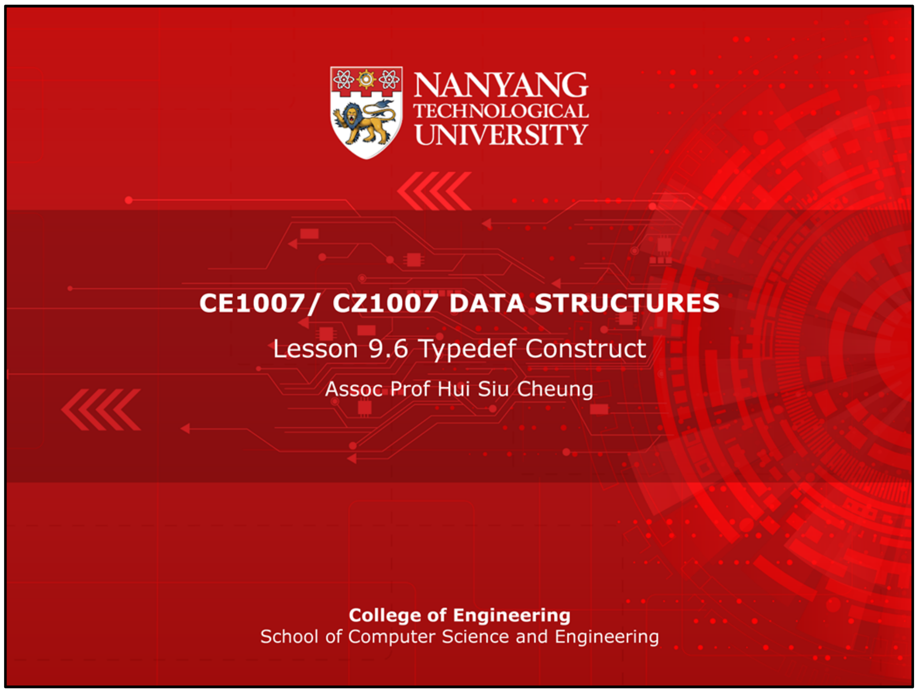


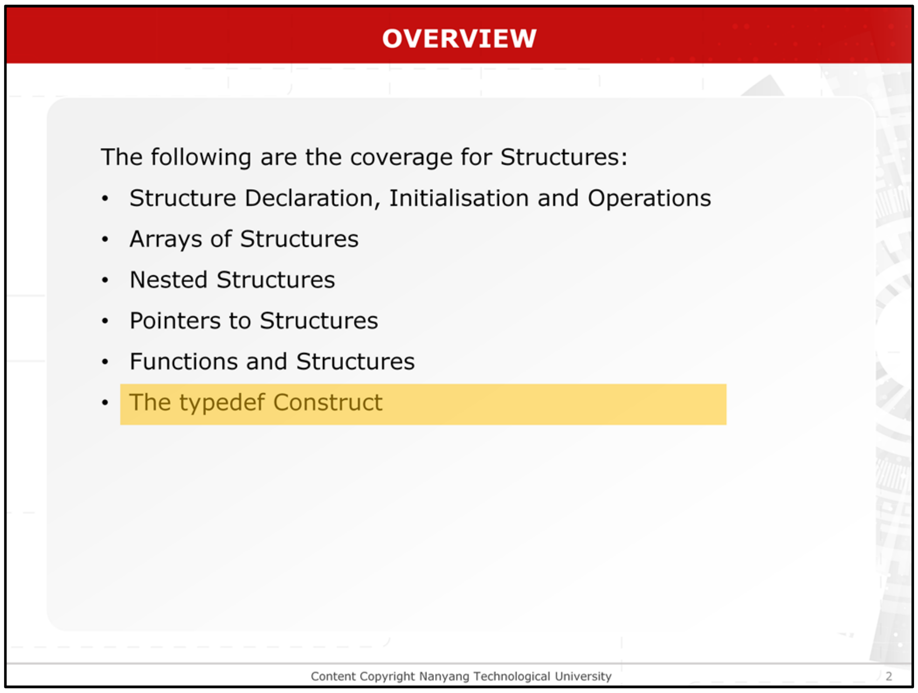
NANYANG TECHNOLOGICAL UNIVERSITY

CE1007/ CZ1007 DATA STRUCTURES

Lesson 9.6 Typedef Construct

Assoc Prof Hui Siu Cheung

College of Engineering
School of Computer Science and Engineering

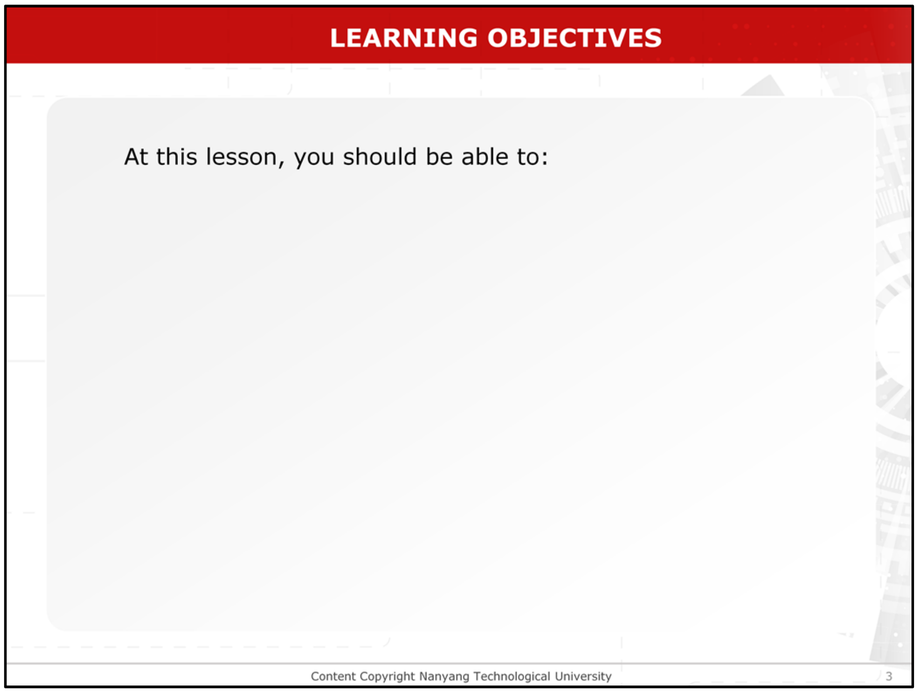


OVERVIEW

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

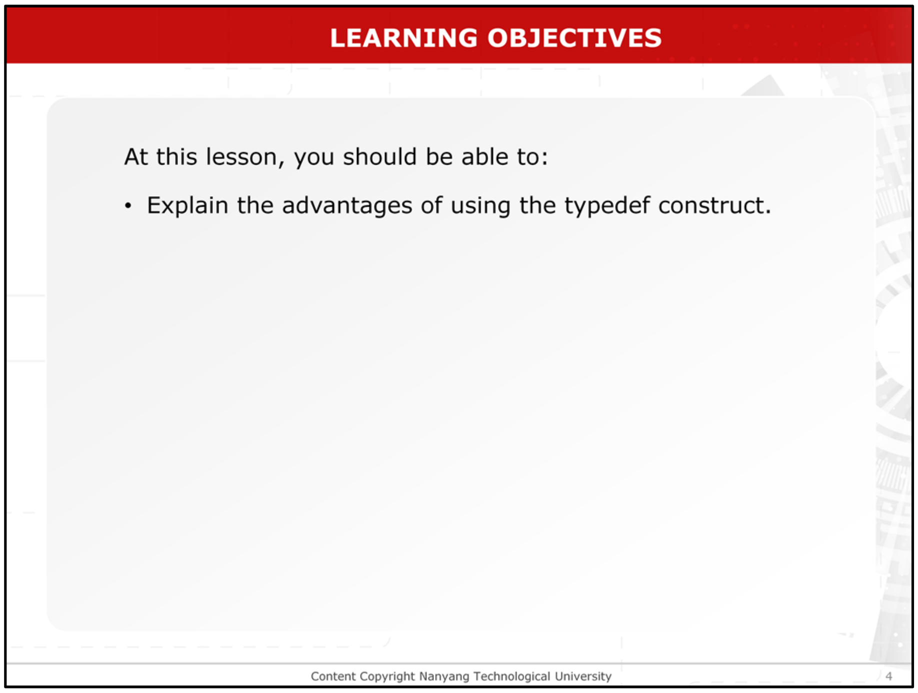
The following are the coverage for Structures: this video focusses on Pointers to Structures.



LEARNING OBJECTIVES

At this lesson, you should be able to:

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



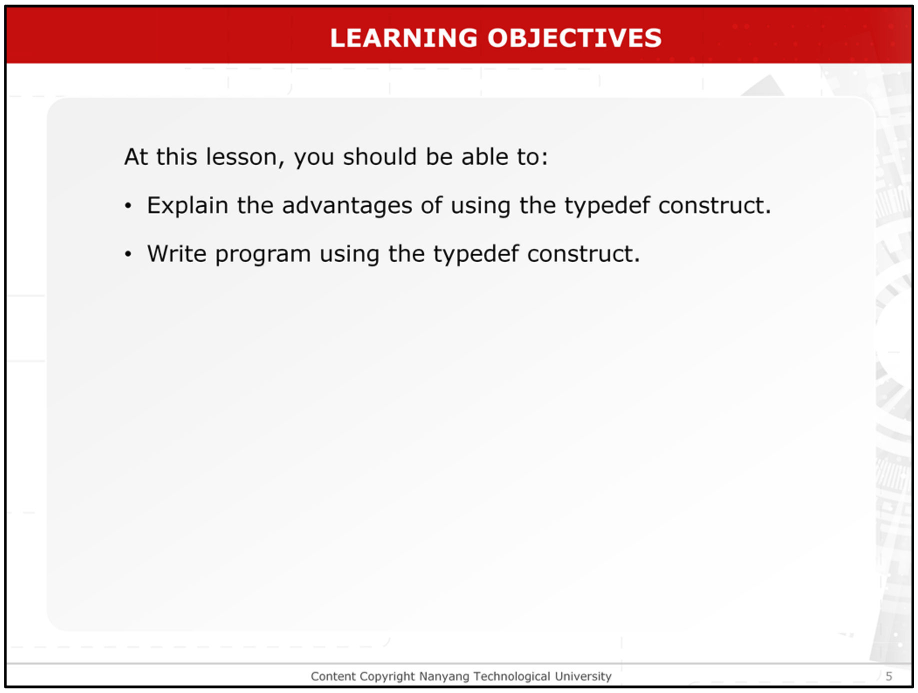
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Explain the advantages of using the typedef construct.

4

- Explain the advantages of using typedef construct.



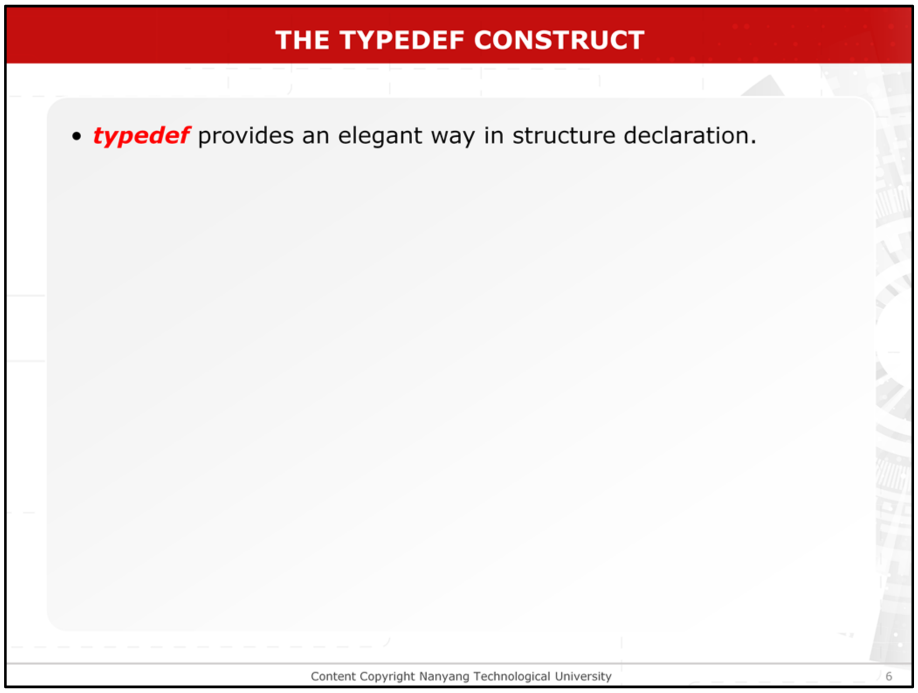
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Explain the advantages of using the typedef construct.
- Write program using the typedef construct.

5

Write program using typedef construct.

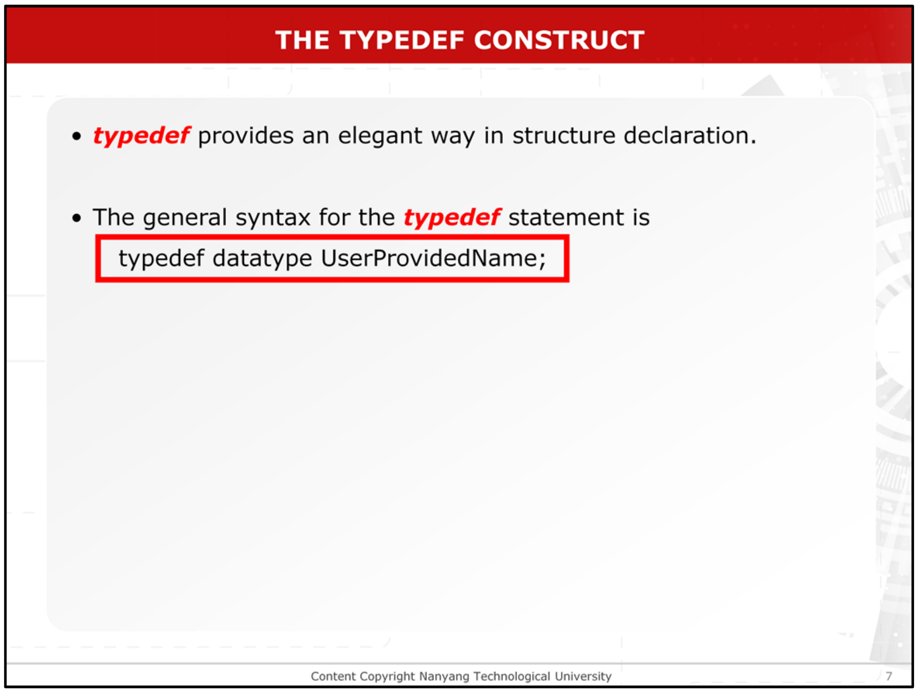


THE TYPEDEF CONSTRUCT

- ***typedef*** provides an elegant way in structure declaration.

