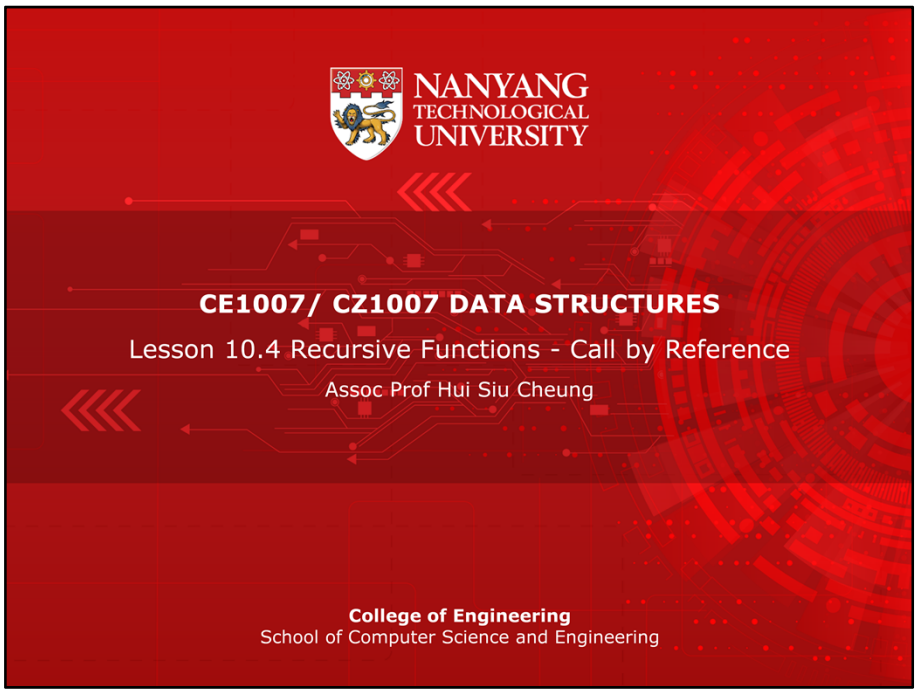
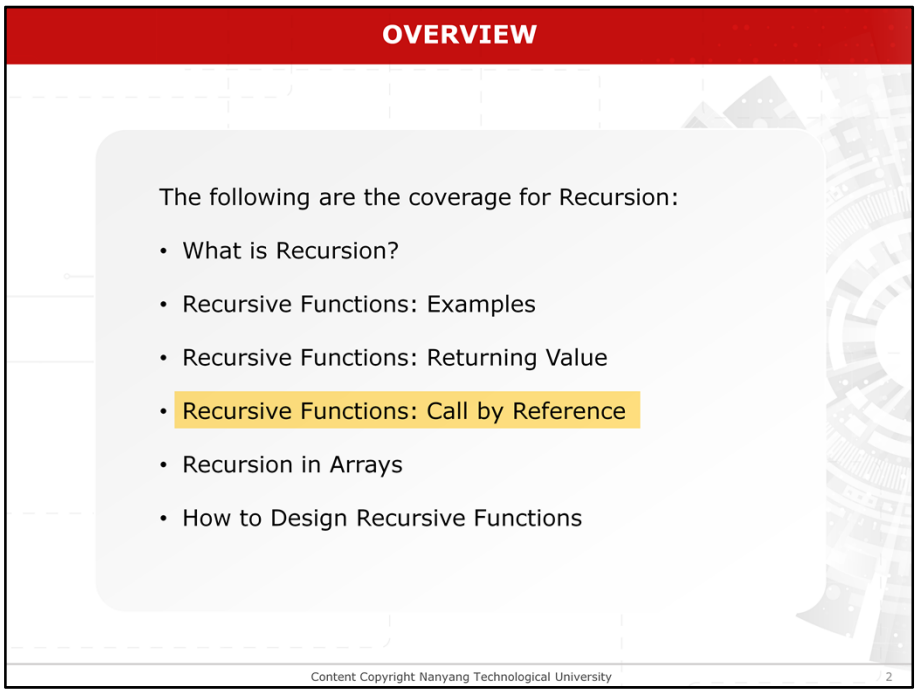
