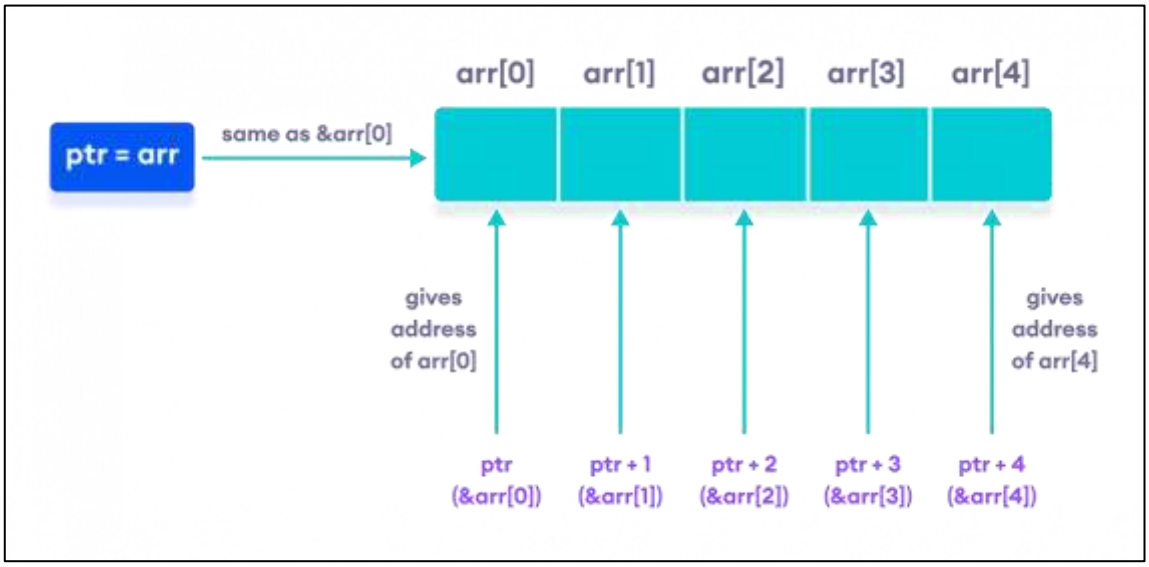
หน่วยที่ 3

อาร์เรย์และพอยน์เตอร์ (Array and Pointer)



หัวข้อเรื่อง

3.1 ตัวแปรอาร์เรย์

3.2 การประกาศตัวแปรอาร์เรย์

3.3 การกำหนดค่าตัวแปรอาร์เรย์

3.4 ตัวแปรพอยน์เตอร์

ในการเขียนโปรแกรมคอมพิวเตอร์นั้น นักพัฒนาโปรแกรมสามารถเลือกใช้งานตัวแปรได้หลากหลายชนิด ไม่ว่าจะเป็นตัวแปรแบบ Integer , Character , Floating และอื่น ๆ ซึ่งจะเห็นได้ว่าตัวแปรที่ประกาศขึ้นมาจะมีข้อจำกัด คือ สามารถเก็บค่าได้เพียง 1 ค่าเท่านั้น ถ้าหากต้องการจัดเก็บหลายค่าก็จะต้องสร้างตัวแปรที่มีชื่อคล้าย ๆ กันขึ้นมาอีกหลายชุด ซึ่งอาจจะทำให้ไม่ได้รับความสะดวก และทำให้ต้องเขียนโปรแกรมเพิ่มเติมอีกมาก ดังนั้นตัวแปรอาร์เรย์จึงเป็นตัวแปรที่สามารถเข้ามาแก้ไขปัญหาดังกล่าวได้เป็นอย่างดี ซึ่งทำให้นักพัฒนาโปรแกรมสามารถสร้างตัวแปรที่มีชนิดเดียวกันได้หลายชุด ซึ่งนับเป็นข้อดีอย่างหนึ่งที่ทำให้ตัวแปรอาร์เรย์ถูกนามาช่วยในการเขียนโปรแกรมเพื่อเก็บค่าจากตัวแปรที่มีจำนวนมาก

3.1 ตัวแปรอาร์เรย์ (Array)

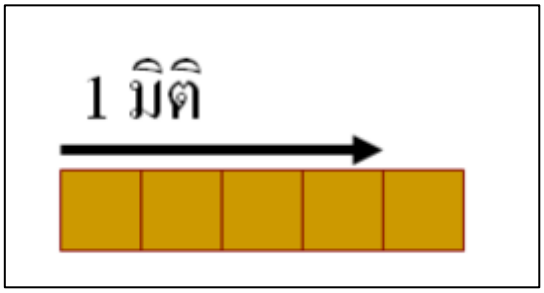
อาร์เรย์ คือ กลุ่มของตัวแปรชนิดเดียวกันมากกว่า 1 ตัวที่ใช้ชื่อตัวแปรเดียวกัน ซึ่งการเข้าถึงสมาชิก (element) แต่ละตัวจะใช้หมายเลขดัชนี (index หรือ subscript) ที่เป็นเลขจำนวนเต็ม ในการระบุตำแหน่งของข้อมูล ตัวแปรอาร์เรย์นั้น สามารถเรียกชื่อได้อีกอย่างหนึ่งว่า ตัวแปรชุด และในระบบคอมพิวเตอร์ก็จะมีการจัดสรรหน่วยความจำ (Memory) ให้กับตัวแปรอาร์เรย์ด้วย หากมีการกำหนดค่าขนาดของอาร์เรย์ไม่เหมาะสม เช่น กำหนดมากเกินไปแต่ไม่มีการใช้งานก็จะทำให้สิ้นเปลืองหน่วยความจำโดยเปล่าประโยชน์

3.1.1 ชนิดของตัวแปรอาร์เรย์

ตัวแปรอาร์เรย์ในภาษาซีพลัสพลัสนั้น สามารถแบ่งออกได้เป็นหลายชนิดด้วยกัน ซึ่งจะขึ้นอยู่กับการใช้งาน และปริมาณการเก็บข้อมูล โดยส่วนใหญ่จะแบ่งออกเป็น 2 ชนิด คือ

3.1.1.1 ตัวแปรอาร์เรย์ 1 มิติ

เป็นตัวแปรชุดที่เป็นพื้นฐาน และนิยมใช้งานมากที่สุด ใช้เก็บข้อมูลแบบแถวเดียว ตามชนิดของข้อมูลที่ได้ระบุไว้ เป็นตัวแปรอาร์เรย์แบบตัวอักขระ (Character) หรือเป็นตัวแปรอาร์เรย์ชนิดเลขจำนวนเต็ม (Integer) เป็นต้น ตัวอย่างเช่น การเก็บข้อมูลความสูงของนักเรียน 10 คน ดังภาพที่ 3.1



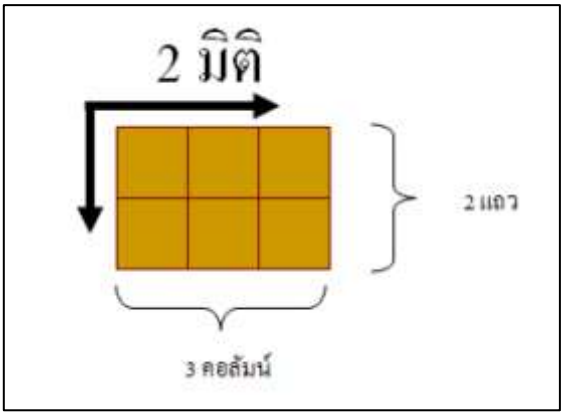
ภาพที่ 3.1 อาร์เรย์ 1 มิติ

3.1.2 ตัวแปรอาร์เรย์หลายมิติ

เป็นตัวแปรชุดที่สามารถเก็บข้อมูลได้หลายมิติ ซึ่งจะมีขนาดของมิติที่แตกต่างกันออกไปขึ้นอยู่กับข้อมูลที่ต้องการจัดเก็บในที่นี้จะยกตัวอย่างเฉพาะ ตัวแปรอาร์เรย์แบบ 2 มิติ และ 3 มิติ

