

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.

OVERVIEW

The main aspects of Pointers are as follows:

Content Copyright Nanyang Technological University

2

We will be covering the following main aspects of pointers.

OVERVIEW

The main aspects of Pointers are as follows:

- Address Operator

Content Copyright Nanyang Technological University

3

Address Operator

OVERVIEW

The main aspects of Pointers are as follows:

- Address Operator
- Pointer Variables

Content Copyright Nanyang Technological University

4

Pointer Variables

OVERVIEW

The main aspects of Pointers are as follows:

- Address Operator
- Pointer Variables
- Call by Reference

Content Copyright Nanyang Technological University

5

Call by Reference.

OVERVIEW

The main aspects of Pointers are as follows:

- Address Operator
- Pointer Variables
- Call by Reference

Content Copyright Nanyang Technological University

6

In this lesson, we will look into Address Operator

LEARNING OBJECTIVES

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

Content Copyright Nanyang Technological University

7

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

LEARNING OBJECTIVES

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

- Explain the relationship between a memory location and pointer

Explain the relationship between a memory location and pointers

LEARNING OBJECTIVES

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

- Explain the relationship between a memory location and pointer
- Apply the address operator (&) to obtain the value of the address of a variable

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

LEARNING OBJECTIVES

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

- Explain the relationship between a memory location and pointer
- Apply the address operator (`&`) to obtain the value of the address of a variable
- Apply `%d` to print the value of a variable

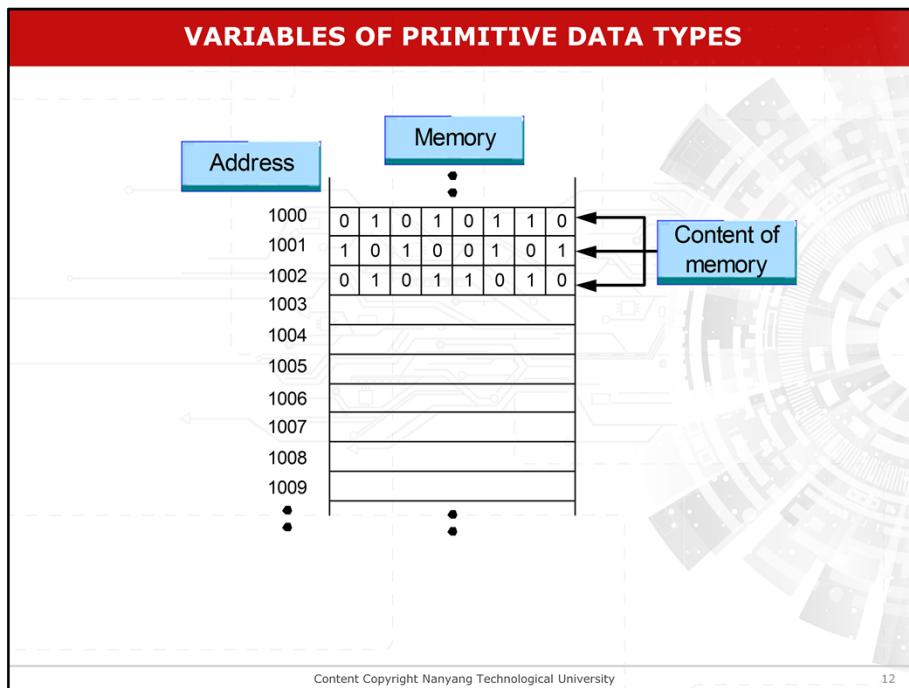
Apply `%d` to print the value of a variable

LEARNING OBJECTIVES

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

- Explain the relationship between a memory location and pointer
- 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

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.]

VARIABLES OF PRIMITIVE DATA TYPES

The diagram illustrates the declaration of a variable 'num' in C code. On the left, a code editor shows the following C code:

```
#include <stdio.h>
int main()
{
    int num = 5;
}
```

The variable 'num' is highlighted in yellow. On the right, a memory diagram shows a vertical stack of memory blocks. The top block is labeled 'num' and its 'Memory Address' is '1024'. Below it is a large gray box labeled 'Output / Input'. At the bottom of the slide, there is copyright information: 'Content Copyright Nanyang Technological University' and the number '13'.

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

VARIABLES OF PRIMITIVE DATA TYPES

```
#include <stdio.h>
int main()
{
    int num = 5;
}
```

Memory Address 1024

num
5

Output / Input

The variable num stores the value.