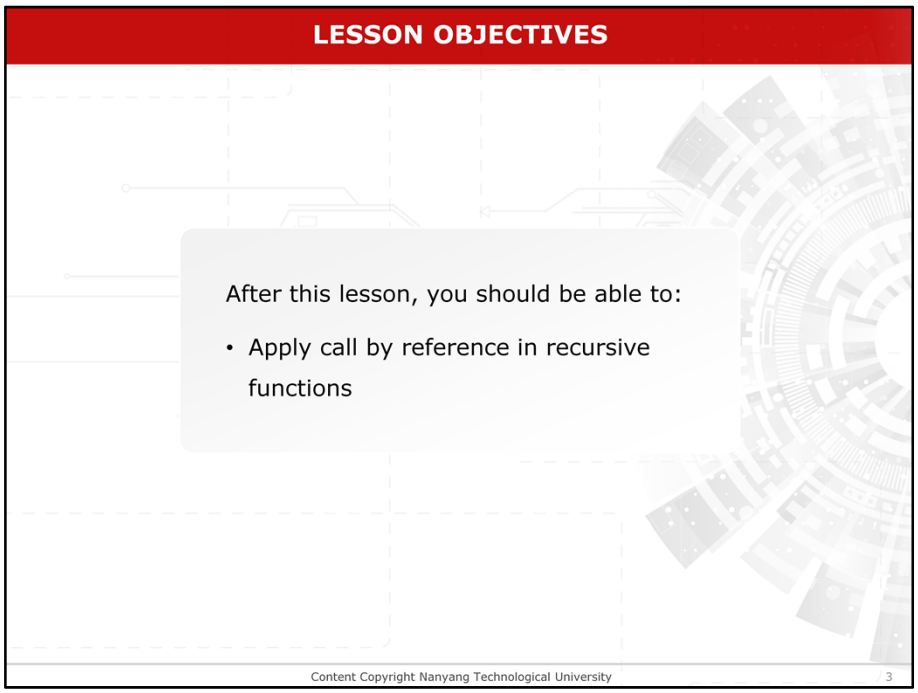
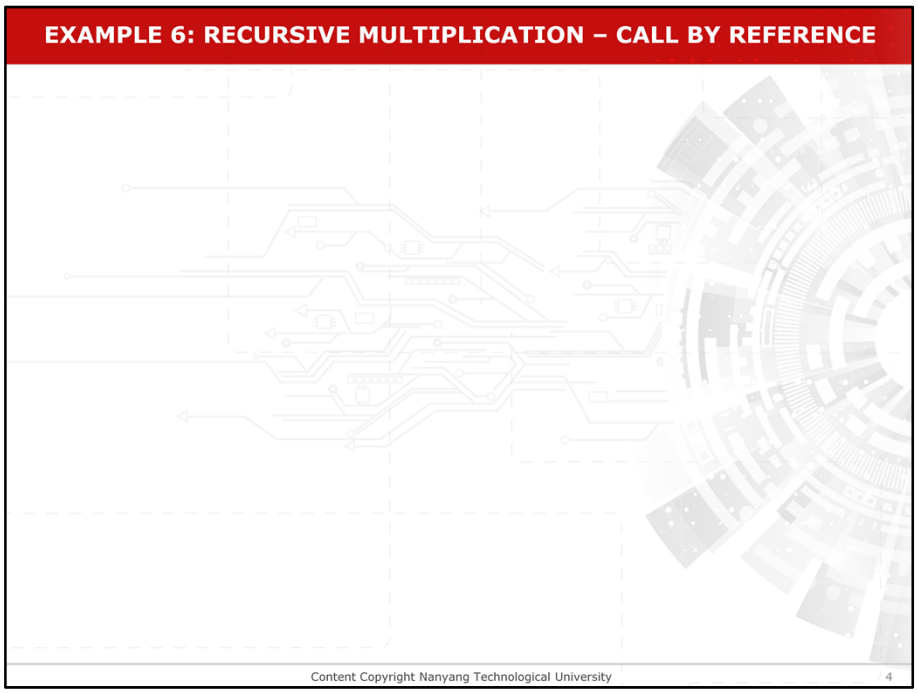

