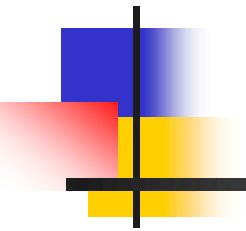
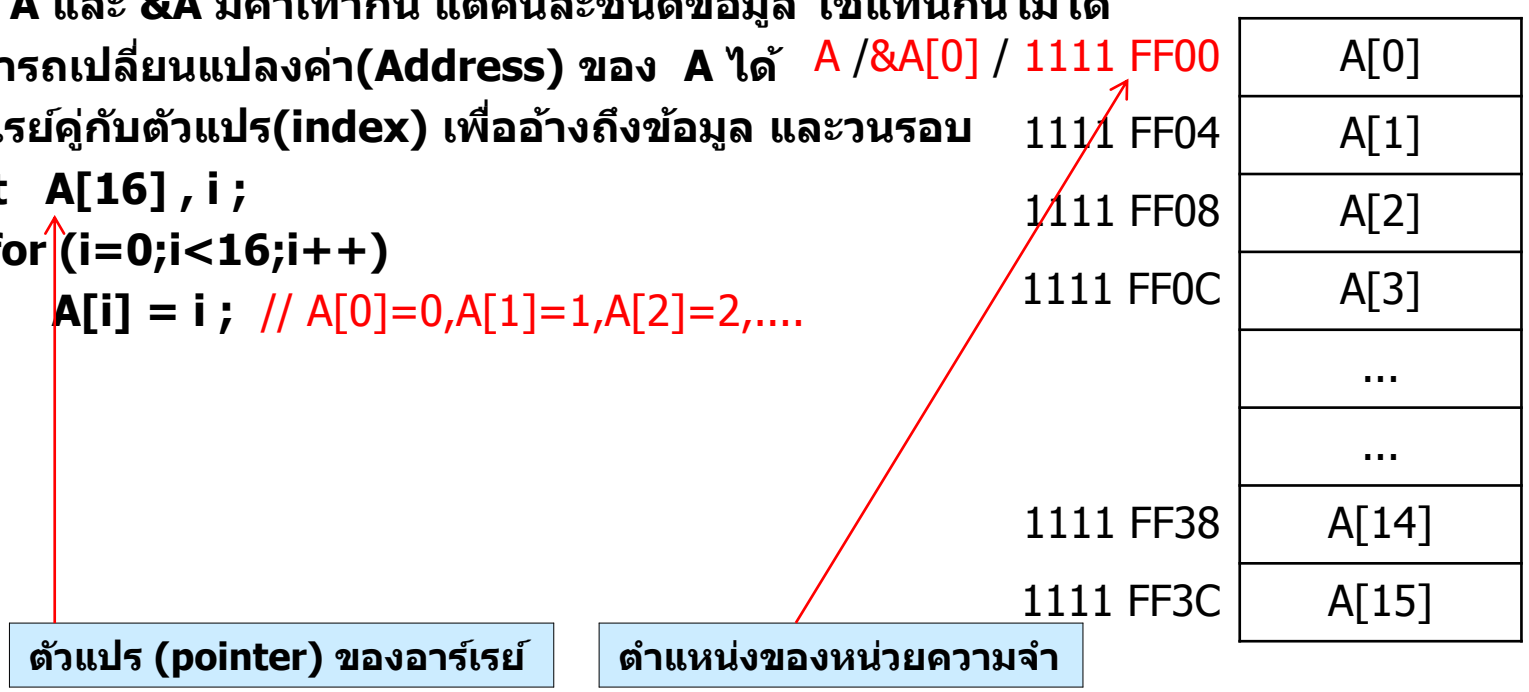


Chapter 5

Array

Array, Address & Pointer

- ตัวแปร ARRAY ใช้เมื่อต้องการจองตัวแปรเป็นจำนวนมากในชื่อเดียวกัน ต้องมีการระบุจำนวนตัวแปรที่ต้องการจองใช้งานด้วย เช่น
  - **int A[16] ;** //จองตัวแปรอาร์เรย์ชื่อ A เป็น int 16 ตัว ขนาด 16*4 = 64 bytes
- ในภาษาซี ตัวแปร ARRAY จะถูกกำหนดให้เป็น Address เริ่มต้นของกลุ่มข้อมูลในหน่วยความจำ แต่ถ้าต้องการอ้างถึงข้อมูลย่อยที่เก็บในอาร์เรย์ ต้องระบุตำแหน่งที่เก็บของตัวแปรย่อยด้วย [index]
- ค่าของ A และ &A มีค่าเท่ากัน แต่คนละชนิดข้อมูล ใช้แทนกันไม่ได้
- ไม่สามารถเปลี่ยนแปลงค่า(Address) ของ A ได้
- ใช้อาร์เรย์คู่กับตัวแปร(index) เพื่ออ้างถึงข้อมูล และวนรอบ
  -
```
int   A[16] , i ;
  for (i=0;i<16;i++)
      A[i] = i ;  // A[0]=0,A[1]=1,A[2]=2,....
```



