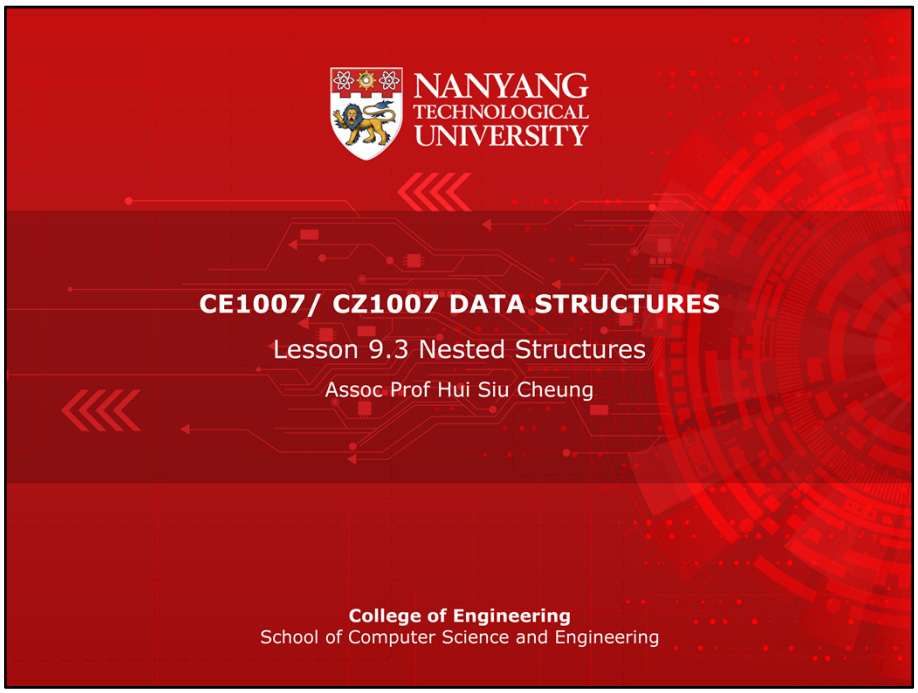
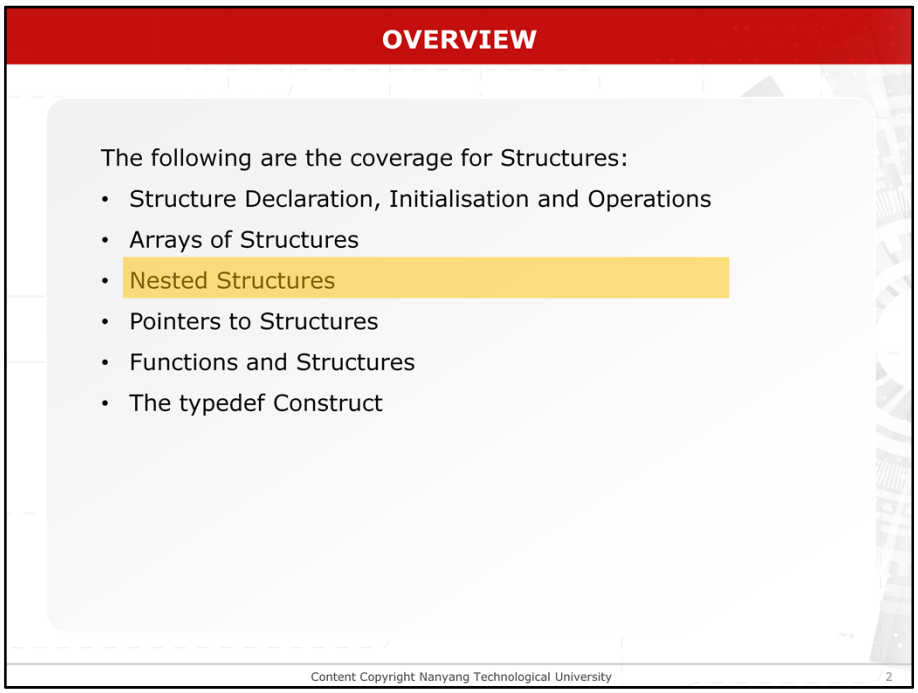
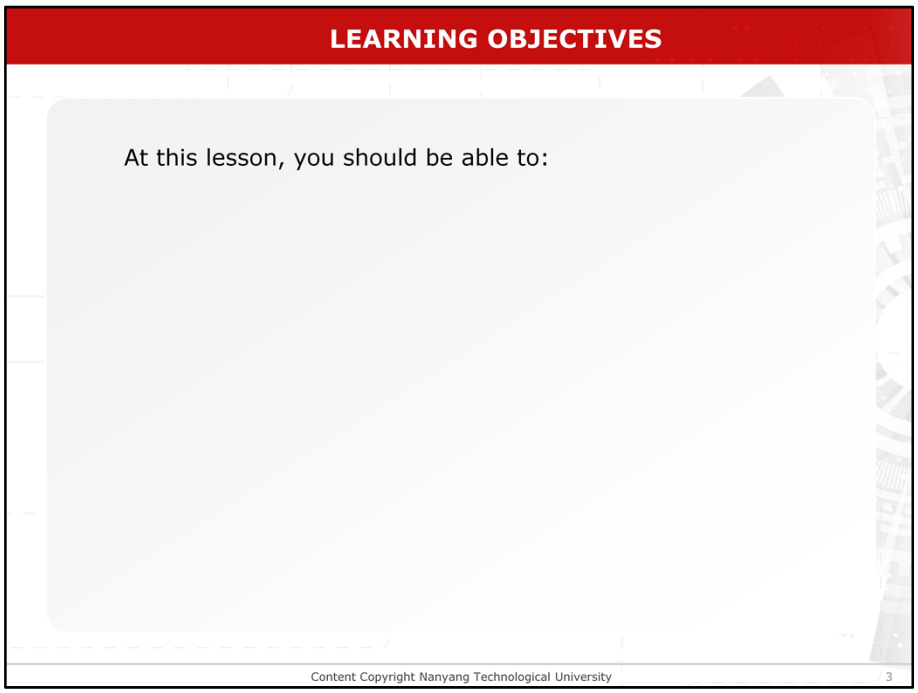
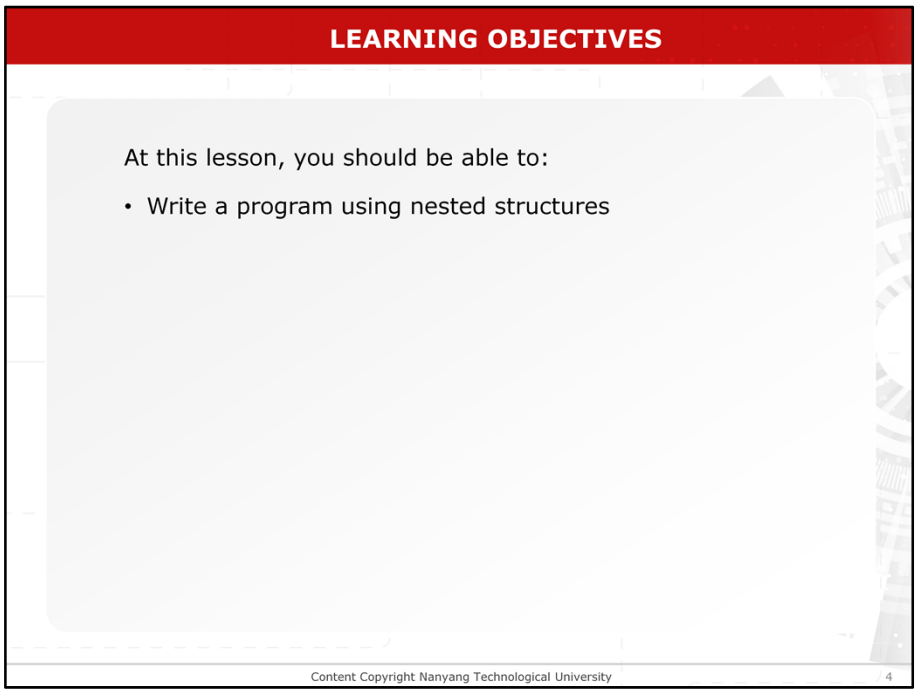

