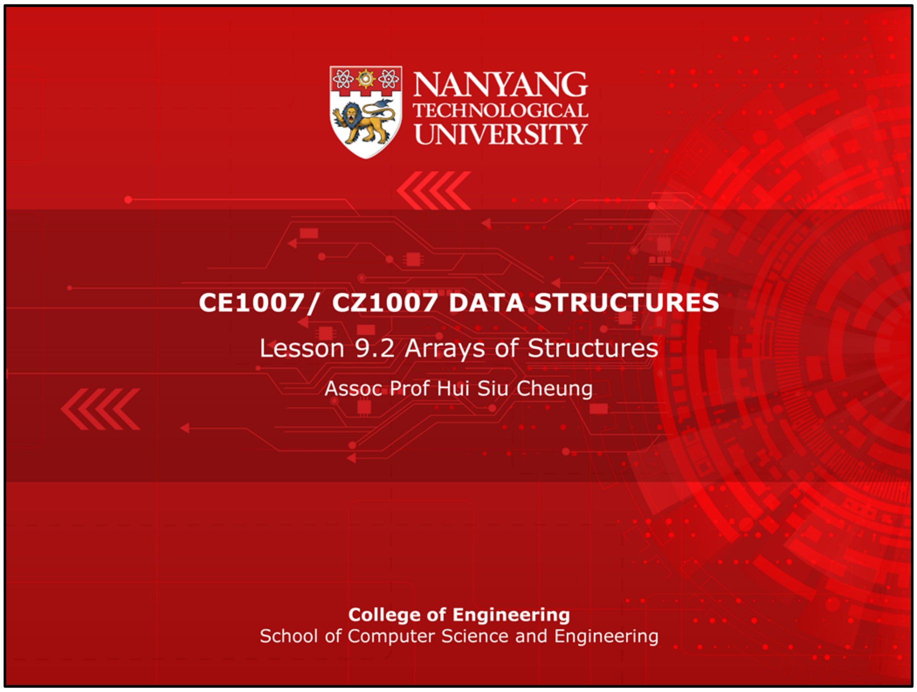


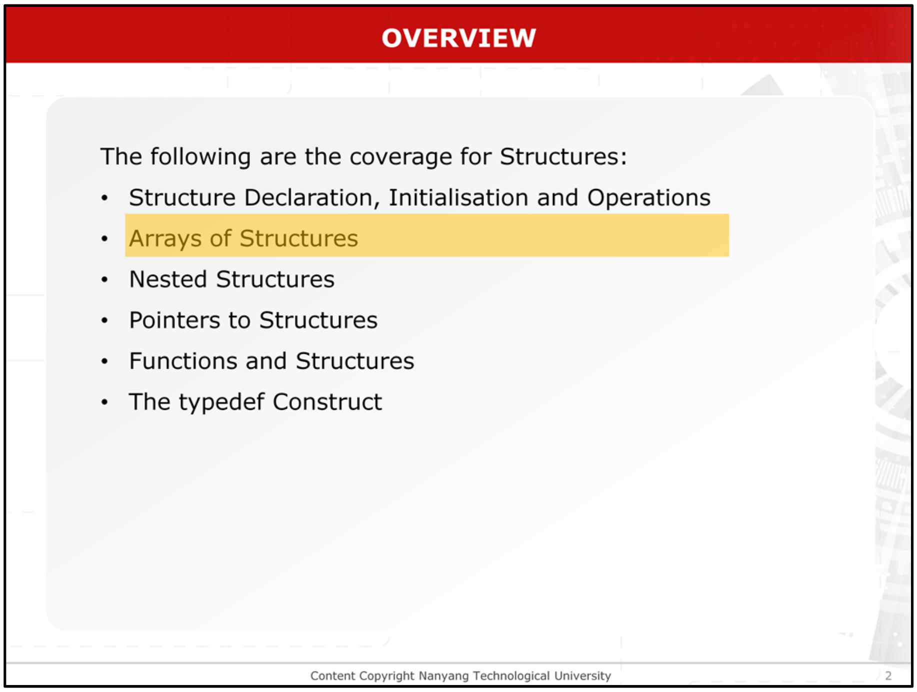
NANYANG TECHNOLOGICAL UNIVERSITY

CE1007/ CZ1007 DATA STRUCTURES

Lesson 9.2 Arrays of Structures

Assoc Prof Hui Siu Cheung

College of Engineering
School of Computer Science and Engineering



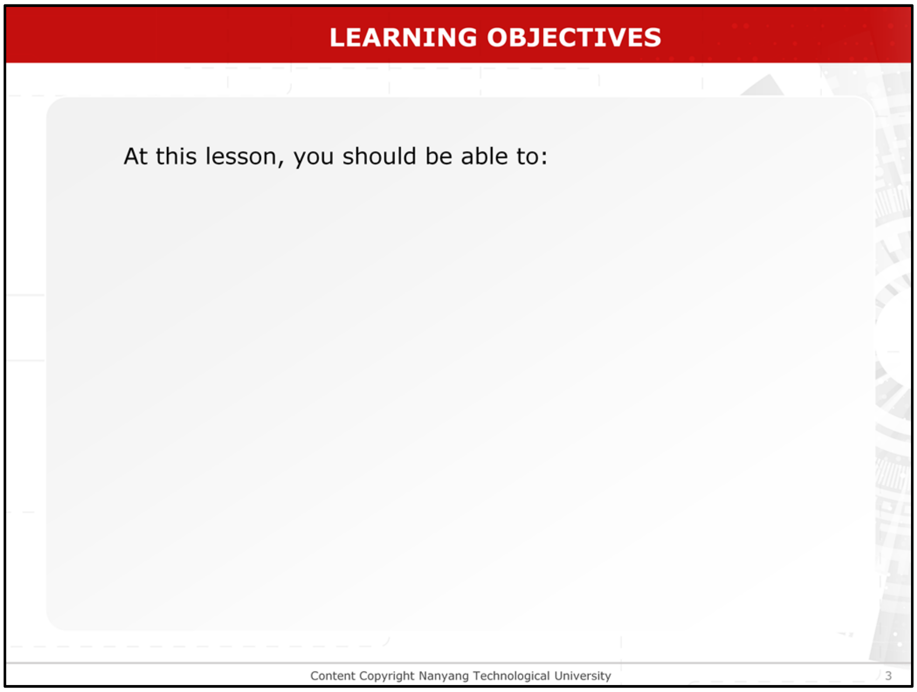
OVERVIEW

The following are the coverage for Structures:

- Structure Declaration, Initialisation and Operations
- Arrays of Structures
- Nested Structures
- Pointers to Structures
- Functions and Structures
- The typedef Construct

