บทที่ 10
พอยน์เตอร์ (Pointer)

♦ address คืออะไร

ในการทำงานของคอมพิวเตอร์ จะเก็บข้อมูล(data) และโปรแกรมหรือชุดของคำสั่ง (instruction set)ไว้ในหน่วยความจำหรือที่เรียกว่า ***เมโมรี*** (memory) เพื่อนำไปประมวลผลใน CPU หรือเก็บผลการทำจาก CPU หน่วยความจำนี้มีหน่วยนับเป็น ***ไบต์ (byte)***

หน่วยความจำทุกหน่วยหรือไบต์จะมีเลขประจำตำแหน่งเพื่อใช้สำหรับอ้างถึง หมายเลขตำแหน่งของหน่วยความจำเปรียบเสมือนบ้านเลขที่ที่เราใช้อ้างถึงในการส่งจดหมาย หมายเลขตำแหน่งหน่วยความจำหน่วยหนึ่งเราเรียกว่า ***แอดเดรส (address)*** ของหน่วยความจำ โดยหมายเลขของแอดเดรสจะเริ่มตำแหน่งที่ 0, 1, 2, 3,…จนถึงค่าสูงสุดของหน่วยความจำที่ติดตั้งอยู่ในคอมพิวเตอร์ในเครื่องนั้น ๆ เช่น

ในเครื่องคอมพิวเตอร์ที่มีหน่วยความจำ 640 KB. มีค่าของแอดเดรสสูงสุด คือ 0 - 655,359

ในเครื่องคอมพิวเตอร์ที่มีหน่วยความจำ 1 MB. มีค่าของแอดเดรสสูงสุด คือ 0 -1,048,575

ในเครื่องคอมพิวเตอร์ที่มีหน่วยความจำ 8 MB. มีค่าของแอดเดรสสูงสุด คือ 0- 8,388,607

หมายเหตุ ในระบบคอมพิวเตอร์ การอ้างอิงถึง address ของหน่วยความจำ จะใช้เลขฐานสิบหก (hexadecimal) สำหรับการอ้างอิงถึงการใช้หน่วยความจำ

การกำหนดตำแหน่งหน่วยความจำเมื่อ C++ มีการประกาศใช้ตัวแปร ซึ่งจะต้องจองพื้นที่ในหน่วยความจำ C++ Compiler จะดำเนินการจองหรือกำหนดตำแหน่งแอดเดรสและจัดสรรหน่วยความจำผ่านซอฟต์แวร์ระบบปฏิบัติการ (Operating System) เช่น DOS, Windows'95 ซึ่งกระบวนการเหล่านี้เป็นหน้าที่ของระบบปฏิบัติการโดยตรง

♦ การแสดงแอดเดรส

ใน C++ เราสามารถทราบ address ของหน่วยความจำที่ตัวแปรตัวใดตัวหนึ่งที่กำลังจองหน่วยความจำหรือใช้งานอยู่ได้โดยใช้ operator คือ **& (address of - แอดเดรสของ)** กำกับหน้าชื่อตัวแปรที่เราต้องการทราบ address เช่น ประกาศตัวแปร number เป็น integer และกำหนดค่าคงที่ให้เป็น 250 ดังนี้

```
int number = 250;
```

ถ้าต้องการแสดงตำแหน่งหรือค่า address ในหน่วยความจำที่เก็บค่า 250 ซึ่งเป็นค่าของตัวแปร number เขียน statement ได้ดังนี้

```
cout<<&number;
```

โปรแกรมจะแสดงค่าตำแหน่ง address เป็นค่าเลขฐานสิบหก เช่น **0x8fa6fff4** สำหรับค่าของ address ที่ได้จากการทำงานของโปรแกรมแต่ละครั้งนี้ ไม่จำเป็นจะต้องได้ค่า address เป็นค่าเดิมทุกครั้ง นั่นคือการจองหน่วยความจำของตัวแปร address อาจเปลี่ยนแปลงตำแหน่งได้ ดังรายละเอียดโปรแกรม disp_add.cpp ต่อไปนี้

- ***ตัวอย่างโปรแกรม disp_add.cpp แสดงค่า address ของตัวแปรในโปรแกรมแต่ละตัว***

```
/*Program : disp_add.cpp
Process : disply address value of variable */

#include <iostream.h>
#include <conio.h>

void main()
{ int    number1=250;
  float  number2=1005.25;
  double number3=1254025.25212;
  char name[]="Mr.Sirichai";
  //display address by & operator
  clrscr();
  cout<< "Display address of variable..."<<endl;
  cout<< "Address of number1 = "<<&number1<<endl;
  cout<< "Address of number2 = "<<&number2<<endl;
  cout<< "Address of number3 = "<<&number3<<endl;
  cout<< "Address of name = "<<&name<<endl;
  getch();
}
```