NANYANG
TECHNOLOGICAL
UNIVERSITY
CE1007/ CZ1007 DATA STRUCTURES
Lesson 9.3 Nested Structures
Assoc Prof Hui Siu Cheung
College of Engineering
School of Computer Science and Engineering



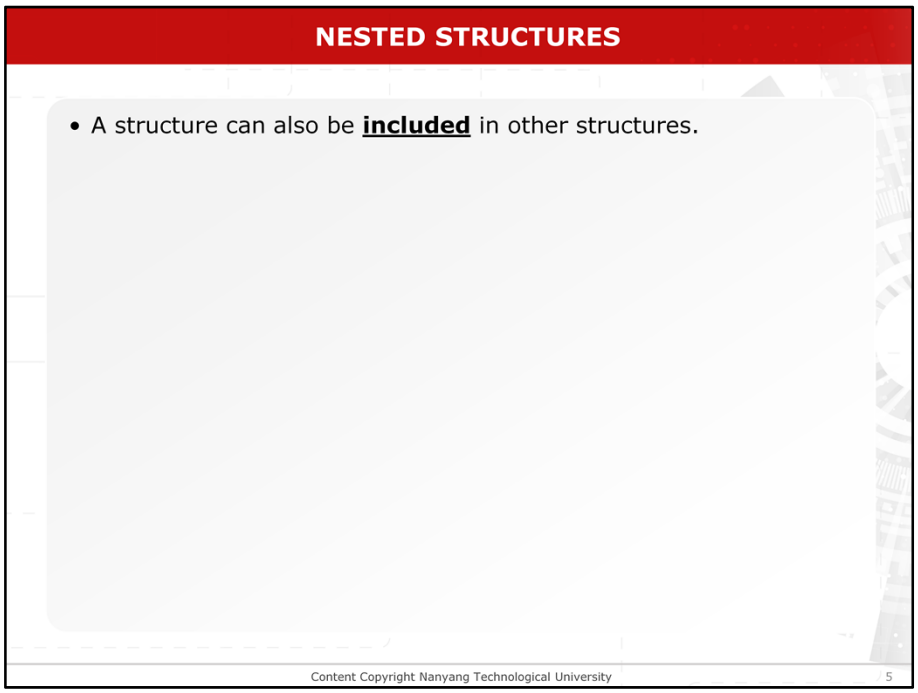
The following are the coverage for Structures: this video focusses on Nested structures



LEARNING OBJECTIVES: At this lesson, you should be able to:

