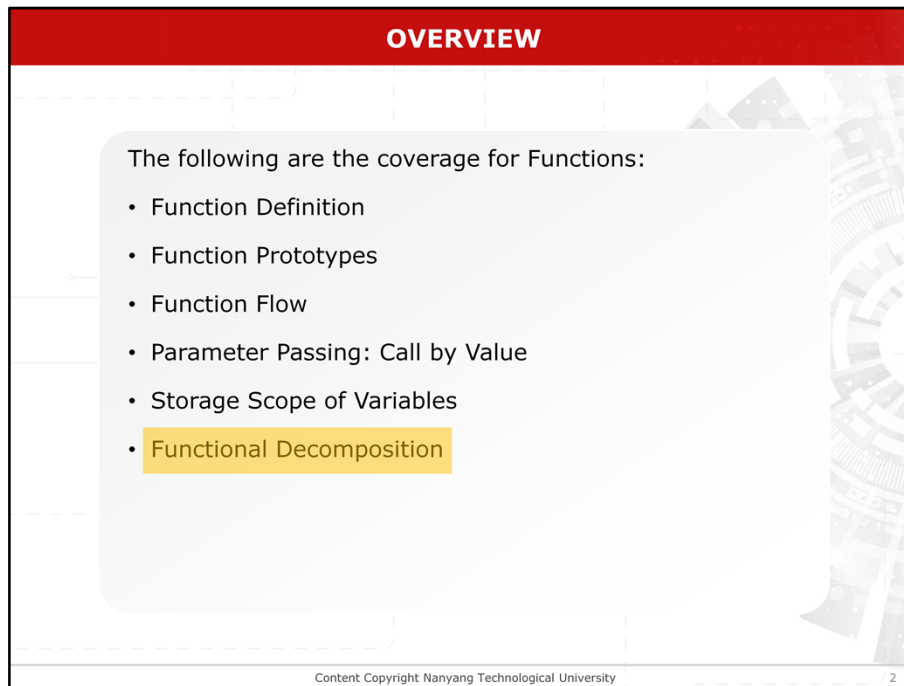


This lesson is on functions

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 background of the slide has a faint, stylized architectural drawing of a building. At the bottom, a thin white bar contains the copyright notice and a page number.

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 5th topic: storage scope of variables



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:

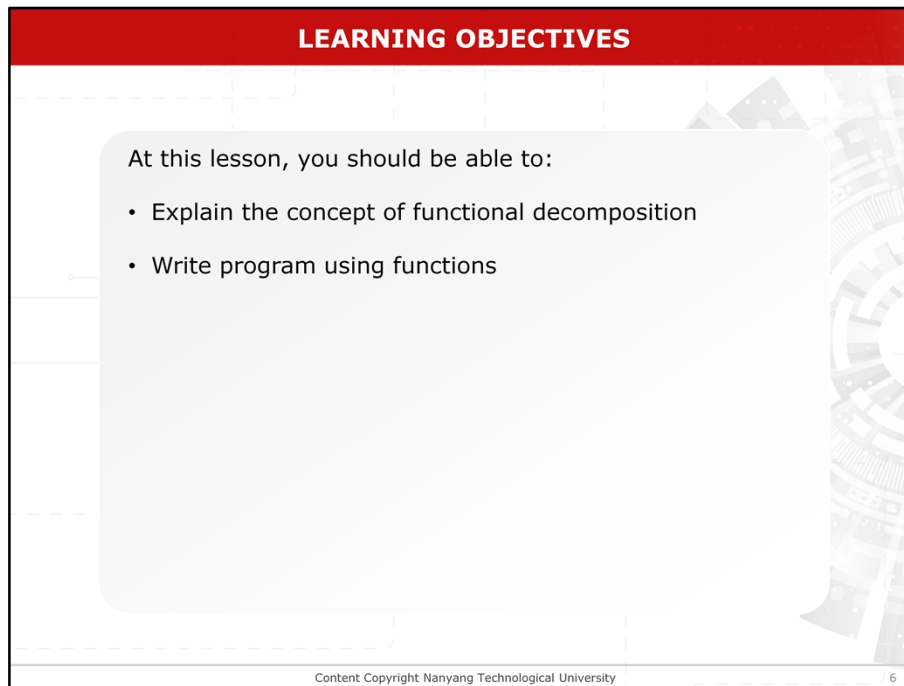
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Explain the concept of function decomposition

