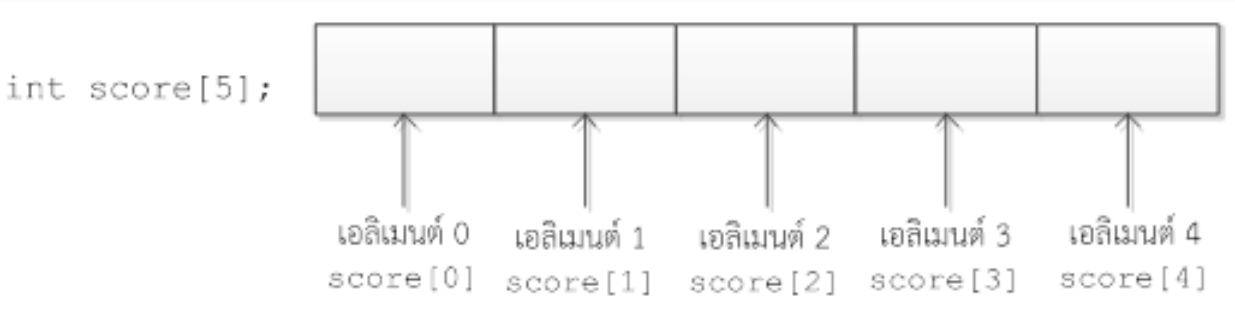
บทที่ 8
พอยท์เตอร์ (Pointer)

รายวิชา การโปรแกรมคอมพิวเตอร์

ดร.นิฏฐิตา เชิดชู

ปัญหาของการใช้งานอาเรย์

- การใช้งานตัวแปรอาเรย์ต้องรู้จำนวนสมาชิกที่แน่นอน ณ เวลาเขียนโปรแกรม
  - บ่อยครั้งเราไม่สามารถรู้ได้ว่าต้องมีสมาชิกกี่ตัว ขึ้นกับผู้ใช้งานโปรแกรม
- **<u>ตัวอย่าง</u>** เช่น

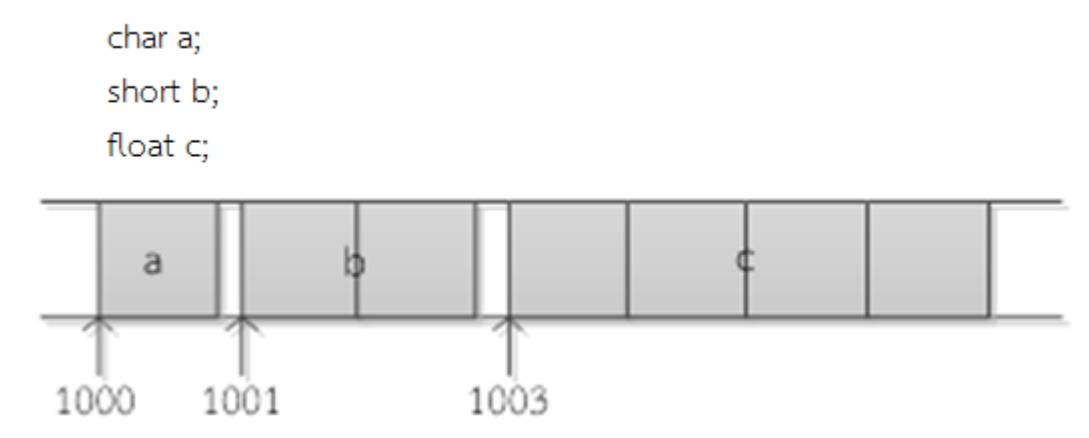


- หากผู้ใช้งานต้องการเก็บค่า score จำนวน 10 คน ?

8.1 ตัวดำเนินการตำแหน่งที่อยู่ (&) และตัวแปรชนิดพอยท์เตอร์

- การประกาศตัวแปร
  - ตัวแปร -> a, b, c
  - หน่วยความจำ

    -> 1000, 1001, 1003

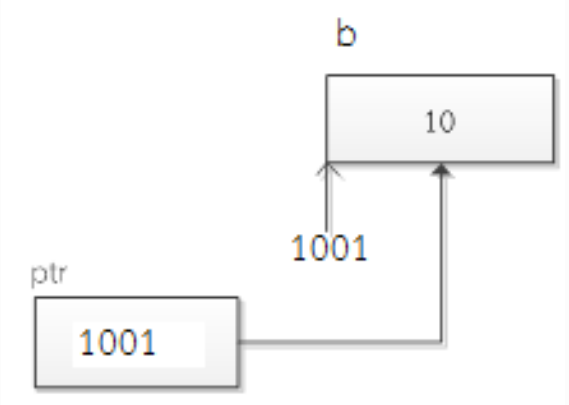


- การเข้าถึงค่า
  - ตัวแปร -> a, b, c
  - หน่วยความจำ -> ใช้เครื่องหมาย & เช่น &a -> 1000, &b, &c

