- Username : class-1
- Password :

- Check VS 2019 whether can use
- We will start our course in 18:30
- we will start demonstrate the exercises at 19:15.
- Do not use scanf_s
- Please make sure the TA has recorded your exercise score here before leaving.

Schedule

We may provide some exercises for you to practice.

12/19	Final Exam
12/26	Review & Practice
1/2	Review & Practice
1/9	The Third Exam

Structure、Union、Enum

Introduction to Computers and Programming

Lab Course

TA 林垣志

2023.11.28

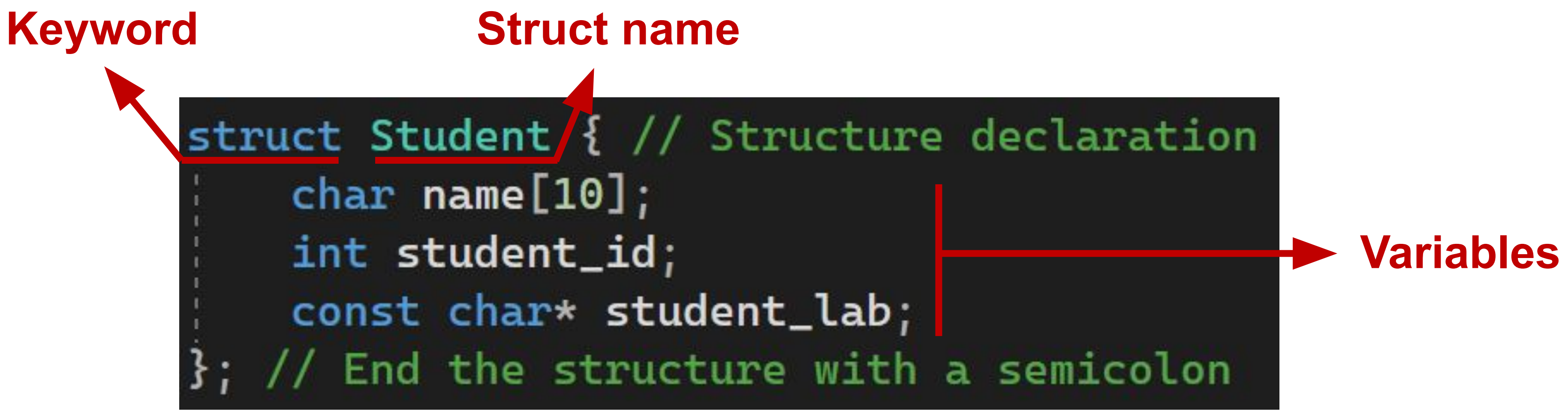
C語言-struct、union、enum | 鋼彈盪單槓 (gundambox.github.io)

What is Structure

- A **struct** is a composite data type declaration that defines a physically grouped list of variables under one name in a block of memory.
- Allowing the different variables to be accessed via a single pointer or by the struct declared name which returns the same address.

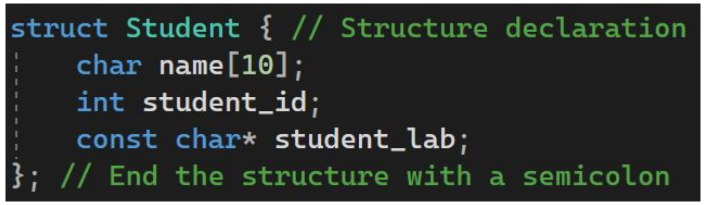
Structure Declaration

- A structure can contain any valid data types such as int, char, float, array, pointer or even other structures.

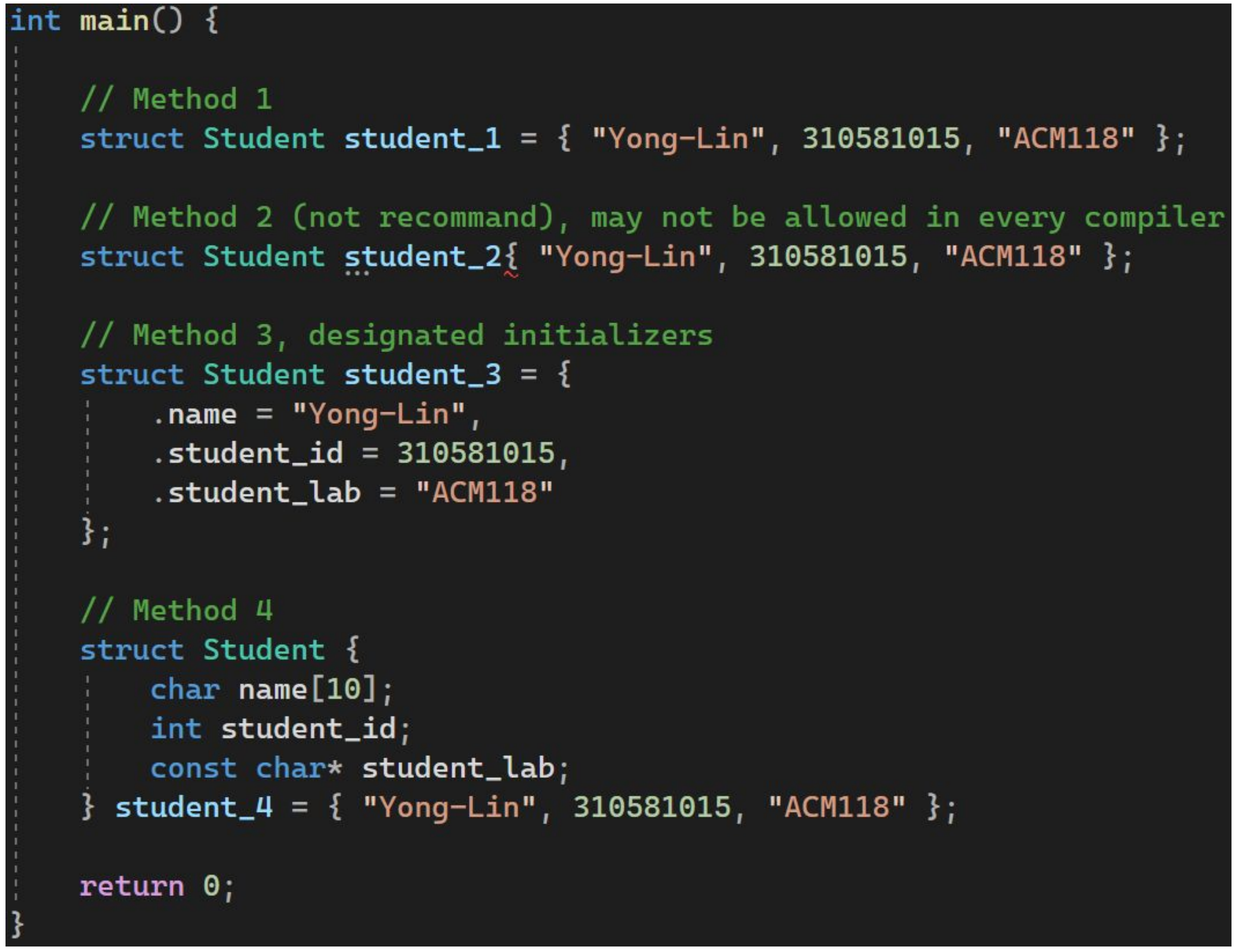


```
struct Student { // Structure declaration
    char name[10];
    int student_id;
    const char* student_lab;
}; // End the structure with a semicolon
```