Content Copyright Nanyang Technological University 5

Explain the concept of function decomposition



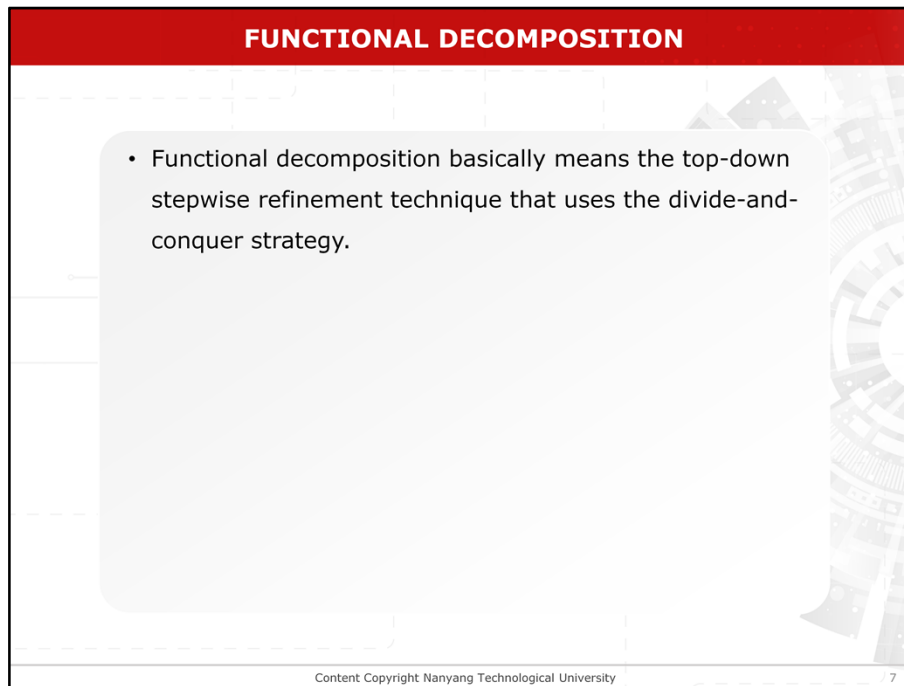
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Explain the concept of functional decomposition
- Write program using functions

Content Copyright Nanyang Technological University 6

Write program using functions

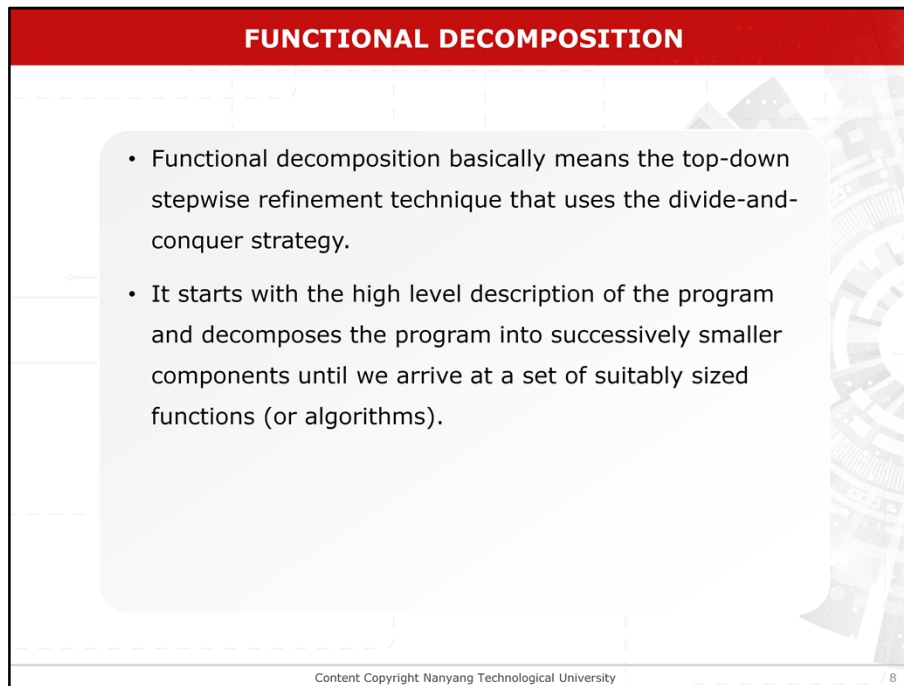
A presentation slide titled "FUNCTIONAL DECOMPOSITION" in a red header bar. The slide features a light gray background with a faint architectural pattern on the right side. A central gray rounded rectangle contains a bulleted list. At the bottom, a thin black bar contains the text "Content Copyright Nanyang Technological University" and the number "7".