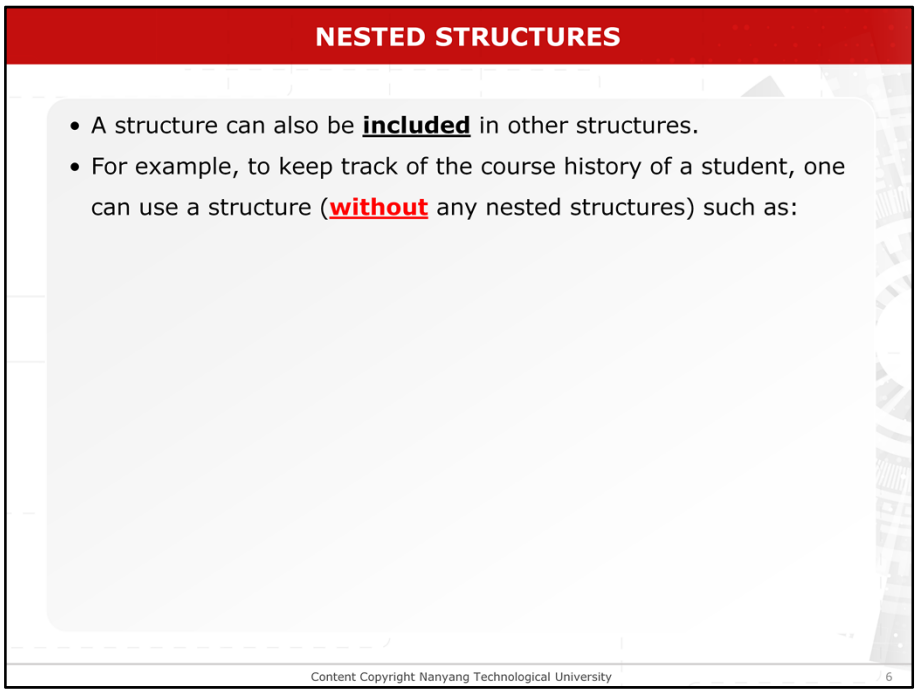


Write a program using Nested structures



Nested Structures

A structure can also be included in other structures.



For example, to keep track of the course history of a student, one can use a structure (without any nested structures) as follows:

NESTED STRUCTURES

- A structure can also be **included** in other structures.
- For example, to keep track of the course history of a student, one can use a structure (**without** any nested structures) such as:

```
struct studentTag { // without any nested structures
    char name[40];
    char id[20];
    char tel[20];
    int   SC101Yr;  /* the year when SC101 is taken * /
    int   SC101Sr;  /* the semester when SC101 is taken */
    char SC101Grade;  /* the grade obtained for SC101 */
    int   SC102Yr;  /* the year when SC102 is taken */
    int   SC102Sr;  /* the semester when SC102 is taken */
    char SC102Grade;  /* the grade obtained for SC102 */
};
 struct studentTag student[1000];
```

