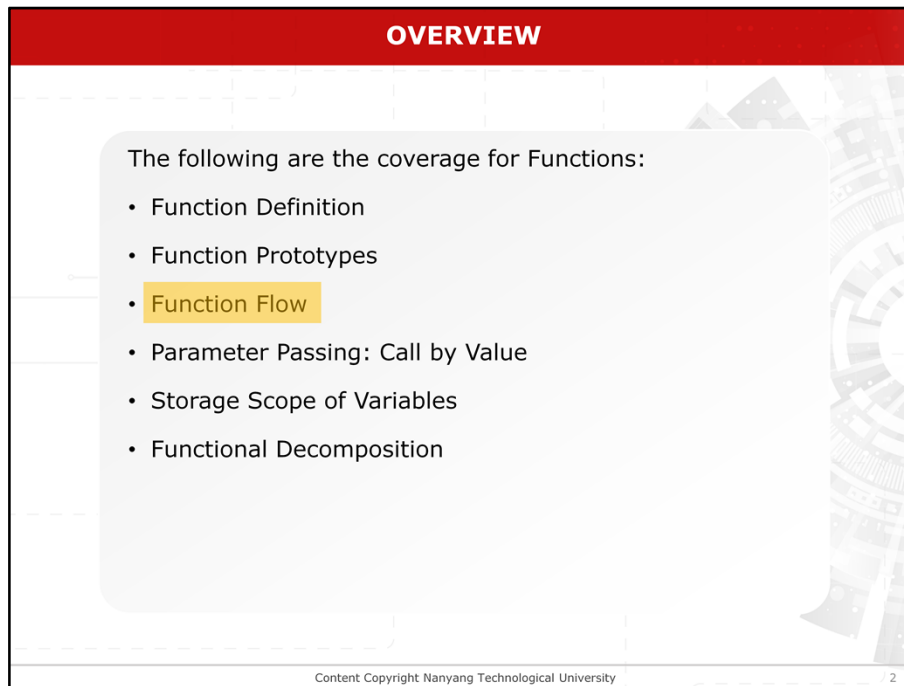


This lesson is on Functions

The slide features a red header with the word "OVERVIEW" in white. Below the header, a light gray rounded rectangle contains the text "The following are the coverage for Functions:" followed by a bulleted list. The list includes "Function Definition", "Function Prototypes", "Function Flow" (which is highlighted with a yellow background), "Parameter Passing: Call by Value", "Storage Scope of Variables", and "Functional Decomposition". The background of the slide has a faint, stylized architectural pattern on the right side. At the bottom, there is a thin white bar containing the text "Content Copyright Nanyang Technological University" and the number "2".

## 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

There are 6 main sections to cover for Functions. This video focused on the 3<sup>rd</sup> topic: Function flow



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 text '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, there is a footer containing the text 'Content Copyright Nanyang Technological University' and the 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 call

Content Copyright Nanyang Technological University 5

Define the format of function call

**LEARNING OBJECTIVES**

At this lesson, you should be able to:

- Define the format of function call
- Design and execute program using function flow

Content Copyright Nanyang Technological University 6

- Design and execute program using Function flow

**FUNCTION CALL**

- A function call causes the function to be executed.

Content Copyright Nanyang Technological University 7

**Function Call**

A function is executed when it is called.