6

typedef provides an elegant way in structure declaration



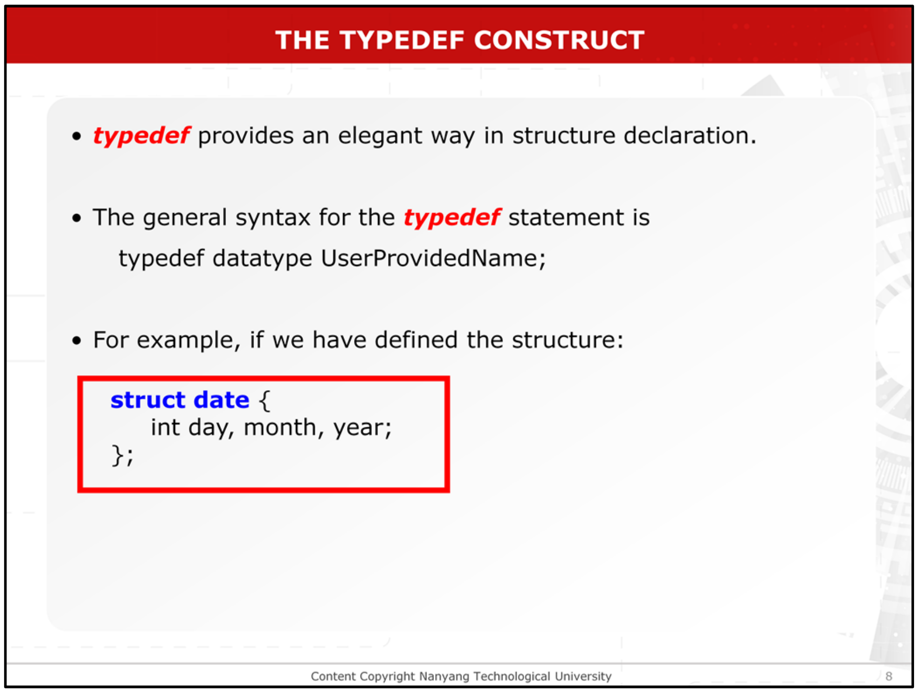
THE TYPEDEF CONSTRUCT

- ***typedef*** provides an elegant way in structure declaration.
- The general syntax for the ***typedef*** statement is

```
typedef datatype UserProvidedName;
```

7

The general syntax for the **typedef** statement is shown here: The **typedef** keyword is followed by the data type and the user provided name for the data type. It is very useful for creating simple names for complex structures



THE TYPEDEF CONSTRUCT

- *typedef* provides an elegant way in structure declaration.
- The general syntax for the *typedef* statement is

  typedef datatype UserProvidedName;
- For example, if we have defined the structure:

```
struct date {
    int day, month, year;
};
```

For example, if we have defined the structure like shown here,
we can define a new data type **Date** as

typedef struct date Date;

Variables can then be declared either as

struct date today, yesterday;

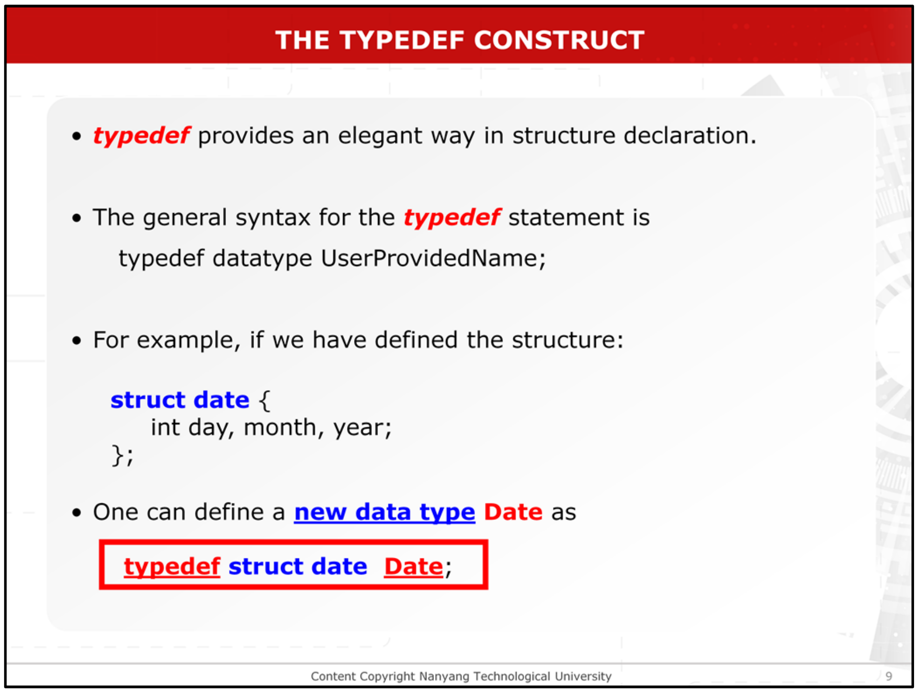
or **Date today, yesterday;**

We can also use the type **Date** in function prototypes and function definitions. When **typedef** is used, tag name is redundant. Therefore, we can declare

```
typedef struct {
    int   day, month, year;
} Date;
Date today, yesterday;
```

There are a number of advantages of using **typedef**. It enhances program documentation by using meaningful names for data types in the programs. It makes the program easier to read and understand. Another advantage is to define simpler data types for complex declarations such as structures.

In addition, **typedef** is similar to the **#define** preprocessor directive. However, there are a number of differences. **typedef** is limited to giving names to data types only and is processed by the compiler, while **#define** is not limited to data types and is processed by the preprocessor.



