CP 111 Computer Programming (2/2554 Section B01-B02)

Handout 7

<u>หัวข้อ:</u>

Pointers and Arrays (Chapter 16 textbook)

- ทบทวน Arrays (Chapter 16.3.1, 16.3.2) ดู Handout 6

- ทบทวน Strings in C (Chapter 16.3.4) ดู Handout 6

- Pointers (Chapter 16.2.1, 16.2.2, 16.2.4)
- Passing a reference using pointers (Chapter 16.2.3)
- Passing arrays as function arguments (Chapter 16.3.3)
- The Relationship between arrays and pointers in C (Chapter 16.3.5)

} Today

<u>Pointers</u>

1. พิจารณาโปรแกรมข้างล่าง ที่ประกอบด้วยฟังก์ชัน Swap() ซึ่งจะทำการสลับค่าของตัวแปรที่ส่งมาเป็นอาร์กิวเมนต์ของฟังก์ชัน จงระบุผลลัพธ์ของโปรแกรม

```
#include <stdio.h>
void Swap(int,int);
int main()
{
int a,b;
        a=3;
        b=4;
        printf("Before: %d %d\n",a,b);
        Swap(a,b);
        printf("After: %d %d\n",a,b);
       return 0;
}
```

```
void Swap(int first,int second)
{
 int temp;

         temp=first;

         first=second;

         second=temp;

}
```

2. จากตัวอย่างโปรแกรมในข้อ 1 การส่งค่าของตัวแปร a และ b มาเป็นอาร์กิวเมนต์ first และ second ตามลำดับ เรียกว่า การส่งค่าแบบ pass by value นั่นคือ Caller function (ในที่นี้คือฟังก์ชัน main) จะ <u>copy</u> ค่าของตัวแปร a และ b ส่งไปเป็นอาร์กิวเมนต์ของ Callee (ผู้ถูกเรียก) function (ในที่นี้คือฟังก์ชัน Swap)

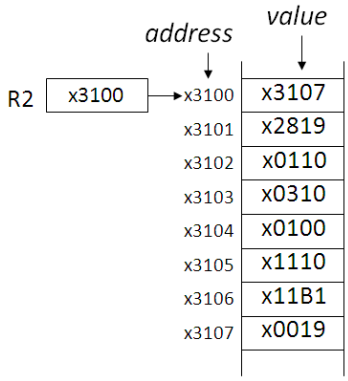
ในกรณีที่เราต้องการให้ฟังก์ชัน Swap ทำการสลับค่าของตัวแปรที่ประกาศใน main() จำเป็นที่ต้องมีกลไกที่จะต้องส่งค่าแอดเดรสของตัวแปรมาให้ฟังก์ชัน ซึ่งในภาษา C จะทำได้โดยตัวแปรชนิด pointer (และเรียกวิธีการส่งค่าแอดเดรสของตัวแปรมาเป็นอาร์กิวเมนต์ว่า pass by reference)

3. Pointer เป็นชนิดของตัวแปร (Derived data type) ที่จะเอาไว้เก็บแอดเดรสในหน่วยความจำของ object (ตัวแปร) หนึ่งๆ

4. ทนทวน Memory address และ value ที่ได้เรียนจาก CP121

เขียนโปรแกรม LC-3 ทำการบวก M[x3100]+M[x3101] +…+M[x3107]

```
.orig x3000
LEA R2,x0FF     ; R2 <- x3100
LDR R4,R2,#0    ;R4 <- M[R2]  (= M[x3100])
LDR R5,R2,#7    ;R5 <- M[R2+7]  (= M[x3107])
```



ในบริบทที่เราจะเรียนต่อไป เราสามารถมองได้ว่า R2 เป็น pointer (ตัวชี้) เพราะมันเก็บแอดเดรสที่ชี้ไปยังข้อมูลที่เราสนใจ

5. การประกาศตัวแปรชนิด pointer

Syntax: `<data-type> *pt_var_name;`

ตัวอย่างเช่น

```
int *ptr1;  /* ประกาศตัวแปรชื่อ ptr1 โดยมีชนิดข้อมูลแบบ pointer to int
               นั้นคือตัวแปร ptr1 จะเอาไว้เก็บค่าแอดเดรสของตัวแปรแบบ int
               หรือสามารถพูดว่าตัวแปร ptr1 จะชี้ไปยังตัวแปรแบบ int ตัวหนึ่ง */
double *pd;  /* ประกาศตัวแปรชื่อ pd โดยมีชนิดข้อมูลแบบ pointer to double
               นั้นคือตัวแปร pd จะเก็บค่าแอดเดรสของตัวแปรแบบ double */
char *pStr;
```

