# ตำแหน่งหน่วยความจำ

ตำแหน่งหน่วยความจำ หรือเรียกว่า พอยน์เตอร์ (pointer) ก็คือ ตัวแปรชนิดหนึ่งที่ใช้เก็บค่าตำแหน่ง หรือแอดเดรสของข้อมูลที่อยู่ในหน่วยความจำ (ศรัณย์ อินทโกสุม, 2539) ซึ่งสามารถนำพอยน์เตอร์มาใช้ในการรับส่งข้อมูลระหว่างฟังก์ชันกับตำแหน่งที่เรียกใช้ฟังก์ชัน หรือส่งค่าผ่านทางอาร์กิวเมนต์ของฟังก์ชันแต่ละฟังก์ชันได้ การจัดการหน่วยความจำในการเขียนโปรแกรมคอมพิวเตอร์ เมื่อมีการประกาศ (จอง) ตัวแปร (variable) ที่ใช้ในการกำหนดค่าเริ่มต้นของตัวแปร ระบบหน่วยความจำจะต้องทำการกำหนดพื้นที่ของตำแหน่ง (address) ที่ใช้ในการเก็บค่าของตัวแปรที่แน่นอน โดยตัวชื่อของตัวแปรเป็นเพียงชื่อที่ใช้ในการอ้างอิงของผู้เขียนโปรแกรมเท่านั้น การจัดการหน่วยความจำทั้งด้านขนาดของหน่วยความจำ และช่วง (rang) พื้นที่ในการใช้เก็บหน่วยความจำถาวร (ROM) กับหน่วยความจำชั่วคราว (RAM) ที่ใช้ในการเก็บข้อมูลของโปรแกรมใช้งาน จะถูกกำหนดโดยโปรแกรมระบบปฏิบัติการ (operating system)

ข้อมูลที่เก็บในหน่วยความจำ ของการเขียนโปรแกรมคอมพิวเตอร์ จะประกอบตำแหน่งที่อยู่ของตัวแปร กับค่าของตัวแปร ซึ่งขนาดของตัวแปรที่ใช้ในการเก็บค่า ขึ้นอยู่กับชนิดของตัวแปร (Steven และ Lutfar, 2006) (Brian W. K., Online) ในภาษาซีกำหนดให้ & เป็นตัวดำเนินการแสดงตำแหน่งหน่วยความจำ (แอดเดรส: address) ถ้าต้องการให้ px เป็นตัวแปรที่ใช้ในการเก็บค่าแอดเดรสของตัวแปร x สามารถเขียนได้เป็น px = &x ดังนั้นการประกาศตัวแปร px ต้องเป็นตัวแปรพอยน์เตอร์ รูปที่ 7.1


รูปที่ 7.1 แสดงความสัมพันธ์ของค่าตำแหน่งของ x เก็บไว้ที่ px

## การประกาศชนิดของพอยน์เตอร์

การเขียนแปรแกรมก่อนใช้ตัวแปรใดๆ ต้องทำการประกาศตัวแปรนั้นๆ ก่อนเรียกใช้งานเสมอ เช่นเดียวกับ พอยน์เตอร์ที่จะทำหน้าที่ในการเก็บที่อยู่ของตัวแปร ต้องมีการประกาศพอยน์เตอร์เช่นเดียวกัน (ชื่อพอยน์เตอร์ต้องขึ้นต้นด้วยเครื่องหมาย *)

**<u>รูปแบบ</u>**

ชนิดของตัวแปร *ชื่อของพอยน์เตอร์

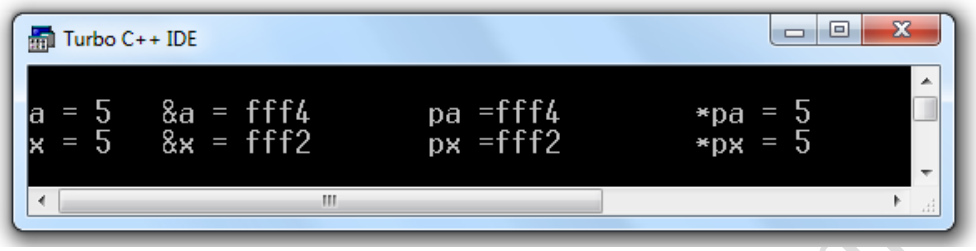
**ตัวอย่างเช่น**

| | |
|---|---|
| int *px, *ptr; | เป็นการประกาศพอยน์เตอร์ตัวแปรชื่อ px และพอยน์เตอร์ชื่อ ptr เป็นพอยน์เตอร์ชนิด integer |
| char *txt; | เป็นการประกาศพอยน์เตอร์ชื่อ txt เป็นพอยน์เตอร์ชนิด character |
| float a; | ประกาศตัวแปรชื่อ a เป็นตัวแปรชนิด float |
| float *ph = &a | ประกาศพอยน์เตอร์ชนิดเลขทศนิยม ph เก็บค่าแอดเดรสของ a |