THE TYPEDEF CONSTRUCT

- ***typedef*** provides an elegant way in structure declaration.
- The general syntax for the ***typedef*** statement is

```
typedef datatype UserProvidedName;
```

- For example, if we have defined the structure:

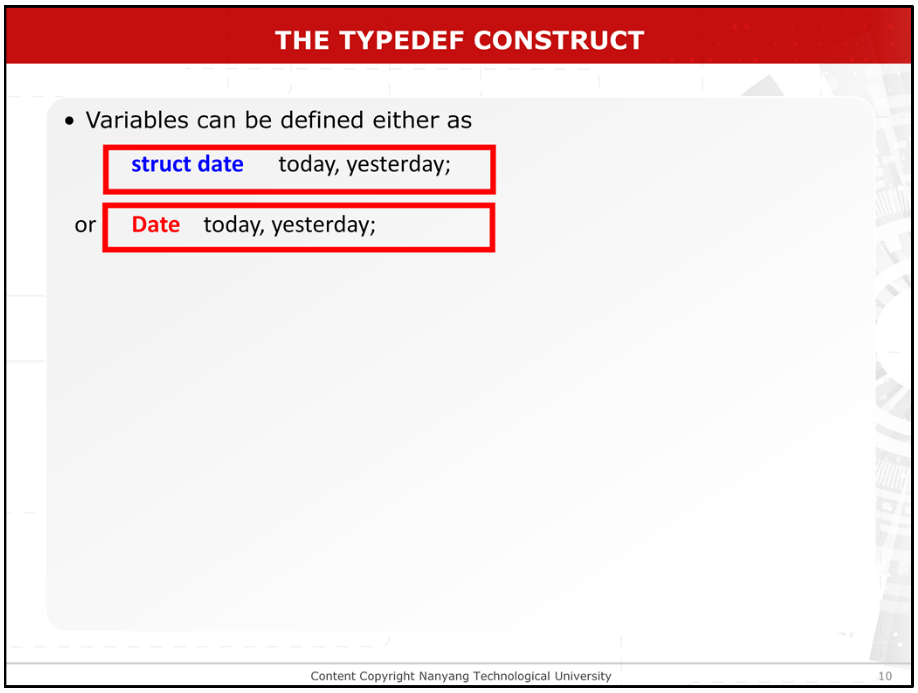
```
struct date {
    int day, month, year;
};
```

- One can define a **new data type** **Date** as

```
typedef struct date  Date;
```

9

For example, if we have defined the structure as shown here, we can define a new data type **Date** as shown.



THE TYPEDEF CONSTRUCT

- Variables can be defined either as

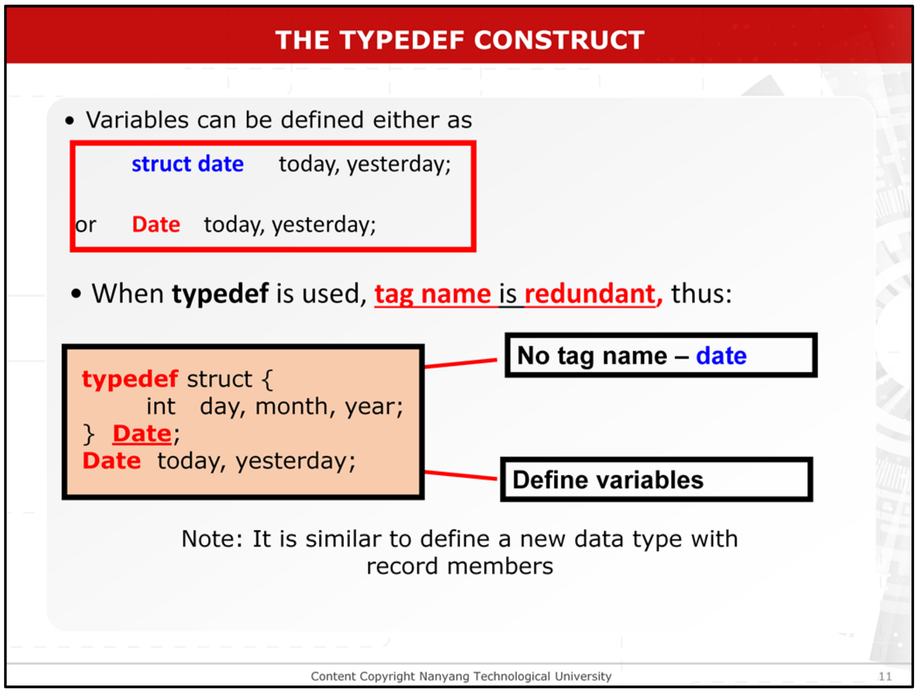
```
struct date     today, yesterday;
```

or

```
Date    today, yesterday;
```

10

Variables can then be declared as shown in 2 ways:



THE TYPEDEF CONSTRUCT

- Variables can be defined either as

```
        struct date      today, yesterday;

or      Date    today, yesterday;
```

- When **typedef** is used, **tag name** is **redundant,** thus:

```
typedef struct {
        int   day, month, year;
}  Date;
Date  today, yesterday;
```

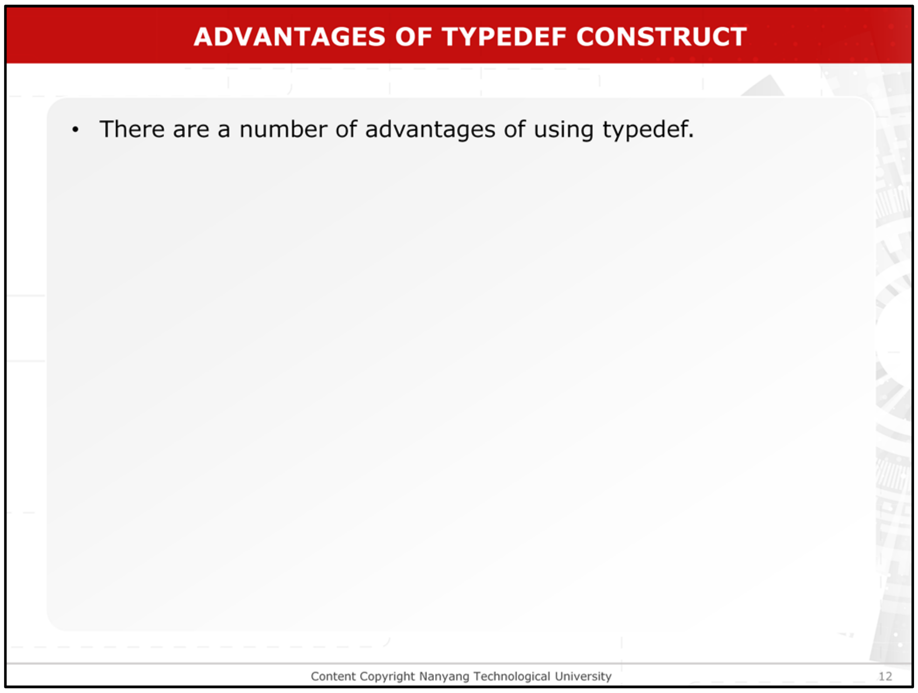
No tag name – date
Define variables

Note: It is similar to define a new data type with record members

11

We can also use the type **Date** in function prototypes and function definitions. When **typedef** is used, tag name is redundant. Therefore, we can declare as shown.

There are a number of advantages of using **typedef**. It enhances program documentation by using meaningful names for data types in the programs. It makes the program easier to read and understand. Another advantage is to define simpler data types for complex declarations such as structures.
In addition, **typedef** is similar to the **#define** preprocessor directive. However, there are a number of differences. **typedef** is limited to giving names to data types only and is processed by the compiler, while **#define** is not limited to data types and is processed by the preprocessor.