NANYANG
TECHNOLOGICAL
UNIVERSITY
CE1007/ CZ1007 DATA STRUCTURES
Lesson 10.4 Recursive Functions - Call by Reference
Assoc Prof Hui Siu Cheung
College of Engineering
School of Computer Science and Engineering



There are 6 main sections to cover for Recursion as shown. This video lesson focuses on the fourth part where we will look into recursive functions using call by reference.



After this lesson, you should be able to apply call by reference in recursive functions.



Example 6: Recursive Multiplication, Call by Reference

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```


This example works on the same problem as Example 5 covered in previous video lecture.

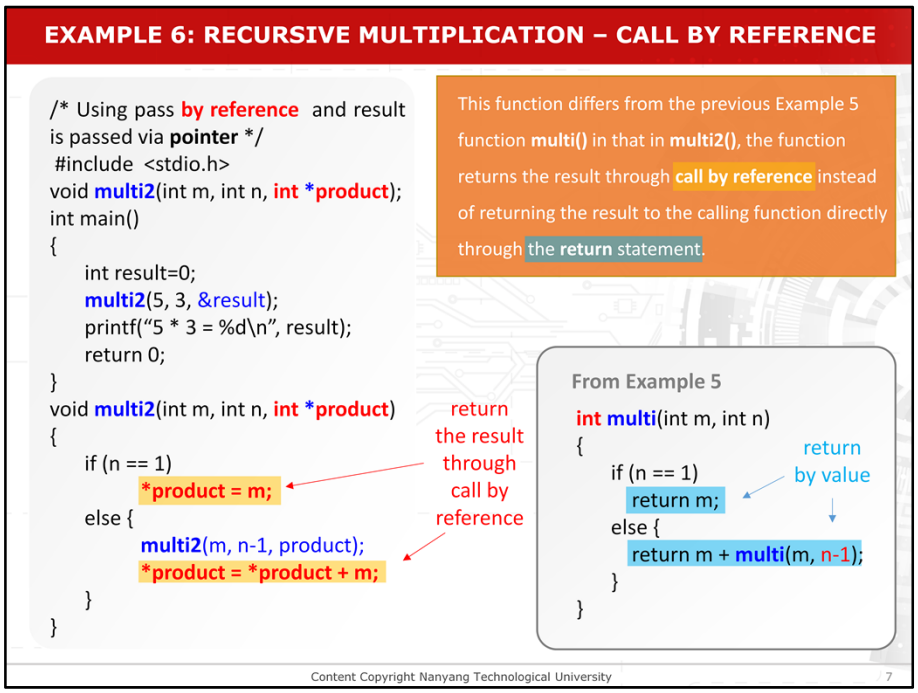
However, instead of using call by value via returning value, this example will implement the recursive function for integer multiplication via call by reference.

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

In the recursive function **multi2()**, it uses call by reference for parameter passing.

In the recursive function **multi 2()**, it uses call by reference for parameter passing.



This function differs from the previous Example 5 function **multi()** in that in **multi 2()**, the function returns the result through **call by reference** instead of returning the result to the calling function directly through the **return** statement.

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

Returning the result through **call by reference** is achieved through the use of the pointer parameter **product** of type **int**.

return the result through call by reference

Returning the result through **call by reference** is achieved through the use of the pointer parameter **product** of type **integer**.

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

The other two parameters, **m** and **n**, are specified of type **int** in the function.

