02204171 Structured Programming

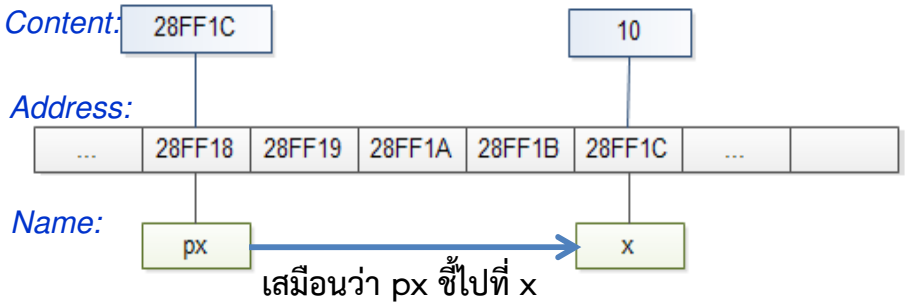
Chapter 14 : Pointer (part 2)

Outline

- Review: Pointer
  - What is a Pointer?
  - Pointer & Array
- Pointer and String
- Pointers and Dynamic Allocation of Memory
- Pointers and Structures

1. Review: Pointer

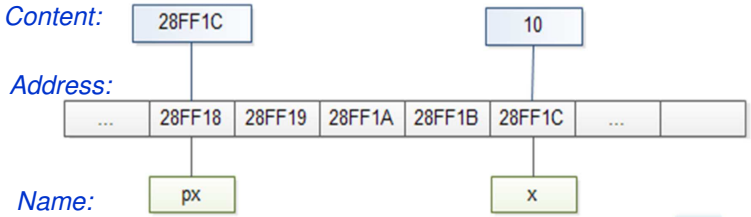
พอยน์เตอร์ (Pointer) เป็นตัวแปรชนิดพิเศษในภาษาซี ทำหน้าที่ **เก็บตำแหน่งในหน่วยความจำ (Address)** ของตัวแปรชนิดอื่นๆ แทนการเก็บข้อมูลเหมือนกันตัวแปรพื้นฐานชนิดอื่นๆ



1. Review: Pointer

The memory address of x is 0028FF1C

```
int *px,x=10;
px=&x;
printf("The memory address of x is %p\n",&x);
printf("The value of the pointer px is %p\n",px);
*px=5;
printf("Direct access, x = %d\n",x);
printf("Indirect access, *px = %d\n",*px);
```



Review: Pointer

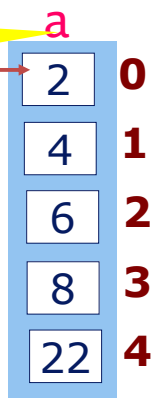
ชื่อของตัวแปรอาเรย์คือพอยน์เตอร์ที่ชี้ไปยังสมาชิกตัวแรก

```
int *p, a[5]={2,4,6,8,22};

p=a;

printf("%d %d\n",*p,a[0]);
printf("%d %d\n",*(p+3),a[3]);
```

การอ้างถึง a จะเหมือนกับอ้าง &a[0]




```
int i;
for(i = 0; i < 5; i++)
   printf("%d %d\n",*(p+i),a[i]);
```

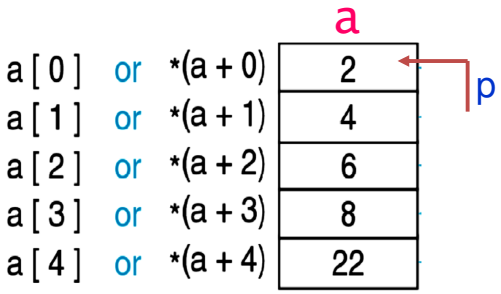

1. Review: Pointer

การอ้างข้อมูลแบบอาเรย์และพอยน์เตอร์

เนื่องจากทุกครั้งที่อ้างถึง a[i] ภาษาซีจะทำหน้าที่แปลงเป็น *(a+i) เพราะฉะนั้นการเขียนในรูปแบบใดก็ให้ผลลัพธ์ในการทำงานเช่นเดียวกัน

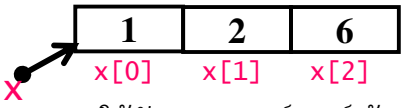
อาเรย์มีความเกี่ยวข้องกับตัวแปรพอยเตอร์อย่างใกล้ชิด

