Computer Programming I:
การเขียนโปรแกรมคอมพิวเตอร์ I

สตรัค (Struct)

อ.ดร.ปัญญนัท อ้นพงษ์

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

aonpong_p@su.ac.th

Outline

- **สตรัค และประโยชน์ของสตรัค**
- การประกาศสตรัค
- typedef
- การเข้าถึงข้อมูลในสตรัค
- การรับข้อมูลและแสดงผลในสตรัค
- สตรัคในสตรัค
- อาเรย์ของสตรัค

สตรัคและประโยชน์ของสตรัค

- สตรัคเป็นโครงสร้างข้อมูลที่รวบรวมข้อมูลหลายอย่างไว้ด้วยกัน
  - ต่างจากอาเรย์ตรงที่อาเรย์จะต้องเก็บเฉพาะข้อมูลประเภทเดียวกันเท่านั้น
  - สตรัคเก็บข้อมูลต่างประเภทกันได้
- กล่าวได้ว่าสตรัคเป็นการสร้างชนิดข้อมูลชนิดใหม่ขึ้นมา
- ประโยชน์คือ ทำให้เรารวมข้อมูลหลายตัวที่มีความสัมพันธ์กันในฐานะองค์ประกอบย่อยของข้อมูลมาอยู่ด้วยกัน

สตรัคและประโยชน์ของสตรัค

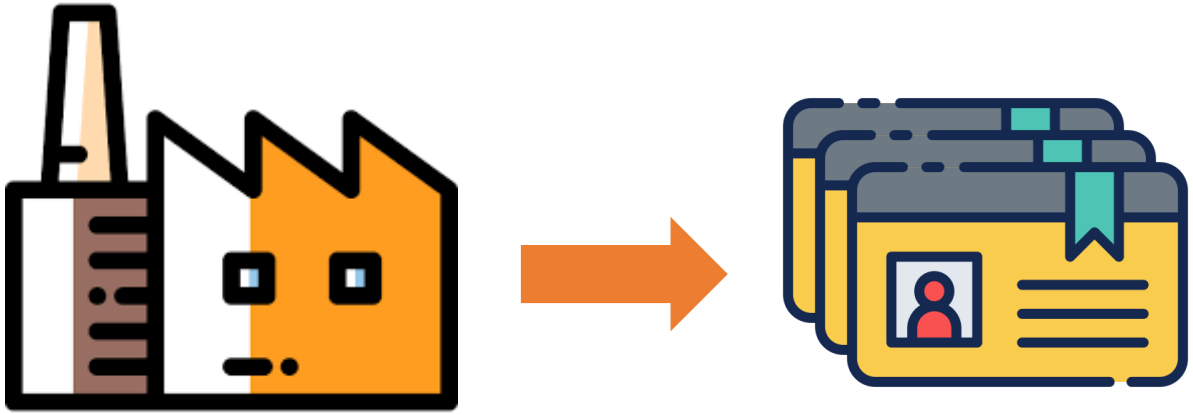
- แนวคิดคล้ายกับบัตรนักศึกษา บัตรประจำตัวประชาชน หรือการ์ดข้อมูลต่างๆ
  - ข้อมูลของสิ่งเดียวกัน ชิ้นเดียวกัน จะถูกเก็บไว้ด้วยกัน
  - เช่น ชื่อ-สกุล, อายุ, สัญชาติ, ที่อยู่ ก็จะเก็บไว้ในการ์ดใบเดียวกันได้ เราก็สามารถดูได้ว่าคนนี้ ชื่อนี้ สัญชาติอะไร ที่อยู่อยู่ที่ไหน เป็นต้น
  - บัตรที่มีรูปแบบโครงสร้างเดียวกันสามารถถูกสร้างได้หลายครั้ง

Outline

- สตรัค และประโยชน์ของสตรัค
- **การประกาศสตรัค**
- typedef
- การเข้าถึงข้อมูลในสตรัค
- การรับข้อมูลและแสดงผลในสตรัค
- สตรัคในสตรัค
- อาเรย์ของสตรัค

การประกาศสตรัค

- ก่อนจะเริ่มการประกาศสตรัค เรามาทำความรู้จักกับสตรัคให้มากขึ้นก่อน
- ตอนนี้เราจะแบ่งสตรัคในภาษาซีออกเป็นสองส่วน เพื่อให้อธิบายง่ายขึ้น



- ส่วนแรกคือส่วนที่ใช้สร้างกลุ่มข้อมูล เป็นการระบุแพทเทิร์นของสตรัค จากนี้จะเรียกส่วนนี้ว่า “โรงงาน” (สตรัคแม่แบบ)
- ส่วนที่สองคือส่วนที่ได้จากโรงงาน ซึ่งก็คือสินค้า เช่น ถ้าเป็นโรงงานทำบัตร สินค้าก็คือบัตร

การประกาศสตรัค

- โรงงานทำบัตร และบัตร มีหน้าที่ไม่เหมือนกัน
- โรงงานทำบัตร มีหน้าที่คือการสร้างบัตร โดยจะสร้างกี่ใบก็ได้
- แต่สมมติถ้าเราจะขึ้นตึก แล้วยามขอดูบัตร เราไม่สามารถเอาโรงงานสร้างบัตรไปให้ยามดูได้ และถึงยามจะได้ดู มันก็ไม่ได้บอกอะไร
- ในทางตรงข้าม บัตรสามารถให้ข้อมูลกับยามได้ (ข้อมูลในบัตรแต่ละใบก็ไม่เหมือนกัน แต่ถ้ามาจากโรงงานเดียวกันจะมีแบบฟอร์มเดียวกัน)
- แต่บัตร ก็ไม่สามารถสร้างบัตรด้วยกันได้ ดังนั้น แม้สองอย่างนี้จะเกี่ยวข้องกัน แต่หน้าที่ของมันคือคนละอย่าง ให้นักศึกษาแยกทั้งสองส่วนนี้จากกันให้ขาด
- เราต้องเขียนโปรแกรม ทั้งส่วนของโรงงาน และส่วนของบัตร