The other two parameters, **m** and **n**, are specified of type **integer** in the function.

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

terminating condition

The terminating condition is when n is equal to 1.

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

recursive condition

The recursive condition is when n is greater than 1.

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
    int result=0;
    multi2(5, 3, &result);
    printf("5 * 3 = %d\n", result);
    return 0;
}
void multi2(int m, int n, int *product)
{
    if (n == 1)
            *product = m;
    else {
            multi2(m, n-1, product);
            *product = *product + m;
    }
}
```

In the **main()** function, the variable **result** is declared to hold the result of the multiplication operation.

result

0

In the **main()** function, the variable **result** is declared to hold the result of the multiplication operation.

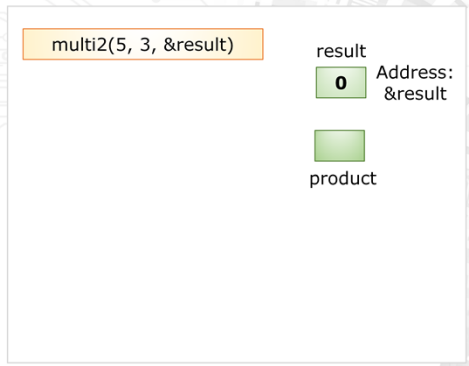
EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

When it calls the function **multi2()**, it passes the address of the variable **result** via call by reference:

multi2(5, 3, &result);

In the function **multi2()**, it declares a pointer parameter **product** of **int** type.



When it calls the function **multi 2()**, it passes the address of the variable **result** via call by reference:

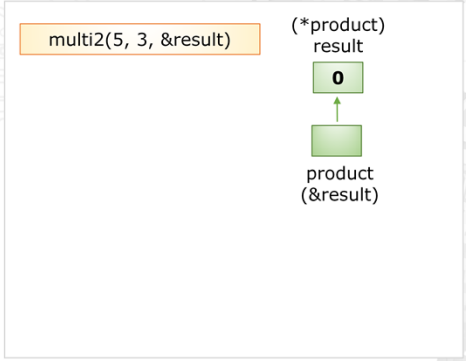
multi 2 (5, 3, ampersand result);

In the function **multi 2()**, it declares a pointer parameter **product** of **integer** type.

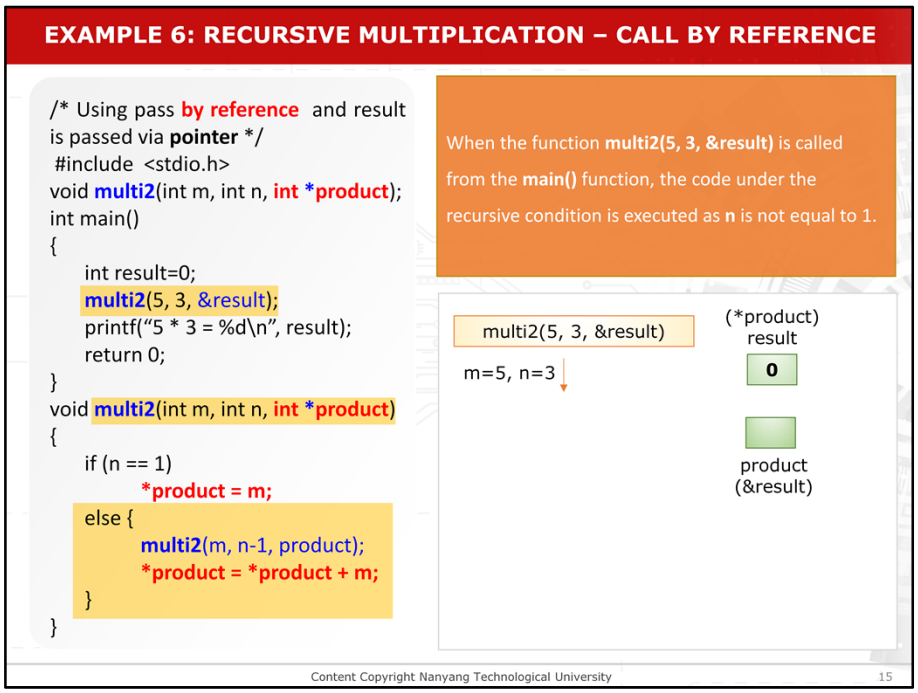
EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

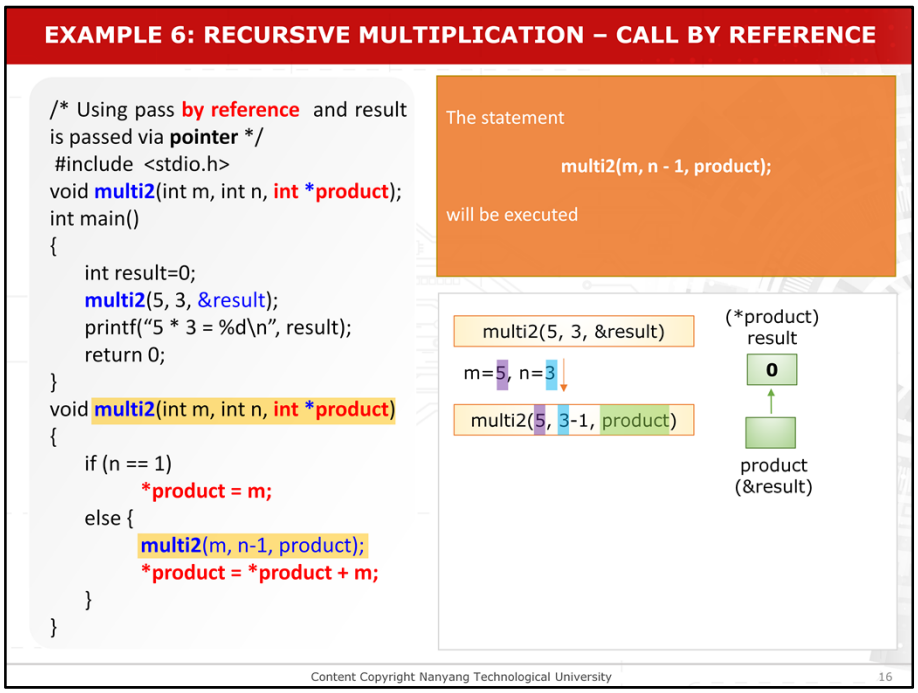
By passing in the address of the variable **result** to **product**, the result will be processed and updated in the memory location of the variable **result**.