ตัวอย่างโปรแกรมที่ 7.1 แสดงตำแหน่งหรือแอดเดรสพอยน์เตอร์เบื้องต้น

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int a=5,x;
        int *pa, *px;

        pa = &a;
        x = *pa;
        px = &
        printf("\na = %d &a = %x pa =%x *pa = %d",a,&a,pa,*pa);
        printf("\nx = %d &x = %x px =%x *px = %d",x,&x,px,*px);
        getch();
}
```

ผลการทำงานโปรแกรม




จากผลการทำงานโปรแกรม

ตัวแปร a ถูกกำหนดค่าเริ่มต้นขณะทำการประกาศตัวแปรให้มีค่าเท่ากับ 5 ส่วนตัวแปร x ไม่ได้กำหนดค่าเริ่มต้น

pa ประกาศให้เป็นพอยน์เตอร์ ใช้ในเก็บที่อยู่ของตัวแปร a เช่นเดียวกับ px เป็นพอยน์เตอร์เก็บค่าตำแหน่งของ x

pa ถูกให้เก็บค่าที่อยู่ของ a (pa = &a) คือ fff4 เป็นค่าเลขฐานสิบหก ซึ่งโปรแกรมระบบปฏิบัติการเป็นผู้กำหนดตำแหน่งแอดเดรสขึ้นมาเอง

x = *pa เป็นการกำหนดให้นำค่าข้อมูลที่ตำแหน่งของ pa ชี้อยู่ (ซึ่งก็คือค่าข้อมูลของตัวแปร a นั่นเอง) ดังนั้น x จึงมีค่าเท่ากับ 5 ด้วย

ส่วนค่าตำแหน่งที่ px ชี้อยู่เป็น fff2 (ในการทดลองของผู้อ่านอาจเป็นแอดเดรสอื่นก็เป็นไปได้ เพราะการจัดการหน่วยความจำของแต่ละเครื่องคอมพิวเตอร์อาจมีการจัดแบ่งช่วง ที่ใช้ในการจัดเก็บพื้นที่ของโปรแกรมคนละตำแหน่งกัน)

ตัวอย่างโปรแกรมที่ 7.2 แสดงการนำค่าข้อมูลที่พอยน์เตอร์ชี้อยู่มาทำการคำนวณได้

```
#include<stdio.h>
#include<conio.h>
void main()
  {
    clrscr();
    int a=5,x1,x2;
    int *pa;

    x1 = (a+10)*5;

    pa = &a;
    x2 = (*pa +10)*5;

    printf("\nx1 = %d    x2 = %d",x1,x2);
    getch();
}
```

ผลการทำงานโปรแกรม


จากผลการทำงานโปรแกรม มีนิพจน์ทางคณิตศาสตร์ x1 = (a+10)*5 ค่าของตัวแปร x1 เท่ากับ 75 ส่วน pa = &a เป็นการกำหนดให้พอยน์เตอร์ pa เก็บค่าที่อยู่ของตัวแปร a ดังนั้นนิพจน์ x2 = (*pa +10)*5 มีค่าเท่ากับ 75 เพราะ *pa เป็นมีค่าเท่ากับตัวแปร a คือ 5

## การส่งพอยน์เตอร์ผ่านทางฟังก์ชัน

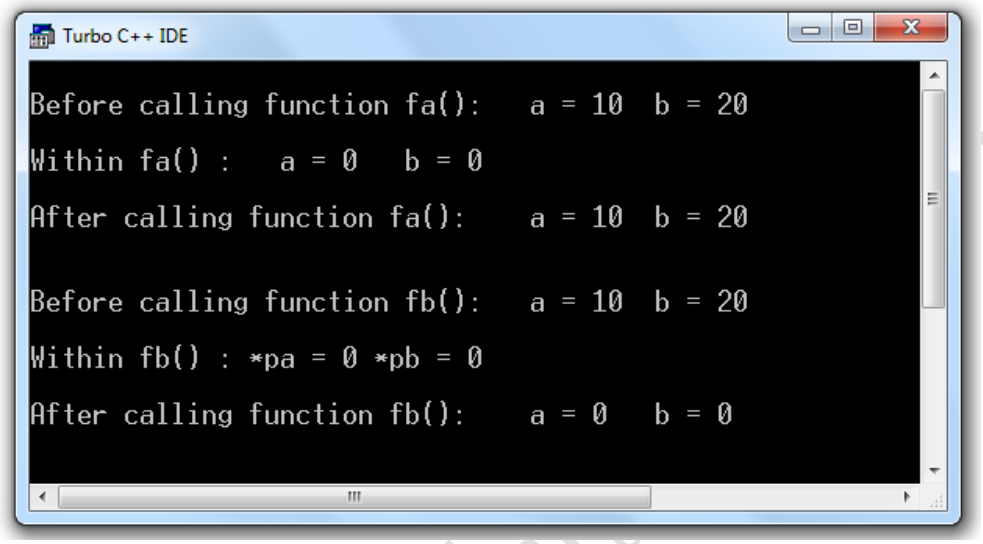
ในการส่งผ่านข้อมูลระหว่างฟังก์ชันหลัก (main()) กับ ฟังก์ชันย่อย (FunctionName()) โดยใช้พอยน์เตอร์ ลงในอาร์กิวเมนต์ให้กับฟังก์ชัน เป็นการส่งค่าข้อมูลที่พอยน์เตอร์ชี้อยู่ มีผลทำให้ค่าที่ถูกส่งผ่านทางฟังก์ชันสามารถปรับปรุง จากตัวแปรหรือตัวฟังก์ชันที่อยู่คนละฟังก์ชันได้ทันทีที่มีการเปลี่ยนแปลงค่าข้อมูล ดังตัวอย่างโปรแกรมที่ 7.3
ตัวอย่างโปรแกรมที่ 7.3 แสดงการส่งค่าข้อมูลระหว่างฟังก์ชันหลักกับฟังก์ชันย่อย โดยทำการเปรียบเทียบค่าการส่งค่าข้อมูลด้วย ตัวแปรทั่วไปกับการส่งค่าข้อมูลด้วยพอยน์เตอร์ผ่านทางฟังก์ชัน