Array, Address & Pointer

- สามารถจองตัวแปร pointer (32 bits) มาชี้ที่ตำแหน่งอาร์เรย์ได้
  - `int A[16] , i, *x;` ← ตัวแปร (pointer) ของอาร์เรย์
    `x = A ; // x = &A[0]`
    `for (i=0;i<16;i++)` ← x[0]-->A[0], x[1]-->A[1], ...
    `x[i] = i ; //A[0]=0,A[1]=1,A[2]=2,....`

    `x = &A[5];` ← x[0]-->A[5], x[1]-->A[6], ...
    `for (i=0;i<=3;i++)`
    `x[i] = i ; //A[5]=0, A[6]=1, A[7]=2, A[8]=3`

1111 FEF8	x
1111 FEFC	i
A /&A[0] / 1111 FF00	A[0]
1111 FF04	A[1]
1111 FF08	A[2]
1111 FF0C	A[3]
	...
	...
1111 FF38	A[14]
1111 FF3C	A[15]

- การเพิ่มค่าของ pointer ทีละ 1 เป็นการอ้างถึงตำแหน่งข้อมูลตัวถัดไป
  - `*(x+1), *(x+i) , *x++ , *++x` // ถ้าใช้ *x+1 จะเป็นการบวกค่าในอาร์เรย์.

ถ้า x = A;
*(x+0)->A[0], *(x+1)-->A[1], *(x+2)-->A[2], ...

```
x = A;
for (i=0;i<16;i++)
    printf("%d\n", *(x+i));  //แสดงค่า A[0]..A[15] ไม่เปลี่ยนค่า x
x = A;
for (i=1;i<=16;i++)
    printf("%d\n", *x++);  //แสดงค่า A[0]..A[15] เปลี่ยนค่า x ไปเรื่อยๆ
```

ถ้า x = A;
*x+0-->A[0]+0, *x+1-->A[0]+1, *x+2-->A[0]+2

แสดงค่า *x ก่อน แล้วจึงเพิ่มค่า x+1

ARRAY - Variable

- ARRAY 1 มิติใช้เมื่อต้องการเก็บข้อมูลจำนวนมาก ไว้ในตัวแปรตัวเดียว เพื่อประมวลผลด้วยวิธีการ ซ้ำๆ กัน
- การจองตัวแปร ARRAY ให้กำหนดจำนวนสูงสุดที่โปรแกรมสามารถทำงานได้ เช่น

```
double  data[200] ;
```

จองตัวแปรชื่อ data ใช้เก็บตัวเลขจำนวนจริงไม่เกิน 200 ตัว

- ตัวแปร ARRAY ในภาษาซีจะเป็นตัวแปรประเภท address ที่ชี้ไปยังที่เก็บข้อมูลทั้งชุด

ตัวแปร data เป็น address เริ่มต้นของข้อมูลที่คือ data[0] จนถึง data[199]

- ใช้งานจริง ต้องมีตัวนับจำนวนข้อมูลที่มีอยู่จริง และตัวนับรอบทั่วไปเพื่อใช้วนรอบ

```
int   count , i ;
```

ใช้งานจริงอาจมีข้อมูลไม่ถึง 200 ตัวจึงต้องมีตัวนับ count เพื่อนับ และใช้ i เป็นตัววนรอบ เพื่อให้เห็นข้อมูลตัวแรกจนถึงตัวสุดท้าย

- สามารถระบุข้อมูลตัวใดตัวหนึ่งในอาร์เรย์ได้ที โดยอ้างชื่อตัวแปรแล้วตามด้วยตำแหน่งอ้างอิง [index] ซึ่งจะใช้งานได้เหมือนตัวแปรทั่วไป เช่น data[0], data[1], data[2], …., data[i]; ไม่ถือว่าเป็น address

```
scanf("%lf", &data[i]) ;
printf("%lf", data[i]) ;
```

คำสั่งทั่วไปจะกระทำกับข้อมูลทีละตัว จึงต้องระบุตัวที่ต้องการทำงาน โดยใช้ตัวชี้ [i]

- ใช้การวนรอบ เพื่อกระทำกับข้อมูลทีละตัว จนครบทุกตัว เช่น

```
for  (i=0 ; i< count ; i++)
    sum = sum + data[i] ;
```

จะมองเห็นข้อมูลรอบละ1ตัว คือ data[i] เมื่อ i มีค่าตั้งแต่ตัวแรกจนถึงตัวสุดท้าย ต้องเขียนโปรแกรมให้ประมวลผลข้อมูล data[i] ทุกตัวตามที่โจทย์ต้องการ

