว้หน้า 12

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
12	48	7	65	59	61	41	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	65	59	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 4 กำหนดให้ 65 เป็นข้อมูลที่ต้องการนำมาจัดเรียง

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	65	59	61	41	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	65	59	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 5 กำหนดให้ 59 เป็นข้อมูลที่ต้องการนำมาจัดเรียง

นำ 59 มาเปรียบเทียบกับข้อมูลชุดที่เรียงแล้วคือ 7,12, 48,65 ทีละลำดับ

พบว่า 59 น้อยกว่า 65 แต่มากกว่า 48 ให้ทำการแทรก 59 ไว้หน้า 65

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	65	59	61	41	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	59	65	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 6 กำหนดให้ 61 เป็นข้อมูลที่ต้องการนำมาจัดเรียง

นำ 61 มาเปรียบเทียบกับข้อมูลชุดที่เรียงแล้วคือ 7,12, 48,59,65 ทีละลำดับ

พบว่า 61 น้อยกว่า 65 แต่มากกว่า 59 ให้ทำการแทรก 61 ไว้หน้า 65

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	59	65	61	41	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	59	61	65	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 7 กำหนดให้ 41 เป็นข้อมูลที่ต้องการนำมาจัดเรียง

นำ 41 มาเปรียบเทียบกับข้อมูลชุดที่เรียงแล้วคือ 7,12, 48,59,61,65 ทีละลำดับ

พบว่า 41 น้อยกว่า 48 แต่มากกว่า 12 ให้ทำการแทรก 41 ไว้หน้า 48

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	48	59	61	65	41	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 8 กำหนดให้ 93 เป็นข้อมูลที่ต้องการนำมาจัดเรียง

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 9 กำหนดให้ 74 เป็นข้อมูลที่ต้องการนำมาจัดเรียง

นำ 74 มาเปรียบเทียบกับข้อมูลชุดที่เรียงแล้วคือ 7, 12, 41, 48, 59, 61, 65, 93 ทีละลำดับ

พบว่า 74 น้อยกว่า 93 แต่มากกว่า 65 ให้ทำการแทรก 74 ไว้หน้า 93

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	93	74	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	74	93	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 10 กำหนดให้ 93 เป็นข้อมูลที่จะนำมาจัดเรียง

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	74	93	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	74	93	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 11 กำหนดให้ 63 เป็นข้อมูลที่ต้องการนำมาจัดเรียง

นำ 63 มาเปรียบเทียบกับข้อมูลชุดที่เรียงแล้วคือ 7, 12, 41, 48, 59, 61, 65, 74, 93 ทีละลำดับ พบว่า 63 น้อยกว่า 65 แต่มากกว่า 61 ให้ทำการแทรก 63 ไว้หน้า 65

ข้อมูลก่อนการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	65	74	93	63

ข้อมูลหลังการจัดเรียง

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
7	12	41	48	59	61	63	65	74	93

9.7 การเรียงลำดับ(ต่อ)

9.7.3 การเรียงลำดับแบบฟองสบู่

- การเรียงลำดับแบบฟองสบู่ (bubble sort) เหมาะสมกับข้อมูลที่มีจำนวนน้อย ถ้าข้อมูลที่มีจำนวนมาก ๆ อัลกอริทึมนี้ถือว่าไม่มีประสิทธิภาพ
- การทำงานเริ่มด้วยการเปรียบเทียบทีละคู่ของข้อมูลที่อยู่ติดกันจากซ้ายไปทางด้านขวา ถ้าอาร์เรย์ข้อมูลที่นำมาเปรียบเทียบกับอาร์เรย์ข้อมูลตัวถัดไปมีค่าน้อยกว่าทีละคู่ ให้ทำการสลับตำแหน่งกัน ทำเช่นนี้จนครบทุกสมาชิกในอาร์เรย์ รอบต่อไปจะพบว่า ข้อมูลตัวแรกเป็นข้อมูลที่ถูกเรียงลำดับแล้ว
- ดังนั้นรอบนี้จะไม่ต้องทำการเปรียบเทียบกับข้อมูลชุดที่ผ่านการทำงานรอบก่อนหน้านี้แล้วทำให้ทุกรอบการทำงานที่เพิ่มขึ้นจะมีข้อมูลที่ต้องนำสู่กระบวนการเปรียบเทียบน้อยลง 1 สมาชิก

ข้อสังเกต การณ์ทำงานแบบ bubble sort เนื่องจากการเรียงลำดับจะสามารถเรียงได้ 2 แบบ

คือ เรียงจากน้อยไปมาก และเรียงจากมากไปน้อย

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.17 จงเขียนโปรแกรมเพื่อทำการเรียงลำดับข้อมูลโดยใช้วิธีการเรียงลำดับแบบฟองสบู่

```
1   #include <iostream>
2   using namespace std;
3
4   void display (int [], int);
5   void Bubble_sort (int [], int);
6   int min_item;
7   int size=10;
8
```

```
9   int main()
10  {
11  int data[] = {12, 48, 7, 65, 59, 61,
12  41,93,74,63 };
13  cout << "Data Before Sorting: ";
14  display (data, size);
15  Bubble_sort(data, size);
16  cout << "Data After Sorting: ";
17  display (data, size);
18  system("pause");
19  return 0;
20  }
```

