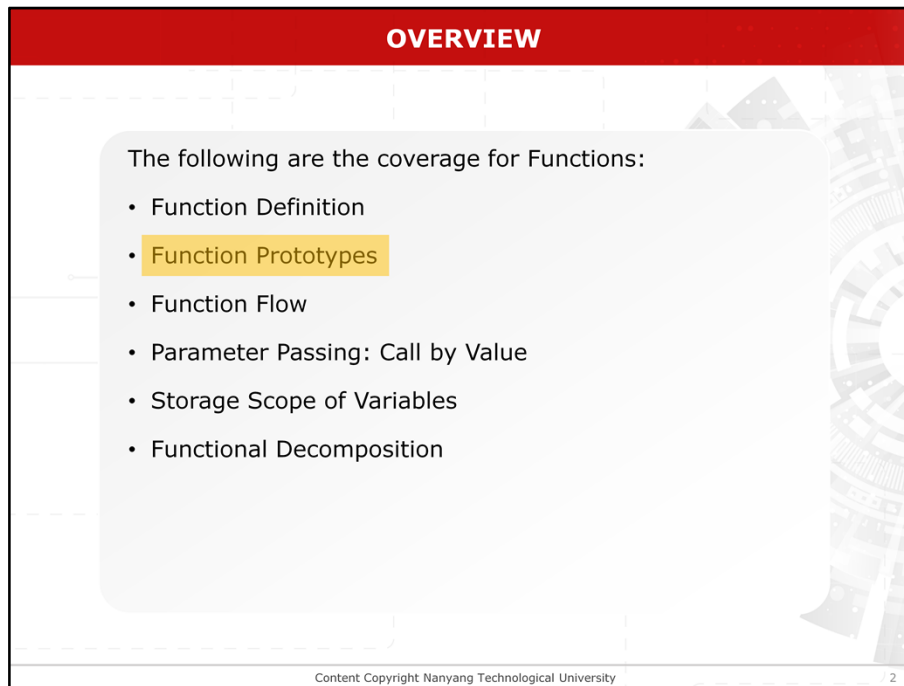


This lesson is on Functions

The slide features a red header with the word "OVERVIEW" in white. Below the header, a light gray rounded rectangle contains a list of topics. The second item, "Function Prototypes", is highlighted with a yellow background. The background of the slide has a faint, stylized architectural pattern on the right side.

OVERVIEW

The following are the coverage for Functions:

- Function Definition
- **Function Prototypes**
- Function Flow
- Parameter Passing: Call by Value
- Storage Scope of Variables
- Functional Decomposition

Content Copyright Nanyang Technological University 2

Basic C Programming

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



Learning objectives

LEARNING OBJECTIVES

At this lesson, you should be able to:

Content Copyright Nanyang Technological University 4

The slide features a red header with the title 'LEARNING OBJECTIVES'. Below the header is a large, light gray rectangular box with rounded corners, intended for listing learning objectives. The text 'At this lesson, you should be able to:' is positioned at the top left of this box. The background of the slide includes a faint, stylized graphic of a building and a gear. At the bottom, a thin black line separates the content from the footer, which contains the text 'Content Copyright Nanyang Technological University' and the page number '4'.

At this lesson, you should be able to:

LEARNING OBJECTIVES

At this lesson, you should be able to:

- Define the format of function prototypes

Content Copyright Nanyang Technological University 5

- Define the format of function prototypes

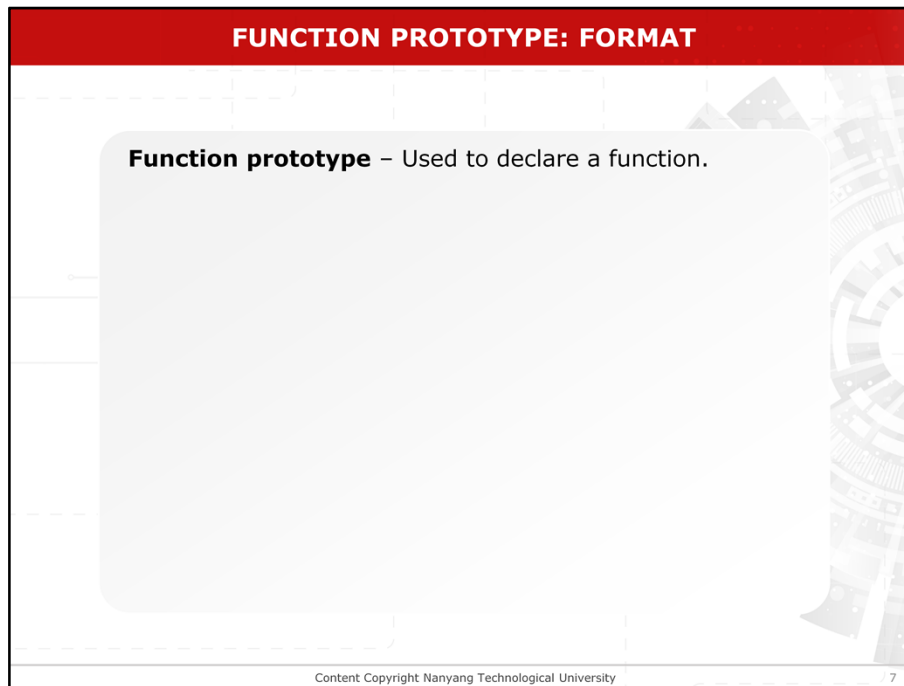
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Define the format of function prototypes
- Declare parameters in the parameter list of function prototype