- **การใช้พอยน์เตอร์แทนอาเรย์**
  เช่น การอ้างถึงค่าในอาเรย์โดยใช้ a[i] สามารถใช้ *(a+i)
- **การใช้งานอาเรย์แทนพอยน์เตอร์**
  เช่น การอ้างถึง *(p+i) สามารถเขียนด้วย p[i]



1. Review: Pointer

การส่งผ่านอาเรย์เข้าฟังก์ชัน



การส่งผ่านอาเรย์ เข้าไปในฟังก์ชันสามารถใช้ตัวแปรพอยน์เตอร์เข้ามารับในส่วนของพารามิเตอร์

```
#include <stdio.h>
void inc(int [], int);
int sumArray(int *, int);

int main()
{
   int x[3] = { 1, 2, 6 }, sum;

   inc(x, 3);
   sum = sumArray(x, 3);

   printf("%d\n", sum);
}
```

```
int sumArray(int *pa, int n) {
  int i, sum = 0;
  for(i = 0; i < n; i++)
  sum += pa[ i ];

  return sum;
}
```

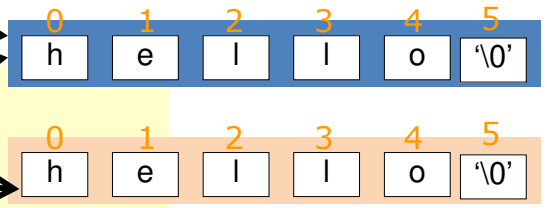
```
void inc(int array[], int size) {
   int i;
   for (i = 0; i < size; i++)
      array[i]++;
}
```


2. Pointer and String

ข้อมูลประเภทข้อความ(String) ในภาษาซี ก็คืออาเรย์ของตัวอักขระ

การทำงานของพอยน์เตอร์ใน String จึงเหมือนกับการทำงานกับอาเรย์ปกติ

ตัวอย่าง 1




```
char s1[]="hello";
char *s2="hello";

char *p;
for (p = s1; *p != '\0'; ++p)
   printf("%c", toupper(*p));

printf("\n%s\n",s2);
printf("%s\n",s1);
```

ตราบเท่าที่ค่าที่พอยน์เตอร์ชี้ไม่ใช่ '\0'

2. Pointer and String

ตัวอย่าง 2

Enter strings … 29 ch…rs: ***C Programming***

```
int main() {
  int i = 0;
  char str[30];
  char *spt;
  spt = str;

  printf("Enter strings not more than 29 characters: ");
  gets(str);

  while(*spt){
     i++;
     spt++;
   }
  printf("%s has %d characters\n",str,i);
}
```

กำหนดให้ พอยน์เตอร์ชี้อาเรย์

0	1	2	3	.....	12	.....	29
C		P	r	o	g	r	a	m	m	i	n	g	\0			

spt

ตราบเท่าที่ค่าที่ พอยน์เตอร์ชี้ไม่ใช่ '\0'

2. Pointer and String

ตัวอย่าง 3

```
#include<stdio.h>
#include<string.h>
int myAtoi(char *snum) {
 int i,j=1,sum=0;
 for(i=strlen(snum)-1; i>=0;i--,j*=10)
   sum+= ((snum[i]-48)*j); // ascii code of 0
  return sum;
}

int main() {
   char s[10]="1234";
   printf("%s\n",s);
   printf("%d\n",myAtoi(s)+5);
}
```

s, snum

0	1	2	3	4	...	9
1	2	3	4	\0		??

3	(52-48) *1
2	(51-48) *10
1	(50-48)*100
0	(50-49)*1000

End Loop

1234

Quick check1

```
int  b = 2;
char  name[64] = "Ariana";
char  *ch_ptr;   int  *i1_ptr,  *i2_ptr;

ch_ptr = name;
 i1_ptr  =  &b;
 i2_ptr  = i1_ptr;
```

จากโปรแกรมข้างต้นจงหาค่าของ

1.1) *i2_ptr = ...
1.2) *i1_ptr + b = ...
1.3) *(i1_ptr + b) = ...
1.4) name[*i2_ptr] = ...
1.5) name[*i1_ptr + *i2_ptr] = ...
1.6) *(ch_ptr + b) = ...

