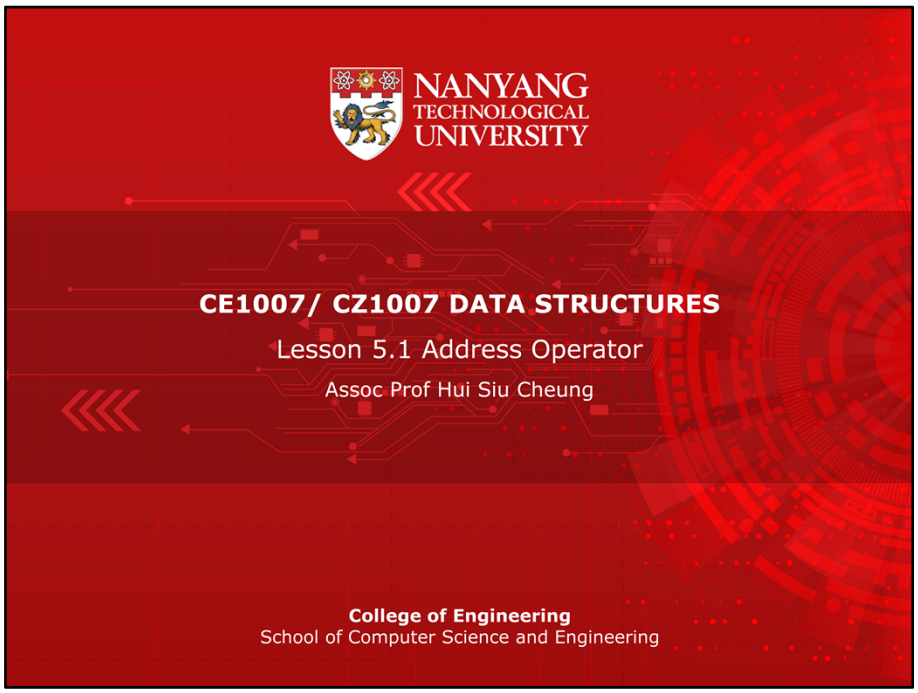
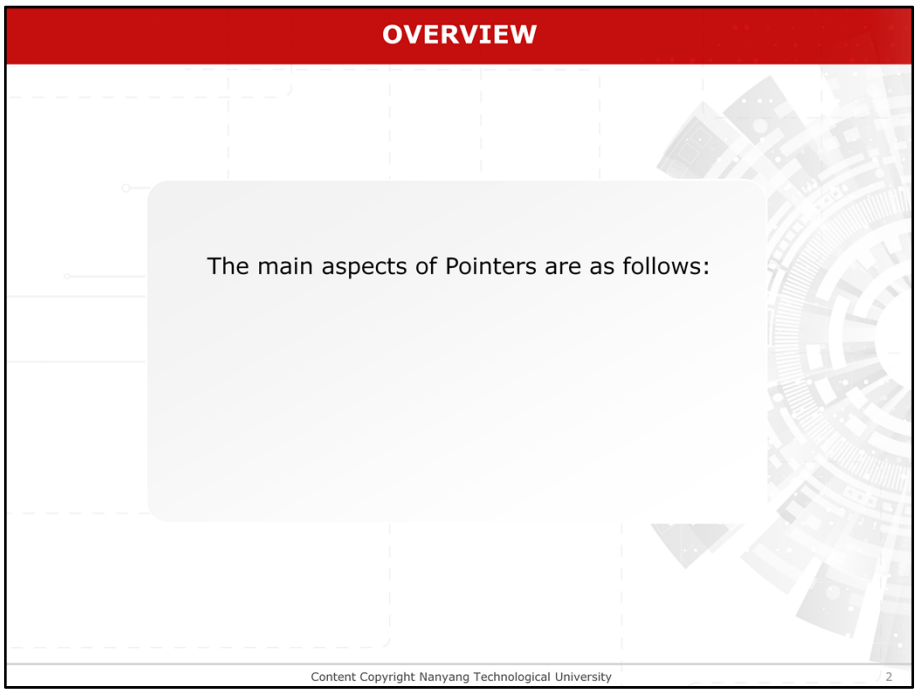
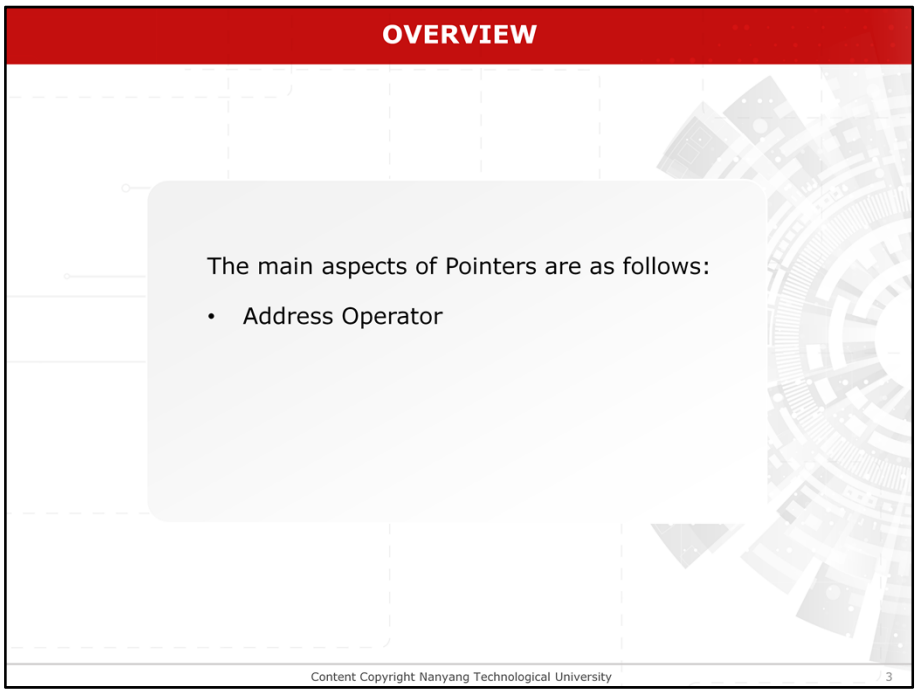


Lesson 5.1 Address Operator

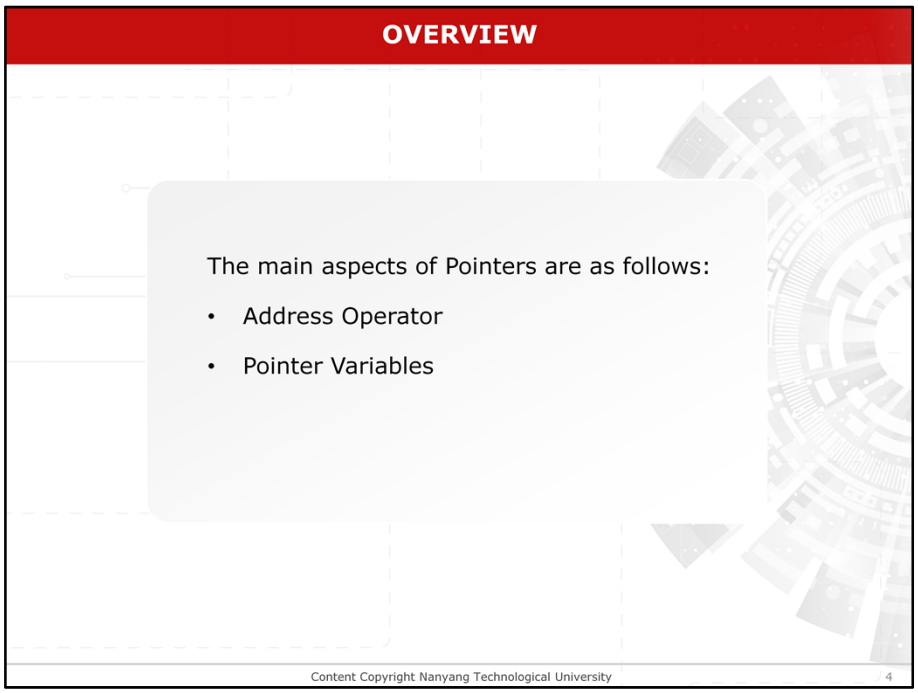
The concept of pointers is important in C programming. Pointer is a very powerful tool for the design of C programs. A pointer is a variable that holds the value of the address or memory location of another data object. In C, pointers can be used in many ways. This includes the passing of variable's address to functions to support call by reference, and the use of pointers for the processing of arrays and strings.



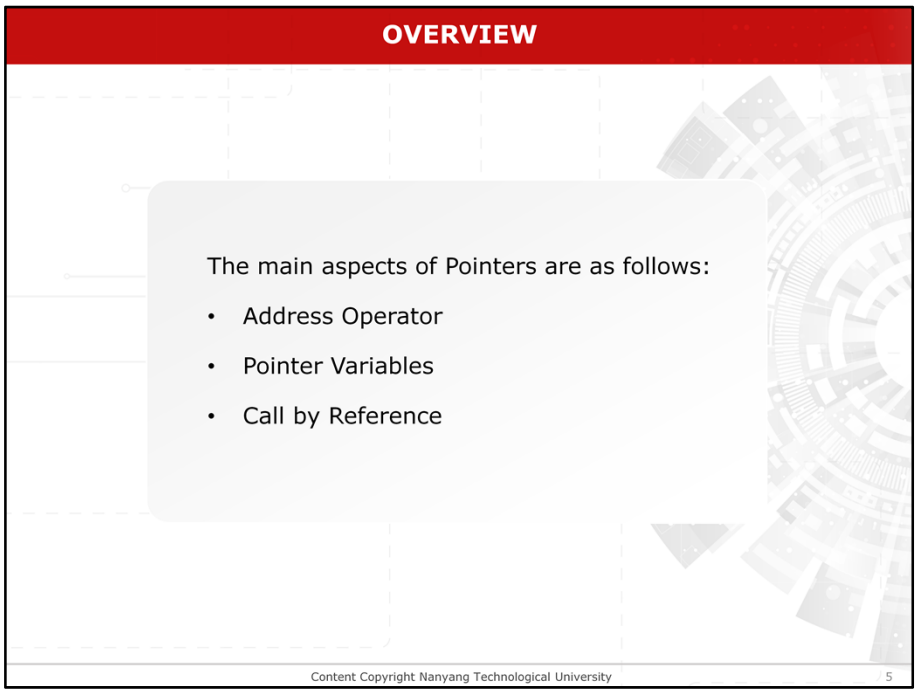
We will be covering the following main aspects of pointers.



