Lecture 2

เอกสารประกอบการบรรยาย
Data Structure

เรื่อง Array and Record

Array and Record

เนื้อหา

-การดำเนินการเกี่ยวข้องกับอะเรย์ 1 มิติ หลายมิติ
-การส่งค่าของอะเรย์ ในโปรแกรม และฟังก์ชัน
-การดำเนินการที่เกี่ยวข้องกับ เรคคอร์ด
-ความสัมพันธ์ระหว่าง เรคคอร์ด กับอะเรย์ และอะเรย์ชนิดโครงสร้าง

จุดประสงค์การเรียนรู้

1. เพื่อให้นักศึกษาทราบวิธีการของอะเรย์
2. เพื่อให้นักศึกษาทราบวิธีการส่งค่าของอะเรย์ในโปรแกรม และฟังก์ชัน
3. เพื่อให้นักศึกษาทราบวิธีการดำเนินการที่เกี่ยวข้องกับเรคคอร์ด ข้อมูล
4. เพื่อให้นักศึกษาทราบความสัมพันธ์ของข้อมูลที่เกิดขึ้นของเรคคอร์ดกับอะเรย์ข้อมูล

Array

อะเรย์เป็นโครงสร้างข้อมูลที่เรียกว่า Linear List มีลักษณะคล้ายเซ็ตในคณิตศาสตร์ คือ อะเรย์จะประกอบด้วยสมาชิกที่มีจำนวนคงที่ มีรูปแบบข้อมูลเป็นแบบเดียวกัน สมาชิกแต่ละตัวใช้เนื้อที่จัดเก็บที่มีขนาดเท่ากัน เรียงต่อเนื่องในหน่วยความจำหลัก

<u>การกำหนด Array</u>

การกำหนดอะเรย์จะต้องกำหนดชื่ออะเรย์ พร้อม subscript ซึ่งเป็นตัวกำหนดขอบเขตของอะเรย์ มีได้มากกว่า 1 ตัวจำนวน subscript จะเป็น ตัวบอกมิติของอะเรย์นั้น อะเรย์ที่มี subscript มากกว่า 1 ตัวขึ้นไป จะเรียกว่า **อะเรย์หลายมิติ**

Array (Cont.)

การกำหนด subscript แต่ละตัวจะประกอบไปด้วย ค่าสูงสุดและ ค่าต่ำสุดของ subscript นั้น การประกาศค่าตัวแปรอะเรย์ ในภาษาคอมพิวเตอร์ บางภาษา เช่น

ภาษาปาสคาล var A: array[1..10] of real;
var K: array[1..5,1..10] of integer;
ภาษาซี float A[10];
int K[5] [10];

Array (Cont.)

ข้อกำหนดของการกำหนดค่าต่ำสุดและค่าสูงสุดของ subscript คือ

1. ค่าต่ำสุดต้องมีค่าน้อยกว่าหรือเท่ากับค่าสูงสุดเสมอ
2. ค่าต่ำสุด เรียกว่า ขอบเขตล่าง (lower bound)
3. ค่าสูงสุด เรียกว่า ขอบเขตบน (upper bound)

Array (Cont.)

ค่า subscript ที่ใช้อ้างอิงถึงสมาชิก จะต้องมีค่ามากกว่า หรือเท่ากับขอบเขตล่าง และน้อยกว่าหรือเท่ากับขอบเขตบน

$$\text{lower bound} \leq \text{subscript} \leq \text{upper bound}$$

ขนาดของ index แต่ละตัว ของ Array หาได้จาก

ขนาดของ subscript = upper bound – lower bound + 1

Array (Cont.)

จำนวนสมาชิกหรือขนาดของอะเรย์ n มิติ หาได้จาก

ขนาดของอะเรย์ = ผลคูณของขนาดของsubscript แต่ละตัว

เช่น

ขนาดของอะเรย์ A = Upper bound - lower bound+1

= 10-1+1 = 10

Array (Cont.)

ขนาดของอะเรย์ K = ผลคูณของขนาดของ subscript แต่ละตัว

= (5-1+1) * (10-1+1)
= 5 * 10
= 50

Array (Cont.)