Content Copyright Nanyang Technological University

14

When variable num is initialized with the value of 5,

VARIABLES OF PRIMITIVE DATA TYPES

```
#include <stdio.h>
int main()
{
    int num = 5;
}
```

Memory Address
1024 num
 5

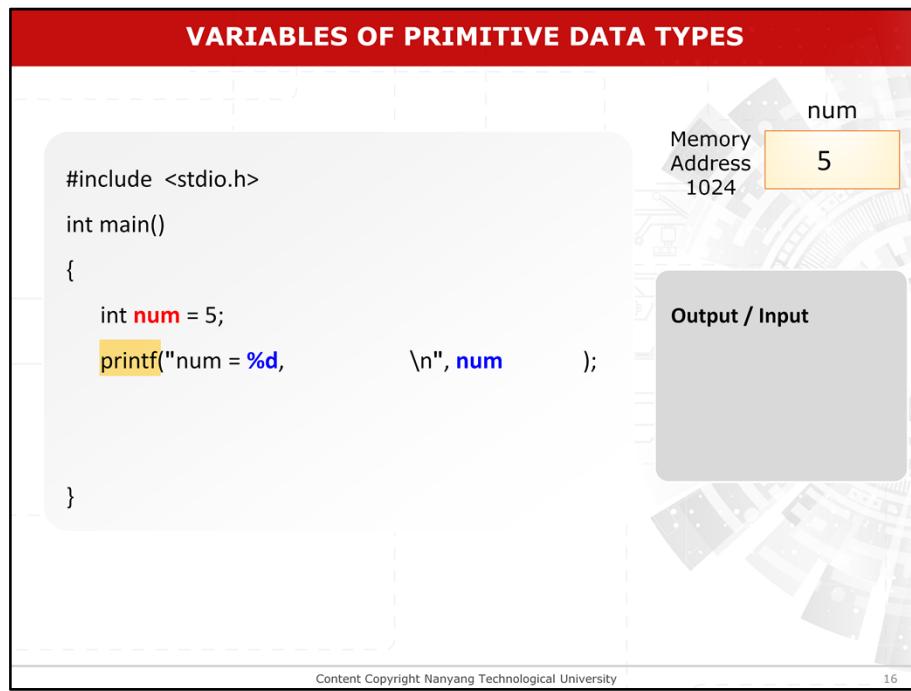
Output / Input

The variable num stores the value.

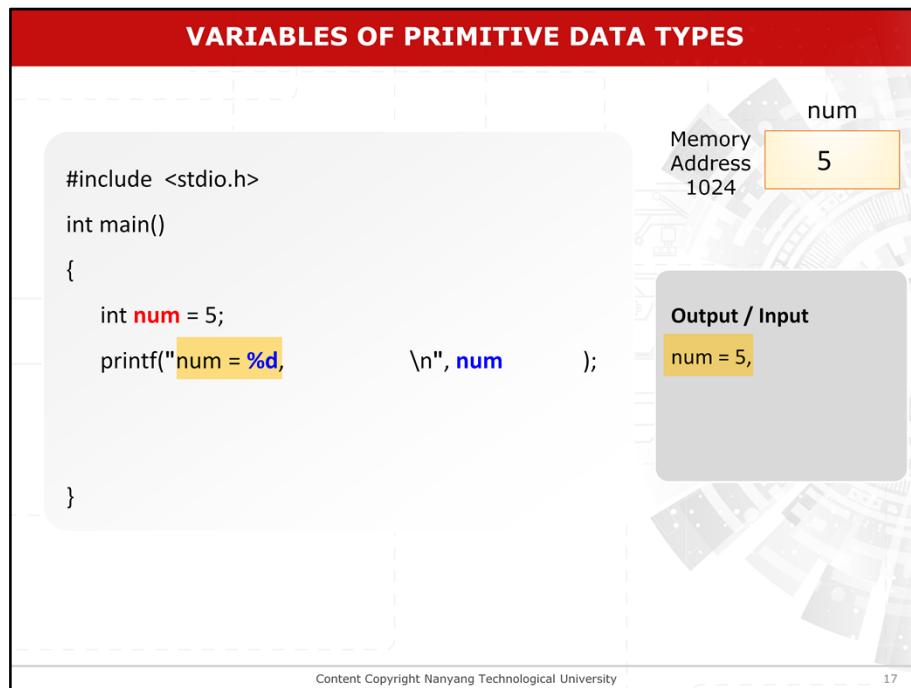
Content Copyright Nanyang Technological University

15

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.