ตัวแปรพอยท์เตอร์

- **ตัวแปรพอยท์เตอร์ หรือ พอยท์เตอร์** คือ ตัวแปรที่ใช้เก็บค่าตำแหน่งหน่วยความจำ
- **การประกาศใช้งาน**

  short* ptr; หรือ short *ptr;



- **การกำหนดค่า**

  ptr = &b; //เก็บค่าที่อยู่ของตัวแปร b (1001) ใน ptr

- **การเข้าถึงค่า**
  - ค่าที่เก็บไว้ในหน่วยความจำ -> *ptr -> 10
  - ตำแหน่งหน่วยความจำที่ ptr เก็บไว้ -> ptr -> 1001

8.2 ความสัมพันธ์ระหว่างอาเรย์และพอยท์เตอร์

- ชื่ออาเรย์ = ตัวแปรพอยท์เตอร์**<u>แบบคงที่</u>**

ตัวอย่างที่ 8.3

```
1  #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int numbers [] = {10, 20, 30, 40, 50};
7       cout<< "The first element in the array is: ";
8       cout<< *numbers;
9       return 0;
10  }
```

ผลลัพธ์ของการรันโปรแกรม

```
The first element in the array is: 10
```

บรรทัดที่ 8 ใช้การอ้างอิงค่าเอลิเมนต์แรกของอาเรย์ numbers ด้วย *numbers

8.2 ความสัมพันธ์ระหว่างอาเรย์และพอยท์เตอร์

- พอยท์เตอร์ = ชื่ออาเรย์

```
1  #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int number[] = {10, 20, 30, 40, 50};
8       int* ptr = number;
9       cout << ptr[1]<<endl;
10      return 0;
11 }
```

ผลลัพธ์ของการรันโปรแกรม

```
20
```

ความแตกต่างระหว่างอาเรย์และพอยท์เตอร์

ถึงแม้ว่าชื่ออาเรย์และพอยท์เตอร์จะมีความคล้ายคลึงกัน แต่ก็ยังคงมีความแตกต่างอยู่เล็กน้อย นั่นคือ ชื่ออาเรย์นั้นไม่สามารถเปลี่ยนแปลงหน่วยความจำที่ชี้ไปหา แต่พอยท์เตอร์นั้นสามารถเปลี่ยนหน่วยความจำที่ชี้ไปหาได้ เช่น

```
double *dptr;
double readings[25], totals[25];
```

ชุดคำสั่งต่อไปนี้สามารถดำเนินการได้

```
dptr = readings;
dptr = totals;
```

แต่ในทางตรงกันข้าม ชุดคำสั่งที่พยายามเปลี่ยนแปลงตำแหน่งของหน่วยความจำที่ชื่ออาเรย์ชี้ไป ไม่สามารถดำเนินการได้

```
readings = totals;
totals = dptr;
```

ดังนั้นเราอาจกล่าวว่า *ชื่ออาเรย์เป็นพอยท์เตอร์แบบคงที่* ไม่สามารถเปลี่ยนตำแหน่งหน่วยความจำที่ชี้ไปได้

8.3 การดำเนินการทางคณิตศาสตร์

- การเพิ่มค่า/ลดค่าด้วยเครื่องหมาย ++, --
- การเพิ่มค่า/ลดค่าตามตัวเลขที่กำหนด
- การลบกันระหว่างพอยท์เตอร์
  (ต้องเป็นพอยท์เตอร์บนอาเรย์เดียวกันเท่านั้น)

8.3 การดำเนินการทางคณิตศาสตร์ (ต่อ)

- การเพิ่มค่า/ลดค่าด้วยเครื่องหมาย ++, --

ตัวอย่างที่ 8.6

```
1 #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int numbers[20] = {10, 20, 30, 40, 50};
7      int* ptr1 = numbers;
8      cout<< *ptr1<<endl;
9      ptr1++;
10     cout<< *ptr1<<endl;
11     ptr1--;
12     cout<< *ptr1;
13
14     return 0;
15 }
```

ผลลัพธ์ของการรันโปรแกรม

```
10
20
10
```

8.3 การดำเนินการทางคณิตศาสตร์ (ต่อ)