```
#include<stdio.h>
#include<conio.h>

void main()
  {
    clrscr();
    int a=10,b=20;
    void fa(int a, int b);
    void fb(int *pa, int *pb);

    printf("\nBefore calling function fa():          a = %d b = %d",a,b);
    fa(a,b);
    printf("\n\nAfter calling function fa():          a = %d b = %d",a,b);

    printf("\n\n\nBefore calling function fb():     a = %d b = %d",a,b);
    fb(&a,&b);
    printf("\n\nAfter calling function fb():          a = %d b = %d",a,b);
    getch();
  }
```

ต่อจากด้านบน

```
void fa(int a, int b)
  {
  a=0,b=0;
  printf("\n\nWithin fa() :        a = %d            b = %d ",a,b);
  }

void fb(int *pa, int *pb)
  {
  *pa=0,*pb=0;
  printf("\n\nWithin fb() : *pa = %d *pb = %d ",*pa, *pb);
```

ผลการทำงานโปรแกรม




จากผลการทำงานโปรแกรมพบว่า เมื่อมีการกำหนดค่าให้กับ พอยน์เตอร์ที่ชี้ของตัวแปร a (*pa = 0) และพอยน์เตอร์ที่ชี้ตัวแปร b (*pb = 0 ) ในฟังก์ชัน fb() มีผลทำให้ค่าของตัวแปร a ถูกเปลี่ยนค่าเท่ากับ 0 (ศูนย์) และตัวแปร b ถูกเปลี่ยนค่าเท่ากับ 0 เช่นกันด้วย ดังผลการแสดงผลหลักจากเรียกฟังก์ชัน fb() ผลของแสดงค่าในตัวแปร a และตัวแปร b เท่ากับศูนย์ ดังแสดงข้อความ

After calling function fb(): a = 0 b = 0

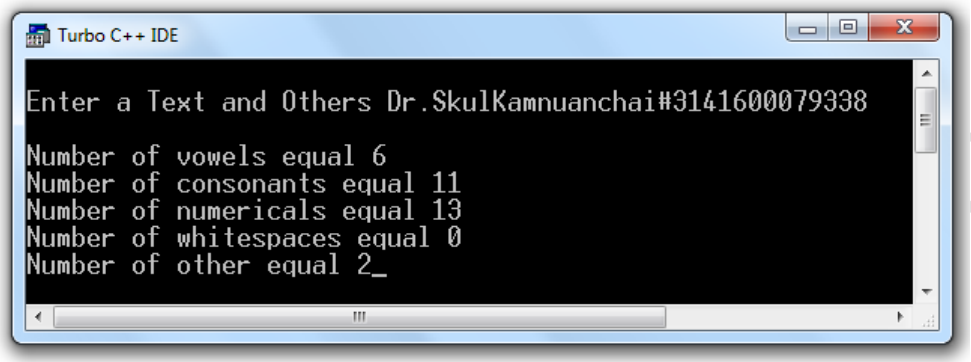
ตัวอย่างโปรแกรมที่ 7.4 แสดงการส่งค่าข้อมูลผ่านทางพอยน์เตอร์ระหว่างฟังก์ชันย่อยมายังฟังก์ชันหลัก เป็นตัวอย่างโปรแกรมที่ทำการวิเคราะห์ตัวอักขระ (ศรัณย์ อินทโกสุม, ทฤษฎีและตัวอย่างโจทย์การเขียนโปรแกรมด้วยภาษาซี, 2539) สระ พยัญชนะ ตัวเลข อักขระไวท์สเปซ (ช่องว่าง) และอักขระอื่นๆ คำนวณหาว่ามีจำนวนเท่าใด

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void main()
  {    clrscr();
    char txt[80];
    int vow = 0,con = 0, num=0, spc=0, oth=0;
    int *pv, *pc, *pn, *ps, *po;
    void scan_txt(char txt[], int *pv, int *pc, int *pn, int *ps, int *po);
    printf("\nEnter a Text and Others");
    scanf("%[^\n]",txt);
    scan_txt(txt, &vow, &con, &num, &spc, &oth);
    printf("\nNumber of vowels equal %d",vow);
    printf("\nNumber of consonants equal %d",con);
    printf("\nNumber of numericals equal %d",num);
    printf("\nNumber of whitespaces equal %d",spc);
    printf("\nNumber of other equal %d",oth);
    getch();
  }
```

ต่อจากด้านบน

```
void scan_txt(char txt[], int *pv, int *pc, int *pn, int *ps, int *po)
  {
  char x;
  int count=0;
  while((x=toupper(txt[count]))!='\0')
  {
  if(x=='A'||x=='E'||x=='I'||x=='O'||x=='U')
          ++*pv;
  else if(x>='A'&&x<='Z')
          ++*pc;
          else if(x>='0'&&x<='9')
                  ++*pn;
              else if(x==' '&&x=='\t')
                  ++*ps;
                  else
                     ++*po;
  ++count;
  }
  return;
}
```

