บทที่ 4 อาร์เรย์ (Array) กับสตริง (String)

ในบทที่ 3 เราได้ศึกษาถึงชนิดของข้อมูล และวิธีใช้เบื้องต้นแล้ว ซึ่งเป็นการใช้ตัวแปรตัวหนึ่งในการเก็บค่าค่าหนึ่ง แต่ในกรณีที่เราเขียนโปรแกรมที่ต้องใช้ข้อมูลที่มีจำนวนค่ามาก ๆ ถ้าหากเราใช้วิธีตามบทที่ 2 เราจะต้องใช้จำนวนตัวแปรเท่าจำนวนค่า ซึ่งเป็นการยุ่งยาก และเสียเวลา เพราะต้องทำการประกาศตัวแปรจำนวนมาก ทั้งยังเป็นการยากต่อการอ้างอิงถึงอีกด้วย อันจะส่งผลต่อการทำซ้ำที่จะกล่าวถึงในบทต่อไป ในกรณีนี้ เราสามารถใช้ตัวแปรหนึ่งในการเก็บข้อมูลจำนวนมากกว่า 1 ขึ้นไป หรือเก็บเป็นชุดข้อมูลได้ ซึ่งเราเรียกข้อมูลชนิดนี้ว่า อาร์เรย์ (Array)

4.1 อาร์เรย์ (Array)

Array เป็นชุดของข้อมูลชนิดเดียวกัน โดยจะใช้หน่วยความจำส่วนที่ติดกันในการบันทึกข้อมูล กล่าวคือ ในการอ้างอิงถึงข้อมูลตัวใดตัวหนึ่งในอาร์เรย์ ทำได้โดยการอ้างอิงจากข้อมูลตัวแรก ซึ่งทำให้สะดวกในการประมวลผลข้อมูลที่ประกอบด้วยข้อมูลหลาย ๆ ตัว ที่ต้องอาศัยการวนลูป เช่น การหาค่ามากสุดจากชุดข้อมูล *n* จำนวน เป็นต้น หรือการเก็บข้อมูลชนิดเดียวกันหลาย ๆ ข้อมูล โดยทำให้ไม่ต้องใช้หลายตัวแปร เช่น การเก็บข้อมูลเกี่ยวกับผลการเรียน เป็นต้น ข้อมูลแต่ละตัวในอาร์เรย์ จะมีชื่อเรียกว่า ตัวประกอบ (element) และจำนวนตัวประกอบในอาร์เรย์มีชื่อเรียกว่า ขนาดของอาร์เรย์ (Array Size)

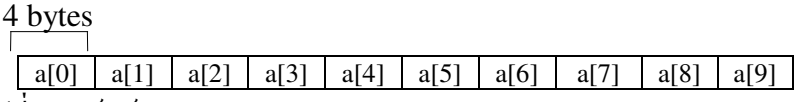
ก่อนทำการใช้อาร์เรย์ ต้องทำการประกาศตัวแปรอาร์เรย์ ซึ่งมีวิธีการประกาศดังนี้

```
Element_data_type array_name[size] ;
```

ยกตัวอย่าง เช่น

```
int    a[10] ;
```

จะเป็นการประกาศอาร์เรย์ที่มีตัวประกอบเป็นข้อมูลชนิด int จำนวน 10 ตัว ดังแสดงในรูปที่ 4.1 อนึ่ง ชื่อของอาร์เรย์ (a ในตัวอย่างนี้เป็นค่าคงที่ที่มีค่าเท่ากับตำแหน่งที่อยู่ของไบต์แรกของตัวประกอบตัวแรกของอาร์เรย์ a) ดังจะกล่าวถึงอย่างละเอียดยิ่งขึ้นในบทที่ 10 พอยเตอร์



รูปที่ 4.1อาร์เรย์ a[10]

การอ้างอิงถึงค่าของตัวประกอบในอาร์เรย์ จะใช้ชื่ออาร์เรย์และดัชนี (index) *ซึ่งดัชนีของตัวประกอบตัวแรกจะเท่ากับ 0* โดยเขียนชื่ออาร์เรย์แล้วตามด้วยวงเล็ก [ ] (Brackets) แล้วใส่ดัชนีของตัวประกอบที่ต้องการภายใน [ ] ยกตัวอย่างเช่น a[3] หมายถึงค่าของตัวประกอบตัวที่ 4 ของอาร์เรย์ a เป็นต้น ดังแสดงในรูปที่ 4.1 เช่นกัน

ตัวอย่างที่ 4.1 : การใช้อาร์เรย์พื้นฐาน

```
#include <stdio.h>
int main()
{
       float  data[5] ;     /* Declare floating point numberอาร์เรย์*/
       float  total, average ;
       data[0] = 34.0 ; data[1] = 27.0 ;data[2] = 45.0 ;
       data[3] = 82.0 ; data[4] = 22.0 ;/*element value assignments */
       total = data[0] + data[1] + data[2] + data[3] + data[4] ;
       average = total / 5.0 ;
       printf("Total %f Average %f\n", total, average) ;
       return 0;
}
```

ผลการรันโปรแกรม

```
Total 210.000000 Average 42.000000
```

ข้อควรระวัง : ดัชนี (index) ของตัวประกอบในอาร์เรย์จะเริ่มนับที่ 0 ดังนั้น ถ้าขนาดของอาร์เรย์เป็น n ดัชนีของตัวประกอบตัวสุดท้ายจะเท่ากับ *n – 1*

