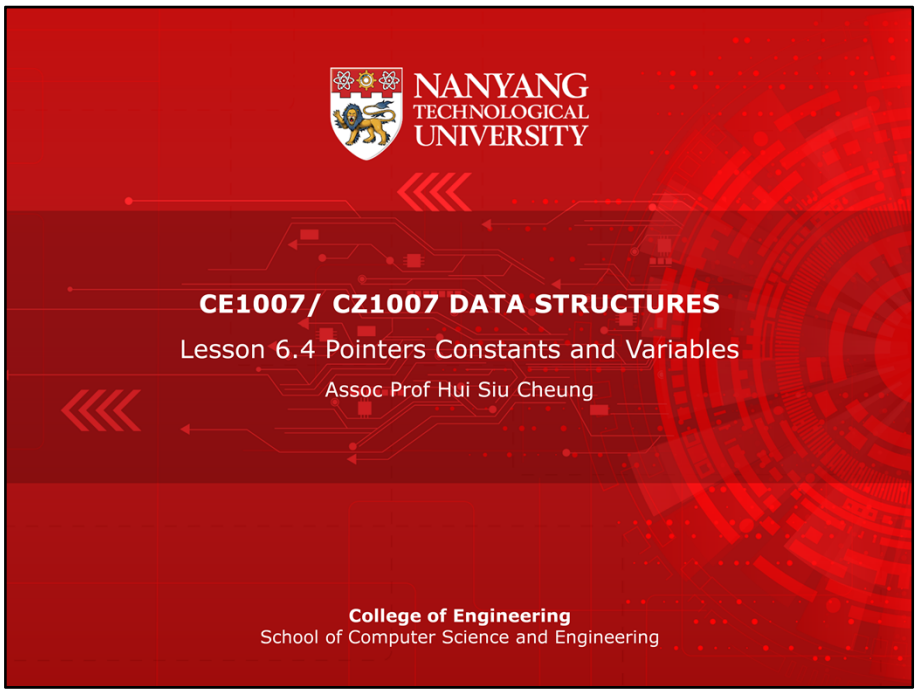
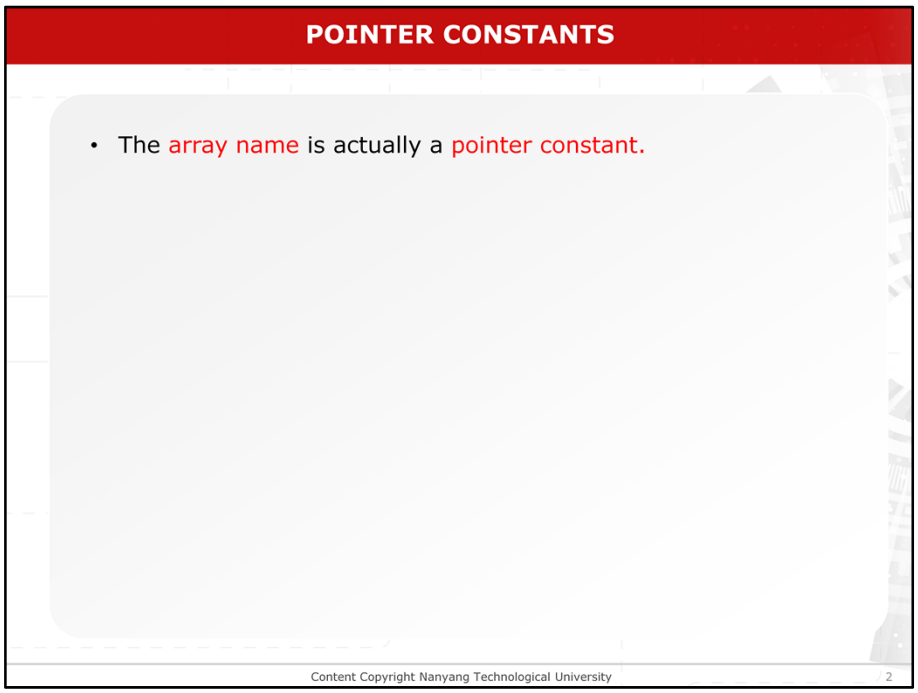


Lesson 6.4 Pointers Constants and Variables

In this lesson, we will learn about pointer constants and variables



There is a strong relationship between pointers and arrays. The array name is actually a pointer constant.