ผลการทำงานโปรแกรม



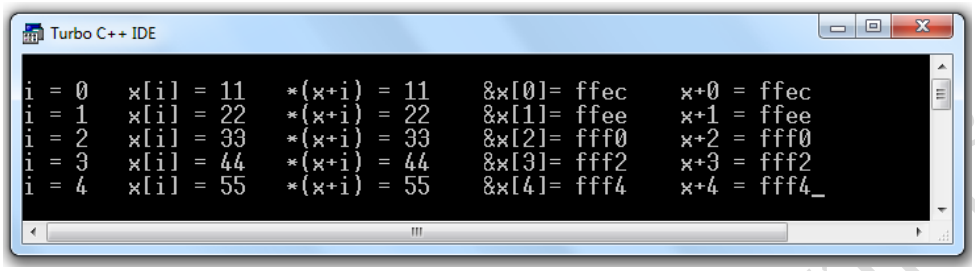

จากผลการทำงานโปรแกรม โดยการป้อนตัวอักขระที่เป็นตัวอักษรและตัวพิเศษอื่นๆ เช่น Dr.SkulKamnuanchai#3141600079338 และ Enter ตัวโปรแกรมจะทำการนำตัวอักขระทั้งหมด ส่งไปยังฟังก์ชัน scan_txt(txt, &vow, &con, &num, &spc, &oth) โดยส่งค่าอาร์กิวเมนต์ ที่เป็นข้อความทั้งหมดด้วย txt ส่วนค่าที่อยู่ของตัวแปรส่งด้วย &vow, &con, &num, &spc, &oth มาทดสอบเงื่อนไขตามคำสั่ง if_else ทำการนับจำนวนของสระ พยัญชนะ ตัวเลข อักขระไวท์สเปซ (ช่องว่าง) และอักขระอื่นๆ โดยทำการเพิ่มค่าจำนวนตามที่เป็นไปตามเงื่อนไขในส่วนของพอยน์เตอร์ของแต่ละส่วนประกอบคำ ++*pv, ++*pc, ++*pn, ++*ps และ ++*po ตามลำดับ ซึ่งมีผลต่อการเพิ่มค่าในตัวแปรที่พอยน์เตอร์แต่ละตัวเป็นตัวชี้ตำแหน่งด้วย เมื่อทำการส่งค่ากลับฟังก์ชันหลัก (void main()) ทำให้ตัวโปรแกรมหลักแสดงการนับจำนวนสระได้จำนวน 6 สระ (u, a, u, a, a ,i) ตัวอักษรจำนวน 11 ตัว (D, r, S, k, l, K, m, n, n, c, h) ตัวเลขจำนวน 13 ตัว (3, 1, 4, 1, 6, 0, 0, 0, 7, 9, 3, 3, 8) ตัวไวท์สเปซ 0 ตัว และตัวอื่นๆ จำนวน 2 ตัว (., #)

## การใช้พอยน์เตอร์กับอาร์เรย์

พอยน์เตอร์ มีความหมายเป็นตัวชี้ตำแหน่งของตัวแปร อาร์เรย์หนึ่งมิติมีการอ้างถึงตำแหน่งของตัวแปรด้วยค่าดัชนี (ค่าที่อยู่ในเครื่องหมาย [ ]) การเขียนอ้างตำแหน่งของอาร์เรย์สมาชิกแรกด้วยค่าดัชนีแรกสามารถเขียนได้เป็น &x[0] หรือเขียนได้อีกรูปแบบหนึ่ง x ถ้า x คืออาร์เรย์ขนาดหนึ่งมิติ ในการอ้างถึงสมาชิกตำแหน่งถัดไปสามารถได้ 2 รูปแบบเช่นกันคือ รูปแบบแรก &x[1] หรือรูปแบบที่สอง (x+1) ส่วนในการอ้างถึงตำแหน่งถัดไปสามารถเขียนในรูปแบบของการวนรอบ ได้ด้วยรูปแบบของคำสั่งวนรอบมาประยุกต์ใช้ในการอ้างถึงตำแหน่ง สามารถเขียนได้ดังนี้ (x+i) โดย i เป็นเลขจำนวนเต็มที่ใช้ในการเพิ่มค่าของสมาชิก x ดังนั้น i ทำหน้าที่ในการระบุค่าตำแหน่งสมาชิกตัวที่ i ของอาร์เรย์หนึ่งมิติ ดังตัวอย่างโปรแกรมที่ 7.5

```
#include<stdio.h>
#include<conio.h>
void main()
  {    clrscr();
   int x[5] = {11, 22, 33, 44, 55};
   int i;
   for (i=0;i<5;i++)
   printf("\ni = %d          x[i] = %d   *(x+i) = %d    &x[%d]= %x    x+%d = %x"
   ,i,x[i],*(x+i),i,&x[i],i,x+i);
   getch();
  }
```