การกำหนดค่าเริ่มต้นของอาร์เรย์

ในกรณีของอาร์เรย์ ก็สามารถทำได้เพียงแต่ต้องใช้เครื่องหมายวงเล็บปีกกา ({}; braces) แล้วใส่ค่าคงที่ภายใน { } โดยคั่นแต่ละค่าด้วยเครื่องหมาย , (comma) ยกตัวอย่างเช่น

```
int    vector[3] = {2, 3, 4} ;
```

มีค่าเท่ากับ

```
int    vector[3] ;
vector[0] = 2 ;     vector[1] = 3 ;     vector[2] = 4 ;
```

กฎเกณฑ์ที่เกี่ยวข้องกับการกำหนดค่าเริ่มต้นของอาร์เรย์ได้แก่

- จำนวนค่าในวงเล็บปีกกาต้องไม่เกินขนาดของอาร์เรย์ เช่น

  ```
  int vector[3] = {1,2,3,4} ;   /* Invalid declaration */
  ```

  เป็นการประกาศอาร์เรย์ที่ผิดไวยากรณ์
- จำนวนค่าในวงเล็บปีกกาน้อยกว่าขนาดของอาร์เรย์ได้ ในกรณีนี้ ค่าของตัวประกอบที่เหลือจะถูกกำหนดให้เป็น 0 ยกตัวอย่างเช่น

  ```
  int vector[5] = {1,2,3} ;
  ```

  มีค่าเท่ากับ

  ```
  int vector[5] = {1,2,3,0,0} ;
  ```
- เมื่อมีการกำหนดค่าเริ่มต้นของอาร์เรย์ ไม่จำเป็นต้องระบุขนาดของอาร์เรย์ ในกรณีนี้ โปรแกรมคอมไพเลอร์จะกำหนดให้ขนาดของอาร์เรย์เท่ากับจำนวนค่าคงที่ในวงเล็บปีกกา ยกตัวอย่างเช่น

  ```
  int vector[] = {1,2,3} ;
  ```

  มีค่าเท่ากับ

  ```
  int vector[3] = {1,2,3} ;
  ```

4.2 สตริง (String)

สตริง (String) หมายถึงอาร์เรย์ของข้อมูลชนิดตัวอักษร ซึ่งใช้ในการบันทึกค่าของตัวแปรที่ประกอบด้วยตัวอักษรหลายตัว เช่น ชื่อคน ที่อยู่ ชื่อวิชา เป็นต้น โดยมีลักษณะพิเศษที่แตกต่างจากอาร์เรย์ทั่วไป คือ จะมีตัวอักษรพิเศษ NULL ('\0'; มีรหัสเท่ากับ 0) เป็นตัวกำหนดตำแหน่งท้ายสุดของสตริง วิธีการประกาศสตริงทำได้โดย

```
char string_name[dimension] ;
```

ยกตัวอย่าง เช่น

```
      char   name[10] ;
```

ตัวอย่างที่ 4.2 : การใช้สตริงพื้นฐาน

```
#include <stdio.h>
int main()
{
      char   name[5] ;
      name[0] = 'S' ;
      /* assign character 'S' to name[0] (firstตัวประกอบof name) */
      name[1] = 'a' ;
      name[2] = 'm' ;
      name[3] = '\0' ;  /* assign '\0' to indicate the end of string */
      printf("%s \n", name) ;          /* Display name */
      return 0;
}
```

ผลการรันโปรแกรม

```
Sam
```

ข้อแนะนำในการใช้สตริง

- คำสั่งกำหนดค่าในลักษณะ name = "Sam" จะไม่สามารถทำได้ เราต้องใช้ฟังก์ชัน strcpy ซึ่งอยู่ในไลบรารีมาตรฐานที่ได้รับการกำหนดรูปแบบภายในไฟล์เฮดเดอร์ชื่อ string.h ดังตัวอย่างต่อไปนี้

```
#include <stdio.h>
#include <string.h>    /* Include header file about string operation */
int main()
{
   char   name[5] ;
   strcpy(name, "Sam") ;/* Use strcpy to assign "Sam" to name */
   printf("%s \n", name) ;
   return 0;
}
```

ให้สังเกตว่า มีคำสั่งรวมไฟล์ string.h ในส่วนพรีโปรเซสเซอร์ อนึ่ง strcpy เป็นฟังก์ชันสำหรับใช้คัดลอกค่าของสตริงตัวที่ 2 ไปใส่ในสตริงตัวที่ 1

- ในการใช้สตริง ต้องกำหนดขนาดของสตริงที่จะมารับค่าให้เพียงพอกับความยาวของสตริงที่ต้องการบันทึก ตัวอย่างต่อไปนี้ เป็นตัวอย่างที่ไม่ถูกต้อง

```
char   data[5] ;            /* declare dimension 5 of string */
strcpy(data,"abcdefghijk");/* input string length is greater than 5 */
```

โดยทั่วไป ควรจะกำหนดขนาดให้มากกว่าที่จำเป็นเล็กน้อย เพื่อป้องกันข้อผิดพลาดที่อาจเกิดขึ้นได้

การกำหนดค่าเริ่มต้นของสตริง มีกฎเกณฑ์ดังนี้

- ใช้เครื่องหมายคำพูด Double quote (" ") ล้อมรอบตัวอักษรในสตริง ยกตัวอย่างเช่น

```
char name[10] = "thammasat";
```

- กรณีที่จองกำหนดขนาดของสตริงเป็น n จำนวนตัวอักษรระหว่าง " " ไม่ควรมากกว่า n-1 มิฉะนั้น จะมีเนื้อที่ไม่เพียงพอสำหรับเก็บค่าคงที่สตริงที่ต้องการ เช่น

```
char name[5] = "thammasat"; /* Insufficient string size */
```

