Chapter 7 Array

- ในบทที่ผ่านมาเป็นการใช้งานตัวแปรซึ่งเก็บค่าคงที่ได้เพียงหนึ่งตัว ในภาษาซีอนุญาตให้มีการทำงานกับตัวแปรที่สามารถเก็บค่าคงที่ได้หลายค่า เราเรียกตัวแปรนั้นว่าอาเรย์ (array)
- สมมติว่ามีคะแนนของนักศึกษาชุดหนึ่งที่ต้องการประมวลผล ในบทที่ผ่านมาเราสามารถหาค่าเฉลี่ยของชุดคะแนนได้โดยการบวกคะแนนเข้าไปในตัวแปรที่เก็บคะแนนรวมทุกๆครั้งที่มีการกรอกคะแนนเข้ามา แต่หากต้องการเรียงคะแนนจากน้อยไปมากเราสามารถทำได้ก็ต่อเมื่อมีคะแนนชุดนั้นทั้งหมด โดยแต่ละคะแนนอาจจะเก็บในตัวแปรแยกจากกัน เช่น grade1, grade2, …แล้วใช้ if ช่วยเปรียบเทียบค่า แต่ถ้าหากทำเช่นนี้โปรแกรมจะมีขนาดใหญ่และซับซ้อนมาก ซึ่งในกรณีนี้ array สามารถช่วยได้

7.1 การประกาศตัวแปร array

- ตัวแปร grades หากประกาศเป็นแบบ array แล้วจะเข้าถึงสมาชิกแต่ละตัวได้โดยใช้ index หรือตัวเลขตามหลัง เช่น grades[0], grade[5]
- สมาชิกแต่ละตัวในตัวแปรสามารถใช้งานได้เช่นเดียวกับตัวแปรปกติ ตัวอย่างเช่น

```
g = grades[50];
g = grades[i];
```

เมื่อ i เป็นจำนวนเต็มบวก

- หากต้องการใส่ค่าให้สมาชิกแต่ละตัวใน array

```
grades[100] = 95;
grades[i] = g;
```

- ซึ่งความสามารถเก็บกลุ่มของข้อมูลชุดเดียวกันไว้ในตัวแปร array ทำให้โปรแกรมมีประสิทธิภาพและไม่ซับซ้อน ตัวอย่างเช่นเราสามารถเข้าถึงสมาชิกแต่ละตัวโดยใช้ลูป for

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

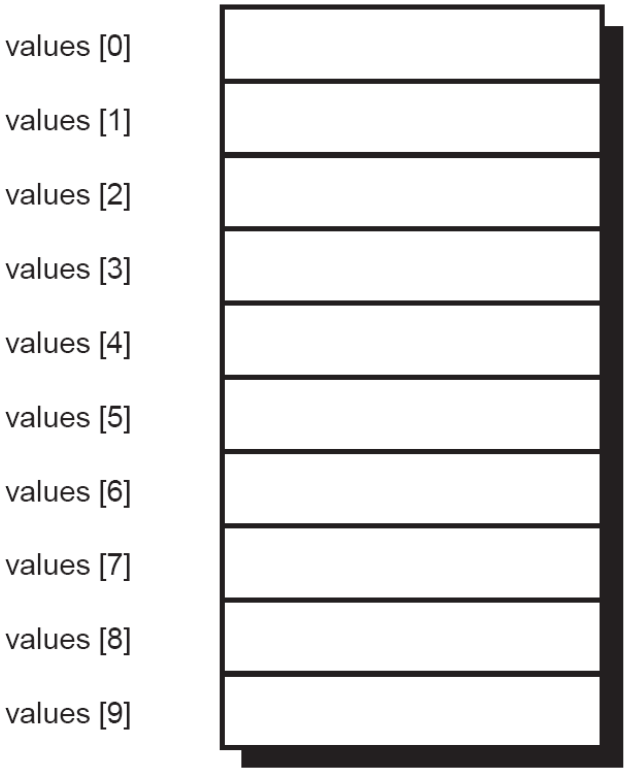
- สมาชิกตัวแรกใน array เริ่มที่ 0 เสมอ และตัวสุดท้ายจบที่จำนวนสมาชิกใน array ลบหนึ่ง
- หากมีการอ้างถึงสมาชิกในตัวแปรเกินกว่าที่กำหนดไว้ อาจจะเกิดข้อผิดพลาดขึ้น หรือค่าที่ได้อาจไม่มีความหมาย
- นอกจากจะใช้จำนวนเต็มเพื่อเข้าถึงสมาชิกใน array แล้วเรายังสามารถเข้าถึงได้โดยใช้นิพจน์ทางคณิตศาสตร์ที่ให้ผลลัพธ์เป็นจำนวนเต็มได้เช่นเดียวกัน สมมติให้ low และ high เป็นตัวแปรแบบ int

```
next_value = sorted_data[(low+high)/2];
```