FUNCTIONAL DECOMPOSITION

- Functional decomposition basically means the top-down stepwise refinement technique that uses the divide-and-conquer strategy.

Content Copyright Nanyang Technological University 7

Functional decomposition basically means the top-down stepwise refinement technique that uses the divide-and-conquer strategy.

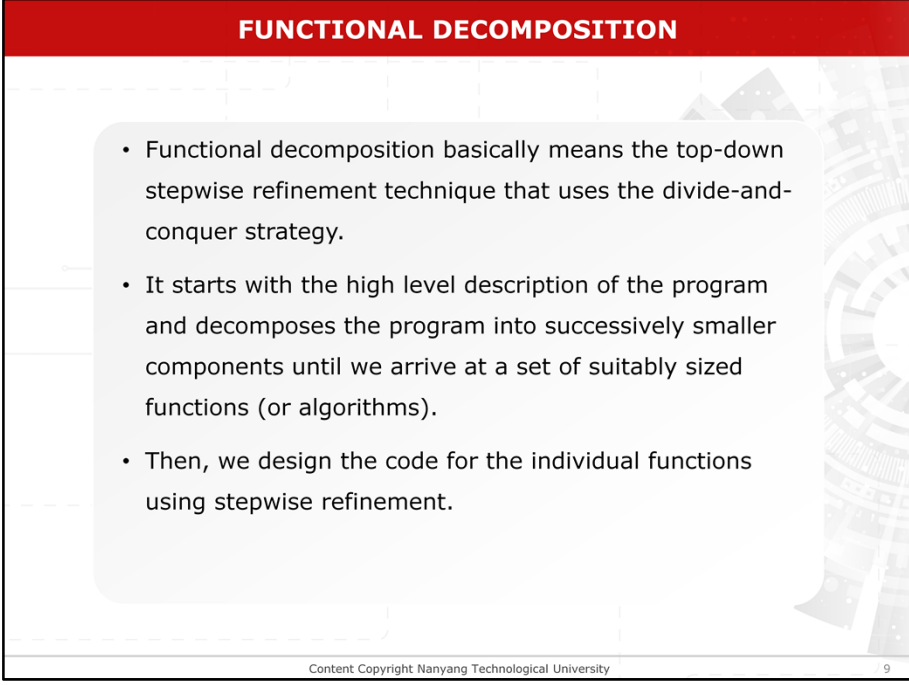
A presentation slide titled 'FUNCTIONAL DECOMPOSITION' in a red header bar. The slide features a light gray background with a faint architectural pattern on the right side. A central white box contains two bullet points. At the bottom, there is a footer with the text 'Content Copyright Nanyang Technological University' and the number '8'.

FUNCTIONAL DECOMPOSITION

- Functional decomposition basically means the top-down stepwise refinement technique that uses the divide-and-conquer strategy.
- It starts with the high level description of the program and decomposes the program into successively smaller components until we arrive at a set of suitably sized functions (or algorithms).

Content Copyright Nanyang Technological University 8

It starts with the high level description of the program and decomposes the program into successively smaller components until we arrive at a set of suitably sized functions (or algorithms).