3. Pointers and Dynamic Memory Allocation

การจัดสรรหน่วยความจำแบบไดนามิค (Dynamic Memory Allocation) เป็นการจับ**จองพื้นที่หน่วยความจำ**แบบชั่วคราว**ในขณะที่โปรแกรมทำงาน (Run time)** เพื่อนำมาเก็บข้อมูล

– ประโยชน์ของการจัดสรรหน่วยความจำแบบไดนามิค เพื่อใช้ในการสร้างโครงสร้างข้อมูล (Data Structure) ที่ขนาดสามารถปรับเปลี่ยนได้ (เพิ่มขึ้น/ลดลง) ตอนเขียนโปรแกรม

การใช้ฟังก์ชั่นเหล่านี้ ต้องระบุ stdlib.h

ฟังก์ชันในภาษาซี ที่ใช้การจัดสรรหน่วยความจำแบบไดนามิค:

จองหน่วยความจำ - **malloc(), calloc(), realloc()**

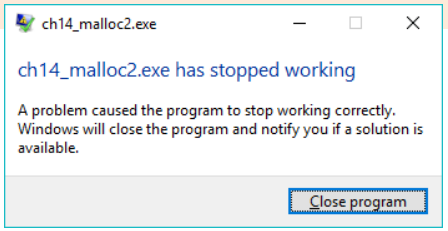
คืนหน่วยความจำ - **free()**

3. Pointers and Dynamic Memory Allocation

- Static vs Dynamic memory allocation

```
char s1[20];
strcpy(s1, "Daenery Targaryen");
printf ("%s\n",s1);
strcpy(s1,"Daenerys Stormborn of the House Targaryen, Khaleesi of the Great Grass Sea,Breaker of Chains, and Mother of Dragons");
printf ("%s\n",s1);
```

การกำหนดค่าใหม่ เกินขนาดของอาเรย์ที่ประกาศไว้



การประกาศอาเรย์แบบนี้ ทำให้เราได้อาเรย์**ขนาดคงที่**ตลอด (Fixed length data structure) นั่นคือ เราไม่สามารถปรับเปลี่ยนขนาดอาเรย์ขณะโปรแกรมทำงานได้

3. Pointers and Dynamic Memory Allocation

- Static vs Dynamic memory allocation

```
char *s2;
```

ใช้ พอยน์เตอร์ ในการประกาศอาเรย์แบบไดนามิค

```
s2 = malloc(sizeof(*s2) * 20);
strcpy(s2, "Daenery Targaryen");
printf ("%s\n",s2);
```

เปลี่ยนขนาดของอาเรย์ที่ประกาศไว้จาก 20 เป็น 200

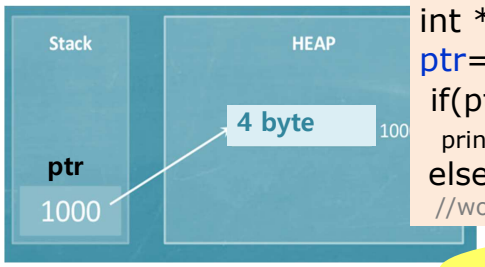
```
s2 = realloc(s2, sizeof(*s2) * 200);
strcpy(s2,"Daenerys Stormborn of the House Targaryen, Khaleesi of the Great Grass Sea,Breaker of Chains, and Mother of Dragons");
printf ("%s\n",s2);

free(s2);
```

```
Daenery Targaryen
Daenerys Stormborn of the ...Mother of Dragons
```


3. Pointers and Dynamic Memory Allocation

- malloc(): memory allocation
  - malloc() ใช้ในการจองหน่วยความจำตามขนาดที่ระบุ (Byte)
  - รูปแบบ(Syntax) `void * malloc(size_t size)`
  - การใช้งาน `ตัวแปรพอยน์เตอร์ = malloc(ขนาดไบต์);`

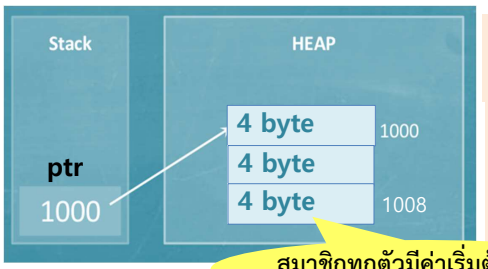



```
int *ptr;
ptr=malloc(sizeof(int));
 if(ptr == NULL)
  printf("Error! memory not allocated.");
 else
  //working with the allocated memory
```