ผลการ run โปรแกรมครั้งแรก แสดงผลลัพธ์ดังนี้

```
Display address of variable...
Address of number1 = 0x8f5afff4
Address of number2 = 0x8f5afff0
Address of number3 = 0x8f5affe8
Address of name = 0x8f5affdc
```

ผลการ run โปรแกรมครั้งที่สอง แสดงผลลัพธ์ค่า address ต่างกัน ดังนี้

```
Display address of variable...
Address of number1 = 0x8f5afff4
Address of number2 = 0x8f5afff0
Address of number3 = 0x8f5affe8
Address of name = 0x8f5affda
```

♦ ตัวแปรชนิด pointer

ตัวแปร(variable) ที่ได้ศึกษามาแล้วได้แก่ ตัวแปรชนิด int, float, double, char ตัวแปรเหล่านี้จะใช้เก็บค่าคงที่ตามชนิดที่ได้ประกาศตัวแปรไว้ เช่น int number; แสดงว่าตัวแปร number ใช้เก็บค่าคงที่ชนิด integer เท่านั้น ไม่สามารถเก็บค่าจำนวนทศนิยมได้

สำหรับตัวแปรชนิด **pointer** จะเป็นตัวแปรที่ประกาศหรือกำหนดขึ้นมาเพื่อเตรียมไว้เก็บค่า ***address*** ของตัวแปรโดยเฉพาะ ไม่สามารถเก็บค่าคงที่ประเภทอื่น ๆ ได้ วิธีการประกาศตัวแปรชนิด pointer ทำได้ดังนี้

```
ชนิดตัวแปร*  ชื่อตัวแปร;        //หรือ
ชนิดตัวแปร   *ชื่อตัวแปร, *ชื่อตัวแปร, *ชื่อตัวแปร;
```

เช่น

```
int*   add_number;      //ตัวแปร add_number เป็นตัวแปรชนิด pointer ใช้เก็บค่า
                        address ของ ตัวแปรชนิด integer เท่านั้น
float*  position;       //ตัวแปร position เป็นตัวแปรชนิด pointer
                        ใช้เก็บค่า address ของตัวแปรชนิด float เท่านั้น
char  *choice, *ptr,  *pnt;
                        //ตัวแปร choice,ptr,pnt เป็นตัวแปรชนิด pointer ใช้เก็บค่า
                        address ของตัวแปรชนิด character เท่านั้น
```

เราเรียกตัวแปรชนิดที่เป็น pointer สั้น ๆ นี้ว่า ***pointer ค่าที่นำมาเก็บใน pointer นี้ได้จะเป็นค่าของ address เท่านั้น และสามารถเก็บค่า address ของตัวแปรชนิดเดียวกับชนิดของ pointer เท่านั้น*** เช่น กำหนด pointer เป็น int* ก็สามารถนำ address ของตัวแปรประเภท int มาเก็บได้เท่านั้น

ตัวอย่าง

```
int  number = 500;                  //กำหนดตัวแปร number เป็นชนิด integer
float  total = 2500.25;             //กำหนดตัวแปร total  เป็นชนิด float
int*  int_ptr;                      //กำหนดตัวแปร pointer ชื่อ int_ptr ชนิด integer
float  float_ptr;                   //กำหนดตัวแปร pointer ชื่อ float_ptr ชนิด float
กำหนดค่าให้ pointer โดยใช้ & ดังนี้
int_ptr= &number;
float_prt = &total;
int_ptr = &total;          //ไม่ถูกต้อง เพราะ int_prt เป็น pointer ชนิด integer
float_ptr = &number;       //ไม่ถูกต้อง เพราะ float_ptr เป็น pointer  ชนิด float
```

- ***ตัวอย่างโปรแกรม pointer.cpp แสดงการเปรียบเทียบการใช้ & operator และการใช้ตัวแปรชนิด pointer แสดงค่าของ address ของตัวแปร***

```
/*Program : pointer.cpp
 Process : disply address value of variable */

 #include <iostream.h>
 #include <conio.h>

 void main()
 { int    number1=250;           //integer variable
   float  number2=1005.25;       //float variable
   double number3=1254025.25212;  //double variable
   int*    int_pointer;    //pointer variable
   float*  float_pointer;  //pointer variable
   double* doub_pointer;   //pointer variable
   char*   char_pointer;   //pointer variable
   //display address by & operator
   clrscr();
   cout<< "Display address of variable by & operator"<<endl;
   cout<< "Address of number1 = "<<&number1<<endl;
   cout<< "Address of number2 = "<<&number2<<endl;
   cout<< "Address of number3 = "<<&number3<<endl<<endl;
   // set address from memory to pointer variable
   int_pointer = &number1;
   float_pointer = &number2;
   doub_pointer = &number3;
```

```
  //display address by pointer variable
  cout<< "Display address from pointer variable "<<endl;
  cout<< "Address of number1 = "<<int_pointer<<endl;
  cout<< "Address of number2 = "<<float_pointer<<endl;
  cout<< "Address of number3 = "<<doub_pointer<<endl;
  getch();
 }
```