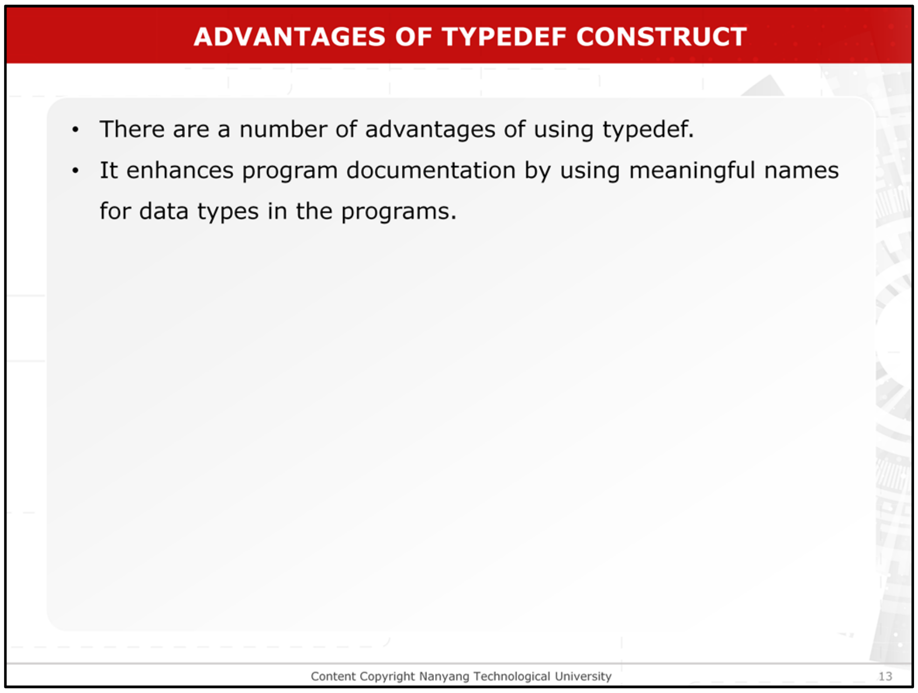


ADVANTAGES OF TYPEDEF CONSTRUCT

- There are a number of advantages of using typedef.

12

There are a number of advantages of using **typedef**

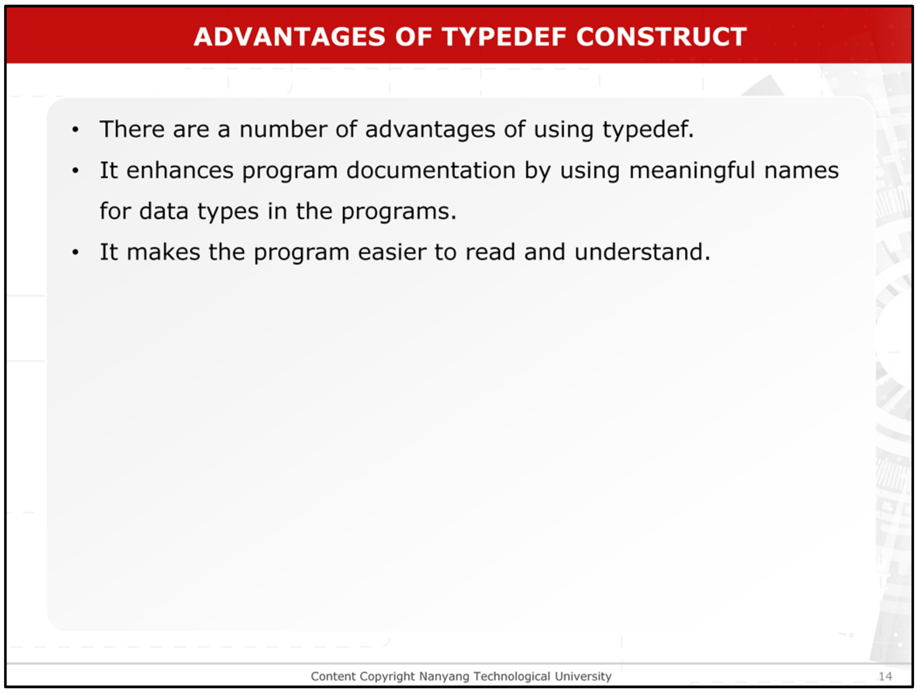


ADVANTAGES OF TYPEDEF CONSTRUCT

- There are a number of advantages of using typedef.
- It enhances program documentation by using meaningful names for data types in the programs.

13

It enhances program documentation by using meaningful names for data types in the programs

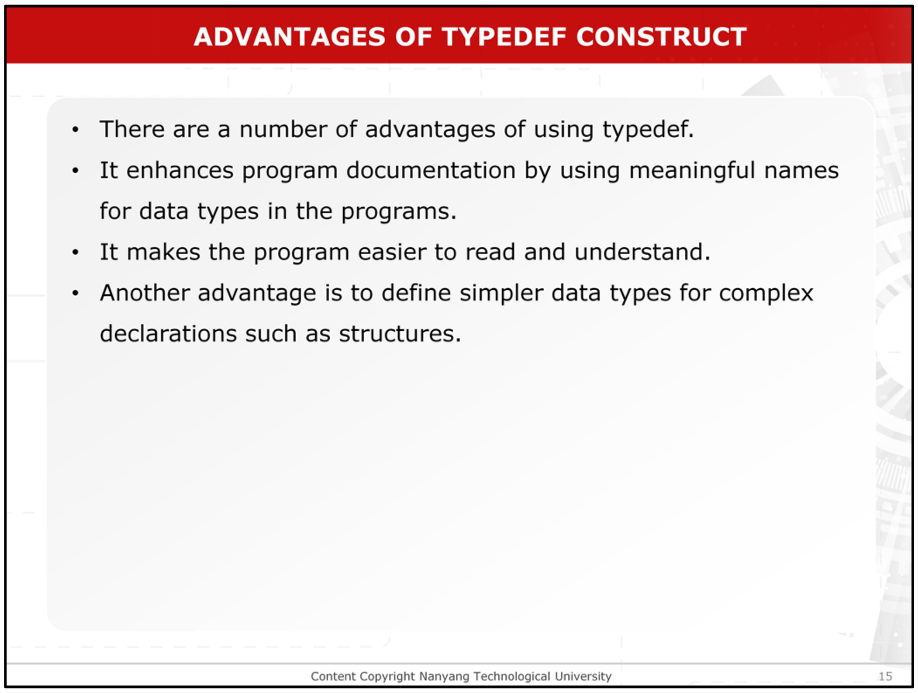


ADVANTAGES OF TYPEDEF CONSTRUCT

- There are a number of advantages of using typedef.
- It enhances program documentation by using meaningful names for data types in the programs.
- It makes the program easier to read and understand.

14

It makes the program easier to read and understand.