Content Copyright Nanyang Technological University

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

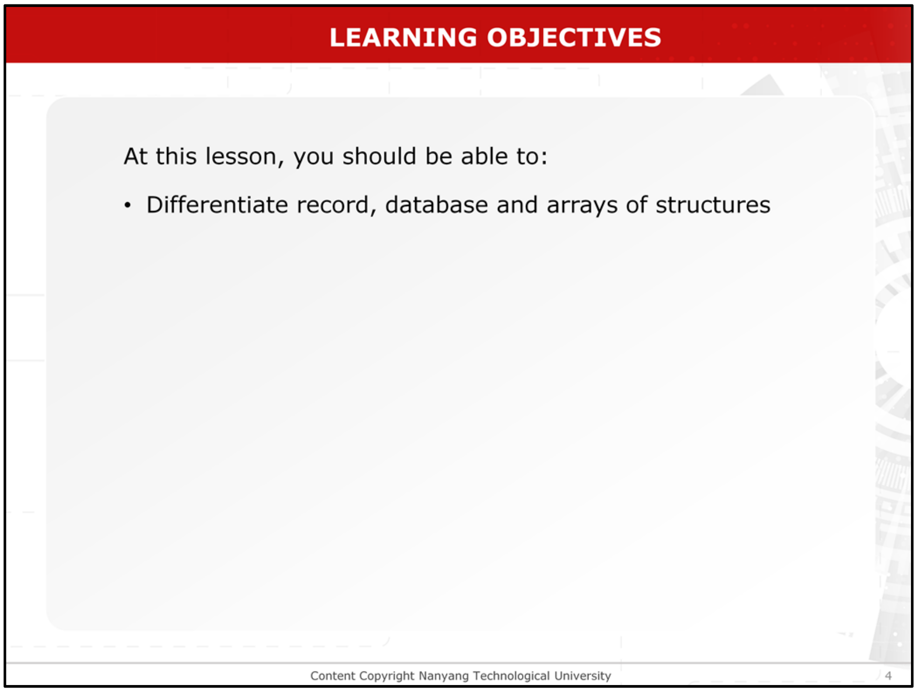


LEARNING OBJECTIVES

At this lesson, you should be able to:

Content Copyright Nanyang Technological University

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



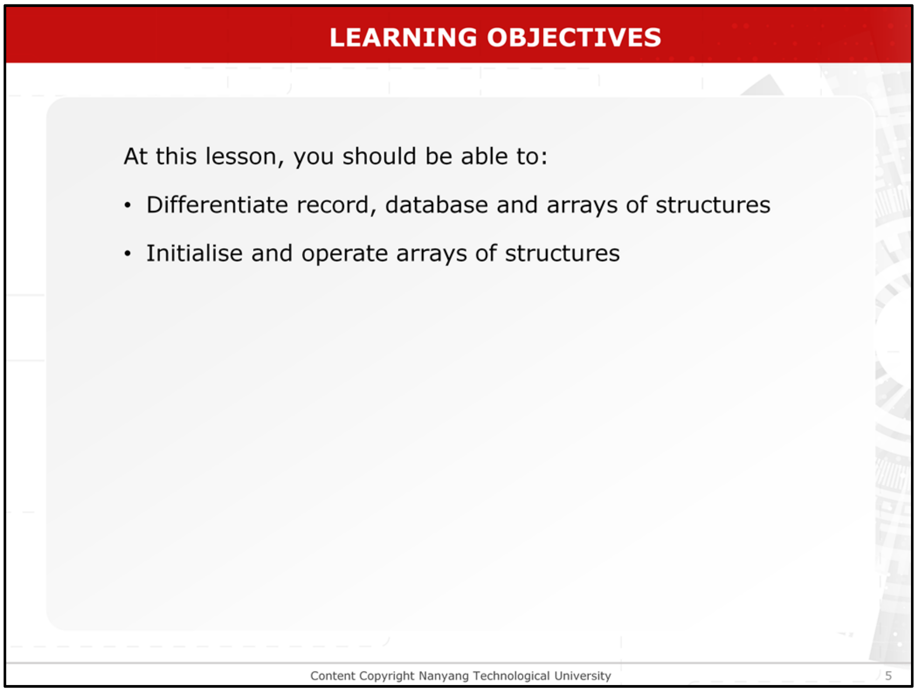
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Differentiate record, database and arrays of structures

Content Copyright Nanyang Technological University

- Differentiate record, database and arrays of structures



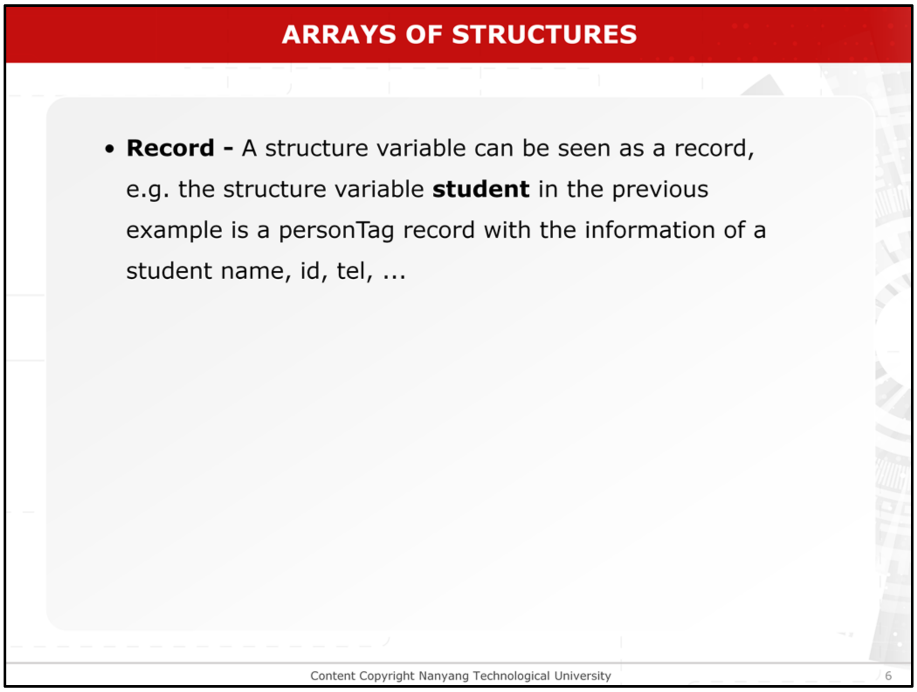
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Differentiate record, database and arrays of structures
- Initialise and operate arrays of structures