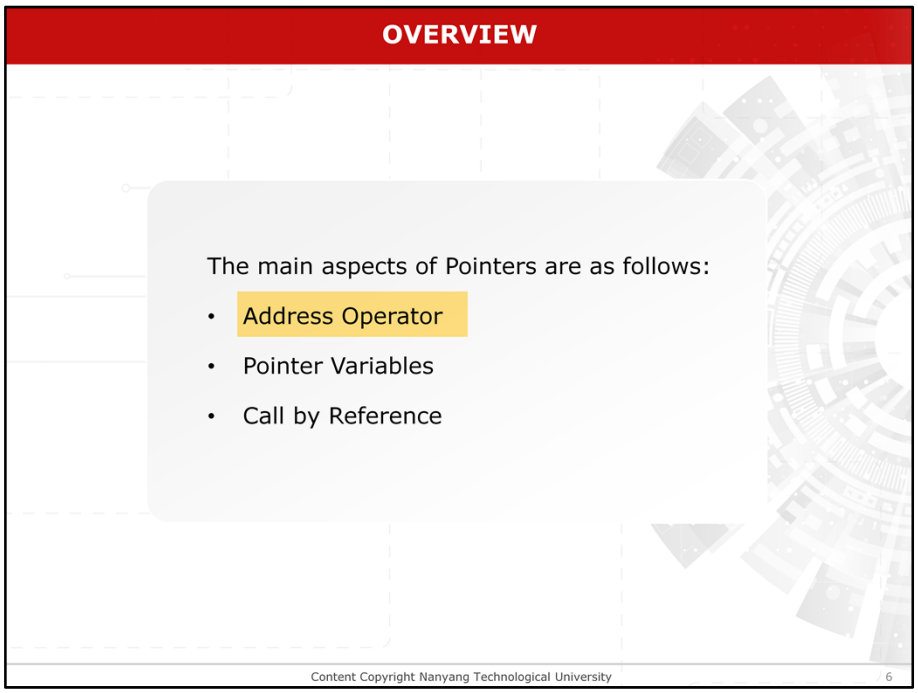
Address Operator



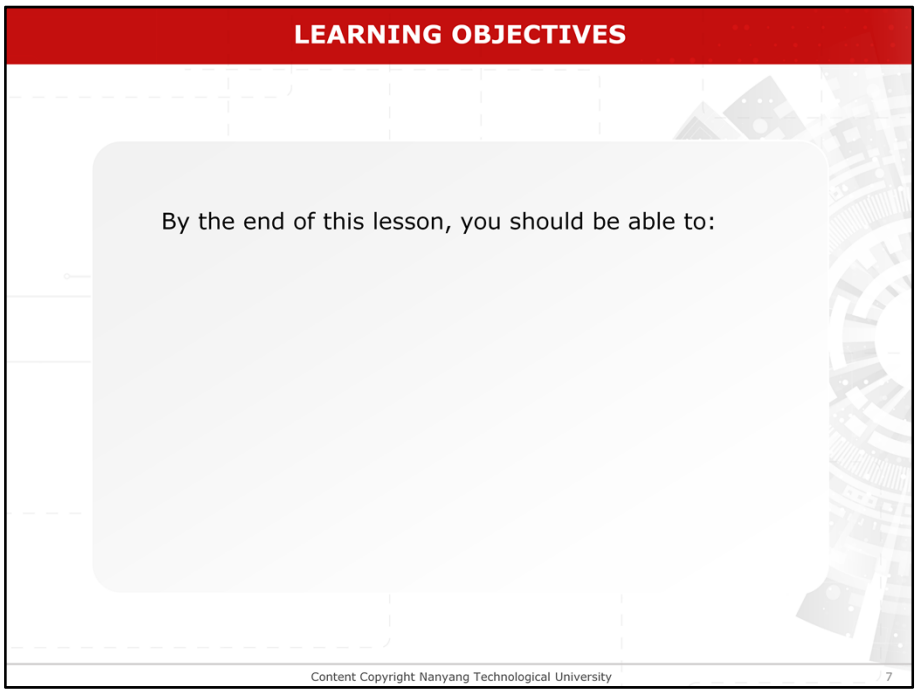
Pointer Variables



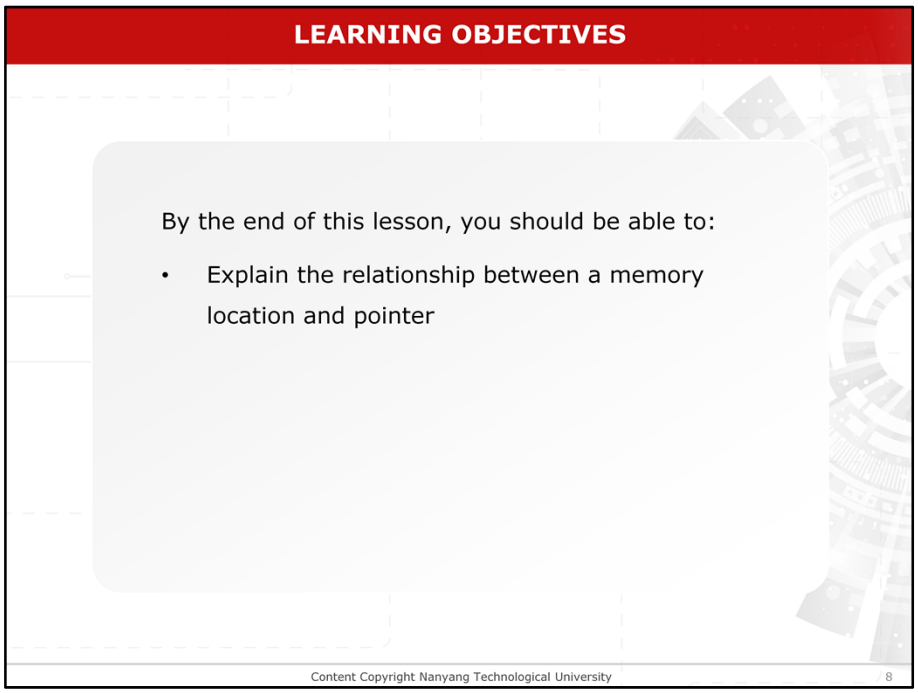
Call by Reference.



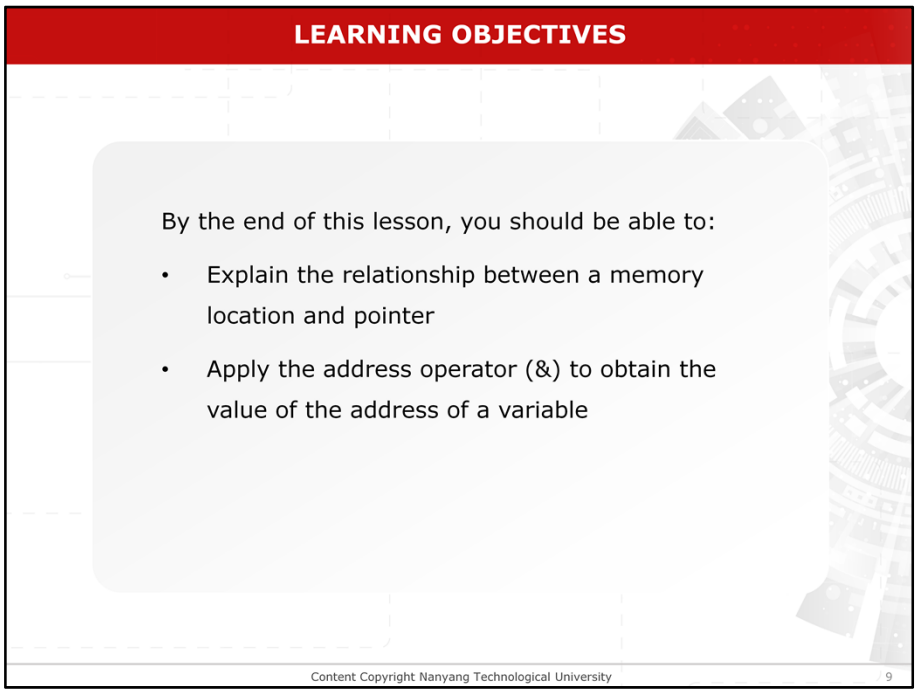
In this lesson, we will look into Address Operator



