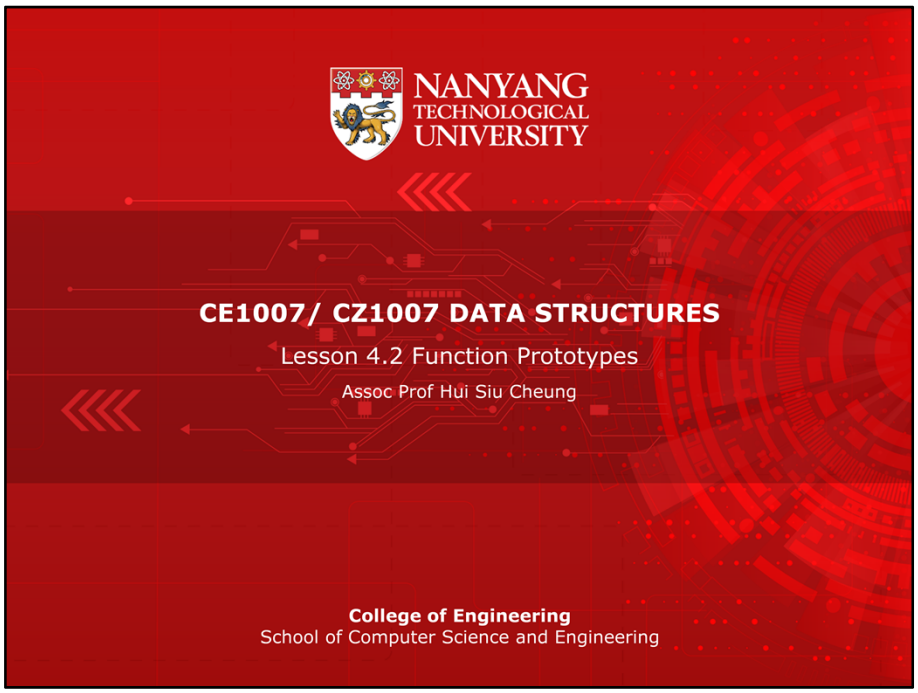
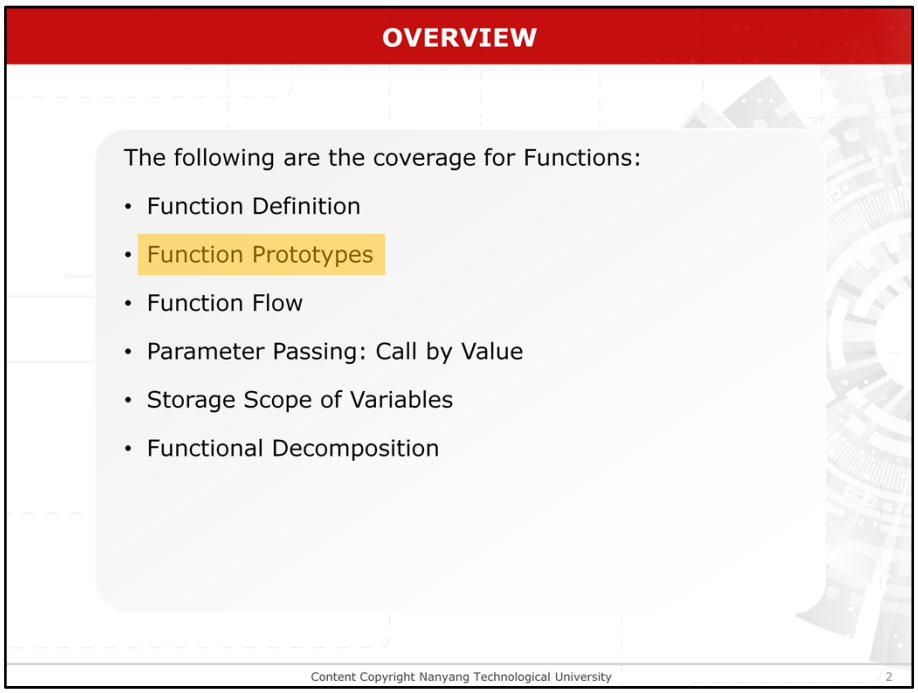