3.1.2.1 ตัวแปรอาร์เรย์แบบ 2 มิติ

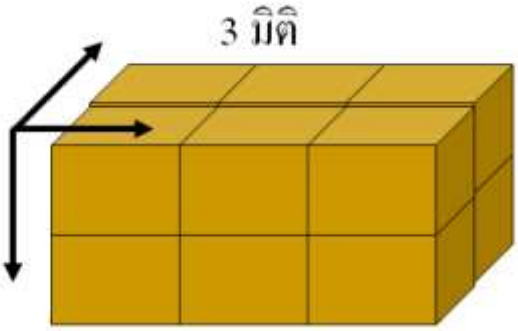
เป็นตัวแปรชุดที่มีการเก็บข้อมูลคล้าย ๆ ตาราง ซึ่งจะมีด้านกว้าง เรียกเป็นแถว (Row) และด้านยาว เรียกเป็น คอลัมน์ (Column) จากภาพที่ยกตัวอย่าง จะเป็นอาร์เรย์แบบ 2 แถว คูณ 3 คอลัมน์ (2x3) ตัวอย่างเช่น การเก็บข้อมูลความสูงของเด็กนักเรียน 2 คน จำนวน 3 ปี เป็นต้น ดังภาพที่ 3.2



ภาพที่ 3.2 อาร์เรย์ 2 มิติ

3.1.2.2 ตัวแปรอาร์เรย์แบบ 3 มิติ

เป็นตัวแปรชุดที่มีการจัดเก็บข้อมูลคล้าย ๆ ตารางลูกบาศก์สี่เหลี่ยม ซึ่งจะมีด้านกว้างยาวและลึก เพื่อใช้ในการจัดเก็บข้อมูล จากตัวอย่าง จะเป็นอาร์เรย์แบบ 2 x 3 x 2 ซึ่งสามารถจัดเก็บข้อมูลได้ 12 ชุดด้วยกัน ดังภาพที่ 3.3



ภาพที่ 3.3 อาร์เรย์ 3 มิติ

ปัจจุบันการเขียนโปรแกรมเพื่อเก็บข้อมูลกับอาร์เรย์ 3 มิติ และหลายมิติ ได้รับความนิยมน้อยลง หรือไม่มีการใช้งานเลย เนื่องจากการเข้าถึงอาร์เรย์ประเภทนี้ สามารถเข้าถึงได้ลำบาก และต้องเขียนโปรแกรมในการเรียกใช้ข้อมูลที่มีความยาวมาก จึงไม่เป็นที่นิยมใช้งานในปัจจุบัน จึงขอนำเสนอเฉพาะตัวแปรอาร์เรย์แบบ 1 และ 2 มิติเท่านั้น

3.2 การประกาศตัวแปรอาร์เรย์

เมื่อมีความต้องการใช้ตัวแปรอาร์เรย์ในโปรแกรมนั้น ก็มีความจำเป็นที่จะต้องสร้างและประกาศตัวแปรอาร์เรย์ขึ้นมาเหมือนกับตัวแปรชนิดอื่น ๆ ซึ่งจะต้องมีการระบุขนาดของตัวแปรที่จะสร้างขึ้นด้วย

3.2.1 การประกาศตัวแปรอาร์เรย์แบบ 1 มิติ

การประกาศตัวแปรอาร์เรย์นั้น จะมีความแตกต่างจากการประกาศตัวแปรชนิดอื่น ๆ เพราะจะต้องมีการระบุขนาดของตัวแปรอาร์เรย์ด้วย ซึ่งควรกำหนดขนาดให้มีความเหมาะสมกับการใช้งานจริง หากกำหนดขนาดของตัวแปรอาร์เรย์มาก หรือน้อยเกินไปก็จะทำให้สิ้นเปลืองหน่วยความจำและบางครั้งอาจไม่สามารถทำงานได้เลย

type_variable array_name[n]

อธิบาย

type_variable คือ ประเภทของตัวแปรที่ต้องการสร้าง เช่น int , char เป็นต้น

array_name คือ ชื่อของตัวแปรอาร์เรย์ที่ต้องการสร้าง โดยใช้หลักการเดียวกับตัวแปรชนิด อื่น ๆ เช่น name

n คือ ขนาดของตัวแปรอาร์เรย์ที่ต้องการสร้าง กำหนดเป็นตัวเลขจานวนเต็มบวก

ตัวอย่างโปรแกรมการประกาศตัวแปรอาร์เรย์ 1 มิติ

int grades[5];

เป็นการประกาศตัวแปรแบบอาร์เรย์ 1 มิติหรือโครงสร้างข้อมูลชื่อ grades ที่เก็บข้อมูลชนิดจำนวนเต็ม (int) จำนวน 5 ข้อมูลหรือเท่ากับ 5 อีลีเมนต์

ในการอ้างถึงแตละอีลีเมนต์ของอาร์เรย์สามารถกระทำได้ โดยการระบุค่าดัชนี (Index) โดยเลขดัชนีนี้จะมีค่าตั้งแต่ 0, 1, ..., (จำนวนของข้อมูล - 1) จากตัวอย่างข้างต้น เลขดัชนีของตัวแปรอาร์เรย์ grades มีค่าเป็น 0, 1, 2, 3, และ 4 ดังรูปที่ 3.4

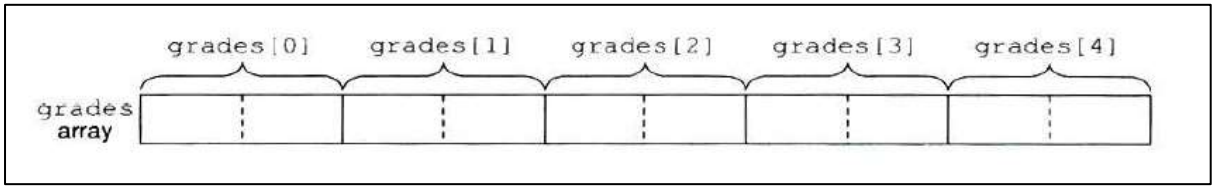
grades[0] อ้างถึงค่าจำนวนเต็มลำดับแรกที่เก็บในอาร์เรย์ชื่อ grades

grades[1] อ้างถึงค่าจำนวนเต็มลำดับแรกที่เก็บในอาร์เรย์ชื่อ grades

grades[2] อ้างถึงค่าจำนวนเต็มลำดับแรกที่เก็บในอาร์เรย์ชื่อ grades

grades[3] อ้างถึงค่าจำนวนเต็มลำดับแรกที่เก็บในอาร์เรย์ชื่อ grades

grades[4] อ้างถึงค่าจำนวนเต็มลำดับแรกที่เก็บในอาร์เรย์ชื่อ grades



ภาพที่ 3.4 แสดงโครงสร้างข้อมูลชื่อ grades ในหน่วยความจำ

3.2.2 การประกาศตัวแปรอาร์เรย์แบบ 2 มิติ

การประกาศตัวแปรอาร์เรย์แบบ 2 มิตินั้น มีความคล้ายกับการประกาศตัวแปรอาร์เรย์แบบ 1 มิติ แต่จะมีความแตกต่างกันเล็กน้อย คือ จะมีการระบุขนาดทั้งจานวนแถว (row) และจำนวนคอลัมน์ (column) ด้วย

type_variable array_name[n][m]

อธิบาย

type_variable คือ ประเภทของตัวแปรที่ต้องการสร้าง เช่น int , char เป็นต้น

array_name คือ ชื่อของตัวแปรอาร์เรย์ที่ต้องการสร้าง โดยใช้หลักการเดียวกับตัวแปรชนิดอื่น ๆ เช่น name

n คือ ขนาดของตัวแปรอาร์เรย์ที่ต้องการสร้าง กำหนดเป็นตัวเลขจานวนเต็มบวกที่ถูกแทนด้วยจำนวนแถว (row)