POINTER CONSTANTS

- The array name is actually a pointer constant.

```
int days[4];              // days – pointer constant
```

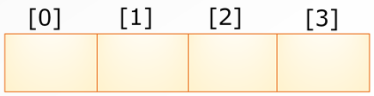

The statement

integer days [4];

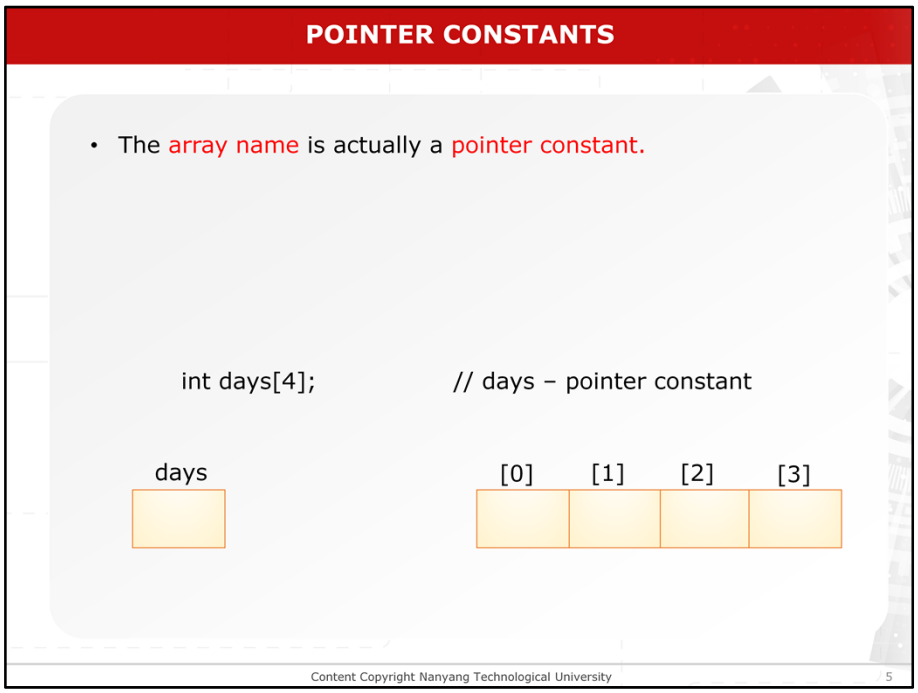
POINTER CONSTANTS

- The array name is actually a pointer constant.

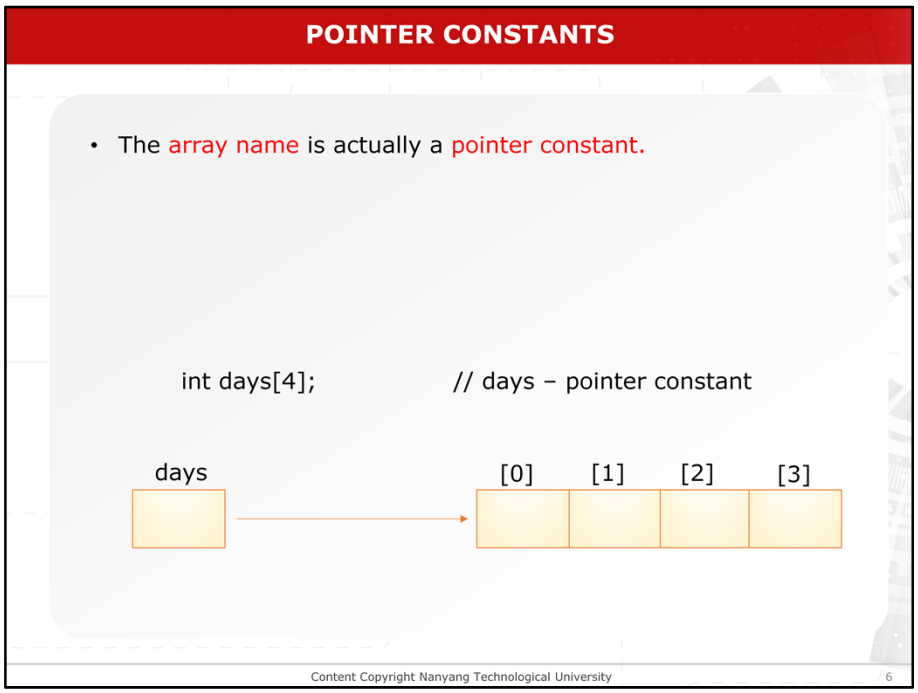
```
int days[4];          // days – pointer constant
```



Creates an array that consists of 4 elements.

