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

## Handout 11

**<u>หัวข้อ:</u>**

**Data structures (Chapter 19 textbook)**

- ตัวแปรแบบ structures ในภาษา C
- Arrays of structures และ Pointers and Structures
- Dynamic Memory Allocation: Dynamically sized array of structures
- Linked lists

**<u>เกริ่นนำ</u>**

1. ทบทวน: ชนิดข้อมูลพื้นฐาน (basic data type) ที่เราได้เรียนกันมา คือ
    - `int` เก็บข้อมูลที่เป็นจำนวนเต็ม (integer) เช่น `int score;`
    - `float/double` เก็บข้อมูลที่เป็นจำนวนจริง (real number)
    - `char` เก็บข้อมูลที่เป็นตัวอักษร (character)
2. ทบทวน: เราได้เรียนชนิดข้อมูลที่ขยายจากชนิดข้อมูลพื้นฐาน (derived data type) ทั้งหมด 2 แบบได้แก่
    - Arrays: เก็บข้อมูลหลาย ๆตัวที่มีชนิดข้อมูลเดียวกันไว้ด้วยกัน เช่น `int student_scores[100];`
    - Pointers: เก็บแอดเดรสในหน่วยความจำของตัวแปร เช่น `int *ptr;`

ในบทนี้เราจะเรียน derived data type แบบที่เรียกว่า structure หรือตัวแปรแบบโครงสร้างซึ่งจะอนุญาตให้มีการเก็บข้อมูลหลาย ๆตัวที่มีชนิดข้อมูลต่างกันไว้ด้วยกัน

**<u>ตัวแปรแบบ structures ในภาษา C</u>**

1. Why? ทำไมต้องใช้ตัวแปรแบบ structure

- ลองพิจารณาปัญหาที่เราต้องการการนิยามตัวแปรเพื่อเก็บข้อมูลผลการเรียนวิชา CP111 ของนิสิตในห้องนี้สมมติมีทั้งหมด N คน โดยที่ N=100
- นิสิต 1 คนเก็บข้อมูล ชื่อ รหัสนิสิต และเกรดวิชา CP111 ที่ได้

```
char name[81];      // สมมติว่าชื่อของนิสิตแต่ละคนมีตัวอักษรไม่เกิน 80 ตัว
int ID;
double grade;       // สมมติว่าเกรดที่เป็นไปได้คือ 4.0 (A), 3.5 (B+), 3.0 (B), …, 0.0 (E)
```

- ถ้าต้องเก็บข้อมูลของนิสิตทั้งหมด N คน วิธีที่เป็นได้คือใช้อาร์เรย์ 3 ตัว

```
char name[N][81];
int ID[N];
double grade[N];
```

โดยที่ `name[i], ID[i], grade[i]` จะแทนข้อมูลของนิสิตที่ array index = i

จะเห็นได้ว่าวิธีนี้จะทำให้การเขียนโปรแกรมไม่ยืดหยุ่น เช่นในกรณีที่เราจะเขียนฟังก์ชันที่รับอาร์กิวเมนต์เป็นข้อมูลของนิสิตทั้ง N คน เราจะต้องส่งอาร์กิวเมนต์ 3 ตัว ซึ่งถ้าเรามีข้อมูลของนิสิต 1 คนเพิ่มขึ้น เช่น คะแนนสอบ midterm 1, midterm 2 และ final จะต้องใช้อาร์เรย์เพิ่มอีก 3 อาร์เรย์

เราจะเรียนรู้การใช้ตัวแปรแบบ structure เพื่อมาเก็บข้อมูลนี้ ซึ่งจะทำให้การเขียนโค้ดกระชับมากขึ้น

2. รูปแบบ (syntax) ในการประกาศตัวแปรแบบ structure ทำได้มากกว่า 1 วิธี แต่ในวิชานี้เราจะใช้การประกาศผ่าน typedef โดยต้องเขียนโปรแกรมทั้งหมด 2 จุด ดังต่อไปนี้

(i) จุดที่ 1 ใช้ typedef นิยามโครงสร้างของตัวแปรแบบ structure ที่ต้นโปรแกรม ก่อน main() ในรูปแบบ

```
typedef struct {
      <type> <member1>;
      <type> <member2>;
      …
} TYPE_NAME;
```

(ii) จุดที่ 2 ประกาศตัวแปรแบบ structure ให้มีชนิดข้อมูลข้างบนในฟังก์ชันใดๆ

```
TYPE_NAME <var_name>;
```

ตัวอย่าง

```
#include <stdio.h>

typedef struct {
    char name[81];
    int ID;
    double grade;
} S_INFO;

int main()
{
  S_INFO st1;

    ...
```