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.17 จงเขียนโปรแกรมเพื่อทำการเรียงลำดับข้อมูลโดยใช้วิธีการเรียงลำดับแบบฟองสบู่

```
21  void Bubble_sort(int data[], int size)
22  {
23     for (int i = 0 ; i < (size - 1); i++)
24     {
25        for (int j = 0 ; j < (size - i – 1); j++)
26        {
27           if (data[j] > data[j+1])
28           {
29             swap (data[j+1], data[j]);
30           }
31        }
32     }
33  }
34
```

```
35  void display (int data[], int size)
36  {
37       for (int i = 0; i < size; ++i)
38       {
39         cout << data[i] << " ";
40       }
41       cout << endl;
42  }
```

9.7 การเรียงลำดับ(ต่อ)

ผลการทำงานของโปรแกรม

Data Before Sorting: 12 48 7 65 59 61 41 93 74 63

Data After Sorting: 7 12 41 48 59 61 63 65 74 93

9.7 การเรียงลำดับ(ต่อ)

ตัวอย่างที่ 9.18 จงแสดงขั้นตอนการเรียงลำดับข้อมูลจากน้อยไปหามากด้วยวิธีเรียงลำดับแบบฟองสบู่

data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
12	48	7	65	59	61	41	93	74	63

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 1 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	12	48	7	65	59	61	41	93	74	63
2	12	7	48	65	59	61	41	93	74	63
3	12	7	48	65	59	61	41	93	74	63
4	12	7	48	59	65	61	41	93	74	63
5	12	7	48	59	61	65	41	93	74	63
6	12	7	48	59	61	41	65	93	74	63
7	12	7	48	59	61	41	65	93	74	63
8	12	7	48	59	61	41	65	74	93	63
9	12	7	48	59	61	41	65	74	63	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 2 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	48	59	61	41	65	74	63	93
2	7	12	48	59	61	41	65	74	63	93
3	7	12	48	59	61	41	65	74	63	93
4	7	12	48	59	61	41	65	74	63	93
5	7	12	48	59	41	61	65	74	63	93
6	7	12	48	59	41	61	65	74	63	93
7	7	12	48	59	41	61	65	74	63	93
8	7	12	48	59	41	61	65	63	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 3 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	48	59	41	61	65	63	74	93
2	7	12	48	59	41	61	65	63	74	93
3	7	12	48	59	41	61	65	63	74	93
4	7	12	48	41	59	61	65	63	74	93
5	7	12	48	41	59	61	65	63	74	93
6	7	12	48	41	59	61	65	63	74	93
7	7	12	48	41	59	61	63	65	74	93
8	7	12	48	41	59	61	63	65	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 4 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	48	41	59	61	63	65	74	93
2	7	12	48	59	41	61	65	63	74	93
3	7	12	48	59	41	61	65	63	74	93
4	7	12	48	41	59	61	65	63	74	93
5	7	12	48	41	59	61	65	63	74	93
6	7	12	48	41	59	61	65	63	74	93
7	7	12	48	41	59	61	63	65	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 5 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	48	41	59	61	63	65	74	93
2	7	12	48	59	41	61	65	63	74	93
3	7	12	48	59	41	61	65	63	74	93
4	7	12	48	41	59	61	65	63	74	93
5	7	12	48	41	59	61	65	63	74	93
6	7	12	48	41	59	61	65	63	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 6 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	48	41	59	61	65	63	74	93
2	7	12	48	41	59	61	65	63	74	93
3	7	12	41	48	59	61	65	63	74	93
4	7	12	41	48	59	61	65	63	74	93
5	7	12	41	48	59	61	65	63	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 7 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	41	48	59	61	65	63	74	93
2	7	12	41	48	59	61	65	63	74	93
3	7	12	41	48	59	61	65	63	74	93
4	7	12	41	48	59	61	65	63	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 8 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	41	48	59	61	65	63	74	93
2	7	12	41	48	59	61	65	63	74	93
3	7	12	41	48	59	61	65	63	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 8 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	41	48	59	61	65	63	74	93
2	7	12	41	48	59	61	65	63	74	93
3	7	12	41	48	59	61	65	63	74	93

9.7 การเรียงลำดับ(ต่อ)

รอบที่ 9 ทำการเปรียบเทียบ 2 ค่าที่อยู่ติดกันแล้วสลับเพื่อให้เรียงลำดับ

รอบที่	data [0]	data [1]	data [2]	data [3]	data [4]	data [5]	data [6]	data [7]	data [8]	data [9]
1	7	12	41	48	59	61	65	63	74	93
2	7	12	41	48	59	61	65	63	74	93

มหาวิทยาลัยราชภัฏนครปฐม