ผลลัพธ์ของการ run โปรแกรม แสดงให้เห็นว่า การใช้ & operator และการใช้ pointer จะให้ค่า address เหมือนกัน Display address of variable by & operator

Address of number1 = 0x8f5efff4

Address of number2 = 0x8f5efff0

Address of number3 = 0x8f5effe8

Display address from pointer variable

Address of number1 = 0x8f5efff4

Address of number2 = 0x8f5efff0

Address of number3 = 0x8f5effe8

♦ การอ้างถึงค่าคงที่ซึ่ง address ชี้อยู่ทางอ้อม ด้วย indirection operator

เมื่อเราทราบที่อยู่หรือ address ของตัวแปรใด ๆ เราสามารถที่จะใช้ค่าของ address นั้นอ้างถึงค่าคงที่ที่ตัวแปรนั้นเก็บอยู่ได้ โดยใช้เครื่องหมาย * กำกับที่หน้าตัวแปรชนิด pointer เรียกเครื่องหมาย * ที่อยู่หน้า pointer ว่า **indirection operator** และเรียกการเข้าถึงข้อมูลที่อยู่ ณ ตำแหน่ง address นั้นด้วยการใช้ * ว่า **การอ้างแอดเดรสทางอ้อม (indirect accessing or dereferencing)** เช่น

```
int   number = 500;      //ตัวแปร number เป็นชนิด integer เก็บค่าคงที่ 500 ไว้
int*  ptr;               //ตัวแปร ptr เป็น pointer ชนิดเก็บ address ของตัวแปร integer
ptr = &number;           //ptr เก็บค่า address ของตัวแปร number
cout<< number ;          //แสดงค่าตัวแปร number โดยตรงทางจอภาพ จะได้ค่า 500
cout<<*ptr;              //แสดงค่าคงที่ ณ ตำแหน่ง address ที่พอยน์เตอร์ ptr ชี้อยู่
                         ซึ่งก็คือค่า 500
```

- ***ตัวอย่างโปรแกรม pointer2.cpp แสดงค่าคงที่ของตัวแปร โดยเปรียบเทียบจากการแสดงค่าตัวแปรโดยตรง และแสดงค่าตัวแปรโดยใช้ * อ้างถึงโดยอ้อมผ่าน address ในตัวแปร pointer ซึ่งจะให้ผลลัพธ์เหมือนกัน***

```
/*Program : pointer2.cpp
  Process : disply and compare constant from variable and *pointer */

  #include <iostream.h>
  #include <conio.h>

  void main()
  { int    number1=250;            //integer variable
    float  number2=1005.25;        //float variable
    double number3=1254025.25212;  //double variable

    int*    int_pointer;    //pointer variable
    float*  float_pointer;  //pointer variable
    double* doub_pointer;   //pointer variable
    char*   char_pointer;   //pointer variable
    //display constant of variable
    clrscr();
    cout<< "Display constant from variable (direct access)"<<endl;
    cout<< "Constant of number1 = "<<number1<<endl;
    cout<< "Constant of number2 = "<<number2<<endl;
    cout<< "Constant of number3 = "<<number3<<endl<<endl;
// set address from memory to pointer variable
    int_pointer = &number1;
    float_pointer = &number2;
    doub_pointer = &number3;
    //display constant at address in pointer variable
    cout<< "Display constant from pointer by * (indirect access)"<<endl;
    cout<< "Constant of number1...*int_pointer = "<<*int_pointer<<endl;
    cout<< "Constant of number2 ...*float_pointer = "<<*float_pointer<<endl;
    cout<< "Constant of number3 ...*doub_pointer = "<<*doub_pointer<<endl;
    getch();
  }
```

♦ พอยน์เตอร์ที่ไม่กำหนดชนิดข้อมูล

ตัวแปร pointer ที่ผ่านมา จะต้องกำหนดชนิดข้อมูลให้ เช่น int* int_ptr; float flt_ptr; เป็นต้น ข้อจำกัดของการกำหนดชนิดของ pointer คือสามารถเก็บ address ได้เฉพาะของตัวแปรที่เป็นประเภทเดียวกันเท่านั้น