3. รูปแบบ (syntax) การอ้างถึงแต่ละสมาชิก (member) ในตัวแปรแบบ structure ที่ประกาศขึ้น

```
<var_name>.<memeber1>
```

ตัวอย่าง (ต่อจากข้างบน)

```
st1.name[0]='B';
st1.name[1]='o';
st1.name[2]='b';
st1.name[3]=0;
gets(st1.name);
st1.ID=5500009999;
st1.grade=4.0; // (A)
```

4. ข้อควรระวังเกี่ยวกับการใช้ตัวแปรแบบ structure

- สมมติมีการประกาศตัวแปรแบบ structure ชนิด `S_INFO` 2 ตัวดังนี้ => `S_INFO s1,s2;`
- การเขียนโค้ด

```
s1+s2
```

จะไม่มีความหมายและจะไม่ถูกต้องตาม syntax เพราะ operator '+' ไม่ได้ถูกนิยามกับตัวแปรแบบ structure

- แต่ถ้าเขียน

```
s1.ID+s2.ID
```

จะถูกต้องตาม syntax แต่ไม่ make sense ที่จะเอา ID ของนิสิต 2 คนมาบวกกัน

- `if(s1.grade > s2.grade)`

จะถูกต้องตาม syntax และ make sense ที่จะทำการเปรียบเทียบ grade ของนิสิต

- อนุญาตให้ใช้ assignment operator ดังเช่น

```
s1=s2;
```

ซึ่งจะทำให้ `s1.name, s1.ID, s1.grade` เท่ากับค่าของฟิลด์ที่ตรงกันใน `s2` (จะขอเรียกแต่ละสมาชิกใน structure ว่า ฟิลด์)

**Arrays of structures**

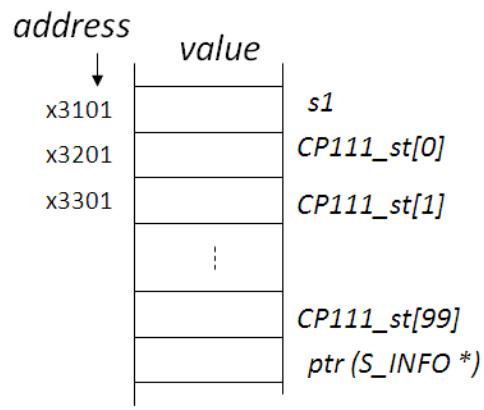
เราสามารถประกาศตัวแปรแบบอาร์เรย์ที่มีชนิดข้อมูลเป็นแบบ structure และการอ้างถึงแต่ละ element ในอาร์เรย์ได้เช่นเดียวกับอาร์เรย์ทั่วๆไป ดังตัวอย่างต่อไปนี้

```
#define N 100
int main()
{
S_INFO CP111_st[N];
       st[0].name[0]='B';
       st[0].name[1]='o';
       st[0].name[2]='b';
       st[0].name[3]=0;
       st[0].ID=5500009999;
       st[0].grade=4.0;
       gets(st[1].name);
       scanf("%d",&st[1].ID);
       st[1].grade=3.0;
       ...
```

**Pointers and Structures**

1. เราสามารถประกาศตัวแปรแบบ pointer ที่ชี้ไปยังตัวแปรแบบ structure ดังตัวอย่างต่อไปนี้

```
#define N 100
int main()
{
S_INFO  s1;
S_INFO  *ptr; //pointer to S_INFO
S_INFO CP111_st[N];
             ptr=&s1;
             ptr=&CP111_st[1];
             ptr=CP111_st;
```




2. เราสามารถอ้างถึงแต่ละฟิลด์ในตัวแปร structure ผ่านตัวแปรแบบ pointer

Syntax:

```
(*<pointer_name>).<member_name>
```

หรือ

```
<pointer_name>-><member_name>
```

Example:

| | |
|---|---|
| `(*ptr).name[0]='B';`<br>`(*ptr).name[1]='o';`<br>`(*ptr).name[2]='b';`<br>`(*ptr).name[3]=0;`<br>`(*ptr).ID=5500009999;`<br>`(*ptr).grade=3.5;` | `ptr->name[0]='B';`<br>`ptr->name[1]='o';`<br>`ptr->name[2]='b';`<br>`ptr->name[3]=0;`<br>`ptr->ID=5500009999;`<br>`ptr->grade=3.5;` |

**โจทย์ตัวอย่าง**

1. (ข้อสอบเก่าปี 2/2551) จงเขียนฟังก์ชันสำหรับการคูณจำนวนเชิงซ้อน (complex number) 2 จำนวน โดยให้นิยามชนิดของข้อมูลแบบ structure ที่ชื่อ `COMPLEX` ซึ่งประกอบไปด้วยฟิลด์ย่อยชื่อ `real` และ `imag` ที่มีชนิดข้อมูล `double` ในการเขียนโปรแกรม และฟังก์ชันที่เขียนขึ้นจะรับอาร์กิวเมนต์เป็นจำนวนเชิงซ้อน 2 ตัวที่จะนำมาคูณกันซึ่งจะมีชนิดข้อมูล