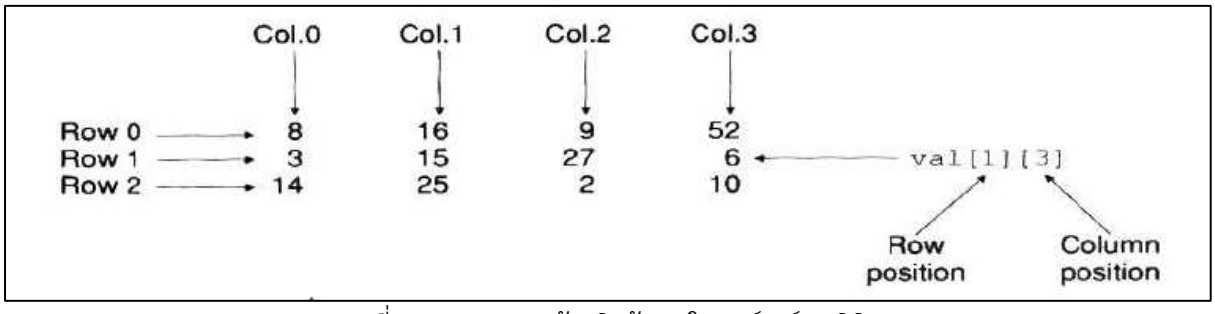
m คือ ขนาดของตัวแปรอาร์เรย์ที่ต้องการสร้าง กำหนดเป็นตัวเลขจานวนเต็มบวกที่ถูกแทนด้วยจำนวนคอลัมน์ (column)

ตัวอย่างโปรแกรมการประกาศตัวแปรอาร์เรย์ 2 มิติ

int val[3][4];

เป็นการประกาศตัวแปรชื่อ val เป็นตัวแปรอาร์เรย์ 2 มิติเพื่อเก็บข้อมูลจำนวนเต็ม ซึ่งจะจองหน่วยความจำเท่ากับ 4 * 3 * 4 = 48 ไบต์

การอ้างอิงถึงข้อมูลสำหรับอีลีเมนต์ใดในอาร์เรย์ 2 มิติ ทำได้โดยการระบุค่าเลขดัชนีทั้งในแนวแถวและแนวคอลัมน์ จากรูปที่ 3.5 ตำแหน่งของ val[1][3] จะสามารถบอกตำแหน่งกำหนดค่าดัชนีในแนวแถวที่ 1 และในแนวคอลัมน์ที่ 3 เช่นเดียวกับตัวแปรอาร์เรย์ 1 มิติ และตัวแปรทั่วไปที่สามารถใช้งานได้ที่ตำแหน่งต่าง ๆ ภายใน



รูปที่ 3.5 แสดงการอ้างอิงข้อมูลในอาร์เรย์ 2 มิติ

3.3 การกำหนดค่าให้กับตัวแปรอาร์เรย์

3.3.1 การกำหนดค่าให้กับตัวแปรอาร์เรย์แบบ 1 มิติ

type_variable array_name[n] = {value1,value2,value3,.....,valueN}

อธิบายรูปแบบ

type_variable คือ ประเภทของตัวแปรที่ต้องการสร้าง เช่น int , char เป็นต้น

array_name คือ ชื่อของตัวแปรอาร์เรย์ที่ต้องการสร้าง โดยใช้หลักการเดียวกับตัวแปรชนิดอื่น ๆ เช่น name

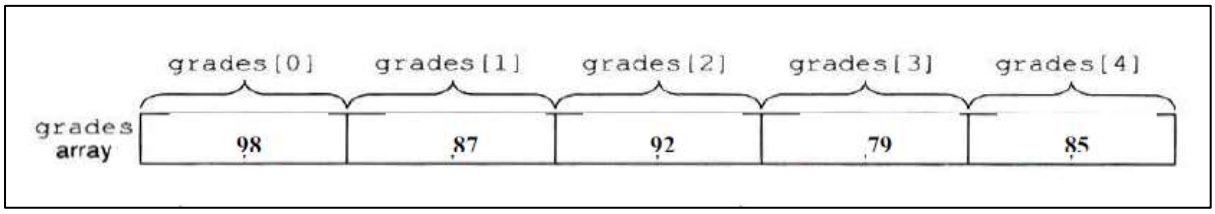
n คือ ขนาดของตัวแปรอาร์เรย์ที่ต้องการสร้าง กำหนดเป็นตัวเลขจำนวนเต็มบวกที่ถูกแทนด้วยจำนวนแถว (row)

value1 คือ ค่าของตัวแปรอาร์เรย์ค่าที่หนึ่ง ที่จะกำหนดให้ตัวแปรอาร์เรย์นั้น ๆ โดยจะถูกกำหนดค่าไปเรื่อย ๆ จนถึง valueN ซึ่งเป็นค่าของตัวแปรอาร์เรย์ตัวสุดท้าย

ตัวอย่าง การกำหนดค่าเริ่มต้นของตัวแปรแบบอาร์เรย์ 1 มิติ สามารถกำหนดได้ โดยให้อยู่ในเครื่องหมาย “{..,..,..,..}” และใช้ได้กับทุกชนิดของข้อมูล ดังตัวอย่างต่อไปนี้

```
int grades[5] = {98,87,92,79,85};
char codes[6] = {'s', 'a', 'm', 'p', 'l', 'e'};
double width[3] = {10.96, 6.43, 2.58};
float temp[4] = {98.6, 97.2, 99.0 , 101.5};
```

ค่าแรกจะถูกกำหนดให้กับอีลีเมนต์ตัวที่ 1 (ดัชนีที่ 0) และ ค่าที่สองจะถูกกำหนดให้อีลีเมนต์ตัวที่ 2 (ดัชนีที่ 1) และ ถัด ๆ ไป ดังในรูปที่ 3.6 ซึ่งแสดงค่าในอาร์เรย์ grades หลังจากการกำหนดค่าเริ่มต้นแล้วข้างต้น



รูปที่ 3.6 แสดงการเก็บค่าเริ่มต้นในโครงสร้างข้อมูลชื่อ grades ในหน่วยความจำ

หากในการกำหนดค่าเริ่มต้นไม่ได้มีการกำหนดขนาดของอาร์เรย์ คอมไพเลอร์จะกำหนดขนาดของอาร์เรย์ให้เท่ากับจำนวนอีลีเมนต์ที่ได้มีการกำหนดค่าเริ่มต้น เช่น

int x[ ] = {15,8,7,10};

ขนาดของอาร์เรย์ x จะได้เท่ากับ 4

การกำหนดค่าให้กับอาร์เรย์

ถ้าประกาศอาร์เรย์ เป็น int a[2]; แสดงว่า array นี้มีสมาชิก 2 ตัวคือ a[0] และ a[1] เนื่องจากค่าดัชนี (index) จะเริ่มจาก 0 จึงสามารถกำหนดค่าอาร์เรย์ ได้ดังนี้

a[0] = 11;

a[1] = 22;

การใช้ตัวแปรในการกำหนดค่าให้กับอาร์เรย์ สามารถทำได้ต่อไปนี้

i = 0; b1 = 50; j = 1; b2 = 100;

สามารถกำหนดค่าอาร์เรย์ ได้เป็น a[i] = b1; a[j] = b2; และสามารถกำหนดค่าอาร์เรย์ โดยใช้ cin เพื่อรับค่าจากไฟล์ได้เช่นเดียวกัน

ตัวอย่างที่ 3.1

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
  cout << cars[0];
  return 0;
}
```

ผลลัพธ์

Volvo

ตัวอย่างที่ 3.1

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
  cars[0] = "Opel";
  cout << cars[0];
  return 0;
}
```