จะเก็บได้เฉพาะ 5 ตัวอักษรแรก (กล่าวคือ thamm) เท่านั้น

- เมื่อมีการกำหนดค่าเริ่มต้นของสตริง ไม่จำเป็นต้องระบุขนาดของสตริง ในกรณีนี้ โปรแกรมคอมไพเลอร์จะกำหนดให้ขนาดของสตริงเท่ากับจำนวนตัวอักษรระหว่าง " " บวกด้วย 1 ยกตัวอย่างเช่น

```
char name[] = "thammasat";
```

มีค่าเท่ากับ

```
char name[10] = "thammasat";
```

โดยเหตุผลที่ต้องการหน่วยความจำเพิ่มอีก 1 ไบต์คือ ต้องมีเนื้อที่สำหรับเก็บตัวอักษร NULL ซึ่งเป็นตัวอักษรสำหรับแสดงจุดสิ้นสุดของสตริง

- สามารถใช้วิธีเดียวกับการกำหนดค่าเริ่มต้นของอาร์เรย์ 1 มิติได้โดยกำหนดค่าคงที่ให้เป็นชนิด char และเขียนทีละตัวคั่นด้วยเครื่องหมาย , ยกตัวอย่างเช่น

```
char     name[4] = {'S', 'a', 'm', '\0'} ;
```

มีค่าเท่ากับ

```
char     name[4] = "Sam";
```

หรือ

```
char     name[] = "Sam";
```

4.3 ฟังก์ชันสำหรับจัดการกับสตริง (String-Handling Functions)

เนื่องจากสตริงไม่ใช่ชนิดข้อมูลพื้นฐานในภาษา C ดังนั้น จึงไม่มีตัวดำเนินการสำหรับจัดการกับสตริงเหมือนอย่าง int, float หรือ double ผู้เขียนโปรแกรมที่ต้องการใช้สตริงจำเป็นที่จะต้องเขียนฟังก์ชันสำหรับจัดการกับสตริงหรือใช้ฟังก์ชันประเภทนี้จากไลบรารีมาตรฐานที่ภาษา C เตรียมไว้ให้

ในที่นี้ จะกล่าวถึงฟังก์ชันพื้นฐานที่ใช้สำหรับจัดการกับสตริงซึ่งถูกกำหนดรูปแบบฟังก์ชัน (Function Prototype) ไว้ในไฟล์เฮดเดอร์ชื่อ *string.h* ดังนั้น การที่จะใช้ฟังก์ชันที่จะกล่าวต่อไปในตอนนี้ จะต้องทำการรวมไฟล์นี้เข้าไปด้วย มิฉะนั้น โปรแกรมคอมไพเลอร์จะไม่รู้จักฟังก์ชันที่ใช้ และไม่สามารถทำการคอมไพล์ต่อไปได้

strcat, strncat

ฟังก์ชัน 2 อันนี้ เป็นชื่อย่อของ *string concatenation* ซึ่งหมายถึง การเชื่อมต่อสตริงการเชื่อม s_1 เข้ากับ s_2 หมายถึง การนำ s_2 มาต่อเข้ากับ s_1 ยกตัวอย่าง เช่น ถ้าเราทำการเชื่อม "Gone with " เข้ากับ "the wind" เราจะได้สตริง"Gone with the wind"

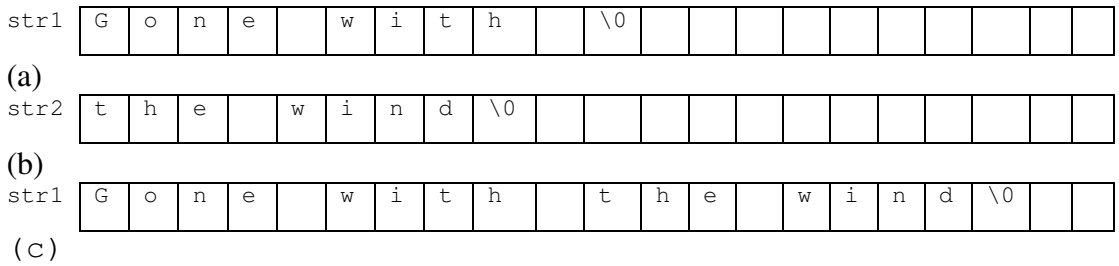
ฟังก์ชัน strcat จะรับอาร์กิวเมนต์ที่เป็นชื่อสตริง(Address ของสตริง) จำนวน 2 ค่าในลักษณะ

```
strcat(string1, string2) ;
```

และจะนำ *string2* ไปเชื่อมต่อท้ายหลัง *string1* อันมีผลทำให้ค่าของ *string1* เปลี่ยนไป แต่ทั้งนี้ จำนวน *dimension* ของ *string1* ต้องมีเพียงพอที่จะสามารถนำ *string2* ไปใส่ได้ ยกตัวอย่าง เช่น สมมุติว่า *str1* = "Gone with " และ *str2* = "the wind" หลังประโยคคำสั่ง

```
strcat(str1, str2) ;
```

แล้ว *str1* จะกลายเป็น "Gone with the wind" ดังแสดงในรูปที่ 4.2



<u>รูปที่ 4.2</u> การใช้ strcat (a) ค่าของ str1 ก่อนใช้ฟังก์ชัน strcat (b) ค่าของ str2 (c) ค่าของ str1 หลังใช้ฟังก์ชัน strcat

ฟังก์ชัน strncat จะคล้ายกับฟังก์ชัน strcat เพียงแต่ต้องการ Argument ตัวที่ 3 เอาไว้ใช้กำหนดจำนวนตัวอักษรที่จะเอามาเชื่อมต่อ โดยรูปแบบทั่วไป จะเป็นดังนี้

```
strncat(string1, string2, n) ;
```