การประกาศสตรัค : ส่วนของโรงงาน

- เราใช้ keyword ว่า struct นำหน้า ตามด้วยชื่อสตรัค (ชื่อโรงงาน) เช่น

```
struct employee {

        ...

};
```

← สังเกตว่ามีเครื่องหมาย ; ด้วย

- จากนั้น เราก็ใส่รายการข้อมูลที่ต้องการลงไป (อยากให้ในบัตรมีข้อมูลอะไรบ้าง)

```
struct employee {
    char name[16];
    char surname[31];
    char address[151];
    float salary;
};
```

การประกาศสตรัค : ส่วนของโรงงาน

- การประกาศสตรัคส่วนของโรงงานนี้จะประกาศในหรือนอก main ก็ได้ แต่ต้องประกาศก่อนนำไปใช้
- ถ้าประกาศนอก main จะเป็น global ที่สามารถนำไปใช้ได้ในทุกฟังก์ชัน
- ถ้าประกาศใน main จะใช้ได้แค่ในฟังก์ชัน main เท่านั้น
- ในเอกสารนี้ ตัวอย่างส่วนใหญ่จะประกาศไว้ข้างนอก จะได้ไม่สับสนระหว่างโรงงานกับตัวบัตร

```c
#include <stdio.h>

struct employee {
        char name[16];
        char surname[31];
        char address[151];
        float salary;
    };

void main() {
    //...
}
```

การประกาศสตรัค : ส่วนของโรงงาน

- ตอนนี้โรงงานของเราก็รู้ว่าแพทเทิร์นของบัตรจะเป็นรูปแบบนี้

```
Name :
Surname :
Address :
Salary :
```

- โรงงานแค่ทำแพทเทิร์นแบบฟอร์มให้ตามสเปคเฉย ๆ ไม่ได้กรอกข้อมูลให้ เราต้องนำแบบฟอร์มนี้ไปกรอกข้อมูลเอง (จะกรอกด้วยมือหรือ scanf ก็แล้วแต่)

การประกาศสตรัค : ส่วนของโรงงาน

ตัวอย่างโจทย์ (1) ให้นักศึกษาเขียน struct แม่แบบให้พร้อมสำหรับบรรจุข้อมูลต่อไปนี้

1. ชื่อ
2. สกุล
3. รหัสนักศึกษา
4. คะแนนสอบควิซ
5. คะแนนสอบไฟนอล

การประกาศสตรัค : ส่วนของโรงงาน

ตัวอย่างโจทย์ (1) ให้นักศึกษาเขียน struct แม่แบบให้พร้อมสำหรับบรรจุข้อมูลต่อไปนี้

1. ชื่อ
2. สกุล
3. รหัสนักศึกษา
4. คะแนนสอบควิซ
5. คะแนนสอบไฟนอล

```
struct student {
    char name[50];
    char surname[50];
    int student_id;
    double quiz_score;
    double final_score;
};
```

การประกาศสตรัค : ส่วนของบัตร

- ประกาศบัตร 1 ใบ (การสั่งให้โรงงานผลิตบัตรออกมา 1 ใบ)
  - ในฟังก์ชัน main เราสามารถสั่งผลิตบัตร 1 ใบได้ดังนี้
  - พิมพ์ keyword ว่า struct ตามด้วยชื่อโรงงาน แล้วตามด้วยชื่อบัตร เช่น

```
struct employee emp1;
```

  - แบบนี้ เราจะได้บัตรที่มีแพทเทิร์นฟอร์มเหมือนที่ประกาศไว้ในสตรัค employee (รูปร่างบัตรเหมือนในหน้า 10) บันทึกชื่อ-สกุล ที่อยู่ เงินเดือนของพนักงานได้ 1 คน
  - เราสามารถใส่ค่าอะไรก็ได้ลงในแบบฟอร์ม ขอแค่ชนิดข้อมูลรองรับการกำหนดค่าก็พอ

การประกาศสตรัค : ส่วนของบัตร

- ประกาศบัตร 2 ใบ (การสั่งให้โรงงานผลิตบัตรออกมา 2 ใบ)
  - ในฟังก์ชัน main เราสามารถสั่งผลิตบัตร 2 ใบได้ดังนี้
  - พิมพ์ keyword ว่า struct ตามด้วยชื่อโรงงาน แล้วตามด้วยชื่อบัตรกี่ใบก็ได้ ขั้นด้วยลูกน้ำ เช่น

```
struct employee emp1, emp2;
```

  - แบบนี้ เราจะได้บัตรที่มีแพทเทิร์นฟอร์มเหมือนที่ประกาศไว้ในสตรัค employee (รูปร่างบัตรเหมือนในหน้า 10) แต่ได้มา 2 ใบ บันทึกข้อมูลชื่อ-สกุล ที่อยู่ เงินเดือนของพนักงานได้ 2 คน

การประกาศสตรัค : ส่วนของบัตร

- ประกาศบัตร 30 ใบ (การสั่งผลิตบัตร; แบบอาเรย์)
  - ในฟังก์ชัน main เราสามารถสั่งผลิตบัตร 30 ใบได้ดังนี้
  - พิมพ์ keyword ว่า struct ตามด้วยชื่อโรงงาน แล้วตามด้วยชื่อบัตร [จำนวนใบ] เช่น

```
struct employee emp[30];
```

  - แบบนี้ เราจะได้บัตรที่มีแพทเทิร์นฟอร์มเหมือนที่ประกาศไว้ในสตรัค employee (รูปร่างบัตรเหมือนในหน้า 10) แต่ได้มา 30 ใบ บันทึกข้อมูลชื่อ-สกุล ที่อยู่ เงินเดือนของพนักงานได้ 30 คน