Structure Initialization

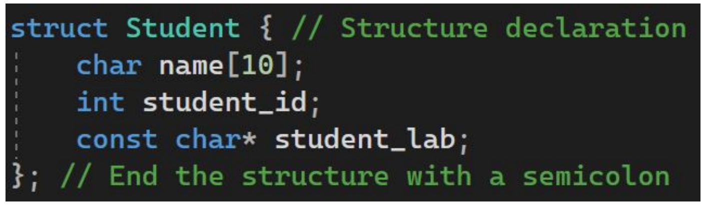


```
struct Student { // Structure declaration
    char name[10];
    int student_id;
    const char* student_lab;
}; // End the structure with a semicolon
```

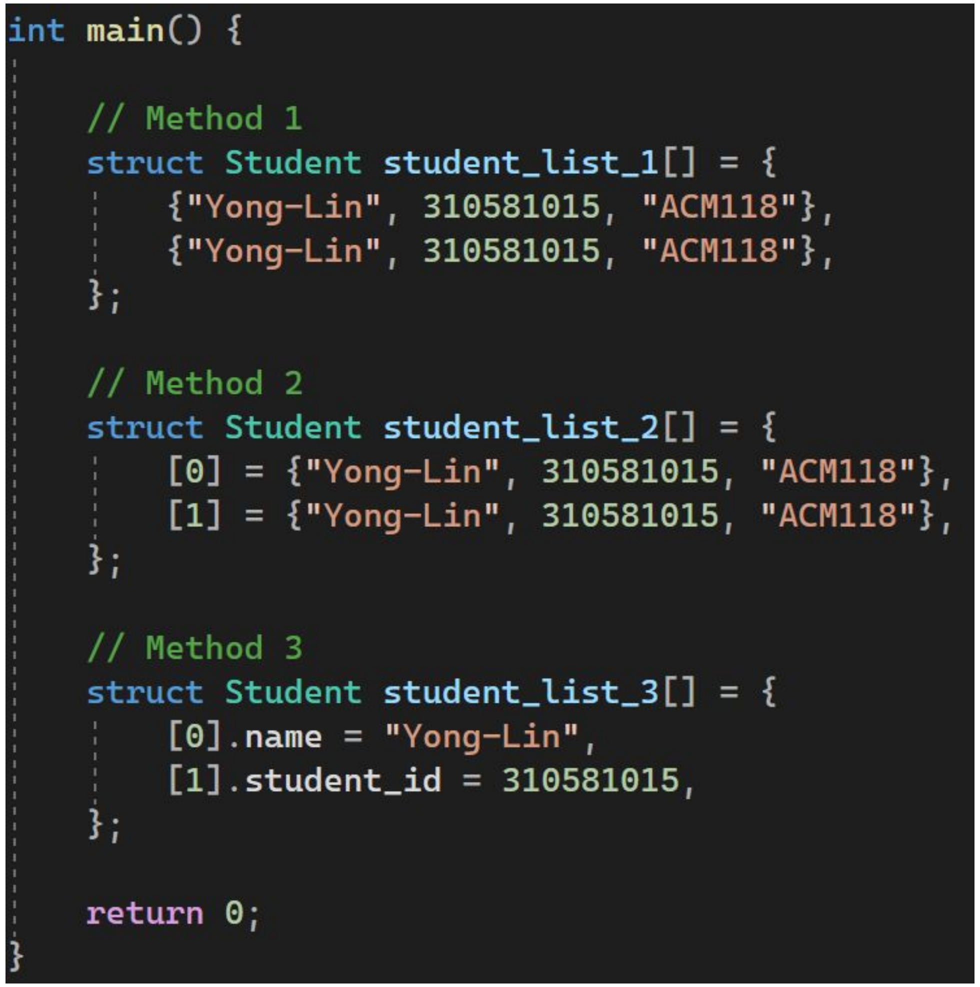


```
int main() {

    // Method 1
    struct Student student_1 = { "Yong-Lin", 310581015, "ACM118" };

    // Method 2 (not recommand), may not be allowed in every compiler
    struct Student student_2{ "Yong-Lin", 310581015, "ACM118" };

    // Method 3, designated initializers
    struct Student student_3 = {
        .name = "Yong-Lin",
        .student_id = 310581015,
        .student_lab = "ACM118"
    };

    // Method 4
    struct Student {
        char name[10];
        int student_id;
        const char* student_lab;
    } student_4 = { "Yong-Lin", 310581015, "ACM118" };

    return 0;
}
```

Structure Array Initialization

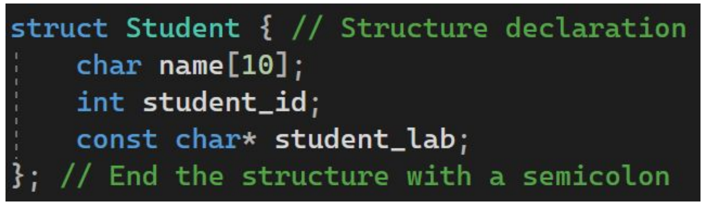


```c
struct Student { // Structure declaration
    char name[10];
    int student_id;
    const char* student_lab;
}; // End the structure with a semicolon
```