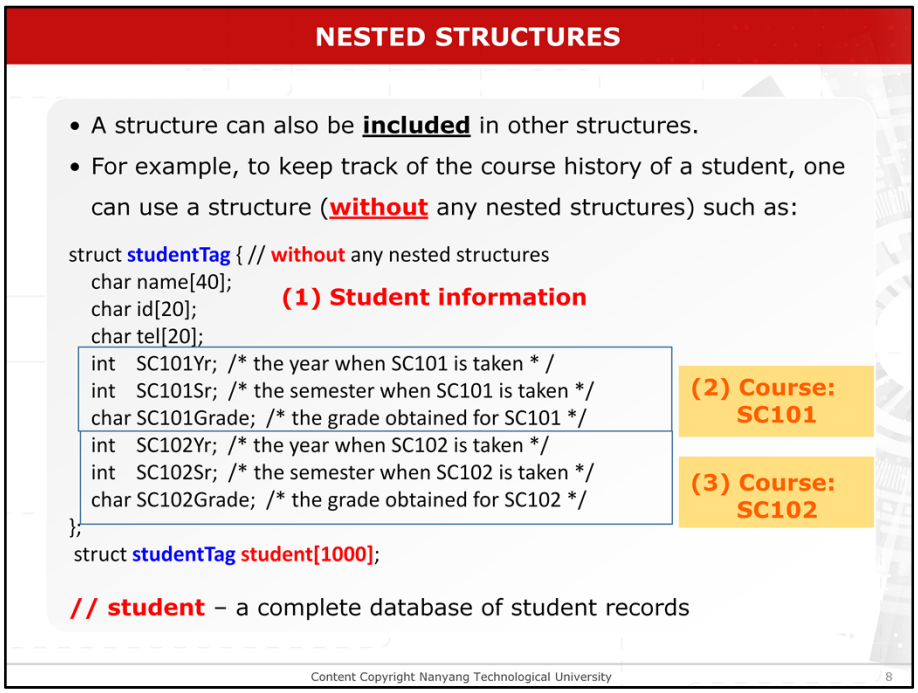
(1) Student information

(2) Course: SC101

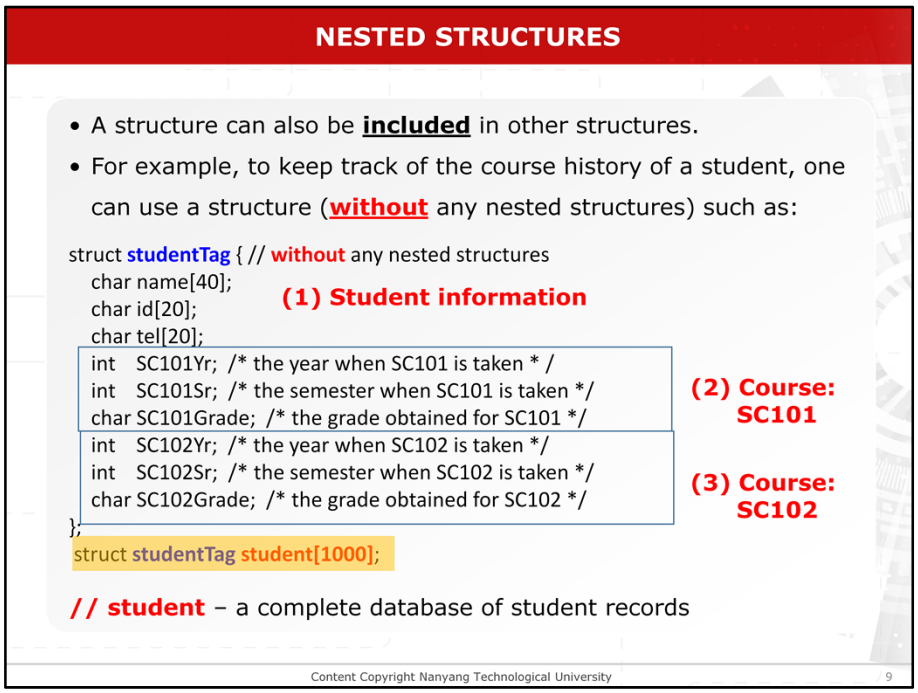
(3) Course: SC102

// student – a complete database of student records

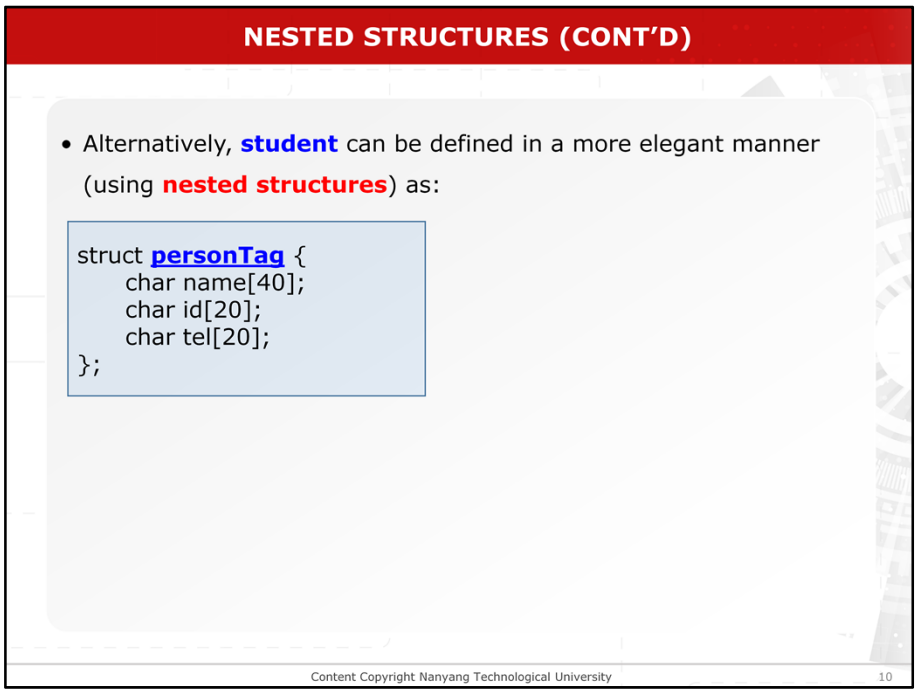
In the structure template definition **struct studentTag**, the members are the student information including **name**, **identity** and **telephone**



In addition, it also includes the courses that are taken by the student.

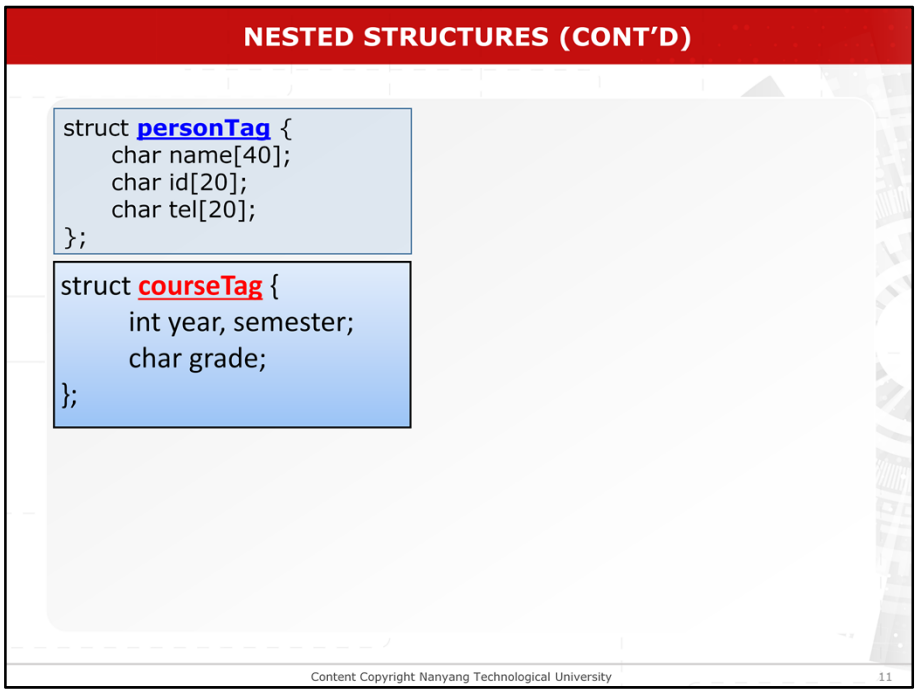


Once the **studentTag** is defined, an array variable **student** of 1000 elements is created.