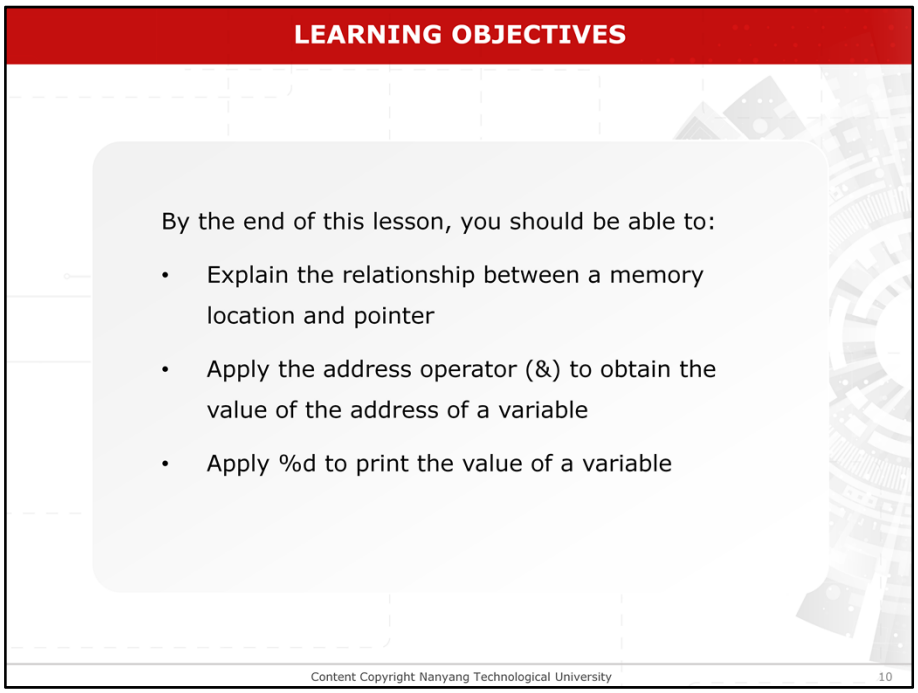
By the end of this lesson, you should be able to



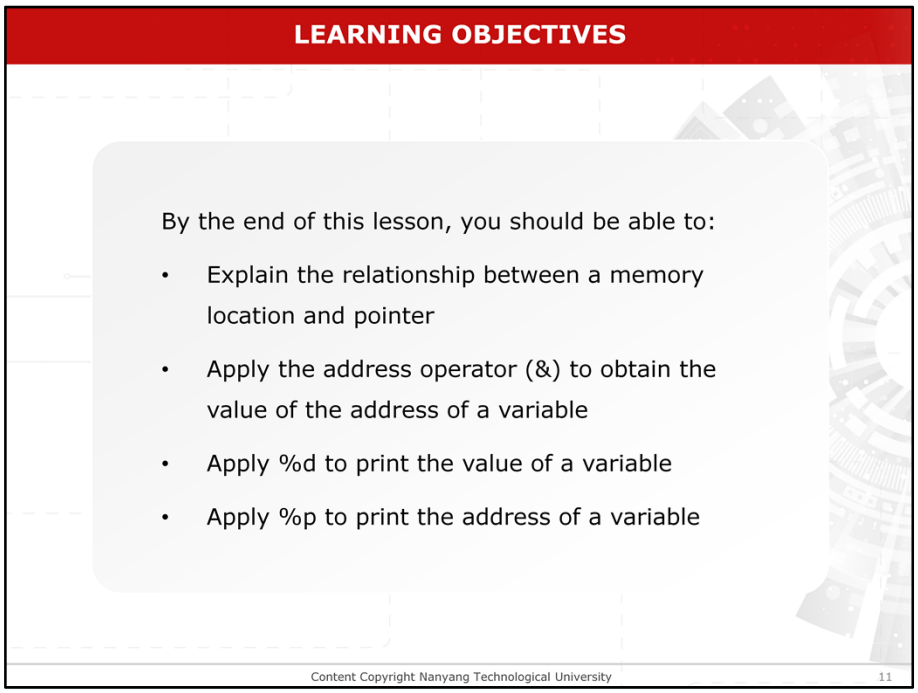
Explain the relationship between a memory location and pointers



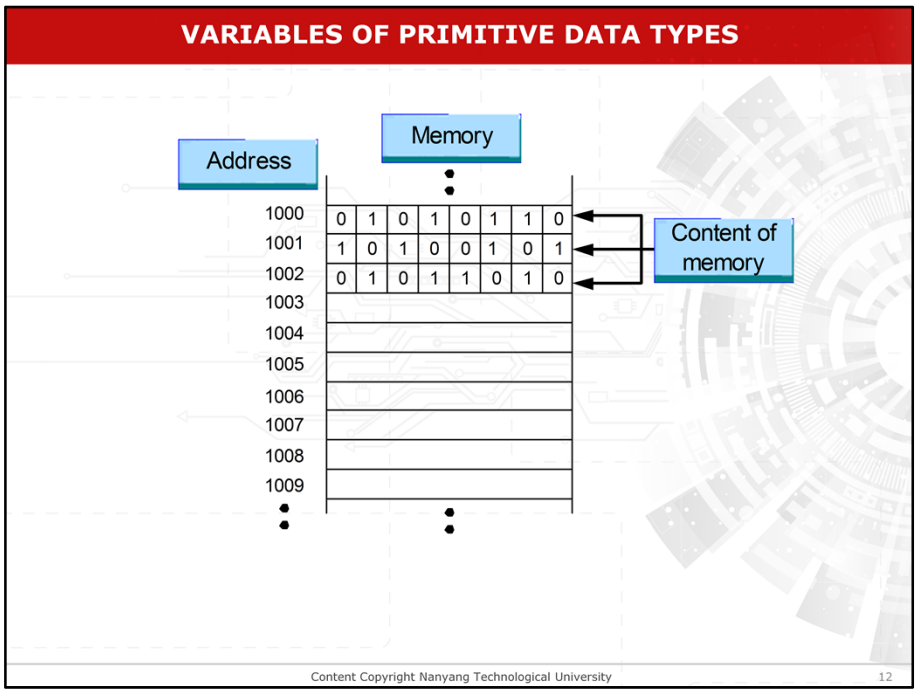
Apply the address operator (&) to obtain the value of the address of a variable.



Apply %d to print the value of a variable

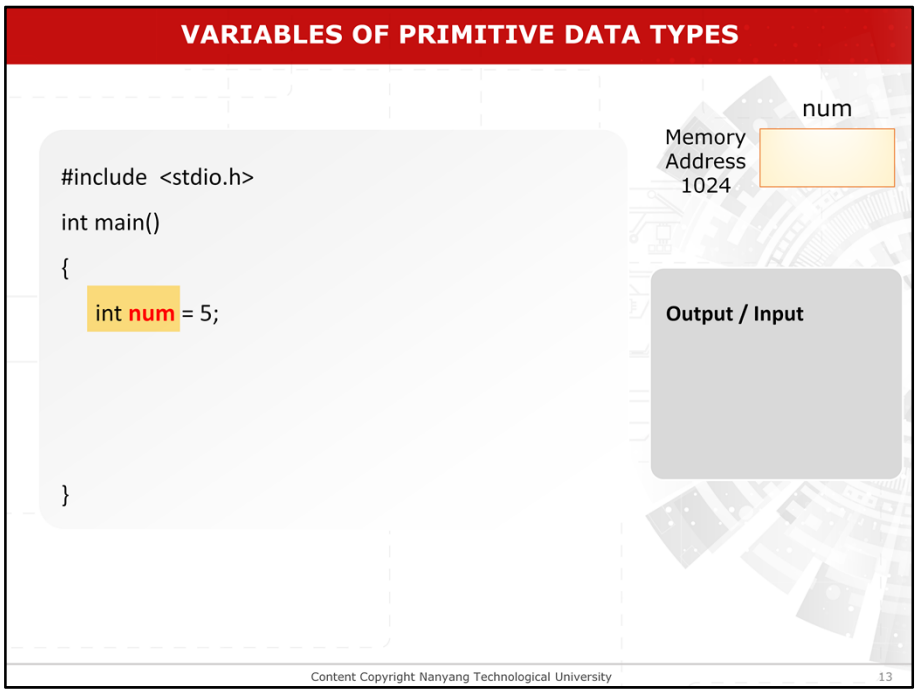


Apply %p to print the address of a variable

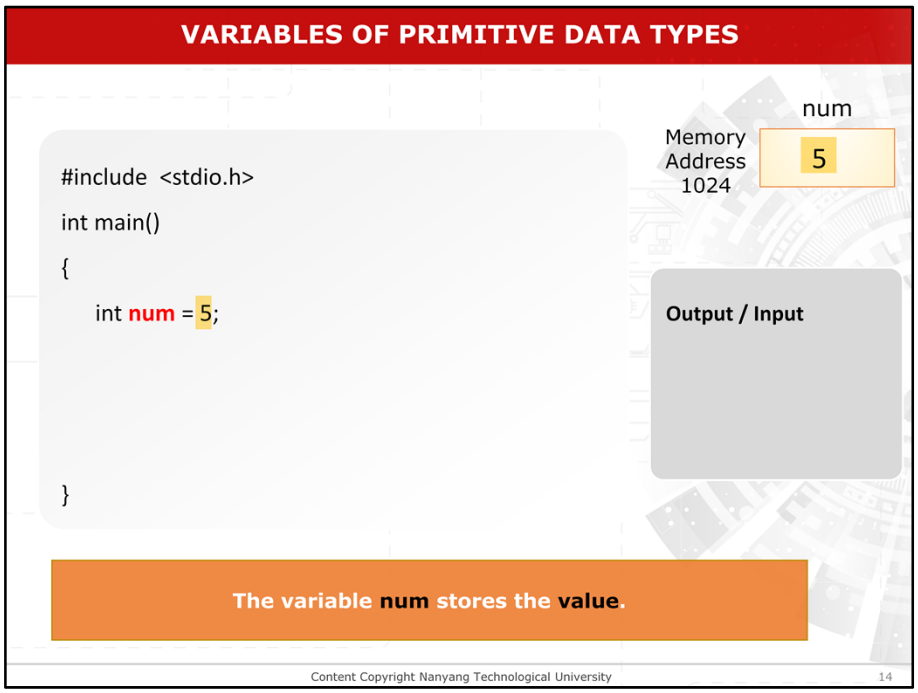


[insert animation done previously: CE1007_A2D_Call by Reference Example_V1.0]
[audio script for animation: CE1007 L4-1S4 storyboard ver 1.1]