การจัดเก็บอะเรย์ในหน่วยความจำหลักจะใช้เนื้อที่ขนาด เท่ากันเพื่อเก็บสมาชิกแต่ละตัว โดยเนื้อที่จะเรียงต่อเนื่องกัน การจัดเก็บอะเรย์ในหน่วยความจำหลัก จะพิจารณาตาม ประเภทของอะเรย์ในมิติต่าง ๆ ดังนี้

- อะเรย์ 1 มิติ
- อะเรย์ หลายมิติ

อะเรย์ 1 มิติ

<u>อะเรย์ 1 มิติ</u>
<u>รูปแบบ</u>

data-type array-name[expression]

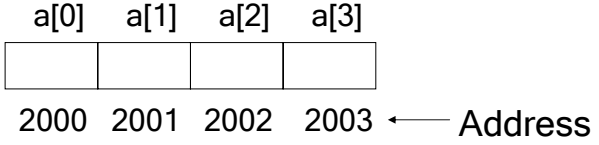
data-type คือ ประเภทของข้อมูลอะเรย์ เช่น int char float
array-name คือ ชื่อของอะเรย์
expression คือ นิพจน์จำนวนเต็มซึ่งระบุจำนวนสมาชิกของอะเรย์

<u>ตัวอย่าง</u> char a[4]; int num[10];

อะเรย์ 1 มิติ (Cont.)

char a[4];

หมายถึง คอมพิวเตอร์จะจองเนื้อที่ในหน่วยความจำสำหรับตัวแปร a ให้เป็นตัวแปรชุดชนิด character ขนาดสมาชิก 4 สมาชิก โดยหน่วย ความจำจะเตรียมเนื้อที่ให้ 1 byte สำหรับ 1 ชื่อตัวแปร



Example

```
#include  <stdio.h>
#define N 5
int  main(void)  {
        int i;
        float score[N];
        float total=0;
        printf("Number of student [%d]\n",N);
        for(i=0; i<N; i++)
                {
                printf(" Input score student[%d]",i);
                scanf("%f",&score[i]);
                total = total + score[i];
                }
        printf("Total is %f\n",total);
        printf("Average is %f\n",total/N);
        return 0;
}
```


Initialization

คือ การกำหนดค่าเริ่มต้นให้กับอะเรย์

<u>การกำหนดค่าให้กับตัวแปรชุดที่มีค่าเป็นตัวเลข</u>

<u>รูปแบบ</u>

data-type array-name[n] = {value1, value2,..,value n};

<u>ตัวอย่าง</u>

int num[5] = {1,2,3,4,5}; หรือ int num[] = {1,2,3,4,5};

float x[6] = {0,0.25,0,0.5,0,0};

Initialization (Cont.)

<u>การกำหนดค่าให้กับตัวแปรชุดชนิด Character</u>

<u>รูปแบบ</u>

char array-name[n] =“string”;

<u>ตัวอย่าง</u>

char ch[9] = “SAWASDEE”;

หรือ char ch[9]={‘S’,‘A’,‘W’,‘A’,‘S’,‘D’,‘E’,‘E’,‘\0’}

มีการกำหนดค่าให้ตัวแปรชุด ch[0] จะเก็บค่า S ถึง ch[7] จะเก็บค่า E และ ch[8] จะเก็บค่า \0 โดยอัตโนมัติ เพื่อแสดงการสิ้นสุดของข้อความ

การส่งอะเรย์ให้ฟังก์ชัน

สามารถกำหนดอะเรย์เป็นพารามิเตอร์ส่งให้กับฟังก์ชันได้ 2 ลักษณะ

1. การกำหนด array element เป็นพารามิเตอร์ส่งค่าให้กับฟังก์ชัน ทำได้โดยอ้างถึงชื่ออะเรย์พร้อมระบุ subscript

<u>ตัวอย่าง</u> swap(num[2],num[3]);

draw_house(color[i],x[i],y[i]);

การส่งอะเรย์ให้ฟังก์ชัน (Cont.)