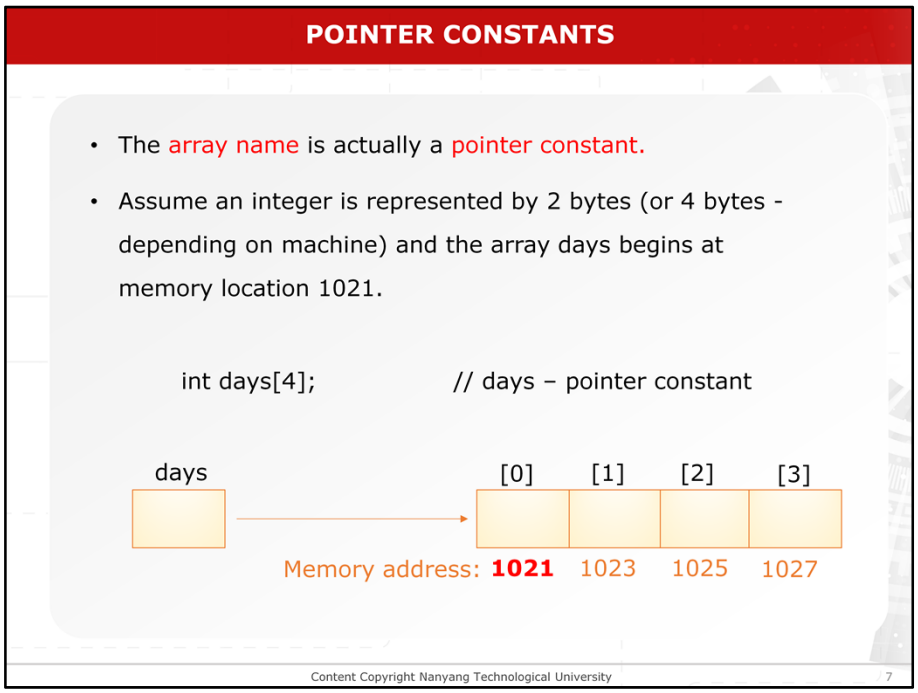


a *pointer constant* with the same name as the array is also created.

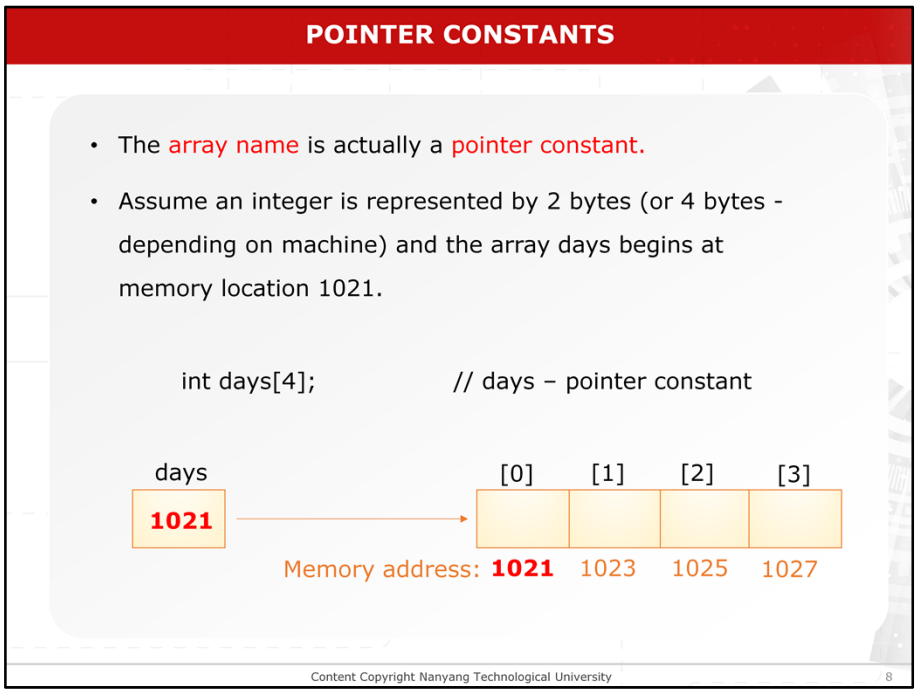


The pointer constant points to the first element of the array.