- การเพิ่มค่า/ลดค่าตามตัวเลขที่กำหนด

ตัวอย่างที่ 8.4

```
1 #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int numbers [] = {10, 20, 30, 40, 50};
7      cout<< "The first element in the array is: ";
8      cout<< *numbers<<endl;
9      cout<< "The forth element in the array is: ";
10     cout<< numbers[3]<<endl;
11     cout<< "The forth element in the array is: ";
12     cout<< *(numbers + 3);
13     return 0;
14 }
```

ผลลัพธ์ของการรันโปรแกรม

```
The first element in the array is: 10
The forth element in the array is: 40
The forth element in the array is: 40
```

8.3 การดำเนินการทางคณิตศาสตร์ (ต่อ)

- การลบกันระหว่างพอยท์เตอร์
  (ต้องเป็นพอยท์เตอร์บนอาเรย์เดียวกันเท่านั้น)

ตัวอย่างที่ 8.7

```
1  #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int numbers[20];
7       int* ptr1 = numbers;
8       int* ptr2 = &numbers[10];
9       int result = ptr1 - ptr2;
10
11      cout<< result;
12      return 0;
13  }
```

ผลลัพธ์ของการรันโปรแกรม

```
-10
```

8.4 การกำหนดค่าเริ่มต้นให้กับพอยท์เตอร์

1) การกำหนดค่าพอยท์เตอร์ให้ชี้ไปที่วัตถุที่มีอยู่แล้ว

```
int myValue;
int* ptr = &myValue;
```

2) หากยังไม่รู้ว่าจะให้พอยท์เตอร์ชี้ไปที่ไหน ให้กำหนดให้ชี้ไปที่ 0 หรือ NULL ซึ่งจะเรียกว่า นัลพอยท์เตอร์

```
int* ptr = 0;   หรือ   int* ptr = NULL;
```

ตัวอย่างที่ 8.8

```
1 #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int* ptr;
7      cout<< ptr<<endl;
8
9      ptr = NULL;
10      cout<< ptr<<endl;
11
12      return 0;
13  }
```

ผลของการรันโปรแกรม

```
0x41659e
0
```

8.4 การกำหนดค่าเริ่มต้นให้กับพอยท์เตอร์ (ต่อ)

- หากลืมกำหนดค่า NULL ให้กับพอยท์เตอร์

ตัวอย่างที่ 8.9

```
1  #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int* ptr;
7       cout<< ptr<<"\t"<<*ptr<<endl;
8
9       ptr = NULL;
10      cout<< ptr<<"\t"<<*ptr<<endl;
11
12      return 0;
13  }
```

ผลลัพธ์ของการรันโปรแกรม

```
0x4165de        1528349827
```

8.5 ตัวดำเนินการเปรียบเทียบ

- >, <, ==, !=, >= และ <=
- ใช้เปรียบเทียบตำแหน่งหน่วยความจำ !!! (ไม่ใช่ค่าที่พอยท์เตอร์ชี้ไปหา)

ตัวอย่างที่ 8.10

```
1  #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int numbers[5], letters[5];
7       int* ptr1 = &letters[2];
8       int* ptr2 = &numbers[4];
9       cout<< "ptr1 = "<<ptr1<<endl;
10      cout<< "ptr2 = "<<ptr2<<endl;
11      if(ptr1 > ptr2)
12      {
13          cout<< "ptr1 is greater than ptr2."<<endl;
14      }
15      else
16          cout<< "ptr1 is not greater than ptr2."<<endl;
17
18      return 0;
19  }
```

ผลลัพธ์ของการรันโปรแกรม

```
ptr1 = 0x22fed8
ptr2 = 0x22fef4
ptr1 is not greater than ptr2.
```

การใช้งาน

- ใช้ตรวจเช็คว่าพอยท์เตอร์เป็น NULL pointer หรือเปล่า

ตัวอย่างที่ 8.11

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int score = 100;
7      int* ptr = NULL;
8      ptr = &score;
9      if(ptr != NULL)
10         cout<<"Your score is "<<*ptr<<".";
11     else
12         cout<<"You have not initialized the pointer ptr yet.";
13
14     return 0;
15 }
```

ผลลัพธ์ของการรันโปรแกรม

```
Your score is 100.
```

8.6 การผ่านค่าพารามิเตอร์ของฟังก์ชันด้วยพอยท์เตอร์

- หากต้องการ return ค่ามากกว่า 1 ค่าจากฟังก์ชัน

ตัวอย่างที่ 8.13