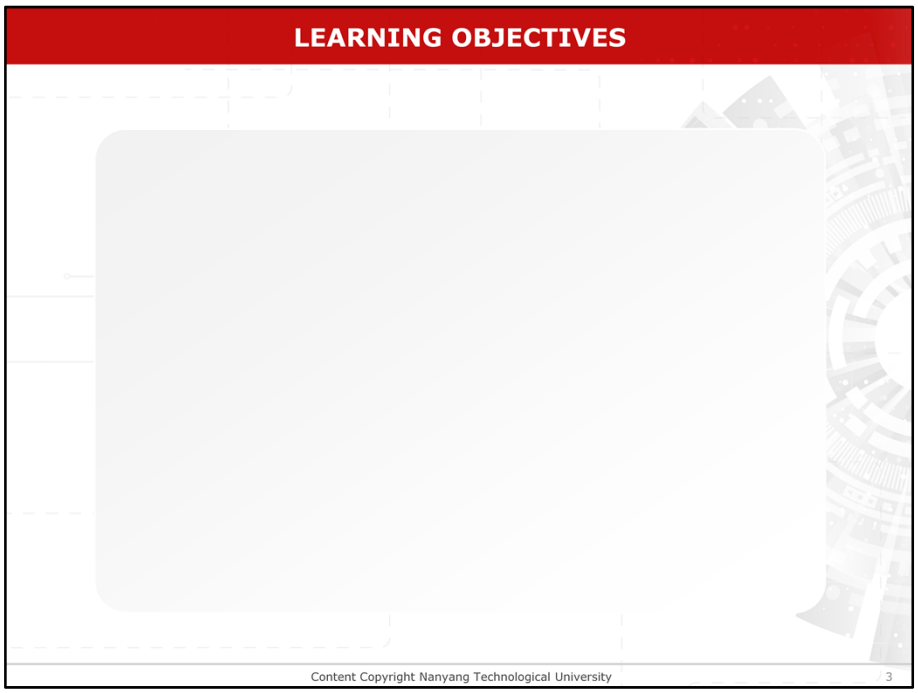



