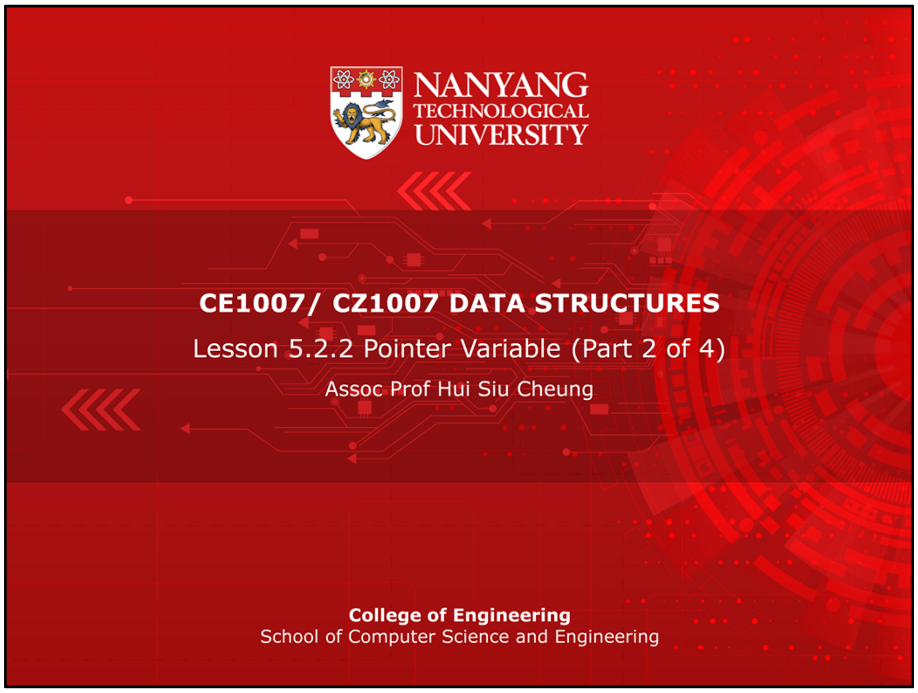


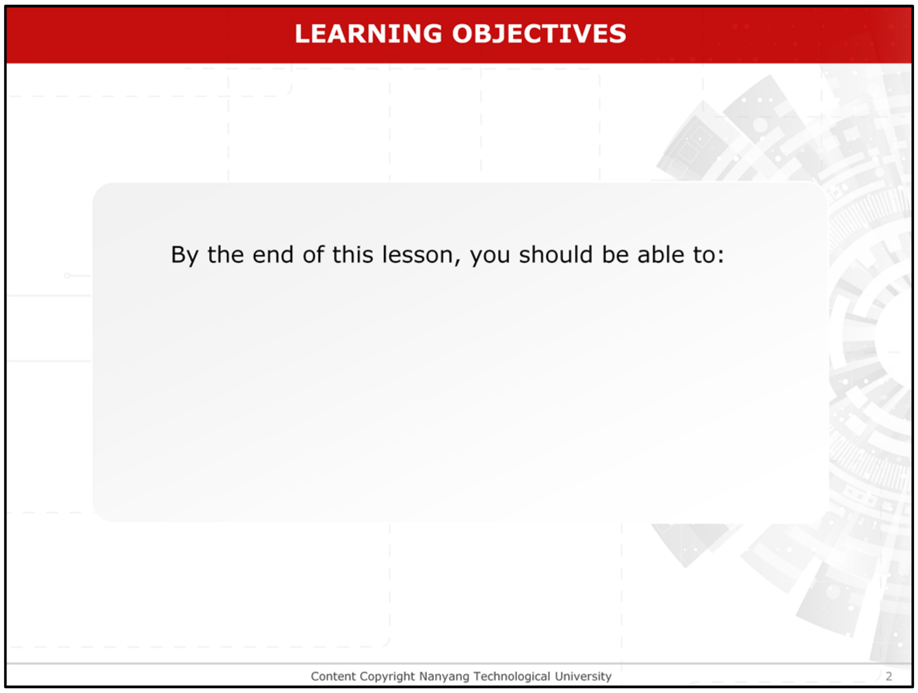
CE1007/ CZ1007 DATA STRUCTURES

Lesson 5.2.2 Pointer Variable (Part 2 of 4)

Assoc Prof Hui Siu Cheung

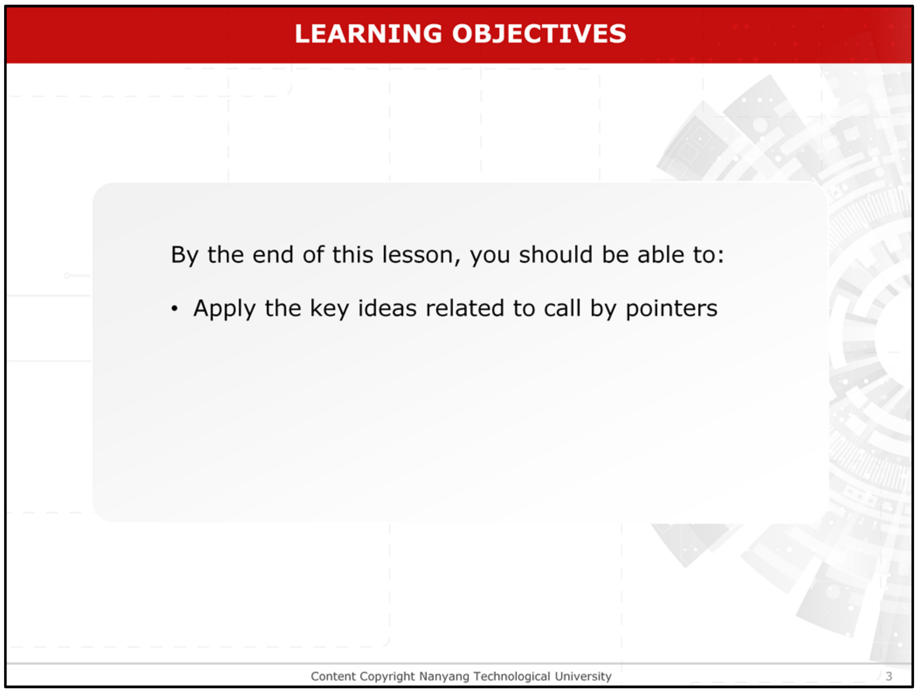
College of Engineering
School of Computer Science and Engineering

This lesson will show you how to apply what you have learnt in the previous lesson using a programming code example with detailed explanation.