การประกาศสตรัค : ส่วนของโรงงาน

ตัวอย่างโจทย์ (2) จากตัวอย่างโจทย์ (1) ให้นักศึกษาประกาศสตรัคสำหรับเก็บข้อมูลของนักศึกษา 1000 คน

*hint : ถ้าโจทย์บอกแบบนี้ต้องเขียนทั้งสองส่วน แต่เราเขียนส่วนโรงงานไปแล้ว ลองเขียนแค่ส่วนของบัตรก็พอ

การประกาศสตรัค : ส่วนของโรงงาน

ตัวอย่างโจทย์ (2) จากตัวอย่างโจทย์ (1) ให้นักศึกษาประกาศสตรัคสำหรับเก็บข้อมูลของนักศึกษา 1000 คน

```
struct student {
    char name[50];
    char surname[50];
    int student_id;
    double quiz_score;
    double final_score;
};
```

```
struct student stu[1000];
```

```
#include <stdio.h>

struct student {
    char name[50];
    char surname[50];
    int student_id;
    double quiz_score;
    double final_score;
};

void main() {
    struct student stu[1000];

}
```

Outline

- สตรัค และประโยชน์ของสตรัค
- การประกาศสตรัค
- typedef
- การเข้าถึงข้อมูลในสตรัค
- การรับข้อมูลและแสดงผลในสตรัค
- สตรัคในสตรัค
- อาเรย์ของสตรัค

typedef

- สังเกตการประกาศสตรัคส่วนของบัตร

```
struct student stu[1000];
```

- จะเห็นว่าเราต้องพิมพ์คำว่า struct ซ้อนกับชื่อของ struct (โรงงาน) ซึ่งทำให้ต้องพิมพ์ยาว และความหมายก็ซ้ำซ้อน
- เราสามารถเขียนโปรแกรมแบบที่ไม่ต้องใช้คำว่า struct ซ้อนกับชื่อได้ โดยการ define type (ประกาศเป็นประเภทใหม่)
- การใช้คำสั่งนี้ไม่ยาก เพียงแค่เราพิมพ์ typedef ตามด้วยชื่อของประเภทข้อมูลที่ต้องการตั้งเป็นอะไรก็ได้ที่ไม่ซ้ำกับข้อมูลอื่นหรือชื่อของ struct ไว้ส่วนท้ายของ struct (โรงงาน) ก่อนปิดด้วย ;

typedef

- เช่น

```
struct employee {
    char name[16];
    char surname[31];
    char address[151];
    float salary;
} typedef worker;
```

เป็นการ define ว่าให้ struct employee เป็นข้อมูลประเภท worker

การประกาศแบบนี้ทำให้เรียกใช้ struct (โรงงาน) นี้ได้ 2 แบบ

```
struct employee emp1;
```

```
worker emp1;
```

Outline

- สตรัค และประโยชน์ของสตรัค
- การประกาศสตรัค
- typedef
- **การเข้าถึงข้อมูลในสตรัค**
- การรับข้อมูลและแสดงผลในสตรัค
- สตรัคในสตรัค
- อาเรย์ของสตรัค

การเข้าถึงข้อมูลในสตรัค

- ตอนนี้เราสร้างได้ทั้งสตรัคส่วนโรงงาน และสตรัคส่วนของบัตรแล้ว แต่เรายังใส่ข้อมูลและอ่านข้อมูลบนบัตรไม่ได้
- ในส่วนนี้จะพูดถึงทั้งการเขียนข้อมูลลงบนบัตร และการอ่านค่า
- สิ่งที่เรากำลังจะพูดกันต่อไปนี้คือสตรัคแบบตัวเดียว (บัตรใบเดียว) และไม่เกี่ยวข้องกับโรงงาน เพราะข้อมูลที่เก็บไม่ได้อยู่ที่โรงงาน แต่อยู่บนบัตรเท่านั้น

การเข้าถึงข้อมูลในสตรัค

- การเข้าถึงข้อมูลในสตรัค (ในบัตร) เราจะใช้เครื่องหมายจุด (.) เพื่อกำหนดบัตรที่จะอ่าน/เขียน และข้อมูลที่เจาะจง เช่น

ตัวอย่างที่ 1

```
worker emp1;
emp1.salary = 18000;
```

แบบนี้คือ ให้เก็บ 18000 ลงบนบัตรชื่อ emp1 ส่วนของ salary

ตัวอย่างที่ 2

```
worker emp1, emp2;
emp1.salary = 31000;
emp2.name = “Surasak”;
```

ยกตัวอย่างว่าถ้ามีบัตรหลายใบ ก็ต้องเลือกให้ถูก

เก็บ 31000 บนบัตร emp1 ส่วนของ salary

เก็บ “Surasak” บนบัตร emp2 ส่วนของ name

นักศึกษาคิดว่า Surasak มีเงินเดือน 31000 ใช่หรือไม่?

การเข้าถึงข้อมูลในสตรัค

```
struct employee {
    char name[16];
    char surname[31];
    char address[151];
    float salary;
} typedef worker;
```

- สามารถใช้กับ printf เลยก็ได้

ตัวอย่างที่ 3

```
worker emp1, emp2;
emp1.salary = 31000;
emp2.name = “Surasak”;
printf(“%lf”, emp1.salary);
printf(“%s”, emp2.name);
```

Outline

- สตรัค และประโยชน์ของสตรัค
- การประกาศสตรัค
- typedef
- การเข้าถึงข้อมูลในสตรัค
- **การรับข้อมูลและแสดงผลในสตรัค**
- สตรัคในสตรัค
- อาเรย์ของสตรัค

การรับข้อมูลและแสดงผลในสตรัค

- การรับข้อมูล สามารถใช้ scanf ได้เลย แต่จะรับค่าให้ข้อมูลได้แค่ทีละตัว (คล้ายอาเรย์)
  - ไม่สามารถทำการ scanf กับสตรัคทั้งก้อนได้
  - ต้องระบุชนิดข้อมูลใน scanf ให้ถูกต้องด้วย

```
scanf("%s", emp.name);
scanf("%s", emp.surname);
scanf("%f", &emp.salary);
```