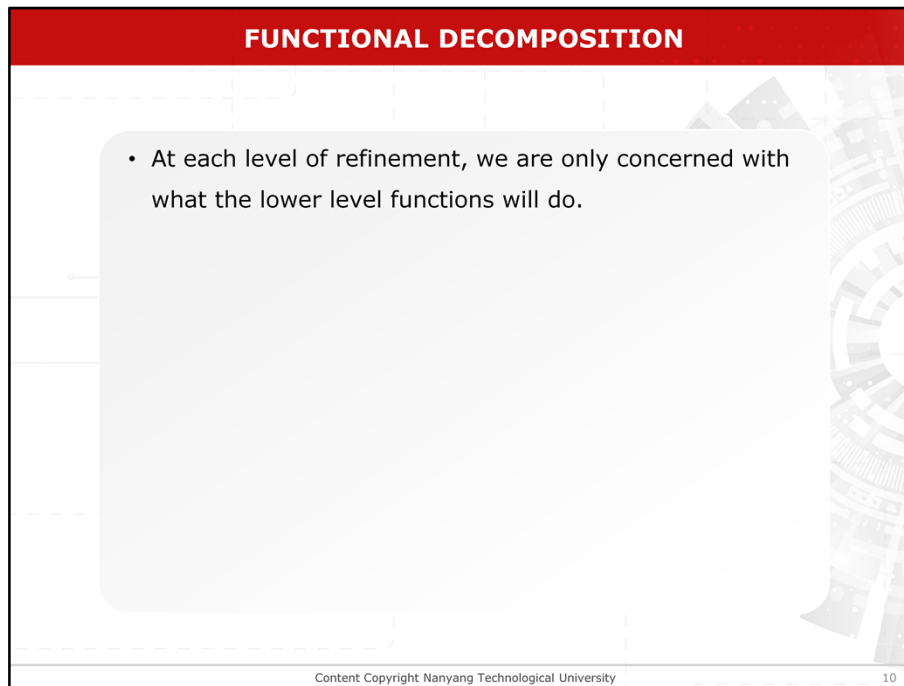
A diagram titled "FUNCTIONAL DECOMPOSITION" in a red header. The background features a faint architectural drawing of a building. A central grey box contains three bullet points. At the bottom, a footer bar contains the text "Content Copyright Nanyang Technological University" and the number "9".

FUNCTIONAL DECOMPOSITION

- Functional decomposition basically means the top-down stepwise refinement technique that uses the divide-and-conquer strategy.
- It starts with the high level description of the program and decomposes the program into successively smaller components until we arrive at a set of suitably sized functions (or algorithms).
- Then, we design the code for the individual functions using stepwise refinement.

Content Copyright Nanyang Technological University 9

Then, we design the code for the individual functions using stepwise refinement.

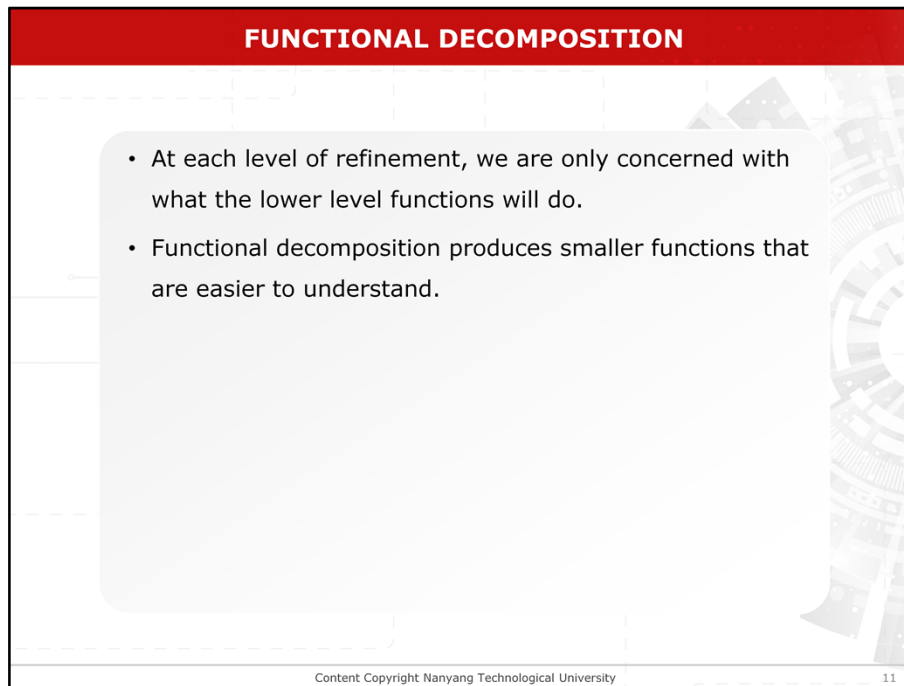
A presentation slide titled "FUNCTIONAL DECOMPOSITION" in a red header bar. The slide features a light gray background with a faint, stylized architectural drawing of a building on the right side. A large, light gray rounded rectangle is centered on the slide, containing a single bullet point. At the bottom of the slide, there is a thin white bar with the text "Content Copyright Nanyang Technological University" on the left and the number "10" on the right.