6. ตัวกระทำ (operator) ที่สำคัญสำหรับตัวแปรแบบ pointer มี 2 ตัวคือ Address operator (ใช้เครื่องหมาย &) และ Indirection operator (ใช้เครื่องหมาย *)

6.1 Address operator

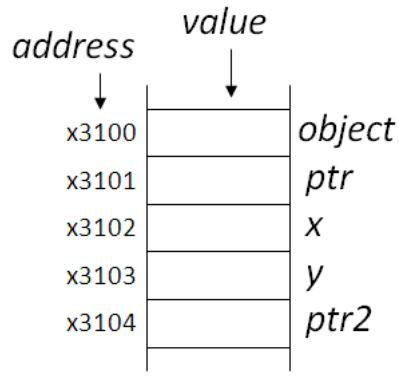
- เป็น operator ที่เอาไว้สำหรับดึงค่า memory address ของตัวแปรใดๆ
- ใช้เครื่องหมาย & นำหน้าตัวแปรใดๆ ซึ่งจะเทียบเท่ากับค่าแอดเดรสของตัวแปรนั้น เช่น &object

```
{
  int object;
  int *ptr;
            object=4;      // Line 1
            ptr=&object;   // Line 2
                              /* &object เทียบเท่ากับค่า memory address ของตัวแปร object */
                              /* object เทียบเท่ากับค่า ของตัวแปร object ซึ่งเท่ากับ 4 */
}
```

Question: สมมติตัวแปร object และ ptr ที่เก็บไว้หน่วยความจำแสดงดังรูปข้างล่าง จงระบุค่าของ object และ ptr เมื่อส่วนย่อยของโปรแกรมข้างบนทำงานถึง Line 2

6.2 Indirection operator

- ใช้เครื่องหมาย * นำหน้าตัวแปรแบบ pointer เพื่ออ้างถึงค่า (value) ของตัวแปรที่แอดเดรสของมันจะถูกไว้ที่ตัวแปรแบบ pointer




```
{
  int object,x,y;
  int *ptr,*ptr2;
              object=4;
              ptr=&object;
              ptr2=&x;        // Line 1
              x=*ptr;         // Line 2
              *ptr=x+1;       // Line 3
              ptr2=ptr;       // Line 4
              y=*ptr2;        // Line 5
}
```

Question: สมมติตัวแปรต่างๆของส่วนย่อยโปรแกรมข้างบน ถูกเก็บไว้หน่วยความจำแสดงดังรูป จงระบุค่าของ object, x, ptr และ ptr2 เมื่อส่วนย่อยของโปรแกรมข้างบนทำงานถึง Line 3

.....

Question: จงระบุค่าของ ptr2 และ y เมื่อส่วนย่อยของโปรแกรมข้างบนทำงานถึง Line 5 .....

7. โจทย์ตัวอย่าง (Final Exam 2553)

กำหนดให้ค่าในหน่วยความจำที่แอดเดรสต่างๆ ที่เกี่ยวข้องกับคำถามเป็นดังรูปข้างล่างซ้าย และกำหนดให้ตัวแปร x ถูกประกาศให้มีชนิดข้อมูลแบบ int ** (pointer to pointer to int) และ**ถูกจัดเก็บไว้ที่หน่วยความจำที่แอดเดรส x4006**

คำถาม: จงระบุค่าของนิพจน์ (Expression) (i) – (iv) ที่ระบุไว้ในตารางข้างล่างขวา

x4000	x4006
x4001	x4002
x4002	x4001
x4003	x4000
x4004	x4004
x4005	x4007
X4006	x4003
x4007	x1234

```
{ int **x;
 //x is at the address  x4006
}
```

(i) x	
(ii) &x	
(ii) *x	
(iv) *(*x)	

8. โจทย์ตัวอย่าง (ดัดแปลงจาก Final Exam 2553)

จากโปรแกรมด้านล่างสมมติว่าตัวแปรต่างๆ ถูกกำหนดให้อยู่ที่หน่วยความจำที่แอดเดรสดังตารางด้านข้าง จงเติมค่าใน Value ให้เป็นค่าที่ตรงต่อการประมวลผลของโปรแกรม

```
int main()
{
int *ptr1,*ptr2;
int **Dptr;
int x,y;
       x=1;
       y=2;
       Dptr =&ptr1;
       ptr1=&x;
       ptr2=&y;
       (*(*Dptr)) = (*(*Dptr)) + 1;
       *ptr1 = *ptr1 + *ptr2;
       *ptr2 = *ptr2 - *ptr1;
}
```