  - การใช้ scanf ก็เหมือนกับการใช้งานกับตัวแปรทั่วไปทุกประการ ทั้ง %s, %d, %f, etc. รวมไปถึงเครื่องหมาย & ข้างหน้าตัวแปรด้วย

การรับข้อมูลและแสดงผลในสตรัค

- การแสดงผลข้อมูล สามารถใช้ printf ได้เลย แต่จะแสดงค่าให้ข้อมูลได้แค่ทีละตัว (คล้ายอาเรย์ และการรับค่า)
  - ไม่สามารถทำการ printf ข้อมูลสตรัคทั้งก้อนพร้อมกันได้
  - ต้องระบุชนิดข้อมูลใน printf ให้ถูกต้องด้วย

```
printf("%s %s\n", emp.name, emp.surname);
printf("%s\n", emp.address);
printf("%.2f", emp.salary);
```

  - การใช้ printf ก็เหมือนกับการใช้งานกับตัวแปรทั่วไปทุกประการ ทั้ง %s, %d, %f, etc.

การรับข้อมูลและแสดงผลในสตรัค

- ตัวอย่างโค้ดเมื่อนำความรู้ทั้งหมดที่มาใช้เขียนโปรแกรมร่วมกัน
- นักศึกษาลองแยกดูว่าส่วนไหนคือโรงงาน ส่วนไหนคือบัตร
- ถ้าจะประกาศบัตร employee เพิ่ม ต้องเพิ่มตรงไหน

```c
#include <stdio.h>

struct employee {
    char name[16];
    char surname[31];
    char address[151];
    float salary;
} typedef worker;

void main() {
    worker emp;
    scanf("%s", emp.name);
    scanf("%s", emp.surname);
    scanf("%s", emp.address);
    scanf("%f", &emp.salary);
    printf("%s %s\n", emp.name, emp.surname);
    printf("%s\n", emp.address);
    printf("%.2f", emp.salary);
}
```

การรับข้อมูลและแสดงผลในสตรัค

ตัวอย่างโจทย์ (2) จากสตรัคที่กำหนดให้ เป็นสตรัคที่ใช้เก็บข้อมูลนักศึกษา ประกอบด้วย

- รหัสประจำตัวนักศึกษา
- ชื่อ-นามสกุลอยู่ด้วยกัน
- ชั้นปี
- ระดับผลการเรียนเฉลี่ย

```
struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;
```

จงเขียนโปรแกรมสร้างสตรัค เก็บข้อมูลนักศึกษา 2 คน และเก็บค่าข้อมูลทั้งหมดเข้าทางคีย์บอร์ด จากนั้นแสดงผลข้อมูลทั้งหมดออกทางจอภาพตามลำดับ

การรับข้อมูลและแสดงผลในสตรัค

ตัวอย่างโจทย์ (2)

```c
#include <stdio.h>
#include <string.h>

struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;

void main(){
    char naame[35];
    STUDENT_RECORD student1, student2;
    printf("Enter student1 ID: ");
    scanf("%d", &student1.ID);
    printf("Enter student1 name: ");
    scanf("%s", student1.name);
    printf("Enter student1 year: ");
    scanf("%d", &student1.year);
    printf("Enter student1 GPA: ");
    scanf("%f", &student1.GPA);
    printf("Enter student2 ID: ");
    scanf("%d", &student2.ID);
    printf("Enter student2 name: ");
    scanf("%s", student1.name);
    printf("Enter student2 year: ");
    scanf("%d", &student2.year);
    printf("Enter student2 GPA: ");
    scanf("%f", &student2.GPA);

    printf("\n\nStudent1 Record:\n");
    printf("ID: %d\n", student1.ID);
    printf("Name: %s\n", student1.name);
    printf("Year: %d\n", student1.year);
    printf("GPA: %.2f\n", student1.GPA);

    printf("\n\nStudent2 Record:\n");
    printf("ID: %d\n", student2.ID);
    printf("Name: %s\n", student2.name);
    printf("Year: %d\n", student2.year);
    printf("GPA: %.2f\n", student2.GPA);
}
```

Outline

- สตรัค และประโยชน์ของสตรัค
- การประกาศสตรัค
- typedef
- การเข้าถึงข้อมูลในสตรัค
- การรับข้อมูลและแสดงผลในสตรัค
- **สตรัคในสตรัค**
- อาเรย์ของสตรัค

สตรัคในสตรัค

- เมื่อสตรัคที่เราสร้างขึ้นมา ถูกมองเป็นประเภทข้อมูลชนิดหนึ่ง
- และเรารู้ว่าเราสามารถนำข้อมูลชนิดใดก็ตามมาเป็นสมาชิกของสตรัค
- ดังนั้นสตรัคที่เป็นข้อมูลชนิดหนึ่งจึงเป็นสมาชิกของสตรัคอื่นได้ด้วย
- ตัวอย่างเช่น เราต้องการสร้างสตรัคของ “กลุ่มของนักศึกษา” ซึ่งประกอบด้วยนักศึกษาในกลุ่ม และข้อมูลอื่นๆ เพิ่มเติมได้ เช่นกิจกรรมที่นักศึกษาเข้าร่วม

สตรัคในสตรัค

- สมมติว่านักศึกษากลุ่มหนึ่งประกอบด้วยนักศึกษา 4 คน จะได้สตรัคของกลุ่มนักศึกษาเป็นลักษณะนี้

```
struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;
```

```
struct group {
    STUDENT_RECORD s1, s2, s3, s4;
    int act1, act2, act3, act4;
} typedef GROUP;
```