Nested Structures

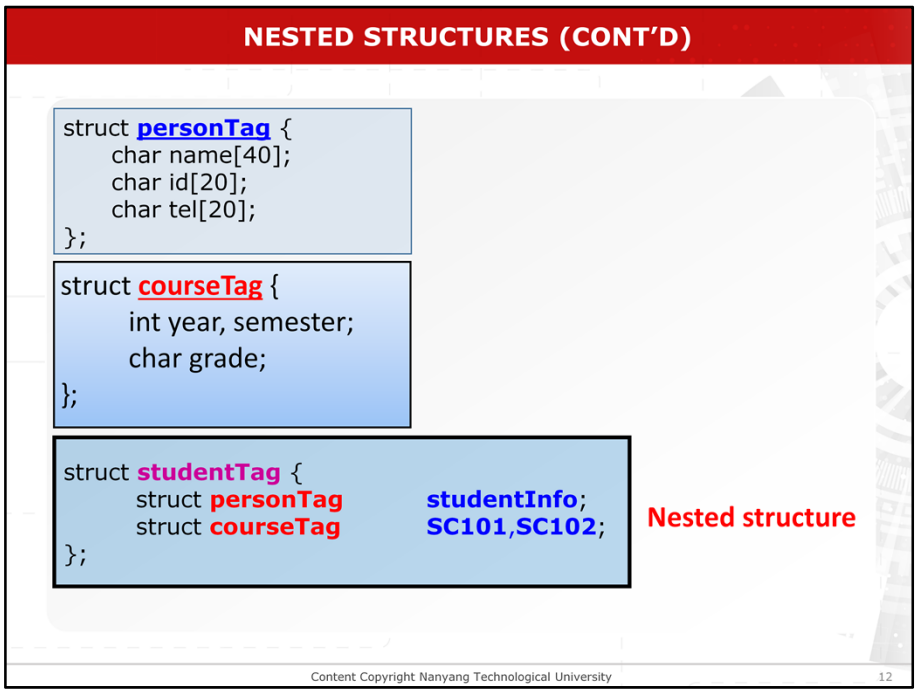
Alternatively, the variable **student** can be defined in a more elegant manner using nested structures.



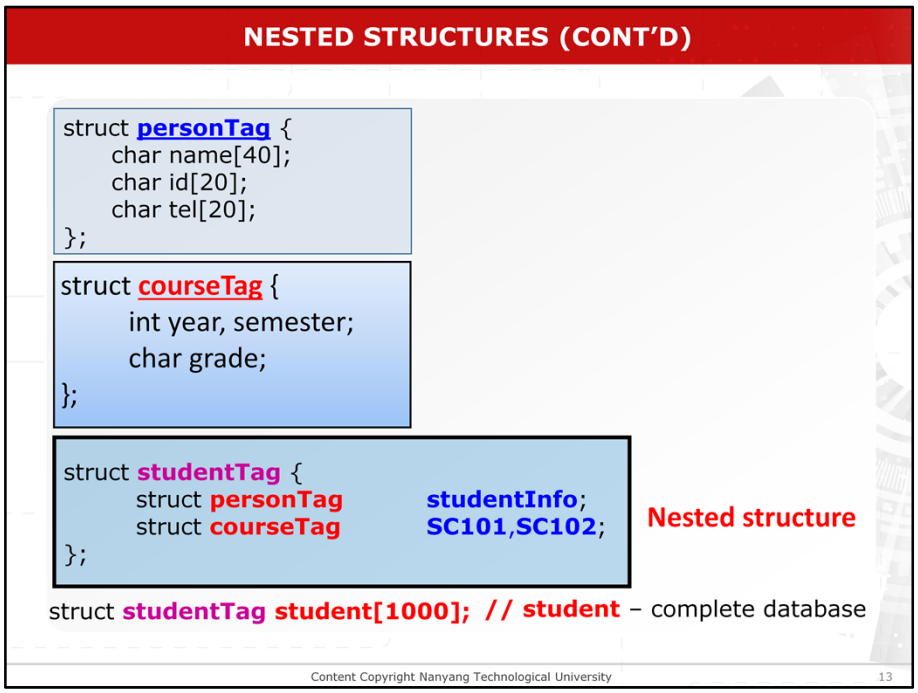
~~Nested Structures~~

We also create a structure template called **courseTag** to contain the course information as follows:

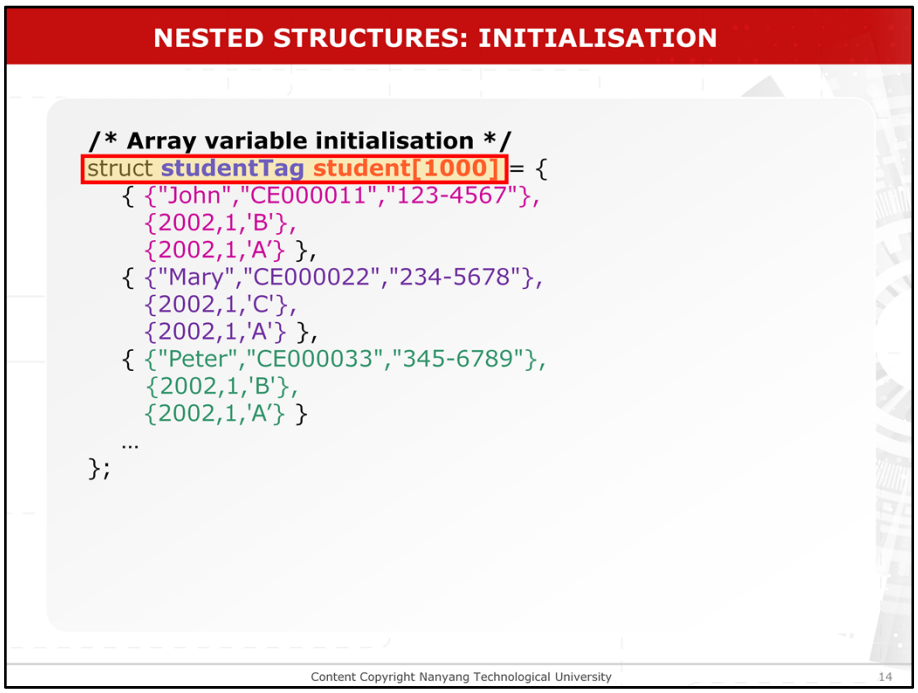
The structure **courseTag** has three members, namely **year** and **semester** of type **int**, and **grade** of type **char**.