Content Copyright Nanyang Technological University 6

- 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:

Content Copyright Nanyang Technological University

8

It provides the information about

FUNCTION PROTOTYPE: FORMAT

Function prototype – Used to declare a function.

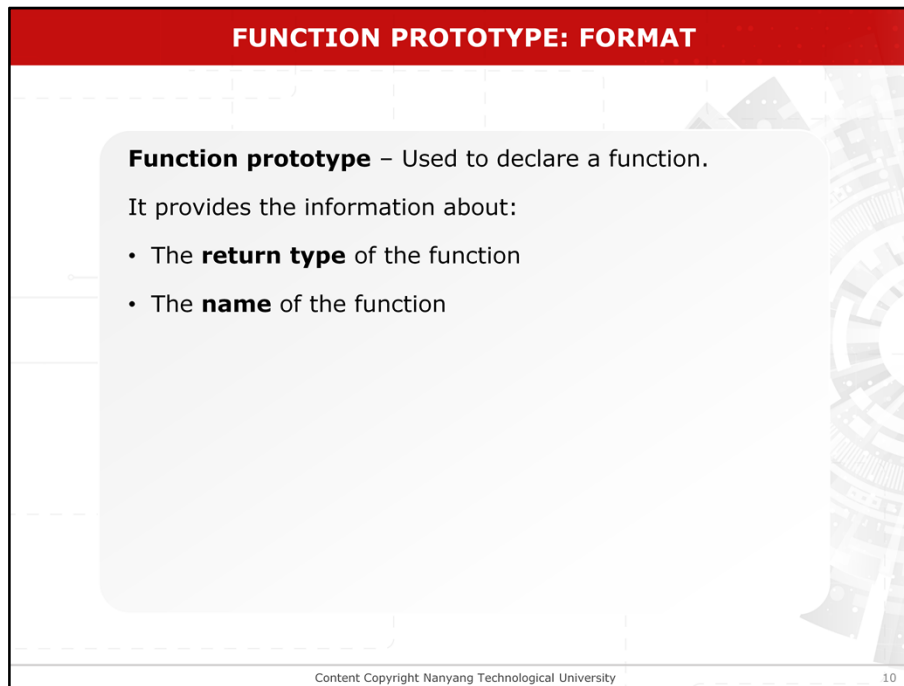
It provides the information about:

- The **return type** of the function

Content Copyright Nanyang Technological University

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

Content Copyright Nanyang Technological University 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