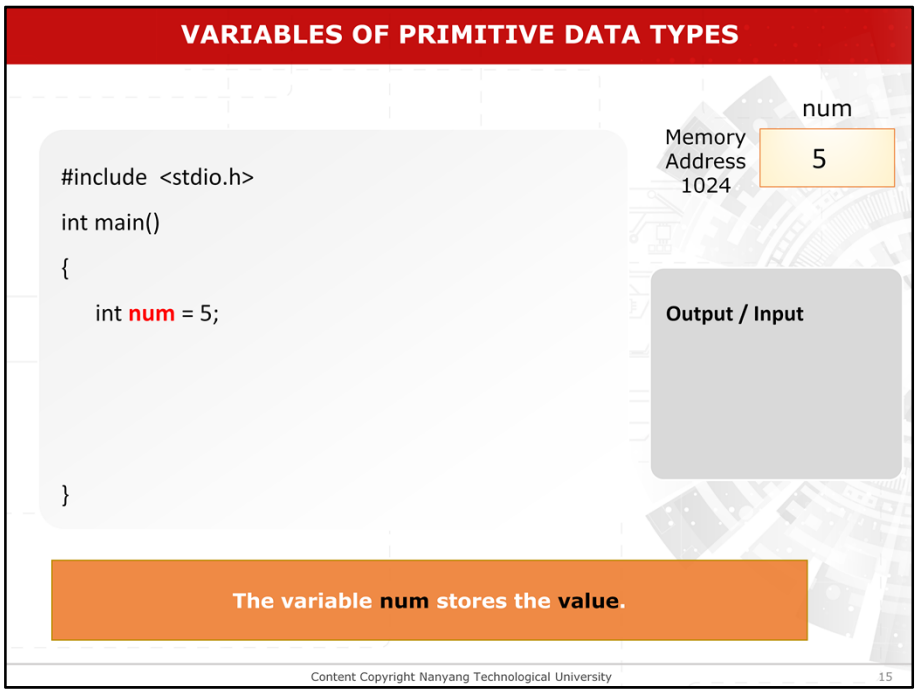
[refer to audio script for the sequence of audio: A computer's memory is used to store data objects such as variables and arrays in C. Each memory location has an address that can hold one byte of information. They are organized sequentially and the addresses range from 0 to the maximum size of the memory. When a variable is declared with a certain data type, the corresponding memory location will be allocated for the variable to hold the data of that type.
Note that variables of primitive data types such as integer, character, float and double are used to store the actual data.]



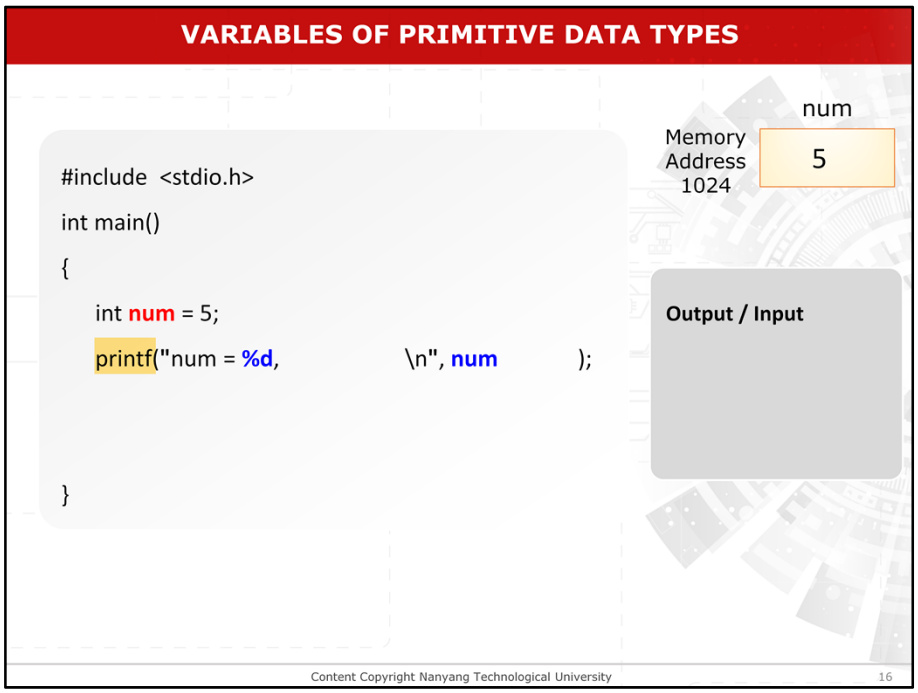
In this example, the variable num is declared as an integer.



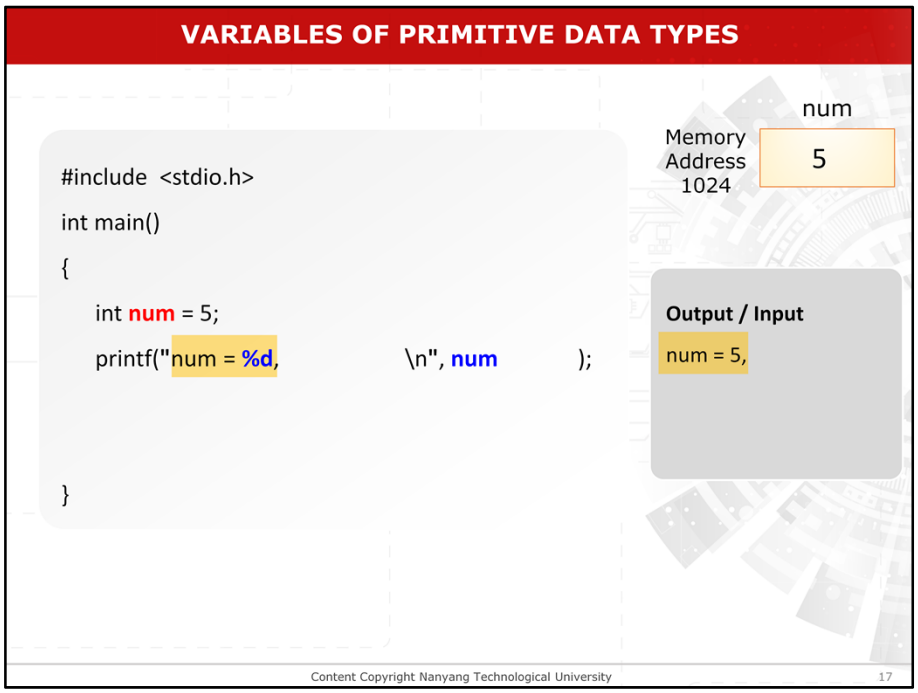
When variable num is initialized with the value of 5,



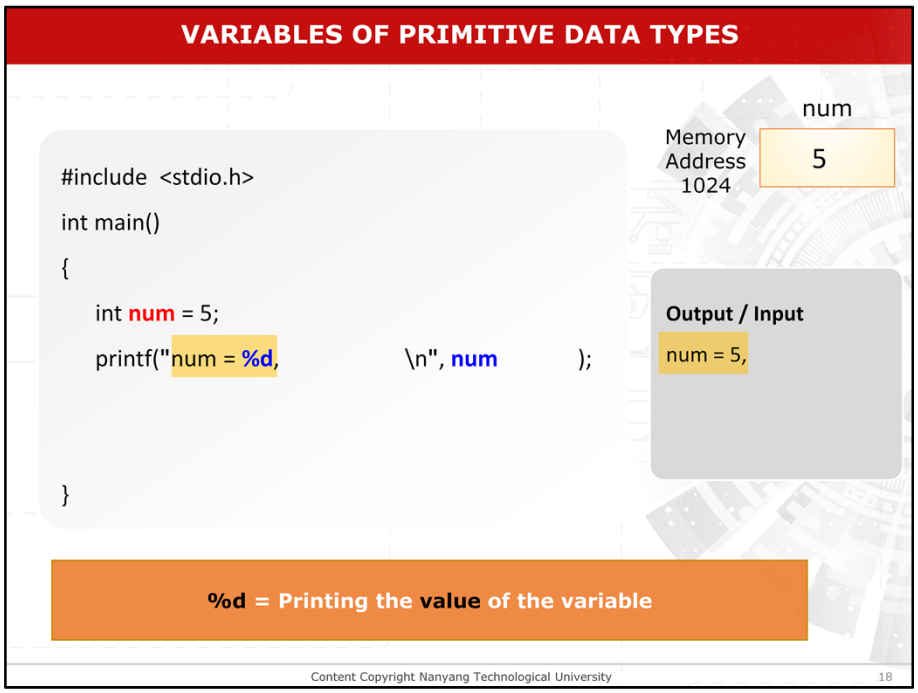
the memory location of the variable is used to store the actual value of 5.



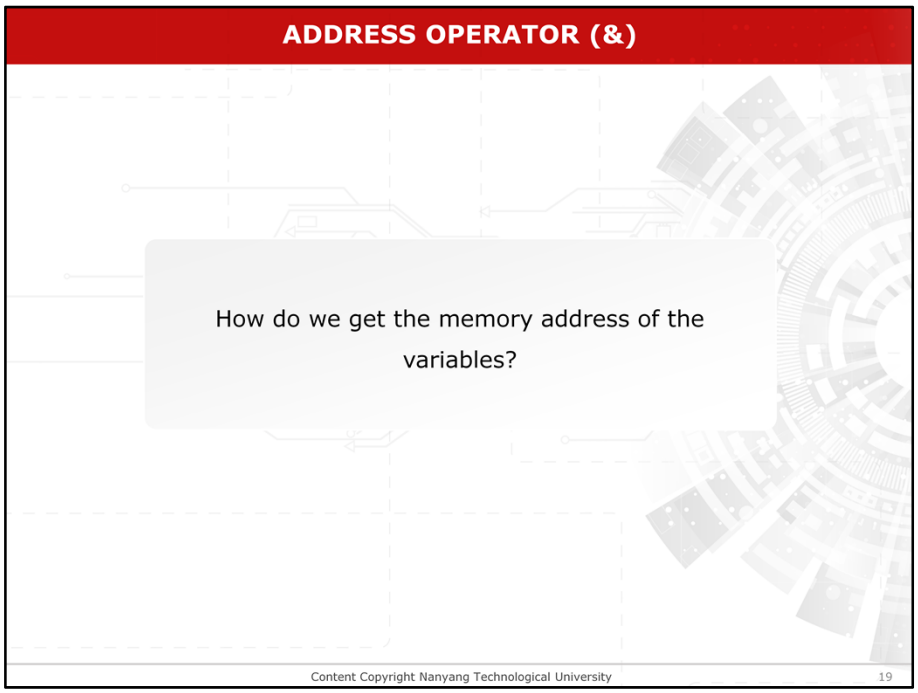
When the variable num is printed with the printf() statement,



the value 5 will then be printed on the output screen.