FUNCTIONAL DECOMPOSITION

- At each level of refinement, we are only concerned with what the lower level functions will do.

Content Copyright Nanyang Technological University 10

At each level of refinement, we are only concerned with what the lower level functions will do.

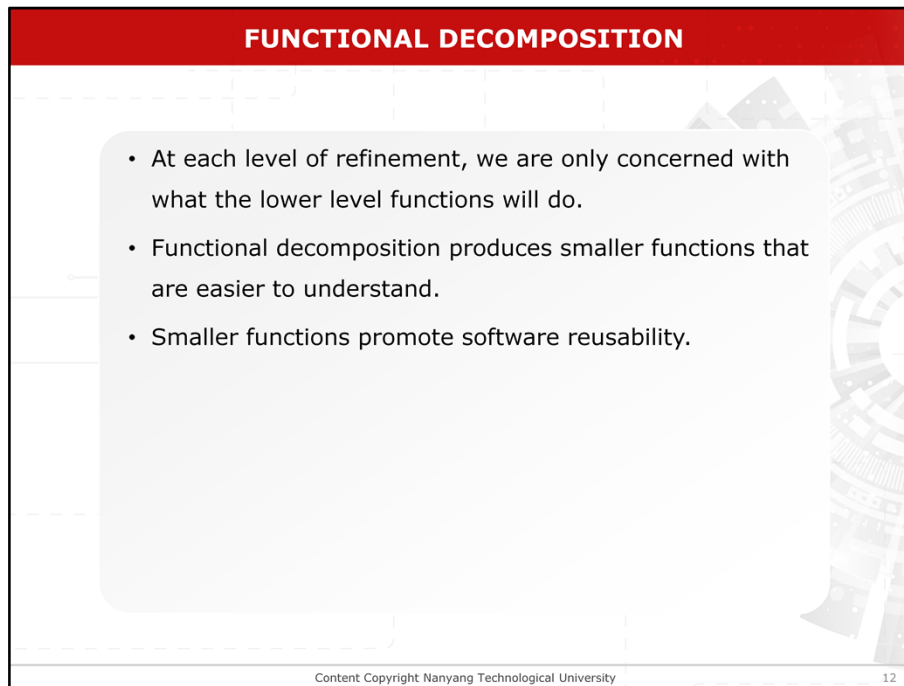
A presentation slide titled "FUNCTIONAL DECOMPOSITION" in a red header bar. The slide features a light gray background with a faint architectural pattern on the right side. A central white rounded rectangle contains two bullet points. At the bottom, a thin gray bar contains the text "Content Copyright Nanyang Technological University" on the left and the number "11" on the right.

FUNCTIONAL DECOMPOSITION

- At each level of refinement, we are only concerned with what the lower level functions will do.
- Functional decomposition produces smaller functions that are easier to understand.

Content Copyright Nanyang Technological University 11

Functional decomposition produces smaller functions that are easier to understand.