- เช่นเดียวกับตัวแปรทั่วๆ ไป array จะต้องถูกประกาศก่อนการใช้งานเสมอ ซึ่งการประกาศจะต้องมีชนิดของตัวแปร (int, float, char, ...) และจำนวนสมาชิกใน array เช่น

```
int grades[100];
float averages[200];
char varchar[15];
unsigned int temp[28];
int values[10];
```

- การประกาศตัวแปรมีผลให้พื้นที่ในหน่วยความจำของคอมพิวเตอร์ที่มีขนาดเพียงพอกับจำนวนสมาชิกถูกจอง

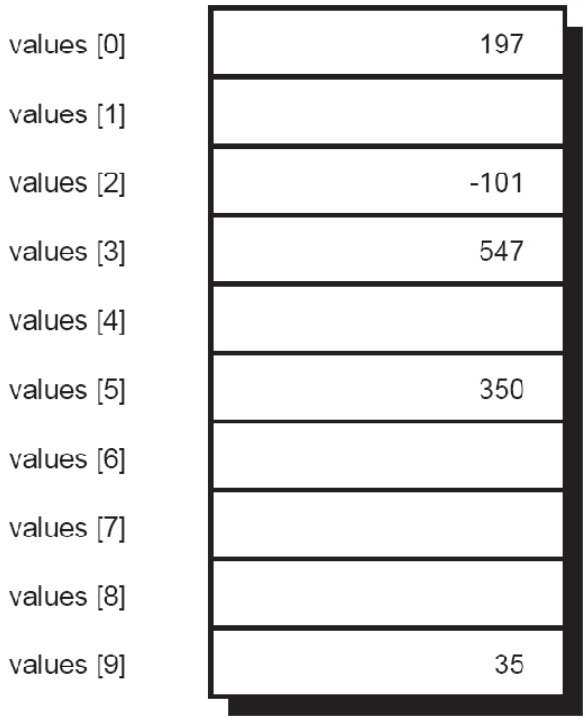


- หากมีการทำงานดังคำสั่งด้านล่างนี้ สมาชิกใน array แต่ละตัวจะเก็บค่าดังรูป

Program 7.1 Working with an Array

```
#include <stdio.h>
int main (void)
{
    int values[10];
    int index;
    values[0] = 197;
    values[2] = -100;
    values[5] = 350;
    values[3] = values[0] + values[5];
    values[9] = values[5] / 10;
    --values[2];

    for( index = 0; index < 10; ++index )
        printf("values[%i] = %i\n", index, values[index]);
    return 0;
}
```



<u>ตัวอย่าง การใช้งานสมาชิกของ array เป็นตัวนับ</u>

- ในหัวข้อนี้เราจะทดลองใช้ array เพื่อเป็นตัวนับคะแนนของการโหวตจากหนึ่งถึงสิบ ของคนยี่สิบคน
- การใช้ array เป็นตัวนับแทนตัวแปรธรรมดาในกรณีนี้ทำให้โปรแกรมกระชับขึ้น

Program 7.2 Demonstrating an Array of Counters

```
#include <stdio.h>
int main (void)
{
    int ratingCounters[11], i, response;

    for ( i = 1; i <= 10; ++i )
        ratingCounters[i] = 0;

    printf ("Enter your responses\n");

    for( i = 1; i <= 20; ++i )
    {
        scanf ("%i", &response);
        if ( response < 1 || response > 10 )
            printf ("Bad response: %i\n", response);
        else
            ++ratingCounters[response];
    }

    printf ("\n\nRating Number of Responses\n");
    printf ("------ -------------------\n");

    for ( i = 1; i <= 10; ++i )
        printf ("%4i%14i\n", i, ratingCounters[i]);
    return 0;
}
```

```
Enter your responses
6
5
8
3
9
6
5
7
15
Bad response: 15
5
5
1
7
4
10
5
5
6
8
9

Rating Number of Responses
------ -------------------
1         1
2         0
3         1
4         1
5         6
6         3
7         2
8         2
9         2
10        1
```