Memory address	Value	variables
xEF00		ptr1
xEF04		ptr2
xEF08		x
xEF0C		y
xEF10		Dptr

Passing a reference using pointers

1. โดยใช้แนวคิดเรื่อง pointer เราสามารถทำการส่งค่าแบบ pass by reference นั้นคือส่งแอดเดรสของตัวแปรไปเป็นอาร์กิวเมนต์ของฟังก์ชันได้ดังตัวอย่างซึ่งดัดมาจากโปรแกรมก่อนหน้านี้ จงระบุผลลัพธ์ของโปรแกรม

```
#include <stdio.h>
void NewSwap(int *,int *);
int main()
{
int a,b;
int *ptr_a,*ptr_b;
        a=3;
        b=4;
        printf("Before: %d %d\n",a,b);
        ptr_a=&a;
        NewSwap(ptr_a,&b);
        printf("After: %d %d\n",a,b);
        return 0;
}
```

```
void NewSwap(int *first,int *second)
{
 int temp;

         temp=*first;

        *first=*second;

        *second=temp;

}
```

2. จริงๆ แล้วก่อนหน้านี้ เราได้เรียกฟังก์ชันโดยการส่งค่าแบบ pass by reference มาตั้งแต่ lecture แรก

```
{
int a;
double x;
     scanf("%d",&a); /* เรียกฟังก์ชัน scanf โดยส่งค่าแอดเดรสของ a ไปเป็นอาร์กิวเมนต์ของฟังก์ชัน
                        ในฟังก์ชัน scanf จะเขียนค่าลงในตัวแปร a */
    scanf("%lf",&x); /* เรียกฟังก์ชัน scanf โดยส่งค่าแอดเดรสของ x ไปเป็นอาร์กิวเมนต์ของฟังก์ชัน */
}
```

3. ประโยชน์ที่สำคัญของการใช้ตัวแปรแบบ pointer ในการส่งค่าระหว่างฟังก์ชันแบบ pass by reference คือทำให้เราสามารถเขียนโปรแกรมเพื่อให้ฟังก์ชันผลิตผลลัพธ์กลับไปได้มากกว่า 1 ค่า

โจทย์ตัวอย่าง: จงทำโปรแกรมข้างล่างให้สมบูรณ์โดยการเขียนและเรียกฟังก์ชัน Quad_Eq ซึ่งจะทำการหาราก (root) หรือค่า x ที่ทำให้สมการ $ax^2 + bx + c = 0$ โดยให้ผลิตค่ารากทั้งสองไปที่ตัวแปรที่แอดเดรสถูกส่งมาเป็น argument ของฟังก์ชัน (การรับส่งพารามิเตอร์เป็นแบบ pass-by-reference)

นอกจากนี้กำหนดให้ฟังก์ชันคืนค่าสถานะการทำงานของฟังก์ชันโดยคืนค่า 0 ถ้ารากทั้งสองของสมการเป็นจำนวนจริง (real number) มิฉะนั้นคืนค่า -1 (ถ้ามีรากอย่างน้อย 1 ตัวไม่ใช่จำนวนจริงหรือ $b^2 - 4ac < 0$)

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

int Quad_Eq(double, double, double, ……..……);

int main()
{
double a,b,c,x1,x2;

      scanf("%lf %lf %lf",&a,&b,&c);

      retval=Quad_Eq(a,b,c, ……);

      if(retval==0)
           printf("%.2f %.2f\n",x1,x2);
      else
           printf("Root is complex.\n");
      return 0;
}
int Quad_Eq(double a,double b, double c, ……)
{
double K;

        K=(b*b)-(4*a*c);

        if(……..……)
        {
              return -1;
        }
        else
        {
             ……..……