By passing in the address of the variable **result** to **product**, the result will be processed and updated in the memory location of the variable **result**.



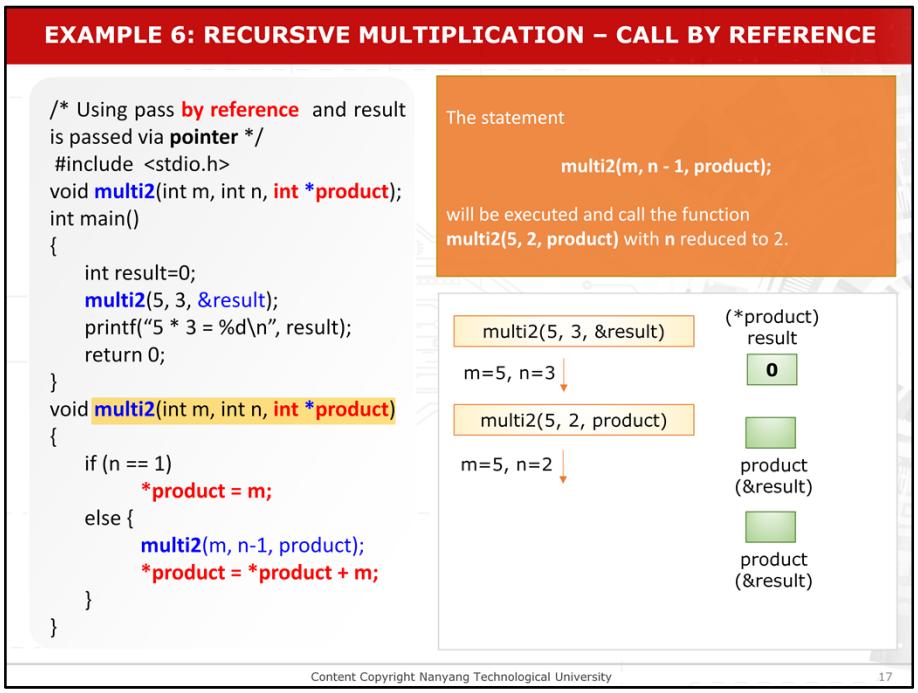
When the function **multi 2 (5, 3, ampersand result)** is called from the **main()** function, the code under the recursive condition is executed as **n** is not equal to 1.



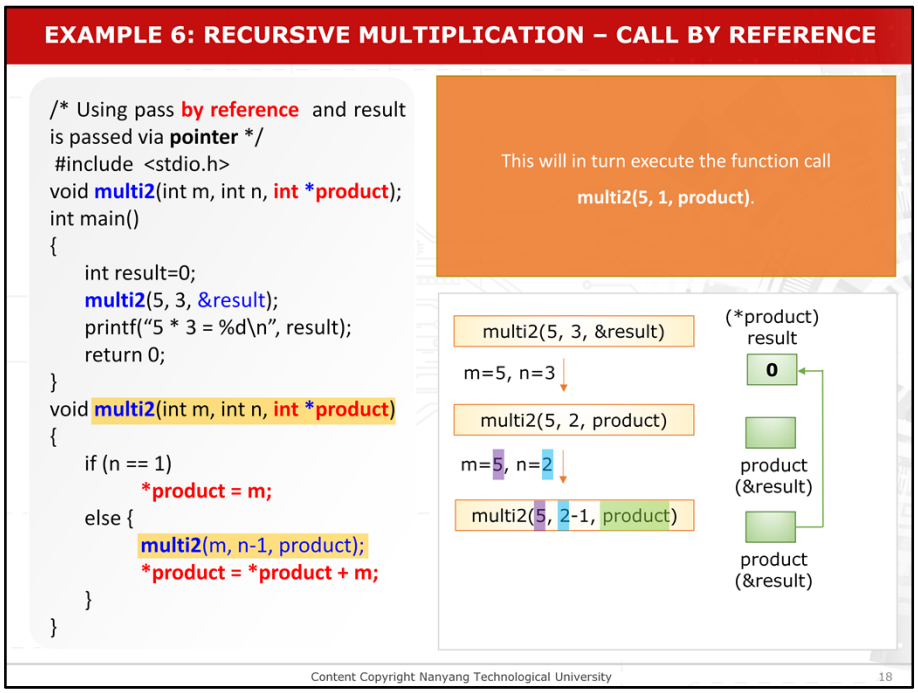
The statement

multi 2 (m, n minus 1, product);

will be executed



and call the function **multi 2 (5, 2, product)** with **n** reduced to 2.

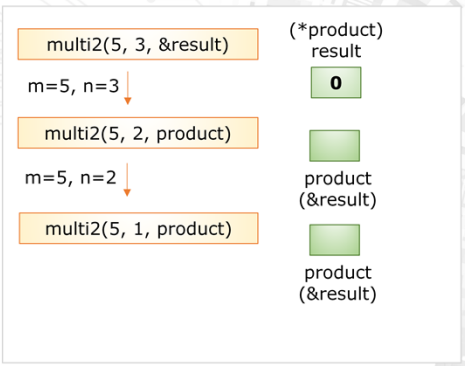


This will in turn execute the function call **multi 2 (5, 1, product)**.

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

This will in turn execute the function call **multi2(5, 1, product)**.

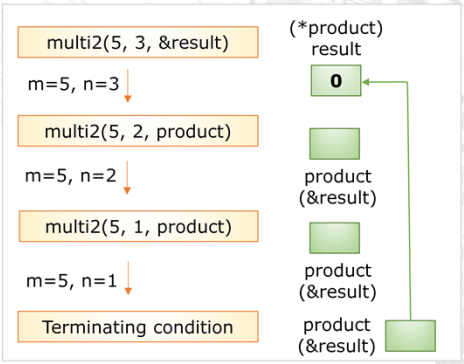


[no audio]

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