สตรัคในสตรัค

- ถ้าแสดงให้เป็นรูปธรรม ก็เหมือนนำข้อมูลบัตรแบบเดียวกัน 4 ใบ มารวมอยู่บนบัตรใบเดียว (ทุกครั้งที่ประกาศบัตรใบใหญ่ขึ้นมา ก็จะมีบัตรใบเล็กถูกสร้างขึ้นข้างใน 4 ใบ)

สตรัคในสตรัค

ตัวอย่างโจทย์ (3) กำหนดกลุ่มนักศึกษา 1 กลุ่ม มี 4 คน และ 4 act โดยมีรายละเอียดดังภาพ จงเขียนโปรแกรมรับค่าข้อมูลนักศึกษาทุกตัวมาเก็บไว้ในกลุ่มนี้

```
struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;
```

```
struct group {
    STUDENT_RECORD s1, s2, s3, s4;
    int act1, act2, act3, act4;
} typedef GROUP;
```

สตรัคในสตรัค

ตัวอย่างโจทย์ (3)

สังเกตการใช้จุดซ้อนกันสองชั้น (สตรัคที่อยู่ในสตรัค)

act1 เป็นตัวแปร ไม่ใช่สตรัค จึงไม่ต้องซ้อนจุด

ทำลักษณะเดียวกันกับ s3, s4 ในทุกข้อมูล

```
void main() {
    GROUP g;
    scanf("%d", &g.s1.ID);
    scanf("%s", g.s1.name);
    scanf("%d", &g.s1.year);
    scanf("%f", &g.s1.GPA);
    scanf("%d", &g.act1);
    scanf("%d", &g.s2.ID);
    scanf("%s", g.s2.name);
    scanf("%d", &g.s2.year);
    scanf("%f", &g.s2.GPA);
    scanf("%d", &g.act2);
    // do the same for s3 and s4
}
```

Outline

- สตรัค และประโยชน์ของสตรัค
- การประกาศสตรัค
- typedef
- การเข้าถึงข้อมูลในสตรัค
- การรับข้อมูลและแสดงผลในสตรัค
- สตรัคในสตรัค
- **อาเรย์ของสตรัค**

อาเรย์ของสตรัค

- สังเกตจากตัวอย่างที่แล้ว เราต้องเขียนโค้ดแบบเดิมซ้ำกันหลายรอบ จาก s1, s2 ไปต่อแบบเดิมที่ s3 และ s4 ด้วย สังเกตว่าเป็นการทำซ้ำ ๆ ในรูปแบบเดิม ๆ
- การทำซ้ำ ๆ อาจจะใช้ loop ในการช่วยเหลือได้ แต่ชื่อตัวแปร s1, s2, s3, s4 ก็จะเปลี่ยนไปตาม loop ได้ยากอยู่ดี นอกเสียจากเราเปลี่ยนมันเป็น s[1], s[2], s[3], s[4] ซึ่งก็คืออาเรย์
  - จัดการอาเรย์ด้วยลูป ใช้โค้ดกับนักเรียนหลาย ๆ คนด้วยลูปเดียว
  - เปลี่ยนดัชนีของอาเรย์เท่ากับเปลี่ยนไปเก็บข้อมูลนักศึกษาคนอื่น
  - เนื่องจากจำนวนกิจกรรมที่นักศึกษาแต่ละคนเข้าร่วมไม่อยู่สตรัค
    - ต้องสร้างอาเรย์ของข้อมูลนี้แยกออกมาเพิ่มเติม
  - เพื่อที่จะใช้เทคนิคนี้ได้ เราต้องทำความเข้าใจเรื่องอาเรย์ของสตรัคเพิ่มเติม

อาเรย์ของสตรัค

- เราได้เกริ่นกันมาตั้งแต่ต้นแล้วว่า...
  - ประกาศบัตรพนักงาน 1 ใบ ใช้ `struct employee emp1;`
  - ประกาศบัตรพนักงาน 2 ใบ อาจใช้ `struct employee emp1, emp2;`
  - ประกาศบัตรพนักงาน 30 ใบ อาจใช้ `struct employee emp[30];`
  - ประกาศบัตรพนักงาน n ใบ ใช้ ...

อาเรย์ของสตรัค

ตัวอย่างโจทย์ (4) ให้นักศึกษาเขียนประโยคสำหรับประกาศสตรัคของนักศึกษา 4 คน จากสตรัคแม่แบบที่กำหนดให้ โดยใช้อาเรย์

```
struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;
```

อาเรย์ของสตรัค

ตัวอย่างโจทย์ (4) ให้นักศึกษาเขียนประโยคสำหรับประกาศสตรัคของนักศึกษา 4 คน จากสตรัคแม่แบบที่กำหนดให้ โดยใช้อาเรย์

วิเคราะห์ ก่อนหน้านี้ก็มีการประกาศสตรัคของนักศึกษา 4 คนเหมือนกัน แต่ไม่ได้ใช้อาเรย์ ถ้าประกาศเป็นอาเรย์ก็จะได้ประโยคสั้น ๆ ดังนี้

```
STUDENT_RECORD s[4];
```

อาเรย์ของสตรัค

- เมื่อเราต้องการเรียกใช้ข้อมูล เราก็ทำเหมือนเดิม แต่เปลี่ยนรูปแบบการเรียกใช้แบบตัวแปรธรรมดา ที่เรียกโดย s1.grade เป็น s[0].grade แทน
  - สังเกต ตรงนี้ผิดกันเยอะ เราต้องใส่ [ ] ของอาเรย์ของสตรัคไว้หลังชื่อสตรัค ไม่ใช่ใส่ไว้หลังตัวแปร
    - s[0].grade แบบนี้ถูกต้อง แต่ s.grade[0] แบบนี้ผิด
  - เว้นแต่ว่าข้อมูลในสตรัค เป็นข้อมูลที่เป็นอาเรย์ เช่น s[0].name[3] หมายถึง อักษรตัวที่ 4 ของชื่อ ของนักศึกษาคนแรก
  - ถ้าสตรัคถูกประกาศเป็นอาเรย์ หลังชื่อสตรัคจะมี [ ] เสมอ
  - สตรัคที่ถูกประกาศเป็นอาเรย์ได้คือส่วนบัตร ไม่ใช่ส่วนโรงงาน