ผลลัพธ์

Opel

ตัวอย่างที่ 3.3 จงเขียนโปรแกรมรับข้อมูลเลขจำนวนเต็ม จำนวน 10 ชุด ทางแป้นพิมพ์และให้แสดงผลลัพธ์ทางจอภาพโดยเรียงจากข้อมูลหลังสุด ไปหน้าสุด

```
1    #include <iostream>
2    using namespace std;
3    int main()
4    {
5            const int size = 10;
6            int x[size];
7            cout << "Pleas enter 10 values." << endl;
8            for (int i = 0; i < size && cin >> x[i]; ++i);
9            for (int i = size-1; i >=0 ; --i)
10                   cout << x[i] << "\t";
11   return 0;
12 }
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
5	size เป็นตัวแปรชนิด int มีค่าคงที่ เท่ากับ 10
6	x เป็นตัวแปรอาร์เรย์ ชนิด int มีจำนวนอีลิเมนต์เท่ากับ 10
8	วนรอบโดย รับค่าเลขจำนวนเต็ม มาเก็บไว้ที่ตัวแปร x จนครบ10 จำนวน
9-10	วนรอบ เพื่อนำข้อมูลที่เก็บตัวแปรอาร์เรย์ X มาแสดง โดยเริ่มจาก X[9] แล้วลดลงมาค่าละหนึ่ง จนถึง X[0]

3.3.2 การกำหนดค่าให้กับตัวแปรอาร์เรย์แบบ 2 มิติ

type_variable array_name[n][m] = {value1,value2,value3,.....,valueN}

อธิบายรูปแบบ

type_variable คือ ประเภทของตัวแปรที่ต้องการสร้าง เช่น int , char เป็นต้น

array_name คือ ชื่อของตัวแปรอาร์เรย์ที่ต้องการสร้าง โดยใช้หลักการเดียวกับตัวแปรชนิดอื่น ๆ เช่น name

n คือ ขนาดของตัวแปรอาร์เรย์ที่ต้องการสร้าง กำหนดเป็นตัวเลขจานวนเต็มบวกที่ถูกแทนด้วยจำนวนแถว (row)

m คือ ขนาดของตัวแปรอาร์เรย์ที่ต้องการสร้าง กำหนดเป็นตัวเลขจำนวนเต็มบวกที่ถูกแทนด้วยจำนวนคอลัมน์ (column)

value1 คือ ค่าของตัวแปรอาร์เรย์ค่าที่หนึ่ง ที่จะกำหนดให้ตัวแปรอาร์เรย์นั้น ๆ โดยจะถูกกำหนดค่าไปเรื่อย ๆ จนถึง valueN ซึ่งเป็นค่าของตัวแปรอาร์เรย์ตัวสุดท้าย

การกำหนดค่าเริ่มต้นในตัวแปรอาร์เรย์ 2 มิติ สามารถกระทำได้เช่นเดียวกันกับตัวแปรอาร์เรย์ 1 มิติ โดยจะเพิ่ม ส่วนการแบ่งแยกในแต่ละแถวด้วยเครื่องหมาย “{ }” และ “,” เช่น

```
int val[3][4] =     {{8,16, 9,52},
                     {3,15,27, 6},
                     {14,25,2,10}};
```

จากคำสั่งเป็นการประกาศตัวแปร val เป็นแบบอาร์เรย์ 2 มิติ ที่มีชนิดของข้อมูลเป็นจำนวนเต็ม โดยมีขนาดเป็น 3 แถว 4 คอลัมน์และมีการประกาศค่าเริ่มต้นด้วย ในการกำหนดค่าเริ่มต้นค่าในเครื่องหมายปีกกาชุดแรกจะเป็นการกำหนดค่าให้กับอาร์เรย์ในแถวที่ 0 และเครื่องหมายปีกกาชุดที่สอง

จะเป็นการกำหนดค่าให้กับอาร์เรย์ในแถวที่ 1 และเครื่องหมายปีกกาชุดที่สามจะเป็นจะเป็นการกำหนดค่าให้กับอาร์เรย์ในแถวที่ 2

ตัวอย่างที่ 3.4

```
#include <iostream>
using namespace std;

int main() {
  string letters[2][4] = {
    { "A", "B", "C", "D" },
    { "E", "F", "G", "H" }
  };

  cout << letters[0][2];
  return 0;
}
```

ผลลัพธ์

C

ตัวอย่างที่ 3.5 โปรแกรมอาร์เรย์ 2 มิติ มีจำนวน 6 อีลิเมนต์

```
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4 int main()
5 {
6     const int sizeCol = 3;
7     double ma[ ][sizeCol] = {{1.1, 1.2, 1.3},{2.1, 2.2}};
8     cout << "Detail of declaring"<<endl;
9      cout << "Array ma[][sizeCol] = {{1.1, 1.2, 1.3},{2.1, 2.2}}" << endl;
```

```
10  cout << "Size of whole array \t= " << sizeof ma << " \tbytes" << endl;
11  cout << "Number of total element\t= " << sizeof ma / sizeof (double)
            << " \telements" << endl;
12          for (int i = 0; i < 2;++i)
13                  for (int j = 0; j < sizeCol; ++j)
14                          cout << "value ma[" << i << "][" << j << "] = " << ma[i][j] << endl;
15  return 0;
16 }
```

ผลลัพธ์

Detail of declaring

Array ma[][sizeCol] = {{1.1, 1.2, 1.3},{2.1, 2.2}}

Size of whole array = 48 bytes

Number of total element = 6 elements

value ma[0][0] =1.1

value ma[0][1] =1.2

value ma[0][2] =1.3

value ma[1][0] =2.1

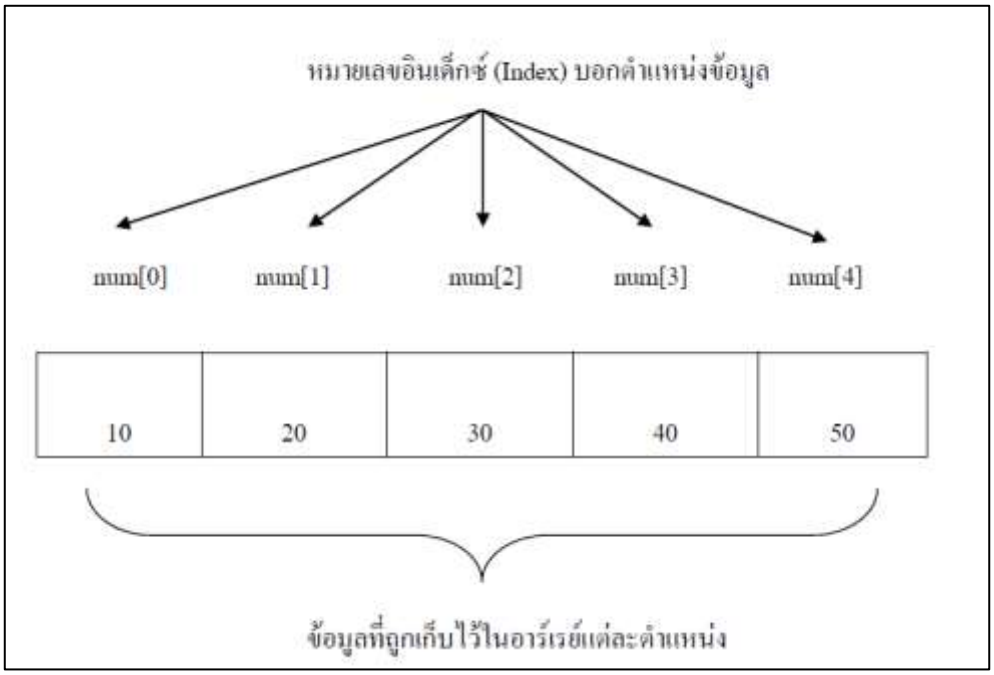
value ma[1][1] =2.2

value ma[1][2] =0

บรรทัดที่	คำอธิบาย
7	ma เป็นตัวแปรอาร์เรย์ 2 มิติ ชนิด double มีจำนวน 6 อีลิเมนต์
10	แสดงหน่วยความจำที่ใช้เป็นจำนวนเท่าไร
11	แสดงจำนวนลีเมนต์ทั้งหมดของตัวแปรอาร์เรย์ ma
12-14	แสดงข้อมูลที่อยู่ในแต่ละลีเมนต์

3.3.3 การเข้าถึงค่าข้อมูลในตัวแปรอาร์เรย์

หลังจากที่สามารถกำหนดค่าให้กับตัวแปรอาร์เรย์ต่าง ๆ ได้แล้ว ต่อไปที่กระทำได้กับตัวแปรอาร์เรย์คือ การเข้าถึงค่าข้อมูลในตัวแปรอาร์เรย์ที่ต้องการ ในการเข้าถึงข้อมูลในตัวแปรอาร์เรย์แต่ละตำแหน่งนั้น จำเป็นอย่างยิ่งที่จะต้องทราบหมายเลขประจำตำแหน่งข้อมูล หรือที่เรียกว่า หมายเลขอินเด็กซ์ (Index) ค่าอินเด็กซ์นี้จะเป็นค่าตัวเลขที่ใช้อ้างอิงตำแหน่งของอาร์เรย์ แต่ละตัว และค่าอินเด็กซ์นี้จะเริ่มต้นที่ตำแหน่งที่ศูนย์ (0) ก็คือตำแหน่งแรกสุด โดยเลขบอกตำแหน่งอินเด็กซ์นี้จะถูกเขียนไว้ในเครื่องหมาย [ ] และหมายเลขอินเด็กซ์นี้จะเพิ่มค่าขึ้นทีละหนึ่งไปเรื่อย ๆ เพื่อให้สามารถเก็บข้อมูลไปจนถึงตำแหน่งสุดท้าย ดังภาพที่ 3.5



ภาพที่ 3.5 การเข้าถึงค่าข้อมูลในตัวแปรอาร์เรย์

ตัวอย่างโปรแกรม

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
  cout << "cars[0]"<<cars[0]<<endl;
  cout << "cars[1]"<<cars[1]<<endl;
  cout << "cars[2]"<<cars[2]<<endl;
  cout << "cars[3]"<<cars[3]<<endl;
  return 0;
}
```