When the function **multi2(5, 1, product)** is executed, the code under the terminating condition will be executed.

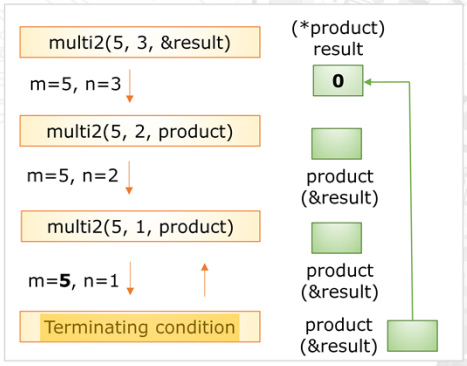


When the function **multi 2 (5, 1, product)** is executed, the code under the terminating condition will be executed.

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

The value 5 will be stored in result via ***product**



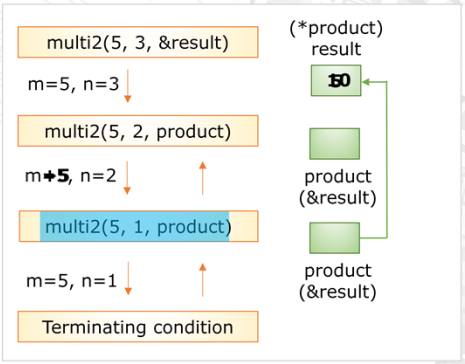
The value 5 will be stored in result via **asterisk product**

EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

multi2() will store the updated result via ***product** as:

***product = *product + m;**



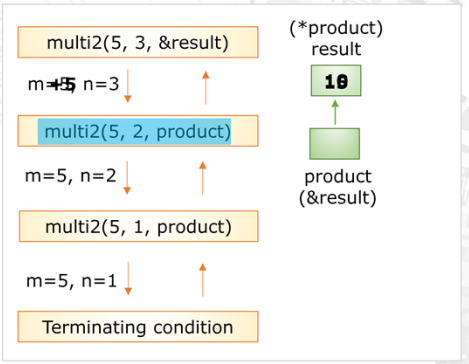
Multi 2() will store the updated result via **asterisk product** as:

asterisk product = asterisk product + m;

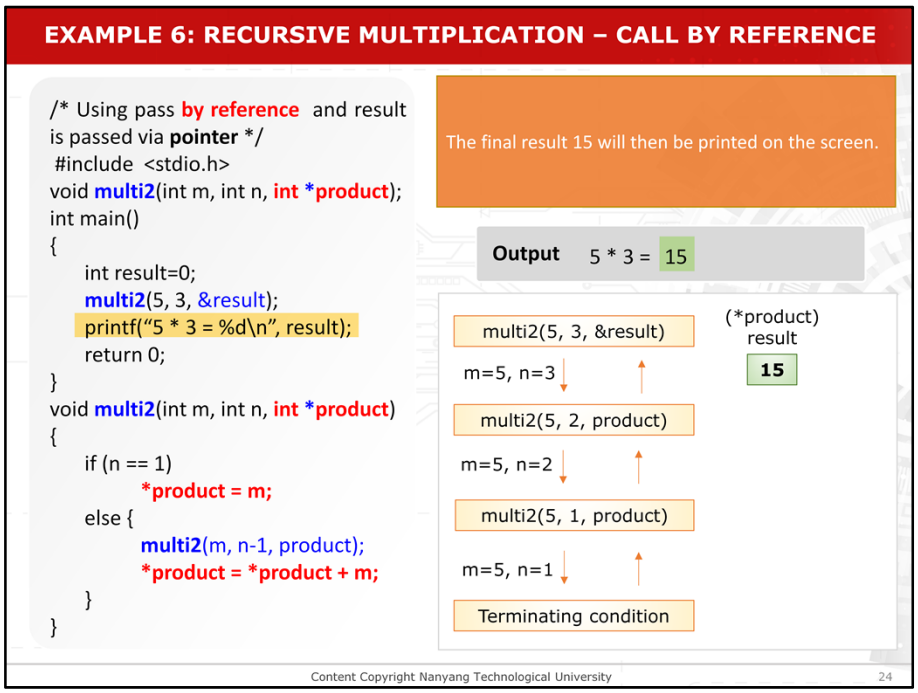
EXAMPLE 6: RECURSIVE MULTIPLICATION – CALL BY REFERENCE