Content Copyright Nanyang Technological University

Initialise and operate arrays of structures



ARRAYS OF STRUCTURES

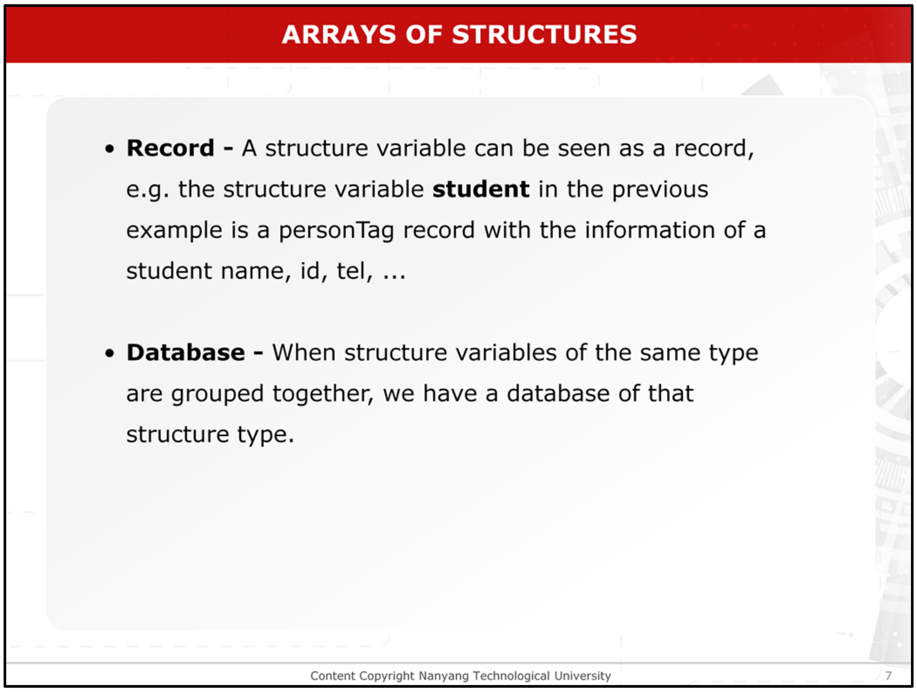
- **Record -** A structure variable can be seen as a record, e.g. the structure variable **student** in the previous example is a personTag record with the information of a student name, id, tel, …

Content Copyright Nanyang Technological University

6

Arrays of Structures

A structure variable can be seen as a record. For example, the structure variable **student** is a personTag record with the information of a student name, identity and telephone number.



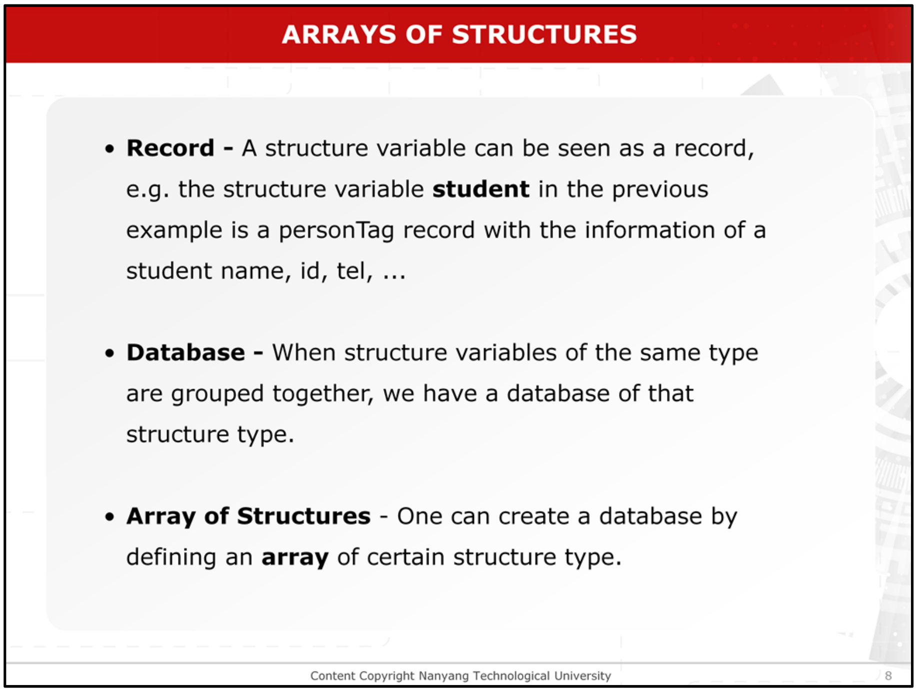
ARRAYS OF STRUCTURES

- **Record -** A structure variable can be seen as a record, e.g. the structure variable **student** in the previous example is a personTag record with the information of a student name, id, tel, …

- **Database -** When structure variables of the same type are grouped together, we have a database of that structure type.

Content Copyright Nanyang Technological University

When structure variables of the same type are grouped together, we can form a database of that structure type. Therefore, we can create a database by defining an array of structures.

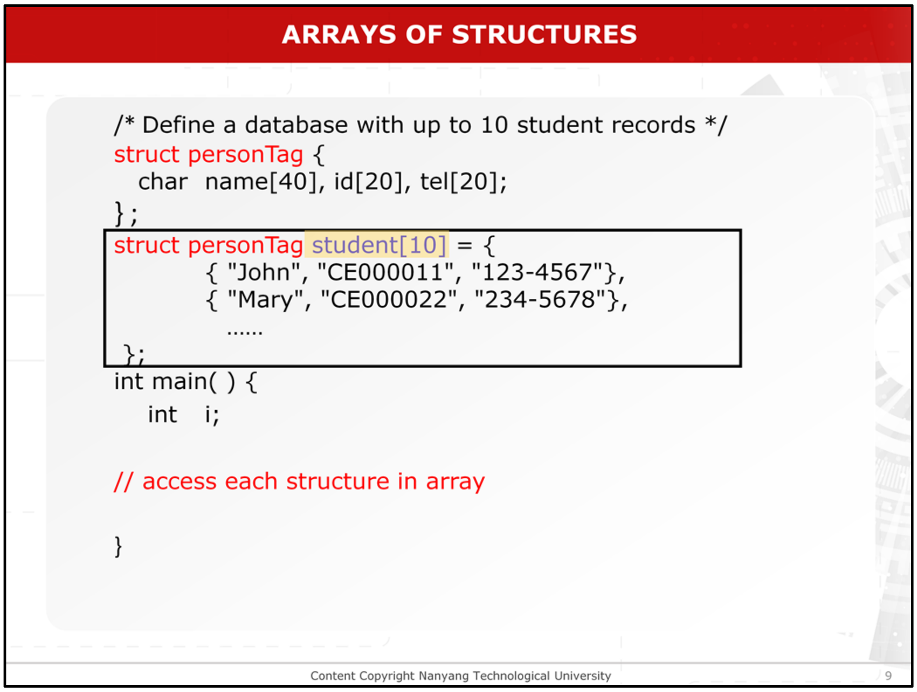


ARRAYS OF STRUCTURES

- **Record -** A structure variable can be seen as a record, e.g. the structure variable **student** in the previous example is a personTag record with the information of a student name, id, tel, …
- **Database -** When structure variables of the same type are grouped together, we have a database of that structure type.
- **Array of Structures** - One can create a database by defining an **array** of certain structure type.

Content Copyright Nanyang Technological University

Therefore, we can create a database by defining an array of structures.