%d is used to print the value of the



How do we get the memory address of the variables?

We use Address Operators.

ADDRESS OPERATOR (&)

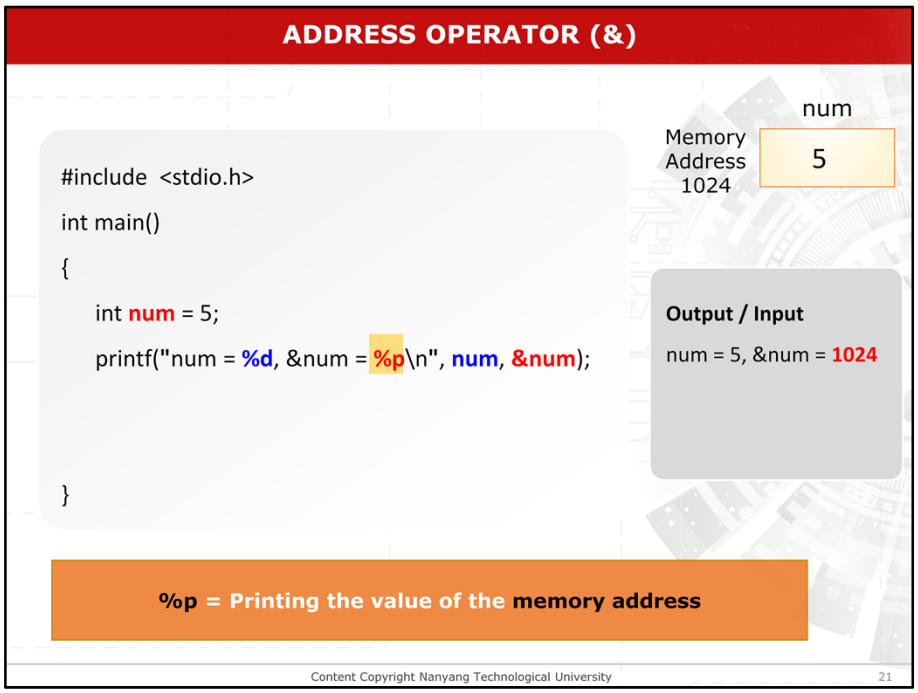

```
#include  <stdio.h>
int main()
{
    int num = 5;
    printf("num = %d, &num = %p\n", num, &num);

}
```

Output / Input

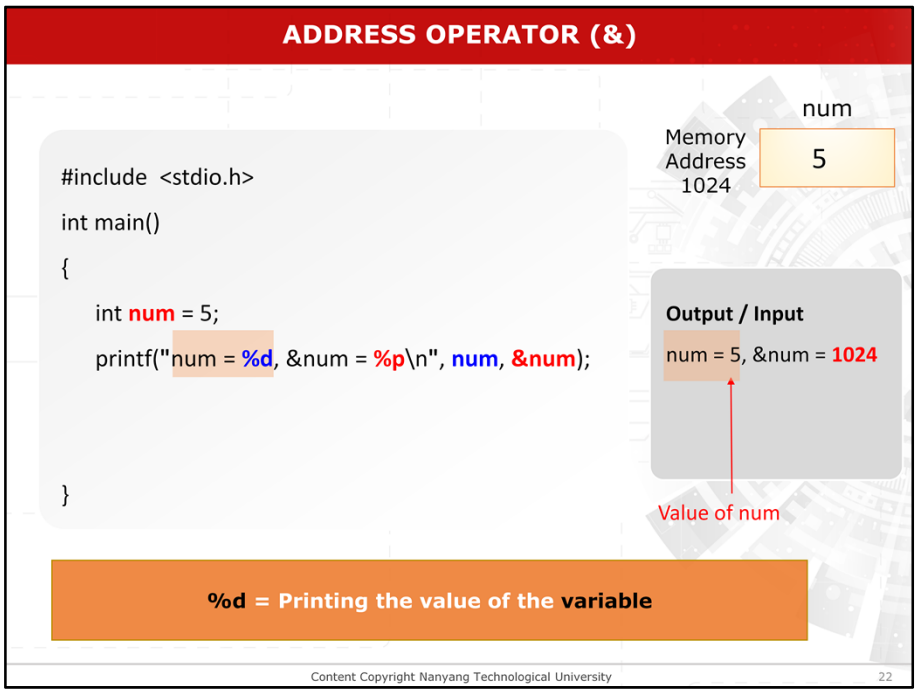
num = 5, &num =

The **address** of a variable can be obtained by the address operator (**&**).

The address of a variable can be obtained by the address operator (&).

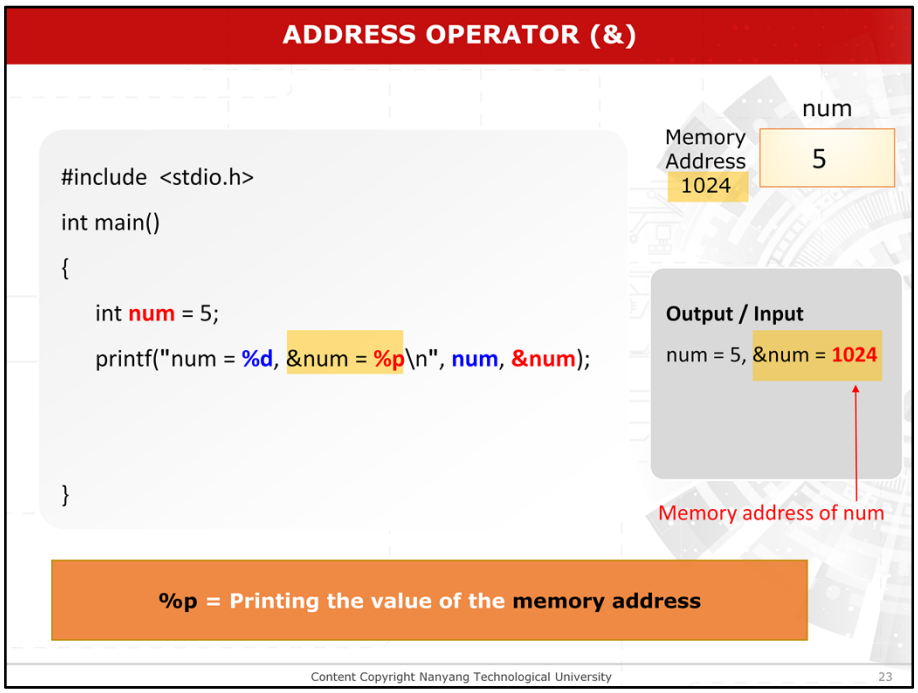


In the program, we can print the address of the variable num. To do this, we need to use %p in the conversion specifier in the control string of the printf() statement.

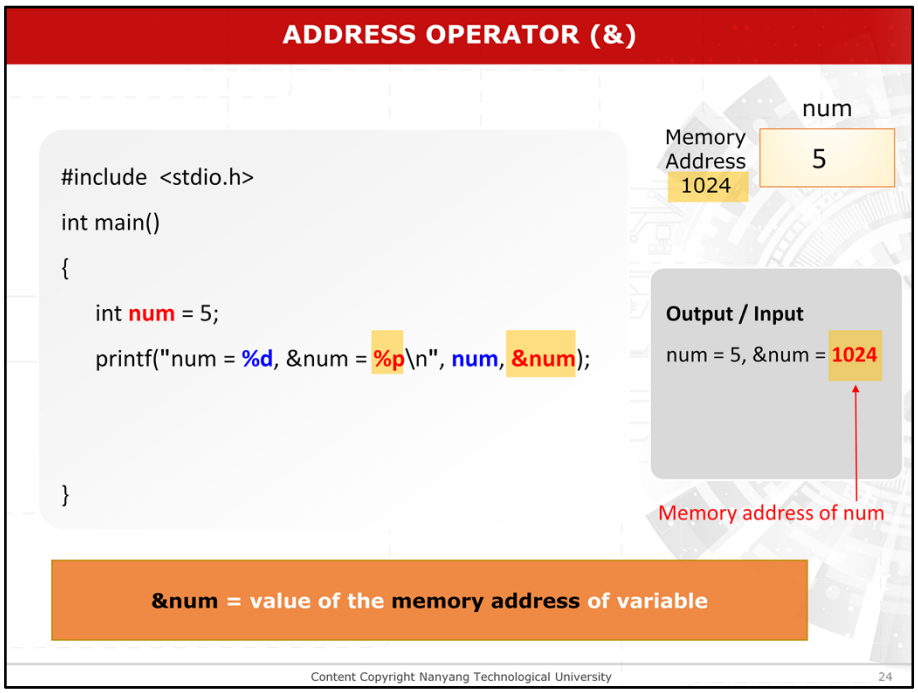


In the printf() statement, it prints two values on the screen.