We define the nested structure **studentTag** as shown here.
Note that the structure definition of **personTag** and **courseTag** must appear before the definition of structure **studentTag**.



Finally, we define the variable **student** as shown here.



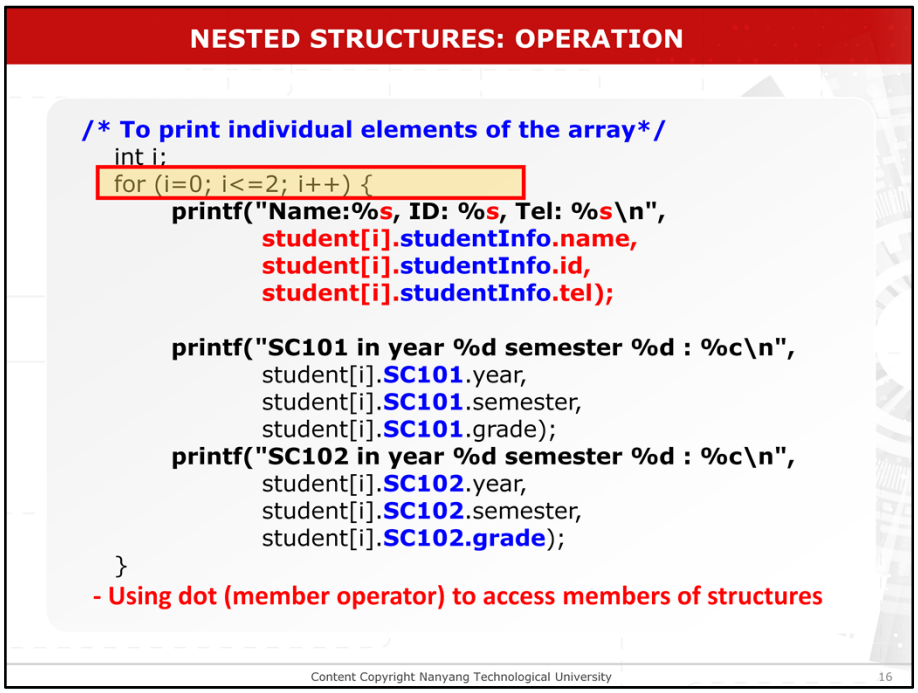
Nested Structures: Initialization

In this program, after defining the nested structure **studentTag** and the array of structures variable **student**,

NESTED STRUCTURES: INITIALISATION

```
/* Array variable initialisation */
struct studentTag student[1000] = {
   { {"John","CE000011","123-4567"},
     {2002,1,'B'},
     {2002,1,'A'} },
   { {"Mary","CE000022","234-5678"},
     {2002,1,'C'},
     {2002,1,'A'} },
   { {"Peter","CE000033","345-6789"},
     {2002,1,'B'},
     {2002,1,'A'} }
   ...
};
```


we initialize the variable **student** with initial data.



Nested Structures: Example

To access each array element, we use a **for** loop to traverse the array.

NESTED STRUCTURES: OPERATION

```
/* To print individual elements of the array*/
   int i;
   for (i=0; i<=2; i++) {
        printf("Name:%s, ID: %s, Tel: %s\n",
                student[i].studentInfo.name,
                student[i].studentInfo.id,
                student[i].studentInfo.tel);

        printf("SC101 in year %d semester %d : %c\n",
                student[i].SC101.year,
                student[i].SC101.semester,
                student[i].SC101.grade);
        printf("SC102 in year %d semester %d : %c\n",
                student[i].SC102.year,
                student[i].SC102.semester,
                student[i].SC102.grade);
   }
```

- Using dot (member access operator) to access members of structures

The array notation and member operator are used for accessing each array element and structure member

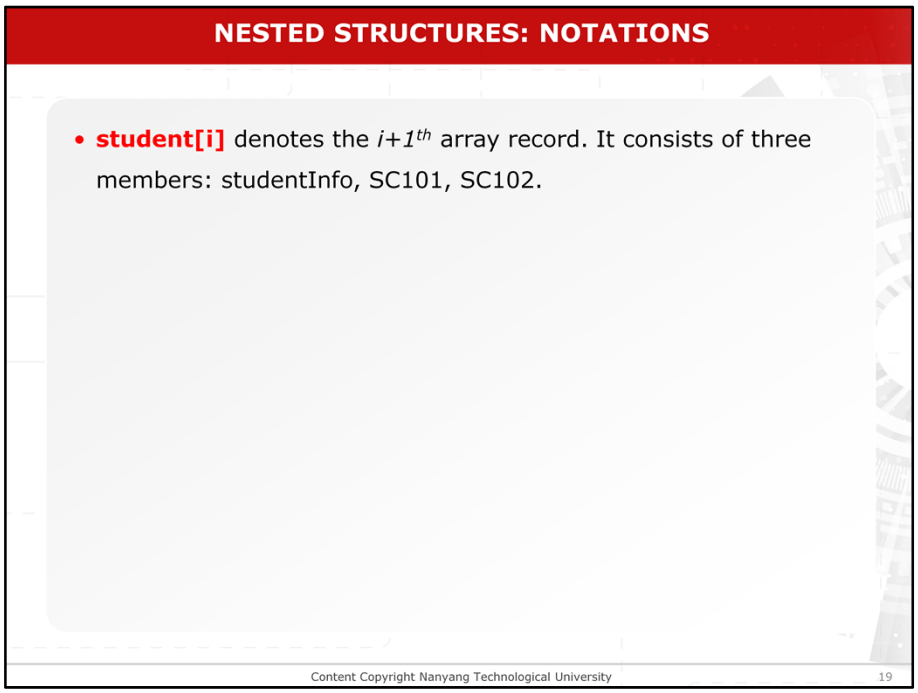
NESTED STRUCTURES: OPERATION

```
/* To print individual elements of the array*/
   int i;
   for (i=0; i<=2; i++) {
        printf("Name:%s, ID: %s, Tel: %s\n",
                student[i].studentInfo.name,
                student[i].studentInfo.id,
                student[i].studentInfo.tel);

        printf("SC101 in year %d semester %d : %c\n",
                student[i].SC101.year,
                student[i].SC101.semester,
                student[i].SC101.grade);
        printf("SC102 in year %d semester %d : %c\n",
                student[i].SC102.year,
                student[i].SC102.semester,
                student[i].SC102.grade);
   }
```