             ……..……

             ……..……
        }
}
```

4. โจทย์ตัวอย่าง Pass by reference: จงทำโปรแกรมข้างล่างให้สมบูรณ์เพื่อสร้างและเรียกใช้ฟังก์ชัน Extract_Date ซึ่งมีฟังก์ชันอาร์กิวเมนต์ทั้งหมด 4 ตัว และฟังก์ชันจะไม่คืนค่าอะไร

- ฟังก์ชันอาร์กิวเมนต์ตัวแรก เป็นค่าแบบ int ซึ่งเป็นตัวเลขจำนวนเต็ม 8 หลักที่เก็บวันเดือนปีในรูปแบบ mmddyyyy โดยที่ dd แทนวันที่ (00-31), mm แทนเดือน (01-12) และ yyyy แทนปี (0000-9999) เช่น 11282554 หมายถึง วันที่ 28 เดือน 11 ปี 2554

- ฟังก์ชันอาร์กิวเมนต์ตัวที่ 2, 3 และ 4 จะเป็นการส่งค่าแบบ <u>pass by reference</u> ซึ่งจะมีชนิดข้อมูลแบบ pointer to an *int* ซึ่งจะเป็นค่าของ address ของตัวแปรแบบ int ที่ใช้เก็บวันที่, ค่าของ address ของตัวแปรแบบ int ที่ใช้เก็บเดือน และ ค่าของ address ของตัวแปรแบบ int ที่ใช้เก็บปี ที่จะส่งมาจากฟังก์ชัน ตามลำดับ

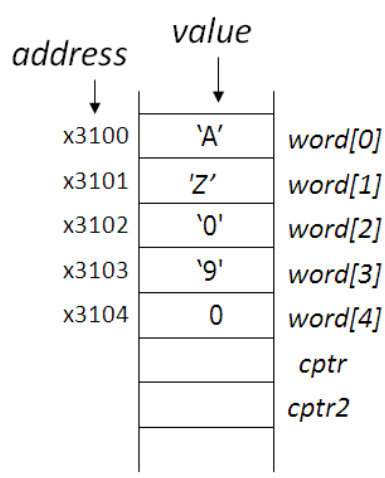
```
#include <stdio.h>
void Extract_Date(int, int *, int *, int *);
int main()
{
int i_date,dd,mm,yyyy;
      scanf("%d",&i_date); // e.g. 11282554
      Extract_Date (i_date,……);
      printf("Today: Day=%d, Month=%d, Year=%d\n",dd,mm,yyyy); // 28-11-2554
      return 0;
}

void Extract_Date(int date, int *d, int *m, int *y)
{
   ……
   ……
   ……
   ……
}
```

<u>The Relationship between arrays and pointers in C</u>

1. ชื่อของตัวแปรอาร์เรย์เทียบเท่ากับค่าแอดเดรสของ element แรกของอาร์เรย์
2. แต่เนื่องจากแอดเดรสนี้ไม่เปลี่ยนแปลงจึงเทียบเท่ากับ pointer constant (ไม่ใช่ pointer variable)
3. เราสามารถใช้ตัวแปรแบบ pointer ในการอ้างถึง array ได้ ดังตัวอย่างต่อไปนี้

```
char word[5];
char *cptr,*cptr2;
char a,b,c,d,e;
      cptr=word;          // Array name = pointer constant
      cptr2=&word[0];  // Address ของ element แรก
      // cptr จะมีค่าเท่ากับ cptr2
      a=word[0];
      b=*cptr;            // a จะมีค่าเท่ากับ b เท่ากับ word[0]
      c=word[3];
      cptr2=cptr+3;
      d=*cptr2;
      e=*(cptr+3);       // c จะมีค่าเท่ากับ d เท่ากับ e เท่ากับ word[3]
  // cptr+3 คือค่าแอดเดรสที่ได้จาก cptr บวกกับค่า 3xขนาดของ element size (sizeof(char))
```



4. สรุป Notation จาก Table 16.1 page 446 textbook ค่าในแต่ละแถวต่อไปนี้เทียบเท่ากัน (เหมือนกัน)

```
char word[10];
char *cptr
```

Pointer	Pointer Constant	Array
cptr	word	&word[0]
(cptr+n)	word+n	&word[n]
*cptr	*word	word[0]
*(cptr+n)	*(word+n)	word[n]

Note: ข้อแตกต่างระหว่าง cptr และ word คือ cptr เป็น pointer variable สามารถเปลี่ยนแปลงค่าได้ ส่วน word (ชื่ออาร์เรย์) เป็น pointer constant ไม่สามารถเปลี่ยนค่าได้ นั้นคือการเขียน cptr=word จะถูกต้องทำงานได้ ส่วนการเขียน word=cptr จะไม่ถูกต้องเนื่องจากมีการกำหนดค่าให้ pointer constant ซึ่งไม่ใช้ variable

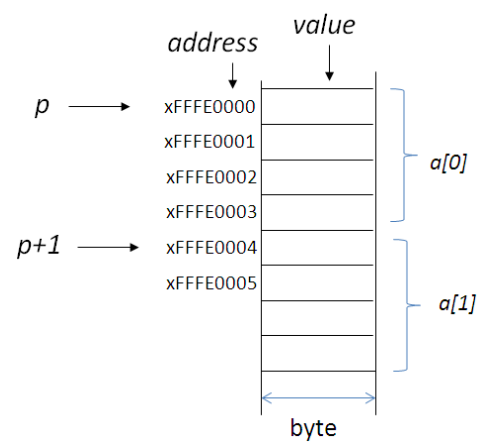
5. โจทย์ตัวอย่าง จงระบุผลลัพธ์ของโปรแกรม