ถ้าไม่สามารถจองหน่วยความจำได้ฟังก์ชันจะคืนค่าเป็น NULL

3. Pointers and Dynamic Memory Allocation

- calloc(): contiguous allocation
  - calloc() ใช้ในการจอง**ชุดหน่วยความจำติดกัน**ตามขนาด (Byte) และจำนวนที่ระบุ
  - รูปแบบ(Syntax) `void *calloc(size_t nitems, size_t size)`
  - การใช้งาน `ตัวแปรพอยน์เตอร์ = calloc(จำนวน,ขนาดไบต์);`




```
int *ptr;
ptr=calloc(3,sizeof(int));
 if(ptr == NULL)
  printf("Error! memory not allocated.");
 else
  //working with the allocated memory
```

สมาชิกทุกตัวมีค่าเริ่มต้นเป็น 0

3. Pointers and Dynamic Memory Allocation

ตัวอย่าง 4

```
#include<stdio.h>
#include<stdlib.h>
int main() {
  int *ptr;
  int i,n;
  printf("Enter size of array: ");
  scanf("%d",&n);
  ptr=calloc(n,sizeof(int));
for(i=0;i<n;i++)
printf("%d ",ptr[i]);
  free(ptr);
}
```

จองหน่วยความจำ 5 ตัวสมาชิกทุกตัว มีค่าเริ่มต้นเป็น 0

```
Enter size of array: 5
0 0 0 0 0
```

```
#include<stdio.h>
#include<stdlib.h>
int main() {
  int *ptr;
  int i,n;
  printf("Enter size of array: ");
  scanf("%d",&n);
  ptr=malloc(n*sizeof(int));
  for(i=0;i<n;i++)
    printf("%d ",ptr[i]);
  free(ptr);
}
```

```
Enter size of array: 3
6829080 6819112 13309243
```


Quick check2

- จงเขียนโปรแกรมนี้ใหม่ โดยให้สร้าง**อาเรย์แบบไดนามิค**

```
#include<stdio.h>
void main()
{
 int i,a[10];
 printf("Enter the number");
 for(i=0;i<10;i++)
   scanf("%d",&a[i]);

}
```


3. Pointers and Dynamic Memory Allocation

- realloc(): re-allocation

—realloc() ใช้ในการปรับเปลี่ยนขนาดหน่วยความจำที่จองด้วยฟังก์ชัน malloc()/calloc() ตามขนาด(Byte) ที่ระบุ

—รูปแบบ(Syntax) `void *realloc(void *ptr, size_t size)`

การปรับขนาดให้ใหญ่ขึ้น ข้อมูลเดิมยังคงอยู่ แต่ปรับขนาดลดลงข้อมูลที่เกินมาจะหายไป

ขนาดใหม่ที่ต้องการ

—การใช้งาน

`ตัวแปรพอยน์เตอร์ = realloc(ตัวแปรพอยน์เตอร์,ขนาดไบต์);`

3. Pointers and Dynamic Memory Allocation

- การคืนหน่วยความจำแบบไดนามิค – free()

หลังจากใช้งานหน่วยความจำที่ได้จองแบบไดนามิคแล้ว ต้องทำการคืนหน่วยความจำที่จองไว้ให้แก่ระบบปฏิบัติการ เพื่อไม่ให้เกิดปัญหาการมีหน่วยความจำไม่เพียงพอที่จะจัดสรรให้โปรแกรมอื่นๆใช้งานได้

—free() ใช้ในการคืนหน่วยความจำที่จองด้วยฟังก์ชัน malloc()/calloc()

—รูปแบบ(Syntax) `void free(void *ptr)`

—การใช้งาน `free(ตัวแปรพอยน์เตอร์);`

3. Pointers and Dynamic Memory Allocation

ตัวอย่าง 5

```
int *ptr,i,n=5;
ptr=calloc(n,sizeof(int));
for(i=0;i<n;i++){
    ptr[i]=2*i;
    printf("%d ",ptr[i]);
}
printf("Enter new size of array: ");
scanf("%d",&n);
ptr=realloc(ptr,n * sizeof(int));
for(i=0;i<n;i++)
   printf("%d ",ptr[i]);
free(ptr);
```

```
0 2 4 6 8
Enter new size of array: 8
```

```
0 2 4 6 8
Enter new size of array: 3
```