ผลการทำงานโปรแกรม




จากผลการทำงานโปรแกรมของการแสดงค่าของการทำงานพอยน์เตอร์กับอาร์เรย์ เริ่มต้นการกำหนดค่าให้กับอาร์เรย์ x[5] มีสมาชิกจำนวน 5 ค่า โดยเริ่มที่สมาชิกตัวแรก x[0]=11 สมาชิกตัวถัดไป x[1] = 22 ตัวถัดไป x[2] = 33 ตัวถัดไป x[3] = 44 และตัวท้ายสุด x[4] = 55 การอ้างถึงตำแหน่งของสมาชิกจึงสามารถทำได้ด้วยการหมุน ค่า i ด้วยคำสั่ง ด้วยคำสั่ง for (i=0;i<5;i++) ในการแสดงค่าต่างๆ ของอาร์เรย์ x ได้ด้วย printf() ด้วยคำสั่งหลีกกับค่าของตัวแปรด้วยคำสั่ง (“\ni = %d”,I) การแสดงค่าของสมาชิกอาร์เรย์ x ด้วยคำสั่ง (“x[i] = %d”, x[i]”) แสดงค่าข้อมูลที่อาร์เรย์ x บวกกับ I มาแสดงค่าด้วยคำสั่ง (“*(x+i) = %d” ,*(x+i)) แสดงค่าตำแหน่งที่อาร์เรย์แต่ละสมาชิกยืนอยู่ด้วย (“&x[%d]= %x” ,i,&x[i]) และการแสดงค่าด้วยการอ้างถึงตำแหน่งด้วยรูปแบบของอาร์เรย์ทั่วไปด้วย (“x+%d = %x" ,I ,x+i)

ในการอ้างตำแหน่งพอยน์เตอร์กับอาร์เรย์สามารถทำได้อีกรูปแบบได้ด้วยการประกาศพอยน์เตอร์มาชี้ตำแหน่งของตัวแปรได้ดัง เช่น

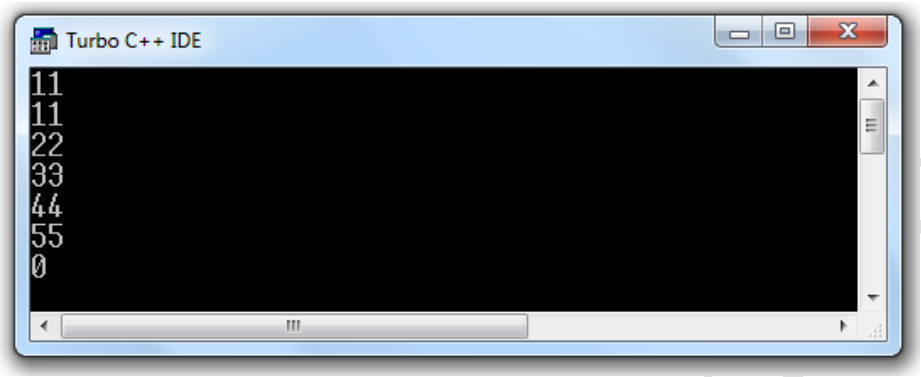
```
int  ab[5];          //ประกาศให้ ab เป็นอาร์เรย์ชนิด integer 1 มิติ มีสมาชิด 5 สมาชิก

int  *pt; //ประกาศให้ pt เป็นพอยน์เตอร์
```

pt = ab; //กำหนดให้การอ้างถึงข้อมูลที่เก็บในอาร์เรย์ ab โดยการนำแอดเดรสของอาร์เรย์ a[0] ซึ่งเป็นตำแหน่งแรกของอาร์เรย์ไปเก็บไว้ในพอยต์เตอร์ pt เหมือนกับการเขียนคำสั่ง pt = &a[0]; แสดงในโปรแกรมที่ 7.6

ตัวอย่างโปรแกรมที่ 7.6 แสดงการอ้างถึงข้อมูลที่อยู่ในอาร์เรย์ อีกรูปแบบหนึ่งทำได้ เช่น อ้างถึงชื่ออาร์เรย์นั้นๆโดยตรง หรือใช้พอยน์เตอร์แสดงตำแหน่งของอาร์เรย์แทนได้

```
#include<stdio.h>
#include<conio.h>
void main()
{   clrscr();
  int ab[5]={11,22,33,44,55};
  int *pt; pt = ab;
  printf("%d\n",ab[0]);
  printf("%d\n",*pt);
  printf("%d\n",*(pt+1));
  printf("%d\n",*(ab+2));
  printf("%d\n",pt[3]);
  printf("%d\n",pt[4]);
  printf("%d\n",pt[5]);
  getch();
}
```

ผลการทำงานโปรแกรม



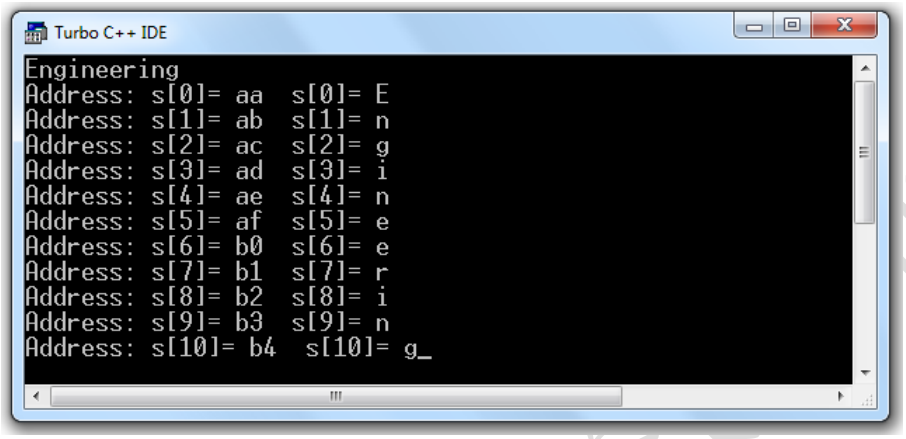

**ข้อสังเกต** ในการกำหนดให้พอยน์เตอร์ ชี้ที่อาร์เรย์ของ ab ห้ามเขียน pt = &ab เพราะอาร์เรย์ ab เป็นการอ้างตำแหน่งโดยหน้าที่อยู่แล้ว ดังนั้นถ้าใส่แอมพาแซนส์ (&) ตัวคอมไพเลอร์จะแจ้งเกิดผิดพลาด