Content Copyright Nanyang Technological University 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) ;
```

Content Copyright Nanyang Technological University 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:

Content Copyright Nanyang Technological University

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

Content Copyright Nanyang Technological University 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

Content Copyright Nanyang Technological University 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.

Content Copyright Nanyang Technological University 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.

Content Copyright Nanyang Technological University 17

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.

Content Copyright Nanyang Technological University 18

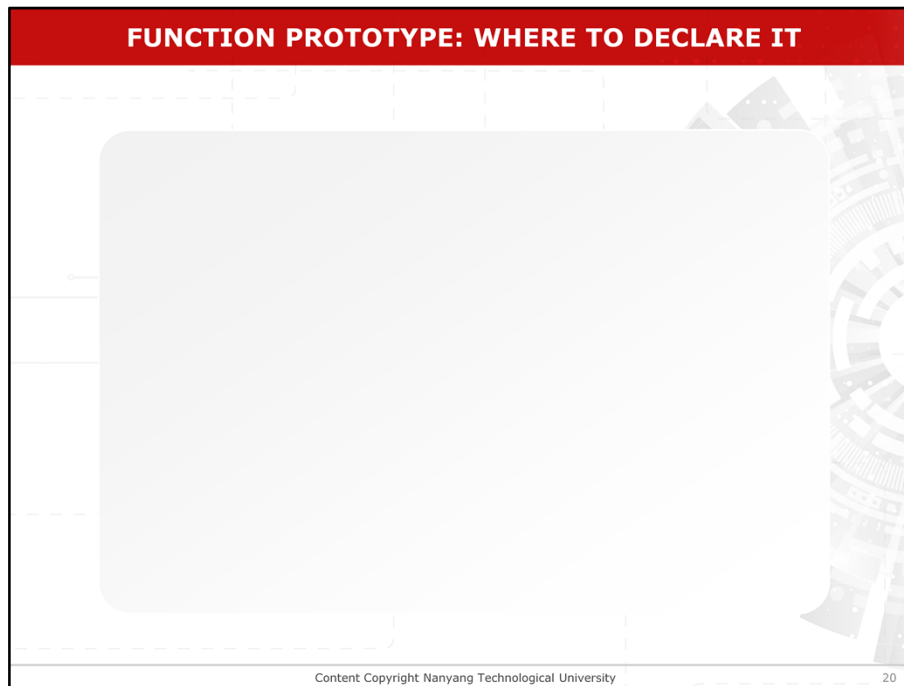
Warning messages will be given if the number of arguments is different. Type casting will also be done if the type of the argument is mismatch.

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.

Content Copyright Nanyang Technological University 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:

Content Copyright Nanyang Technological University 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,

Content Copyright Nanyang Technological University 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

Content Copyright Nanyang Technological University 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.

Content Copyright Nanyang Technological University 24

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*/
{
    ....
}
```

Content Copyright Nanyang Technological University 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*/
{
    ....
}
```

Content Copyright Nanyang Technological University 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

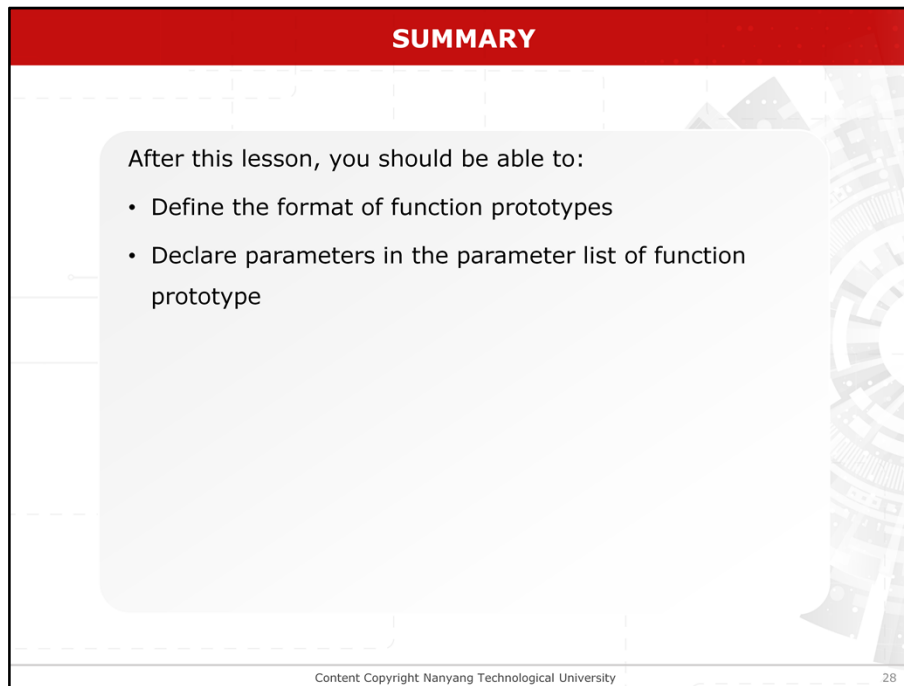
```
#include <stdio.h>

int main()
{
    int x;
    int fact(int); // function prototype
    x = fact(5); // use fact() here
    ....
}

int fact(int n) // function definition
{
    ....
}
```

Content Copyright Nanyang Technological University 27

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



SUMMARY

After this lesson, you should be able to:

- Define the format of function prototypes
- Declare parameters in the parameter list of function prototype

Content Copyright Nanyang Technological University 28

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