## FUNCTION CALL

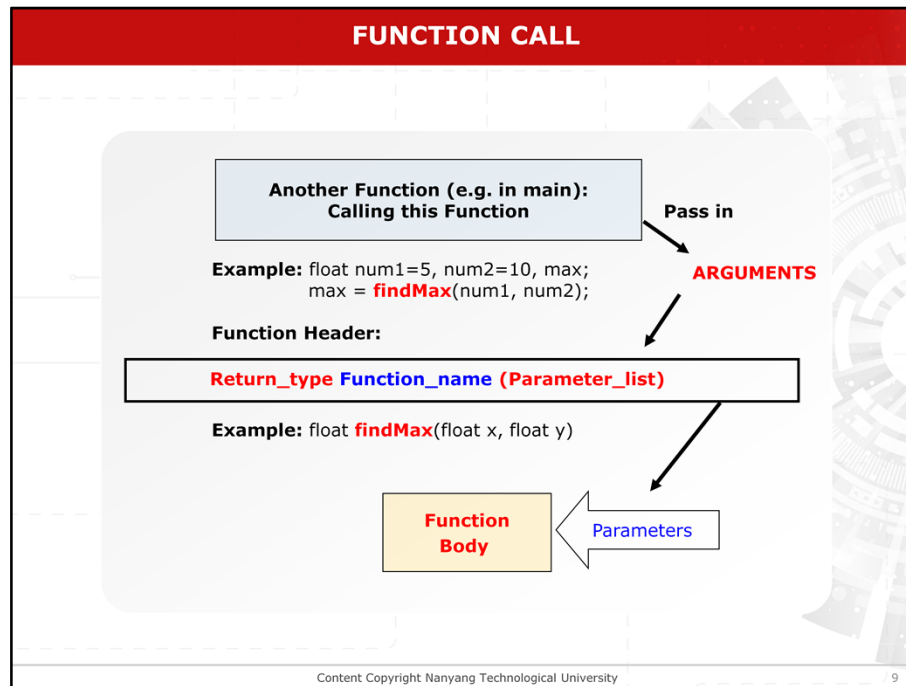
- A function call causes the function to be executed.
- A function call has the following format:  
**Function\_name (Argument\_list);**

Content Copyright Nanyang Technological University 8

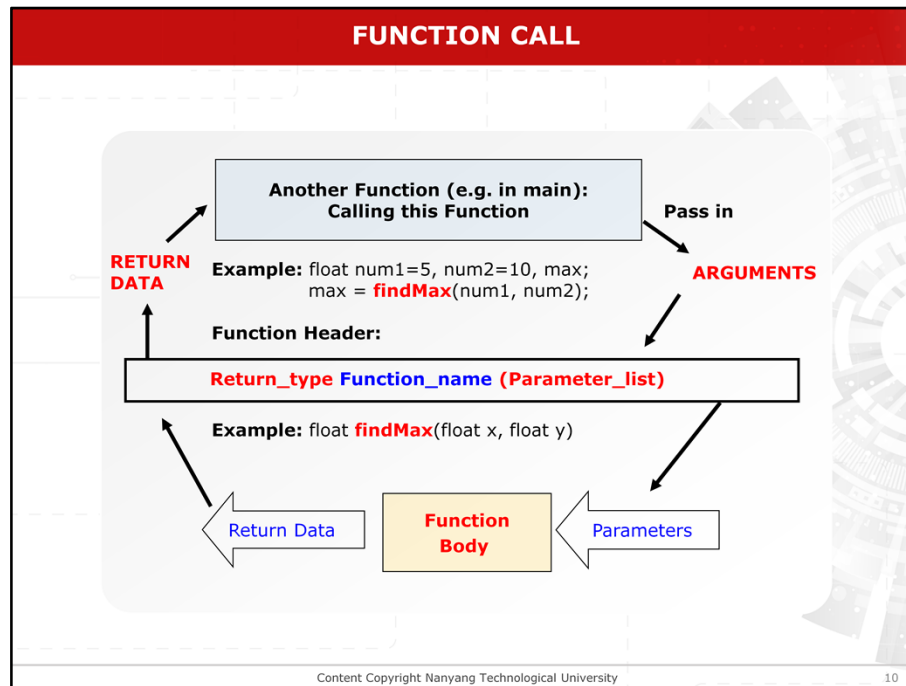
### Function Call

A function is executed when it is called. A function call has the format as shown:





The function can be called by using the function name followed by a list of *arguments* as shown here