ผลลัพธ์ทางจอภาพ

การเข้าถึงค่าข้อมูลของตัวแปรอาร์เรย์แต่ละตำแหน่งนั้น จะทำให้เข้าใจถึงการจัดเก็บข้อมูลที่อยู่ในตัวแปรอาร์เรย์มากยิ่งขึ้น ซึ่งจะสอดคล้องกับผลลัพธ์ที่ปรากฏบนจอภาพ

```
cars[0]Volvo
cars[1]BMW
cars[2]Ford
cars[3]Mazda
```

สำหรับการเข้าถึงค่าข้อมูลในตัวแปรอาร์เรย์แบบ 2 มิตินั้น ก็จะใช้หลักการคล้าย ๆ กับการเข้าถึงค่าข้อมูลในตัวแปรอาร์เรย์แบบ 1 มิติ โดยจะมีการอ้างอิงเลขตำแหน่งแถว (Row) และคอลัมน์ (Column) ของตำแหน่งที่ต้องการเข้าถึงข้อมูลนั้น ๆ

ตัวอย่างโปรแกรม

```
#include <iostream>
using namespace std;
```

```
int main()
{
    char letters[2][4] = {
        {'A','B','C','D'},
        {'E','F','G','H'}
    };
    cout << "letters[0][0] : " << letters[0][0] <<endl;
    cout << "letters[0][1] : " << letters[0][1] << endl;
    cout << "letters[0][2] : " << letters[0][2] << endl;
    cout << "letters[0][3] : " << letters[0][3] << endl;
    cout << "letters[1][0] : " << letters[1][0] << endl;
    cout << "letters[1][1] : " << letters[1][1] << endl;
    cout << "letters[1][2] : " << letters[1][2] << endl;
    cout << "letters[1][3] : " << letters[1][3] << endl;
    letters[0][3] = 'Z';
    cout << "letters[0][3] : " << letters[0][3] << endl;
}
```

ผลลัพธ์ทางจอภาพ

จากโปรแกรมตัวอย่าง จะเห็นได้ว่าการเข้าถึงค่าข้อมูลของตัวแปรแบบ 2 มิตินั้น ไม่ได้ยากแต่อย่างใด ก็จะมีการอ้างอิงตำแหน่งของตัวแปรอาร์เรย์คล้าย ๆ กับแบบ 1 มิติ แต่จะเป็นการอ้างอิงตำแหน่งแถวและคอลัมน์ และจากตัวอย่างที่ได้ทดสอบนั้นจะเห็นได้ว่า สามารถแทนที่ค่าใหม่ให้กับ ตัวแปรอาร์เรย์ได้ทันทีเพียงอ้างอิงตำแหน่งแถว และคอลัมน์และค่าที่ต้องการแทนที่

```
letters[0][0] : A
letters[0][1] : B
letters[0][2] : C
letters[0][3] : D
letters[1][0] : E
letters[1][1] : F
```

letters[1][2] : G

letters[1][3] : H

letters[0][3] : Z

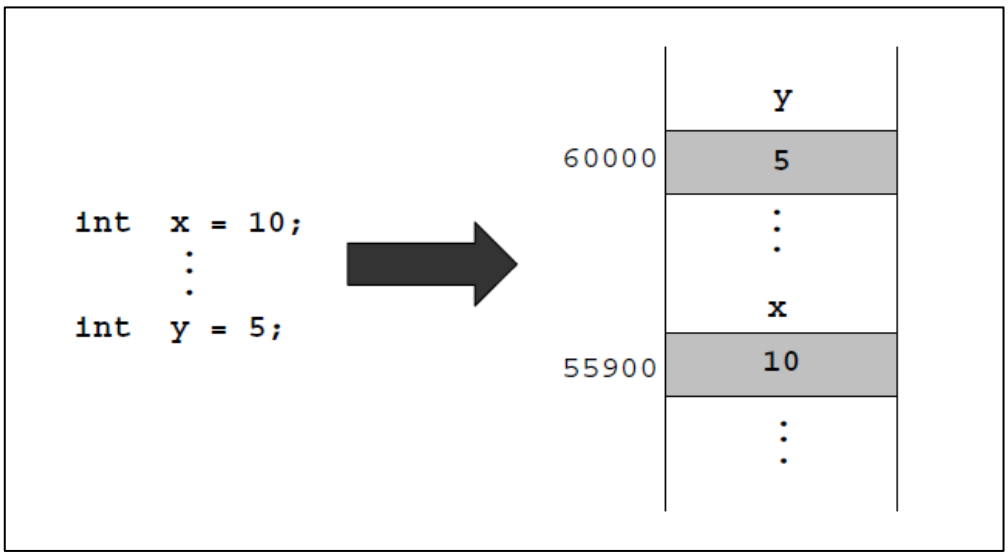
3.4 ตัวแปรพอยน์เตอร์ (Pointer)

ในการประกาศตัวแปรแต่ละครั้ง ไม่ว่าจะเป็นตัวแปรชนิด int float char หรือชนิดใดก็ตามตัวแปรเหล่านี้จะถูกเก็บอยู่ในหน่วยความจำของคอมพิวเตอร์ ซึ่งตัวแปรแต่ละตัวก็จะมีตำแหน่งระบุเอาไว้ว่าอยู่ในส่วนของหน่วยความจำ เรียกว่า “แอดเดรส (Address) ของตัวแปร” ซึ่งในการใช้งานตัวแปรโดยทั่วไปนั้นจะกำหนดได้เฉพาะค่าของตัวแปรเท่านั้น แต่ไม่สามารถที่จะกำหนดแอดเดรสของตัวแปรได้ ตัวอย่างเช่น มีตัวแปรอยู่ 2 ตัวคือ x และ y