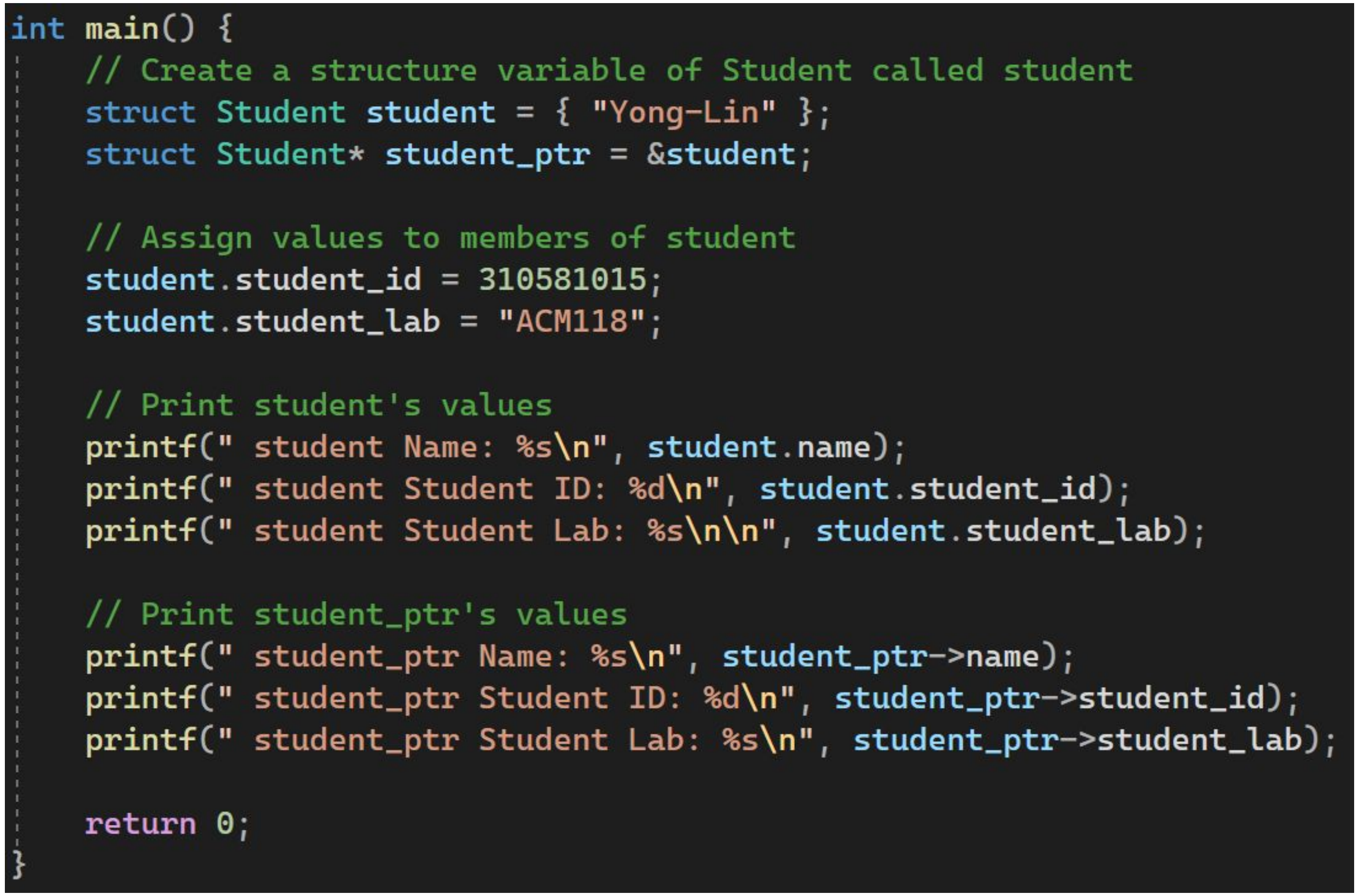


```c
int main() {

    // Method 1
    struct Student student_list_1[] = {
        {"Yong-Lin", 310581015, "ACM118"},
        {"Yong-Lin", 310581015, "ACM118"},
    };

    // Method 2
    struct Student student_list_2[] = {
        [0] = {"Yong-Lin", 310581015, "ACM118"},
        [1] = {"Yong-Lin", 310581015, "ACM118"},
    };

    // Method 3
    struct Student student_list_3[] = {
        [0].name = "Yong-Lin",
        [1].student_id = 310581015,
    };

    return 0;
}
```

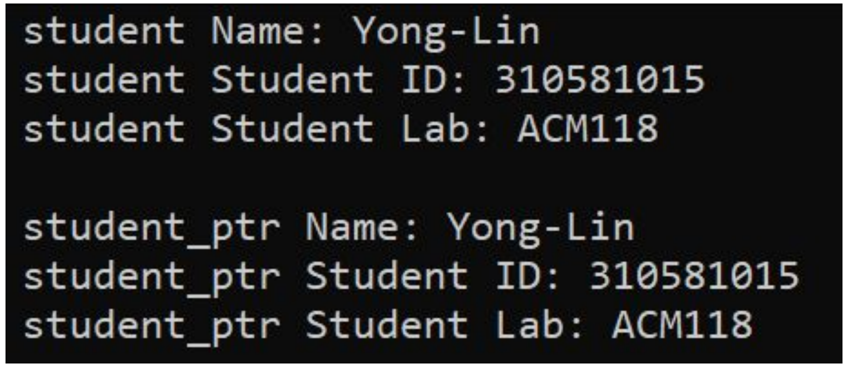
Access Structure Members



```c
struct Student { // Structure declaration
    char name[10];
    int student_id;
    const char* student_lab;
}; // End the structure with a semicolon
```