ยกตัวอย่าง เช่น

```
strncat(str1, str2, 5) ;
```

จะหมายถึง ให้นำตัวอักษร 5 ตัวแรกจาก *str2* ไปเชื่อมต่อกับ *str1* สมมุติว่า *str1, str2* เป็นดังตัวอย่างข้างต้น หลังประโยคคำสั่งนี้ *str1* จะเปลี่ยนเป็น "Gone with the w"

strcmp, strncmp

ชื่อฟังก์ชัน strcmp, strncmp ย่อมาจากคำว่า *string comparison* ฟังก์ชันเหล่านี้ มีไว้สำหรับการเปรียบเทียบสตริง2 อัน ซึ่งผลการเปรียบเทียบที่เป็นไปได้ คือ

- String ทั้งสองเหมือนกัน ทั้งตัวอักษรในสตริงและความยาว
- ความยาวของสตริงทั้งสองไม่เท่ากัน แต่ตัวอักษรในสตริงที่สั้นกว่าเหมือนกับช่วงหน้าของสตริงที่ยาวกว่า เช่น Nation กับ National เป็นต้น
- ความยาวของสตริงทั้งสองอาจจะเท่ากันหรือต่างกัน แต่ตัวอักษรจะต่างกันในบางตำแหน่ง เช่น Computer กับ Computing เป็นต้น

ในกรณีสุดท้าย ฟังก์ชันจะตรวจสอบตำแหน่งซ้ายสุดของตัวอักษรที่แตกต่างกัน (ตำแหน่งแรกที่ตัวอักษรแตกต่างกัน) แล้วทำการเปรียบเทียบตัวอักษรที่แตกต่างกันนั้น จากรหัสของตัวอักษร ซึ่งโดยทั่วไป ลำดับของ a...z และ

A...Z จะไม่เปลี่ยนแปลง ไม่ว่าจะใช้รหัสแบบไหน ยกตัวอย่าง เช่น Computer กับ Computing จะแตกต่างกันที่ตัวที่ 7 และ e จะอยู่หน้า i ดังนั้น Computer จะอยู่หน้า Computing เป็นต้น วิธีการเรียงลำดับสตริงวิธีนี้ เราเรียกว่าลำดับตามพจนานุกรม (**Lexicographic Order**)

ฟังก์ชัน strcmp จะรับค่า Argument จำนวน 2 ค่า คือ ชื่อของสตริง 2 อัน (Address ของสตริง) แล้วทำการเปรียบเทียบสตริงทั้งสอง และคืนค่าต่อไปนี้

- 0 ถ้าสตริงทั้งสองเหมือนกัน
- จำนวนเต็มลบ ถ้าลำดับตามพจนานุกรมของ สตริงตัวแรกอยู่หน้าตัวที่สอง
- จำนวนเต็มบวก ถ้าลำดับตามพจนานุกรมของ สตริงตัวแรกอยู่หลังตัวที่สอง

ยกตัวอย่าง เช่น ถ้ามีสตริง 2 อันต่อไปนี้

```
char  str1[] = "Computer" ;
char  str2[] = "Computing" ;
int   strcmp_result ;
```

เมื่อเราทำประโยคคำสั่งต่อไปนี้

```
strcmp_result = strcmp(str1, str2) ;
if (strcmp_result < 0)
      printf(" %s < %s ", str1, str2) ;
else if (strcmp_result == 0)
      printf(" %s == %s ", str1, str2) ;
else
      printf(" %s > %s ", str1, str2) ;
```

ผลที่ได้จะเป็น

```
Computer < Computing
```

เนื่องจาก strcmp(str1, str2) คืนค่าจำนวนเต็มลบ ซึ่งแสดงว่า Computer อยู่หน้า Computing ในพจนานุกรม

strncmp คล้ายกับ strcmp เพียงแต่ต่างกันที่จะรับ Argument เพิ่ม 1 ตัวสำหรับใช้กำหนดจำนวนตัวอักษรที่จะทำการเปรียบเทียบ ถ้าตัวเลขตัวนี้เป็น 0 หรือจำนวนเต็มลบ ฟังก์ชันนี้จะคืนค่า 0 เสมอ เนื่องจากไม่มีการเปรียบเทียบ กรณีที่เป็นจำนวนเต็มบวก จะทำการคืนค่าในลักษณะเดียวกับ strcmp ยกตัวอย่าง เช่น ถ้าสตริงที่เปรียบเทียบเป็น str1, str2 ในตัวอย่างข้างต้น ประโยคคำสั่ง

```
strncmp(str1, str2, 5) ;
```

จะคืนค่า 0 เนื่องจากตัวอักษร 5 ตัวแรกของสตริงทั้งสองเป็นตัวอักษรเหมือนกัน

strcpy, strncpy

ดังที่กล่าวไว้ข้างต้นแล้วว่า เนื่องจากสตริงไม่ใช่ชนิดข้อมูลพื้นฐานในภาษา C ดังนั้น จึงไม่สามารถใช้กับตัวดำเนินการได้ ยกตัวอย่าง เช่น เราจะกำหนดค่าของสตริงโดยใช้ตัวดำเนินการ = ดังต่อไปนี้ไม่ได้

```
char  name[ 20 ] ;
...
name = "David" ;         /* Illegal Assignment Statement */
```