Array & Parameter

- การกำหนดค่าพารามิเตอร์ที่เป็นอาร์เรย์ทั้งตัวในการสร้างฟังก์ชัน เพื่อใช้รับส่งข้อมูล พารามิเตอร์ที่ใช้แทนอาร์เรย์จะถูกบังคับให้เป็นพอยท์เตอร์ที่ชี้ไปยังตำแหน่งเริ่มต้นของอาร์เรย์ จึงอาจจองชื่อพารามิเตอร์ที่เป็น pointer (ใช้ * นำหน้าชื่อตัวแปร) เพื่อใช้แทนอาร์เรย์ได้ทันที (ไม่นิยม เพราะอาจเข้าใจผิด คิดว่าเป็นตัวแปร reference ตัวเดียว) หรือใช้วิธีจองชื่อพารรามิเตอร์ที่จะใช้เป็นอาร์เรย์โดยไม่จำเป็นต้องกำหนดขนาดข้อมูลในวงเล็บ

```
void  Read_Data (double *data, int *count);
void  Read_Data (double  data[], int *count);
```

พารามิเตอร์แบบ pointer เพื่อชี้ไปยังอาร์เรย์

พารามิเตอร์แบบอาร์เรย์ไม่ระบุขนาด

- เนื่องจากตัวแปรอาร์เรย์เป็น address อยู่แล้ว ดังนั้นการส่งผ่านข้อมูลไปยังโปรแกรมย่อยจึงเป็นการส่งค่าแบบ pass by reference เท่านั้น และจะเป็นการส่งค่าอาร์เรย์ทั้งตัวโดยไม่ต้องใส่ &นำหน้า

```
Read_Data ( data, &count);
```

เรียกใช้ฟังก์ชันที่กระทำกับข้อมูล data ทั้งก้อน ไม่ต้องมี index

- ฟังก์ชันที่มีพารามิเตอร์แบบ reference ที่กระทำกับข้อมูลตัวเดียว ถ้านำมาใช้กับอาร์เรย์ เวลาเรียกใช้ต้องจะต้องระบุตำแหน่งอ้างอิงที่จะกระทำด้วย

```
void swap(double *a, double *b) // ฟังก์ชัน swap ใช้สลับค่าระหว่างตัวแปร 2 ตัว
{ double c;
      c = *a ;  *a = *b ; *b = c;
}
      swap (&data[i], &data[j]);  // ต้องการสลับค่าที่อยู่ในตำแหน่ง [i] กับ [j]
```

ถ้า reference กับข้อมูลตัวเดียวต้องมี index และ &

Initial Array

```
int  data[100], i ;
```

- สร้างตาราง 100 ตัว ที่มีค่าเริ่มต้นทุกตัวเป็น 0

```
      for (i=0; i<100; i++)
            data[i] = 0;
```

- สร้างตาราง 10 ตัว มีค่าเริ่มต้นเป็นในตารางเป็น i!

```
      for (i=0; i<10; i++)
            data[i] = factorial(i) ;
```

ต้องสร้างฟังก์ชัน factorial() ไว้ก่อนแล้ว

- ฟังก์ชันสร้างตารางของ fibonacci ถ้าสร้างได้ ให้ return 1 ไม่ได้ให้ return 0

```
int Create_Fibo_Array(int fibo[], int max)
 { int  i ;
   if (max >1 && max <= 40)
      {  fibo[0] = 0;  fibo[1] = 1;
        for (i=2; i <= max ; i++)
            fibo[i] =  fibo[i-1] + fibo[i-2];
        return 1;
      }
  else
     return 0;
}
```

ถ้า max อยู่ระหว่าง 2..40 แสดงว่าสร้างได้ return 1
ถ้า ไม่อยู่ในช่วง สร้างไม่ได้ return 0

Read Data to ARRAY

- ตัวอย่าง การวนรอบอ่านข้อมูลจำนวนจริง 0-100 เพิ่มทีละตัวไปเรื่อยๆ จนกว่าจะเจอเลข สมมุติว่าหมดข้อมูลจึงหยุด (-1)

```
void  Read_Data_Until_Mark(double data[], int *count )
 { double  a  ;
      do  {  printf("Enter Data #%d (-1 to END) ", *count+1);
             a = get_double(-1,100);
             if (a >= 0)
              {data[*count] = a;
               *count = *count +1;
              }
       } while (a >= 0 ) ;
}
```

ในคำถาม อาจต้องมีการชดเชยตัวเลขที่แสดงผล

สร้างฟังก์ชันเพื่ออ่านเลขจำนวนจริง

ถ้า a ที่อ่านได้มีค่ามากกว่าหรือเท่ากับศูนย์ แสดงว่าเป็นข้อมูล ให้เอาค่า a ไปเก็บใน data[*count] แล้วเพิ่มจำนวนตัวนับอีก 1

ไม่สามารถใช้ *count++ แทนการบวก 1

กำหนดให้ข้อมูลที่น้อยกว่าศูนย์คือรหัสการจบ

```
double get_double(double min, double max)
{ double  a;
      while ((scanf("%lf",&a)!=1) || a<min|| a>max))
      {   printf("Invalid data input\n");
          printf("Please Enter between %g to %g = ",min,max);
          rewind(stdin);
       }
       return a;
}
```