LEARNING OBJECTIVES

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

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

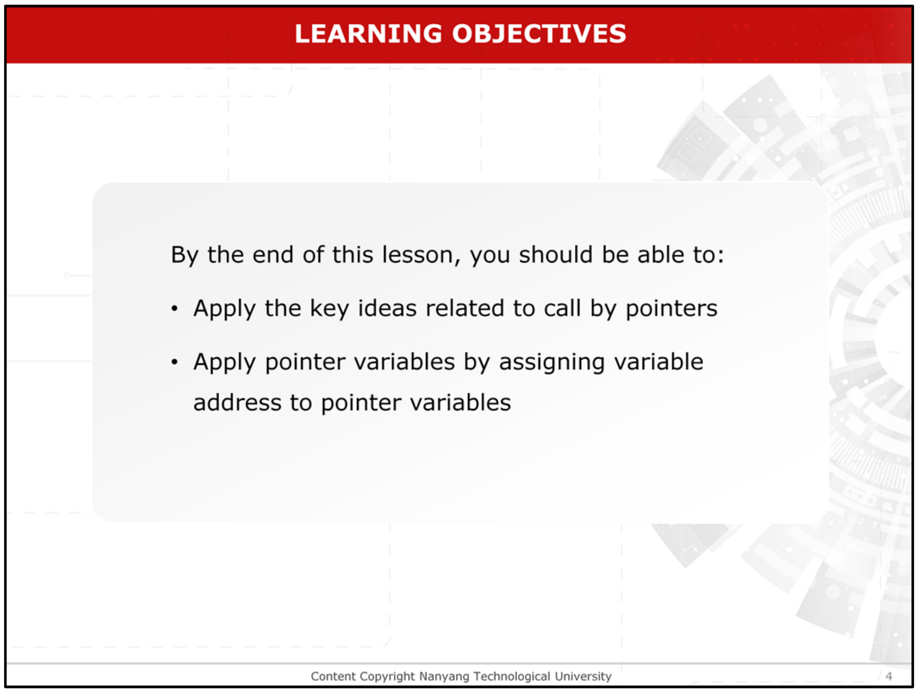


LEARNING OBJECTIVES

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

- Apply the key ideas related to call by pointers

Apply the key ideas related to call by pointers

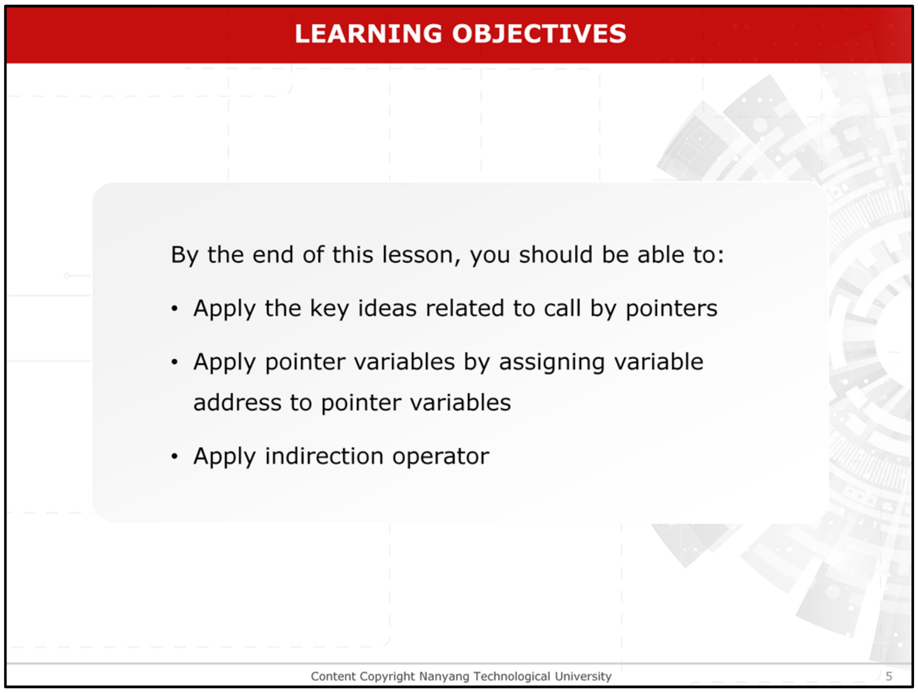


LEARNING OBJECTIVES

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

- Apply the key ideas related to call by pointers
- Apply pointer variables by assigning variable address to pointer variables

Apply pointer variables by assigning variable address to pointer variables



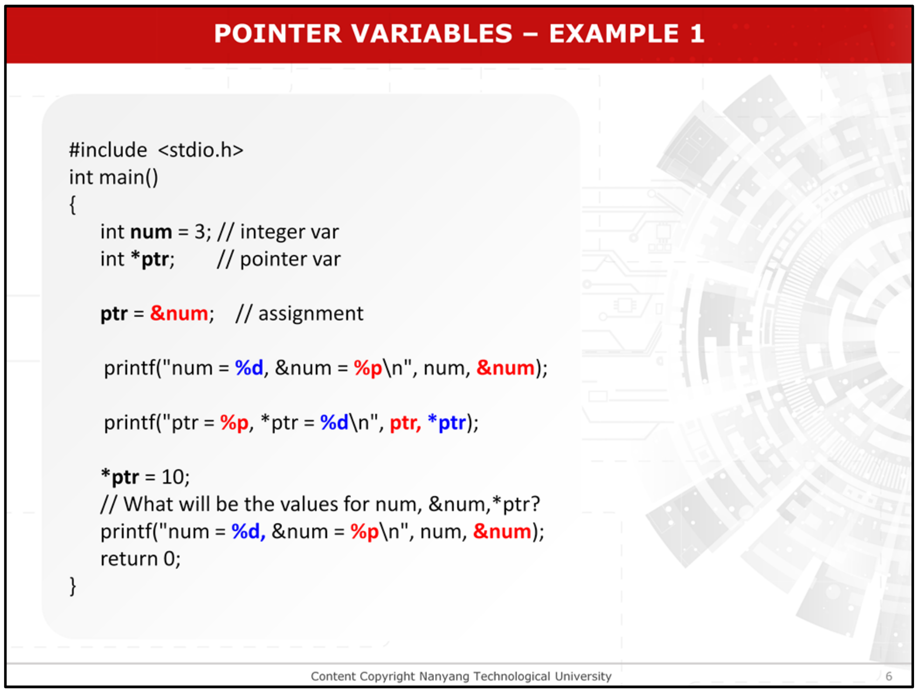
LEARNING OBJECTIVES

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

- Apply the key ideas related to call by pointers
- Apply pointer variables by assigning variable address to pointer variables
- Apply indirection operator