Quick check3

- จงพิจารณาโปรแกรมต่อไปนี้ แล้วตอบคำถาม a-c

```
int i,*x;
x = (int *) calloc(10, sizeof(int));
for(i=0;i<10;i++)
   x[i] = i;
x = (int *) realloc(x, 20*sizeof(int));
for(i=10;i<20;i++)
   x[i] = i;
```

a. เขียนคำสั่งจองพื้นที่หน่วยความจำใหม่โดยใช้ malloc()

b. ขนาดของอาเรย์ x เมื่อเริ่มต้น และสิ้นสุดโปรแกรม

c. ค่าที่เก็บในอาเรย์ x เมื่อสิ้นสุดโปรแกรม

4. Pointers and Structures

- สตรัคเจอร์ (Structure) หรือกลุ่มข้อมูลชนิดโครงสร้าง เป็นการกำหนด**ชนิดของข้อมูล (Data type)** ขึ้นมาใหม่
  - โดยนำตัวแปรชนิดพื้นฐาน (simple data) ในภาษาซี เช่น int, char, float มาประกอบกันเป็นโครงสร้างของข้อมูลชนิดใหม่ โดยข้อมูลซึ่งเป็นสมาชิกของโครงสร้างใหม่ อาจมีหลายตัว และเป็น*ชนิดเดียวกันหรือต่างชนิด*กันก็ได้


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


4. Pointers and Structures

- **ประโยชน์ของสตรัคเจอร์**

เพื่อกำหนดหน่วยข้อมูลใหม่ ให้เหมาะสมกับข้อมูลที่ต้องการเก็บ เช่น ข้อมูลของพนักงานซึ่งอาจจะประกอบไปด้วย ชื่อ-นามสกุล,รหัสพนักงาน ,เงินเดือน

การเก็บข้อมูลพนักงานหนึ่งคนจะต้องสร้างตัวแปรขึ้นมา 3 ตัว

```
char   name[30];
int    code;
float  salary;
```

ถ้าพนักงานมีจำนวนมากก็จำเป็นต้องสร้างตัวแปรมากขึ้นตามไปด้วย (ทั้งที่จริงแล้วข้อมูลทั้งหมดมีส่วนเกี่ยวข้องกัน) ซึ่งอาจจะเกิดความสับสนในการเรียกใช้งานตัวแปรเหล่านั้น

```
char   name1[30], name2[30],..., name100[30];
int    code1,code2,...., code100;
float  salary1,salary2,..., salary100;
```

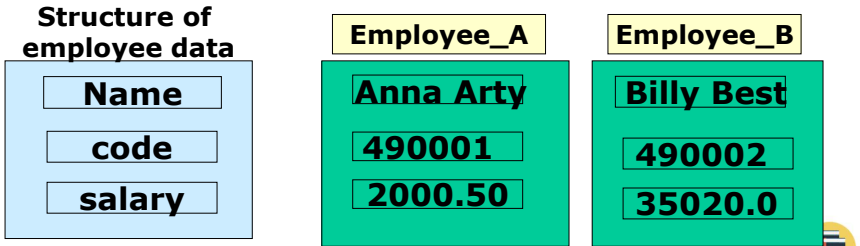
การนำสตรัคเจอร์มาใช้จะช่วยให้การทำงานในลักษณะนี้ง่ายขึ้น

4. Pointers and Structures

การนิยามและการประกาศสตรัคเจอร์
(Structure definition & declaration)

การประกาศตัวแปรสตรัคเจอร์ในภาษาซี แบ่งเป็น 2 ขั้นตอน

1. *Structure definition* การนิยามกลุ่มข้อมูลที่สร้างใหม่ ว่ามีสมาชิกอะไรบ้าง เป็นชนิดใด
2. *Structure declaration* ประกาศ*ตัวแปร*สำหรับกลุ่มข้อมูลที่นิยาม



4. Pointers and Structures

การนิยามและการประกาศสตรัคเจอร์
(Structure definition & declaration)

- *Structure definition* การนิยามกลุ่มข้อมูลที่สร้างใหม่ ว่ามีสมาชิกอะไรบ้าง เป็นชนิดใด เสมือนเป็นการสร้างแบบ (template) ที่จะสามารถนำไปใช้ในการสร้างตัวแปรต่อไป