ตัวแปร x มีแอดเดรสอยู่ที่ตำแหน่ง 55900 และค่าของ x คือ 10

ตัวแปร y มีแอดเดรสอยู่ที่ตำแหน่ง 60000 และค่าของ y คือ 5

ดังในรูปที่ 3.6



ภาพที่ 3.6 การเก็บค่าของตัวแปรในหน่วยความจำ

หากต้องการที่จะรู้ว่าตัวแปรแต่ละตัวอยู่ที่ตำแหน่งใดในหน่วยความจำ ให้ใช้เครื่องหมาย & นำหน้าตัวแปร ตัวอย่างเช่น แอดเดรสของตัวแปร x เขียนแทนได้ด้วย &x และ แอดเดรสของตัวแปร y เขียนแทนได้ด้วย &y เมื่อต้องการเก็บตำแหน่งในหน่วยความจำไว้ที่ตัวแปร จะมีตัวแปรอยู่ชนิดหนึ่งที่ใช้ในการเก็บค่าแอดเดรสของตัวแปรอื่น ตัวแปรชนิดนั้นคือ “พอยน์เตอร์ (Pointer)”

3.4.1 การประกาศตัวแปรพอยน์เตอร์

ในการประกาศตัวแปรพอยน์เตอร์จะใช้สัญลักษณ์ * เป็นการบ่งบอกว่าตัวแปรที่ประกาศนั้นเป็นพอยน์เตอร์ ตัวอย่างเช่น

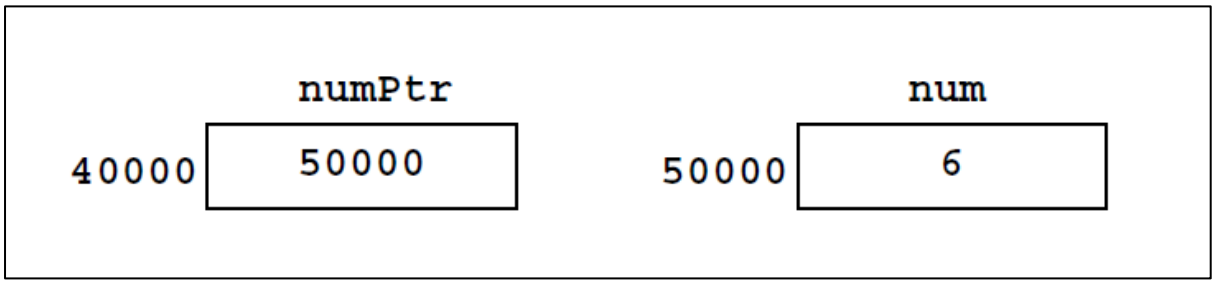
```
int *numPtr;
```

เป็นการประกาศตัวแปร numPtr เป็นชนิดพอยน์เตอร์ที่ใช้เก็บแอดเดรสของตัวแปรชนิด int ได้ และจากที่ได้กล่าวไปแล้วว่าตัวแปรพอยน์เตอร์จะใช้ในการเก็บค่าแอดเดรสของตัวแปรตัวอื่น ดังนั้นในการกำหนดค่าของตัวแปรพอยน์เตอร์ ก็มักจะกำหนดด้วยแอดเดรสของตัวแปร

ตัวอย่างเช่น

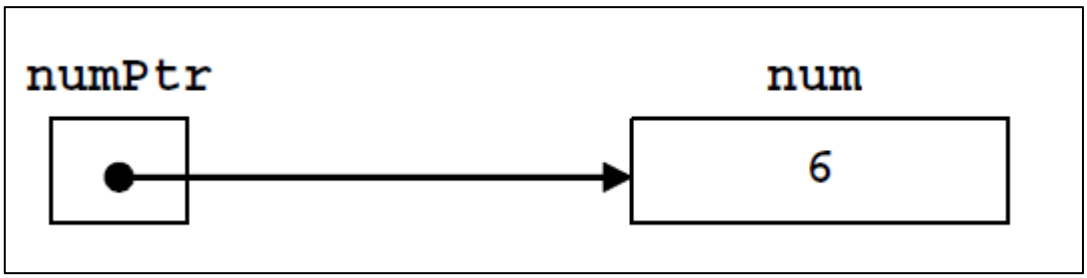
```
int num = 6;
int *numPtr;
numPtr = &num;
```

เป็นการกำหนดให้ numPtr มีค่าเท่ากับแอดเดรสของตัวแปร num ซึ่งจากรูปที่ 3.7 ถ้าสมมุติว่าตัวแปร num ถูกเก็บอยู่ที่ตำแหน่ง 50000 ตัวแปรพอยน์เตอร์ numPtr ก็จะมีค่าเท่ากับ 50000 ด้วย



ภาพที่ 3.7 การเก็บค่าของ num และ numPtr ในหน่วยความจำ

แต่ในการเขียนโปรแกรมนั้น ไม่จำเป็นจะต้องรู้ว่าว่าตัวแปร num ถูกเก็บอยู่ที่ตำแหน่งไหน เพียงแค่รู้ว่าตัวแปร numPtr เป็นพอยน์เตอร์ที่ใช้ในการอ้างอิงไปยังตำแหน่งของตัวแปร num ได้ก็พอ ซึ่งมักจะใช้คำพูดว่า “numPtr ชี้ไปยัง num” และในการอธิบายก็จะใช้แผนภาพตามภาพที่ 3.7



ภาพที่ 3.8 แผนภาพที่ใช้ในการอธิบายว่า numPtr ชี้ไปยัง num

ในการอ้างถึงค่าในตำแหน่งที่พอยน์เตอร์ชี้อยู่นั้น ให้ใช้เครื่องหมาย * นำหน้าตัวแปร ซึ่งเรียกว่า การ Dereferencing ตัวอย่างเช่น จากคำสั่งต่อไปนี้

```
cout << *numPtr;
```

ค่าที่พิมพ์ออกทางหน้าจอคือ 6 เนื่องจากในภาพที่ 3.8 นั้นจะเห็นว่า numPtr ชี้ไปยังตัวแปร num ดังนั้นค่าในตำแหน่งที่ numPtr ชี้อยู่ก็คือค่าของ num นั้น เอง ซึ่งก็คือ 6 หากสังเกตจะเห็นว่าสามารถแทนค่าของ *numPtr ด้วย num ได้เลย

นอกจากนี้หากต้องการแก้ไขค่าในตำแหนงที่พอยน์เตอร์ชี้อยู่ก็สามารถทำได้ในทำนองเดียวกันตัวอย่างเช่น จากคำสั่ง

```
*numPtr = 20;
```

เป็นการกำหนดให้ค่าในตำแหน่งที่ numPtr ชี้อยู่มีค่าเท่ากับ 20 ดังนั้นค่าของตัวแปร num ก็จะถูกเปลี่ยนเป็น 20 ไปด้วย

ตัวอย่างที่ 3.4 โปรแกรมแสดงการใช้พอยน์เตอร์

```
#include <iostream>
using namespace std;
int main ()
{
    int firstvalue=2, secondvalue=3;
    int * mypointer;
    cout << "Befor firstvalue is " << firstvalue << endl;
    cout << "Befor secondvalue is " << secondvalue << endl;
    mypointer = &firstvalue;
    *mypointer = 10;
    mypointer = &secondvalue;
    *mypointer = 20;
    cout << "After firstvalue is " << firstvalue << endl;
    cout << "After secondvalue is " << secondvalue << endl;
return 0;
}
```

ผลลัพธ์

Befor firstvalue is 2

Befor secondvalue is 3

After firstvalue is 10

After secondvalue is 20

3.4.2 พอยน์เตอร์และอาร์เรย์ (pointers and arrays)

จากเรื่องอาร์เรย์ถ้ามีตัวแปรอาร์เรย์อยู่ 1 ชุด ชื่อตัวแปรอาร์เรย์ จะหมายถึงแอดเดรสของอีลีเมนต์แรก เช่น หากประกาศอาร์เรย์ int b[5];