```
void main()
{
int word[5];
int *cptr;
int i;
        word[0]=10;
        word[1]=11;
        cptr=word;
        *cptr=12;
        *(cptr+1)=13;
         i=2;
        *(cptr+i)=20;
        printf("%d %d %d\n",word[0],*(cptr+1),*(word+2));
}
```

6. Pointer arithmetic and element size

- เราสามารถทำการบวกหรือลบค่าของตัวแปรแบบ pointer ได้ดังตัวอย่างเช่น

```
int *p;
int a[5]={1,2,3,4,5};
int b,c;
   p=a;     // กำหนดให้ p เก็บแอดเดรสของ element แรกของอาร์เรย์ a (a[0])
   p=p+1;   // p จะเปลี่ยนป็นแอดเดรสของ element ถัดไปของอาร์เรย์ a (a[1])
   b=*p;    // b=2
   p=p+2;   // p จะเปลี่ยนป็นแอดเดรสของสอง element ถัดไปของอาร์เรย์ a (a[3])
   c=*p;    // c=4
   // Note บน intel win32: ตัวแปร int จะใช้ 4 bytes ในการเก็บข้อมูล
```



- ตัวกระทำ sizeof() คืนค่าจำนวน byte ที่ใช้เก็บตัวแปรหนึ่งๆ

```
int a,s;
        s=sizeof(int); // คืนค่าจำนวน byte ที่ใช้เก็บ int
        s=sizeof(a);   // คืนค่าจำนวน byte ที่ใช้เก็บ a (ตัวแปรแบบ int)
```

Passing arrays as function arguments

1. เราสามารถส่งตัวแปรแบบ array เข้าไปเป็น อาร์กิวเมนต์ของฟังก์ชันได้ดังตัวอย่างโปรแกรมข้างล่าง ซึ่งประกอบไปด้วยฟังก์ชัน Calc_Avg ที่จะทำการคำนวณค่าเฉลี่ยของข้อมูลในอาร์เรย์ x ขนาด n element ที่เป็นอินพุตอาร์กิวเมนต์

```
#include <stdio.h>
double Calc_Avg(int [], int);
/* or
double Calc_Avg(int *, int);
*/
int main()
{
int data[10];
double avg;
int i;
        for(i=0;i<10;i++)
            scanf("%d",&data[i]);

        avg=Calc_Avg(data,10); //Line A

        printf("%.2f\n",avg);
        return 0;
}
```

```
double Calc_Avg(int x[],int n)
/*
 double Calc_Avg(int *x,int n)
*/
{
int sum,i;
double avg;

       sum=0;
       for(i=0;i<n;i++)
           sum=sum+x[i];

      avg=(double)sum/n;

    return(avg);
}
```

Note:

(i) การส่งอาร์เรย์ data มาเป็นอาร์กิวเมนต์ของฟังก์ชัน Calc_Avg ที่บรรทัด Line A ใน main() จะเป็นการส่งแบบ pass by reference นั้นคือจะส่งแอดเดรสของ element แรกของอาร์เรย์ data

(ii) ชนิดของข้อมูลของอาร์กิวเมนต์ที่เป็น array สามารถเขียนในรูป int ชื่อตัวแปร [] หรืออาจจะใช้ตัวแปรแบบ pointer ก็ได้

2. โจทย์ตัวอย่าง จงทำโปรแกรมข้างล่างให้สมบูรณ์เพื่อสร้างและใช้ฟังก์ชัน Calc_Avg_Std ซึ่งจะทำการคำนวณหาค่าเฉลี่ย (mean) และค่าเบี่ยงเบนมาตราฐาน (std) ของอินพุตอาร์เรย์และผลิตผลลัพธ์ (ส่งค่าแบบ pass by reference) กลับไปสองค่าคือค่า mean และ std ที่คำนวณได้

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

void Calc_Avg_Std.......................................

int main()
{
int data[10];
double avg,std;

  for(i=0;i<10;i++)
      scanf("%d",&data[i]);

   Calc_Avg(data,10,...............................);

   printf("%.2f %.2f\n",avg,std);

   return 0;
}
```

```
void Calc_Avg_Std.....................................
{

}
```