ดังนั้น เพื่อความยืดหยุ่นในโปรแกรม ถ้าต้องการกำหนด pointer ให้สามารถเก็บ address ของตัวแปรหลาย ๆ ชนิดก็สามารถทำได้ โดยการกำหนดให้ pointer เป็นชนิด void หรือชี้ไปที่ void เช่น

void* many_add;

//pointer ชื่อ many_add สามารถใช้เก็บ address ของตัวแปรหลายชนิดได้

- ***ตัวอย่างโปรแกรม pointer3.cpp แสดงการใช้ pointer ชนิด void ที่ชื่อ many_ptr ที่สามารถเก็บ address ของตัวแปรที่เป็นชนิดตัวเลขได้หลายชนิด ดังนี้***

```
/*Program : pointer3.cpp
 Process : disply void pointer type */
 #include <iostream.h>
 #include <conio.h>

 void main()
 { int    number1=250;            //integer variable
   float  number2=1005.25;        //float variable
   double number3=1254025.25212;  //double variable
   void*    many_ptr;    //pointer of many type of variable
   clrscr();
   //display constant at address in pointer variable
   cout<< "Display address from  'void*  many_ptr' pointer"<<endl;
   many_ptr = &number1;
   cout<< "Address of number1 integer...many_ptr = "<<many_ptr<<endl;
   many_ptr = &number2;
   cout<< "Address of number2 float...many_ptr = "<<many_ptr<<endl;
   many_ptr = &number3;
   cout<< "Address of number3 double...many_ptr = "<<many_ptr<<endl;
   getch();
 }
```

♦ ความสัมพันธ์ระหว่าง pointer กับ array

การอ้างถึงข้อมูลที่อยู่ในตัวแปรประเภท array นั้น ปกติจะใช้ **อินเด็กซ์ (index)** เป็นตัวชี้ถึงค่าคงที่หรือข้อมูลที่อยู่ใน array แต่ละช่อง ตามที่ได้ศึกษามาแล้ว เช่น

```
int number[5] = {10,20,30,40,50};
cout<<number[0];   //จะได้ค่า 10 บนจอภาพ
cout<<number[2];   //จะได้ค่า 30 บนจอภาพ
```

นอกจากการใช้ index ดังกล่าวแล้ว เรายังสามารถใช้ pointer ชี้เพื่ออ้างถึงค่าคงที่ของ array ได้โดยการใช้ **** (indirection operator)*** ซึ่งทำได้ได้ 2 วิธี คือ

1. ใช้ * นำหน้าชื่อของ array เพื่ออ้างถึงค่าใน array แต่ละช่องได้ ***ทั้งนี้เนื่องจากชื่อของ array นั้นเป็น pointer ชี้ address ของ array ช่องแรกสุดอยู่แล้ว*** เช่น

```
cout<< *(number);   // จะได้ค่า 10 เนื่องจากชื่อของ array จะเริ่มชี้ตำแหน่งช่องแรกคือ 0
cout<< *(number+1); // จะได้ค่า 20 เนื่องจากชื่อของ array จะชี้ตำแหน่งช่องที่ 0+1 คือ ช่องที่ 1 หรือสมาชิกตัวแรกของ array
cout<< *number+3; // จะได้ค่า 40 เนื่องจากชื่อของ array จะชี้ตำแหน่งช่องที่ 0+3 คือ ช่องที่ 3
```

2. ใช้ * นำหน้าชื่อของ pointer ที่ได้สร้างขึ้นและเก็บ address ของ array ช่องแรกสุดไว้ ก็สามารถอ้างถึงช่องถัดไปได้ เนื่องจากโครงสร้างของ array จะจองพื้นที่ในหน่วยความจำต่อเนื่องกันไป เช่น

```
int number[5] = {10,20,30,40,50};
int* num_add;
num_add = number;  //เก็บ address ของอาร์เรย์ ณ ตำแหน่งสมาชิกตัวแรก คือสมาชิกตัวที่ 0
                   //เขียนอีกอย่างหนึ่ง ดังนี้ num_add = &number[0];
cout<< *(num_add);   // จะได้ค่า 10 เนื่องจากชื่อของ array จะเริ่มชี้ตำแหน่งช่องแรกคือ 0
cout<< *(num_add+1); // จะได้ค่า 20 เนื่องจากชื่อของ array จะชี้ตำแหน่งช่องที่ 0+1 คือ ช่องที่ 1
cout<< *num_add+3; // จะได้ค่า 40 เนื่องจากชื่อของ array จะชี้ตำแหน่งช่องที่ 0+3 คือ ช่องที่ 3
```