5

Apply indirection operator

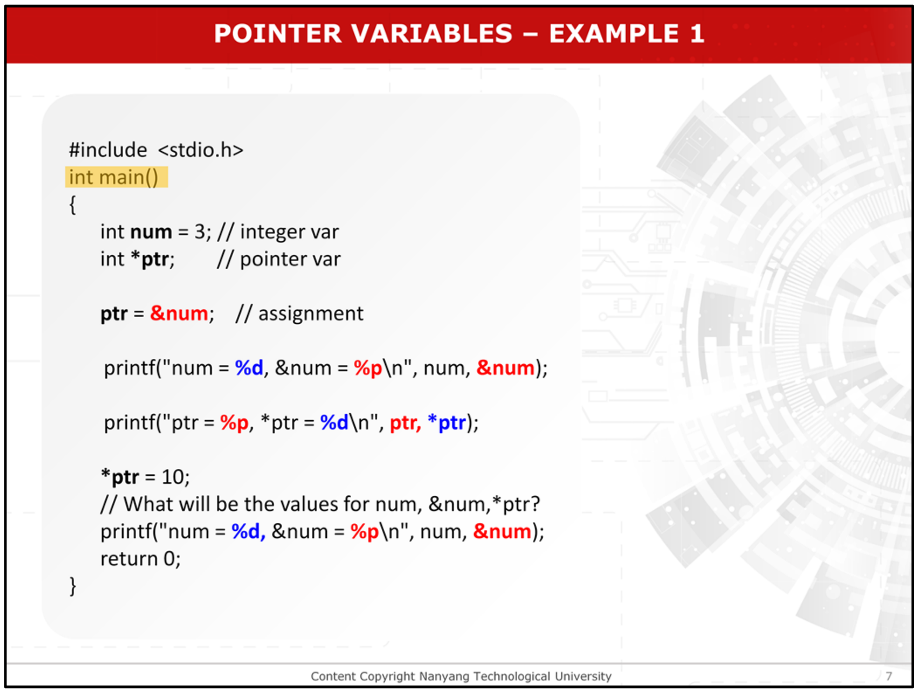


POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

      printf("num = %d, &num = %p\n", num, &num);

      printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

6

This is a complete C program that uses the concepts of pointers. Let’s go through the code in detail.

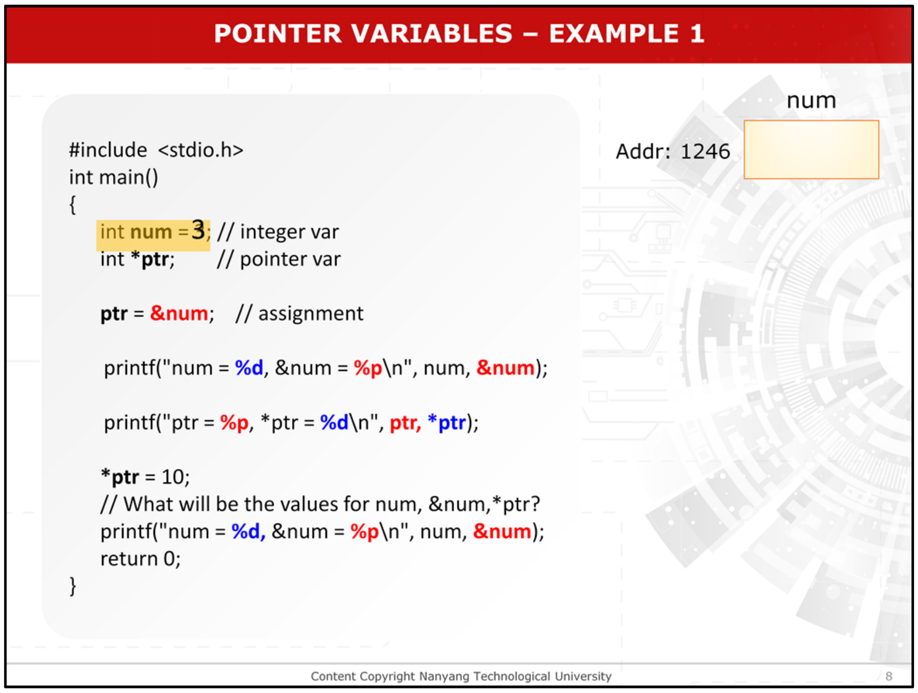


POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

      printf("num = %d, &num = %p\n", num, &num);

      printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

7

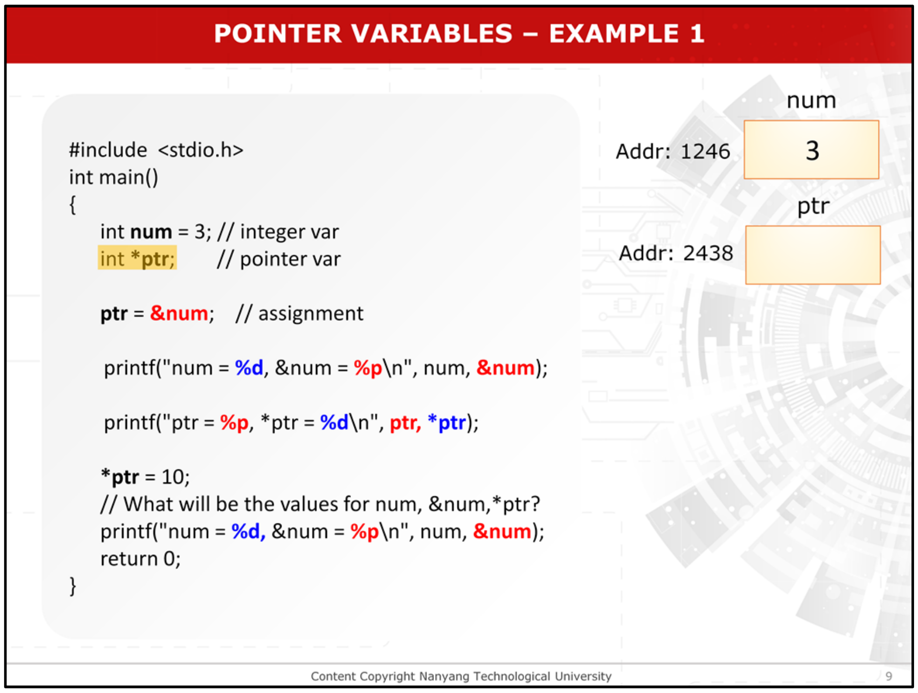
In this main program,



POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
    int num = 3; // integer var
    int *ptr;      // pointer var

    ptr = &num;    // assignment

     printf("num = %d, &num = %p\n", num, &num);

     printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

    *ptr = 10;
    // What will be the values for num, &num,*ptr?
    printf("num = %d, &num = %p\n", num, &num);
    return 0;
}
```

Variable num of integer data type is declared. Let's assume that the system allocated this variable num at memory address 1246. And the integer value of 3 is stored in this variable num.

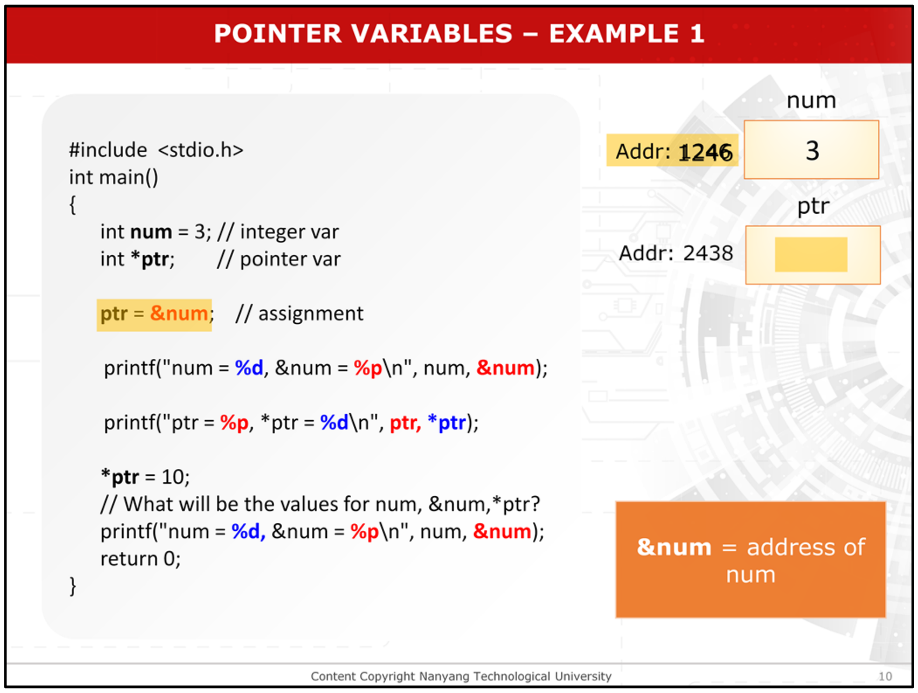


POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
    int num = 3; // integer var
    int *ptr;      // pointer var

    ptr = &num;    // assignment

     printf("num = %d, &num = %p\n", num, &num);

     printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

    *ptr = 10;
    // What will be the values for num, &num,*ptr?
    printf("num = %d, &num = %p\n", num, &num);
    return 0;
}
```

Next, this statement integer asterisk pointer declares a pointer variable named pointer. It is to point to a memory address. This address that it points to is to store integer value.

Let's assume the system allocated memory address 2438 to this pointer variable. Note that no value has been assigned to this pointer variable yet.



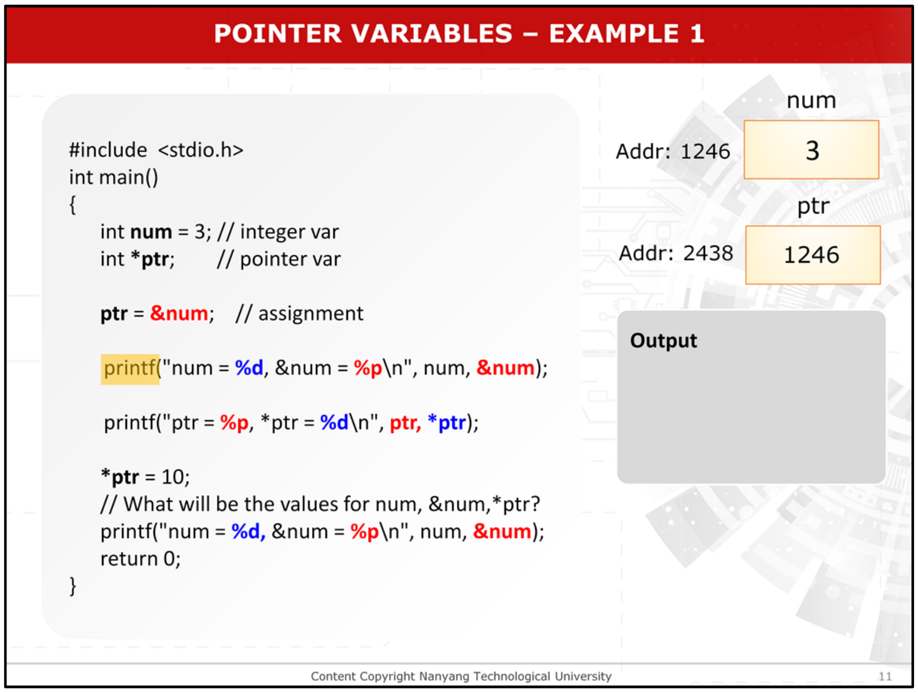
POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
    int num = 3; // integer var
    int *ptr;       // pointer var

    ptr = &num;    // assignment

     printf("num = %d, &num = %p\n", num, &num);

     printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

    *ptr = 10;
    // What will be the values for num, &num,*ptr?
    printf("num = %d, &num = %p\n", num, &num);
    return 0;
}
```

&num = address of num

10

The highlighted statement assigns a value to pointer. The ampersand num is the address of num, which is 1246 in this example. This address value is assigned to the pointer

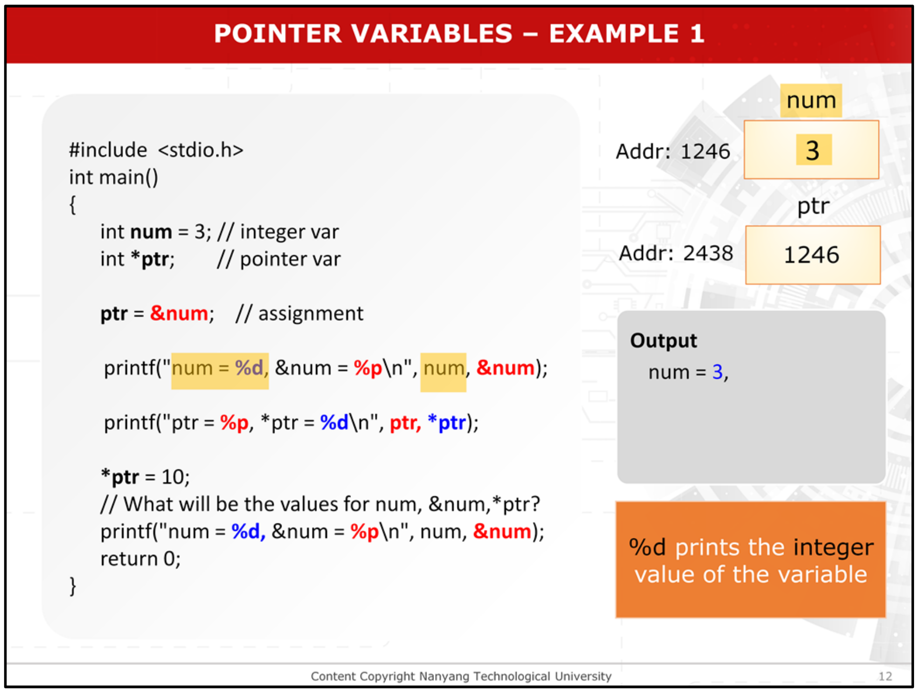


POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

     printf("num = %d, &num = %p\n", num, &num);

     printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

num
Addr: 1246
3
ptr
Addr: 2438
1246
Output

Next, the printf function will gives an output.



POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;      // pointer var

     ptr = &num;    // assignment

     printf("num = %d, &num = %p\n", num, &num);

     printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

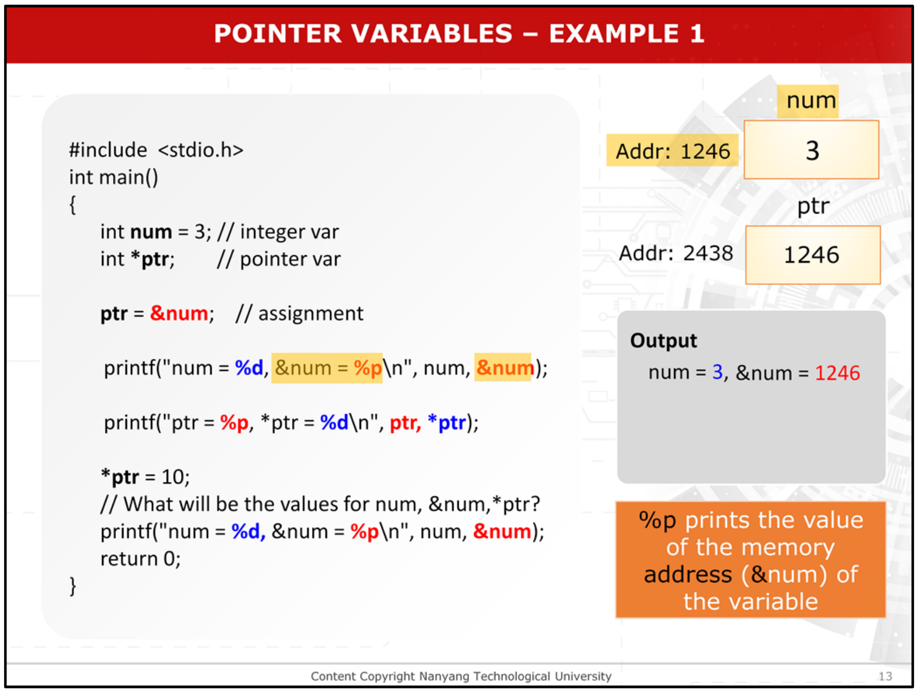
num
Addr: 1246
3
ptr
Addr: 2438
1246

Output

```
num = 3,
```

%d prints the integer value of the variable

Percent d prints the integer value of the variable. So the output screen will show num equals 3.



POINTER VARIABLES – EXAMPLE 1

```
#include <stdio.h>
int main()
{
    int num = 3; // integer var
    int *ptr;       // pointer var

    ptr = &num;    // assignment

    printf("num = %d, &num = %p\n", num, &num);

    printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

    *ptr = 10;
    // What will be the values for num, &num,*ptr?
    printf("num = %d, &num = %p\n", num, &num);
    return 0;
}
```

num
Addr: 1246
3
ptr
Addr: 2438
1246

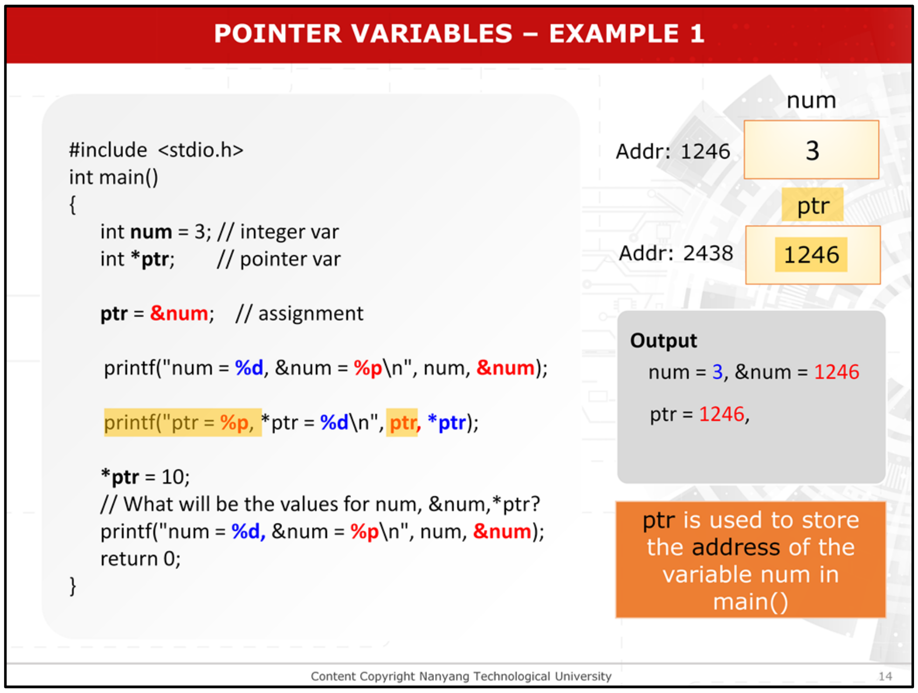
Output