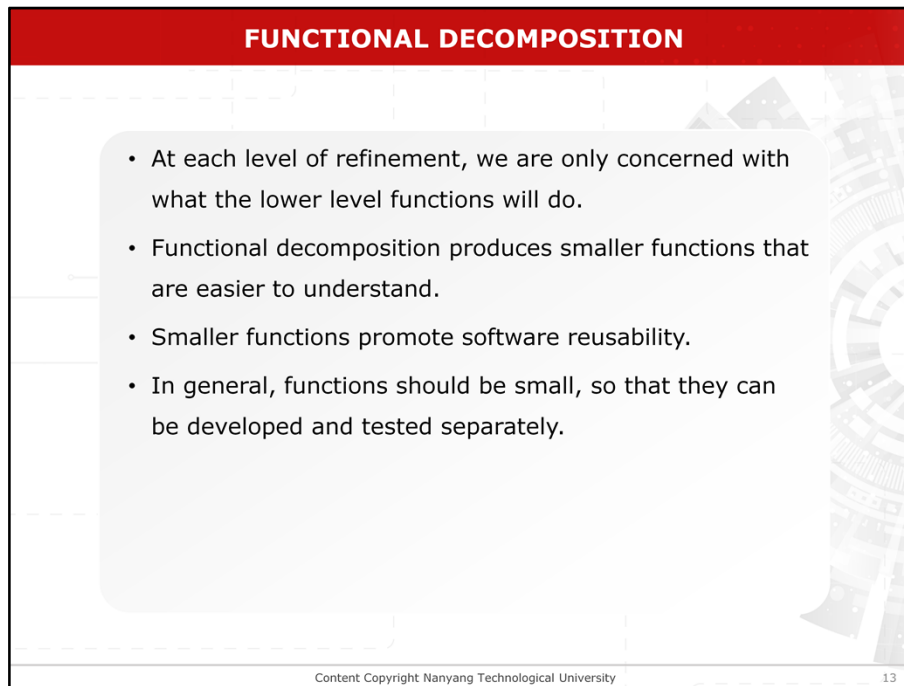
A presentation slide titled "FUNCTIONAL DECOMPOSITION" in a red header bar. The slide features a light gray background with a faint architectural pattern on the right side. A central white rounded rectangle contains a bulleted list. At the bottom, a thin gray bar contains the text "Content Copyright Nanyang Technological University" on the left and the number "12" on the right.

FUNCTIONAL DECOMPOSITION

- At each level of refinement, we are only concerned with what the lower level functions will do.
- Functional decomposition produces smaller functions that are easier to understand.
- Smaller functions promote software reusability.

Content Copyright Nanyang Technological University 12

Smaller functions promote software reusability.

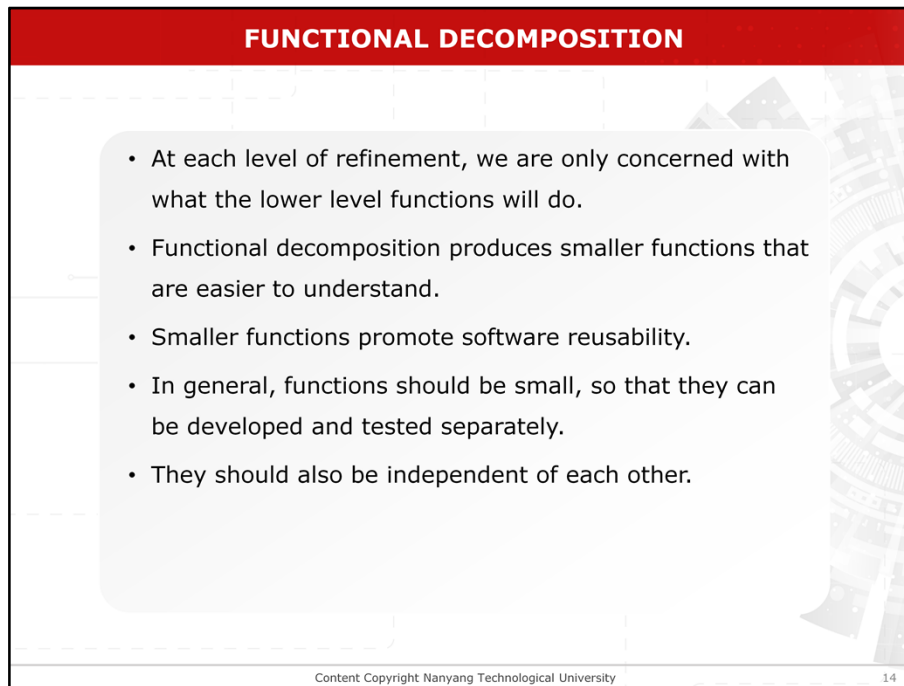
A presentation slide titled 'FUNCTIONAL DECOMPOSITION' in a red header bar. The slide features a light gray background with a faint architectural pattern on the right side. A central white rounded rectangle contains a bulleted list of four points. At the bottom, a thin gray bar contains the text 'Content Copyright Nanyang Technological University' on the left and the number '13' on the right.

FUNCTIONAL DECOMPOSITION

- At each level of refinement, we are only concerned with what the lower level functions will do.
- Functional decomposition produces smaller functions that are easier to understand.
- Smaller functions promote software reusability.
- In general, functions should be small, so that they can be developed and tested separately.