- ***ดังตัวอย่างโปรแกรม pointer4.cpp แสดงการอ้างถึงค่าคงที่ในอาร์เรย์ โดยวิธีปกติคือใช้ index และการใช้ pointer โดยใช้ * นำหน้าชื่อของอาร์เรย์ และนำหน้าชื่อของตัวแปร pointer ดังนี้***

```
/*Program : pointer4.cpp
  Process : disply relation array and pointer */
  #include <iostream.h>
  #include <conio.h>
  void main()
  { int  number[5]= {10,20,30,40,50}; //integer variable
    int  i;
    int* num_ptr;
    clrscr();
    num_ptr = number; //same as write this => num_ptr = &number[0]
    //display by index
    cout<<"Display constant by index of array"<<endl;
    for(i=0;i<=4;++i)
      cout<<"number"<<i<<" = "<<number[i]<<endl;
    //display by defined  pointer of array
    cout<<endl<<"Display constant by * (indirect operator) of pointer"<<endl;
    for(i=0;i<=4;++i)
      cout<<"number"<<i<<" = "<< *(num_ptr+i) <<endl;
    //display by * (indirect operator) of array  name
    cout<<endl<<"Display constant by * (indirect operator) of array name"<<endl;
    for(i=0;i<=4;++i)
      cout<<"number"<<i<<" = "<< *(number+i) <<endl;
    getch();
  }
```

ผลลัพธ์ที่ได้จากการ run โปรแกรม จะให้ค่าคงที่ของ array ถูกต้อง เนื่องจากเป็นการอ้างถึงค่าคงที่ ณ ตำแหน่ง address เดียวกัน ดังนี้

```
Display constant by index of array
number0 = 10
number1 = 20
number2 = 30
number3 = 40
number4 = 50
```

Display constant by * (indirect operator) of pointer

number0 = 10

number1 = 20

number2 = 30

number3 = 40

number4 = 50

Display constant by * (indirect operator) of array name

number0 = 10

number1 = 20

number2 = 30

number3 = 40

number4 = 50

♦ ความสัมพันธ์ระหว่าง pointer กับ function

การส่งค่า argument ของฟังก์ชั่น (function) โดยวิธี passed argument by reference เป็นการเรียกใช้ฟังก์ชั่นโดยส่งค่า argument ไปให้ฟังก์ชั่น และมีการเปลี่ยนแปลงค่า argument ส่งคืนกลับมา วิธีการดังกล่าวก็สามารถทำได้โดยใช้การส่ง argument ที่เป็น pointer ไป เรียกว่า **passed argument pointer**

- ***ตัวอย่างโปรแกรม pointer5.cpp แสดงการส่งค่า argument ด้วยวิธี passed argument by reference ที่ได้ศึกษามาแล้ว และโปรแกรม pointer6.cpp แสดงการส่งค่า argument ด้วยวิธี passed argument pointer ซึ่งให้ผลลัพธ์เหมือนกัน***

```
/*Program : pointer5.cpp
  Process : passed argument by reference */
  #include <iostream.h>
  #include <conio.h>
  //prototype function
  void CelToFah(float& degree);

  void main()  //begin main program
  { float celsius;
    clrscr();
```

```
  cout<< "Enter Celsius degree for convert to Fahenhiet : ";
  cin>>celsius;
  //call function and passed argument by reference
  CelToFah(celsius);  //value of celsius will return and changed
  cout<< "Result  = "<<celsius<<" Fahrenhiet degree";
  getch();
 }  //end main program

void CelToFah(float& degree)  //function convert Celsius to Fahrenhiet
 {
  degree = degree*9/5+32;
 }
```

- ***ตัวอย่างโปรแกรม pointer6.cpp แสดงการผ่านค่าให้แก่ฟังก์ชันด้วย pointer เมื่อมีการเรียกใช้ฟังก์ชัน CelToFah จะต้องส่งค่า argument มาให้ฟังก์ชันในลักษณะของ reference คือส่งค่า celsius มาให้ degree ซึ่งเป็น parameter ของฟังก์ชัน จากนั้นนำมาแปลงค่าแล้วส่งกลับไปยังชื่อของ celsius ซึ่งเป็น argument ทำให้ค่าของ celsius เปลี่ยนไป***

```
/*Program : pointer6.cpp
  Process : passed argument by pointer */
  #include <iostream.h>
  #include <conio.h>
  //prototype function
  void CelToFah(float* degree);

  void main()
  { float celsius;
   clrscr();
   cout<< "Enter Celsius degree for convert to Fahenhiet : ";
   cin>>celsius;
   //call function and passed argument by address of celsius
   CelToFah(&celsius);  //value of celsius will return and changed
   cout<< "Result  = "<<celsius<<" Fahrenhiet degree";
   getch();
  }
void CelToFah(float* degree)  //function convert Celsius to Fahrenhiet
 {  //* operater refer to value at address in degree pointer
   *degree = (*degree)*9/5+32;
 }
```