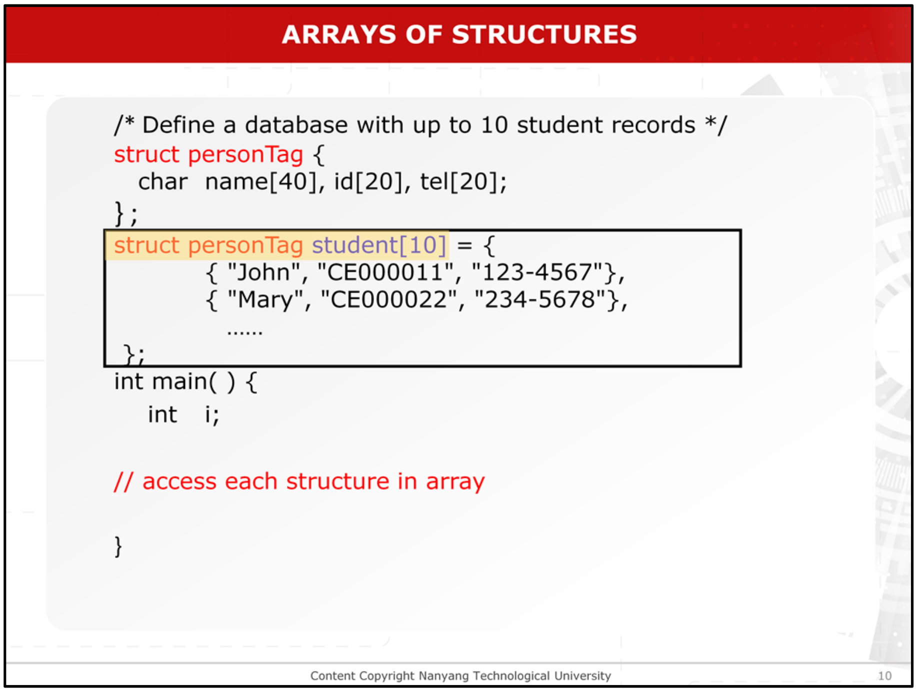
ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
  char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ......
 };
int main( ) {
   int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University 9

Arrays of Structures: Initialization

In the program, the variable **student** defines an array of structures, which is a database of student records.



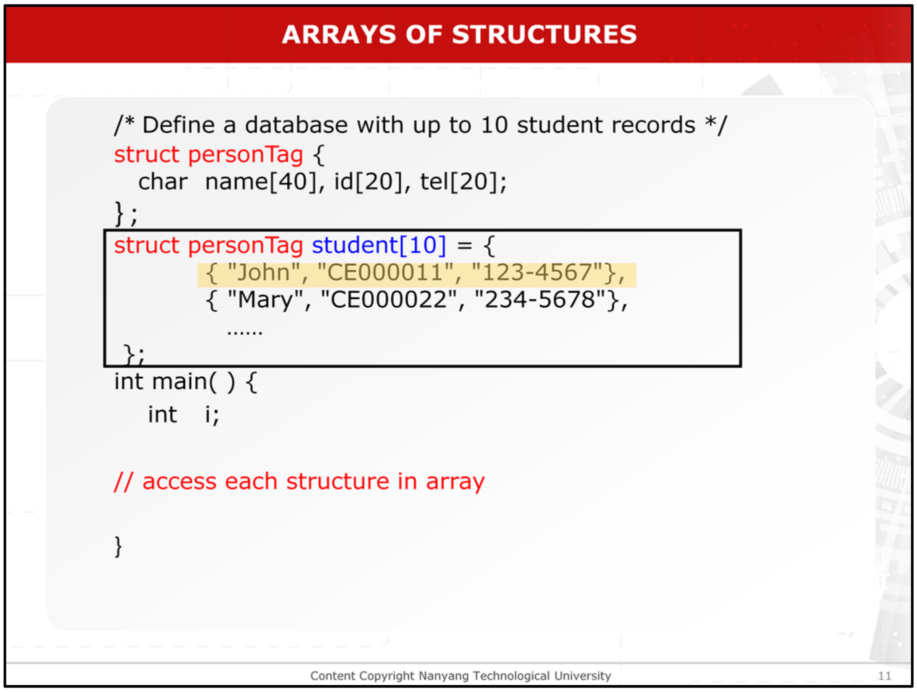
ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
  char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ......
 };
int main( ) {
   int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University

10

Each element of the array is of **struct personTag**.

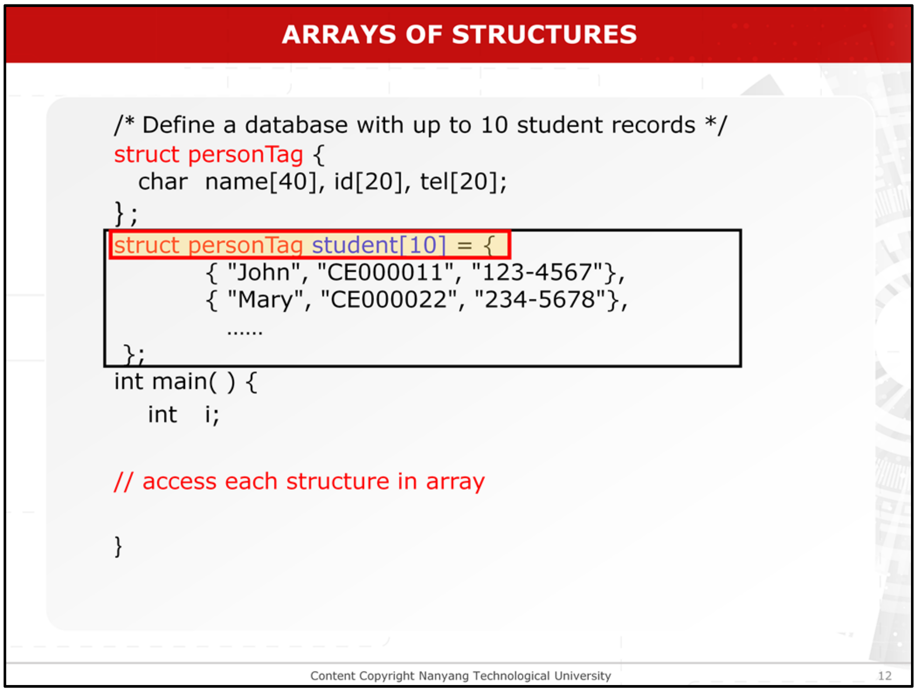


ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
   char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
         { "John", "CE000011", "123-4567"},
         { "Mary", "CE000022", "234-5678"},
           ......
 };
int main( ) {
    int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University 11

means each array element contains three members, namely **name**, **id** and **tel**, of the structure.