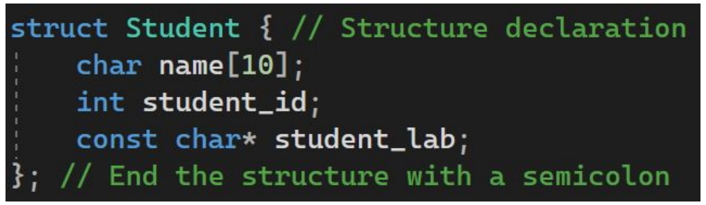


```c
int main() {
    // Create a structure variable of Student called student
    struct Student student = { "Yong-Lin" };
    struct Student* student_ptr = &student;

    // Assign values to members of student
    student.student_id = 310581015;
    student.student_lab = "ACM118";

    // Print student's values
    printf(" student Name: %s\n", student.name);
    printf(" student Student ID: %d\n", student.student_id);
    printf(" student Student Lab: %s\n\n", student.student_lab);

    // Print student_ptr's values
    printf(" student_ptr Name: %s\n", student_ptr->name);
    printf(" student_ptr Student ID: %d\n", student_ptr->student_id);
    printf(" student_ptr Student Lab: %s\n", student_ptr->student_lab);

    return 0;
}
```

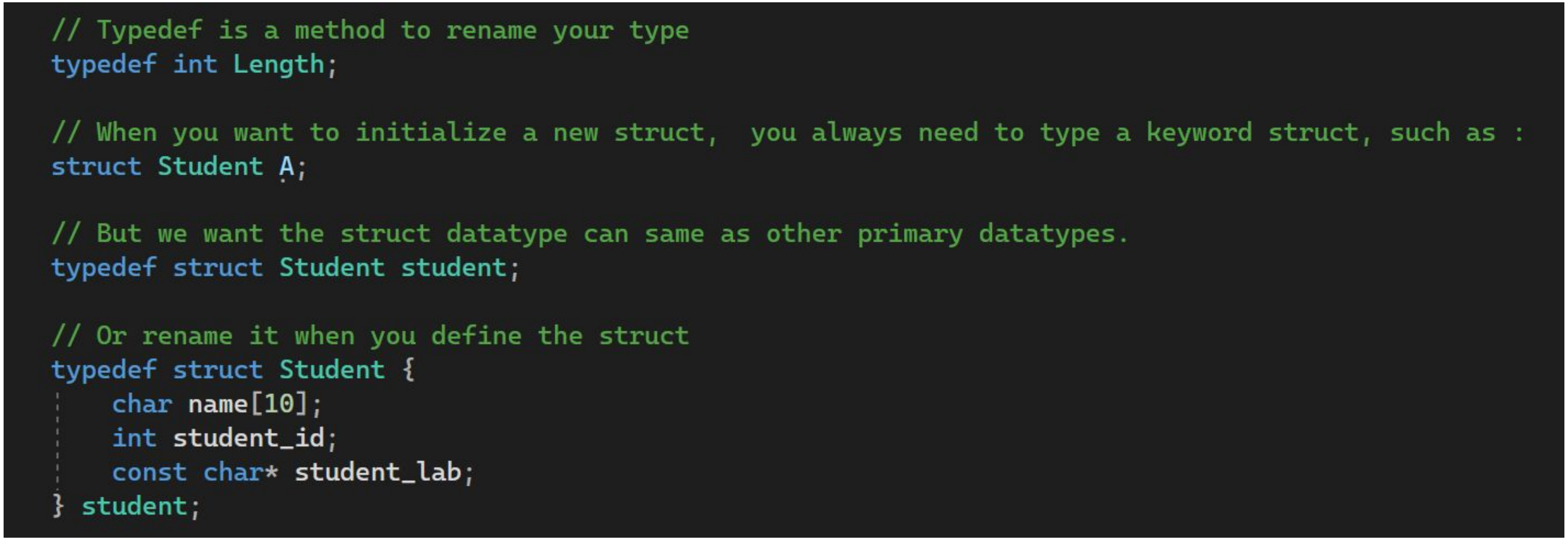


```
 student Name: Yong-Lin
 student Student ID: 310581015
 student Student Lab: ACM118

 student_ptr Name: Yong-Lin
 student_ptr Student ID: 310581015
 student_ptr Student Lab: ACM118
```

Keyword :: typedef



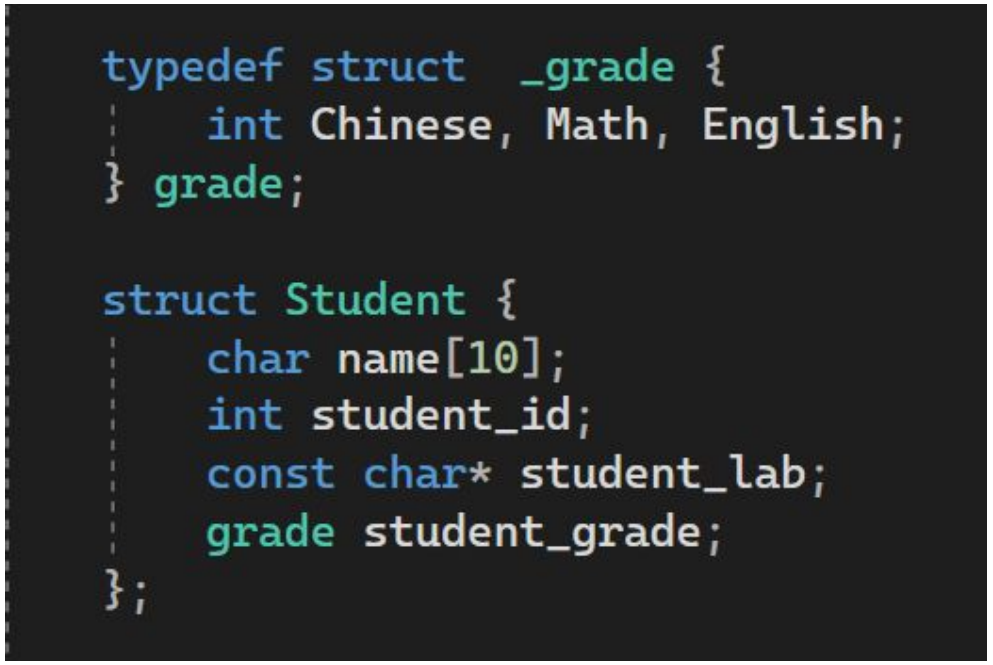
```c
struct Student { // Structure declaration
    char name[10];
    int student_id;
    const char* student_lab;
}; // End the structure with a semicolon
```



```c
// Typedef is a method to rename your type
typedef int Length;

// When you want to initialize a new struct,  you always need to type a keyword struct, such as :
struct Student A;

// But we want the struct datatype can same as other primary datatypes.
typedef struct Student student;

// Or rename it when you define the struct
typedef struct Student {
    char name[10];
    int student_id;
    const char* student_lab;
} student;
```