ตัวแปรอาร์เรย์ b จะมีค่าเท่ากับ &b[0] ซึ่งจะเห็นได้ว่าตัวแปรอาร์เรย์ก็มีลักษณะคล้ายกับพอยน์เตอร์เนื่องจากใช้เก็บค่าแอดเดรสเมือนกัน แต่ก็มีแตกต่างอยู่ก็คือจะไม่สามารถเปลี่ยนแปลงค่าของตัวแปรอาร์เรย์ให้เลื่อนไปชี้ที่อื่นเหมือนอย่างตัวแปรพอยน์เตอร์ได้ เช่น จะเพิ่มค่าของ b โดยใช้คำสั่ง b = b + 2; นั้นไม่สามารถทำได้ ดังนั้นตัวแปรอาร์เรย์มักจะถูกเรียกว่า Constant Pointer ซึ่งก็คือ พอยน์เตอร์ที่ไม่มีการเปลี่ยนแปลงนั่นเอง ตัวแปรอาร์เรย์จะชี้อยู่ที่เดียวก็คือที่แอดเดรสของอีลีเมนต์แรก

เนื่องจากอาร์เรย์ก็จัดเป็นพอยน์เตอร์ชนิดหนึ่ง ดังนั้นจึงสามารถที่จะกำหนดให้พอยน์เตอร์แทนอาร์เรย์ได้ ตัวอย่างเช่น

```
int b[5];
int *bPtr;
bPtr = b;
หรือ bPtr = &b[0];
```

นอกจากนี้ยังสามารถใช้สัญลักษณ์ [ ] กับพอยน์เตอร์ได้อีกด้วย เช่น

bPtr[2] หมายถึง ค่าในตำแหน่งที่ถัดจาก bPtr ชี้อยู่ไปอีก 2 บล็อก

bPtr[0] หมายถึง ค่าในตำแหน่งเดียวกับที่ bPtr ชี้อยู่

ซึ่ง bPtr[0] ก็จะมีค่าเท่ากับ b[0] และ bPtr[2] ก็จะมีค่าเท่ากับ b[2]

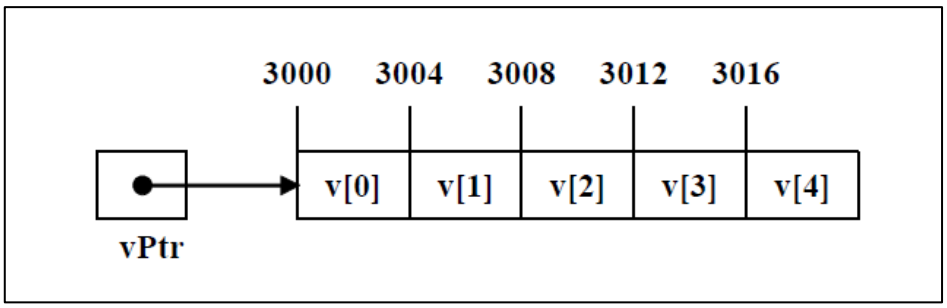
3.4.3 การใช้ตัวดำเนินการทางคณิตศาสตร์กับพอยน์เตอร์ (Pointer Arithmetic)

เนื่องจากตัวแปรพอยน์เตอร์ใช้ในการเก็บแอดเดรสของตัวแปรตัวอื่น ดังนั้นจึงสามารถใช้ตัวดำเนินการทางคณิตศาสตร์ในการแก้ไขค่าของพอยน์เตอร์ได้ แต่ใช้ได้เฉพาะตัวดำเนินการบางตัวเท่านั้น คือ ตัวดำเนินการเพิ่มค่า (++) ตัวดำเนินการลดค่า (--) การเพิ่มค่าหรือลดค่าพอยน์เตอร์ด้วยจำนวนเต็มโดยใช้เครื่องหมาย +,-,+=,-= จากคำสั่ง ต่อไปนี้

int v[5];

int *vPtr;

เป็นการประกาศอาร์เรย์ของ int ชื่อ v ขนาด 5 อีลีเมนต์ และประกาศพอยน์เตอร์ 1 ตัวชื่อ vPtr โดยสมมุติว่าอีลีเมนต์แรก (v[0]) ตั้งอยู่ที่ตำแหน่ง 3000 ดังที่แสดงในภาพที่ 3.9



ภาพที่ 3.9 อาร์เรย์ v และพอยน์เตอร์ vPtr ที่ชี้ไปยังอีลีเมนต์แรกของ v

และหากต้องการกำหนดให้พอยน์เตอร์ vPtr ชี้ไปยังอาร์เรย์ v ก็สามารถทำได้ 2 แบบคือ

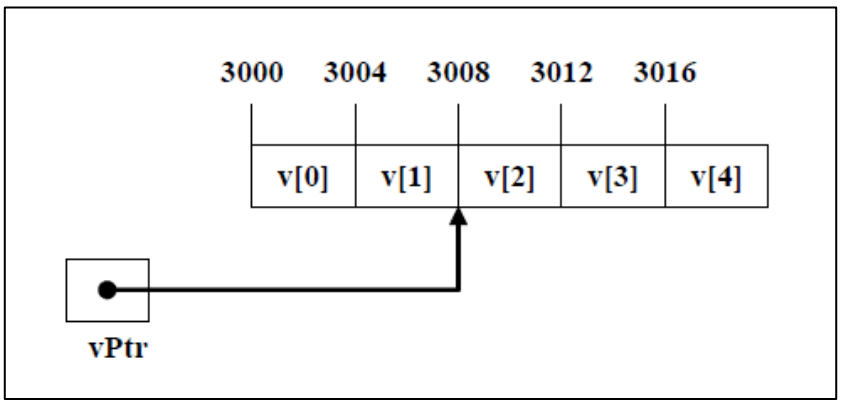
vPtr = v;

หรือ vPtr = &v[0]; ก็ได้

ทั้งนี้เนื่องจากว่าชื่อของอาร์เรย์จะหมายถึงแอดเดรสของอีลีเมนต์แรกอยู่แล้ว ดังนั้น v จึงมีค่าเท่ากับ &v[0] จากรูปที่ 4-5 เมื่อกำหนดให้ vPtr ชี้ไปที่อาร์เรย์ v แล้ว ค่าที่เก็บใน vPtr ก็จะมีค่าเท่ากับ 3000 นั่นเอง และหากต้องการเพิ่มค่าของพอยน์เตอร์ v ก็สามารถทำได้ เช่น

vPtr = vPtr + 2;

ซึ่งดูแล้วค่า vPtr น่าจะเปลี่ยนเป็นค่า 3002 (3000 + 2) แต่สำหรับพอยน์เตอร์แล้ว การเพิ่มค่าหรือลดค่าของพอยน์เตอร์จะไม่เหมือนกับตัวแปรโดยทั่วไป สำหรับคำสั่งนี้จะทำให้ค่าของ vPtr เปลี่ยนไปเป็น 3008 ซึ่งมาจาก 3000 + 2*4 โดย 4 คือขนาด int (เนื่องจาก vPtr เป็นพอยน์เตอร์ที่ใช้ชี้ไปยังตัวแปร int) ซึ่งตำแหน่ง 3008 ก็คือแอดเดรสของ v[2] นั่นเอง ดังที่ได้แสดงในภาพที่ 3.10



ภาพที่ 3.10 การเพิ่มค่าของพอยน์เตอร์

สรุปได้ว่าการเพิ่มค่าของ vPtr ด้วย 2 เป็นการเลื่อนตำแหน่งของ vPtr ไปข้างหน้าอีก 2 บล็อกของ int (2 * 4) และในทำนองเดียวกันถ้าหาก vPtr เป็นพอยน์เตอร์ที่ใช้ชี้ไปยังตัวแปร char การเพิ่มค่า vPtr ด้วย 2 ก็จะเป็นการเลื่อนตำแหน่งของ vPtr ไปข้างหน้าอีก 2 บล็อกของ char (2 * 1) นอกจากนี้ยังสามารถเลื่อนพอยน์เตอร์ถอยหลังได้อีกด้วยโดยการลดค่าของพอยน์เตอร์นั่นเอง ซึ่งการเลื่อนพอยน์เตอร์ในลักษณะนี้จะมีประโยชน์มากเมื่อต้องนำพอยน์เตอร์มาใช้ในการเข้าถึงค่าของอาร์เรย์