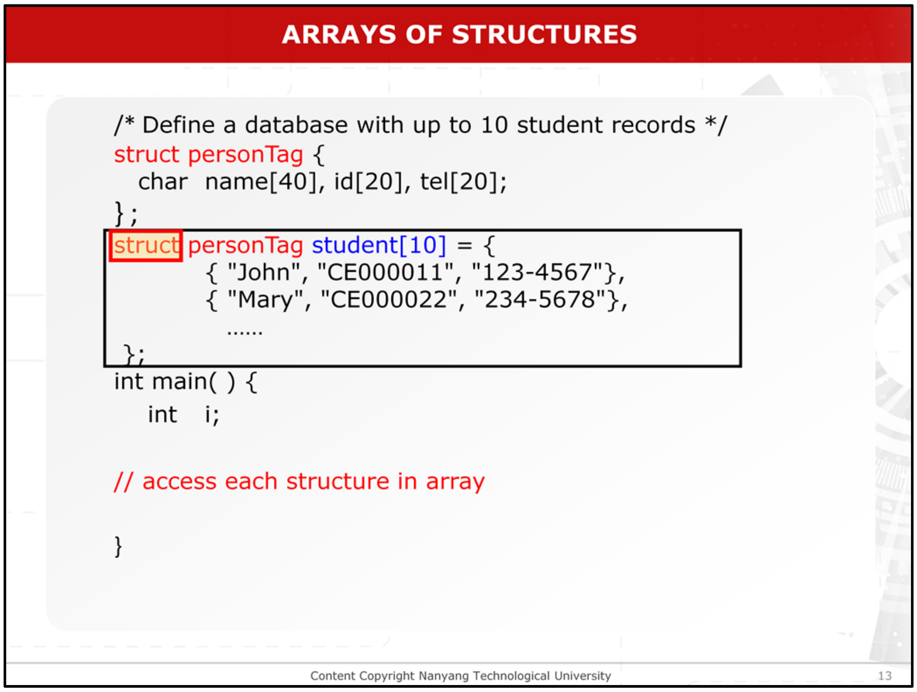


ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
  char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ……
 };
int main( ) {
   int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University 12

The syntax for declaring an array of structures is highlighted here.

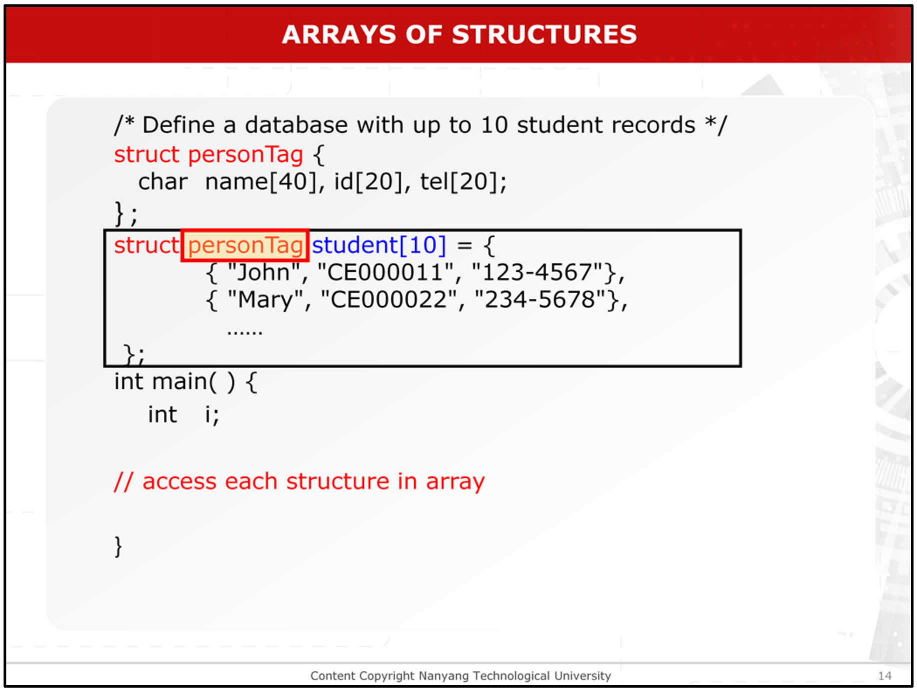


ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
  char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ......
 };
int main( ) {
   int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University

It starts with the keyword **struct**,

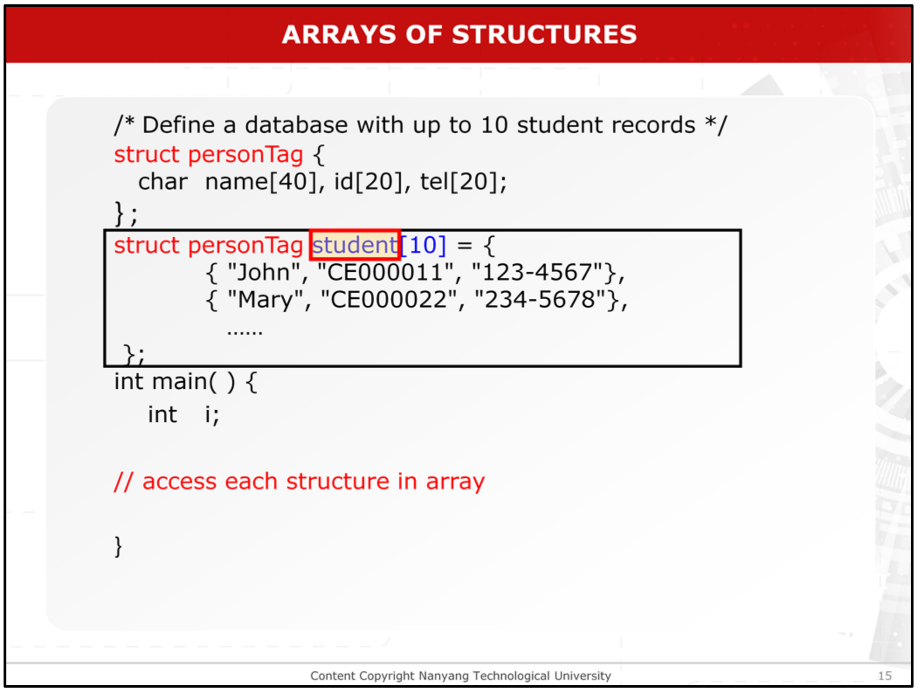


ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
  char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ……
 };
int main( ) {
   int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University 14

followed by the name of the structure **personTag** ~~that identifies the data type~~.