เป็น structure ชนิด COMPLEX ที่ได้กล่าวไปแล้ว และฟังก์ชันนี้จะคืนค่าเป็นจำนวนเชิงซ้อน (ชนิดข้อมูลเป็น structure ชนิด COMPLEX) ที่เป็นค่าผลคูณของจำนวนเชิงซ้อนทั้งสองตัว กลับไปยัง calling function

นอกจากนี้ในฟังก์ชัน main() ให้เขียนโปรแกรมทำการรับค่าจำนวนเชิงซ้อน 2 ตัวจากผู้ใช้ลงมาเก็บในตัวแปร structure ชนิด COMPLEX ที่เหมาะสม แล้วเรียกฟังก์ชัน C_Mult() โดยส่งตัวแปรทั้งสองเป็นอาร์กิวเมนต์ของฟังก์ชันและใช้ตัวแปร structure ชนิด COMPLEX อีกตัวหนึ่งมารับค่าที่คืนมาจากฟังก์ชัน หลังจากนั้นให้แสดงผลลัพธ์ค่าจำนวนเชิงซ้อนที่คืนมาออกสู่หน้าจอดังตัวอย่างการทำงานของโปรแกรมข้างล่าง

**<u>หมายเหตุ</u>**: สูตรการคูณจำนวนเชิงซ้อน $(a_R+ia_I)*(b_R+ib_I) = (a_Rb_R-a_Ib_I)+ i(a_R b_I+a_Ib_R)$

```
#include <stdio.h>
/*  นิยาม Structure ชนิด COMPLEX */




COMPLEX C_Mult(COMPLEX, COMPLEX);

void main()
{
COMPLEX z1,z2,z3;

 scanf("%lf %lf",                    );
 scanf("%lf %lf",                    );
            -- continue --
```

```
 z3= C_Mult(z1,z2);

 printf("%d %d\n",                    );

}
COMPLEX C_Mult(COMPLEX a, COMPLEX b)
{




}
```

2. (ดัดแปลงจากข้อสอบเก่าปี 2/2551) กำหนดให้เลขเศษส่วน (Rational number) หนึ่งๆ (เช่น 11/10, 12/13 เป็นต้น) ถูกเก็บอยู่ในตัวแปรแบบโครงสร้างชนิด RATIONAL ดังนิยามข้างล่าง โดย a แทนเศษ และ b แทนส่วน (เลขเศษส่วน 11/10 จะมีค่า a เท่ากับ 11 และ b เท่ากับ 10)

จงเขียนโค้ดในฟังก์ชัน R_ADD เพื่อทำการคำนวณหาและคืนค่าเลขเศษส่วนที่ได้จากการนำเลขเศษส่วน X มาบวกด้วยเลขเศษส่วน Y (ถ้า X แทน 11/10 และ Y แทน 12/13 ค่าที่ได้จากฟังก์ชันคือค่า (143+120)/130

```
#include <stdio.h>
typedef struct {
   int a;
   int b;
} RATIONAL;

RATIONAL R_ADD(RATIONAL, RATIONAL);
int main()
{
   RATIONAL r1,r2,r3;




}
```

```
RATIONAL R_ADD(RATIONAL X, RATIONAL Y)
{




}
```

3. จงเขียนโปรแกรมเพื่ออ่านข้อมูลผลการเรียนวิชา CP111 ของนิสิตชั้นเรียนปี 1 เอกคอม ซึ่งจะถูกป้อนมาจากคีย์บอร์ด หลังจากนั้นให้บันทึกข้อมูลลงใน text file ที่เก็บอยู่บน harddisk

```
#include <stdio.h>

#define N_Students 100

typedef struct {
     char name[81];
     int ID;
     double grade;
} S_INFO;

void EnterStudents(S_INFO *,int);

void SaveToFile(S_INFO *,int, char *);

void main()
{
S_INFO CP111_st[N_Students];

 EnterStudents(CP111_st, N_Students);
 SaveToFile(CP111_st, N_Students,"CP111.dat");


}
```

```
void EnterStudents(S_INFO *s,int N)
{ int i;









}
void SaveToFile(S_INFO *s,int N,char *filename)
{









}
```