**FUNCTION CALL**

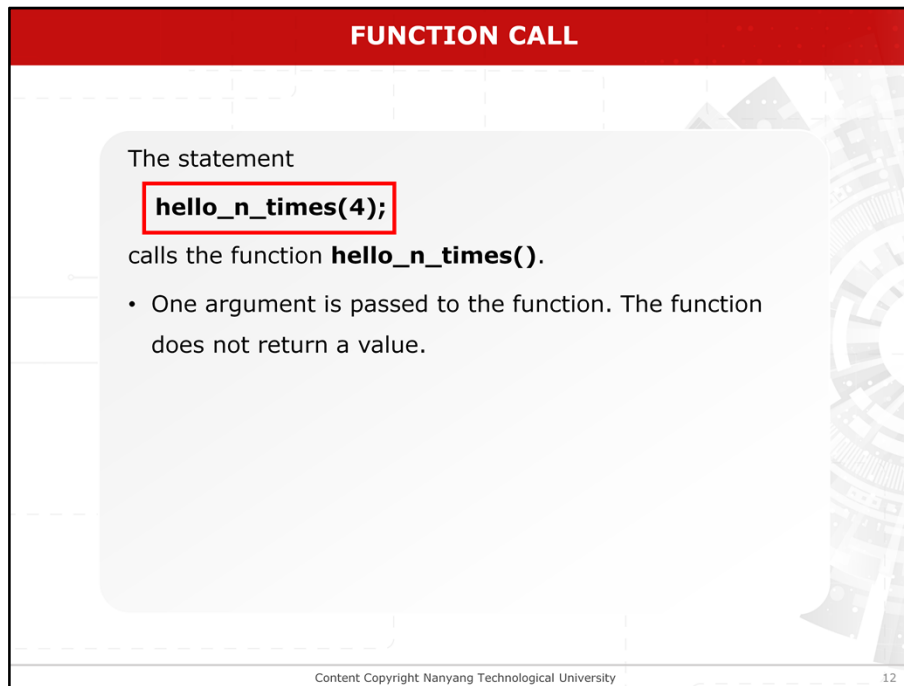
The statement

**hello\_n\_times(4);**

calls the function **hello\_n\_times()**.

Content Copyright Nanyang Technological University 11

The statement **hello\_n\_times(4)** calls the function **hello\_n\_times()**.



**FUNCTION CALL**

The statement

**hello\_n\_times(4);**

calls the function **hello\_n\_times()**.

- One argument is passed to the function. The function does not return a value.