ตัวอย่างที่ 4.5 การใช้ตัวดำเนินการทางคณิตศาสตร์ในพอยน์เตอร์

```
#include <iostream>
using namespace std;
int main ()
{
    int numbers[5];
    int * p;
    p = numbers; *p = 10;
    p++; *p = 20;
    p = &numbers[2]; *p = 30;
    p = numbers + 3; *p = 40;
    p = numbers; *(p+4) = 50;
    for (int n=0; n<5; n++)
        cout << numbers[n] << ", ";
    return 0;
}
```

ผลลัพธ์

```
    10, 20, 30, 40, 50,
```

3.4.4 สตริงกับพอยน์เตอร์ (pointer and string)

สตริงหรือข้อความก็คืออาร์เรย์ของ char นั่นเอง โดยสตริงจะมีตัวอักษรปิดท้ายคือ '\0' แต่ความจริงแล้วสตริงนั้นมีอยู่ 3 ประเภท คือ ชนิดข้อมูลสตริง อาร์เรย์ของ char และพอยน์เตอร์ชนิด char *

ชนิดข้อมูลสตริง

ชนิดข้อมูลสตริง เป็นส่วนหนึ่งของไลบารีมาตรฐานใน ซีพลัสพลัส ในคลาสสตริงสามารถนำมากำหนดชนิดข้อมูลของตัวแปรได้ เช่น string mystring="Computer" ตัวแปรที่ถูกกำหนดเป็นชนิดข้อมูลสตริงนั้นนำไปใช้งานได้ตามปกติลักษณะตัวแปรทั่วไป

อาร์เรย์ของ char

อาร์เรย์ของ char คือตัวแปรที่ใช้เก็บสตริงไว้ในตัวแปรอาร์เรย์ โดยแต่ละอีลีเมนต์ก็จะเก็บตัวอักษรแต่ละตัวเอาไว้ และจะต้อง '\0' เป็นตัวปิดท้าย เช่น

char name[8] = "Somchai";

ค่าที่เก็บในแต่ละอีลีเมนต์ของอาร์เรย์ name จะเป็นไปตามภาพที่ 3.11

name[0]	name[1]	name[2]	name[3]	name[4]	name[5]	name[6]	name[7]
'S'	'o'	'm'	'c'	'h'	'a'	'i'	'\0'

ภาพที่ 3.11 ค่าในอาร์เรย์ name หลังจากที่ได้กำหนดค่าเริ่มต้นแล้ว

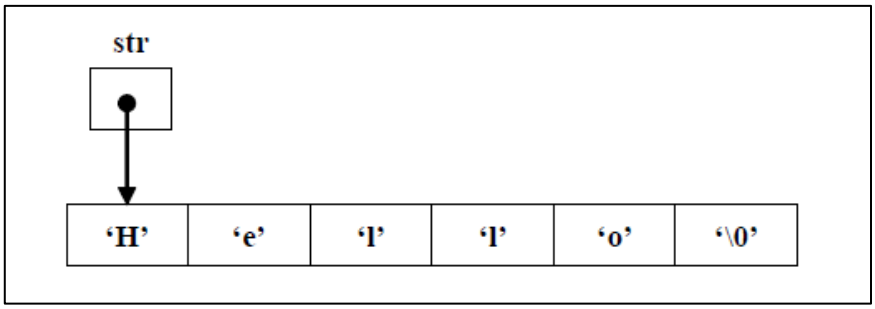
พอยน์เตอร์ชนิด char *

พอยน์เตอร์ชนิด char * เป็นตัวแปรใช้เก็บสตริงได้เหมือนกับอาร์เรย์ แต่ต่างกันตรงที่ว่าพอยน์เตอร์ชนิด char * ไม่ได้ใช้เป็นที่เก็บตัวอักษรเหมือนกับอาร์เรย์ แต่ใช้ในการชี้ไปยังตำแหน่งของตัวอักษรตัวแรกเท่านั้น เนื่องจากว่าถ้ารู้วาตัวอักษรตัวแรกอยู่ที่ไหนกจะรู้ได้ว่าข้อความทั้งหมดเป็นอย่างไร เพราะว่าสตริงจะต้องมี '\0' ปิดท้ายเสมอ พอยน์เตอร์ชนิด char * มีการใช้งานที่ต่างกับอาร์เรย์อยู่เล็กน้อย คือ สามารถใช้เครื่องหมาย = ได้ทั้งตอนที่ให้ค่าเริ่มต้นพร้อมกับการประกาศตัวแปร หรือว่าการกำหนดค่าในภายหลังก็ได้ เช่น

char *str;

str = "Hello";

จากการทำงานของคำสั่งนี้จะเริ่มจากการจองหน่วยความจำชั่วคราวขึ้นมาเพื่อเก็บข้อความว่า "Hello" จากนั้นก็นำพอยน์เตอร์ str ไปชี้อยู่ที่ตำแหน่งของตัวอักษรตัวแรกดังที่แสดงในภาพที่ 3.12



รูปที่ 3.12 พอยน์เตอร์ str ชี้อยู่ที่ตำแหน่งแรกของข้อความ "Hello"

จากที่ได้กล่าวมา จึงสามารถใช้ประโยชน์ตัวแปรพอยน์เตอร์ กับข้อความได้

ตัวอย่างทหี่ 4.6 การใช้พอยน์เตอร์กับสตริง

```
1    #include <iostream>
2    using namespace std;
3    int main()
4    {
5            char msg[10];
6            char *ptr;
7            cout << "Enter text to reverse: ";
8            cin >> msg;
9            int len = strlen(msg);
10           ptr = &msg[len-1];
11           while(ptr >= &msg[0])
12           {
13                   cout << *ptr;
14                   ptr--;
15           }
16           return 0;
17   }
```

ผลลัพธ์

Enter text to reverse: Program

margorP

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
5	msg เป็นตัวแปรอาร์เรย์ ชนิด char มีจำนวน 10 อีลิเมนต์
6	ptr เป็นตัวแปรพอยน์เตอร์
7	แสดงข้อ Enter text to reverse
8	รับข้อความจากแป้นพิมพ์ไปเก็บไว้ที่ตัวแปรอาร์เรย์ msg
9	สร้างตัวแปร len ชนิด int เก็บจำนวนตัวอักษรทั้งหมด
10	ใช้ตัวแปร ptr เก็บตำแหน่งก่อนหน่วยความจำสุดท้าย
11	ใช้คำสั่ง while ตรวจสอบเงื่อนไขว่า ตำแหน่งหน่วยความจำในตัวแปร ptr มีค่ามากกว่าตำแหน่งหน่วยความจำแรกของตัวแปร msg
13	จากบรรทัด 11 ถ้าเงื่อนไขเป็นจริงให้แสดงตัวอักษรที่ตัวแปร ptr ชี้อยู่
14	ให้เลื่อนตำแหน่งหน่วยความจำในตัวแปร ptr ลงมาอีก 1 บล็อก แล้วไปตรวจสอบเงื่อนไขอีก จนกว่าเงื่อนไขจะเป็นเท็จ

สรุป

ตัวแปรอาร์เรย์ หมายถึง ตัวแปรทที่เก็บข้อมูลได้เป็นชุด โดยสร้างชื่อตัวแปรขึ้นมาเพียงตัวเดียว และต้องเป็นข้อมูลชนิดเดียวกัน

ตัวแปรพอยน์เตอร์ หมายถึง หมายถึงตัวแปรที่ใช้เก็บตำแหน่ง (address) ของตัวแปรอื่น จึงสามารถใช้ตัวแปรพอยน์เตอร์ ในการจัดการตัวแปรอาร์เรย์หรือข้อความได้อย่างมีประสิทธิภาพโดยการอ้างอิงตำแหน่งของตัวแปรแทนการอ้างโดยตรง