```
num = 3, &num = 1246
```

%p prints the value of the memory address (&num) of the variable

13

Percent p prints the value of the memory address of num, represented by ampersand num. So the output will show ampersand num equals 1246.



POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

      printf("num = %d, &num = %p\n", num, &num);

      printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

num
Addr: 1246
3
ptr
Addr: 2438
1246

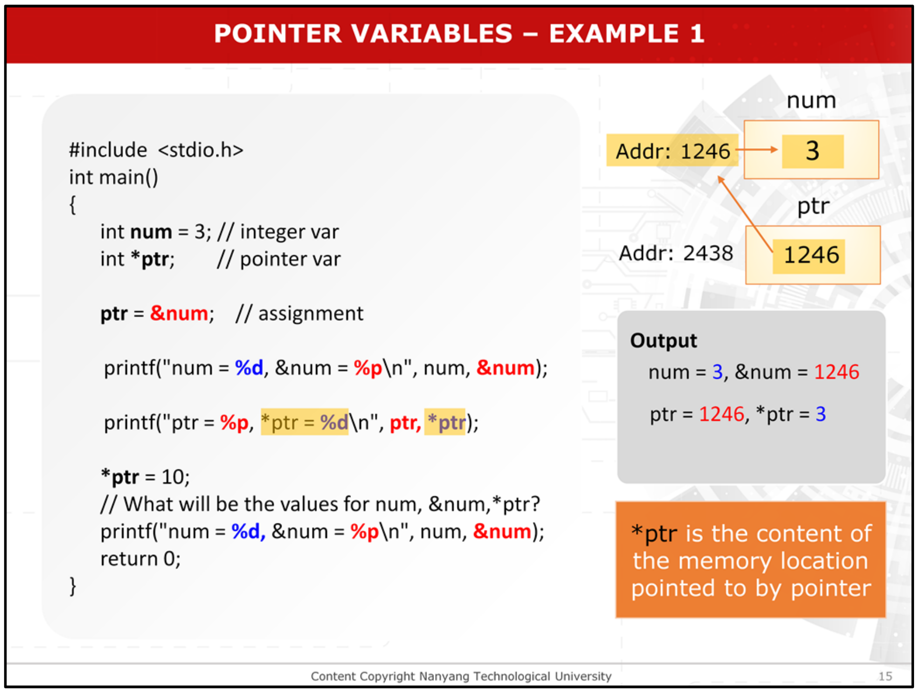
Output

```
num = 3, &num = 1246

ptr = 1246,
```

ptr is used to store the address of the variable num in main()

14

In this next print f function, percent p is used to print memory address. Since pointer is used to store the address of the variable num, there is no need to use the ampersand operator. Thus, the output will show pointer equals 1246 which is the address stored in pointer.



POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

      printf("num = %d, &num = %p\n", num, &num);

      printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

num
Addr: 1246
3
ptr
Addr: 2438
1246

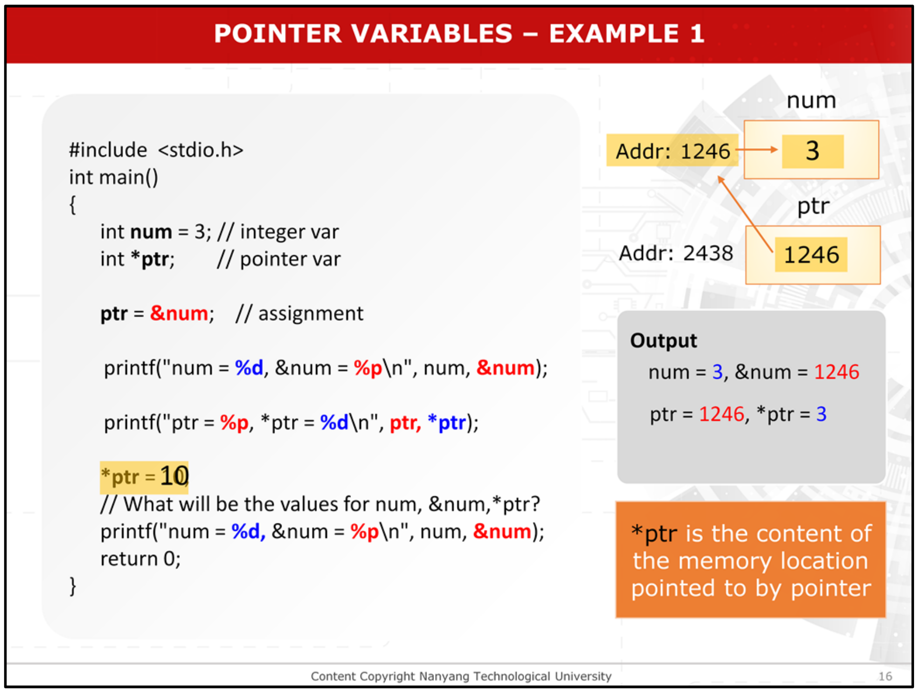
Output

```
num = 3, &num = 1246
ptr = 1246, *ptr = 3
```

*ptr is the content of the memory location pointed to by pointer

15

percent d is to print the integer value of asterisk pointer. Asterisk pointer is the content of the memory location pointed to by pointer. So the output will show asterisk pointer equals 3.



POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;      // pointer var

     ptr = &num;    // assignment

      printf("num = %d, &num = %p\n", num, &num);

      printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

num
Addr: 1246
3
ptr
Addr: 2438
1246

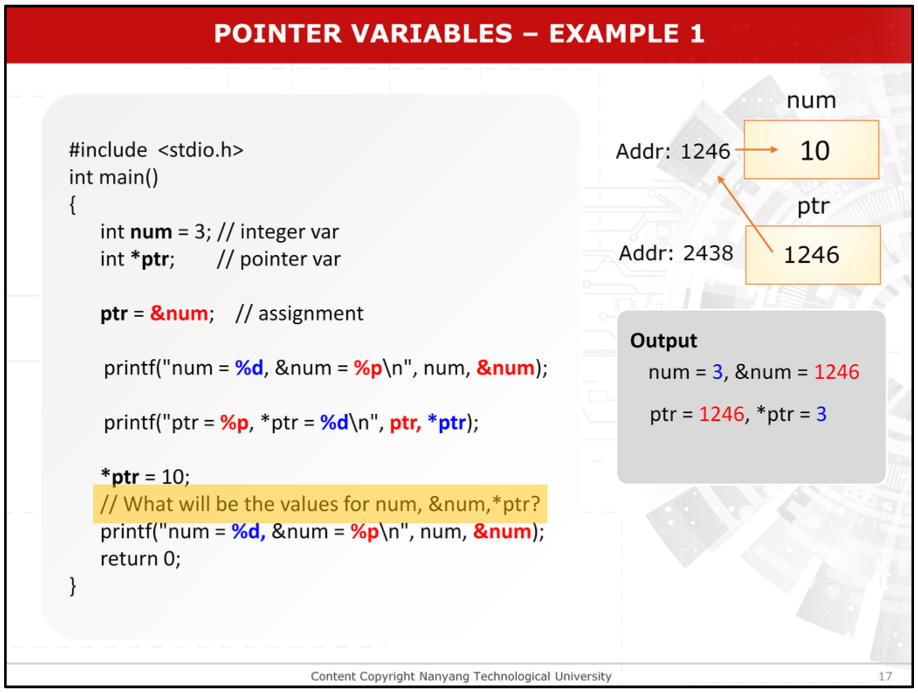
Output