- **ตัวอย่างโปรแกรม *Ptr_Arr4.cpp* แสดงการใช้ pointer ในการเขียนที่เป็นฟังก์ชันแบบมีการใช้อากิวเมนต์ที่เป็น pointer**

```
/*Proram: Ptr_Arr4.cpp
  Process: Uses pointer in parameter function  manage array*/
#include <iostream.h>
#include <conio.h>

void Input();
void Display(int *point);
void Sum_avg(int *ptr_number);
int number[5];
void main()
{ clrscr();
  Input();
  Display(&number[0]);
  getch();
  Sum_avg(&number[0]);
}
void Input()
{ int i, *ptr;
  ptr=&number[0];
  cout<<"Enter number 5 item to array by pointer: "<<endl;
  for (i=0;i<=4;i++)
  cin>> *(ptr+i);
}
void Display(int *point)
{ int i;
  clrscr();
  cout<<"Display number 5 item from array by pointer: "<<endl;
  for (i=0;i<=4;i++)
     cout<< *(point+i)<<endl;
}
void Sum_avg(int *ptr_number)
{ float sum, average, *ptr_avg;
  sum=average=0;
  ptr_avg=&average;
  for (int i=0;i<=4;i++)
     sum = sum+ *(ptr_number+i);
  *ptr_avg =sum/5;
  cout<<"Summation of array 5 item: "<<sum<<endl;
  cout<<"Average of array 5 item: "<< *ptr_avg;
  getch();  }
```

♦ ความสัมพันธ์ระหว่าง pointer กับ string

เนื่องจากใน C++ ข้อมูลประเภท string ก็คือ อาร์เรย์ 1 มิติ ชนิดที่เป็น character ดังนั้นเราจึงสามารถใช้ pointer ชี้ตำแหน่งแต่ละสมาชิกของ string ได้

- ***ตัวอย่างโปรแกรม str_ptr1.cpp แสดงการอ้างถึงข้อมูลในสตริง โดยใช้ pointer โดยตรง การใช้ตัวแปร pointer และการอ้างโดยใช้ชื่อสตริงตามปกติ***

```
Program : str_ptr1.cpp
Process : display string by name of array and pointer */
#include <iostream.h>
#include <conio.h>

void main()
{  char text[] = "Mr.Sirichai Namburi";
   char* ptr_str = "Computer Department , RIPA";
   char* ptr_var;
   //comparative display constant from string
   clrscr();
   cout<< "Display constant of string by..."<<endl<<endl;
   cout<< " Name of teacher : "<<text<<endl; //display by array
   cout<< "Office : "<<ptr_str<<endl;  //display by pointer
   ptr_var=text;  //use pointer variable store address of text[0]
   cout<< "Name of teacher(*ptr_var)= "<<*ptr_var<<endl;//refer only address of  text[0]
   cout<< "Name of teacher(ptr_var)= "<<ptr_var<<endl; //point at text[0]
   cout<< "Name of teacher(++ptr_var)= "<<++ptr_var<<endl; //point at text[1]
   cout<< "Name of teacher(++ptr_var)= "<<++ptr_var<<endl; //point at text[2]
   cout<< "Name of teacher(++ptr_var)= "<<++ptr_var<<endl; //point at text[3]
   cout<< "Name of teacher(++ptr_var)= "<<++ptr_var<<endl; //point at text4]
   getch();
}
```

- ***ตัวอย่างโปรแกรม str_ptr2.cpp แสดงการใช้ array 2 มิติ และใช้ pointer ของสตริงในการอ้างถึงสมาชิกแต่ละรายการ เพื่อแสดงรายการเมนูให้เลือก***

```
/*Program : str_ptr2.cpp
  Process : display 2 dimension array and use pointer */
  #include <iostream.h>
  #include <conio.h>
```

```
void main()
{  const int no = 4;
   const char*  MenuItem[no] =  { "0. Open File",
                                  "1. Write File",
                                 "2. Read File",
                                  "3. Exit"};
   //begin statement
   clrscr();
   int choice,row=4;
   do
    { gotoxy(30,2);cout<< "Main Menu";
     for(int i=0;i<=3;++i)
       {gotoxy(30,row);  cout<<MenuItem[i];  //display menu item by pointer
        row++;
       }
     gotoxy(30,row);cout<<"Select your choice <0-3> : ";  cin>>choice;
     if((choice<0)||(choice>3))
       { cout<<endl<< "Error !! choice \a";
         getch();clrscr();row=4;
       }
    }while((choice<0)||(choice>3));
    cout<<endl<<endl<<"Your choice is : " <<choice<<endl;
    getch();
}
```