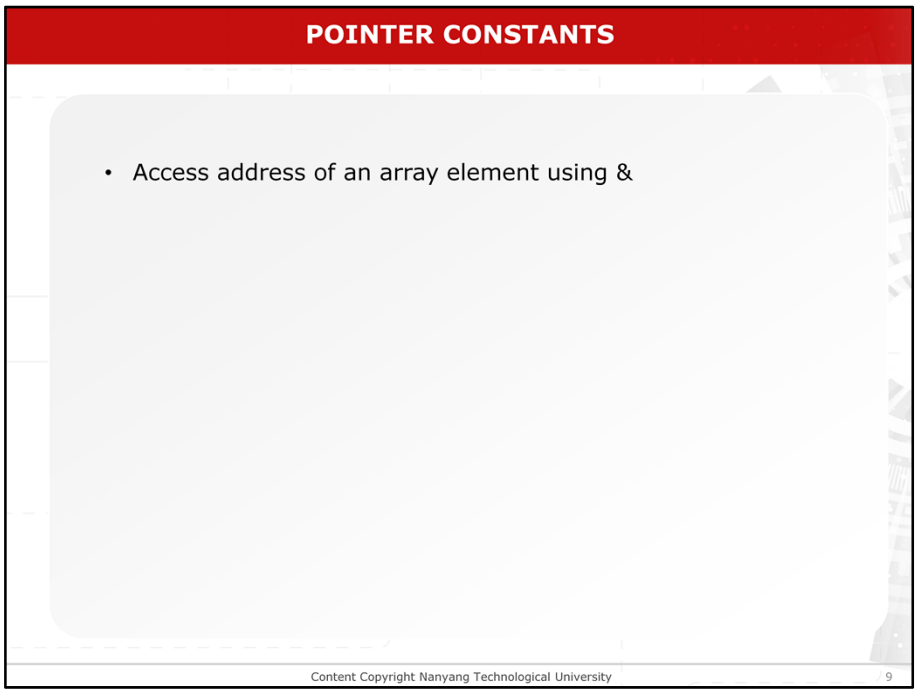
Therefore, the array name by itself, **days**, is actually the address (or pointer) of the first element of the array.



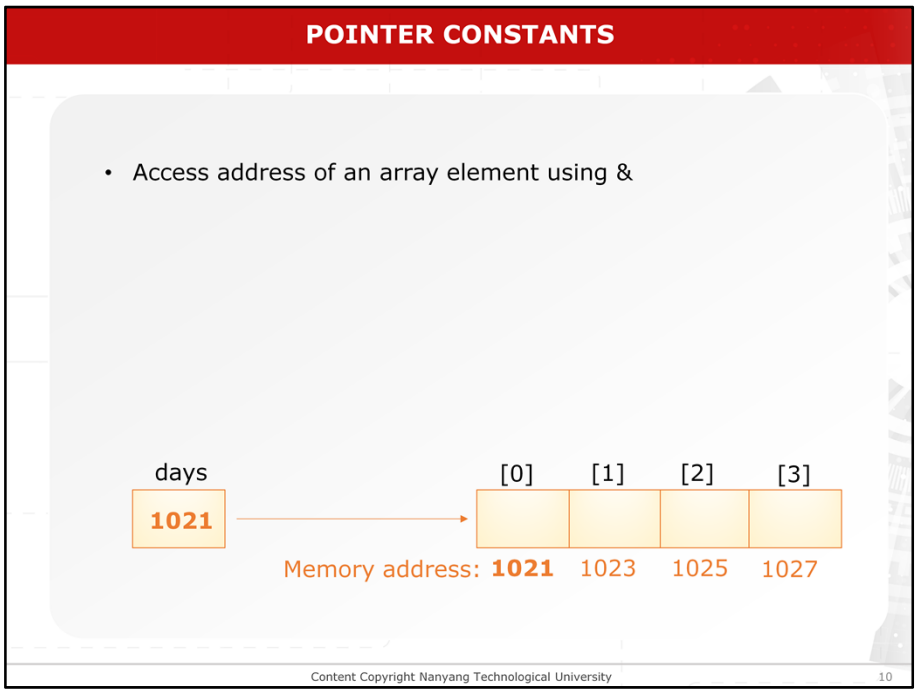
Assume an integer is represented by 2 bytes (in older machines) and the array **days** begins at memory location 1021.