รูปแบบ (Syntax)

```
struct ชื่อโครงสร้างข้อมูล {
    ชนิดข้อมูล  ชื่อข้อมูลที่หนึ่ง;
    ชนิดข้อมูล  ชื่อข้อมูลที่สอง;
          . . .
    ชนิดข้อมูล  ชื่อข้อมูลที่ n;
};
```

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

โดยทั่วไปแล้วสมาชิกทุกตัวของสตรัคเจอร์ควรจะมีความสัมพันธ์กัน

4. Pointers and Structures

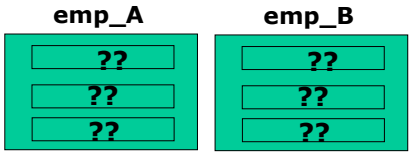
การนิยามและการประกาศสตรัคเจอร์
(Structure definition & declaration)

- *Structure declaration* เมื่อกำหนดกลุ่มข้อมูลเรียบร้อยแล้ว เราสามารถกำหนดให้ตัวแปรใดๆ มีโครงสร้างตามที่กำหนดไว้แล้ว โดยใช้รูปแบบต่อไปนี้

รูปแบบ (Syntax)

```
struct ชื่อโครงสร้างข้อมูล ตัวแปร1 [,ตัวแปร2, ...];
```

```
struct employee emp_A,empB;
```



4. Pointers and Structures

การนิยามและการประกาศสตรัคเจอร์
(Structure definition & declaration)

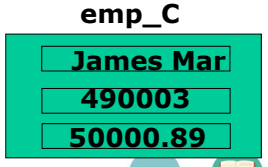
- *Structure declaration* การกำหนดให้ตัวแปรใดๆ มีโครงสร้างตามที่กำหนดไว้แล้ว พร้อมทั้ง**กำหนดค่าเริ่มต้น** โดยใช้รูปแบบต่อไปนี้

รูปแบบ (Syntax)

```
struct ชื่อโครงสร้างข้อมูล ตัวแปร =
  { ค่าของสมาชิกตัวที่ 1,
    ค่าของสมาชิกตัวที่ 2,
    ค่าของสมาชิกตัวสุดท้าย
  };
```

ใช้ comma(,) คั่นเพื่อแยกระหว่างข้อมูลของสมาชิกแต่ละตัวในสตรัคเจอร์

```
struct employee
emp_C ={"James Mar",
    490003,
    50000.89
  };
```



4. Pointers and Structures

- ตัวอย่าง การนิยามและประกาศสตรัคเจอร์ (เพิ่มเติม)

```
struct book {
    char code[6];
    float price;
    int year;
};
```

```
struct book {
    char code[6];
    float price;
    int year;
} book1,book2;
```

```
struct book book1,book2;
```

```
struct book HP7=
  { "HP7JK", 300, 2556};
```

การประกาศตัวแปรพร้อมกับการนิยามสตรัคเจอร์โดยการระบุชื่อตัวแปรต่อท้ายก่อนเครื่องหมาย ;

การประกาศตัวแปรชนิดสตรัค book ชื่อ HP7 พร้อมกับกำหนดค่าเริ่มต้น

Quick check4

1. จงนิยามสตรัคเจอร์ชื่อว่า **date** เพื่อใช้ในการเก็บข้อมูลของวันที่โดยประกอบด้วยสมาชิก 3 ตัวชื่อว่า day, month และ year ซึ่งสมาชิกทั้งสามเป็นจำนวนเต็ม

2. ประกาศตัวแปรชื่อ birthday และ exam_date เป็นชนิดสตรัคเจอร์ชื่อ date และกำนดค่าเริ่มต้นเป็นวันเกิดของนิสิต และวันสอบบรรยายวิชา structured programming

ALLPPT

4. Pointers and Structures

การเข้าถึงสมาชิกในตัวแปรชนิดสตรัคเจอร์
(Accessing element of structure variable)

- การเข้าถึงสมาชิกในตัวแปรชนิดสตรัคเจอร์ ทำได้โดยบอกชื่อตัวแปรสตรัคเจอร์ ตามด้วยจุด "." และต่อด้วยชื่อสมาชิกของสตรัคเจอร์นั้นๆ

รูปแบบ (Syntax)

ตัวแปรสตรัคเจอร์.ชื่อสมาชิกของสตรัคเจอร์;