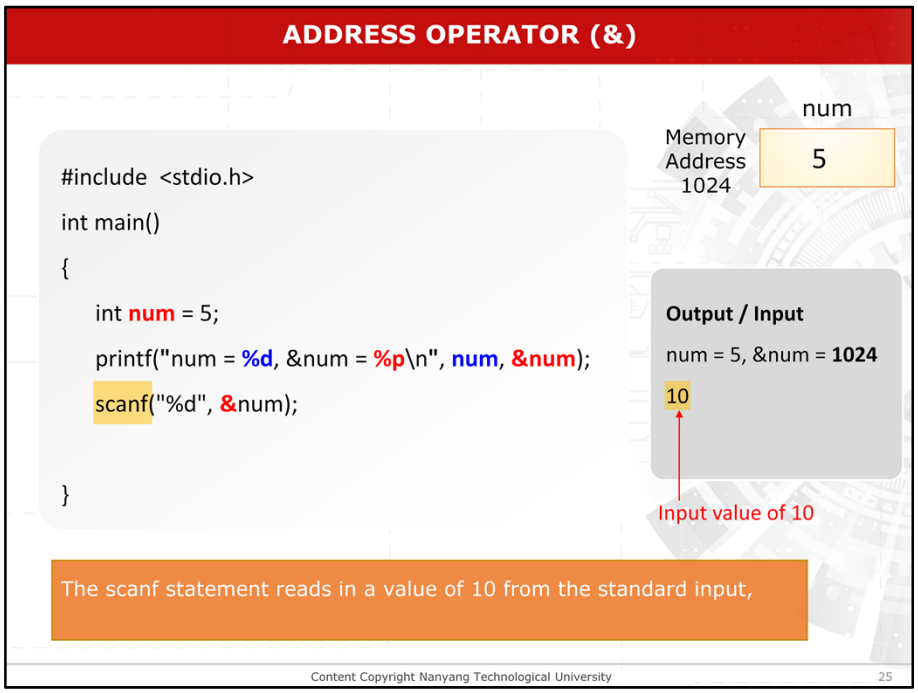
The first one uses %d to print the value 5 that is the initialized value stored at the memory location of the variable num.



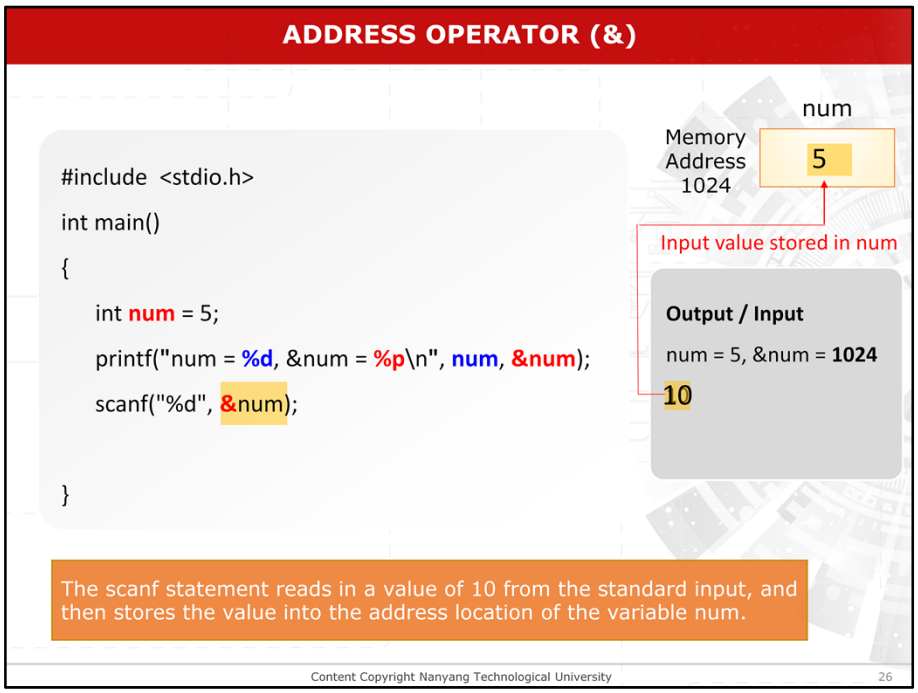
The other uses %p to print the memory address at which the value 5 is stored. In this example, the address of this memory location is 1024.



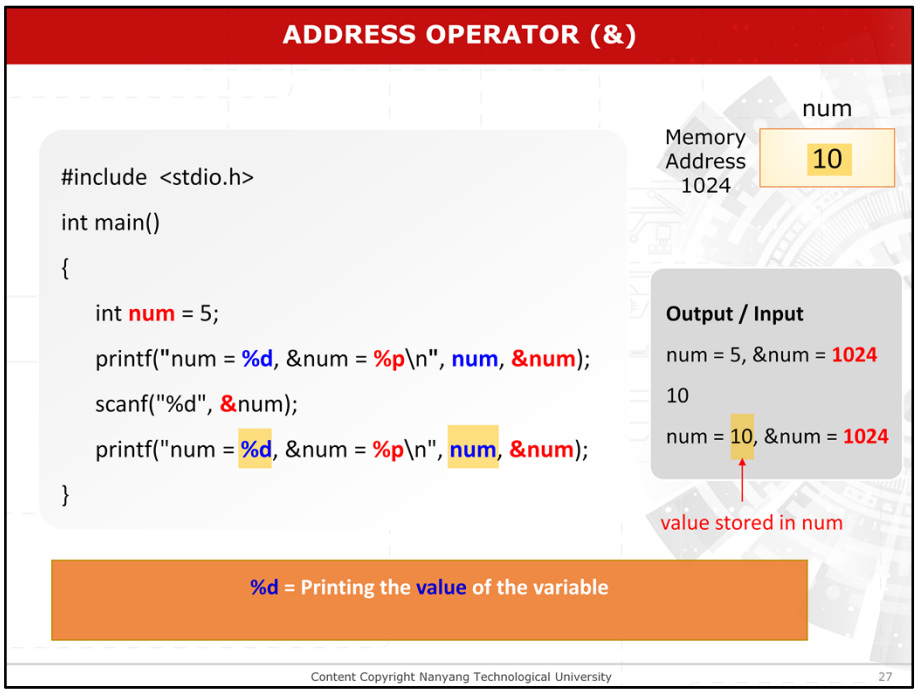
However, as the memory location is assigned by the computer, it may be different every time the same program is run. We use &num to find the value of the address.



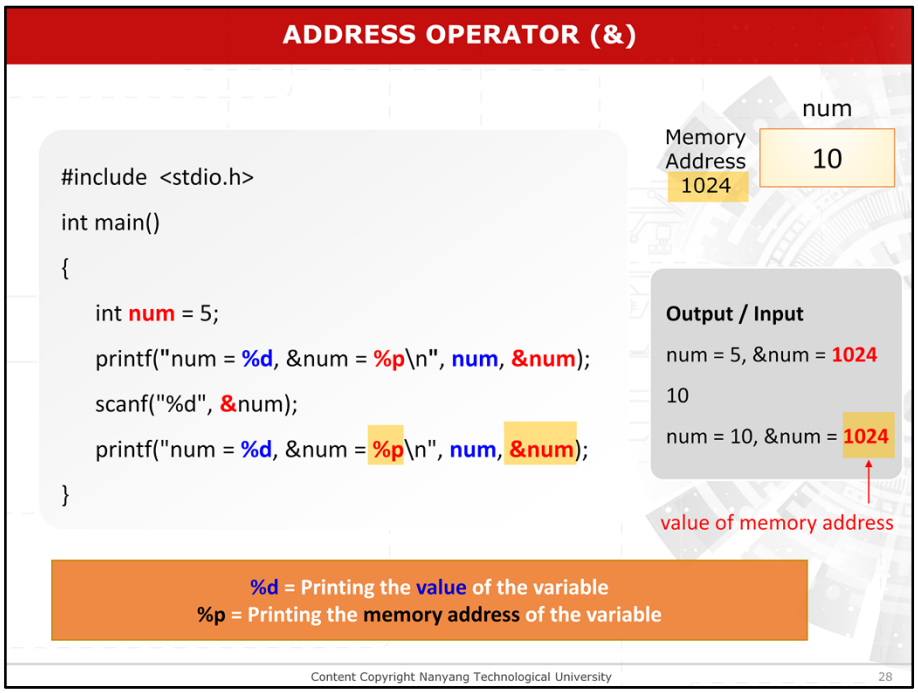
The scanf statement reads in a value of 10 from the standard input,



and then stores the value into the address location of the variable num.

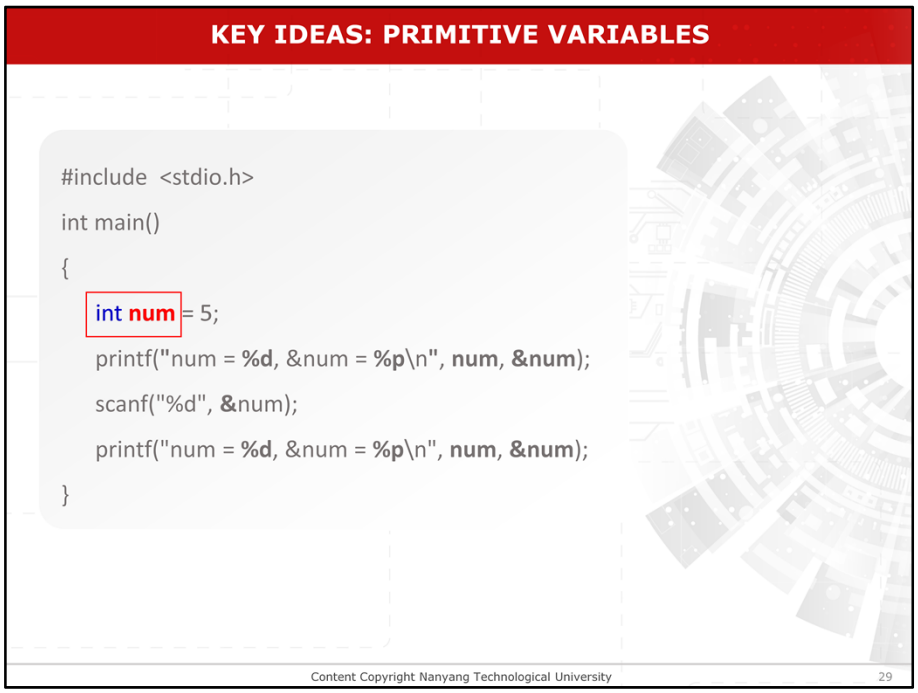


When we perform the printf() statement again, the value stored in variable num has been updated to 10 due to user input.