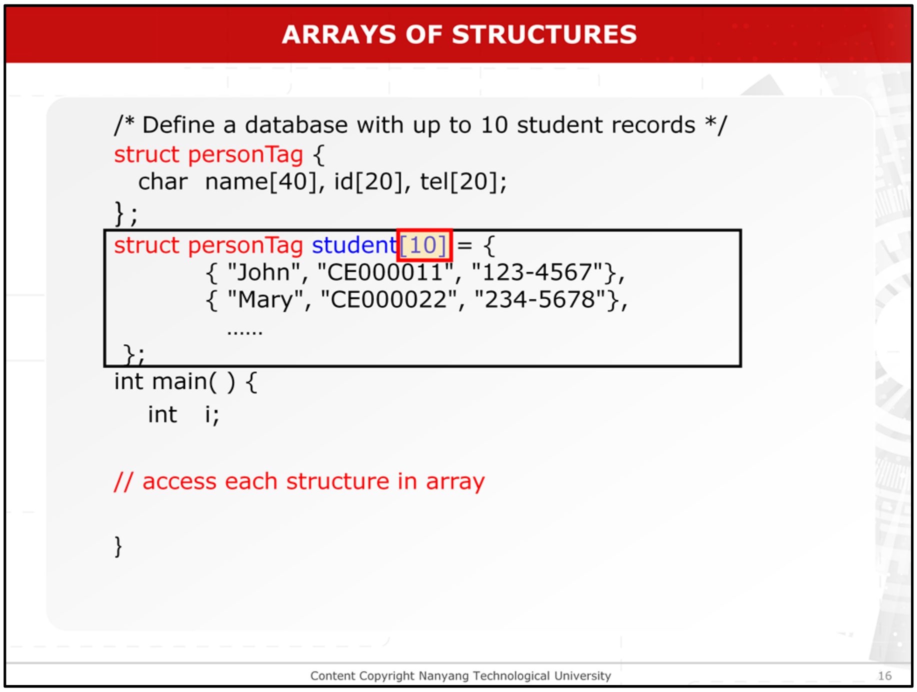


ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
  char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ……
 };
int main( ) {
   int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University

This is then followed by the name of the array, **student**.

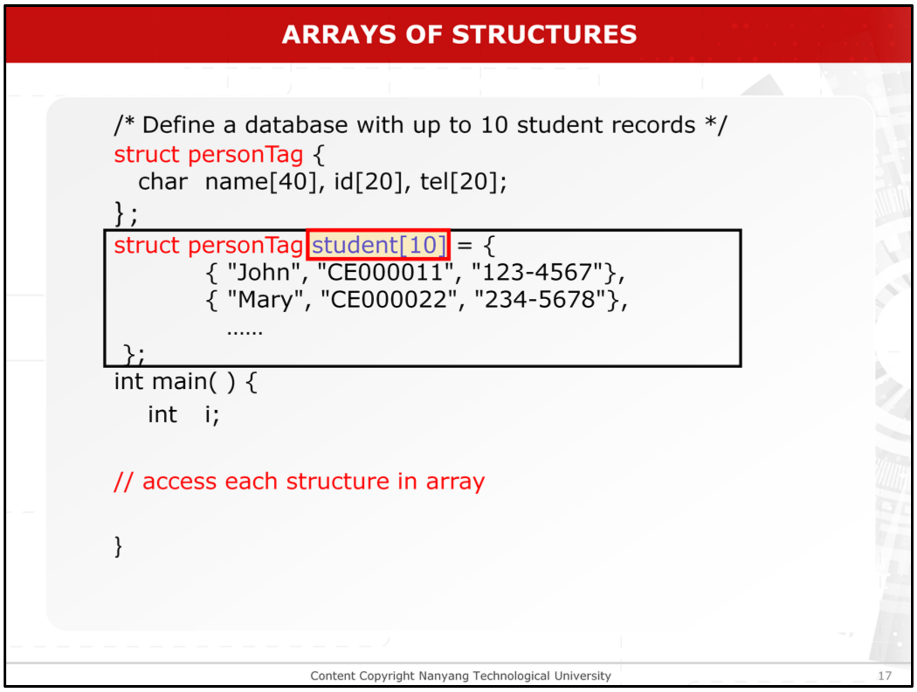


ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
   char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ......
 };
int main( ) {
    int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University 16

The values specified within the square brackets specify the total number of elements in the array.

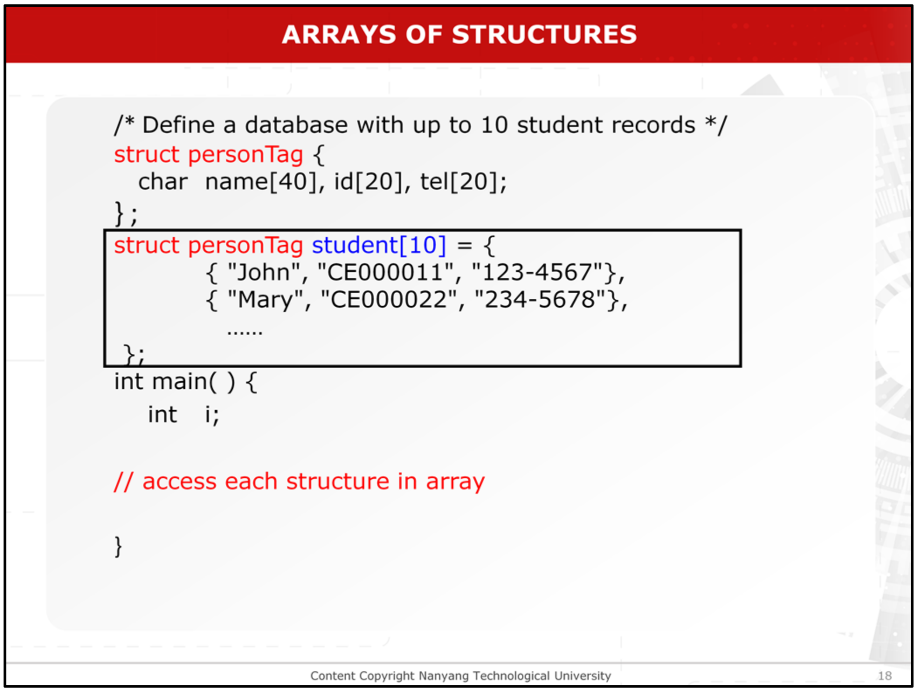


ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
  char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ……
 };
int main( ) {
   int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University 17

Array index is used when accessing individual elements of an array of structures.
We use **student[i]** to denote the (i+1)th record.



ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
   char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
        { "John", "CE000011", "123-4567"},
        { "Mary", "CE000022", "234-5678"},
          ......
 };
int main( ) {
    int   i;

// access each structure in array

}
```

Content Copyright Nanyang Technological University

The first element starts with index 0.
To access a member of a specific element, we use

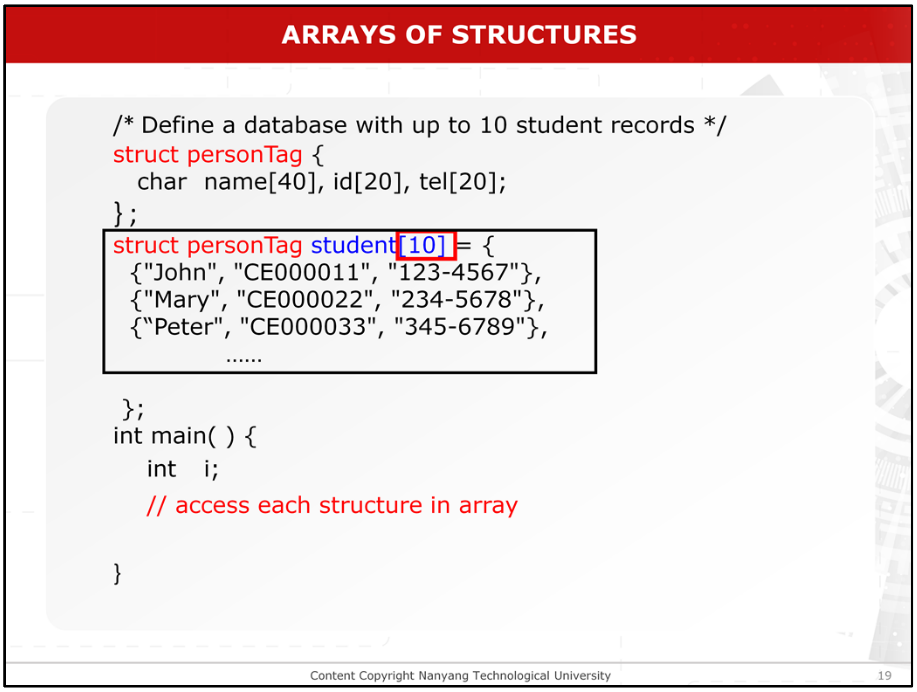
student[i].name

which denotes a member of the (i+1)th record.
We use

student[i].name[j]

to denote a single character value of a member of the (i+1)th record.
Array of structures can be initialized. The initializers for each element are enclosed in braces, and each member is separated by a comma. An example is given as follows:

```
     struct personTag student[10] = {
          {"John", "CE000011", "123-4567"},        /* initialize values for
student[0] */
          {"Mary", "CE000022", "234-5678"},        /* initialize values for
student[1] */
          {"Peter", "CE000033", "345-6789"},       /* initialize values for
student[2] */
          …
};
```



ARRAYS OF STRUCTURES

```
/* Define a database with up to 10 student records */
struct personTag {
   char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
  {"John", "CE000011", "123-4567"},
  {"Mary", "CE000022", "234-5678"},
  {“Peter", "CE000033", "345-6789"},
            ......

 };
int main( ) {
    int   i;
    // access each structure in array

}
```

Content Copyright Nanyang Technological University 19

The first element starts with index 0.
To access a member of a specific element, we use

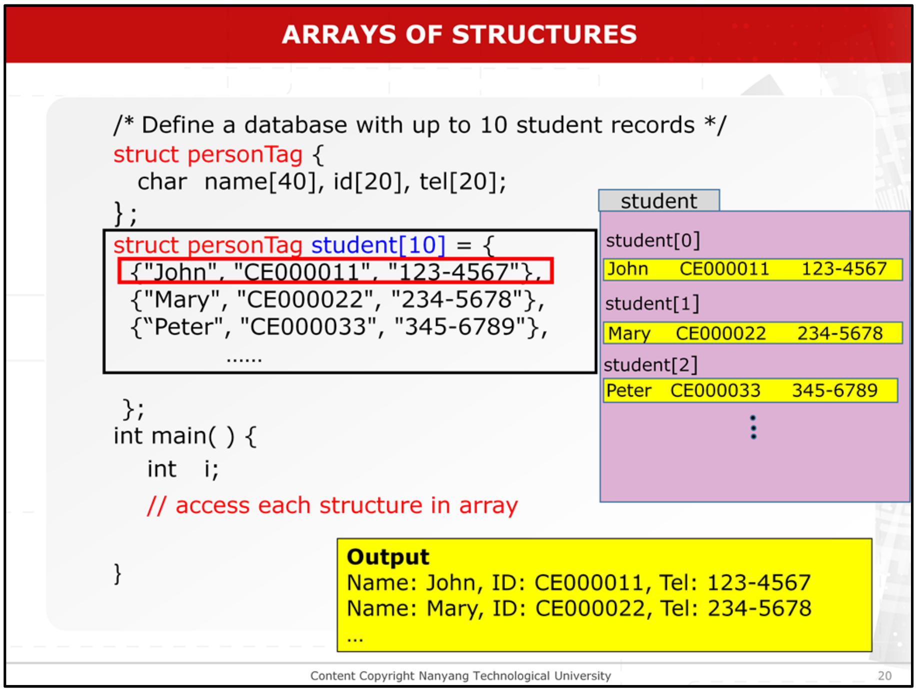
student[i].name

which denotes a member of the (i+1)th record.
We use

student[i].name[j]

to denote a single character value of a member of the (i+1)th record.

};



ARRAYS OF STRUCTURES

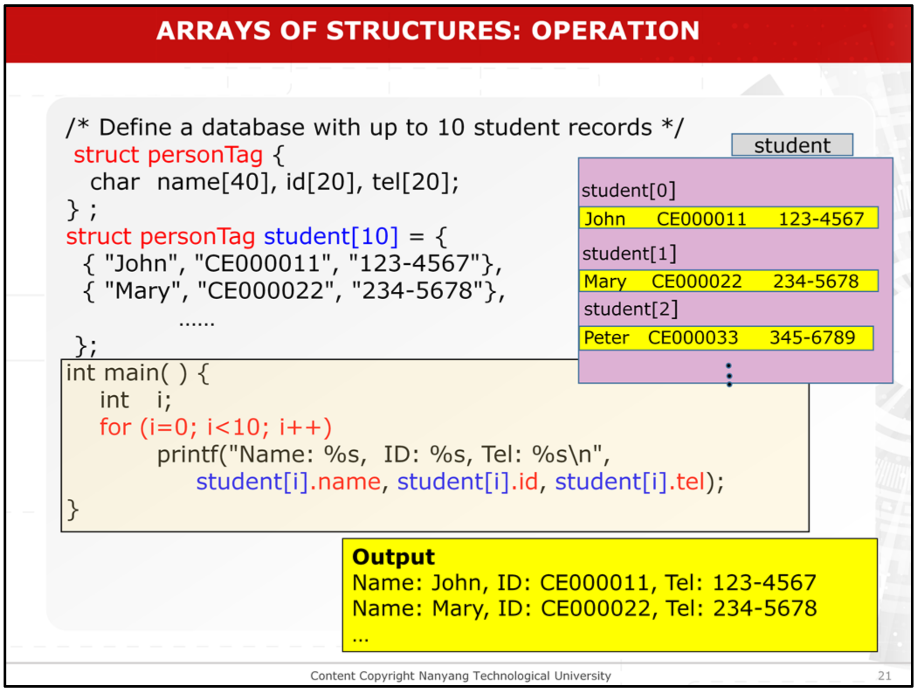
```
/* Define a database with up to 10 student records */
struct personTag {
   char  name[40], id[20], tel[20];
};
struct personTag student[10] = {
  {"John", "CE000011", "123-4567"},
  {"Mary", "CE000022", "234-5678"},
  {“Peter", "CE000033", "345-6789"},
           ......