- สังเกตว่าเราใช้ array ที่มีสมาชิก 11 ตัว เพื่อนับจำนวนคะแนนโหวตจาก 1-10 เนื่องจากการอ้างอิงสมาชิกใน array จะเริ่มจากศูนย์ไปจนถึงสิบ และในกรณีนี้ สมาชิก array ตัวแรกหรือ ratingCounter[0] จะไม่ถูกใช้งาน

<u>ตัวอย่าง การหาอนุกรม Fibonacci โดยใช้ array</u>

$$F_n = F_{n-1} + F_{n-2}; \quad \text{เมื่อ } F_0 = 0, F_1 = 1$$

Program 7.3 Generating Fibonacci Numbers

```
// Program to generate the first 15 Fibonacci numbers
#include <stdio.h>
int main (void)
{
    int Fibonacci[15], i;

    Fibonacci[0] = 0; // by definition
    Fibonacci[1] = 1; // ditto

    for ( i = 2; i < 15; ++i )
        Fibonacci[i] = Fibonacci[i-2] + Fibonacci[i-1];

    for ( i = 0; i < 15; ++i )
        printf ("%i ", Fibonacci[i]);

    return 0;
}
```

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

<u>ตัวอย่างการใช้ array เพื่อหาจำนวนเฉพาะ</u>

- ในบทที่แล้วเราหาจำนวนเฉพาะของ x โดยใช้การหารด้วยตัวหาร 2 ถึง x-1 หากหารไม่ลงตัวจะถือว่า x เป็นจำนวนเฉพาะ ซึ่งวิธีดังกล่าวนี้ไม่มีประสิทธิภาพ เนื่องจากจะต้องตรวจสอบตัวหารทุกตัว
- การตรวจสอบว่าจำนวนเป็นจำนวนเฉพาะหรือไม่อาจจะใช้วิธีหารด้วยจำนวนเฉพาะตัวที่ต่ำกว่า เนื่องจากเลขใดๆที่ไม่ใช่จำนวนเฉพาะจะสามารถกระจายออกได้เป็นผลคูณของจำนวนเฉพาะ (อย่างเช่น 20 สามารถกระจายเป็นผลคูณของ 2, 2, 5) ดังนั้นหากหารด้วยจำนวนเฉพาะแล้วไม่ลงตัวก็จะถือว่าเลขนั้นเป็นจำนวนเฉพาะด้วย
- ดังนั้นเราจะใช้ array เพื่อเก็บค่าจำนวนเฉพาะที่หาได้เอาไว้ เพื่อใช้ตรวจสอบหาจำนวนเฉพาะตัวถัดไป
- นอกจากนี้จำนวนใดๆที่ไม่ใช่จำนวนเฉพาะ สามารถหารลงตัวด้วยจำนวนเฉพาะที่มีค่าต่ำกว่าหรือเท่ากับรากที่สองของตัวมันเอง

Program 7.4 Revising the Program to Generate Prime Numbers

```
#include <stdio.h>
#include <stdbool.h>
// Modified program to generate prime numbers
int main (void)
{
    int p, i, primes[50], primeIndex = 2;
    bool isPrime;

    primes[0] = 2;
    primes[1] = 3;

    for( p = 5; p <= 50; p = p + 2 )
    {
        isPrime = true;

        for( i = 1; isPrime && p >= primes[i]*primes[i]; ++i )
            if ( p % primes[i] == 0 )
                isPrime = false;

        if ( isPrime == true )
        {
            primes[primeIndex] = p;
            ++primeIndex;
        }
    }

    for ( i = 0; i < primeIndex; ++i )
        printf ("%i ", primes[i]);

    printf ("\n");
    return 0;
}
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

7.2 การใส่ค่าเริ่มต้นให้กับ array

- เราสามารถใส่ค่าเริ่มต้นให้สมาชิกในตัวแปร array ได้เช่นเดียวกับตัวแปรทั่วไป ตัวอย่างเช่น