Read Data to ARRAY

- อ่านข้อมูลที่พิมพ์เข้า ต่อเนื่องจนกว่าจะพิมพ์ผิด

```
void  Read_Data_Array(double data[], int *count )
 { double  a  ;
       printf("\nEnter Data ");
       while (scanf("%lf",&a) ==1)
        {  data[*count] = a;
           printf("data[%d] = %g\n", *count, data[*count]);
            *count = *count + 1;
         }
       rewind(stdin);
       printf("End of input\n");
}
```

ขณะที่ยังอ่านได้ (ทีละตัว)

นำข้อมูลที่อ่านไปเก็บ และแสดงผล

เมื่ออ่านผิด ให้ลบ buffer

Print Data in ARRAY

ตัวอย่าง การวนรอบเพื่อแสดงค่าที่เก็บไว้ใน data ตั้งแต่ตัวแรกจนถึงตัวสุดท้าย

```
void  Print_All_Data(double data[], int count )
 { int  i ;
      for  (i = 0; i < count ; i++)
       {  printf("Data[%d] = %g\n", i, data[i]);
       }
      printf("End of Data Press any key\n ");
      getch();
}
```

ถ้าข้อมูลมากจนอ่านไม่ทัน เพิ่มคำสั่งให้หยุดรอ (ทุก 20 บรรทัด)

```
if ( i % 20 == 19)
   { printf("\n Press any key to continue");
     getch();  }
```

```
//หยุดรอ char  s[10];   gets(s);
```

```
void main (void)
{   double data[200];
    int  count = 0 ;
    Read_Data_Array(data, &count);
    Print_ All_ Data_in_Array (data, count );
}
```

Find Mean, Sd

```
double  Find_Mean (double data[], int count )
{ int i;
  double   sum, mean ;
    for (sum = 0, i=0; i<count; i++)
          sum = sum+data[i];
    mean  = sum/count;
    return mean;
}
double  Find_SD (double data[], int count )
{ int i;
  double   sum1 = 0 , sum2 = 0, mean,  sd;
    for (sum1=0, sum2=0, i=0; i<count; i++)
        { sum1 = sum1+data[i];
          sum2 = sum2+data[i]*data[i];
        }
    mean  = sum1/count;
    sd = sqrt(sum2/count - pow(mean,2) );
    return sd ;
}
```

$$mean = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$SD = \sqrt{\frac{\sum x_i^2}{n} - mean^2}$$

ตัวอย่างการเรียกใช้

```
Read_Data_Until_Mark(data, &count);
mean  = Find_Mean(data, count);
sd  = Find_SD(data, count);
printf(....แสดงค่า mean & sd .....);
```

ใช้วนรอบร่วมกัน ในการหา sum,sum2

Find minimum data in ARRAY

- ตัวอย่าง การวนรอบเพื่อหาค่า และตำแหน่งที่มีข้อมูลน้อยที่สุด ที่เก็บอยู่ในตัวแปร data
- ข้อกำหนดฟังก์ชัน ให้ return ค่าที่น้อยที่สุดที่อยู่ในตัวแปร data

```
double  Find_Min (double data[], int count)
 { double  min;
    int i  ;
      min = data[0];
      for (i=1; i<count; i++)
          if (data[i] < min)
              min = data[i] ;
    return  min ;
 }
```

เมื่อเริ่มต้น ค่าตัวแรก จะเป็นค่าต่ำสุด

วนรอบตรวจสอบจนครบทุกตัว
สามารถเริ่มต้นที่ i=1 (ตัวที่สอง)ได้ เนื่องจากเห็นตัวแรกแล้ว

ถ้าข้อมูลตัวไหนมีค่าน้อยกว่าค่าต่ำสุด
ให้เปลี่ยนไปจำค่าตัวนั้นแทน

```
void main (void)
{   double data[200] , min ;
    int  count = 0 ;
    Read_Data_Until_Mark(data, &count);
    min  = Find_Min(data, count);
    printf("\nMinimum of data in array = %g",min);
}
```

Find maximum data in ARRAY

- ตัวอย่าง การวนรอบเพื่อหาค่าสูงสุด และตำแหน่งที่เก็บ ในตัวแปร data
- ข้อกำหนดฟังก์ชัน ให้ return ตำแหน่งที่เก็บค่ามากที่สุด และค่าที่มากที่สุด

```
int  Find_Max (double data[], int count , double *max )
 {    int  i  , pos ;
      *max = data[0];
      for (i=1; i<count; i++)
          if (data[i] > *max)
           { pos = i;   //  ตำแหน่งที่เก็บค่ามากที่สุด
             *max = data[i] ; }  // ค่าที่มากที่สุด
      return pos ;
 }
```

วนรอบตั้งแต่ตัวที่สอง จนถึงตัวสุดท้าย

```
void main (void)
{   double data[200] , max ;
    int  count = 0, i ;
    Read_Data_Array(data, &count);
    i  = Find_Max(data, count, &max);
    printf("\nMaximum of data in array is data[%d] = %g", i , max);
}
```