```
num = 3, &num = 1246

ptr = 1246, *ptr = 3
```

*ptr is the content of the memory location pointed to by pointer

16

In this statement asterisk pointer equals 10, asterisk p which is the content of the memory location pointed to by pointer, will assign the new value of 10. So the value of 3 stored in variable num will change from 3 to 10.



POINTER VARIABLES – EXAMPLE 1

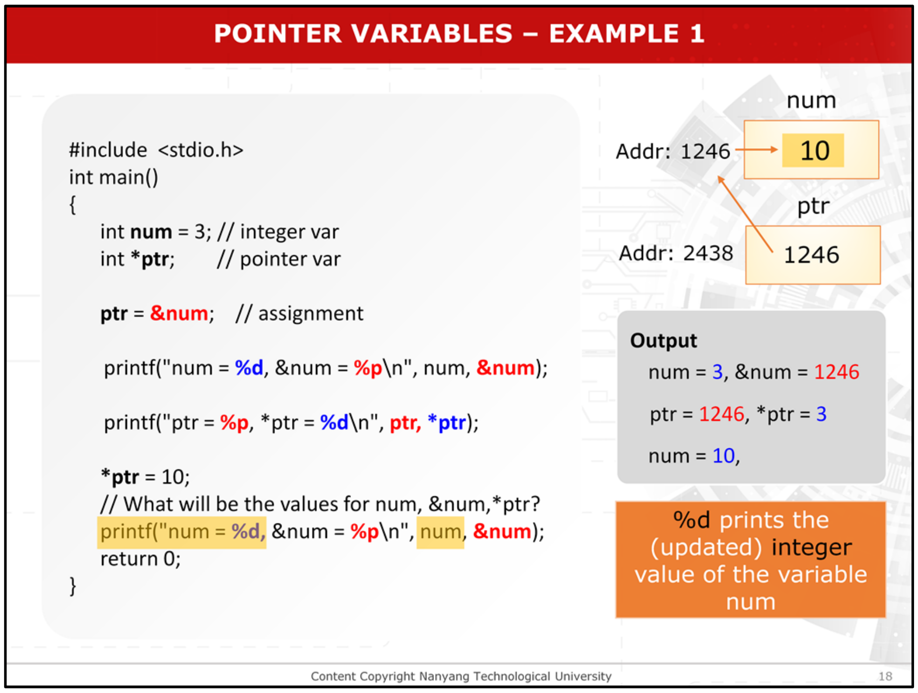
```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

      printf("num = %d, &num = %p\n", num, &num);

      printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

num
Addr: 1246
10
ptr
Addr: 2438
1246

Output

```
num = 3, &num = 1246

ptr = 1246, *ptr = 3
```

Before we move on to the next statement, what do you think will be the values for num, ampersand num and asterisk pointer respectively?



POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

      printf("num = %d, &num = %p\n", num, &num);

      printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

num
Addr: 1246
10
ptr
Addr: 2438
1246

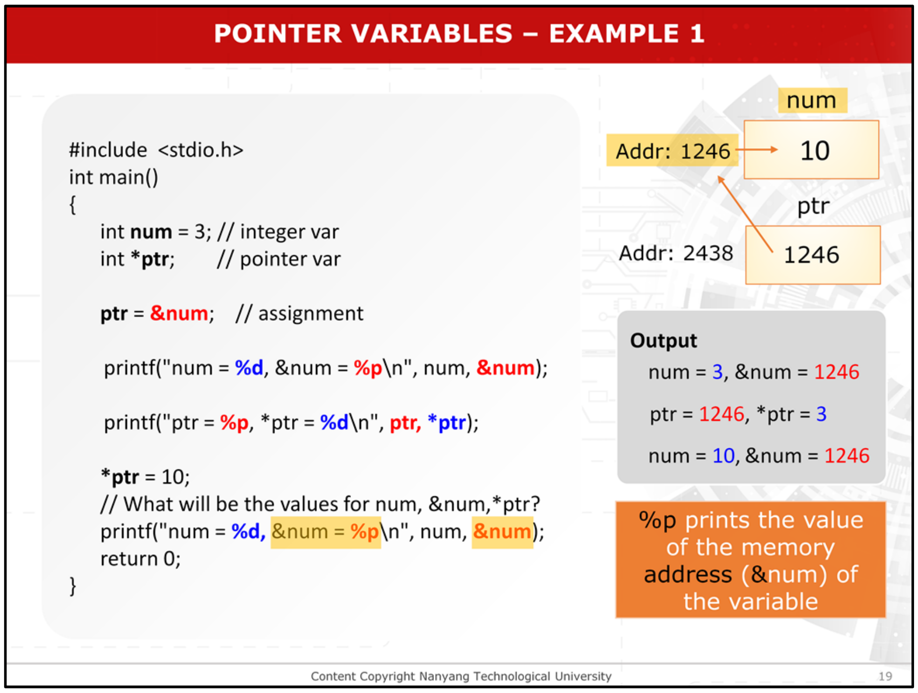
Output

```
num = 3, &num = 1246
ptr = 1246, *ptr = 3
num = 10,
```

%d prints the (updated) integer value of the variable num

18

Since **pointer** stores the address of **numb**, the change of value at this memory location has the same effect as changing the value stored at **numb**. Therefore, the value stored at **numb** is 10. Percent d   integer



POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
    int num = 3; // integer var
    int *ptr;       // pointer var

    ptr = &num;    // assignment

    printf("num = %d, &num = %p\n", num, &num);

    printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

    *ptr = 10;
    // What will be the values for num, &num,*ptr?
    printf("num = %d, &num = %p\n", num, &num);
    return 0;
}
```

num
Addr: 1246
10
ptr
Addr: 2438
1246

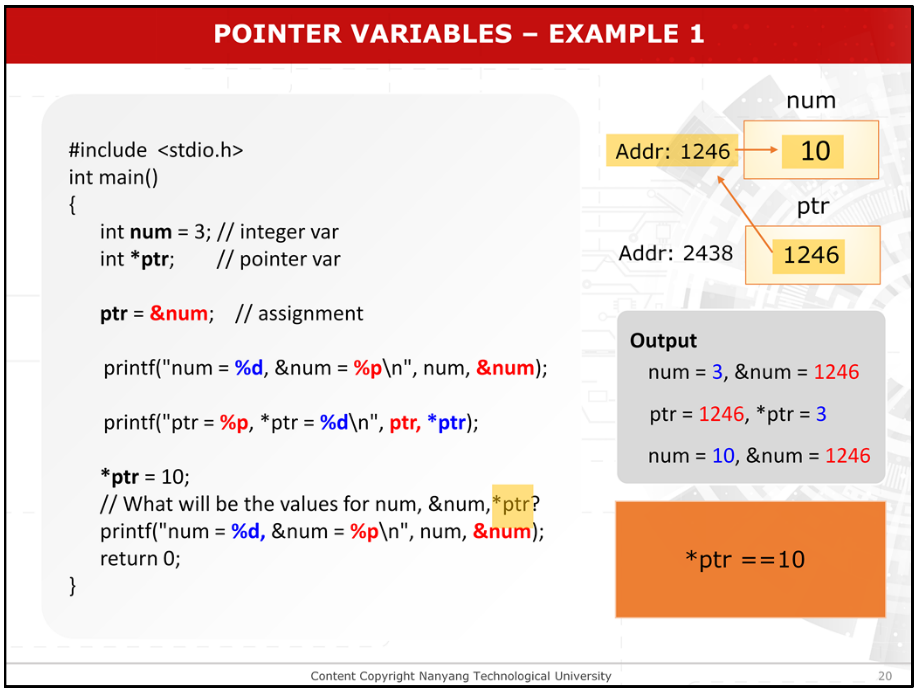
Output