ในการกำหนดมาตรฐานของภาษา C จึงได้มีฟังก์ชันสำหรับใช้แทนตัวดำเนินการ = นี้ ให้ชื่อว่า strcpy และ strncpy ซึ่งย่อมาจาก *string copy*

strcpy จะรับ Argument ซึ่งเป็นชื่อสตริง(Address ของสตริง) โดยมีรูปแบบการใช้ทั่วไปดังนี้

```
strcpy(string1, string2) ;
```

โดยฟังก์ชันจะทำการ Copy ค่าของ string2 ไปเก็บไว้ที่ string1 อันมีผลเหมือนกับการกำหนดค่าของ string1 ให้เท่ากับค่าของ string2 และจะคืนค่า Address ของ string1 กลับมา ยกตัวอย่าง เช่น

```
char  str1[] = "Men in Black" ;
char  str2[] = "Face Off" ;
```

ดังแสดงในรูปที่ 4.3 หลังจากทำประโยคคำสั่ง

```
strcpy(str1, str2) ;
```

ค่าของ str1 จะเปลี่ยนเป็น str2 ดังแสดงในรูป 4.4

str1	'M'	'e'	'n'	' '	'i'	'n'	' '	'B'	'l'	'a'	'c'	'k'	'\0'

str2	'F'	'a'	'c'	'e'	' '	'O'	'f'	'f'	'\0'

รูปที่ 4.3 ค่าของ String ก่อนการใช้ฟังก์ชัน strcpy

str2	'F'	'a'	'c'	'e'	' '	'O'	'f'	'f'	'\0'	'a'	'c'	'k'	'\0'

รูปที่ 4.4 ค่าของ String หลังการใช้ฟังก์ชัน strcpy

strncpy จะคล้ายกับ strcpy เพียงแต่ต้องการ Argument เพิ่ม 1 ตัวสำหรับกำหนดจำนวนตัวอักษรที่จะทำการ Copy โดยมีรูปแบบการใช้ทั่วไป ดังนี้

```
strncpy(string1, string2, max_len) ;
```

จะหมายถึง การสั่งให้โปรแกรมทำการ Copy ตัวอักษรจำนวน *max_len* ตัวจาก *string2* ไปเก็บเป็นค่าของ *string1* ถ้า *max_len* มากกว่าจำนวนตัวอักษรที่ไม่ใช่ NUL ของ *string2* โปรแกรมจะทำการเติม NUL ต่อท้ายเข้าไปให้เต็มจำนวน Cell ของ *string1* ตรงกันข้าม ถ้าหากว่า *max_len* น้อยกว่าจำนวนตัวอักษรที่ไม่ใช่ NUL ของ *string2* ค่าของ *string1* จะเป็น String ที่ไม่มี NUL ปิดท้าย

ยกตัวอย่าง เช่น ถ้าสตริงเป็นตามตัวอย่างข้างต้น หลังจากทำประโยคคำสั่งต่อไปนี้

```
strncpy(str1, str2, 5) ;
```

ค่าของ *string1* จะเปลี่ยนเป็น "Face "

strlen

ชื่อฟังก์ชันนี้ ย่อมาจากคำว่า *string length* มีไว้สำหรับหาความยาวของสตริงหรือจำนวนตัวอักษรที่ไม่ใช่ NUL นับถึงอักษรตัวสุดท้ายก่อนอักษร NUL ตัวแรก ยกตัวอย่าง เช่น

```
char  str1[] = "The Rock" ;
char  str2[] = "Sphere" ;
char  null_string[] = "" ;
```

str1, str2 จะมีความยาวเท่ากับ 8 และ 6 ตามลำดับ ในขณะที่ null_string จะมีความยาวเป็น 0 เพราะไม่มีตัวอักษรที่ไม่ใช่ NUL อยู่หน้า NUL สมมุติว่า เราสั่งให้โปรแกรมทำประโยคคำสั่งต่อไปนี้

```
strcpy(str1, str2) ;
```

ความยาวของ str1 จะเปลี่ยนเป็น 6 ตาม str2 ถึงแม้ว่าจำนวนตัวอักษรที่ไม่ใช่ NULL จะมีมากกว่า 6 ก็ตาม

ตัวอย่าง 4.3 ตัวอย่างการใช้ฟังก์ชันจัดการกับสตริง

```
/* This program demonstrates how to use string handling functions */
#include <stdio.h>
#include <string.h>
/* string.h is necessary when using string function */
int main()
{
  char  first[20], last[20], full[50] ; /* Declare strings */
  int   length, result ;
  printf("Enter your first name :") ;
  scanf("%s",first); /* Read string from standard input */
  printf("Enter your last name :") ;
  scanf("%s",last);

  strcpy(full, first) ; /* Copy first name to full */
  strcat(full," ") ; /* Concatenate space onto full */
  strcat(full, last) ; /* Concatenate last name to full */
  /* Calculate total alphabets in full name */
  length = strlen(first)+strlen(last);

  printf("Your name is %s.\n", full) ;   /* Print your full name */
  /* Print total alphabets of your name */
```

```
  printf("Your name is %d long.\n", length) ;

  /* Comparison */
  result = strcmp(first, "Sakda") ;    /* Compare with "Sakda" */
  /* Print the result of comparison */
  printf("Your name is %s", result < 0 ? "BEFORE" : "AFTER") ;
  printf(" Sakda") ;
  result = strcmp(first, "Somsak") ;    /* Compare with "Somsak" */
  printf(" and %s", result < 0 ? "BEFORE" : "AFTER") ;
  printf(" Somsak\n") ;
  return 0;
}
```