ในการเก็บข้อมูลที่เป็นข้อความในลักษณะ string จะใช้อาร์เรย์ในการเก็บข้อมูลได้ โดยทำการกำหนดผ่านพอยน์เตอร์ ที่เป็นชนิดข้อมูล character ซึ่งพอยน์เตอร์จะทำการเก็บข้อมูลลักษณะอาร์เรย์ ดังตัวอย่างโปรแกรมที่ 7.7

ตัวอย่างโปรแกรมที่ 7.7 แสดงการจัดการพอยน์เตอร์กับอาร์เรย์เก็บตัวอักขระ

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        char *s;
        int i;
        s = "Engineering";
        printf(s);
        for(i=0;i<11;i++)
        printf("\nAddress: s[%d]= %x  s[%d]= %c",i,&s[i],i,s[i]);
        getch();
}
```

ผลการทำงานโปรแกรม




## สรุป

พอยน์เตอร์ มีประโยชน์อย่างมากในการจัดการจัดระบบหน่วยความจำ ให้สามารถจองตำแหน่งที่ใช้ในการเก็บข้อมูล ผู้เขียนโปรแกรมจึงทราบตำแหน่งที่แท้จริงของฮาร์ดแวร์ว่าในการเก็บข้อมูลของส่วนที่เป็นตัวโค๊ดโปรแกรม กับส่วนของข้อมูลที่ใช้ร่วมในการประมวลผล เก็บไว้ตำแหน่งเลขที่(address)ใด มีขนาดเท่าใด ทำให้ผู้เขียนโปรแกรมสามารถวางแผนการใช้หน่วยความจำได้อย่างมีประสิทธิภาพ โดยตัวดำเนินการของพอยน์เตอร์ที่สำคัญมีอยู่ 2 เครื่องหมายคือ เครื่องหมาย * ทำหน้าที่ให้ตัวแปรหลังเครื่องหมาย * เป็นตัวแปรเก็บเลขที่ตำแหน่งหน่วยความจำ ส่วนเครื่องหมาย & ทำหน้าที่ให้ตัวแปรหลังเครื่องหมาย & เป็นตัวแปรเก็บค่าของข้อมูล เมื่อมีการเรียกใช้ &ตัวแปร ผลการประมวลผลโปรแกรม จำทำการนำค่าที่ ตำแหน่งของตัวแปรออกไปใช้งาน ส่วนประโยชน์อีกด้านของพอยน์เตอร์ คือ การจัดการตำแหน่งหน่วยความจำได้ง่าย กล่าวคือ เมื่อกำหนดให้พอยน์เตอร์ชี้แอสเดรสเริ่มต้นที่ใช้ในการเก็บข้อมูลของโปรแกรมแล้ว การเขียนโปรแกรมให้มีการเพิ่มค่าตำแหน่งแอสเดรสสามารถเพิ่มได้โดยอัตโนมัติ อาจกำหนดได้ด้วยฟังก์ชัน for() หรือ ฟังก์ชันอื่นๆ ได้เป็นต้น

# แบบฝึกหัดท้ายบทที่ 7

**ตอนที่ 1** จงเติมคำหรือข้อความในช่องว่างต่อไปนี้ให้ถูกต้อง

1. จงเขียนอธิบายความแตกต่างระหว่างคำสั่ง *Pointer1 กับ &Pointer2

   .....

2. จงเขียนความสัมพันธ์ระหว่างตัวแปร (ตัวแปรชื่อ var) กับพอยน์เตอร์ที่เก็บที่อยู่ (พอยน์เตอร์ชื่อ Pvar)

   .....

3. พอยน์เตอร์นำมาใช้งานในการเขียนโปรแกรมมีเป้าหมายด้านใด

   .....

4. จงเขียนความสัมพันธ์ระหว่างพอยน์เตอร์กับอาร์เรย์ ว่ามีความเหมือนและความแตกต่างกันอย่างไร

   .....

5. ใจงอธิบายการใช้อาร์เรย์หนึ่งมิติกับพอยน์เตอร์ในการเก็บค่าตัวอักขระมา 1 ตัวอย่าง

   .....

   จากการประกาศอาร์เรย์ด้านล่างดังนี้ ใช้ในการตอบข้อ 6-10