```
int counters[5] = { 0, 0, 0, 0, 0 };
int integers[5] = { 0, 1, 2, 3, 4 };
char letters[5] = { 'a', 'b', 'c', 'd', 'e' };
float sample_data[500] = { 100.0, 300.0, 500.5 };
```

- ในกรณีที่กำหนดค่าเริ่มต้นให้สมาชิกเพียงบางตัว สมาชิกที่เหลือจะถูกกำหนดเป็น 0 ให้โดยอัตโนมัติ
- ถ้าใส่ค่าเริ่มต้นตอนประกาศตัวแปรแล้ว ไม่จำเป็นจะต้องกำหนดจำนวนสมาชิกใน array ก็ได้

```
char word[] = { 'H', 'e', 'l', 'l', 'o', '!' };
```

- เราไม่จำเป็นจะต้องกำหนดค่าให้สมาชิกแบบเรียงตัว

```
float sample_data[500] ={[2] = 500.5, [1] = 300.0, [0] = 100.0};

int x = 1233;
int a[10] = { [9] = x + 1, [2] = 3, [1] = 2, [0] = 1 };

float sample_data[] = { [0] = 1.0, [49] = 100.0, [99] = 200.0 };
```

- หากต้องการกำหนดค่าเริ่มต้นให้สมาชิกจำนวนมาก อาจจะต้องใช้ลูปเข้าช่วย

Program 7.5 Initializing Arrays

```
#include <stdio.h>
int main (void)
{
    int array_values[10] = { 0, 1, 4, 9, 16 };
    int i;

    for ( i = 5; i < 10; ++i )
        array_values[i] = i * i;

    for ( i = 0; i < 10; ++i )
        printf ("array_values[%i] = %i\n", i, array_values[i]);

    return 0;
}
```

```
array_values[0] = 0
array_values[1] = 1
array_values[2] = 4
array_values[3] = 9
array_values[4] = 16
array_values[5] = 25
array_values[6] = 36
array_values[7] = 49
array_values[8] = 64
array_values[9] = 81
```

7.3 array สำหรับตัวอักษร

Program 7.6 Introducing Character Arrays

```
#include <stdio.h>
int main (void)
{
    char word[] = { 'H', 'e', 'l', 'l', 'o', '!' };
    int i;
    for ( i = 0; i < 6; ++i )
        printf ("%c", word[i]);
    printf ("\n");
    return 0;
}
```

```
Hello!
```

ตัวอย่าง การใช้งาน array ชนิดเลขจำนวนเต็มและตัวอักษรเพื่อช่วยแปลงเลขฐาน

- โปรแกรมจะแปลงตัวเลขจากฐานสิบไปเป็นฐานอื่นๆ รวมถึงฐานสิบหก โดยผู้ใช้กำหนดค่าเลขฐานสิบและฐานที่ต้องการจะแปลง
- ในขั้นแรกเราจะต้องหา algorithm ในการแปลงเลขฐาน ซึ่งทำได้โดยหารเอาเศษตัวเลขฐานสิบที่จะแปลง ด้วยค่าฐาน จากนั้นหารค่าตัวเลขที่จะแปลงโดยค่าฐานโดยทิ้งเศษไป และทำซ้ำการหารเอาเศษและหารไปเรื่อยๆ จนเหลือศูนย์
- สมมติว่าต้องการแปลงเลขสิบในฐานสิบไปเป็นเลขฐานสอง

10%2 = <u>0</u>, 10/2 = 5, 5%2 = <u>1</u>, 5/2 = 2, 2%2 = <u>0</u>, 2/2 = 1, 1%2 = <u>1</u>, 1/2 = 0

- จะได้ว่า $10_{10} = 1010_2$

Program 7.7 Converting a Positive Integer to Another Base

```
// Program to convert a positive integer to another base
#include <stdio.h>
int main (void)
{
    const char baseDigits[16] = {
    '0', '1', '2', '3', '4', '5', '6', '7',
    '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };

    int convertedNumber[64];
    long int numberToConvert;
    int nextDigit, base, index = 0;

    // get the number and the base
    printf ("Number to be converted? ");
    scanf ("%ld", &numberToConvert);
    printf ("Base? ");
    scanf ("%i", &base);

    // convert to the indicated base
    do {
        convertedNumber[index] = numberToConvert % base;
        ++index;
        numberToConvert = numberToConvert / base;
    }
    while ( numberToConvert != 0 );