Structure in Structure

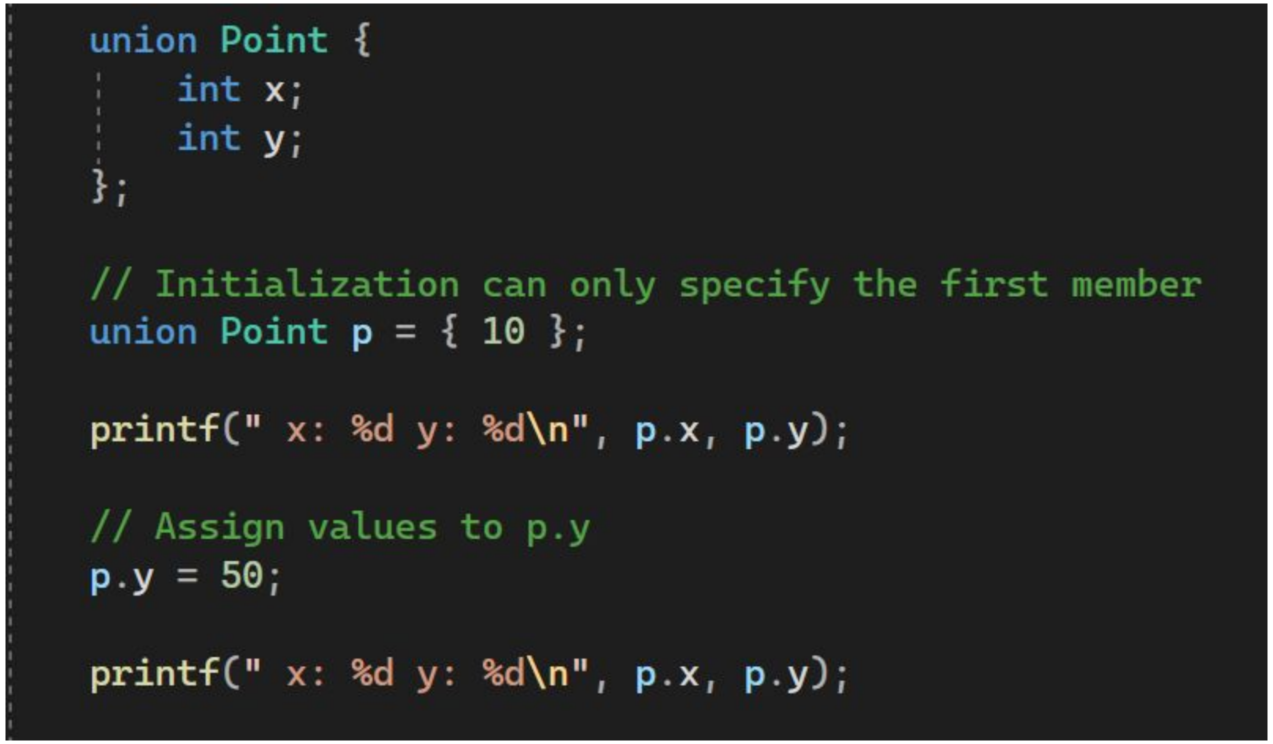
- In the large project, we always define more complicate structure type, and they may have the **structure in the structure**.



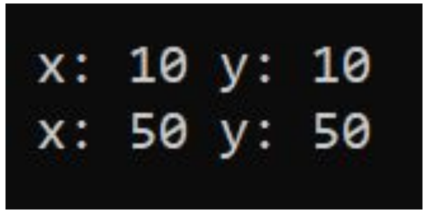
```
typedef struct  _grade {
    int Chinese, Math, English;
} grade;

struct Student {
    char name[10];
    int student_id;
    const char* student_lab;
    grade student_grade;
};
```

Union

- Union is similar to Struct, but the data in the union share the same memory, and the operation same as the Struct.



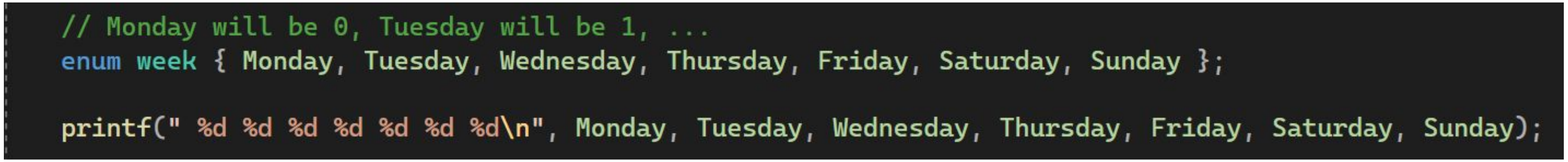
```c
union Point {
    int x;
    int y;
};

// Initialization can only specify the first member
union Point p = { 10 };

printf(" x: %d y: %d\n", p.x, p.y);

// Assign values to p.y
p.y = 50;

printf(" x: %d y: %d\n", p.x, p.y);
```



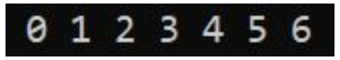
```
x: 10 y: 10
x: 50 y: 50
```

Enum

- Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

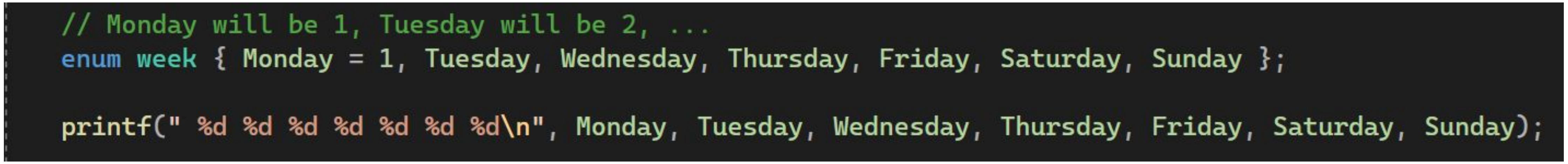


```c
// Monday will be 0, Tuesday will be 1, ...
enum week { Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday };

printf(" %d %d %d %d %d %d %d\n", Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday);
```