```
float arr[2][3] = {{10.5, 20.6, 30.7},
                   {100.55, 200.55, 300.55}};
```

6. อาร์เรย์ด้านบนเป็นชนิดใด

   .....

7. (arr+1) มีผลต่ออาร์เรย์อย่างไร

   .....

8. *(arr+1) มีผลต่ออาร์เรย์อย่างไร

   .....

9. (*(arr+1) +1) มีผลต่ออาร์เรย์อย่างไร

   .....

10. 7.5 (*(arr)+1) มีผลต่ออาร์เรย์อย่างไร

    .....

**ตอนที่ 2** จงทำเครื่องหมายกากบาท (x) ทับหน้าข้อที่ถูกต้องที่สุด

1. คำสั่งข้อใดเป็นการประกาศพอยน์เตอร์ชนิดเลขจำนวนเต็ม

   ก. int *pa;  ข. char *pb;

   ค. float *pc;  ง. double *pd;

2. คำสั่งข้อใดเป็นการประกาศพอยน์เตอร์ชนิดเลขทศนิยมชนิดความยาวสองเท่า

   ก. int *pa;  ข. char *pb;

   ค. float *pc;  ง. double *pd;

3. คำสั่งข้อใดเป็นการประกาศพอยน์เตอร์ชนิดเก็บตัวอักษร

   ก. int *pa;  ข. char *pb;

   ค. float *pc;  ง. double *pd;

4. เครื่องหมายใดใช้ในการกำหนดให้ตัวแปรเก็บที่อยู่

   ก. *  ข. &

   ค. %  ง. #

จากโค๊ดด้านล่างใช้ตอบคำถามข้อ 5-8

```
บรรทัดที่ 1    int one=55, two;
บรรทัดที่ 2     int *p_one, *p_two;
บรรทัดที่ 3    p_one = &one;
บรรทัดที่ 4    two = *p_one;
บรรทัดที่ 5    p_two = &two;
```

5. บรรทัดใดเป็นการประกาศตัวแปรให้เป็นพอยน์เตอร์ให้สามารถเก็บที่อยู่ของหน่วยความจำ

ก. 1 ข. 2

ค. 3 ง. 4

6. บรรทัดใดเป็นการนำค่าตำแหน่งหน่วยความจำไปเก็บที่พอยน์เตอร์

ก. 1 ข. 2

ค. 3 ง. 4

7. บรรทัดใดเป็นการนำค่าในพอยน์เตอร์ ไปเก็บที่ตัวแปร

ก. 2 ข. 3

ค. 4 ง. 5

8. บรรทัดใดเป็นการนำค่าตำแหน่งหน่วยความจำไปเก็บที่ตัวแปรพอยน์เตอร์

ก. 2 ข. 3

ค. 4 ง. 5

9. ข้อใดเป็นการประกาศพอยน์เตอร์ที่ใช้ในการเก็บตัวแปรเลขจำนวนเต็มอาร์เรย์

ก. int x; int x2; x = x2 ข. int x; int *x2; x = x2

ค. float y; int *y2; x = x2 ง. float y[ ]; float *y2; y = y2

10. ข้อใดเป็นการประกาศพอยน์เตอร์ที่ใช้ในการเก็บตัวแปรเลขจำนวนเต็ม

ก. int x; int x2;x = x2 ข. int x; int *x2;x = x2

ค. float y; int *y2;x = x2 ง. float y[ ]; int *y2;y = y2

---

**ตอนที่ 3** จงทำการวิเคราะห์คำถามและทำการเขียนอภิปรายคำตอบตามที่ผู้อ่านเข้าใจโดยยึดความถูกต้องของเนื้อหาประกอบการบรรยาย

1. จงเขียนความสัมพันธ์ระหว่างการส่งอาร์กิวเมนต์กับฟังก์ชันจากตัวอย่างโปรแกรมด้านล่างดังนี้

```
void main
{
void func(int *p);
int x[10]={1,2,3,4,5,6,7,8,9,0};
.          .          .
func(x);
}
void func(int* p)
{
int i, total = 0;
for (i =0; i <10; i ++)
        total += *(p+i);
printf("Total = %d",total);
return;
}
```

ก. การส่งค่าอาร์กิวเมนต์ให้กับ func เป็นชนิดใด

ข. ข้อมูลที่ส่งมาจาก func กลับมาให้ฟังก์ชันหลักเป็นข้อมูลชนิดใด

ค. การวนรอบของคำสั่ง for ใน func มีวัตถุประสงค์ใด

ง. จงเขียนผลการพิมพ์ใน func ว่าแสดงค่าอะไร

2. จากการประกาศอาร์เรย์ด้านล่างดังนี้

```
float arr[2][3] = {{10.5, 20.6, 30.7},
                   {100.55, 200.55, 300.55}};
```

   ก. ค่าของ *(*(arr)+1) หมายถึงอะไร

   ข. ค่าของ *(*(arr+1)) หมายถึงอะไร

   ค. ค่าของ *(*(arr)+1)+1) หมายถึงอะไร

3. จากการประกาศอาร์เรย์ด้านล่างดังนี้