2. ส่งอะเรย์ทั้งชุดให้ฟังก์ชัน ทำได้โดยอ้างถึงชื่ออะเรย์โดยไม่มี subscript

ตัวอย่าง

```
#define N 10
float a[N]; float avg;
avg = average(N,a);
```


การประกาศอาร์กิวเมนต์ในฟังก์ชันเป็นอะเรย์

ถ้าเป็นอะเรย์มิติเดียว สามารถทำได้ทั้งหมด 3 วิธี

1. มีการประกาศขนาดของอะเรย์ที่ทำหน้าที่ในการรับค่า
2. ไม่ต้องมีการประกาศขนาดของอะเรย์ที่ทำหน้าที่ในการรับค่า
3. ตัวแปรที่ทำหน้าที่รับค่าถูกกำหนดเป็นพอยน์เตอร์

การประกาศอาร์กิวเมนต์ในฟังก์ชันเป็นอะเรย์

1. มีการประกาศขนาดของอะเรย์ที่ทำหน้าที่ในการรับค่า

ตัวอย่าง

```
#define N 5;
    void  display(int b[N])
    {
    int i;
    for(i=0; i<N;i++)
        printf("%d\t",b[i]);
    printf("\n");
}
```


การประกาศอาร์กิวเมนต์ในฟังก์ชันเป็นอะเรย์

2. ไม่ต้องมีการประกาศขนาดของอะเรย์ที่ทำหน้าที่ในการรับค่า

ตัวอย่าง

```
    void  display(int b[ ])
{
    int i;
    for(i=0; i<10;i++)
        printf("%d\t",b[i]);
    printf("\n");
}
```

การประกาศอาร์กิวเมนต์ในฟังก์ชันเป็นอะเรย์

3. ตัวแปรที่ทำหน้าที่รับค่าถูกกำหนดเป็นพอยน์เตอร์

ตัวอย่าง

```
void  display(int *b);
```

** การส่งผ่านอะเรย์ให้กับฟังก์ชันเป็นการส่งผ่านโดยการอ้างอิง เรียกว่า Pass by reference คือ ค่าของสมาชิกแต่ละตัวจะไม่ได้ถูกส่งไปให้ฟังก์ชัน แต่ชื่อของอะเรย์จะถูกมองเป็นตำแหน่งในหน่วยความจำที่ใช้เก็บสมาชิกตัวแรก ซึ่ง address นี้จะถูกส่งให้กับอาร์กิวเมนต์ที่ตรงกัน ดังนั้น อาร์กิวเมนต์จึงเป็น pointer ชี้ไปยังสมาชิกตัวแรก ของอะเรย์

อะเรย์ 2 มิติ

รูปแบบ

type array-name[n] [m];

type	หมายถึง	ชนิดของตัวแปรที่ต้องการประกาศเป็นอะเรย์
array-name	หมายถึง	ชื่อของตัวแปรที่ต้องการประกาศเป็นอะเรย์
n	หมายถึง	ตัวเลขที่แสดงตำแหน่งของแถว
m	หมายถึง	ตัวเลขที่แสดงตำแหน่งของคอลัมน์

อะเรย์ 2 มิติ (Cont.)

ตัวอย่าง

```
char  a[2][3];
          col1        col2        col3
row1   a[0][0]      a[0][1]     a[0][2]
row 2  a[1][0]      a[1][1]     a[1][2]
```

หมายถึง คอมพิวเตอร์จะจองเนื้อที่ในหน่วยความจำ จำนวน 6 ที่ สำหรับตัวแปร a

Initialization

กำหนดค่าเริ่มต้นได้หลายลักษณะ

ตัวอย่าง

กำหนดค่าเริ่มต้นให้ int a[2][3]

```
int a[2][3]  =  {1,2,3,4,5,6};  หรือ
int a[2][3]  =  {{1,2,3},{4,5,6}};  หรือ
int a[][3]   = {{1,2,3},{4,5,6}};
```

Example

```
#include  <stdio.h>
int main(void)
{
        int  a[3][4],r,c;
        for(r=0; r<=2; r++)
                for(c=0;c<=3;c++)
                {
                  printf(" Entry Number  ");
                  scanf("%d",&a[r][c]);
                }
        for(r=0; r<=2; r++)
                for(c=0;c<=3;c++)
                {
                  printf("\n Display");
                  printf("%4d",a[r][c]);
                }
        return 0;
}
```