Find Min, Max, Mean, Sd

- ตัวอย่าง ฟังก์ชันหาค่าทางสถิติทั้ง 4 ตัวแล้วส่งกลับ (2 input , 4 output)

```
void  Calc_Statistic (double data[], int count,
                      double *min, double *max, double  *mean, double *sd )
{ int i;
  double   sum1 = 0, sum2 = 0;
  *min = *max = data[0] ;
    for (i=0; i<count; i++)
        { if (data[i] < *min)
            *min = data[i] ;
          if (data[i] > *max)
            *max = data[i];
          sum1 = sum1+data[i];
          sum2 = sum2+data[i]*data[i];
        }
   *mean  = sum1/count;
   *sd = sqrt(sum2/count - pow(*mean,2));
}
```

กำหนดค่าเริ่มต้นของ sum1, sum2, min, max ตอนจองตัวแปร หรืออาจกำหนดในวนรอบก็ได้
`for(sum1=0,sum2=0,min=data[0],max=data[0],i=0;`

สามารถใช้วนรอบเดียวกัน เพื่อคำนวณค่าที่ต้องการทุกค่า พร้อมกันได้

ตัวอย่างการเรียกใช้

```
Calc_Statistic (data, count, &min, &max, &mean, &sd);
```

Search Data

- ตัวอย่างการค้นหาข้อมูล โดยส่งค่าที่ต้องการค้น และตำแหน่งเริ่มต้น แล้ว return ตำแหน่งข้อมูลตัวแรกที่เจอ ถ้าไม่เจอ return -1 ;

```
int Search_index(double data[] , int count , double key, int start )
 { int  i=start ;
     while ( (i<count)&&(data[i] != key) )
            i++;          //  for (i=start; i <count && data[i] != key ; i++) ;
     if (i<count)
          return i ;
     else
          return -1 ;
}
```

ข้อมูลยังไม่หมด

ยังไม่เจอ

ตำแหน่งที่เริ่มค้น

ถ้าค้นเจอ

ไม่ต้องทำอะไร

return ตำแหน่งที่ค้นเจอ 0..count-1

ถ้าค้นไม่เจอ

```
void Search_all_print (double data[], int count, double key)
{ int pos=-1, start = -1;
      do { pos = Search_Index(data, count, key, start+1);
           if  (pos>=0)
              { printf("Found data[%d] = %g \n" , pos, data[pos]);
                start = pos+1;   }
         } while (pos !=-1);
     if (start==-1)
        printf("Can't found your data %d\n", key);
}
```

ยังไม่ค้น คือไม่เจอ start = -1

ถ้าค้นเจอ เปลี่ยนค่า start เพื่อค้นตัวถัดไป

หลุดจากวนรอบ ถ้า startไม่เปลี่ยน แสดงว่าค้นไม่เจอ

Delete Data

- ตัวอย่างการลบข้อมูลออกไป 1 ตัว
  - ข้อมูลเดิมเก็บอยู่ในตัวแปร data จำนวน count ตัว
  - ต้องการลบข้อมูลในตำแหน่งที่กำหนด
  - ถ้าข้อมูลอยู่ในช่วงที่ลบได้ให้ return 1 ลบไม่สำเร็จ return 0
  - ใช้วิธีแทนที่ข้อมูลตำแหน่งที่จะลบ ด้วยข้อมูลตัวถัดไป เพื่อรักษาลำดับของข้อมูล แล้วลดจำนวนข้อมูลที่มีอยู่ลง 1 ตัว

```
int  Delete_at(double data[], int *count, int pos )
 { int  i , success = 0 ;
    if  (pos >=0 && pos <*count)
      { for (i=pos; i<*count-1; i++)
            data[i] = data[i+1];
        *count = *count - 1;
        return 1 ;
      }
    else
        return 0;
}
```

ถ้าลบสำเร็จ ค่า count จะเปลี่ยน

ถ้าตำแหน่งไม่อยู่ในช่วง ห้ามลบ return 0

วนรอบ copy ข้อมูลขึ้นไปทับตัวที่อยู่ข้างหน้าทีละตัว จนครบทุกตัว

ทำเสร็จ ต้องลดจำนวนข้อมูลรวมลง 1 ตัว

ตัวอย่างการเรียกใช้

```
success = Delete_Index(data, &count, pos);
if (success==1) printf("Deletion complete\n");
   else printf("Can't Delete position %d\n", pos);
```

Insert Data

- ตัวอย่าง การเพิ่มข้อมูล 1 ตัว แทรกลงในตำแหน่งที่กำหนด
  - ถ้าตำแหน่งที่แทรกไม่อยู่ในช่วงมีข้อมูล ให้ return 0
  - เลื่อนข้อมูลตัวสุดท้ายลงข้างล่าง 1 ตัว ทีละตัวจนถึงตำแหน่งที่ต้องการ
  - ใส่ข้อมูลลงในตำแหน่งที่ต้องการ

```
int Insert_at(double data[], int *count, double key,  int pos )
 { int  i ;
    if  (pos >=0 && pos <=*count)
       { for (i=*count; i>pos ; i--)
              data[i] = data[i-1];
         data[i] = key;
        *count = *count + 1;
        return 1 ;
       }
    else
        return 0;
}
```