Content Copyright Nanyang Technological University 12

One argument is passed to the function

### FUNCTION CALL

The statement

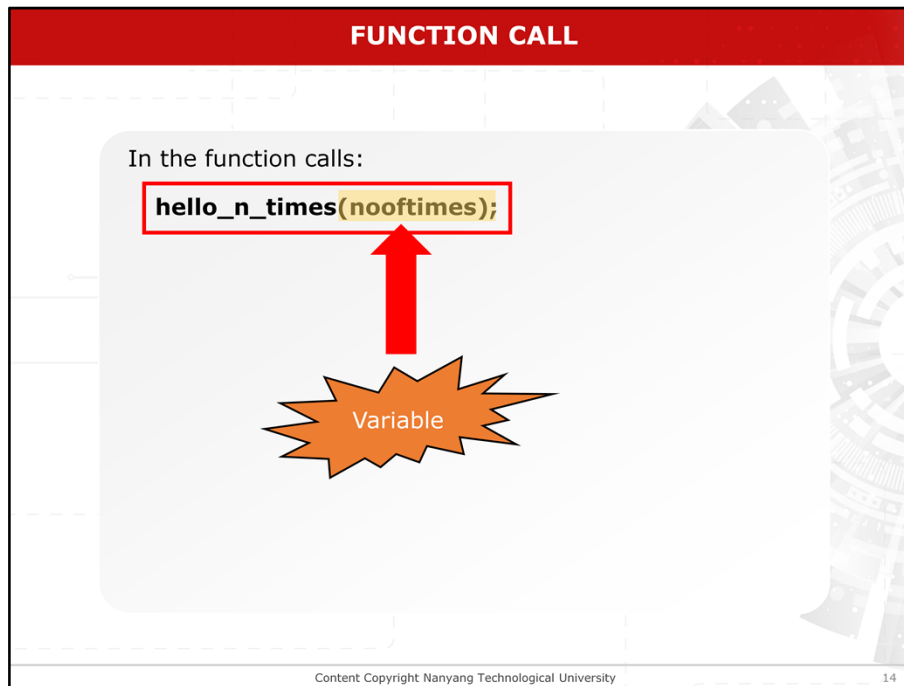
```
hello_n_times(4);
```

calls the function **hello\_n\_times()**.

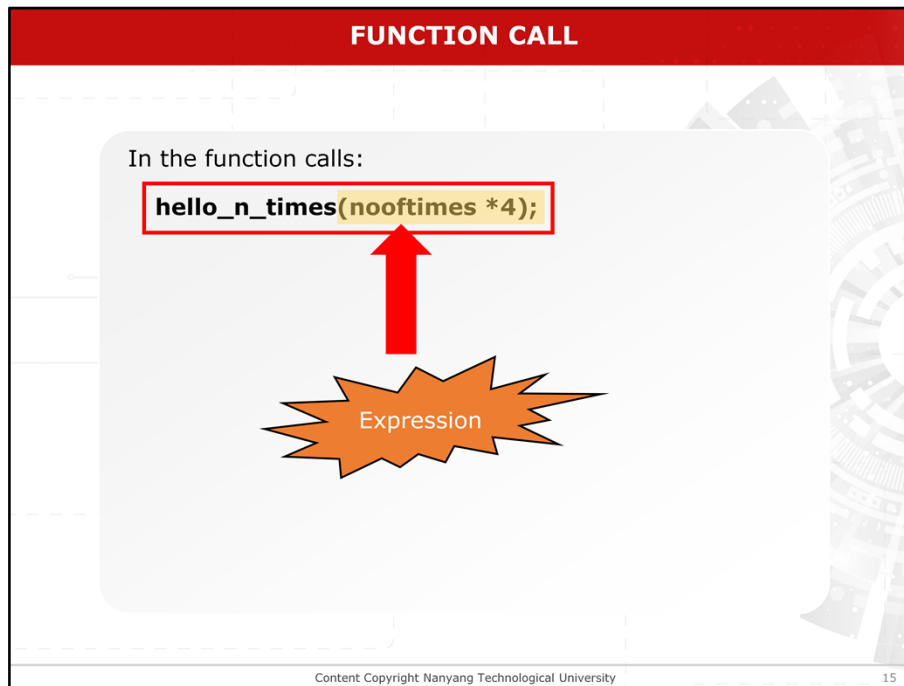
- One argument is passed to the function. The function does not return a value.
- Function arguments can be constants, variables or expressions.