Record or Structure

เป็นโครงสร้างข้อมูลที่ประกอบขึ้นมาจากข้อมูลพื้นฐานต่างประเภทกัน รวมเป็น 1 ชุดข้อมูล คือ จะประกอบด้วย data element หรือ field ต่างประเภทกันอยู่รวมกัน ในภาษา C ก็คือการกำหนดข้อมูลเป็นรูปแบบของ **Structure**

Structure คือ โครงสร้างที่สมาชิกแต่ละตัวมีประเภทข้อมูลแตกต่างกันได้ โดยที่ใน structure อาจมีสมาชิกเป็นจำนวนเต็ม ทศนิยม อักขระ อะเรย์ หรือพอยเตอร์ หรือแม้แต่ structure ด้วยกันก็ได้

Structure (Cont.)

<u>การนิยาม structure</u>

รูปแบบ struct struc-name {
type name-1;
type name-2;
.....
type name-n;
} struc-variable;

Structure (Cont.)

struct	เป็นคำหลักที่ต้องมีเสมอ
struc-name	ชื่อกลุ่ม structure
type	ชนิดของตัวแปรที่อยู่ในกลุ่ม structure
name-n	ชื่อของตัวแปรที่อยู่ในกลุ่ม structure
struc-variable	ชื่อตัวแปรชนิดโครงสร้าง คือ ตัวแปรที่มีโครงสร้าง เหมือนกับที่ประกาศไว้ใน ชื่อของกลุ่ม structure อาจมีหรือไม่มีก็ได้ ถ้ามีมากกว่า 1 ชื่อ แยกกันด้วยเครื่องหมายคอมม่า (,)

Structure (Cont.)

การประกาศสมาชิกแต่ละตัวของ structure

สมาชิกแต่ละตัวของ structure จะเป็นตัวแปรธรรมดา พอยน์เตอร์ อะเรย์หรือ structure ตัวอื่นก็ได้ โดยชื่อของสมาชิกแต่ละตัวต้องแตกต่างกัน

ตัวอย่าง

```
struct    employee {
        char name[30];
        int   age;
        float   salary;
} personel;
```


Structure (Cont.)

จากตัวอย่าง เป็นการกำหนดให้ตัวแปร employee เป็นชื่อของกลุ่ม structure ที่ประกอบไปด้วย ตัวแปร name[30],age และ salary โดยมีตัวแปร personel เป็นตัวแปรชนิดโครงสร้างที่มีข้อมูลแบบเดียวกับตัวแปร employee

Structure (Cont.)

การกำหนดให้ตัวแปรมีโครงสร้างข้อมูลเหมือนกับ structure ที่ประกาศไว้แล้ว

สามารถกำหนดให้ตัวแปรอื่น ๆ มีโครงสร้างข้อมูลเหมือนกับ structure ที่ประกาศไว้ได้โดยใช้คำสั่ง struct

รูปแบบ

```
struct  struc-name  struc-variable;
```

ถ้ามีหลายตัวแปรจะคั่นด้วยเครื่องหมายคอมม่า (,)

Structure (Cont.)

ตัวอย่าง

```
struct employee {
        char  name[30];
        int    age;
        float  salary;
};
struct  employee  emp1,  emp2;
```

emp1 และ emp2 เป็นตัวแปรแบบ structure ซึ่งมีการระบุ องค์ประกอบไว้ใน employee

Structure (Cont.)

เราสามารถที่จะประกาศ structure หนึ่งเป็นสมาชิกของอีก structure หนึ่งได้ โดยจะต้องประกาศ structure ที่จะนำไปฝังไว้ก่อนหน้า structure ตัวนอก

ตัวอย่าง

```
struct  date  {
        int  month;
        int  day;
        int  year;
};
```


Structure (Cont.)

```
struct account {
        int  acct_no;
        char    name[30];
        struct   date  lastpayment;
}oldcustomer;
```


Structure (Cont.)