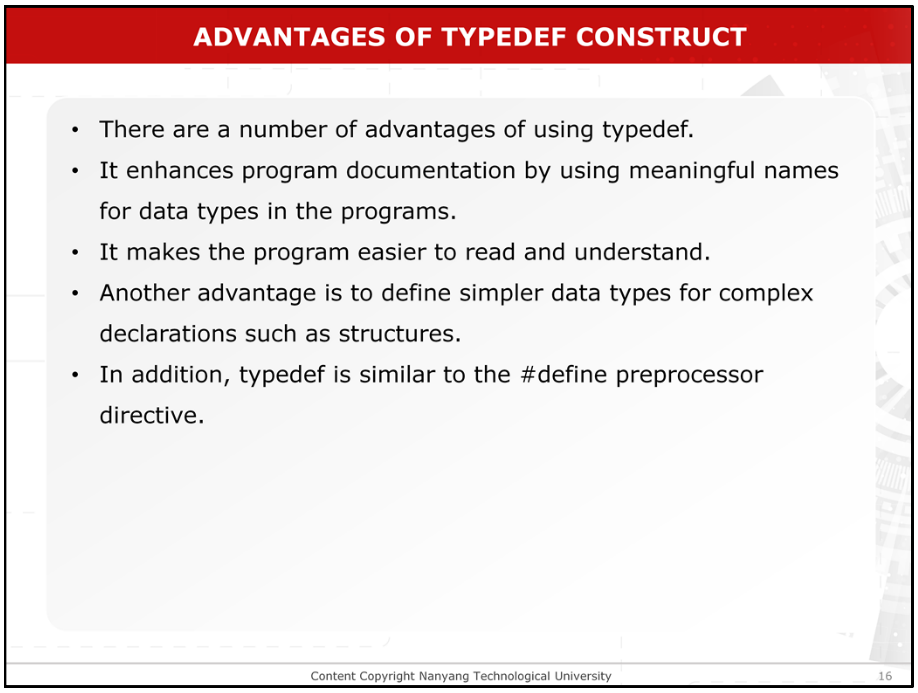


ADVANTAGES OF TYPEDEF CONSTRUCT

- There are a number of advantages of using typedef.
- It enhances program documentation by using meaningful names for data types in the programs.
- It makes the program easier to read and understand.
- Another advantage is to define simpler data types for complex declarations such as structures.

15

Another advantage is to define simpler data types for complex declarations such as structures.

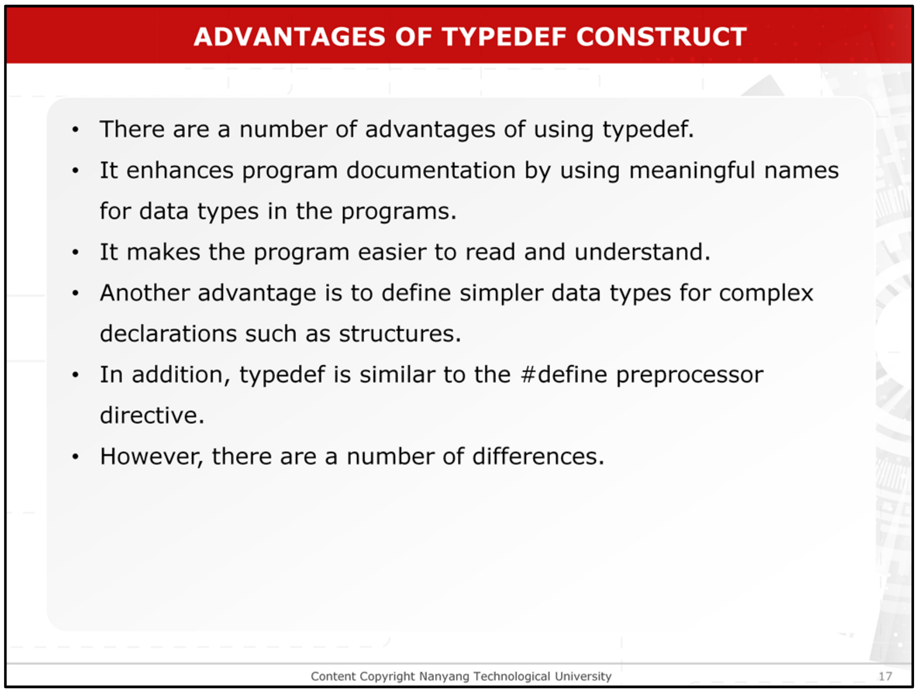


ADVANTAGES OF TYPEDEF CONSTRUCT

- There are a number of advantages of using typedef.
- It enhances program documentation by using meaningful names for data types in the programs.
- It makes the program easier to read and understand.
- Another advantage is to define simpler data types for complex declarations such as structures.
- In addition, typedef is similar to the #define preprocessor directive.

16

In addition, **typedef** is similar to the **#define** preprocessor directive.

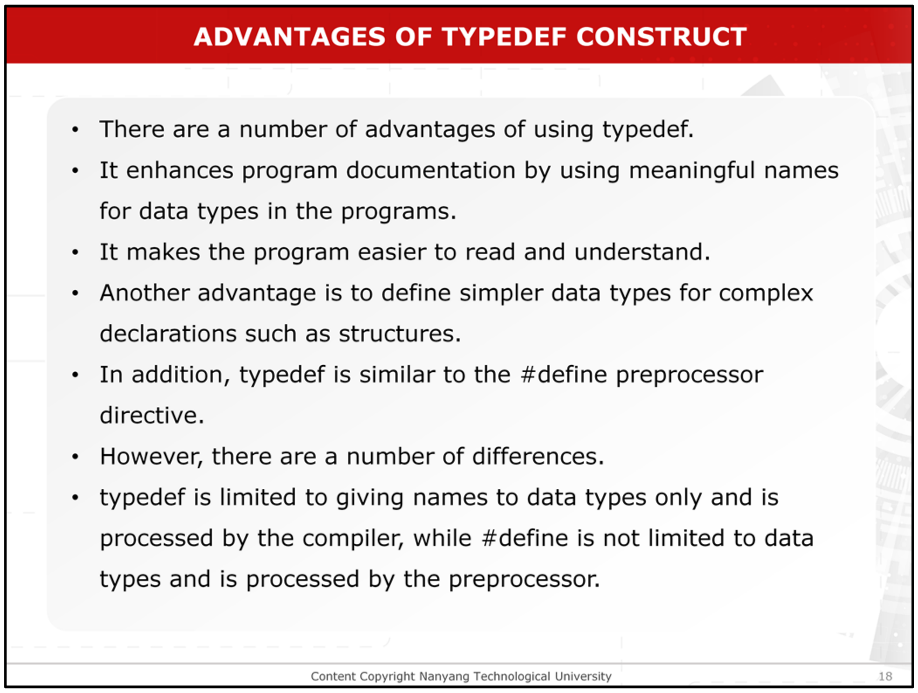


ADVANTAGES OF TYPEDEF CONSTRUCT

- There are a number of advantages of using typedef.
- It enhances program documentation by using meaningful names for data types in the programs.
- It makes the program easier to read and understand.
- Another advantage is to define simpler data types for complex declarations such as structures.
- In addition, typedef is similar to the #define preprocessor directive.
- However, there are a number of differences.

17

However, there are a number of differences. **typedef** is limited to giving names to data types only and is processed by the compiler, while **#define** is not limited to data types and is processed by the preprocessor.