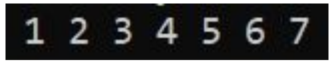


```
0 1 2 3 4 5 6
```

- Or you can indicate the number of first element.



```c
// Monday will be 1, Tuesday will be 2, ...
enum week { Monday = 1, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday };

printf(" %d %d %d %d %d %d %d\n", Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday);
```



```
1 2 3 4 5 6 7
```

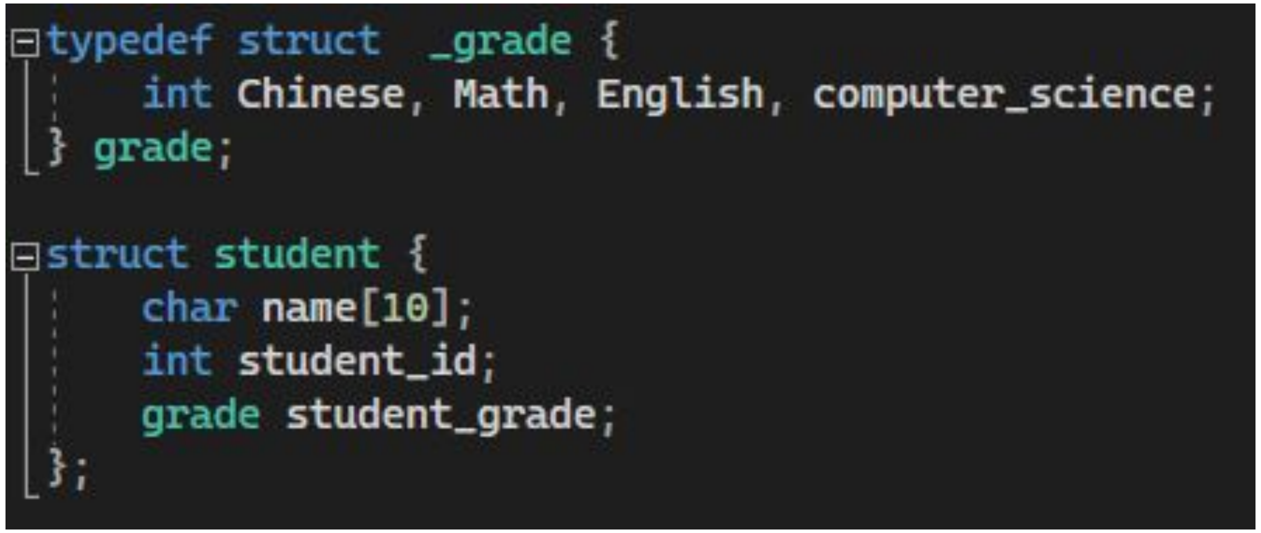
Exercises

2023.11.28

Exercise1

Create a grade alert system

- Please define a structure named **student with name, student_id**, and a **nested structure** named **grade with Chinese, Math, English, Computer_science**.



```
typedef struct  _grade {
        int Chinese, Math, English, computer_science;
} grade;

struct student {
        char name[10];
        int student_id;
        grade student_grade;
};
```

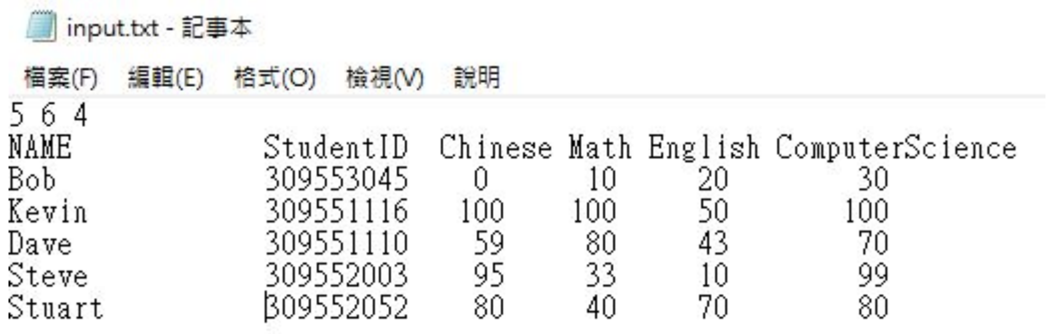
Exercise1

Create a grade alert system

- Given the input.txt (like example), please input all the contents of the txt file to create a table of student exam scores.
  - First line is number of students, number of attribute, and number of the queries.
- User inputs a subject and a grade score (a query), we need to **output all students id** that their grade of that subject **less than score**.
- If there are **no student** needed to be alerted, then **nothing** need to print.