```
num = 3, &num = 1246

ptr = 1246, *ptr = 3

num = 10, &num = 1246
```

%p prints the value of the memory address (&num) of the variable

19

There is no change to the address of the memory location of **num.** So the output will be ampersand num equals 1246.



POINTER VARIABLES – EXAMPLE 1

```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

     printf("num = %d, &num = %p\n", num, &num);

     printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

num
Addr: 1246
10
ptr
Addr: 2438
1246

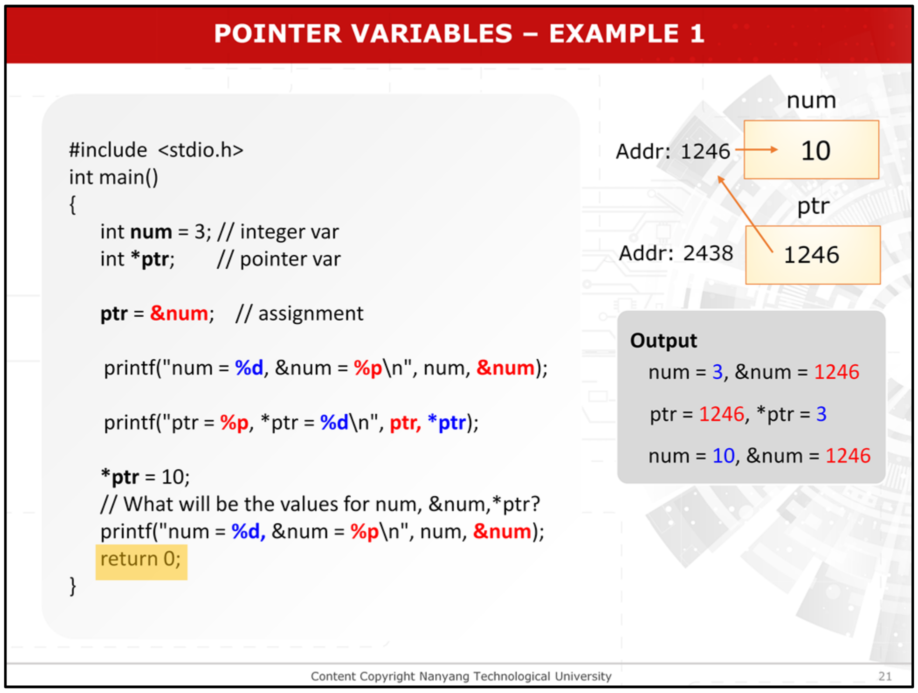
Output

```
num = 3, &num = 1246
ptr = 1246, *ptr = 3
num = 10, &num = 1246
```

*ptr ==10

Content Copyright Nanyang Technological University 20

How about the value of asterisk pointer? Basically, asterisk pointer points to the value of num. So the value of asterisk pointer is 10.



POINTER VARIABLES – EXAMPLE 1

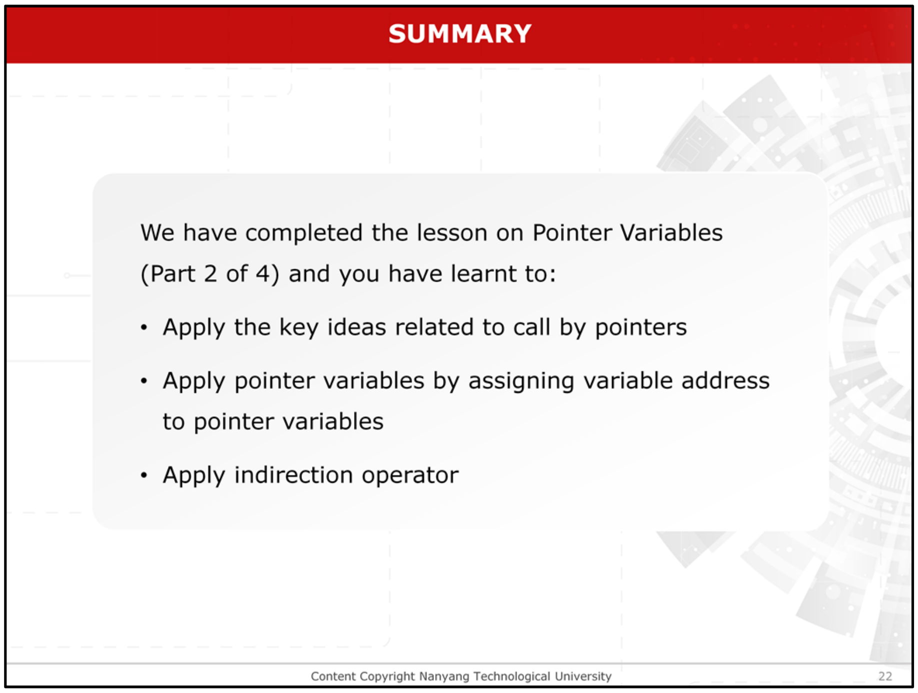
```
#include  <stdio.h>
int main()
{
     int num = 3; // integer var
     int *ptr;       // pointer var

     ptr = &num;    // assignment

      printf("num = %d, &num = %p\n", num, &num);

      printf("ptr = %p, *ptr = %d\n", ptr, *ptr);

     *ptr = 10;
     // What will be the values for num, &num,*ptr?
     printf("num = %d, &num = %p\n", num, &num);
     return 0;
}
```

Output

```
num = 3, &num = 1246

ptr = 1246, *ptr = 3

num = 10, &num = 1246
```

21

The whole program ends with the statement return zero.



SUMMARY

We have completed the lesson on Pointer Variables (Part 2 of 4) and you have learnt to:

- Apply the key ideas related to call by pointers
- Apply pointer variables by assigning variable address to pointer variables
- Apply indirection operator

22

Through this code example, you have learnt the points listed. In the next lesson, we will look into applying these points again in another slightly more complicated code.