Content Copyright Nanyang Technological University 13

In general, functions should be small, so that they can be developed and tested separately.

A presentation slide titled 'FUNCTIONAL DECOMPOSITION' in a red header bar. The slide features a light gray background with a faint architectural drawing of a building on the right side. A central white rounded rectangle contains a bulleted list of five points. At the bottom, a thin gray bar contains the text 'Content Copyright Nanyang Technological University' on the left and the number '14' on the right.

FUNCTIONAL DECOMPOSITION

- At each level of refinement, we are only concerned with what the lower level functions will do.
- Functional decomposition produces smaller functions that are easier to understand.
- Smaller functions promote software reusability.
- In general, functions should be small, so that they can be developed and tested separately.
- They should also be independent of each other.

Content Copyright Nanyang Technological University 14

They should also be independent of each other.

FUNCTIONAL DECOMPOSITION: EXAMPLE

<pre>#include <stdio.h> #define ... int main() { } /* end. line 2000 */</pre>	<pre>#include <stdio.h> #define ... int main() { } /* line 20 */ float f1(float h) { } /* line 55 */ void f18() { } /* line 1560 */</pre>
--	--

Content Copyright Nanyang Technological University

15

In the program, it is decomposed into a number of smaller functions.

FUNCTIONAL DECOMPOSITION: EXAMPLE

<pre>#include <stdio.h> #define ... int main() { } /* end. line 2000 */</pre>	<pre>#include <stdio.h> #define ... int main() { } /* line 20 */ float f1(float h) { } /* line 55 */ void f18() { } /* line 1560 */</pre>
--	--

Content Copyright Nanyang Technological University 16

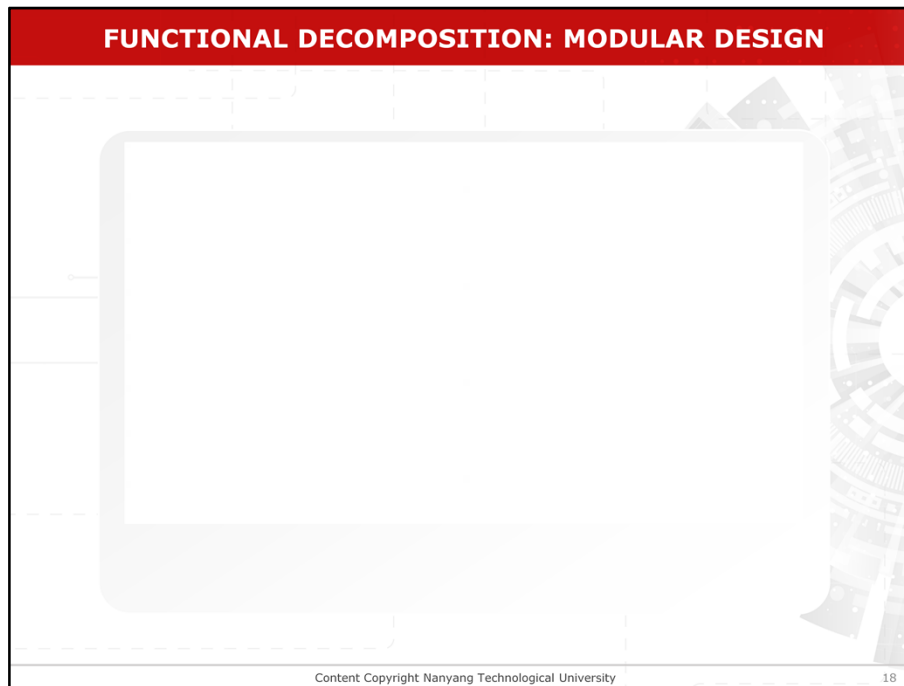
The **main()** function will start the execution of the program and call other functions to perform different required operations.