```
/* Using pass by reference  and result
is passed via pointer */
 #include  <stdio.h>
void multi2(int m, int n, int *product);
int main()
{
     int result=0;
     multi2(5, 3, &result);
     printf("5 * 3 = %d\n", result);
     return 0;
}
void multi2(int m, int n, int *product)
{
     if (n == 1)
          *product = m;
     else {
          multi2(m, n-1, product);
          *product = *product + m;
     }
}
```

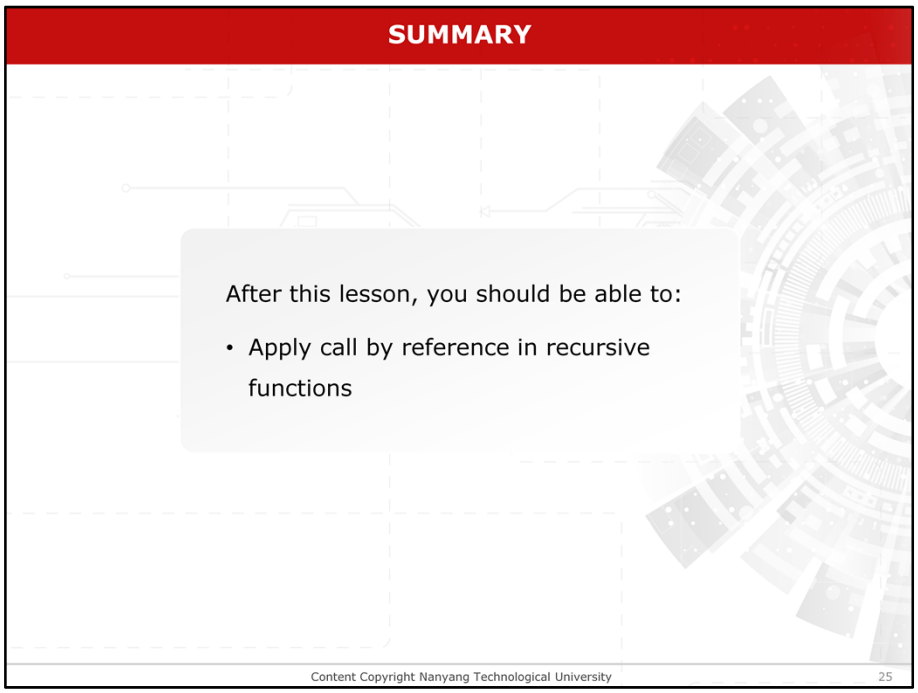
That is, after each function call, the result will be updated directly via the memory location of the variable **result** in the **main()** function.



That is, after each function call, the result will be updated directly via the memory location of the variable **result** in the **main()** function.



The final result 15 will then be printed on the screen.



In summary, after viewing this video lesson, you should be able to apply call by reference in recursive functions.