ถ้าตำแหน่งไม่อยู่ในช่วง ห้ามลบ return 0

วนรอบ copy ข้อมูลขึ้นไปทับตัวที่อยู่ข้างหน้าทีละตัว จนครบทุกตัว

ทำเสร็จ ต้องลดจำนวนข้อมูลรวมลง 1 ตัว

ตัวอย่างการเรียกใช้

```
success = Delete_Index(data, &count, pos);
if (success==1) printf("Deletion complete\n");
   else printf("Can't Delete position %d\n", pos);
```

Scan Sort

- ใช้เทคนิคการกำหนดจุดเริ่มต้นเพื่ออ้างอิงทีละตัว แล้ววนรอบและเปรียบเทียบกับตัวถัดไปเรื่อยๆ ถ้าพบว่าอยู่ในตำแหน่งไม่เหมาะสมให้สลับค่ากันจนครบทุกตัว จะทำให้ได้ค่าที่เหมาะสมสำหรับตำแหน่งนั้น แล้ววนรอบเปลี่ยนตำแหน่งเริ่มต้นใหม่ ทำซ้ำจนกว่าจะเรียงลำดับได้ครบทุกตัว

```
void  Sort_Min_to_Max(double data[], int count )
 { double  x  ;
    int  i, j;
     for (i = 0; i<count-1; i++)
          for (j=i+1; j<count; j++)
               if (data[j] < data[i])
                    { x = data[i];
                      data[i]  =  data[j];
                      data[j] = x; }
}
```

วนรอบ i (reference) เพื่อเรียงตั้งแต่ตัวแรกจนถึงตัวก่อนสุดท้าย (ไม่จำเป็นต้องเรียงตัวสุดท้าย)

วนรอบ j (scan) หาตำแหน่งเหมาะสมที่จะต้องนำมาสลับค่าเพื่อเรียงลำดับ

ถ้าต้องการเรียงจากมากไปน้อยให้ใช้
if (data[j] > data[i])

สลับค่าข้อมูลที่อยู่ในตำแหน่ง data[i] กับ data[j] โดยใช้ x เป็นตัวพักข้อมูล

Selection Sort

- ตัวอย่างการเรียงลำดับข้อมูลตัวเลขจากน้อยไปมาก Ascending Sort
  - หาค่าต่ำสุดในแต่ละรอบ นำมาสลับตำแหน่งที่เหมาะสมเพื่อเรียงลำดับ
  - วิธีนี้จะมีการสลับค่าครั้งเดียวในแต่ละรอบ ทำให้เรียงได้เร็วขึ้น

```
void  swap(double *x, double *y)
{ double z;
     z = *x ;  *x = *y;   *y = z;
}
void  Sort_Min_to_Max(double data[], int count )
 { double  x ;
    int  i, j, min;
         for (i = 0; i<count-1; i++)
         {  min = i ;
            for (j=i+1; j< count; j++)
                if (data[j] < data[min])
                    min = j ;
            swap (&data[i], &data[min]);
         }
}
```

สร้างฟังก์ชัน swap() เพื่อใช้สลับค่าตัวแปร

วนรอบ i เพื่อเรียงตั้งแต่ตัวแรกจนถึงตัวก่อนสุดท้าย (ไม่จำเป็นต้องเรียงตัวสุดท้าย)

ในแต่ละรอบของ i

จะวนรอบใช้ตัวแปร min หาตำแหน่งข้อมูลที่มีค่าน้อยที่สุด (data[min]) ที่เหลืออยู่ เพื่อนำไปสลับตำแหน่งกับ data[i]

สลับค่าข้อมูลที่อยู่ในตำแหน่ง data[i] กับ data[min] โดยใช้ ฟังก์ชัน swap()

Search Position

- ตัวอย่างการค้นหาตำแหน่งที่เหมาะสมกับข้อมูลที่กำหนด กรณีข้อมูลเรียงลำดับ
  - ถ้าค้นเจอ return ตำแหน่งที่ค้นเจอ
  - ถ้าค้นไม่เจอ return ตำแหน่งที่เหมาะสำหรับเก็บข้อมูลตัวนั้น
  - ต้องตรวจสอบตำแหน่งที่ return ซ้ำว่าเจอหรือไม่
- ค้นจากข้อมูลตัวแรกไปจนกว่าจะเจอตำแหน่งที่เหมาะสม

```
int Search_forward(double data[] , int count , double key, int *found )
 { int  i ;
      *found = 0;
      if (count == 0)
          return 0;
        else { for (i=0; i <count && data[i] < key ; i++) ;
                    if (data[i]==key)
                        *found = 1;
               }
      return i ;
}
```

```
i = Search_forward(data,count,key,&found);
if (found==0) printf("%lf Not found in position ",key);
   else printf("%lf Found in  position",key);
printf ("%d\n",i);
```

2-Dimensional Arrays (MATRIX)