ผลการรันโปรแกรม

```
Enter your first name :Somchai
Enter your last name :Thammasat
Your name is Somchai Thammasat.
Your name is 16 long.
Your name is AFTER Sakda and BEFORE Somsak
```

strstr, strchr, strrchr

ชื่อฟังก์ชัน strstr ย่อมาจากคำว่า *string in string* มีไว้สำหรับทำการค้นหา string อันหนึ่ง ใน string อีกอันหนึ่ง โดยมีรูปแบบการใช้ทั่วไป เป็นดังนี้

```
strstr(string1, string2) ;
```

หมายถึง เป็นการสั่งให้หา string2 ใน string1 โดยจะคืน Address ของตัวอักษรที่เหมือนกัน หรือ String ย่อย (Substring) โดยเริ่มจากตัวที่เหมือนกัน ยกตัวอย่าง เช่น

```
strstr("The Sound of Music", "Sound") ;
```

จะคืนค่า "Sound of Music" กลับมา

ชื่อฟังก์ชัน strchr, strrchr ย่อมาจากคำว่า *string has character* มีไว้สำหรับใช้ค้นหาตัวอักษรใน String โดยมีรูปแบบทั่วไป ดังนี้

```
strchr(string1, character) ;
strrchr(string1, character) ;
```

จะหมายถึง ให้ทำการหาตัวอักษร character โดย strchr และ strrchr จะต่างกันที่ strchr จะค้นหาตัวอักษร character ตัวแรกที่ปรากฏใน string1 (ค้นจากหน้าไปหลัง) ในขณะที่ strrchr จะค้นหาตัวสุดท้าย (ค้นจากหลังไปหน้า) ฟังก์ชันทั้งสองนี้จะคืนค่าในลักษณะเดียวกับ strstr กล่าวคือ Address ของตัวอักษรที่ค้นได้ ยกตัวอย่าง เช่น

```
printf("%s", strchr("The Sound of Music", 'o')) ;
```

จะแสดงข้อความ "ound of Music" ในขณะที่ประโยคคำสั่ง

```
printf("%s", strrchr("The Sound of Music", 'o')) ;
```

จะแสดงข้อความ "of Music"

ตัวอย่างที่ 4.4 ตัวอย่างการใช้ strstr, strchr และ strrchr

```
/* This program demonstrate how to use strstr, strchr & strrchr */
#include <stdio.h>
#include <string.h>  /* Notice that this file is necessary */
int main()
{
  char str[20] = "Gone with the wind" ;
  printf("strstr (word with) : %s\n", strstr(str, "with")) ;
  printf("strchr (character n) : %s\n", strchr(str, 'n')) ;
  printf("strrchr (character n) : %s\n", strrchr(str, 'n')) ;
  return 0;
}
```

ผลการรันโปรแกรม

```
strstr (word with) : with the wind
strchr (character n) : ne with the wind
strrchr (character n) : nd
```

4.4 อาร์เรย์หลายมิติ (Multiple Dimensional Array)

เราสามารถกำหนดให้อาร์เรย์เป็นอาร์เรย์ที่มีจำนวนมิติเกิน 1 ได้ การประกาศสำหรับอาร์เรย์2 มิติ จะเป็นในลักษณะดังนี้

```
data type   array_name[size1][size2] ;
```

ยกตัวอย่าง เช่น

```
int   matrix[2][4] ;
```

จะหมายถึง อาร์เรย์ชื่อ matrix ที่มีตัวประกอบเป็นชนิด int และขนาดของมิติที่ 1, 2 เท่ากับ 2,4 ตามลำดับ ดังนั้น จำนวนตัวประกอบในอาร์เรย์ matrix ทั้งหมดจะเท่ากับ 2×4=8 ตัว

ตารางที่ 4.1 แสดงตัวอย่างของอาร์เรย์หลายมิติ

ตารางที่ 4.1 ตัวอย่างการประกาศอาร์เรย์หลายมิติ

การประกาศอาร์เรย์	*จำนวนมิติ*	*จำนวนตัวประกอบ*
`int id[100]`	1	100
`int matrix[10][10]`	2	100
`char name[100][20]`	2	2,000
`float data[10][10][10]`	3	1,000
`int count[40][10][20]`	3	8,000

วิธีการอ้างอิงใช้ ก็จะเหมือนกับอาร์เรย์ 1 มิติทั่วไป เพียงแต่ต้องใส่จำนวนดัชนีตามจำนวนมิติ ยกตัวอย่างเช่น ถ้าต้องการอ้างอิงถึงค่าของตัวประกอบหนึ่งในอาร์เรย์ matrix ที่ทำการประกาศข้างต้น จะเป็นในลักษณะนี้

```
matrix[0][0] = 2 ;       matrix[0][1] = 3 ;
```

ข้อแนะนำในการใช้อาร์เรย์หลายมิติ อาร์เรย์หลายมิติจะใช้กันมากในกรณีต้องการเก็บข้อมูลที่อยู่ในรูป*อาร์เรย์ของสตริง* กล่าวคือ ชุดข้อมูลที่มีชนิดข้อมูลเป็นสตริง เช่น เวลาต้องการเก็บข้อมูลรายชื่อวิชาที่ลงทะเบียน ข้อมูลแต่ละตัวเป็นชื่อวิชา ซึ่งต้องในรูปของสตริง เป็นต้น กรณีที่เป็นอาร์เรย์หลายมิติของข้อมูลชนิดอื่น ๆ เช่น int float เราสามารถแปลงอาร์เรย์หลายมิติเป็นอาร์เรย์ 1 มิติได้ เช่น ถ้าต้องการเก็บข้อมูลของเมตริกซ์ 3×3 แทนที่จะใช้ matrix[3][3] เราสามารถใช้ matrix[9] แทนได้ เพราะจำนวนตัวประกอบของอาร์เรย์ 2 อันนี้จะเท่ากัน (กล่าวคือ ใช้เนื้อที่ในหน่วยความจำเท่ากัน) การใช้อาร์เรย์ 1 มิติมีข้อดีคือ จะมีความเร็วในการประมวลผลสูงกว่า เพราะการอ้างอิงถึงตัวประกอบในอาร์เรย์ใช้ดัชนีเพียงค่าเดียว ส่งผลให้เข้าถึงหน่วยความจำได้เร็วกว่ากรณีที่ต้องใช้ดัชนีมากกว่า 1 ค่า แต่อาจจะเกิดความยุ่งยากในการกำหนดดัชนีของตัวประกอบ จึงเป็นปัญหาความสมดุลย์ระหว่างประสิทธิภาพกับความสะดวก