การกำหนดค่าเริ่มต้นให้กับสมาชิกของ structure

สามารถกำหนดค่าเริ่มต้นให้กับสมาชิกของ structure ได้ โดยค่าเริ่มต้นที่จะกำหนดให้กับสมาชิกตัวใด จะต้องอยู่ในตำแหน่งที่ตรงกับสมาชิกตัวนั้นค่าเริ่มต้นจะต้องอยู่ในวงเล็บปีกกาและข้อมูลค่าเริ่มต้นแต่ละตัวแยกกันด้วยเครื่องหมาย ,

ตัวอย่าง

structure account ประกอบด้วยสมาชิก ดังนี้

- เลขจำนวนเต็ม (int acct_no)
- อะเรย์ของอักขระจำนวน 30 ตัว (char name[30]);
- structure date

Structure (Cont.)

```
struct  date  {
        int   month;
        int   day;
        int   year;
};
struct account {
        int      acct_no;
        char    name[30];
        struct  date lastpayment;
};
struct account customer = {1234, “John Smith”, 5,24,46};
```

Structure (Cont.)

จะได้ว่า customer เป็นตัวแปรแบบ structure ประเภท account มีการกำหนดค่าเริ่มต้นให้สมาชิกแต่ละตัว ดังนี้

acct_no มีค่าเป็นจำนวนเต็ม 1234
name[30]มีค่าเป็น string “John Smith”
month มีค่าเป็นจำนวนเต็ม 5
day มีค่าเป็นจำนวนเต็ม 24
year มีค่าเป็นจำนวนเต็ม 46

Structure (Cont.)

<u>การอ้างถึงตัวแปรที่อยู่ในตัวแปรชนิดโครงสร้าง</u>

สามารถอ้างถึงตัวแปรที่อยู่ในตัวแปรชนิดโครงสร้างได้

<u>รูปแบบ</u>

struct-variable.element-name
struct-variable ชื่อตัวแปรชนิดโครงสร้าง
element-name ชื่อตัวแปรที่อยู่ภายใน structure

Structure (Cont.)

<u>ตัวอย่าง</u>

```
struct employee {
        char   name[30];
        char   address[20];
        float  salary;
} personel;
```

จากตัวอย่าง ถ้าต้องการนำตัวแปร salary มาใช้งานก็จะอ้างถึงตัวแปร salary ได้โดย

personel.salary

<u>ตัวอย่าง</u>

```
int  main(void) {
        struct  account {
                char  name[30];
                char  addr[30];
                float  salary;
                int  age;
        }new;
        strcpy(new.name, “somsri”);
        strcpy(new.addr, “jantaburi”);
        new.salary = 7500;
        new.age= 35;
        printf(“Name: %s\n Address: %s\n Salary : %.2f\n Age : %d\n”,
                new.name,new.addr,new.salary,new.age);
        return 0;
}
```

Structure (Cont.)

<u>อะเรย์ชนิดโครงสร้าง</u>

<u>รูปแบบ</u>

```
struct  struc-name {
        type  name-1;
        type  name-2;
        .....
        type  name-n;
        } struct-array variable;
```


Structure (Cont.)

<u>ตัวอย่าง</u>

```
struct  employee {
        char    name[30];
        int     age;
        float   salary;
}input[5];
```


Structure (Cont.)

จากตัวอย่าง input เป็นอะเรย์ที่มีสมาชิกสูงสุดได้ 5 ตัว สมาชิกแต่ละตัวเป็น structure การอ้างถึงแต่ละสมาชิกในอะเรย์ ทำได้โดยระบุ subscript และระบุสมาชิกใน structure

<u>รูปแบบ</u>

struct-array-name[subscript].member-name

<u>ตัวอย่าง</u>

ถ้าต้องการเรียกใช้เงินเดือนของสมาชิกของอะเรย์ input ตัวที่ 2 สามารถอ้างถึงได้ดังนี้

input[1].salary

Structure (Cont.)

<u>การกำหนดค่าเริ่มต้นให้กับตัวแปรชุดชนิดโครงสร้าง</u>

<u>ตัวอย่าง</u>

```
struct      account {
        int     acct_no;
        char    name[30];
        int     age;
}customer[2] = {
                {123, “Bill”, 23},
                {125, “John”, 25}
                 };
```

Structure (Cont.)