- MATRIX กับ ARRAY 2 มิติ
- การจองตัวแปร matrix ให้จองเป็นอาร์เรย์ 2 มิติของตัวเลข เช่น

```
double   M[10][10] ;
```

มีการจองตัวแปร 10x10 = 100 ตัว

- ต้องกำหนดตัวแปร row และ col เพื่อระบุขอบเขตที่ใช้งานจริง

```
int   row, col ;
```

- ใช้การวนรอบซ้อน 2 ชั้น เพื่อกระทำกับข้อมูลทีละตัวจนครบ เช่น

```
for  (i=1; i<= row; i++) // เริ่มใช้ที่ 0 หรือ 1 แล้วแต่ผู้ใช้กำหนด
   { รอบการประมวลผลของ i ทีละแถว
     for ( j = 1; j<= col ; j++)
        { รอบของการประมวลผล j  ทีละหลัก จนครบทุกหลัก
          ประมวลผลข้อมูล  M[i][j] }
   }
```

- การกำหนดค่าพารามิเตอร์ที่เป็นอาร์เรย์ มากกว่า 1 มิติในฟังก์ชัน ให้ละเว้นขนาดข้อมูลในวงเล็บช่องแรก และใช้ได้ทั้ง input , output

```
void  Process_Matrix (double M[][10], int row, int col)
```

- การเรียกใช้ฟังก์ชันที่ต้องส่งค่าอาร์เรย์ ให้ใช้ชื่อตัวแปรอาร์เรย์ โดยไม่ต้องกำหนดขนาด

```
Process_Matrix ( M,  row,  col);
```

ถ้าส่งเป็นพอยต์เตอร์ *M เวลาใช้งานจะต้องคำนวณตำแหน่งใหม่เป็น *(M+(i*10)+j) และส่งตัวแปรด้วย &M[0][0]

การอ่านข้อมูลใส่ใน Matrix

```
void Read_Matrix(char *name, double M[][10] , int *row, int *col)
{ int i , j ;
   printf("Enter no. row and column of matrix %s ",name);
   scanf("%d%d",&*row, &*col);
   printf(" input elements of matrix %s [%dx%d]\n", name, *row, *col);
  for (i =1; i <= *row; i++)
      for (j=1; j<= *col; j++)
          { printf("\n%s[%d,%d] = ",name,i,j);
            scanf("%lf", &M[i][j]); }
  Print_Matrix (name, M, *row, *col);
}
```

ตั้งชื่อเพื่อการแสดงผล

อ่านข้อมูล 2 ตัว คือ row และ col ม่ได้ป้องกัน error

ตั้งรอบเมตริกซ์ row แถวแรกจนถึงแถวสุดท้าย

ในแต่ละ row ตั้งรอบคอลัมน์แรกจนถึงคอลัมน์สุดท้าย

ไม่ได้ป้องกันการอ่านผิด

แสดงผลเมตริกซ์ที่อ่านได้ในฟังก์ชันนี้

```
if (select == 1)
{  printf("Enter 1 for read MA\n");
   printf("Enter 2 for read MB\n");
   choice = readint(0,2);
   if  (choice == 1)
      Read_Matrix("MA", Ma, &rowa, &cola);
   else if (choice == 2)
      Read_Matrix("MB", Mb, &rowb, &colb);
```

```
Enter no. row and column of matrix MA 3 4
input elements of matrix MA[3x4]
MA[1,1] =  3      7      14      9
.....
```

การแสดงผล Matrix

```
void  Print_Matrix (char *name, double M[][10], int row, int col )
{ int  i , j ;
    printf("\n MATRIX |%s| =\n", name);
    for (i =1; i <= row; i++)
    {    for (j=1; j<= col; j++)
            {printf("%7g", M[i][j] );}
         printf("\n");
    }
    printf("\n Press any key ") ;
    getch() ;
}
```

```
MATRIX |MA|=
      3      7     14      9
     10     11     13      3
      4      3      8     11
```

ขึ้นบรรทัดใหม่ เมื่อพิมพ์ค่าครบ 1 บรรทัด (col)
แล้วจึงเปลี่ยวนรอบ i รอบต่อไป

```
Print_Matrix("MA", Ma, rowa, cola);
Print_Matrix("MB", Mb, rowb, colb);
Print_Matrix("MC", Mc, rowc, colc);
```

การเติม Matrix ด้วยค่าคงที่

- เติมค่าเริ่มต้น a ให้กับ เมตริกซ์

```
void Fill_Matrix(double M[][10] , int row, int col, double  a)
{ int i , j ;
  for (i =1; i <= row; i++)
      for (j=1; j<= col; j++)
          {M[i][j] = a;}
}
```

```
Fill_Matrix(Ma, rowa, cola, 1);
Print_Matrix ("MA", Ma, rowa, cola ) ;
```