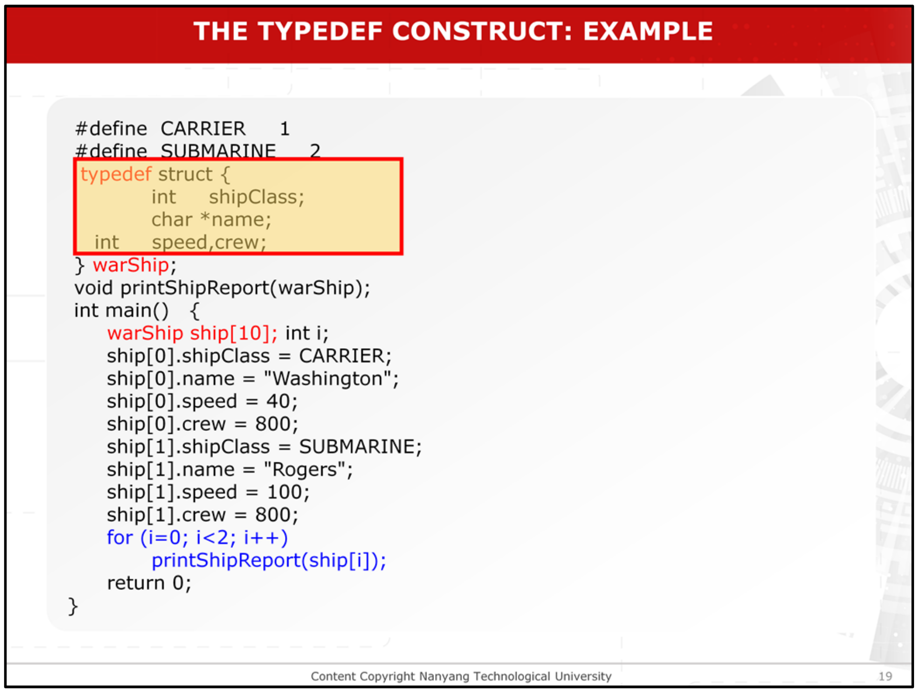


ADVANTAGES OF TYPEDEF CONSTRUCT

- There are a number of advantages of using typedef.
- It enhances program documentation by using meaningful names for data types in the programs.
- It makes the program easier to read and understand.
- Another advantage is to define simpler data types for complex declarations such as structures.
- In addition, typedef is similar to the #define preprocessor directive.
- However, there are a number of differences.
- typedef is limited to giving names to data types only and is processed by the compiler, while #define is not limited to data types and is processed by the preprocessor.

18

typedef is limited to giving names to data types only and is processed by the compiler, while **#define** is not limited to data types and is processed by the preprocessor.



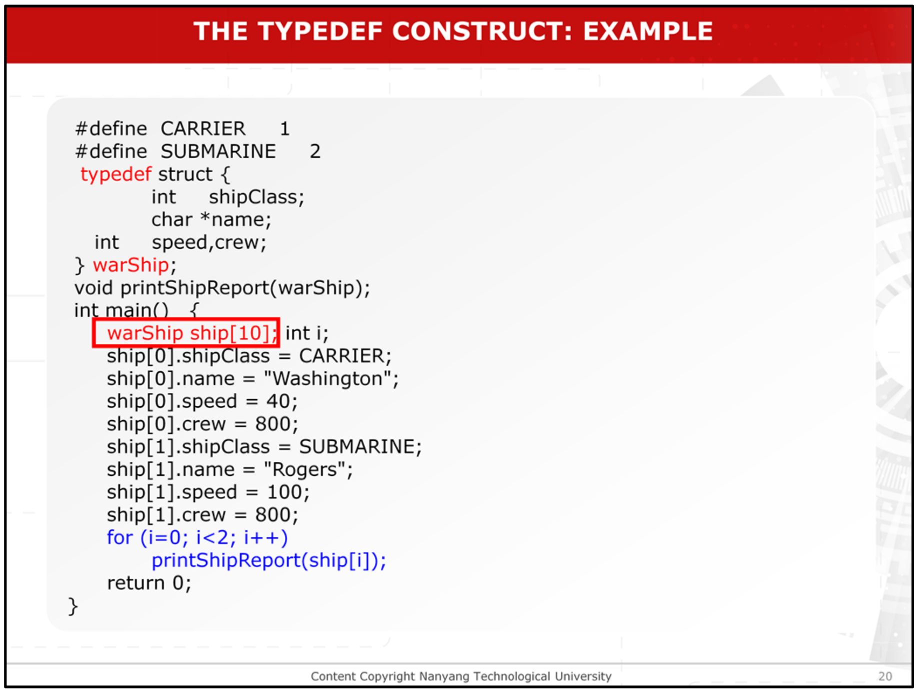
THE TYPEDEF CONSTRUCT: EXAMPLE

```
#define  CARRIER     1
#define  SUBMARINE     2
typedef struct {
          int     shipClass;
          char *name;
  int     speed,crew;
} warShip;
void printShipReport(warShip);
int main()   {
     warShip ship[10]; int i;
     ship[0].shipClass = CARRIER;
     ship[0].name = "Washington";
     ship[0].speed = 40;
     ship[0].crew = 800;
     ship[1].shipClass = SUBMARINE;
     ship[1].name = "Rogers";
     ship[1].speed = 100;
     ship[1].crew = 800;
     for (i=0; i<2; i++)
          printShipReport(ship[i]);
     return 0;
}
```

19

The typedef Construct: Example

In this program, we use **typedef** to define a new structure type **warShip**:

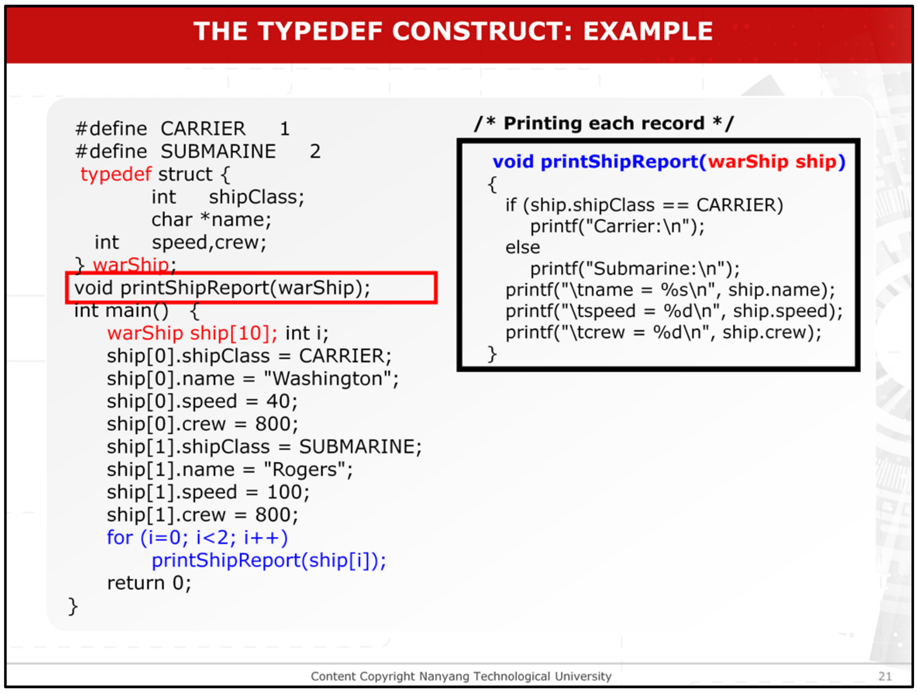


THE TYPEDEF CONSTRUCT: EXAMPLE

```
#define  CARRIER     1
#define  SUBMARINE     2
 typedef struct {
          int     shipClass;
          char *name;
   int     speed,crew;
} warShip;
void printShipReport(warShip);
int main()   {
     warShip ship[10]; int i;
     ship[0].shipClass = CARRIER;
     ship[0].name = "Washington";
     ship[0].speed = 40;
     ship[0].crew = 800;
     ship[1].shipClass = SUBMARINE;
     ship[1].name = "Rogers";
     ship[1].speed = 100;
     ship[1].crew = 800;
     for (i=0; i<2; i++)
          printShipReport(ship[i]);
     return 0;
}
```

20

In the **main()** function, we declare an array of **warShip** structures variable called **ship**.