This lesson is on Functions

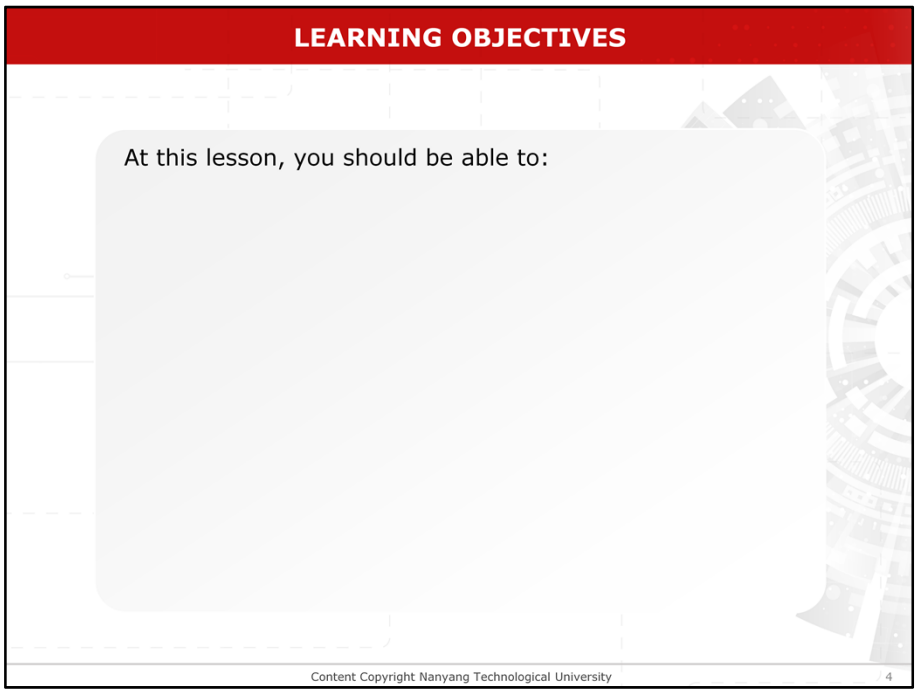



**Basic C Programming**

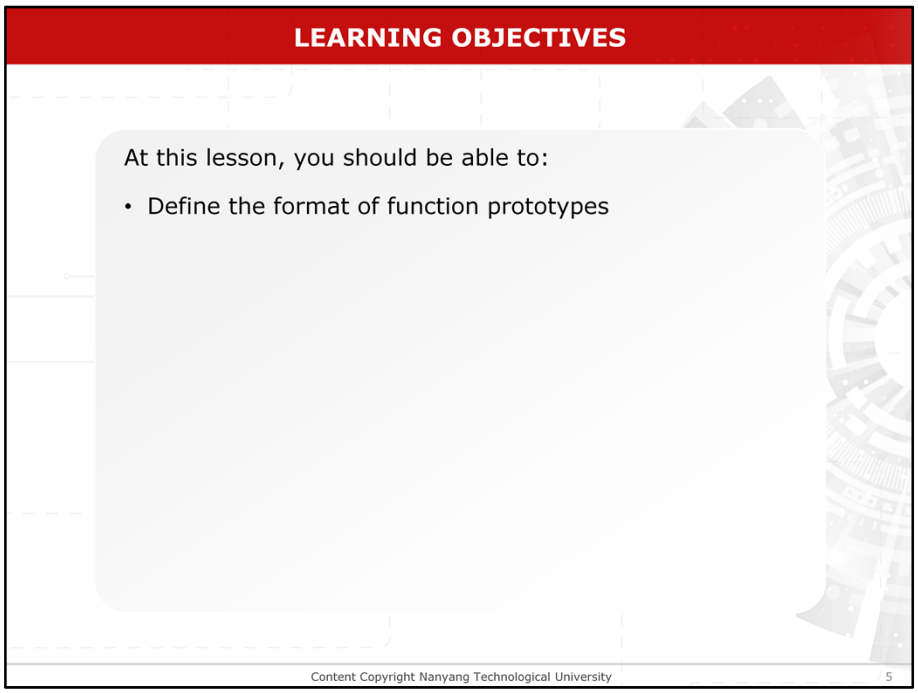
There are 6 main sections to cover for Functions. This video focusses on the 2nd topic: Function prototypes.