```
char *arr2[5] = {“somsri”, “somchai”, “somjai”, “sombat”, “somsak”};
```

   ก. อาร์เรย์ด้านบนเป็นชนิดใด

   ข. (arr2+2) มีผลต่ออาร์เรย์อย่างไร

   ค. *arr2 มีหมายความว่าอย่างไร

   ง. *(arr2+2) มีความหมายว่าอย่างไร

4. ประกาศพอยน์เตอร์ของการเขียนโปรแกรมดังนี้

   ก. float (*p) (int *x) มีความหมายว่าอย่างไร

   ข. float (*p (int *x))[10] มีความหมายว่าอย่างไร

   ค. float p (int (*x)[ ]) มีความหมายว่าอย่างไร

   ง. float p (int *x[ ]) มีความหมายว่าอย่างไร

   จ. float *p (int x[ ]) มีความหมายว่าอย่างไร

ฉ. float *p (int (*x)[ ]) มีความหมายว่าอย่างไร

ช. float *p (int *x[ ]) มีความหมายว่าอย่างไร

ซ. float (*p) (int (*x)[ ]) มีความหมายว่าอย่างไร

ฌ. float *(*p) (int *x[ ]) มีความหมายว่าอย่างไร

ญ. float (*p[10] (int x) มีความหมายว่าอย่างไร

ฎ. float *(*p[10] (int *x) มีความหมายว่าอย่างไร

5. การประกาศพอยน์เตอร์ดังนี้

ก. ฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นพอยน์เตอร์เป็นชนิดจำนวนเต็ม และส่งค่ากลับเป็นพอยน์เตอร์เป็นอาร์เรย์ชนิดตัวอักขระที่มีจำนวน สมาชิก 10 ตัว

ข. ฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นพอยน์เตอร์ไปยังอาร์เรย์ของจำนวนเต็ม และส่งค่ากลับเป็นพอยน์เตอร์เป็นชนิดตัวอักขระ

ค. ฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นอาร์เรย์ของพอยน์เตอร์ไปยังจำนวนเต็ม และส่งค่ากลับเป็นพอยน์เตอร์เป็นชนิดตัวอักขระ

ง. ฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นอาร์เรย์ของจำนวนเต็ม และส่งค่ากลับเป็นพอยน์เตอร์เป็นชนิดตัวอักขระ

จ. ฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นพอยน์เตอร์ไปยังอาร์เรย์จำนวนเต็ม และส่งค่ากลับเป็นพอยน์เตอร์ไปยังตัวอักขระ

ฉ. ฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นอาร์เรย์ของพอยน์เตอร์ไปยังจำนวนเต็ม และส่งค่ากลับเป็นพอยน์เตอร์ไปยังตัวอักขระ

ช. พอยน์เตอร์ไปยังฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นพอยน์เตอร์ไปยังอาร์เรย์ของจำนวนเต็ม และส่งค่ากลับเป็นตัวอักขระ

ซ. พอยน์เตอร์ไปยังฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นพอยน์เตอร์ไปยังอาร์เรย์ของจำนวนเต็ม และส่งค่ากลับเป็นพอยน์เตอร์ไปยังตัวอักขระ

ฌ. พอยน์เตอร์ไปยังฟังก์ชันซึ่งรับอาร์กิวเมนต์เป็นอาร์เรย์ของพอยน์เตอร์ไปยังจำนวนเต็ม และส่งค่ากลับเป็นพอยน์เตอร์ไปยังตัวอักขระ

ญ. อาร์เรย์ซึ่งมีสมาชิกจำนวน 10 ตัวของพอยน์เตอร์ไปยังฟังก์ชัน ซึ่งแต่ละฟังก์ชันรับอาร์กิวเมนต์ 2 ตัว เป็นตัวแปรชนิดทศนิยมความเที่ยงสองเท่า และส่งค่ากลับเป็นทศนิยมความเที่ยงสองเท่า

ฎ. อาร์เรย์ซึ่งมีสมาชิกจำนวน 10 ตัวของพอยน์เตอร์ไปยังฟังก์ชัน ซึ่งแต่ละฟังก์ชันรับอาร์กิวเมนต์ 2 ตัว เป็นตัวแปรชนิดทศนิยมความเที่ยงสองเท่า และส่งค่ากลับเป็นพอยน์เตอร์ไปยังทศนิยมความเที่ยงสองเท่า

ฏ. อาร์เรย์ซึ่งมีสมาชิกจำนวน 10 ตัวของพอยน์เตอร์ไปยังฟังก์ชัน ซึ่งแต่ละฟังก์ชันรับอาร์กิวเมนต์ 2 ตัว เป็นพอยน์เตอร์ไปยังตัวแปรชนิดทศนิยมความเที่ยงสองเท่า และส่งค่ากลับเป็นพอยน์เตอร์ไปยังทศนิยมความเที่ยงสองเท่า

---

# เอกสารอ้างอิง

ศรัณย์ อินทโกสุม (2539). ทฤษฎีและตัวอย่างโจทย์การเขียนโปรแกรมด้วยภาษาซี กรุงเทพฯ : แมคกรอฮิล อินเตอร์เนชั่นแนล เอ็นเตอร์ไพรส์, อิงค์.

ธันวา ศรีประโมง (2539). การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม. พิมพ์ครั้งที่ 4. กรุงเทพฯ : มหาวิทยาลัยเทคโนโลยีมหานคร.

วิจักษณ์ ศรีสัจจะเลิศวาจา และดุษฎี ประเสริฐธิติพงษ์ ออนไลน์ : www.satit.su.ac.th/soottin.

Alexander, A. Tutorial Online: http://www.cprogramming.com.

Brian, W. K. Programming in C: A Tutorial Online : http://www.lysator.liu.se/c/bwktutor.html.

Byron S. Gottfried, “Schaum ’s Theory and problems of programming with c” McGraw-Hill,Inc., 1990.

Steven, H. & Lutfar, R. (2006). Art of Programming Contest: C Programming | Data Structure | Algorithms (ACM supported), 2nd Edition.