 };
int main( ) {
    int   i;
    // access each structure in array

}
```

Output

```
Name: John, ID: CE000011, Tel: 123-4567
Name: Mary, ID: CE000022, Tel: 234-5678
…
```

Content Copyright Nanyang Technological University 20

Array of structures can be initialized. The initializers for each element are enclosed in braces, and each member is separated by a comma. An example is shown here on how the values are intialised for student 0, 1 , 2 and so on..



ARRAYS OF STRUCTURES: OPERATION

```
/* Define a database with up to 10 student records */
 struct personTag {
   char  name[40], id[20], tel[20];
} ;
struct personTag student[10] = {
  { "John", "CE000011", "123-4567"},
  { "Mary", "CE000022", "234-5678"},
            ......
 };
int main( ) {
   int   i;
   for (i=0; i<10; i++)
        printf("Name: %s,  ID: %s, Tel: %s\n",
           student[i].name, student[i].id, student[i].tel);
}
```

student

student[0]

John   CE000011   123-4567

student[1]

Mary   CE000022   234-5678

student[2]

Peter   CE000033   345-6789

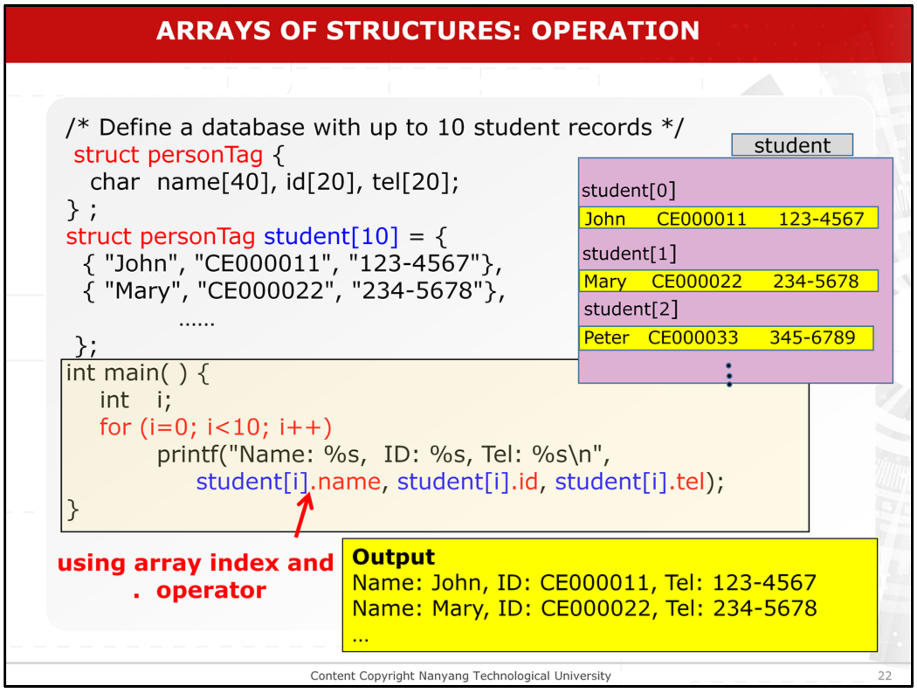
⋮

```
Output
Name: John, ID: CE000011, Tel: 123-4567
Name: Mary, ID: CE000022, Tel: 234-5678
…
```

Content Copyright Nanyang Technological University

Arrays of Structures: Operation

Note that the array index is used to traverse the array, and the member (or dot) operator is used to access each member of the structure in the array element.



ARRAYS OF STRUCTURES: OPERATION

```
/* Define a database with up to 10 student records */
 struct personTag {
   char  name[40], id[20], tel[20];
} ;
struct personTag student[10] = {
  { "John", "CE000011", "123-4567"},
  { "Mary", "CE000022", "234-5678"},
          ……
 };
int main( ) {
    int   i;
    for (i=0; i<10; i++)
          printf("Name: %s,  ID: %s, Tel: %s\n",
              student[i].name, student[i].id, student[i].tel);
}
```

student

student[0]

John CE000011 123-4567

student[1]

Mary CE000022 234-5678

student[2]

Peter CE000033 345-6789

⋮

using array index and . operator

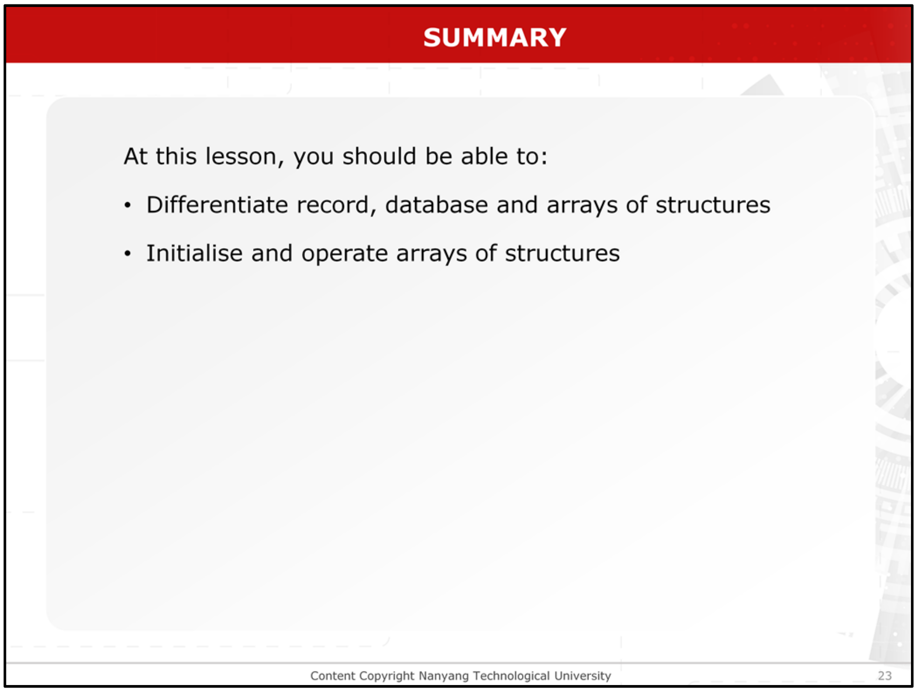
Output

```
Name: John, ID: CE000011, Tel: 123-4567
Name: Mary, ID: CE000022, Tel: 234-5678
…
```

Content Copyright Nanyang Technological University

Arrays of Structures: Operation

Note that the array index is used to traverse the array, and the member (or dot) operator is used to access each member of the structure in the array element.



SUMMARY

At this lesson, you should be able to:

- Differentiate record, database and arrays of structures
- Initialise and operate arrays of structures

Content Copyright Nanyang Technological University

In summary, after viewing this video lesson, your should be able to do the listed.