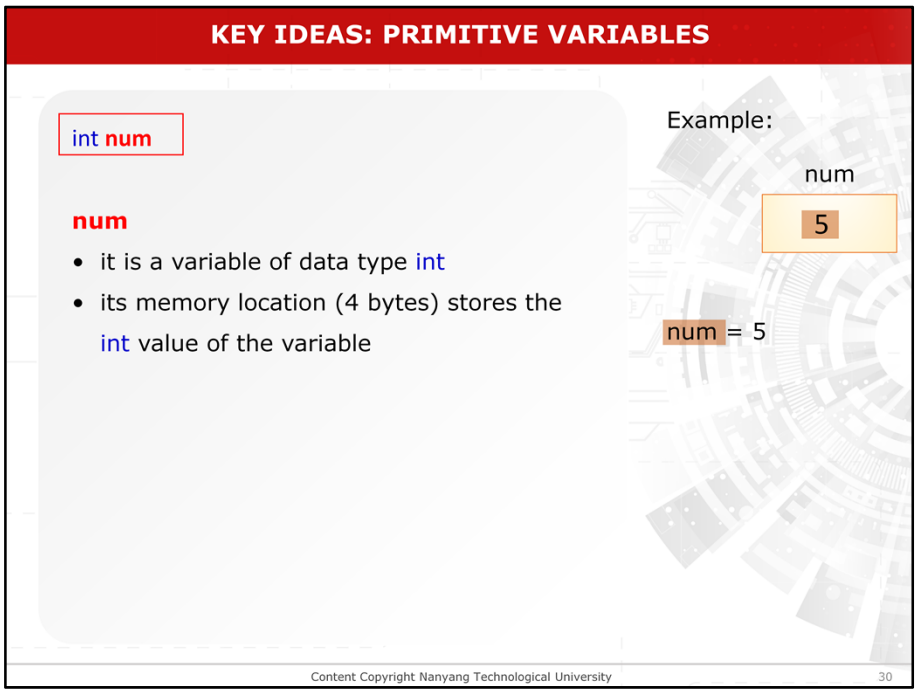
However, the memory address of the variable num remains the same throughout the execution of the program.

[Longer pause to transit to another chapter]

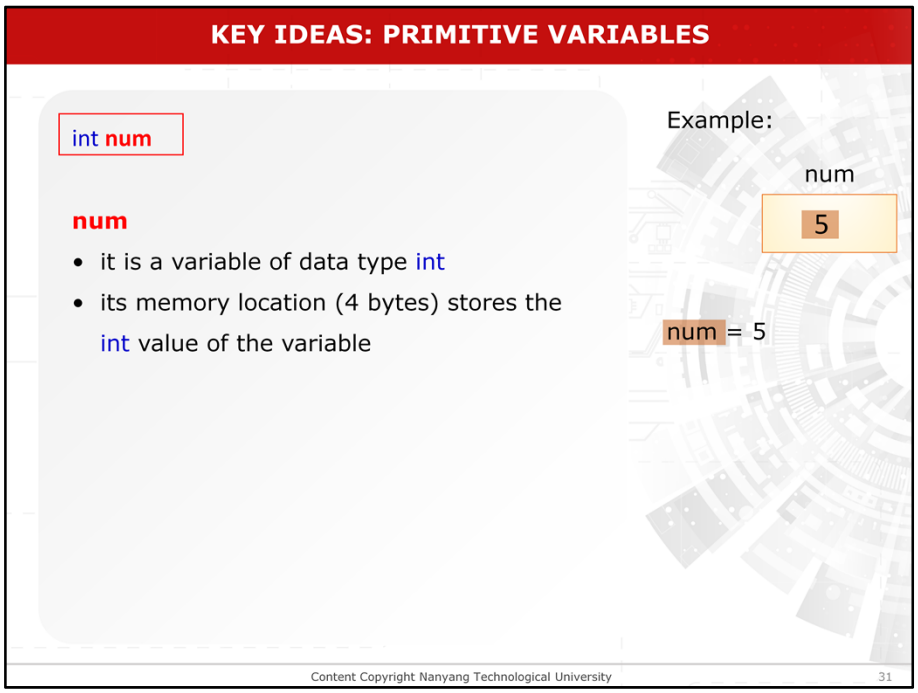


Key ideas of primitive variables [pause 2 sec]

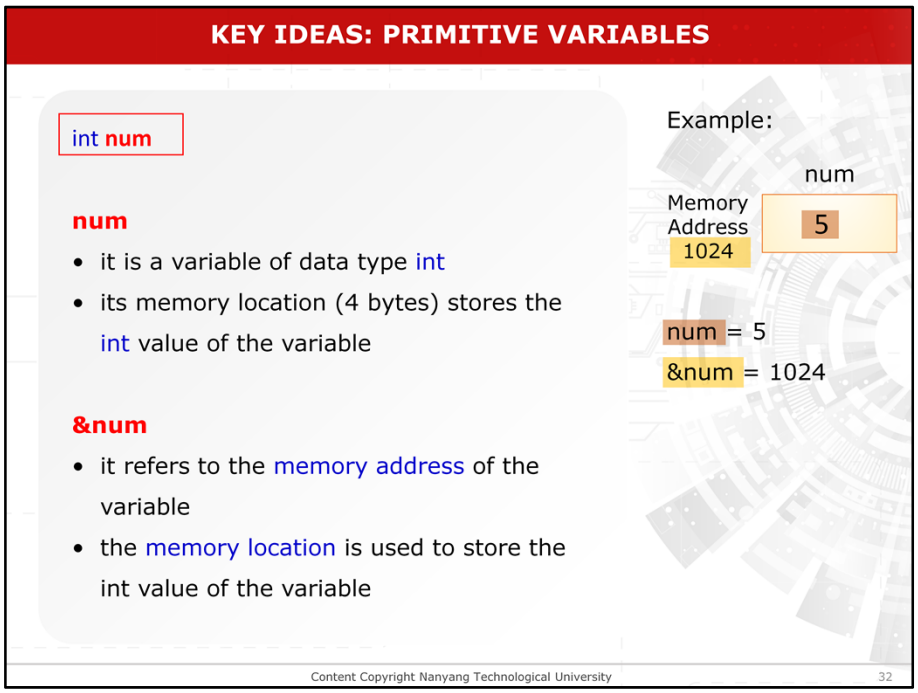
There are two key ideas related to variables that you must know.



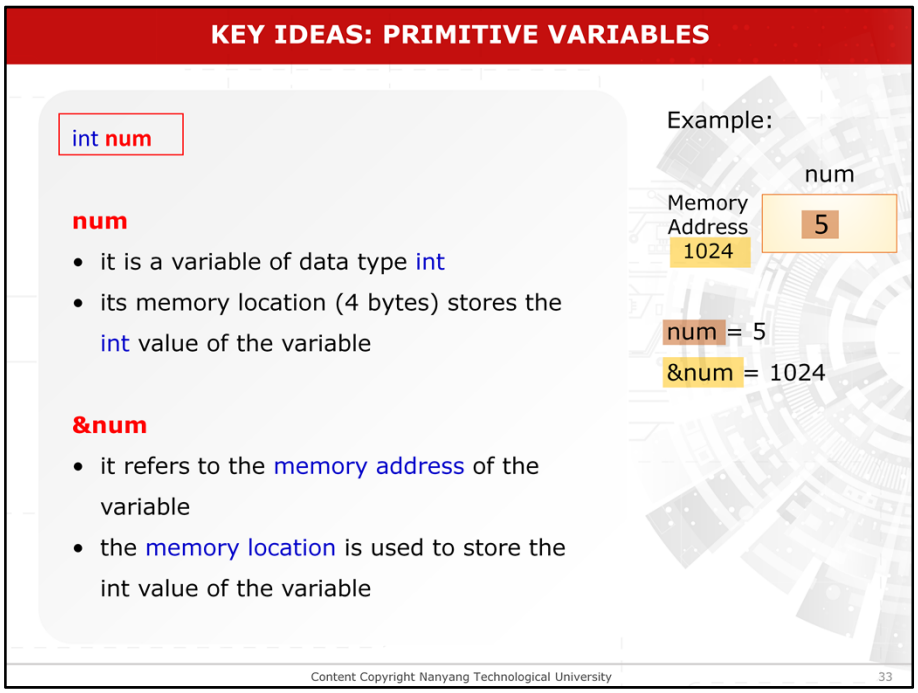
Primitive variable num stores a variable value of data type integer