- ***ตัวอย่างโปรแกรม prt_str.cpp แสดงการใช้ pointer เพื่อจัดการข้อมูลประเภท string โดยกำหนดค่าคงที่โดยใช้ตัวแปรชนิด pointer ที่เป็นอาเรย์ 1 มิติ***

```
/*Program: ptr_str.cpp*/
#include <iostream.h>
#include <conio.h>

const num=4;
//declaration pointer and set value
char* size[num]={"Big","Middel","Small","Little"};

void main()
{ clrscr();
  //display first character of array by pointer
  for(int i=0;i<=3;i++)
    cout<< *size[i];
  cout<<endl<<endl;
```

```
 //display string by first to last pointer
 for(i=0;i<=3;i++)
   cout<<size[i]<<" ";
 cout<<endl<<endl;
 //display string by last to first pointer
 for(i=3;i>=0;i--)
   cout<<size[i]<<" ";
getch();
}
```

ผลการทำงานของโปรแกรม เป็นดังนี้

BMSL

Big Middel Small Little

Little Small Middel Big

♦ Pointer กับข้อมูลชนิด Structure

เราสามารถใช้ pointer กับข้อมูลชนิดโครงสร้างได้ โดยใช้ operator เครื่องหมาย -> ตามหลัง pointer แล้วตามด้วยชื่อ field ใน structure เพื่ออ้างอิงถึงข้อมูลในแต่ละ field ได้ ดังตัวอย่าง

```
struct  STUDENT        //สร้าง structure ชื่อ STUDENT
{       char  code[8];
        char  name[25];
        int     midterm;
        int     final;
        char  grade;
} STD;                 //ประกาศตัวแปร STD เป็นชนิด structure ที่ชื่อ  STUDENT
STUDENT   *ptr;        //ประกาศตัวแปร  pointer ชื่อ  ptr  เป็นชนิดเก็บ address ของ
                       //structure
ptr = &STD;            //ให้ pointer ptr  เก็บ address ของตัวแปร  STD ซึ่งเป็นชนิด
                       //structure
```

การอ้างอิงถึงข้อมูลใน structure โดยใช้ pointer มีรูปแบบ คือ **pointer->field_name** ดังตัวอย่าง

```
strcpy(ptr->code, "1000105");
strcpy(ptr->name,"Sirichai Namburi");
ptr->midterm=30;
ptr->final=50;
ptr->grade='A';
cin>>ptr->code; //รับข้อมูลทางแป้นพิมพ์
cin>>ptr->midterm;      //รับข้อมูลทางแป้นพิมพ์
cout<<ptr->code;        //แสดงข้อมูล
cout<<ptr->midterm;     //แสดงข้อมูล
```

- ***ตัวอย่างโปรแกรม ptr_stru.cpp แสดงการใช้เครื่องหมาย -> เพื่อใช้ pointer อ้างอิงถึงข้อมูลประเภท structure ทั้งการกำหนดค่าคงที่ด้วยเครื่องหมาย = และการรับค่าคงที่ทางแป้นพิมพ์ผ่าน cin และแสดงผลข้อมูลใน structure โดยเปรียบเทียบกับการใช้ . ในการอ้างอิงถึงฟิลด์ข้อมูลในโครงสร้างตามที่เคยได้ศึกษามา***

```
/*Program: ptr_stru.cpp
  Process: display to use pointer refered to structure
           for input and display data in structure
*/

#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
void set_display();
void Input();
void Display1();
void Display2();
//daclare structure
struct mystruct
 {
  int number;
  char name[21];
  double age;
} s;  //declare variable

mystruct *sptr = &s; //declare pointer and set pointer
void main()
```

```
{ clrscr();
  set_display();
  Input();
  Display1();
  Display2();getch();
}

void set_display()
{
  sptr->number=300;
  strcpy(sptr->name,"Sirichai");
  sptr->age=250.125;
  //display by pointer
  cout<<"Display data from structure by pointer  -> \a"<<endl;
  cout<<sptr->number<<endl;
  cout<<sptr->name<<endl;
  cout<<sptr->age<<endl;
}
void Input()
{ cout<<endl<<"Input data to structure by pointer  3  Item: \a"<<endl;
  cout<<"Code: ";cin>>sptr->number;
  cout<<"Name: ";cin>>sptr->name;
  cout<<"Age: ";cin>>sptr->age;
}

void Display1()
{ cout<<endl<<"Display data from structure by use pointer ->\a"<<endl;
  cout<<sptr->number<<endl;
  cout<<sptr->name<<endl;
  cout<<sptr->age<<endl;
}

void Display2()
{ cout<<endl<<"Display  data from structure by use  .  reference..."<<endl;
  cout<<s.number<<endl;
  cout<<s.name<<endl;
  cout<<s.age<<endl;
}
```