ตัวอย่าง

```
printf("%d/",birthday.day)
printf("%d/",birthday.month)
printf("%d\n",birthday.year)
```


4. Pointers and Structures

- ตัวอย่าง 6

```
int main() {
  struct subject {
    char name[30];
    int credit;
  } s1,s2;
  printf("Enter subject name: ");
  gets(s1.name);
  printf("Enter credit: ");
  scanf("%d",&s1.credit);

  strcpy(s2.name, "Math II");
  s2.credit=4;
}
```

```
Enter subject name: Physics I
Enter credit: 3
```

s1

name	
credit	

s2

name	
credit	

4. Pointers and Structures

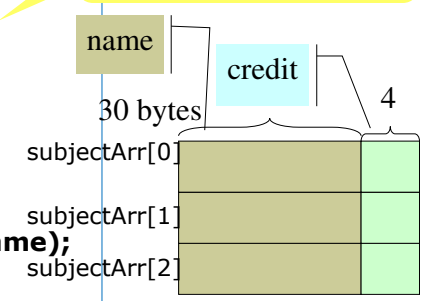
- ตัวอย่าง 7

```
//using structure and variable from Ex 6
struct subject subjectArr[3];
int i;
subjectArr[0] = s1;
subjectArr[1] = s2;

strcpy(subjectArr[2].name,s2.name);
subjectArr[2].credit = s2.credit;

for (i=0;i<3;i++)
printf("%s,%d\n", subjectArr[i].name
   , subjectArr[i].credit);
```

ได้ตัวแปรอาเรย์ที่มีสมาชิก 3 ตัว (เพื่อเก็บข้อมูลวิชา 3 วิชา) สมาชิกอาเรย์ทั้งหมดเป็นชนิด subject



Quick check 5: จงบอกผลลัพธ์ที่ได้ทางหน้าจอ

```
#include <stdio.h>

struct subject {
    char name[30];
    int credit;
};
int main()
{
    struct subject mySubjects[5] = { };

    for (int i = 0; i < 7; i++){
        if (strcmp(mySubjects[i].name,"") == 0)
            printf("empty:");
        printf("%s %d\n",mySubjects[i].name, mySubjects[i].credit);
    }

    return 0;
}
```


Quick check 6

- ประกาศตัวแปร mySubjects ที่เป็น array ชนิด subject แทนวิชาที่ลงทะเบียน สมมติให้ลงทะเบียนไม่เกิน 7 ตัว

- เติมส่วนของโปรแกรมเพื่อรับวิชาที่นิสิตลงทะเบียน โดยหยุดเมื่อป้อน # ที่ชื่อวิชา และแสดงวิชาที่ลงทะเบียน

ตัวอย่างผลการทำงานเป็นดังนี้

```
Enter subject 1: Structured Programming
Structured Programming's credit: 3
Enter subject 2: Physics I
Physics I's credit: 3
Enter subject 3: Lab Phy
Lab Phy's credit: 1
Enter subject 4: Math I
Math I's credit: 3
Enter subject 5: #
----------------------
3       Structured Programming
3       Physics I
1       Lab Phy
3       Math I
```


Quick check 6 (ต่อ)

```
int n = 0; //number of registered subjects
 for (int i = 0; i < 7; i++) {
     printf("Enter subject %d: ",i+1);
     gets(_____________________);
     if (_____________________________________)
     {
         _____________________
         _____________________
     }
     ___________________________________
     scanf("%d", _____________________);
     fflush(stdin);
 }
 printf("----------------------\n");

 for(int i = 0; i<_____ ; i++) {
     printf("%d\t%s\n", ____________________________________________);
 }
```

Summary

- การใช้ Pointer กับ String ก็เหมือนกับ Pointer กับอาเรย์
- Dynamic memory allocation
  - จองหน่วยความจำ - **malloc(), calloc(), realloc()**
  - คืนหน่วยความจำ - **free()**
- Struct: สร้างชนิดข้อมูลชนิดใหม่แบบซับซ้อน โดยเราสามารถกำหนดองค์ประกอบของโครงสร้างข้อมูลได้เอง
  - ประกาศ: **struct** <struct_name> { *[*<type> <var>;*]*+ };
  - ประกาศตัวแปร: **struct** *<struct_name>* <variable_name>;
  - เข้าถึงองค์ประกอบภายในด้วย dot operator (.)