The value stored at **days** is 1021, which corresponds to the address of the first element of the array.

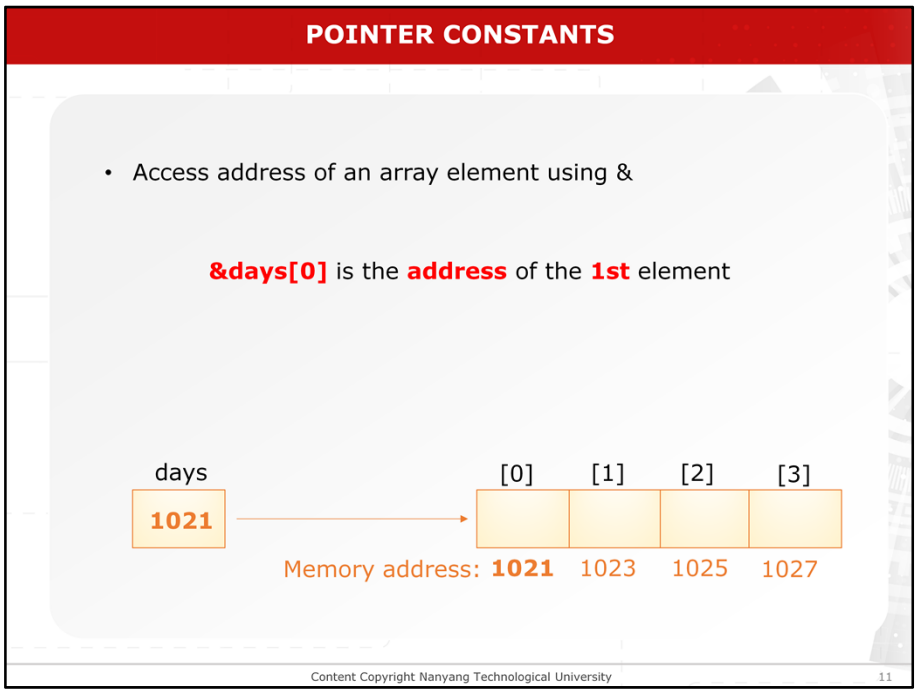


Note that to access the address of an array element, we can use the address operator, the ampersand sign.