- ***ตัวอย่างโปรแกรม pt_stra.cpp แสดงการใช้ array จัดเก็บ pointer เพื่ออ้างอิงถึงข้อมูลใน structure***

```
/*Program: pt_stra.cpp
  Process: display to use pointer refered to array of structure
           for input and display data in array of structure
*/
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
//declaration prototype functions  in  program
void Input();
void Display();

//daclare structure
struct mystruct
 {
   int number;
   char name[21];
   double age;
};
mystruct s[3];  //declare variable array of structure 3 elements
mystruct *sptr[3]; //declare array of pointer  3  elements

void main()
{ clrscr();
  Input();
  Display();
  getch();
}

void Input()  // input data to array by pointer
{ cout<<endl<<"Input data to structure by pointer  3  Item: \a"<<endl;
  for(int i=0;i<=2;i++)
  { sptr[i]=&s[i];  //pointer point at each array of structure
    cout<<"Code: ";cin>>sptr[i]->number;
```

```
   cout<<"Name: ";cin>>sptr[i]->name;
   cout<<"Age: ";cin>>sptr[i]->age;
   cout<<endl;
  }
}
```

void Display() //display all data from array by using pointer sptr[i]

```
{ cout<<endl<<"Display data from structure by use pointer ->\a"<<endl;
  for(int i=0;i<=2;i++)
  { cout<<sptr[i]->number<<endl; //use pointer
    cout<<sptr[i]->name<<endl;
    cout<<sptr[i]->age<<endl<<endl;
  }
}
```

♦ แบบฝึกหัดท้ายบท

1. กำหนดตัวแปรดังนี้

   A=250 B=1250.125 C='p' Str[]="Computer"

   จงสร้าง pointer เพื่อแสดง address ของตัวแปรต่าง ๆ พร้อมทั้งแสดงผลค่าคงที่ของตัวแปรเหล่านี้โดยใช้ pointer

2. จงเขียนโปรแกรมโดยกำหนดตัวแปรประเภทอาร์เรย์ 1 มิติ number[10] นำค่า 10, 15, 20, 25, 30, 35, 40, 45,50,55 เข้าไปเก็บในตัวแปรอาร์เรย์นี้ และให้กำหนดตัวแปรอาร์เรย์ชนิดที่เป็นพอยน์เตอร์ ชื่อ num_ptr[10] เพื่อเก็บตำแหน่ง address ของตัวแปรอาเร์เรย์ number[10] แต่ละสมาชิก ให้แสดงการตำแหน่งของ address ของแต่ละสมาชิกใน number[10] และแสดงค่าคงที่ที่เก็บไว้ใน number[10] โดยใช้ตัวแปร num_ptr ซึ่งเป็นพอยน์เตอร์แสดงค่าคงที่ของตัวแปรบนจอภาพ โดยให้โปรแกรมแสดงผล ดังนี้

num_ptr[i]	*num_ptr[i]
address 0x8f8effe2	Value 5
address 0x8f8effe4	Value 10
address 0x8f8effe6	Value 15
address 0x8f8effe8	Value 20
address 0x8f8effea	Value 25
address 0x8f8effec	Value 30

address 0x8f8effee Value 35

address 0x8f8efff0 Value 40

address 0x8f8efff2 Value 45

address 0x8f8efff4 Value 50

3. ให้นักศึกษาเขียนโปรแกรมเก็บรายละเอียดสินค้า โดยมีรายละเอียดที่ต้องจัดเก็บ คือ รหัสสินค้า, ชื่อสินค้า, ราคาต่อหน่วย, ส่วนลดเงินสด(%), หมายเลขโกดังที่เก็บสินค้า, จำนวนหน่วยสินค้าคงเหลือ โดยใช้ pointer อ้างอิงถึงข้อมูลสินค้า โดยสร้างฟังก์ชันที่ทำหน้าที่

- รับข้อมูล (ยกเลิกการกรอกข้อมูลเมื่อกรอกรหัสสินค้าเป็น 0)
- แสดงผลรายงานข้อมูลทั้งหมด
- ค้นหาสินค้าด้วยสหัสสินค้า

โดยโปรแกรมสามารถจัดเก็บประวัติสินค้าได้ไม่เกิน 100 ชนิด