อาเรย์ของสตรัค

ตัวอย่างการใช้อาเรย์ของสตรัค

```c
#include<stdio.h>
struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;
```

```c
void main() {
    STUDENT_RECORD S[4];
    int i;
    for(i = 0; i < 4; ++i) {
        scanf("%d", &S[i].ID);
        scanf("%s", S[i].name);
        scanf("%d", &S[i].year);
        scanf("%f", &S[i].GPA);
    }
    for(i = 0; i < 4; ++i) {
        printf("ID #%d: %d\n", i, S[i].ID);
        printf("name #%d: %s\n", i, S[i].name);
        printf("year #%d: %d\n", i, S[i].year);
        printf("GPA #%d: %f\n", i, S[i].GPA);
    }
}
```

อาเรย์ของสตรัค

ตัวอย่างโจทย์ (3) วิธีที่ 2

```c
#include<stdio.h>

struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;

struct group {
    STUDENT_RECORD s1, s2, s3, s4;
    int act1, act2, act3, act4;
} typedef GROUP;
```

สตรัคตัวเดิม

```c
void main() {
    STUDENT_RECORD S[4];
    int A[4];
    int i;
    for(i = 0; i < 4; ++i) {
        scanf("%d", &S[i].ID);
        scanf("%s", S[i].name);
        scanf("%d", &S[i].year);
        scanf("%f", &S[i].GPA);
        scanf("%d", &A[i]);
    }
    GROUP g;
    g.s1 = S[0]; g.act1 = A[0];
    g.s2 = S[1]; g.act2 = A[1];
    g.s3 = S[2]; g.act3 = A[2];
    g.s4 = S[3]; g.act4 = A[3];
}
```

ประกาศสตรัคใหม่เป็นอาเรย์

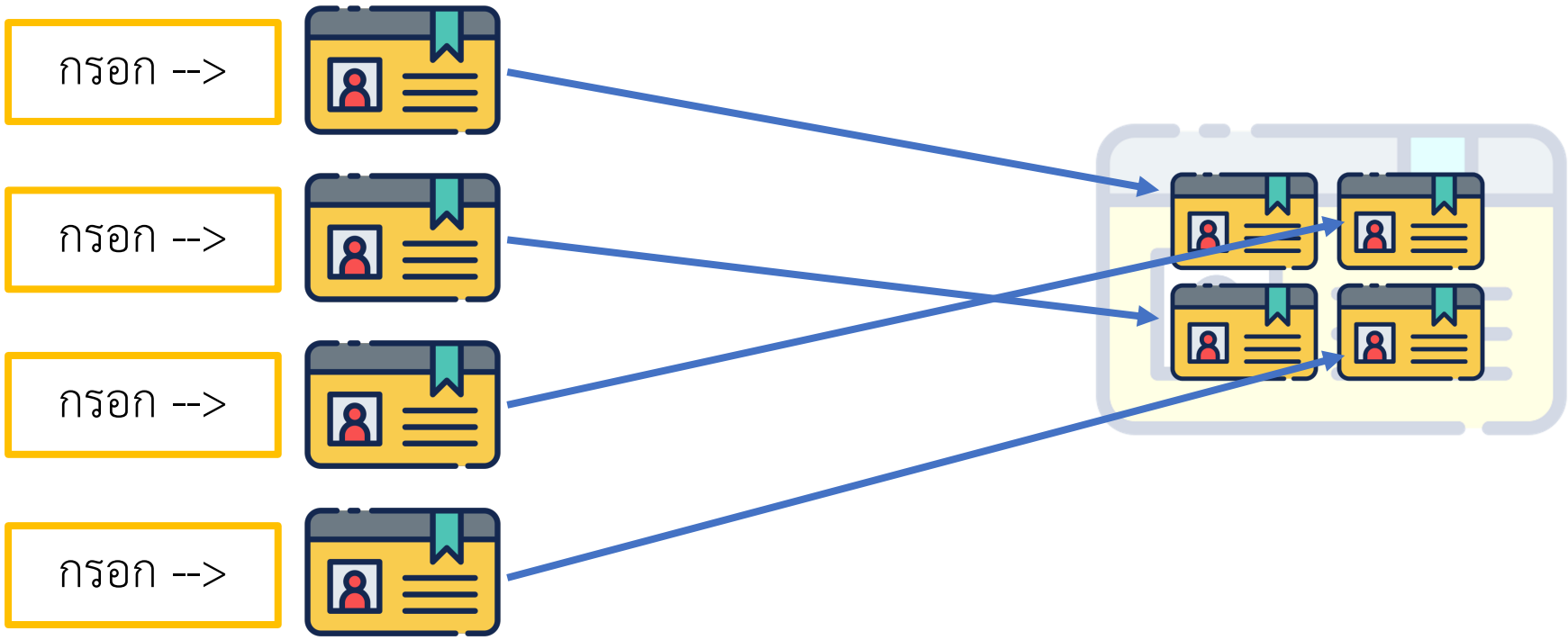
รับและพักข้อมูลไว้ในอาเรย์ของสตรัคก่อน

ค่อยๆถ่ายอาเรย์ที่พักไว้เข้าสู่สตรัคใหญ่

พอเป็นสตรัครูปแบบเดียวกับ ใช้ = ได้เลย

อาเรย์ของสตรัค

- ถ้าแสดงให้เป็นรูปธรรม ก็เหมือนเรากรอกข้อมูลบัตรทีละใบให้เรียบร้อยโดยใช้ loop เมื่อกรอกข้อมูลครบแล้ว ก็นำบัตรทั้งหมดไปใส่ไว้ในกลุ่ม (ในกลุ่มมีบัตรว่างรออยู่แล้ว)