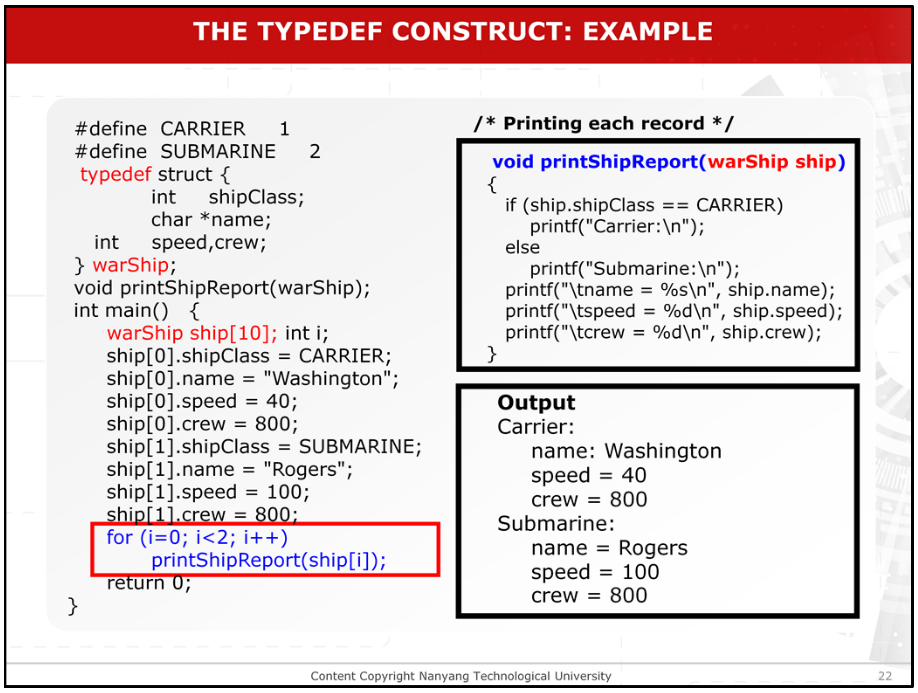
THE TYPEDEF CONSTRUCT: EXAMPLE

```
#define  CARRIER     1
#define  SUBMARINE     2
 typedef struct {
            int     shipClass;
            char *name;
   int     speed,crew;
} warShip;
void printShipReport(warShip);
int main()   {
     warShip ship[10]; int i;
     ship[0].shipClass = CARRIER;
     ship[0].name = "Washington";
     ship[0].speed = 40;
     ship[0].crew = 800;
     ship[1].shipClass = SUBMARINE;
     ship[1].name = "Rogers";
     ship[1].speed = 100;
     ship[1].crew = 800;
     for (i=0; i<2; i++)
            printShipReport(ship[i]);
     return 0;
}
```

```
/* Printing each record */
 void printShipReport(warShip ship)
 {
   if (ship.shipClass == CARRIER)
       printf("Carrier:\n");
   else
       printf("Submarine:\n");
   printf("\tname = %s\n", ship.name);
   printf("\tspeed = %d\n", ship.speed);
   printf("\tcrew = %d\n", ship.crew);
 }
```

21

lThe function **printShipReport()** is used for printing the member information of the **warShip** structure.



THE TYPEDEF CONSTRUCT: EXAMPLE

```
#define  CARRIER     1
#define  SUBMARINE    2
 typedef struct {
            int     shipClass;
            char *name;
   int     speed,crew;
} warShip;
void printShipReport(warShip);
int main()   {
     warShip ship[10]; int i;
     ship[0].shipClass = CARRIER;
     ship[0].name = "Washington";
     ship[0].speed = 40;
     ship[0].crew = 800;
     ship[1].shipClass = SUBMARINE;
     ship[1].name = "Rogers";
     ship[1].speed = 100;
     ship[1].crew = 800;
     for (i=0; i<2; i++)
            printShipReport(ship[i]);
     return 0;
}
```

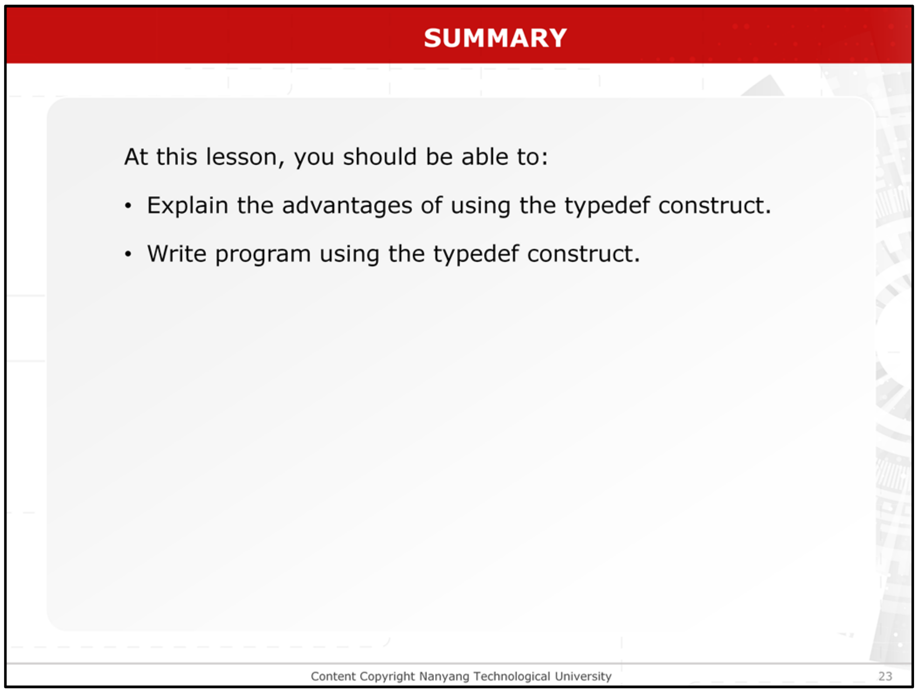
/* Printing each record */

```
 void printShipReport(warShip ship)
{
   if (ship.shipClass == CARRIER)
       printf("Carrier:\n");
   else
       printf("Submarine:\n");
   printf("\tname = %s\n", ship.name);
   printf("\tspeed = %d\n", ship.speed);
   printf("\tcrew = %d\n", ship.crew);
}
```

Output

```
Carrier:
     name: Washington
     speed = 40
     crew = 800
Submarine:
     name = Rogers
     speed = 100
     crew = 800
```

In the **main()** function, a **for** loop is used to print the member information of the **ship** variable using the **printShipReport()** function.



SUMMARY

At this lesson, you should be able to:

- Explain the advantages of using the typedef construct.
- Write program using the typedef construct.

23

After viewing this video lecture, you will be able to do the listed.