การกำหนดค่าเริ่มต้นของอาร์เรย์หลายมิติ สามารถทำได้เหมือนกับกรณีกำหนดค่าเริ่มต้นของอาร์เรย์ 1 มิติ เพียงแต่เพิ่มจำนวนค่าคงที่ และจำนวนวงเล็บปีกกาตามจำนวนตัวประกอบและจำนวนมิติ

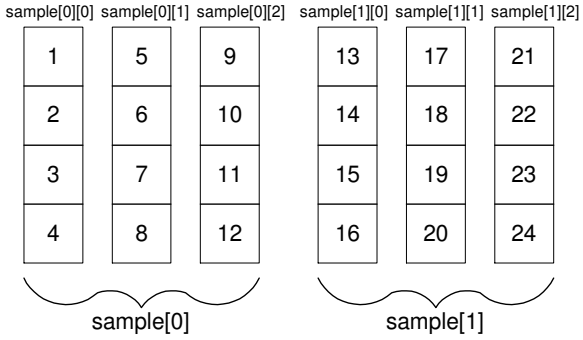
- ใช้จำนวนวงเล็บปีกกาเท่ากับจำนวนมิติ และจำนวนค่าในวงเล็บปีกกาแต่ละคู่ให้เท่ากับขนาดของมิตินั้นๆ ยกตัวอย่างเช่น ถ้าทำการประกาศอาร์เรย์ 3 มิติดังต่อไปนี้

```
int sample[2][3][4] = { {1,2,3,4},{5,6,7,8},{9,10,11,12},
                        {13,14,15,16},{17,18,19,20},{21,22,23,24} };
```

กรณีนี้ ค่าของตัวประกอบต่าง ๆ ในอาร์เรย์ *sample* จะเป็นดังนี้

```
sample[0][0][0] = 1          sample[0][0][1] = 2
sample[0][1][0] = 5          sample[0][2][0] = 9
sample[1][0][0] = 13         sample[1][1][0] = 17
```

อาร์เรย์ sample จะเป็นดังแสดงในรูปที่ 4.5 ทั้งนี้ จำนวนค่าในวงเล็บปีกกาแต่ละคู่ต้องไม่เกินขนาดของมิตินั้นๆ



รูปที่ 4.5 ตัวอย่างอาร์เรย์หลายมิติ

- ใช้จำนวนวงเล็บปีกกาเพียงคู่เดียว แต่ใส่จำนวนค่าทั้งหมดในวงเล็บปีกกาให้เท่ากับจำนวนตัวประกอบทั้งหมดของอาร์เรย์นั้น ยกตัวอย่างเช่น ถ้าทำการประกาศอาร์เรย์ 3 มิติดังต่อไปนี้

```
int sample[2][3][4] = { 1,2,3,4,5,6,7,8,9,10,11,12,
                        13,14,15,16,17,18,19,20,21,22,23,24};
```

อาร์เรย์ที่ได้จะเหมือนกับตัวอย่างข้างต้นที่ดังแสดงในรูปที่ 4.2 ทั้งนี้ จำนวนค่าทั้งหมดในวงเล็บปีกกาต้องไม่เกินจำนวนตัวประกอบทั้งหมดของอาร์เรย์นั้น

- หากจำนวนค่าที่ใช้กำหนดค่าเริ่มต้นน้อยกว่าขนาดของแต่ละมิติ ค่าของตัวประกอบที่เหลือจะถูกกำหนดให้เป็น 0 ยกตัวอย่างเช่น

```
int mat[2][3] = {{1,2},{3,4}};
```

มีค่าเท่ากับ

```
int mat[2][3] = {{1,2,0},{3,4,0}};
```

- ในทำนองเดียวกัน หากจำนวนค่าที่ใช้กำหนดค่าเริ่มต้นน้อยกว่าจำนวนตัวประกอบทั้งหมดของอาร์เรย์ค่าของตัวประกอบที่เหลือจะถูกกำหนดให้เป็น 0 ยกตัวอย่างเช่น

```
int mat[2][3] = {1,2,3,4};
```

มีค่าเท่ากับ

```
int mat[2][3] = {1,2,3,4,0,0};
```

การกำหนดค่าเริ่มต้นของอาร์เรย์ของสตริง สามารถทำได้โดยใช้วงเล็บปีกกา แล้วใส่สตริงตามจำนวนขนาดของอาร์เรย์ของสตริง (จำนวนของสตริงในอาร์เรย์) คั่นด้วยเครื่องหมาย , ยกตัวอย่างเช่น

```
char  weekday[7][10] = { "Monday", "Tuesday", "Wednesday",
                         "Thursday, "Friday", "Saturday", "Sunday" } ;
```