อาเรย์ของสตรัค

ตัวอย่างโจทย์ (4) **โจทย์ตัวอย่างที่ยากที่สุดและซับซ้อนที่สุดของวิชานี้** *อย่าเพิ่งตกใจ เขาแค่ขู่เฉยๆ

จากสตรัคในตัวอย่างโจทย์ (3) จงแก้ไขสตรัคให้รองรับนักศึกษาทั้งหมด 200 คน โดยให้แบ่งกลุ่มนักศึกษาออกเป็น 20 กลุ่ม กลุ่มละ 10 คน

วิเคราะห์โจทย์

1. เราควรใช้อาเรย์มาเก็บกลุ่ม เพราะมีถึง 20 กลุ่ม (ตัวอย่างก่อนหน้านี้มีกลุ่มเดียว)
2. สตรัคของกลุ่มก็ควรเก็บอาเรย์ของนักศึกษา (ก่อนหน้านี้มีแค่ 4 คน เลยเก็บเป็นตัวแปร)
3. จะเก็บลงไปในสตรัคของกลุ่มตรง ๆ เลยก็ได้ หรือจะค่อย ๆ เก็บแบบบัตรนักศึกษาทีละใบแล้วค่อยบรรจุลงในกลุ่มตามลำดับก็ได้เหมือนกัน

อาเรย์ของสตรัค

ตัวอย่างโจทย์ (4) วิธีที่ 1 รับค่าโดยตรง แบบนี้จะเก็บข้อมูลเข้าสู่บัตรที่อยู่ในกลุ่มโดยตรง ไม่ต้องแยกบัตรออกมาเขียน

```c
#include<stdio.h>

struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;

struct group {
    STUDENT_RECORD member[10];
    int act[10];
} typedef GROUP;
```

สตรัค StudentRecord เหมือนเดิม

ในสตรัค group ปรับตัวแปรเป็นอาเรย์

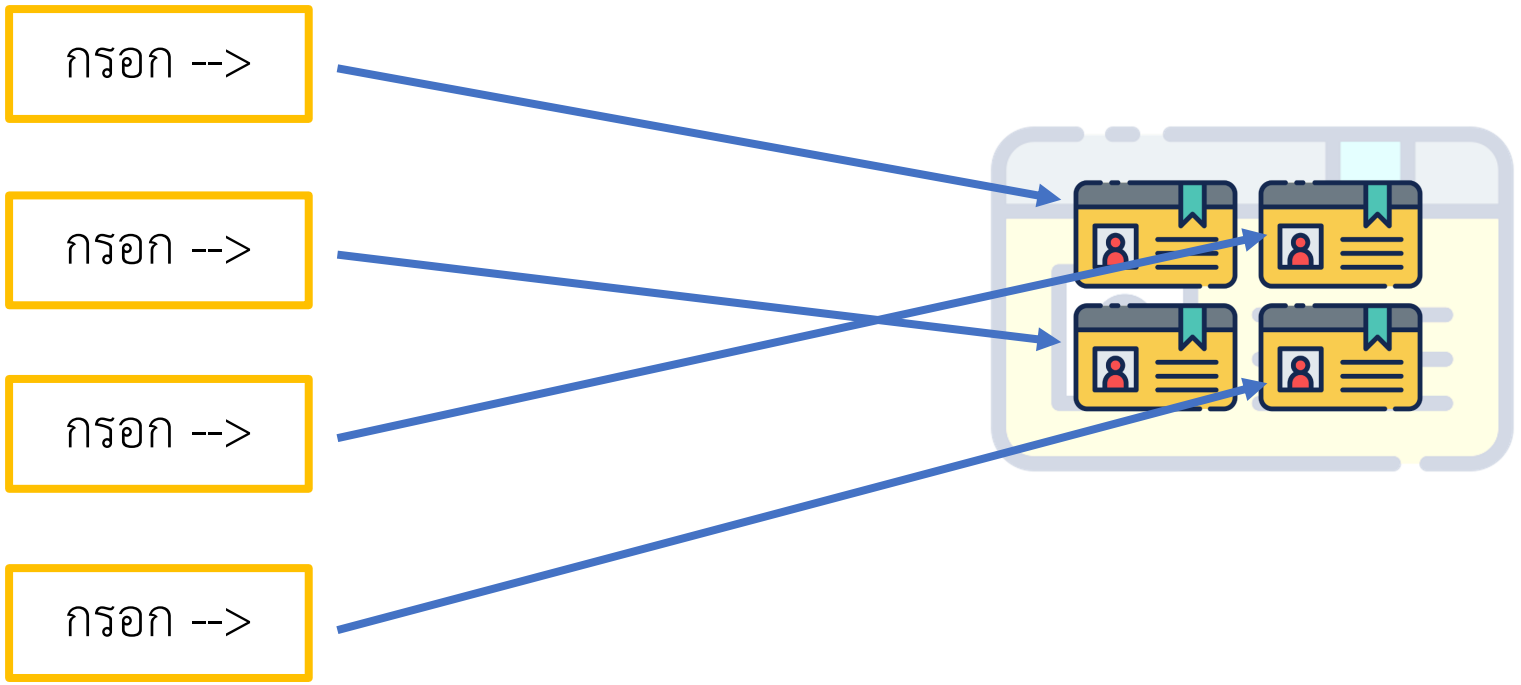
```c
void main() {
    GROUP g[20];
    int i, j;
    for(i = 0; i < 20; ++i) {
        for(j = 0; j < 10; ++j) {
            scanf("%d", &g[i].member[j].ID);
            scanf("%s", g[i].member[j].name);
            scanf("%d", &g[i].member[j].year);
            scanf("%f", &g[i].member[j].GPA);
            scanf("%d", &g[i].act[j]);
        }
    }
}
```