Content Copyright Nanyang Technological University 13

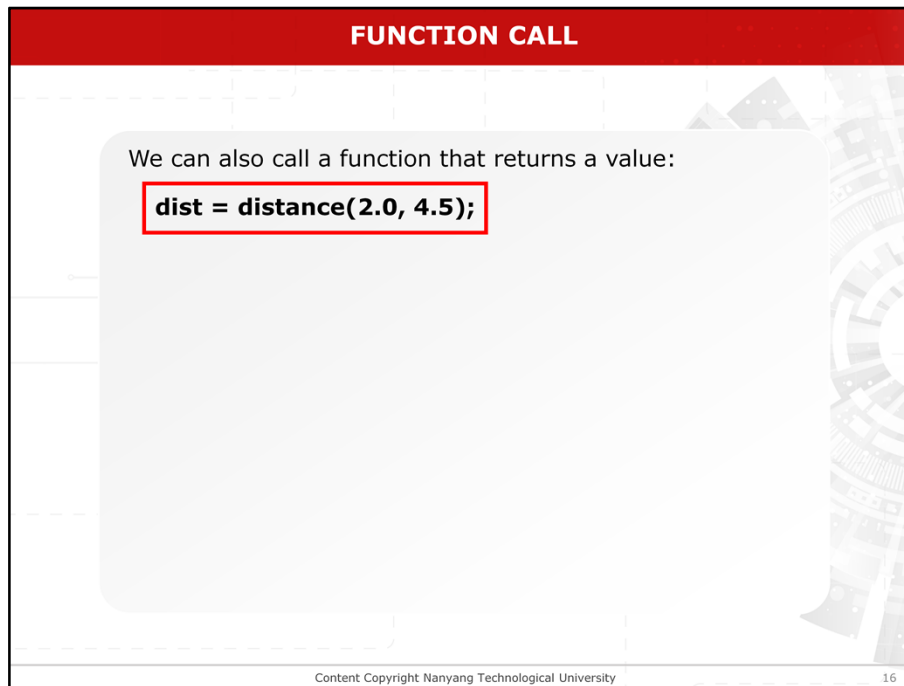
Function arguments can be constants, variables or expressions.



In the function call shown here, the argument **nooftimes** is a variable,



Whereas In the function calls shown here, where the argument **nooftimes\*4** is a expression



**FUNCTION CALL**

We can also call a function that returns a value:

```
dist = distance(2.0, 4.5);
```

Content Copyright Nanyang Technological University 16

We can also call a function that returns a value as shown here.



### FUNCTION CALL

We can also call a function that returns a value:

```
dist = distance(2.0, 4.5);
```

- The function call has two arguments, separated by a comma.

Content Copyright Nanyang Technological University 17

The function call has two arguments, separated by a comma.

### FUNCTION CALL

We can also call a function that returns a value:

```
dist = distance(2.0, 4.5);
```

- The function call has two arguments, separated by a comma.
- The function also returns a value.

Content Copyright Nanyang Technological University 18

The function also returns a value.

### FUNCTION CALL

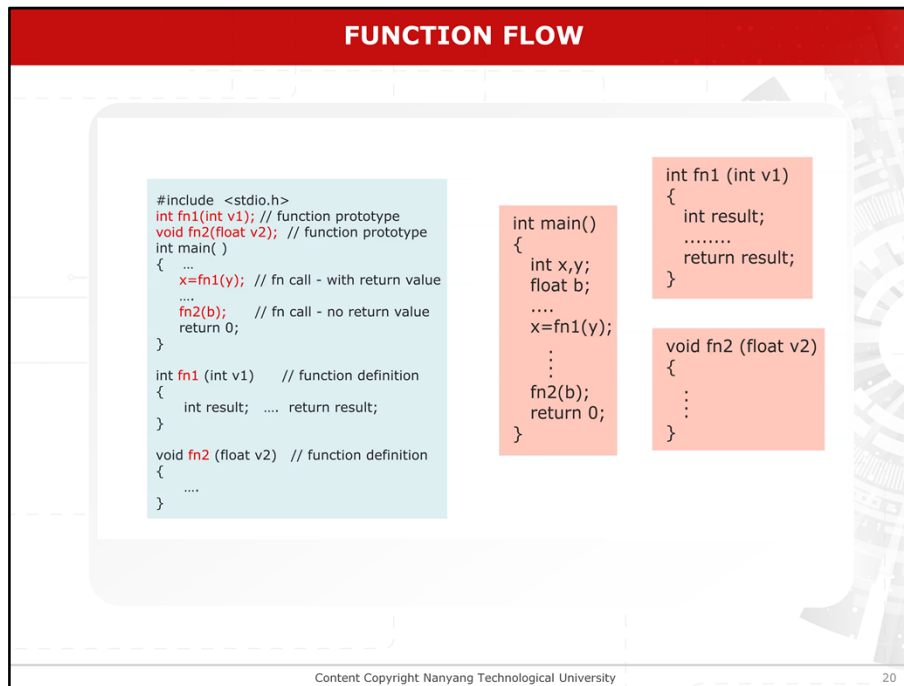
We can also call a function that returns a value:

```
dist = distance(2.0, 4.5);
```