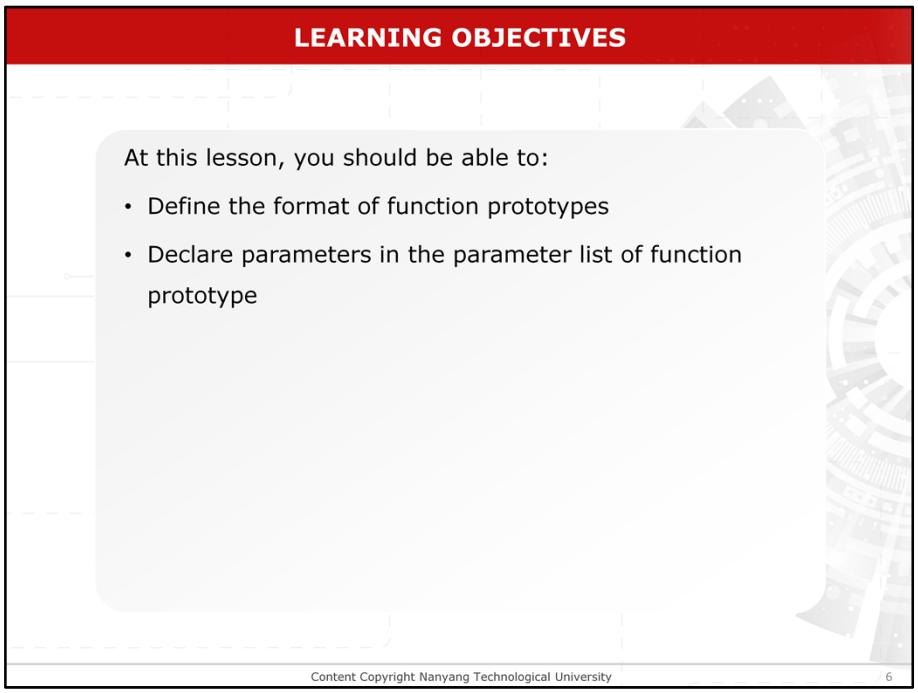

Learrning objectives



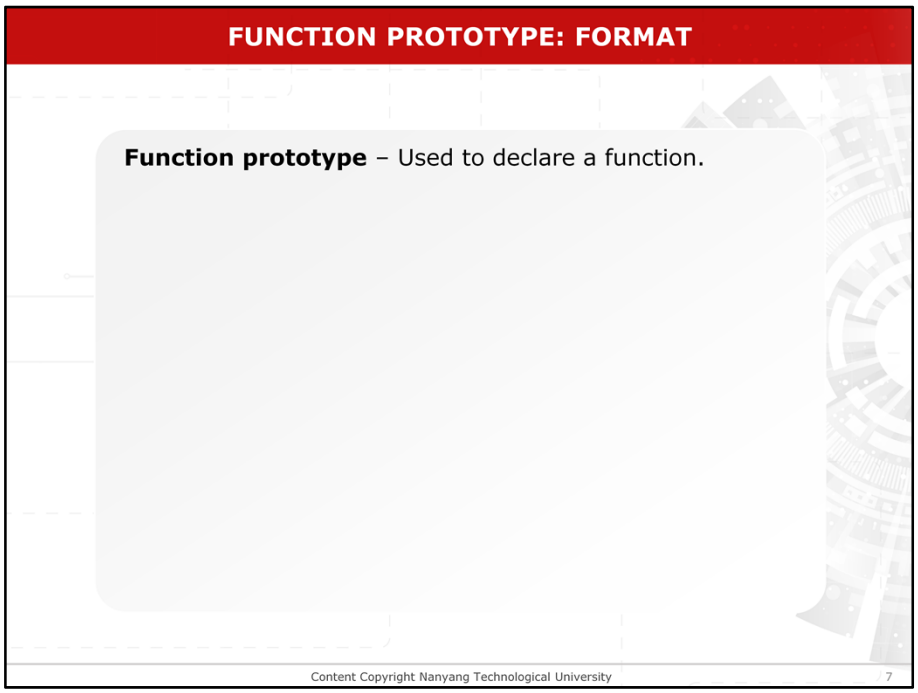

At this lesson, you should be able to:




- Define the format of function prototypes




- Declare parameters in the parameter list of function prototype




**Function Prototype**

We need to declare a function before using it in the **main()** function or other functions. A function declaration is called a *function prototype*.

## FUNCTION PROTOTYPE: FORMAT

**Function prototype** – Used to declare a function.

It provides the information about:

 8

It provides the information about

## FUNCTION PROTOTYPE: FORMAT

**Function prototype** – Used to declare a function.

It provides the information about:

- The **return type** of the function

 9

the type of the function

## FUNCTION PROTOTYPE: FORMAT

**Function prototype** – Used to declare a function.

It provides the information about:

- The **return type** of the function
- The **name** of the function

 10

the name of the function

## FUNCTION PROTOTYPE: FORMAT

**Function prototype** – Used to declare a function.

It provides the information about:

- The **return type** of the function
- The **name** of the function
- The **number** and **types** of the arguments

 11

and the number and types of the arguments.

## FUNCTION PROTOTYPE: FORMAT

**Function prototype** – Used to declare a function.

It provides the information about:

- The **return type** of the function
- The **name** of the function
- The **number** and **types** of the arguments

The declaration may be the same as the function header but terminated by a **semicolon**.

**Example:**

```
float findMax(float x, float y) ;
```

 12

The declaration may be the same as the function header but terminated by a **semicolon**. An example is shown here

## FUNCTION PROTOTYPE: FORMAT

Two ways to declare parameters in the parameter list of function prototype:

 13

There are two ways to declare parameters in the parameter list of function prototype.