อาเรย์ของสตรัค 20 กลุ่ม

Loop สำหรับรับค่า 20 กลุ่ม กลุ่มละ 10 คน

อาเรย์ของสตรัค

- ถ้าแสดงให้เป็นรูปธรรม ก็เหมือนเรากรอกข้อมูลลงไปในบัตรที่อยู่ในกล่องอยู่แล้ว ไม่ใช่หยิบออกมาเขียนทีละใบ (ภาพประกอบแสดงแค่ 4 ใบ แต่จริงๆ มี 20 ใบ)



อาเรย์ของสตรัค

ตัวอย่างโจทย์ (4) วิธีที่ 2

```c
#include<stdio.h>

struct StudentRecord {
    int ID;
    char name[256];
    int year;
    float GPA;
} typedef STUDENT_RECORD;

struct group {
    STUDENT_RECORD member[10];
    int act[10];
} typedef GROUP;
```

สตรัค StudentRecord เหมือนเดิม

ในสตรัค group ปรับตัวแปรเป็นอาเรย์

```c
void main() {
    STUDENT_RECORD S[200];
    int A[200];
    GROUP g[20];
    int i, j;
    for(i = 0; i < 200; ++i) {
        scanf("%d", &S[i].ID);
        scanf("%s", S[i].name);
        scanf("%d", &S[i].year);
        scanf("%f", &S[i].GPA);
        scanf("%d", &A[i]);
    }
    for(i = 0; i < 20; ++i) {
        for(j = 0; j < 10; ++j) {
            g[i].member[j] = S[j + 10*i];
            g[i].act[j] = A[j + 10*i];
        }
    }
}
```

สตรัค StudentRecord และ A สำหรับ 200 คน (บัตร 200 ใบ)

อาเรย์ของสตรัค 20 กลุ่ม

รับค่าเข้าทีละคนเหมือนเดิม ตัวอย่างที่ 3 แค่เปลี่ยนจากรับ 4 คนเป็น 200 คน

ค่อยๆจับบัตรใส่ในกลุ่มทีละใบ ตรงนี้จะดูยากหน่อยเพราะต้องพิจารณาว่าบัตรใบไหนใส่กลุ่มที่เท่าไหร่

อาเรย์ของสตรัค

ตัวอย่างโจทย์ (4) วิธีที่ 2+

```
int cnt=0;
for(i = 0; i < 20; ++i) {
    for(j = 0; j < 10; ++j) {
        g[i].member[j] = S[cnt];
        g[i].act[j] = A[cnt];
        cnt++;
    }
}
```

ในส่วนหยิบบัตรใส่กลุ่มตอนท้าย เราอาจจะสร้างตัวนับ (cnt) ขึ้นมาแล้วไล่เก็บอาเรย์เรียงตัวไปเรื่อยๆ แบบนี้ก็ได้ ถ้ารู้สึกง่ายกว่า

แบบนี้จะค่อยๆ หยิบบัตรเข้าสู่กลุ่มทีละใบจนเต็มกลุ่ม จึงจะขยับต่อเนื่องไปเรื่อยๆ

การกำหนดค่าเริ่มต้นให้อาเรย์และสตรัคในโค้ด

- เราสามารถกำหนดค่าเริ่มต้นให้อาเรย์ได้ ผ่านการใช้เครื่องหมาย { } เช่น

```
int A[5] = {9, 7, 10, 0, 2};

float F[4] = {2.35, 1.78, -1.2, 0.5};
```

- วิธีข้างบนนี้จะทำให้ตัวเลขไปปรากฏในอาเรย์เรียงตามลำดับจากช่องที่ 0 ไปช่องที่ 1, 2, ...
- เราสามารถกำหนดค่าเริ่มต้นให้กับสตรัคได้เหมือนกันผ่านเครื่องหมาย { }
- เราใส่ข้อมูลเข้าไปทีละตัวตามลำดับการปรากฏตอนประกาศสตรัค
- เช่น STUDENT_RECORD s = {112233, “N0tail”, 1, 3.21};

การกำหนดค่าเริ่มต้นให้อาเรย์และสตรัคในโค้ด

ถ้าเป็นอาเรย์ของสตรัค เราใช้ { } กับอาเรย์ด้านนอกตามปรกติ ส่วนข้อมูลของสตรัคแต่ละตัวจะมี { } ของมันเอง

```
STUDENT_RECORD SR[3] = {    {112233, “N0tail”, 1, 3.21},
                            {156274, “Topson”, 3, 3.53},
                            {186461, “ana”, 2, 3.08}    };
```

ตัวแปรในสตรัคซ้ำกับตัวแปรภายนอกได้หรือไม่

ได้

ตัวแปรในสตรัคซ้ำกับตัวแปรภายนอกได้หรือไม่

ตัวอย่างที่มีการตั้งชื่อตัวแปรข้างนอกซ้ำกับข้างในสตรัค จะพบว่าใช้ได้ เพราะตัวแปรมีความแตกต่างกัน ตัวแปรในสตรัคจะมีชื่อสตรัคและเครื่องหมาย . กำกับ ในขณะที่ตัวแปรอิสระภายนอกจะเป็นชื่อโดด ๆ ให้สังเกตตัว S และตัว A

```
for(i = 0; i < 200; ++i) {
    scanf("%d", &S[i].ID);
    scanf("%s %s", S[i].name, S[i].surname);
    scanf("%d", &S[i].year);
    scanf("%f", &S[i].GPA);
    scanf("%d", &A[i]);
}
for(i = 0; i < 20; ++i) {
    for(j = 0; j < 10; ++j) {
        g[i].S[j] = S[j + 20*i];
        g[i].A[j] = A[j + 20*i];
    }
}
```

สรุป

- สตรัค และประโยชน์ของสตรัค
- การประกาศสตรัค
- typedef
- การเข้าถึงข้อมูลในสตรัค
- การรับข้อมูลและแสดงผลในสตรัค
- สตรัคในสตรัค
- อาเรย์ของสตรัค