**<u>Dynamic Memory Allocation: Dynamically sized array of structures</u>**

นอกจกานี้เราสามารถประยุกต์แนวคิดเรื่อง Dynamically sized arrays กับตัวแปรแบบ structure เพื่อสร้าง array of structures ที่ขนาดจะรู้เมื่อ run time โดยการใช้ฟังก์ชัน malloc ในการจองเนื้อที่หน่วยความจำ ดังตัวอย่างต่อไปนี้

```
void main()
{
S_INFO *CP111_stu;
int n_stu;

       scanf("%d",&n_stu);


       ...............................................................................
       if(CP111_stu ==NULL)
        {
          printf("can not allocate memory\n");
          return;
        }
       EnterStudents(CP111_stu, n_stu);

       SaveToFile(CP111_stu, n_stu,"CP111.dat");


       ...............................................................................
}
```

**<u>โจทย์ตัวอย่าง (ต่อ)</u>**

4. (ข้อสอบ Lab 2/2553) กำหนดให้ข้อมูลนำเข้าเป็นข้อมูลของนักเรียนที่ลงทะเบียนเรียนในวิชาหนึ่งๆ โดยที่บรรทัดแรกคือจำนวนนักเรียนทั้งหมดสมมติว่าเท่ากับ N และในแต่ละบรรทัดของบรรทัดที่ 2 ถึงบรรทัดที่ N+1 คือข้อมูลของนักเรียนแต่ละคนซึ่งประกอบไปด้วย *<u>รหัส ชื่อ เพศ คณะ ชั้นปี และ คะแนนสอบ</u>* โดยที่รหัส, ชั้นปี และคะแนนสอบ คือตัวเลขจำนวนเต็ม, ชื่อและคณะคือสตริงที่ความยาวไม่เกิน 30 ตัวอักษร, เพศคือตัวอักษร F (เพศหญิง) หรือ M (เพศชาย)

จงเขียนโปรแกรมเพื่อค้นหาและแสดงข้อมูลของ<u>นักเรียนผู้ชาย ที่อยู่ชั้นปีที่ 1 ที่มีคะแนนสอบมากกว่าหรือเท่ากับคะแนนเฉลี่ยของคะแนนสอบของนักเรียนทุกคน</u> โดยให้แสดงผลลัพธ์เป็นคะแนนเฉลี่ย (จุดทศนิยม 2 ตำแหน่ง) ในบรรทัดแรกและแสดงข้อมูลนักเรียนแต่ละคนที่ค้นเจอที่ละบรรทัด ตามลำดับที่ข้อมูลถูกป้อนเข้ามา โดยสมมติว่าในชุดของข้อมูลทดสอบจะมีคำตอบอย่างน้อย 1 คนเสมอ

ตัวอย่างการทำงานของโปรแกรม

| ข้อมูลนำเข้า (Input) | ข้อมูลส่งออก (Output) |
|---|---|
| 10<br>10101 John M SCI 2 60<br>10009 Sarah F ECENG 2 50<br>10001 Abby M MED 1 100<br>10084 Onny M COMSCI 2 90<br>11101 Jane M ECON 1 50<br>12009 Soe F DENT 2 80<br>12001 Tab M MUSIC 1 80<br>13084 Anny F BUSINESS 3 50<br>13184 Peter M BUSINESS 1 40<br>13284 Bob M HUMANSCI 2 40 | 64.00<br>10001 Abby M MED 1 100<br>12001 Tab M MUSIC 1 80 |

5. (ดัดแปลงจากข้อสอบ Final 2/2553) กำหนดให้ข้อมูลผลการเรียนวิชาหนึ่งๆ แทนด้วยตัวแปรแบบโครงสร้างชนิด `course_t` ที่นิยามดังข้างล่าง โดยที่ code แทนรหัสวิชา, units แทนจำนวนหน่วยกิตของวิชานั้น และ grade แทนผลการเรียนที่สอบได้ โดยที่ 4, 3.5, 3.0, 2.5, 2.0, 1.5, 1.0 และ 0 แทนคะแนนเกรด A, B+, B, C+, C, D+, D และ E ตามลำดับ

จงเขียนโค้ดในฟังก์ชัน `calculate_GPA` ซึ่งจะคำนวณและคืนค่า GPA ของนิสิตคนหนึ่งที่ลงเรียนในรายวิชาที่ลงทะเบียนเรียนทั้งหมดซึ่งกำหนดโดยอาร์กิวเมนต์ courses ซึ่งจะเป็นแอดเดรสของ element ตัวแรกใน array of structure course_t ที่มีขนาดระบุโดยอาร์กิวเมนต์ `N_courses`

```
typedef struct {
  int code;
  int credits;
  double grade;
} course_t;
```

| | code | credits | Grade |
|---|---|---|---|
| courses[0] | 101 | 4 | 1.5 |
| courses[1] | 102 | 2 | 0 |
| courses[2] | 111 | 3 | 2.0 |
| courses[3] | 121 | 1 | 4.0 |

```
double calculate_GPA (course_t *courses, int N_courses)
{

}
```