    // display the results in reverse order
    printf ("Converted number = ");
    for (--index; index >= 0; --index ) {
        nextDigit = convertedNumber[index];
        printf ("%c", baseDigits[nextDigit]);
    }

    printf ("\n");
    return 0;
}
```

```
Number to be converted? 10
Base? 2
Converted number = 1010

Number to be converted? 128362
Base? 16
Converted number = 1F56A
```

- สังเกตการใช้ const เพื่อประกาศตัวแปรที่ค่าจะไม่เปลี่ยนแปลงตลอดการใช้งานโปรแกรม ซึ่งค่าที่กำหนดให้จะถูกเก็บอยู่ในส่วนของ Read-only memory หากมีความพยายามที่จะเปลี่ยนค่าของตัวแปรที่ถูกประกาศเป็น const คอมไพล์เลอร์อาจจะแจ้งความผิดพลาด
- ในโปรแกรมนี้หากผู้ใช้ใส่เลขฐานเป็น 0 หรือ 1 จะมีปัญหาคือ_____

7.4 array หลายมิติ

- array ที่กล่าวถึงตั้งแต่เริ่มเป็นแบบหนึ่งมิติ คือเก็บค่าคงที่ในมิติเดียว
- ในภาษาซีเราสามารถกำหนดให้ array มีมากกว่าหนึ่งมิติได้
- การเอาไปใช้งานอย่างเช่น matrix $M_{i,j}$

	j=0	1	2	3	4
i=0	10	5	-3	17	82
1	9	0	0	8	-7
2	32	20	1	0	14
3	0	0	8	7	6

- เราสามารถสร้าง array เพื่อเก็บ matrix นี้ได้ แต่ทั้งแถวและหลักต้องเริ่มจาก 0
- โดยสามารถอ้างถึงสมาชิกแต่ละตัวในรูปแบบ M[i][j]
- ดังนั้น sum = M[0][2] + M[2][4] จะเท่ากับ_____
- การประกาศตัวแปรสองมิติไม่แตกต่างจากการประกาศตัวแปรหนึ่งมิติ

```
int M[4][5];

int M[4][5] = {
                { 10, 5, -3, 17, 82 },
                { 9, 0, 0, 8, -7 },
                { 32, 20, 1, 0, 14 },
                { 0, 0, 8, 7, 6 }
              };

int M[4][5] = { 10, 5, -3, 17, 82, 9, 0, 0, 8, -7, 32, 20, 1, 0,
14, 0, 0, 8, 7, 6 };

int matrix[4][5] = { [0][0] = 1, [1][1] = 5, [2][2] = 9 };
```

7.5 array ที่ขนาดสมาชิกปรับเปลี่ยนได้

- จากโปรแกรม 7.3 ที่ใช้หาอนุกรม Fibonacci เราจำกัดขนาดของ array ไว้ที่ 15 ทำอย่างไรถ้าอยากให้ผู้ใช้เป็นคนกำหนดขนาดของ array?

Program 7.8 Generating Fibonacci Numbers Using Variable-Length Arrays

```
// Generate Fibonacci numbers using variable length arrays
#include <stdio.h>
int main (void)
{
    int i, numFibs;
    printf ("How many Fibonacci numbers do you want? ");
    scanf ("%i", &numFibs);
    if (numFibs < 1 || numFibs > 75) {
        printf ("Bad number, sorry!\n");
        return 1;
    }

    unsigned long long int Fibonacci[numFibs];
    Fibonacci[0] = 0; // by definition
    Fibonacci[1] = 1; // ditto

    for ( i = 2; i < numFibs; ++i )
        Fibonacci[i] = Fibonacci[i-2] + Fibonacci[i-1];

    for ( i = 0; i < numFibs; ++i )
        printf ("%llu ", Fibonacci[i]);

    printf ("\n");
    return 0;
}
```

```
How many Fibonacci numbers do you want (between 1 and 75)? 50
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584
4181 6765 10946 17711 28657 46368 75025 121393 196418 317811
514229 832040 1346269 2178309 3524578 5702887 9227465 14930352
24157817 39088169 63245986 102334155 165580141 267914296
433494437 701408733 1134903170 1836311903 2971215073 4807526976
7778742049
```