Structure กับ pointer

เราสามารถที่จะอ้างถึงที่อยู่เริ่มต้นของ structure ได้เหมือนกับตัวแปรอื่น ๆ โดยใช้ตัวดำเนินการ &

ดังนั้น ถ้า variable เป็นตัวแปรประเภท structure &variable จะเป็นเลขที่อยู่เริ่มต้นของตัวแปร นอกจากนี้ยังสามารถประกาศตัวแปรพอยน์เตอร์สำหรับ structure ดังนี้

type *ptvar

type คือ ประเภทข้อมูลที่เป็น structure

ptvar คือ ชื่อของตัวแปรพอยน์เตอร์

Structure (Cont.)

ดังนั้น สามารถกำหนดเลขที่อยู่เริ่มต้นของตัวแปร structure ให้กับตัวแปรพอยน์เตอร์นี้ ได้ ดังนี้

ptvar = &variable

ตัวอย่าง

```
struct account {
        int     acct_no;
        char    name[20];
        int     age;
}customer;
struct  account *p;
```


Structure (Cont.)

จากตัวอย่าง ตัวแปรพอยเตอร์ p จะเป็นตัวแปรที่ทำหน้าที่ชี้ตำแหน่งที่อยู่ของตัวแปรชนิดโครงสร้างที่ชื่อ account

จะกำหนดเลขที่อยู่เริ่มต้นของ customer ให้กับ p ได้ คือ

p = &customer;

Structure (Cont.)

เราสามารถที่จะเรียกใช้สมาชิกแต่ละตัวใน structure ได้จาก ตัวแปรพอยน์เตอร์ ดังนี้

รูปแบบ

ptvar -> member-name

ptvar คือ ตัวแปรพอยน์เตอร์ที่ชี้ไปยัง structure

-> คือ ตัวดำเนินการที่เทียบได้กับตัวดำเนินการ .

member-name คือ สมาชิกของ structure

Structure (Cont.)

หรือ (*ptvar).member-name

ข้อสังเกต ที่ต้องใส่ ( ) ที่ตัวแปรพอยน์เตอร์เพราะเครื่องหมาย .
จะมีลำดับการทำงานสูงกว่าเครื่องหมาย *

Structure (Cont.)

ตัวอย่าง

```
struct  account {
        int   acct_no;
        char      name[30];
        int   age;
}customer;
struct account  *p = &customer;
```


Structure (Cont.)

ถ้าต้องการเรียกใช้ acct_no ของตัวแปร structure ชื่อ customer จะเขียนได้ ดังนี้

customer.acct_no
หรือ p->acct_no
หรือ (*p).acct_no

Structure (Cont.)

การผ่าน structure ให้ฟังก์ชัน

ประเภทของการส่งผ่าน structure ให้ฟังก์ชันนั้น มี 2 ประเภท คือ

1. ส่งสมาชิกแต่ละตัวของ structure
2. ส่งทั้ง structure

Structure (Cont.)

1. ส่งสมาชิกแต่ละตัวของ structure

สมาชิกแต่ละตัวของ structure สามารถส่งเป็นอาร์กิวเมนต์ ของฟังก์ชันและส่งกลับจากฟังก์ชันได้โดยใช้คำสั่ง return ซึ่งมีทั้งการส่งค่าของตัวแปรที่อยู่ในตัวแปรstructure และก็ส่งตำแหน่งที่อยู่ของตัวแปรนั้น ๆ ไปยังฟังก์ชัน

ตัวอย่างการส่งค่าของตัวแปรใน structure ไปยังฟังก์ชัน

```
float cal_annual(float);
int main(void) {
        struct employee{
                char   name[20];
                int    age;
                float  salary;
        }emp1= {“somporn”,26,6500.50};
        float annual;
        annual = cal_annual(emp1.salary);
        return 0;
}
float  cal_annual(float sal) {
        return(sal*12);
}
```


ถ้าส่งเป็นตำแหน่งที่อยู่ของตัวแปรใน structure ไปยังฟังก์ชัน จะทำได้โดยใส่เครื่องหมาย & หน้าตัวแปรแบบ structure ดังนี้

```
struct      sample{
        char   x[10];
        int    y;
        float  z;
};
```

สามารถส่งตำแหน่งที่อยู่ของตัวแปรใน structure ดังกล่าวไปยังฟังก์ชันได้ ดังนี้

```
f1(sample.x);
f2(&sample.y);
f3(&sample.z);
```

ข้อสังเกต ถ้าเป็นตัวแปรชนิดอะเรย์ไม่ต้องใส่เครื่องหมาย & เพราะเป็นการส่งค่า ตำแหน่งที่อยู่ไปทั้งชุดอยู่แล้ว

Structure (Cont.)