FUNCTIONAL DECOMPOSITION: EXAMPLE

<pre>#include <stdio.h> #define ... int main() { } /* end. line 2000 */</pre>	<pre>#include <stdio.h> #define ... int main() { } /* line 20 */ float f1(float h) { } /* line 55 */ void f18() { } /* line 1560 */</pre>
--	---

Content Copyright Nanyang Technological University

17



Functional Decomposition: Modular Design

Using the functional decomposition and top-down stepwise refinement technique, a problem is broken up into a number of smaller subproblems or functions. We then develop the algorithms for the functions. These functions can then be implemented using a programming language such as C. These functions are also called *modules*. This approach of designing programs as functional modules is called *modular design*. The functions or modules should be small and self-contained, so that they can be developed and tested separately. They should also be independent of each other. There are a number of advantages for modular design. Modular programs are easier to write and debug, since they can be developed and tested separately. Another advantage is that modular programs can be developed by different programmers as each programmer can work on a single module of the program independently. Moreover, a library of modules can be developed which can then be reused in other programs that require the same implementation. This can reduce program development time and enhances program reliability. Therefore, modular design can simplify program development significantly.

In the figure, it shows a typical structure of a program consisting of the main function and other functions for solving a problem. Usually, the functions could be quite complex as well, and they can be divided further into smaller functions.

PROBLEM: WRITING A C PROGRAM WITH A FUNCTION

```
/* Purpose: Write a function to find the maximum of two numbers.
The main() function calls the findMax() function using call by value */
#include <stdio.h>
float findMax(float x, float y);
int main()
{
    float x,y, max;
    printf("Enter 2 numbers: ");
    scanf("%f %f", &x, &y);
    max = findMax(x, y);
    printf("The max is %f.\n", max);
    return 0;
}

float findMax(float x, float y)
{
    float maxnum;
    /* write your code here */
}
```

OUTPUT

Content Copyright Nanyang Technological University 19

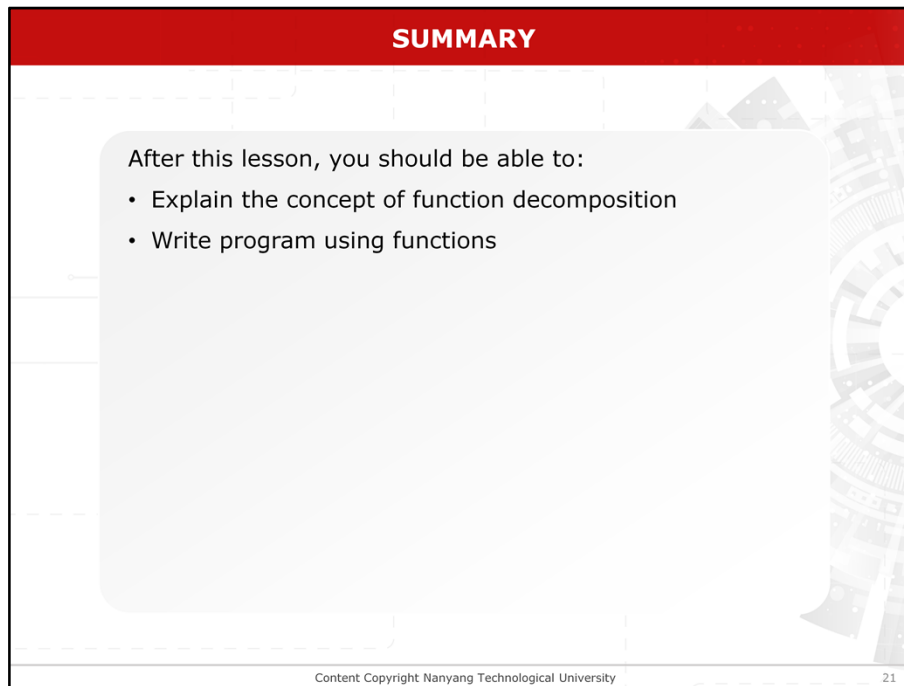
Programming Problem

The purpose of the program is to compute the maximum of two floating point numbers. Write a function called **findMax()** to achieve this purpose. The function will take in two arguments in floating point type, compute the maximum of the two values and return the maximum number to the calling function.



Programming Problem: Suggested Code

The suggested code for the function `findMax()` is given. It will take in two arguments **x** and **y** from the calling function, compute the maximum number and returns the maximum number to the calling function. The calling function **main()** will then stores the returned result in the variable **max** which is then printed on the screen.



SUMMARY

After this lesson, you should be able to:

- Explain the concept of function decomposition
- Write program using functions

Content Copyright Nanyang Technological University 21

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