- The function call has two arguments, separated by a comma.
- The function also returns a value.
- The returned value is assigned to the variable **dist**.

Content Copyright Nanyang Technological University 19

The returned value is assigned to the variable **dist**.



### Function Flow

In the program, the **main()** function will start the execution. When the function **fn1()** is called, the program transfers the control to the **fn1()** function which then starts execution. As **fn1()** will return a value back to the calling function, the statements in the function body of **fn1()** will be executed until a **return** statement is encountered. Control is then transferred back to the **main()** function. The value of the variable **result** will be assigned to the variable **x** in **main()**. The next statement after the function call then starts execution. When the second function **fn2()** is called. The control is then transferred to **fn2()**. The function will execute until the end of the function body. Control will then be transferred back to the **main()** function.

**FUNCTION FLOW: EXAMPLES**

```

#include <stdio.h>
char findGrade(float marks);
int main( )
{
    char answer;
    answer = findGrade(68.5);
    printf("Grade is %c", answer);
    return 0;
}

char findGrade(float marks){
    char grade; // variable

    if (marks >= 50)
        grade = 'P';
    else
        grade = 'F';
    return grade;
}

```

Click to play

Content Copyright Nanyang Technological University

21

### Function Flow: findGrade()

In the program, the **main()** function calls the function **findGrade()**. When the statement **answer = findGrade(68.5)** is executed, it calls the function **findGrade()**. Control is then transferred to the function **findGrade()**. Information is passed between the calling function and the called function through the argument. In this case, the function receives one argument with the value of **68.5**. It is assigned to the corresponding parameter in the function definition to compute the grade. When the execution of statements in the function body encounters the **return** statement, the control is then transferred back to the **main()** function, and the statement just after the function call in **main()** will continue to execute. The name for parameter needs not be the same as function argument. However, the number of arguments and the data type of the arguments must be the same as parameters defined in function definition. In the program, the argument **68.5** must correspond to the parameter **marks** in the function call. Notice that the function prototype is declared as: **float findGrade(float marks)** and is placed at the beginning of the program before the **main()** function.

**FUNCTION FLOW: EXAMPLES**

```
#include <stdio.h>
float areaOfCircle(float);
int main( )
{
    float answer;
    answer = areaOfCircle(2.5);
    printf("Area is %.1f", answer);
    return 0;
}

float areaOfCircle(float radius){
    const float pi = 3.14;
    float area;
    area = pi*radius*radius;
    return area;
}
```

Content Copyright Nanyang Technological University 22

### Function Flow: areaOfCircle()

In the program, the **main()** function calls the function **areaOfCircle()**. When the statement **answer = areaOfCircle(2.5)** is executed, it calls the function **areaOfCircle ()**. Control is then transferred to the function **areaOfCircle()**. Information is passed between the calling function and the called function through the argument. In this case, the function receives one argument with the value of **2.5**. It is assigned to the corresponding parameter in the function definition to compute the area of the circle. When the execution of statements in the function body encounters the **return** statement, the control is then transferred back to the **main()** function, and the statement just after the function call in **main()** will continue to execute. Notice that the function prototype is declared as **float areaOfCircle(float)** and is placed at the beginning of the program before the **main()** function.

## SUMMARY

After this lesson, you should be able to:

- Define the format of function call
- Design and execute program using function flow

Content Copyright Nanyang Technological University 23

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