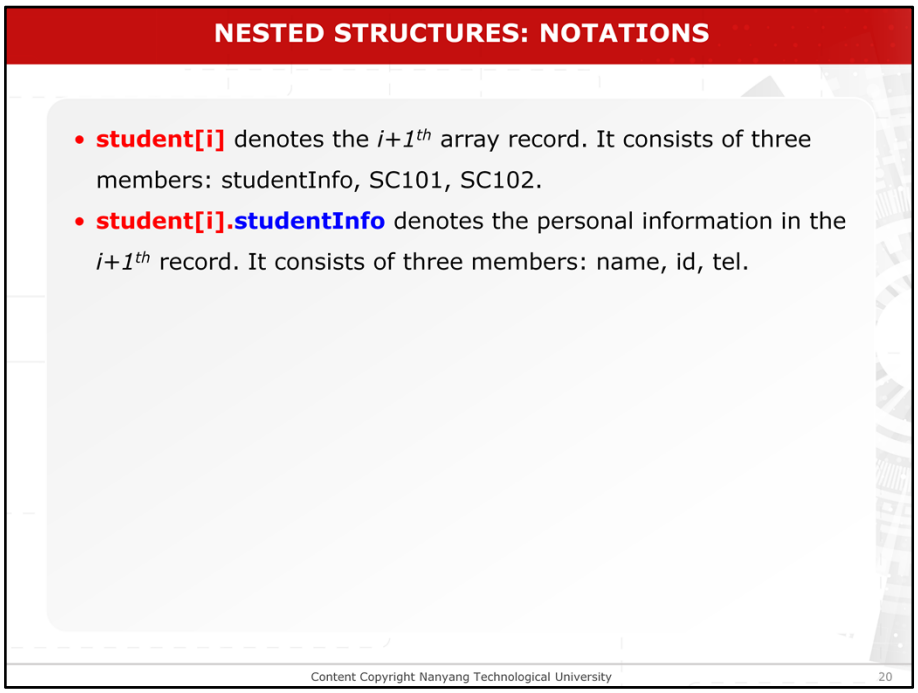
- Using dot (member access operator) to access members of structures

The data can then be processed and printed on the screen.