```
1  #include <iostream>
2   using namespace std;
3
4   void compute(int* data, int dataSize, int* maxData,
                 int* minData, float* avg)
5   {
6       *maxData = 0;
7       *minData = 1000000;
8       float sum = 0;
9       for (int loop = 0; loop < dataSize; loop++)
10      {
11          if(*(data + loop) > *maxData)
12          {
13              *maxData = *(data + loop);
14          }
15          if (*(data + loop) < *minData)
16          {
17              *minData = *(data + loop);
18          }
19          sum = sum + *(data + loop);
20      }
21      *avg = sum/dataSize;
22  }
23  int main()
24  {
25      int score[] = {10, 20, 30, 40, 50, 60};
26      int sizes = sizeof(score)/sizeof(score[0]);
27      int maximum = 0, minimum = 0;
28      float average = 0.0;
29      compute(score, sizes, &maximum, &minimum, &average);
30      cout<< "Maximum = "<<maximum<<endl;
31      cout<< "Mininum = "<<minimum<<endl;
32      cout<< "Average = "<<average;
33      return 0;}
```

8.7 การจองหน่วยความจำแบบไดนามิก

1) การจองหน่วยความจำแบบไดนามิก ให้ใช้คำสำคัญ new เพื่อบ่งบอกว่าต้องการจองพื้นที่หน่วยความจำแบบไดนามิก ดังนี้

- กรณีต้องการจองหน่วยความจำสำหรับตัวแปรธรรมดา

```
int* ptr;
ptr = new int;
```

- กรณีต้องการจองหน่วยความจำสำหรับตัวแปรแบบอาเรย์ที่มีขนาดเท่ากับ 5

```
int* ptr;
ptr = new int [5];
```

2) การจองหน่วยความจำแบบไดนามิก ด้วยคำสำคัญ new นั้นจะต้องมีคำสำคัญ delete ควบคู่ไปด้วยเสมอ โดย delete มีหน้าที่ในการคืนหน่วยความจำที่ถูกจองมาใช้แบบไดนามิก และไม่ต้องการใช้งานแล้ว สาเหตุที่ต้องมีการคืนหน่วยความจำก็เนื่องจาก จำนวนหน่วยความจำที่เครื่องคอมพิวเตอร์จัดเตรียมเอาไว้เพื่อการใช้งานแบบไดนามิกมีจำกัด หากถูกใช้งานจนหมดก็จะไม่สามารถทำการจองพื้นที่เพื่อใช้งานเพิ่มเติมได้ ส่งผลให้โปรแกรมอาจหยุดทำงานได้ การคืนหน่วยความจำสามารถทำได้ ดังนี้

- กรณีตัวแปรธรรมดา

```
delete ptr;
ptr = NULL;
```

- กรณีตัวแปรแบบอาเรย์

```
delete [] ptr;
ptr = NULL;
```

8.7 การจองหน่วยความจำแบบไดนามิก (ต่อ)

ตัวอย่างที่ 8.14

```
1 #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int studentSize = 0;
7      cout<< "How many students: ";
8      cin>> studentSize;
9      int* ptr = new int [studentSize];
10     float sum = 0.0;
11
12     // ask for the score of each student
13     for(int loop = 0; loop < studentSize; loop++)
14     {
15         cout<<endl<<"Student #"<< loop + 1<<" :";
16         cin>>*(ptr + loop);
17         sum = sum + *(ptr + loop);
18     }
19
20     // calculate the average score
21     float avg = sum/studentSize;
22     cout<<endl<<"The average score is "<< avg;
23     delete [] ptr;
24     ptr = NULL;
25     return 0;
26 }
```

ผลของการรันโปรแกรม (ตัวหนา คือ สิ่งที่ผู้ใช้โปรแกรมป้อนให้กับโปรแกรม)

```
How many students: 5
Student #1 :10
Student #2 :20
Student #3 :30
Student #4 :40
Student #5 :50
The average score is 30
```

8.8 สรุป

- พอยท์เตอร์ คือ ตัวแปรที่ใช้เก็บค่าตำแหน่งหน่วยความจำ
- การประกาศใช้งานพอยท์เตอร์ int* ptr; หรือ int *ptr;
- ชื่ออาเรย์ = พอยท์เตอร์ชนิดคงที่

  พอยท์เตอร์ = ชื่ออาเรย์
- พอยท์เตอร์สามารถดำเนินการทางคณิตศาสตร์ (เพิ่มค่า, ลดค่า, ลบกัน)
- พอยท์เตอร์สามารถดำเนินการเปรียบเทียบ (>, <, ==, !=, >= และ <=)
- Application -> Dynamic memory allocation