หรืออาจจะไม่กำหนดมิติ (จำนวนสตริงในอาร์เรย์)ในลักษณะ

```
char  weekday[ ][10] = { "Monday", "Tuesday", "Wednesday",
                         "Thursday, "Friday", "Saturday", "Sunday" } ;
```

ซึ่งโปรแกรมจำทำการจองเนื้อที่สำหรับสตริงจำนวน 7 อัน โดยสตริงแต่ละตัวมีตัวประกอบ 10 ตัว

สิ่งที่พึงระวังคือ จะต้องใส่จำนวนมิติตัวหลังเข้าไปด้วย เพราะมิฉะนั้น โปรแกรมจะไม่สามารถรู้ได้ว่า เป็นอาร์เรย์ของสตริงที่มีความยาวขนาดเท่าไร และความยาวของสตริงที่เป็นตัวประกอบของอาร์เรย์ของสตริงจะต้องเท่ากัน เพื่อที่จะได้สามารถอ้างอิงได้ ยกตัวอย่าง weekday ข้างต้น ถ้าเราอ้างอิงถึง weekday[1] โปรแกรมก็จะนับจาก weekday[0] ไป 10 ช่อง ซึ่งก็คือ "Tuesday"

แบบฝึกหัด

1. สมมุติว่า เราทำการประกาศอาร์เรย์พร้อมกำหนดค่าเริ่มต้นดังต่อไปนี้

```
int   a[10] = { 1,3,5,7,9,8,6,4,2,0 } ;
```

ค่าต่อไปนี้ มีค่าเท่ากับเท่าไร

a) a[0] b) a[5] c) a[9] d) a[10]

2. จงบอกข้อผิดพลาดของประโยคคำสั่งต่อไปนี้

```
char  c = "A" ;
```

3. จงบอกผลที่ได้จากประโยคคำสั่งต่อไปนี้

```
char  c[5] = "A" ;
printf("%c\n", c[0]) ;
printf("%s\n", c) ;
```

สำหรับข้อ 4 และ 5 สมมุติว่าทำการประกาศดังต่อไปนี้

```
char s1[] = " Gone with the wind " ;
char s2[50] = " The sound of music " ;
char s3[100] = " Casablanca ", s4[100] = "", s5[100] ;
```

4. จงบอกจำนวนหน่วยความจำที่ใช้ในการเก็บค่าของ String s1

5. จงบอกผลที่ได้จากประโยคคำสั่งต่อไปนี้ โดยสมมุติให้เป็นการรันที่เรียงตามลำดับ a) – j) โดยต่อเนื่องกัน

```
a) printf("%c\n", s1[10]) ;
b) printf("%s\n", s2) ;
c) strcpy(s4, s1) ; printf("%s\n", s4) ;
d) strcat(s4, s2) ; printf("%s\n", s4) ;
e) printf("%d\n", strlen(s4)) ;
f) strncpy(s5, s4, 12) ; s5[12] = '\0' ; printf("%s\n", s5) ;
g) printf("%s\n", strstr(s1, "he")) ;
h) printf("%s\n", strchr(s2, 'o')) ;
i) printf("%s\n", strrchr(s4, 'e')) ;
j) printf("%s %s %s %s\n", s1, s2, s3, s4) ;
```

พร้อมอธิบายการทำงานของฟังก์ชัน strstr, strchr, และ strrchr ที่ใช้ในโปรแกรม

6. สมมุติว่า ทำการประกาศอาร์เรย์2 มิติ ดังต่อไปนี้

```
int   mat[10][10], i, j ;
```

และทำการรันประโยคคำสั่งต่อไปนี้

```
for (i = 0 ; i < 10 ; i++)
      for (j = 0 ; j < 10 ; j++)
            mat[i][j] = (9 – i) + 10*(9 – j) ;
```

จงบอกค่าของ Expression ต่อไปนี้

```
a) mat[0][0]  b) mat[5][0]  c) mat[9][5]  d) mat[9][9]
```

7. เขียนโปรแกรมสำหรับรับค่า ชื่อ วัน-เดือน-ปีเกิด เลขทะเบียน วิชาที่ชอบ และงานอดิเรก แล้วนำมาแสดงผลออกหน้าจอในรูปแบบดังนี้

   Name : XXX YYYYY (XXX แทนชื่อ, YYYYY แทนนามสกุล)

   Birthday : DD MM YY (DD, MM, YY แทนวัน เดือน ปีเกิดตามลำดับ)

   ID : III (III แทนเลขทะเบียน)

   Favorite Subject : SSS (SSS แทนชื่อวิชา)

   Hobby : HHH (HHH แทนงานอดิเรก)

8. เขียนโปรแกรมสำหรับรับคำศัพท์ภาษาอังกฤษหลาย ๆ คำ แล้วนำมาเชื่อมต่อกันเป็น 1 ประโยค

9. จงเขียนโปรแกรมสำหรับเรียงลำดับข้อมูลด้วยวิธี Insertion Sort

10. จงเขียนโปรแกรมสำหรับหาผลบวก ลบ คูณ ของ Matrix 2 อัน โดยให้สามารถกำหนดจำนวนแถวและหลักของเมตริกซ์ได้

11. เขียนโปรแกรมสำหรับรับคำศัพท์ภาษาอังกฤษ แล้วทำการแปลงอักษรพิมพ์เล็กให้เป็นตัวพิมพ์ใหญ่ (จำลองการทำงานของ Function *toupper*)

12. เขียนโปรแกรมสำหรับแปลงเลขฐานสิบหกให้เป็นเลขฐานสิบ โดยรับข้อมูลเข้ามาในรูปของสตริง ยกตัวอย่าง เช่น A12 เท่ากับ 2578 ในเลขฐานสิบ เป็นต้น