2. ส่งผ่านทั้ง structure ให้กับฟังก์ชัน

จะส่งผ่านในลักษณะของพอยน์เตอร์ไปยัง structure โดยหลักการจะเหมือนกับการส่งผ่านอะเรย์ไปให้ฟังก์ชัน ซึ่งเป็นลักษณะที่เรียกว่า Pass by reference

Pointer

Pointer เป็นตัวแปรชนิดหนึ่งที่ทำหน้าที่เก็บตำแหน่งที่อยู่ (Address) ของตัวแปรที่อยู่ในหน่วยความจำ

<u>การประกาศชนิดของตัวแปรพอยน์เตอร์</u>

รูปแบบ

type *variable-name

type หมายถึง ชนิดของตัวแปร

* หมายถึง เป็นเครื่องหมายที่แสดงว่า ตัวแปรที่ตามหลังเครื่องหมายนี้เป็นตัวแปรพอยน์เตอร์

variable-name เป็นชื่อของตัวแปรที่ต้องการประกาศว่าเป็นชนิดพอยน์เตอร์

Pointer (Cont.)

<u>ตัวอย่าง</u>

char *prt;

หมายความว่า ประกาศว่าตัวแปร prt เป็นตัวแปรพอยน์เตอร์ที่ใช้เก็บตำแหน่งเริ่มต้นที่จะเก็บ character

int *a;

หมายความว่า ประกาศว่าตัวแปร a เป็นตัวแปรพอยน์เตอร์ที่ใช้เก็บตำแหน่งเริ่มต้นที่จะใช้เก็บ integer

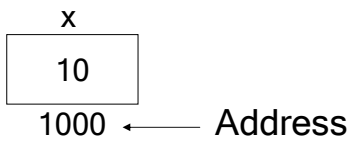
Pointer (Cont.)

<u>เครื่องหมายที่ใช้ทำงานกับตัวแปรพอยน์เตอร์</u>

1. เครื่องหมาย & เป็นเครื่องหมายที่ใช้เมื่อต้องการให้เอาค่าตำแหน่งที่อยู่ของตัวแปรที่เก็บไว้ในหน่วยความจำออกมาใช้

เช่น

x=10



หมายความว่ากำหนดให้ตัวแปร x ซึ่งอยู่ที่ตำแหน่ง 1000 มีค่า 10 เก็บอยู่

Pointer (Cont.)

<u>ตัวอย่าง</u>

```
int  *y , x=10;
y = &x;
```

หมายความว่า ตัวแปร y ซึ่งประกาศเป็นตัวแปรพอยน์เตอร์จะเก็บค่า 1000 ซึ่งเป็นตำแหน่งที่อยู่

<u>ข้อสังเกต</u> ตัวแปรที่มีเครื่องหมาย & นำหน้าจะไม่สามารถนำมาทำการคำนวณได้

เช่น &x = &x + 1; ไม่ได้

Pointer (Cont.)

2. เครื่องหมาย *
   มีการใช้งาน 2 ลักษณะ คือ
   - ใช้ในการประกาศ parameter ว่าเป็นตัวแปรแบบพอยน์เตอร์ เช่น

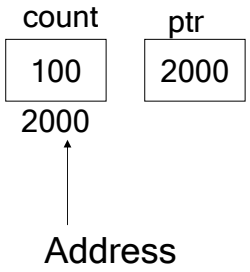
```
void  swap(int *p,int *q)
{
      .....
}
```


Pointer (Cont.)

- ใช้เป็น dereferencing operator จะใช้เมื่อต้องการนำค่าที่อยู่ในตำแหน่งที่ตัวแปรพอยน์เตอร์นั้นชี้อยู่ออกมาแสดง

<u>ตัวอย่าง</u>

```
int  *ptr,count;
count = 100;
ptr = &count;
printf("%d\n",*ptr);
```



Pointer (Cont.)