VARIABLES OF PRIMITIVE DATA TYPES

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

Memory Address 1024

Output / Input

num = 5,

%d = Printing the value of the variable

Content Copyright Nanyang Technological University

18

%d is used to print the value of the

ADDRESS OPERATOR (&)

How do we get the memory address of the variables?

Content Copyright Nanyang Technological University

19

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);
}
```

Memory Address 1024 num

Output / Input num = 5, &num =

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

Content Copyright Nanyang Technological University 20

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

ADDRESS OPERATOR (&)

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

Memory Address 1024

Output / Input

Content Copyright Nanyang Technological University 21

%p = Printing the value of the memory address

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.

ADDRESS OPERATOR (&)

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

Memory Address 1024

num

5

Output / Input

num = 5, &num = 1024

Value of num

%d = Printing the value of the variable

Content Copyright Nanyang Technological University

22

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.

ADDRESS OPERATOR (&)

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

Memory Address 1024

num

Output / Input

num = 5, &num = 1024

Memory address of num

%p = Printing the value of the memory address

Content Copyright Nanyang Technological University 23

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.

ADDRESS OPERATOR (&)

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

Memory Address 1024

num

Output / Input

num = 5, &num = 1024

Memory address of num

&num = value of the memory address of variable

Content Copyright Nanyang Technological University

24

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.

ADDRESS OPERATOR (&)

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

}
```

Memory Address 1024

Output / Input

num = 5, &num = 1024

10

Input value of 10

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

Content Copyright Nanyang Technological University 25

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

ADDRESS OPERATOR (&)

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

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`.

Memory Address 1024

num

5

Input value stored in num

Output / Input

num = 5, &num = 1024

10

Content Copyright Nanyang Technological University

26

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

ADDRESS OPERATOR (&)

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

Memory Address 1024

Output / Input

num = 5, &num = 1024
10
num = 10, &num = 1024

value stored in num

%d = Printing the value of the variable

Content Copyright Nanyang Technological University 27

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

ADDRESS OPERATOR (&)

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

Memory Address 1024 num
1025

Output / Input

num = 5, &num = 1024
10
num = 10, &num = 1024

value of memory address

**%d = Printing the value of the variable
%p = Printing the memory address of the variable**

Content Copyright Nanyang Technological University

28

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: PRIMITIVE VARIABLES

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

Content Copyright Nanyang Technological University

29

Key ideas of primitive variables [pause 2 sec]

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

KEY IDEAS: PRIMITIVE VARIABLES

`int num`

num

- it is a variable of data type `int`
- its memory location (4 bytes) stores the `int` value of the variable

Example:

num

5

num = 5

Content Copyright Nanyang Technological University

30

Primitive variable num stores a variable value of data type integer

KEY IDEAS: PRIMITIVE VARIABLES

```
int num
```

num

- it is a variable of data type **int**
- its memory location (4 bytes) stores the **int** value of the variable

Example:

num

5

num = 5

Content Copyright Nanyang Technological University

31

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

KEY IDEAS: PRIMITIVE VARIABLES

`int num`

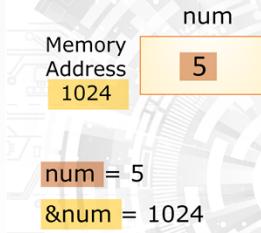
num

- it is a variable of data type `int`
- its memory location (4 bytes) stores the `int` value of the variable

&num

- it refers to the `memory address` of the variable
- the `memory location` is used to store the `int` value of the variable

Example:



Content Copyright Nanyang Technological University

32

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

KEY IDEAS: PRIMITIVE VARIABLES

`int num`

num

- it is a variable of data type `int`
- its memory location (4 bytes) stores the `int` value of the variable

&num

- it refers to the `memory address` of the variable
- the `memory location` is used to store the `int` value of the variable

Example:

Memory Address 1024 num 5

`num = 5`

`&num = 1024`

Content Copyright Nanyang Technological University

33

The memory location is used to store the variable value.

SUMMARY

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

Content Copyright Nanyang Technological University

34

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

SUMMARY

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

- Explain the relationship between a memory location and pointer

Explain the relationship between a memory location and pointers

SUMMARY

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

- Explain the relationship between a memory location and pointer
- Apply the address operator (&) to obtain the value of the address of a variable

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

SUMMARY

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

- Explain the relationship between a memory location and pointer
- Apply the address operator (`&`) to obtain the value of the address of a variable
- Apply `%d` to print the value of a variable

Apply `%d` to print the value of a variable

SUMMARY

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

- Explain the relationship between a memory location and pointer
- 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

Apply `%p` to print the address of a variable