## FUNCTION PROTOTYPE: FORMAT

Two ways to declare parameters in the parameter list of function prototype:

1. `void hello_n_times(int n); // with parameter name n`

 14

This declaration specifies that the function **hello_n_times()** expects one argument of type **int** and does not return any value.

## FUNCTION PROTOTYPE: FORMAT

Two ways to declare parameters in the parameter list of function prototype:

1. `void hello_n_times(int n); // with parameter name n`
2. `double distance(double, double); // no parameter name`

 15

The function prototype can also be declared without giving the argument names

## FUNCTION PROTOTYPE: FORMAT

- Function prototypes enable the compiler to ensure that functions are being called properly.

 16

Function prototypes enable the compiler to ensure that functions are being called properly.

## FUNCTION PROTOTYPE: FORMAT

- Function prototypes enable the compiler to ensure that functions are being called properly.
- The compiler will check whether the number of arguments and the type of the arguments of the function call match with the parameters used in the function definition.

The compiler will check whether the number of arguments and the type of the arguments of the function call match with the parameters used in the function definition.

## FUNCTION PROTOTYPE: FORMAT

- Function prototypes enable the compiler to ensure that functions are being called properly.
- The compiler will check whether the number of arguments and the type of the arguments of the function call match with the parameters used in the function definition.
- Warning messages will be given if the number of arguments is different.

 18

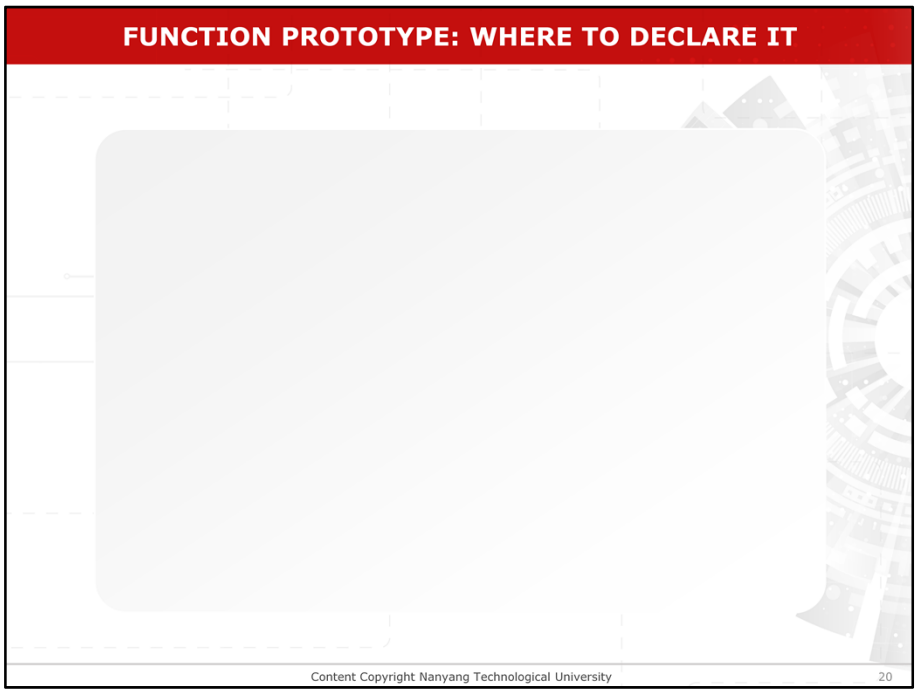
Warning messages will be given if the number of arguments is different. Type casting will also be done if the type of the argument is mismatch.

## FUNCTION PROTOTYPE: FORMAT

- Function prototypes enable the compiler to ensure that functions are being called properly.
- The compiler will check whether the number of arguments and the type of the arguments of the function call match with the parameters used in the function definition.
- Warning messages will be given if the number of arguments is different.
- Type casting will also be done if the type of the argument is mismatch.

 19

Type casting will also be done if the type of the argument is mismatch.