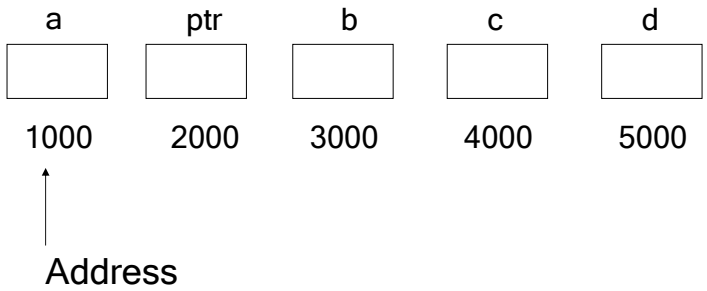
<u>ตัวอย่าง</u>

```
int  a, *ptr, b, c , *d;
a   =25;
ptr = &a;
b = a;
c = *ptr;
d = ptr;
```


Pointer (Cont.)

<u>ลักษณะที่อยู่ในหน่วยความจำ</u>

คำสั่งที่ 1 เตรียมเนื้อที่ให้กับตัวแปรภายในหน่วยความจำ ตามตำแหน่งต่าง ๆ ดังนี้



Pointer (Cont.)

คำสั่งที่ 2 a = 25

a	ptr	b	c	d
25				
1000	2000	3000	4000	5000

คำสั่งที่ 3 ptr = &a;

a	ptr	b	c	d
25	1000			
1000	2000	3000	4000	5000

Pointer (Cont.)

คำสั่งที่ 4 b = a;

a	ptr	b	c	d
25	1000	25		
1000	2000	3000	4000	5000

คำสั่งที่ 5 c = *ptr;

a	ptr	b	c	d
25	1000	25	25	
1000	2000	3000	4000	5000

Pointer (Cont.)

คำสั่งที่ 6 d = ptr;

a	ptr	b	c	d
25	1000	25	25	1000
1000	2000	3000	4000	5000

Pointer (Cont.)

<u>การใช้ตัวแปรพอยน์เตอร์กับอะเรย์</u>

ตัวแปรพอยน์เตอร์จะใช้อ้างถึงค่าที่เก็บไว้ในตัวแปรชุดได้ ดังนี้

<u>ตัวอย่าง</u>
```
char  str[80], *pl;
pl = str;
```

บรรทัดที่ 1 เป็นการประกาศว่า str เป็นตัวแปรชุด ชนิด character 1 มิติ มีขนาดสมาชิก 80 สมาชิกและ pl เป็นตัวแปรพอยน์เตอร์

บรรทัดที่ 2 เป็นการอ้างถึงข้อมูลที่เก็บในตัวแปรชุด str โดยการนำตำแหน่งที่อยู่ของตัวแปร str[0] ซึ่งเป็นสมาชิกตัวแรกไปเก็บไว้ใน ตัวแปรพอยน์เตอร์ pl

Pointer (Cont.)

เหมือนกับใช้คำสั่ง

pl = &str[0];

การอ้างถึงตัวแปรชุด สามารถอ้างถึงโดยการเพิ่มหรือ ลดตัวแปรพอยน์เตอร์ได้

เช่น ต้องการอ้างถึงตัวแปร str[4] ทำได้โดย *(pl+4)

Pointer (Cont.)

ตัวอย่าง

```
#include <stdio.h>
#define N 3
int main(void)  {
        int num[N], *value;
        num[0] = 100;
        num[1] = 200;
        num[2] = 300;
        value = num;
        print("num[0] = %d\n",*(value));
        print("num[1] = %d\n",*(value+1));
        print("num[2] = %d\n",*(value+2));
        return 0;
}
```


แบบฝึกหัด

1. ให้นักศึกษากำหนดค่าของ Array 1 มิติ และ Array 2 มิติ
2. ให้นักศึกษาหาค่าของ A[2], A[6] จากค่า A={2,8,16,24,9,7,3,8}
3. จากค่าของ int a[2][3] = {{6,5,4},{3,2,1}};
   ให้นักศึกษา หาค่าของ a[1][0] และ a[0][2]
4. ให้นักศึกษากำหนด Structure ที่มีค่าของข้อมูลจากน้อย 6 Records
5. ให้นักศึกษาบอกความแตกต่างของการกำหนดตัวชนิด Array กับตัวแปร Pointer ในสภาพของการกำหนดที่อยู่ของข้อมูล