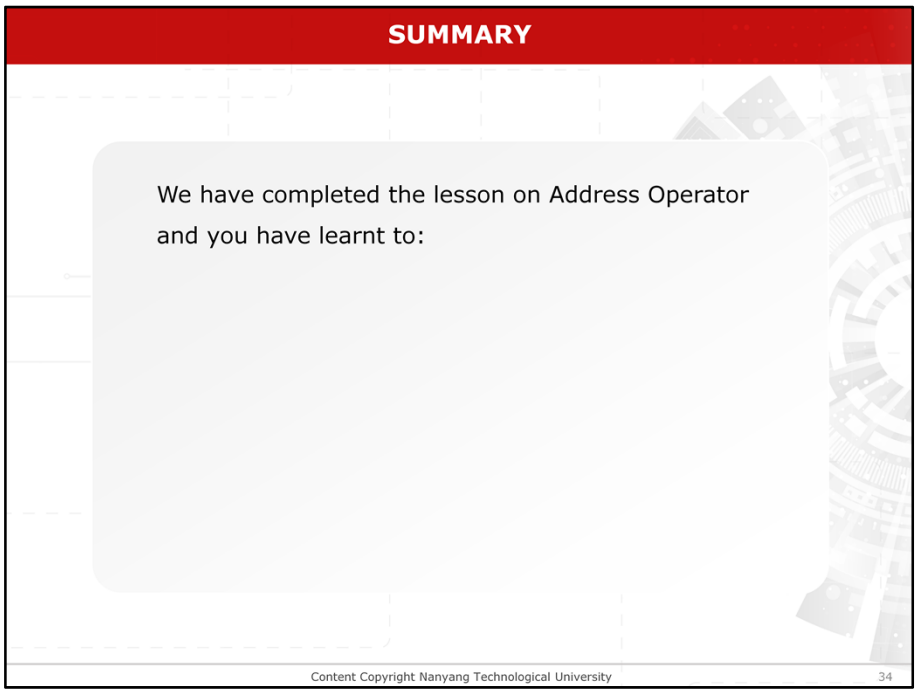
its memory location (4 bytes) stores the integer value of the variable



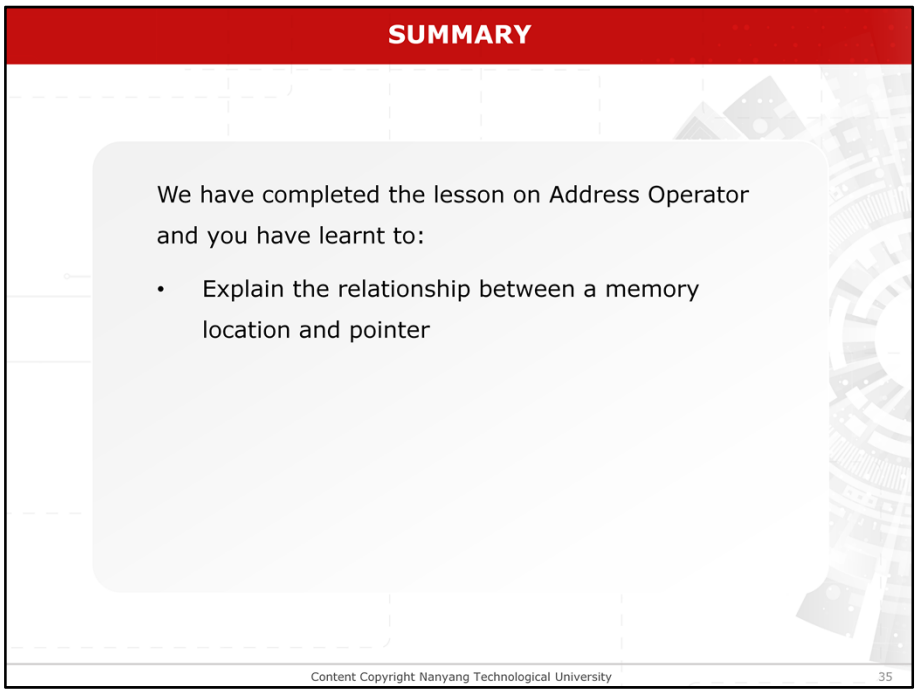
Address of Variable num (&num) refers to the memory address of the variable.



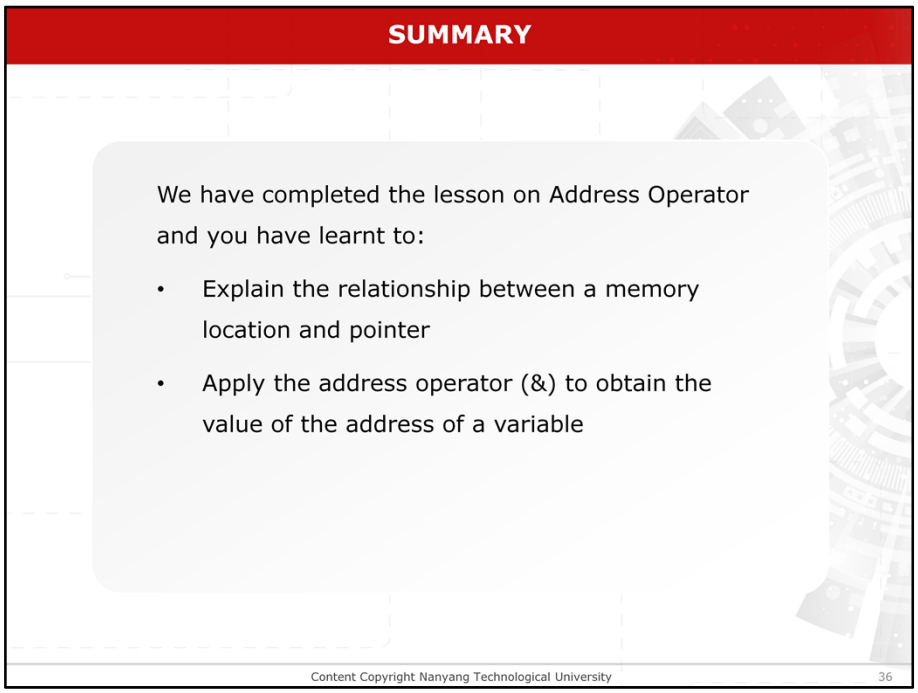
The memory location is used to store the variable value.



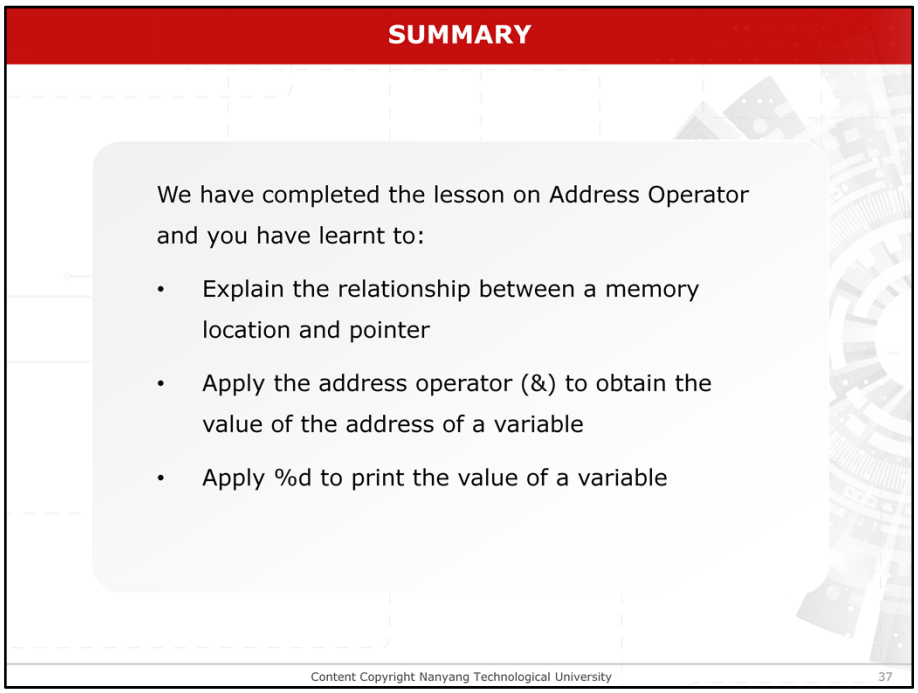
We have completed the lesson on Address Operator and you have learnt to:



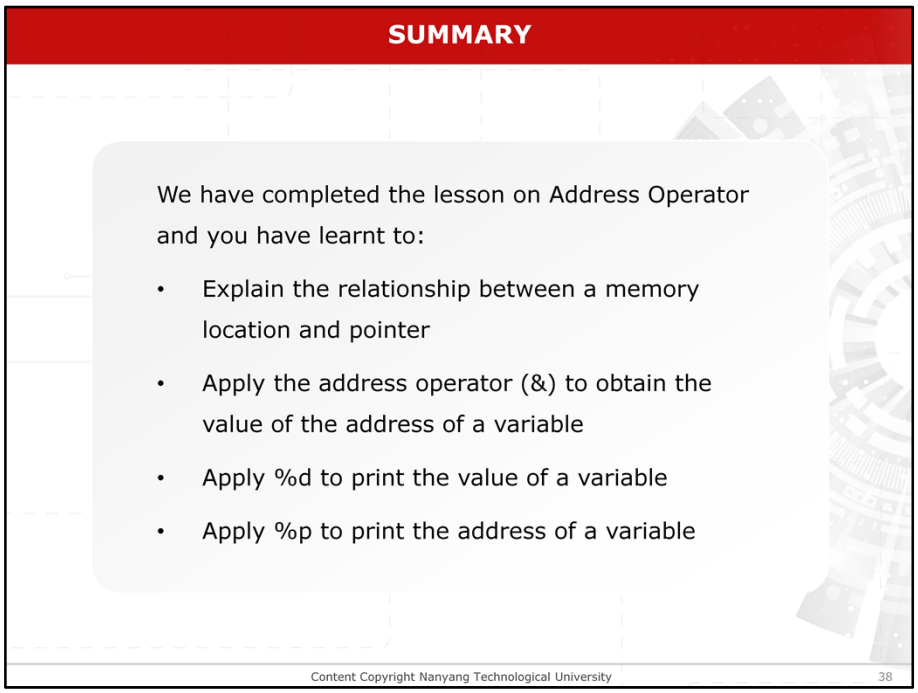
Explain the relationship between a memory location and pointers



Apply the address operator (&) to obtain the value of the address of a variable



Apply %d to print the value of a variable



Apply %p to print the address of a variable