**Function Prototype: Where to declare it**

## FUNCTION PROTOTYPE: WHERE TO DECLARE IT

The declaration has to be done **before** the function is called:

 21

**Function Prototype: Where to declare it**

A function must be declared before it is actually called.

## FUNCTION PROTOTYPE: WHERE TO DECLARE IT

The declaration has to be done **before** the function is called:

(1) Before the main() header,

 22

It can be declared either before the **main()** header

## FUNCTION PROTOTYPE: WHERE TO DECLARE IT

The declaration has to be done **before** the function is called:

(1) Before the main() header,

(2) Inside the main() body, or

 23

inside the **main()** body

## FUNCTION PROTOTYPE: WHERE TO DECLARE IT

The declaration has to be done **before** the function is called:

(1) Before the main() header,

(2) Inside the main() body, or

(3) Inside any function which uses it.

or inside any function which uses it.

## FUNCTION PROTOTYPE: WHERE TO DECLARE IT

```
#include <stdio.h>
int factorial(int n);  // function prototype
int main( )
{
    int x;
    x = factorial(5); // use factorial() here
}
int factorial(int n)  /* function definition*/
{
    ....
}
```

 25

If the function prototype is placed before the **main()** function and at the beginning of the program, it makes the function available to all the functions in the program.

## FUNCTION PROTOTYPE: WHERE TO DECLARE IT

```
#include <stdio.h>
int factorial(int n);   // function prototype
int main( )
{
    int x;
    x = factorial(5); // use factorial() here
}
int factorial(int n)  /* function definition*/
{
    ....
}
```

 26

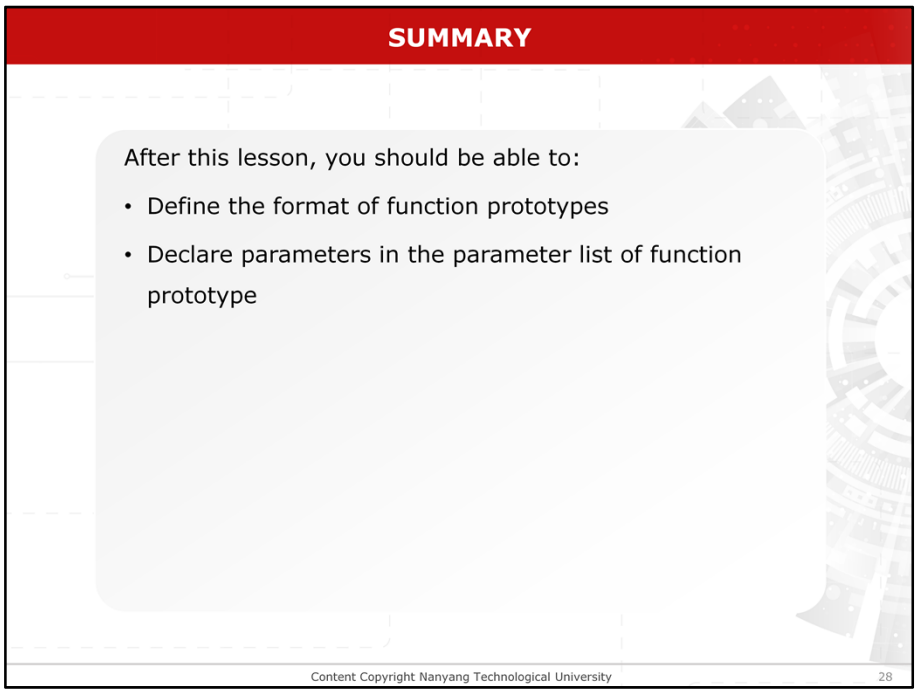
The function **factorial()** is declared outside the **main()** and can be used by all the functions in the program.

## FUNCTION PROTOTYPE: WHERE TO DECLARE IT

```
#include <stdio.h>
int main()
{
    int x;
    int fact(int); // function prototype
    x = fact(5);   // use fact() here
    ....
}
int fact(int n)   // function definition
{
    ....
}
```

The function **fact()** is declared inside the **main()** function. This makes the function callable only within the **main()** function.




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