Exercise1

- Example of the content in input.txt

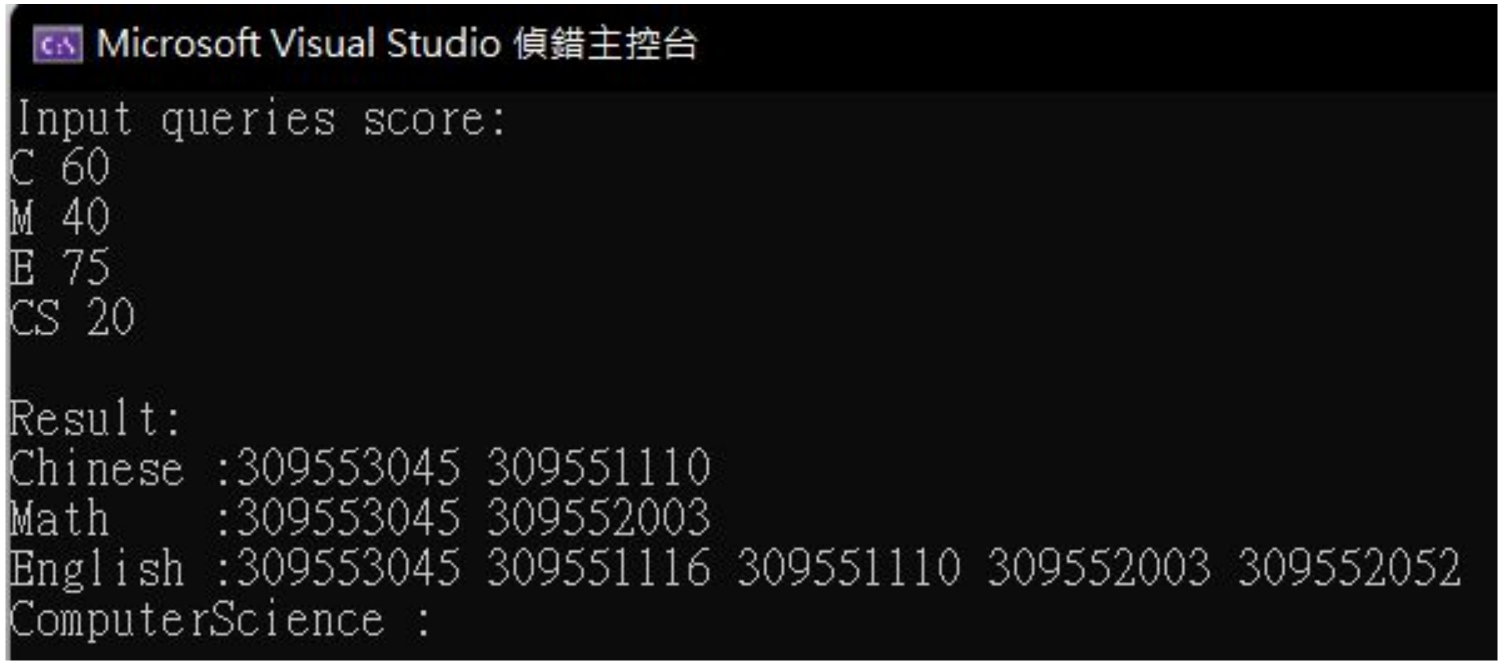


```
5 6 4
NAME            StudentID  Chinese Math English ComputerScience
Bob             309553045    0      10    20         30
Kevin           309551116   100    100    50        100
Dave            309551110    59     80    43         70
Steve           309552003    95     33    10         99
Stuart          309552052    80     40    70         80
```

- Queries
  - C : Chinese, M : Math, E : English, CS : ComputerScience

Exercise1

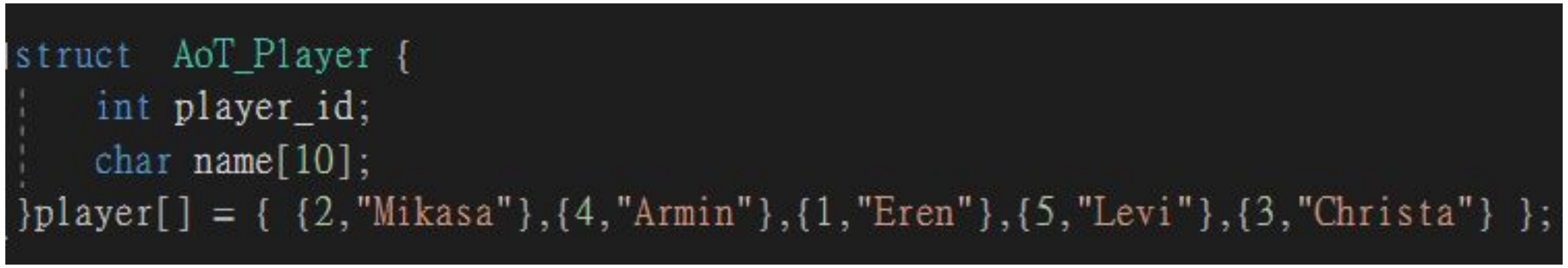
- Sample Output :



```
Input queries score:
C 60
M 40
E 75
CS 20

Result:
Chinese :309553045 309551110
Math    :309553045 309552003
English :309553045 309551116 309551110 309552003 309552052
ComputerScience :
```

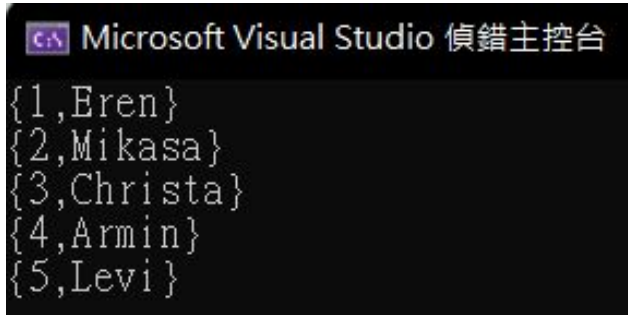
Exercise2

- Please define a structure named **AoT_Player with player_id and name**.
- Please design a program using the **Bubble sort** method to output the information of all students on the screen in ascending order of student ID.



```
struct  AoT_Player {
    int player_id;
    char name[10];
}player[] = { {2,"Mikasa"},{4,"Armin"},{1,"Eren"},{5,"Levi"},{3,"Christa"} };
```

Output :