- เมตริกซ์จัตุรัส ที่มีตัวเลขบนเส้นทแยงมุมเท่ากับ 1 ที่เหลือเป็น 0

```
int  Fill_Identity_Matrix( double M [][10], int row, int col)
{int  i ,  j;
    if  (row == col)
      { for (i =1; i <= row; i++)
            for (j=1; j<= col; j++)
                if (i==j) M[i][j] = 1 ; else  M[i][j] = 0;
        return 1;
      }
   else  return 0;
}
```

```
check = Fill_Identity_Matrix(Ma, rowa, cola);
if (check==1)
    Print_Matrix ("MA", Ma, rowa, cola ) ;
else
    printf("Can't create Identity Matrix");
```

การสลับเปลี่ยน Transpose Matrix

```
void Transpose (double M1[][10], int row1, int col1,
                double M2[][10], int *row2, int *col2)
{int i , j ;
    *row2= col1;
    *col2 = row1;
    for (i =1; i <= *row2; i++)
        for (j=1; j<= *col2; j++)
          M2[i][j] = M1[j][i];
}
```

```
Transpose(Ma, rowa, cola, Mc, &rowc, &colc);
Print_Matrix ("Ma", Ma, rowa, cola ) ;
Print_Matrix ("Mt", Mc, *rowc, *colc ) ;
```

$$Ma = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{vmatrix} \qquad Ma^T = Mt = \begin{vmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{vmatrix}$$

การ Copy Matrix

```
void Copy_Matrix( double M1 [][10], int row1, int col1,
                  double M2[][10], int *row2 , int *col2)
{int  i ,  j;
  *row2 = row1;
  *col2= col1;
   for (i =1; i <= *row2; i++)
        for (j=1; j<= *col2; j++)
            M2[i][j] = M1[i][j];
}
```

```
Copy_Matrix(Ma, rowa, cola, Mc, &rowc, &colc);
Print_Matrix ("MA", Ma, rowa, cola ) ;
Print_Matrix ("MC", Mc, *rowc, *colc ) ;
```

การคูณ Matrix ด้วย Scalar

```
void Scalar_Multiplication(double M1[][10], int row1, int col1,
                           double M2[][10], int *row2, int *col2, double  sc)
{ int  i , j ;
        *row2 = row1;
        *col2 = col1;
        for (i =1; i <= *row2; i++)
            for (j=1; j<= *col2; j++)
                M2[i][j] = sc * M1[i][j] ;
}
```

```
printf("Enter Scalar number ");
scanf("%lf",&s)
Scalar_Multiplication(Ma, rowa, cola, Mc, &rowc, &colc, s);
Print_Matrix ("MA", Ma, rowa, cola);
Print_Matrix ("MC", Mc, *rowc, *colc);
```

การบวก หรือ ลบ Matrix

```
int Matrix_Addition(double M1[][10], int row1, int col1,
                    double M2[][10], int row2, int col2,
                    double M3[][10], int *row3, int *col3)
{ int  i , j;
  if (row1 == row2 && col1 == col2)
  { *row3 = row1;
    *col3 = col1;
    for (i =1; i <= *row3; i++)
        for (j=1; j<= *col3; j++)
            M3[i][j] = M1[i][j] + M2[i][j] ;
    return 1;
  }
  else return 0;
}
```

ตรวจสอบว่าบวกกันได้

ถ้าต้องการลบ Matrix ให้เปลี่ยนเครื่องหมายเป็น -

ส่งค่า กลับว่าบวกสำเร็จหรือไม่
0 =บวกไม่ได้ , 1 = บวกสำเร็จ

```
success = Matrix_Addition(Ma, rowa, cola, Mb, rowb, colb, Mc, &rowc, &colc);
if (successs == 1)
{   Print_Matrix ("MA", Ma, rowa, cola ) ;
    Print_Matrix ("MB", Mb, rowc, colc ) ;
    Print_Matrix ("MC", Mc, *rowc, *colc ) ; }
else
    printf("Can't Add Matrix\n");
```

การคูณ Matrix

```
int  Matrix_Multiplication(double M1[][10], int row1, int col1,
                           double M2[][10], int row2, int col2,
                           double M3[][10], int *row3, int *col3)
{ int i , j , m;
  if (col1 == row2)
  { *row3 = row1;
    *col3 = col2;
    for (i =1; i <= *row3; i++)
        for (j=1; j<= *col3; j++)
            for (M3[i][j] = 0, m = 1; m <= col1; m++)
                M3[i][j] = M3[i][j]+M1[i][m]*M2[m][j];
    return 1;
  }
  else
   return  0;
}
```

$$Mc[i][j] = \sum_{m=1}^{col.a} Ma[i][m] * Mb[m][j]$$

```
success  = Matrix_Multiplication (Ma, rowa, cola, Mb, rowb, colb, Mc, &rowc, &colc);
if (success)  // กรณี success มีค่าเป็น 1  ไม่ต้องเขียนเงื่อนไข == 1 ก็ได้
  {Print_Matrix ("MA", Ma, rowa, cola ) ;
   Print_Matrix ("MB", Mb, rowc, colc ) ;
   Print_Matrix ("MC", Mc, *rowc, *colc ) ; }
else
   printf("Can't Multiply Matrix\n ");
```