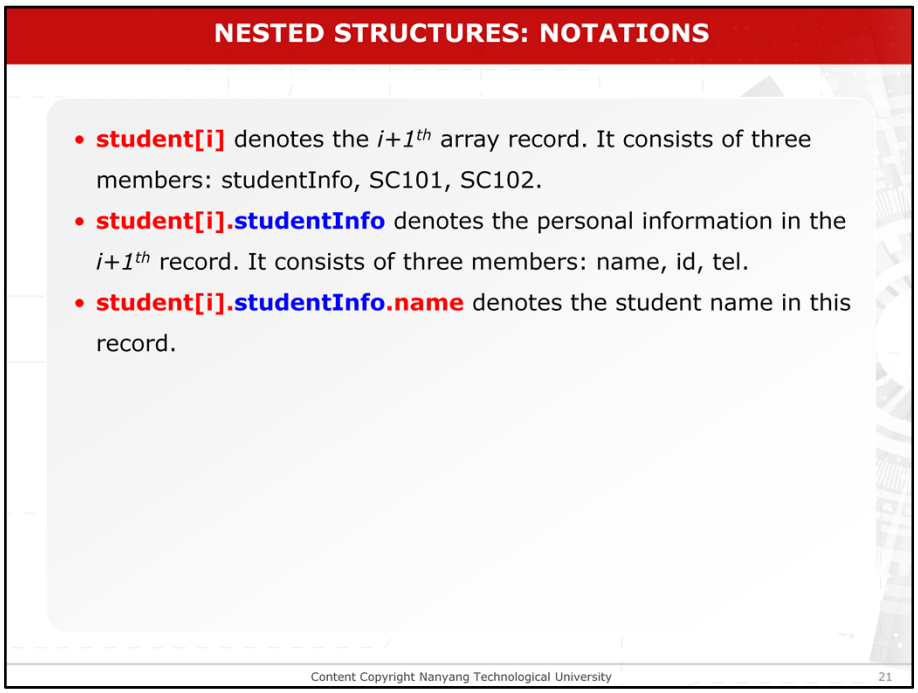


Nested Structures: Notations

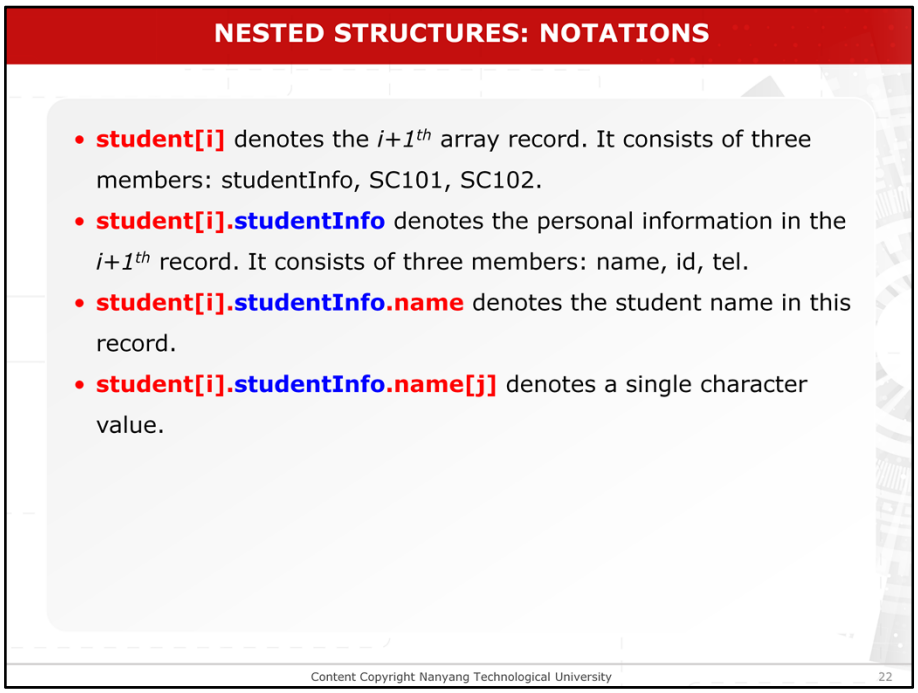
This statement denotes the *I plus oneth* array record.



- This statement denotes the personal information in the I plus oneth record; It consists of three members: name, id, tel.

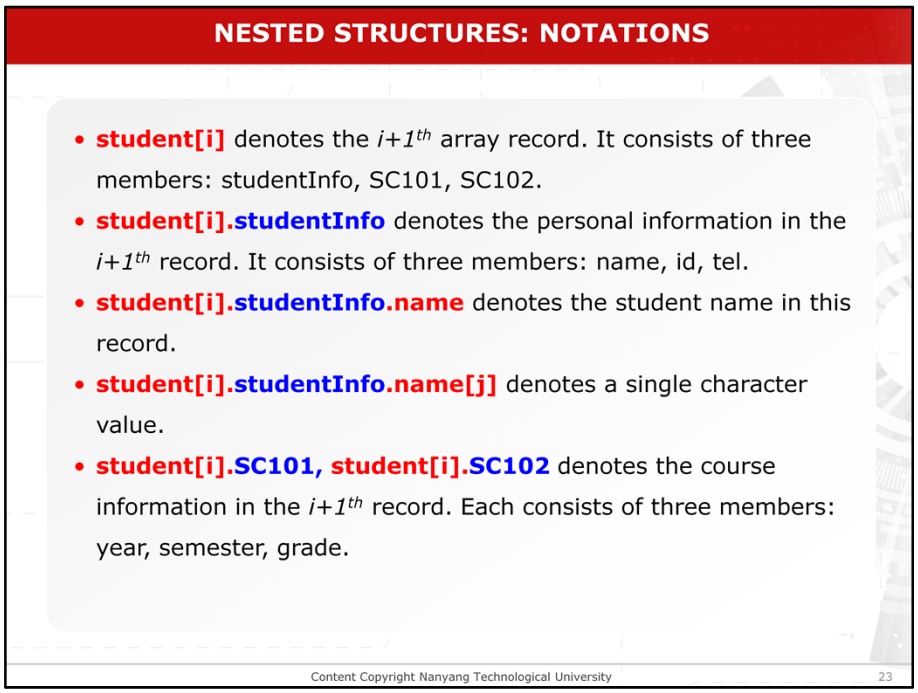


- **This statement** denotes the student name in this record

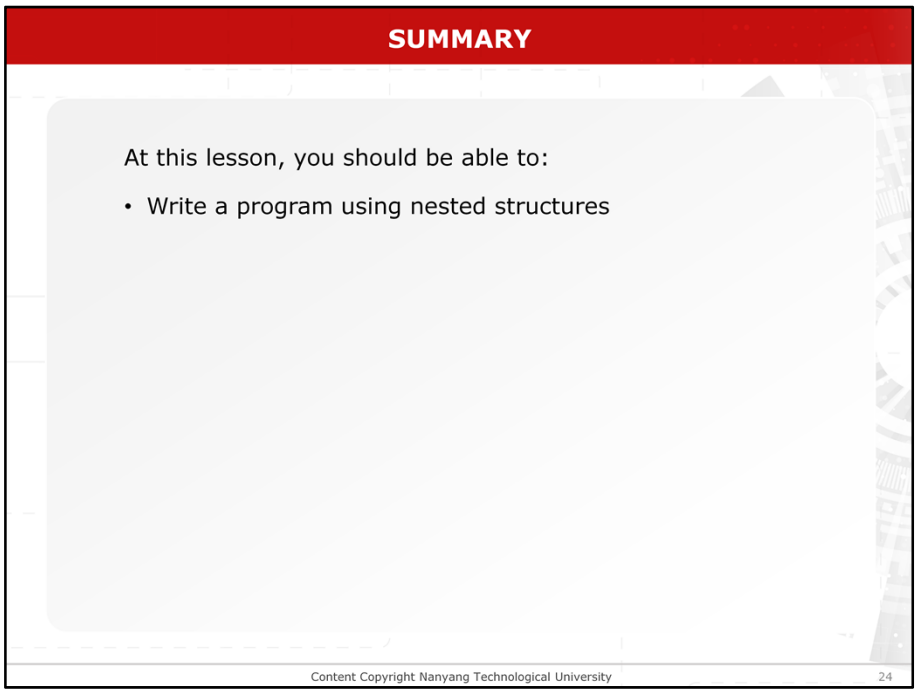


Nested Structures: Notations

This statement denotes a single character value.



This statement denotes the course information in the I plus one record.



After watching this video lesson, you should be able to do the listed.