Microsoft Visual Studio 偵錯主控台

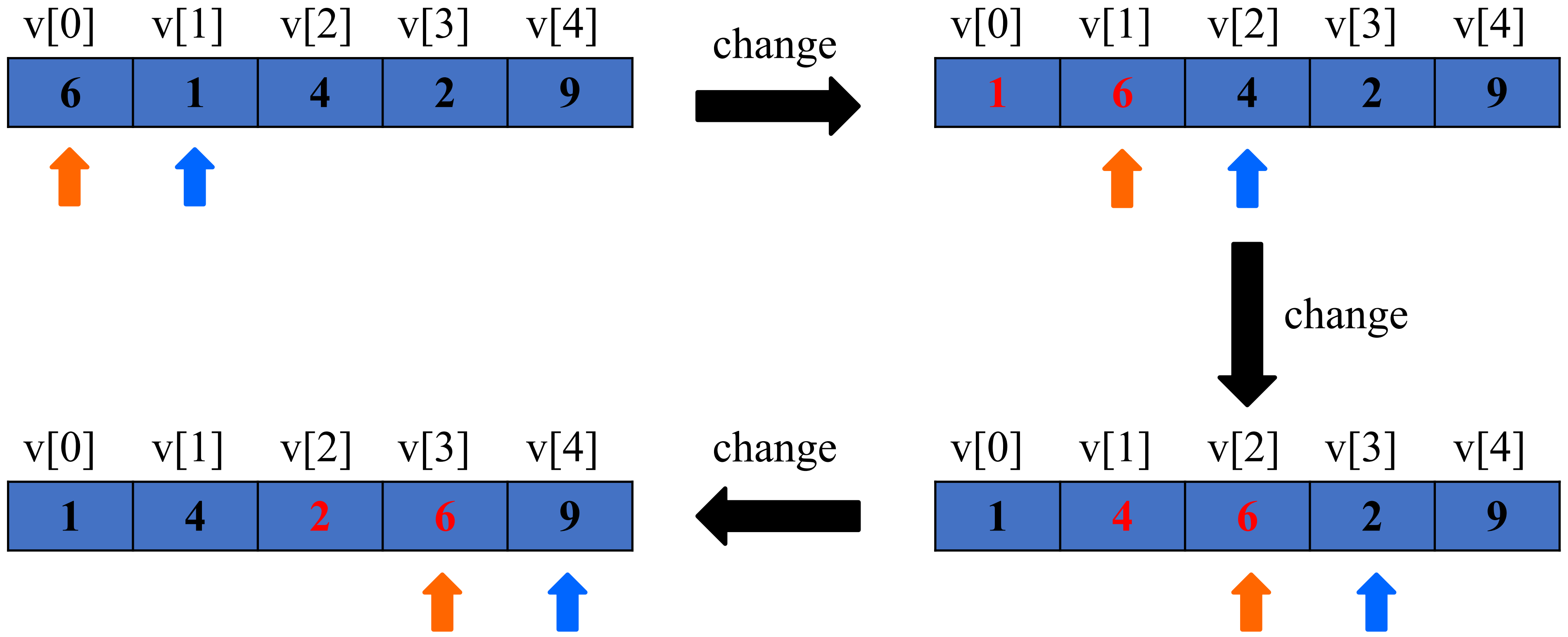
```
{1,Eren}
{2,Mikasa}
{3,Christa}
{4,Armin}
{5,Levi}
```

Reference: https://www.geeksforgeeks.org/bubble-sort/

Appendix

- Bubble sort □ Ascending order (small □ large)

Round1

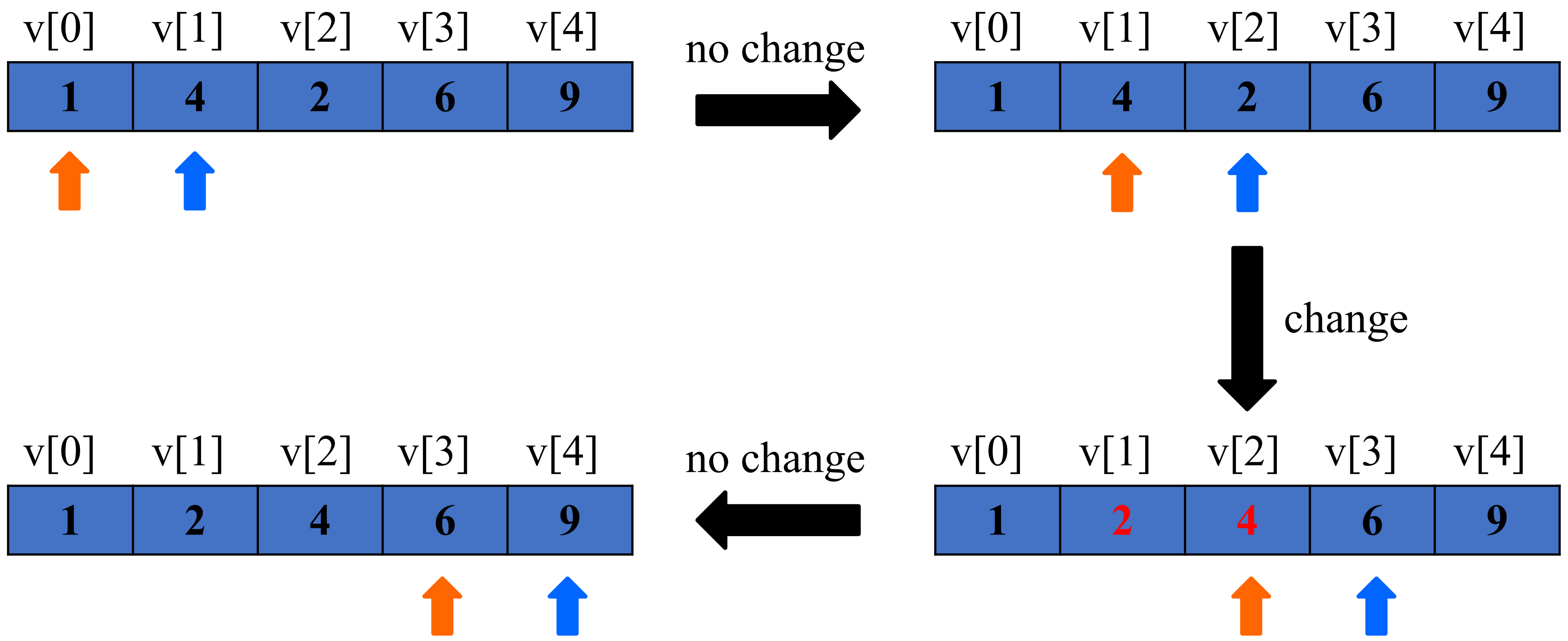


Reference: https://www.geeksforgeeks.org/bubble-sort/

Appendix

- Bubble sort □ Ascending order (small □ large)

Round2



v[0] v[1] v[2] v[3] v[4]
1 4 2 6 9
no change
v[0] v[1] v[2] v[3] v[4]
1 4 2 6 9
change
v[0] v[1] v[2] v[3] v[4]
1 2 4 6 9
no change
v[0] v[1] v[2] v[3] v[4]
1 2 4 6 9