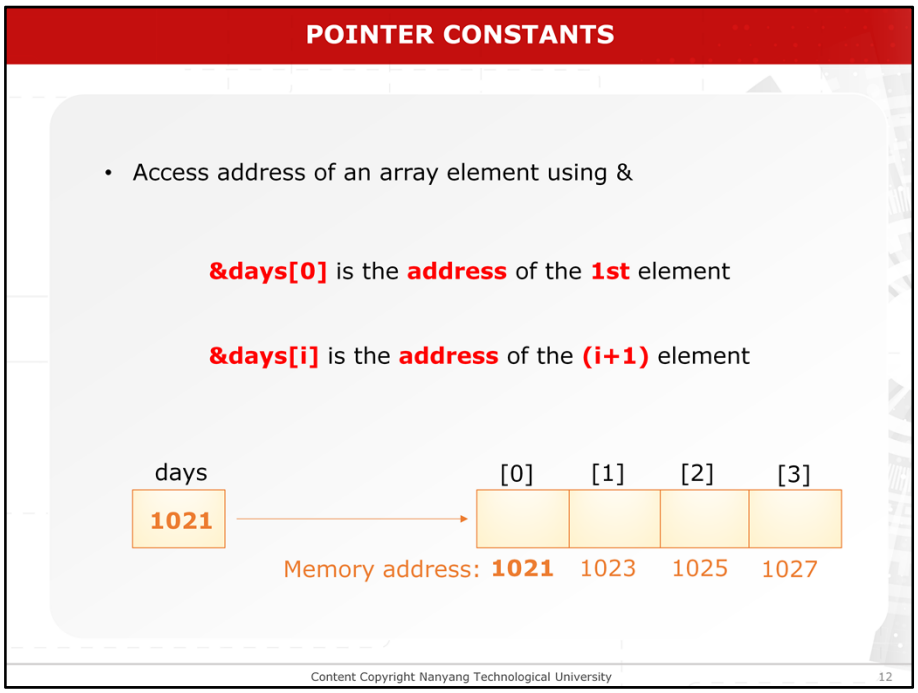


For example, if we declare an array as

integer days [4];



then **ampersands days square bracket 0 square bracket** is the address of the 1st element;

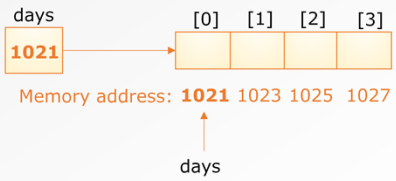


and **ampersands days square bracket I square bracket** is the address of the (i+1) element.
The address of an array element is important when performing pointer arithmetic with array.

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```

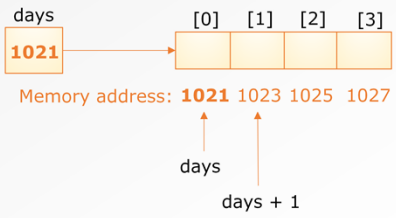


Thus, the array name by itself, **days**, is really the address (or pointer) of the 1st element of the array declared.

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```

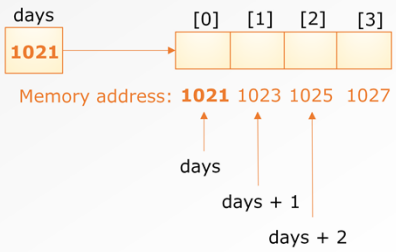


Since the array name is the pointer to the first element of the array, we have days plus 1 points to the second element.

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```

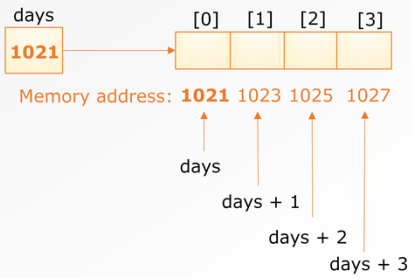


days plus 2 points to the third element.

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```

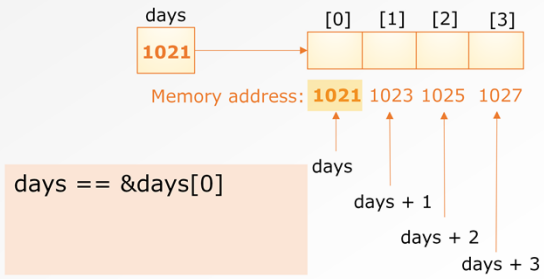


And days plus 3 points to the fourth element and so on.

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```




```
days == &days[0]
```

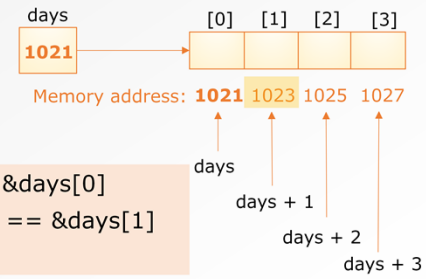
days
days + 1
days + 2
days + 3

So we have days is equivalent to the address of the first element.

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```



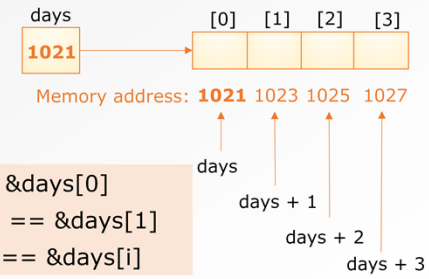

```
days == &days[0]
days + 1 == &days[1]
```


Days + 1 is equivalent to the address of the second element.

USING POINTERS IN ARRAYS

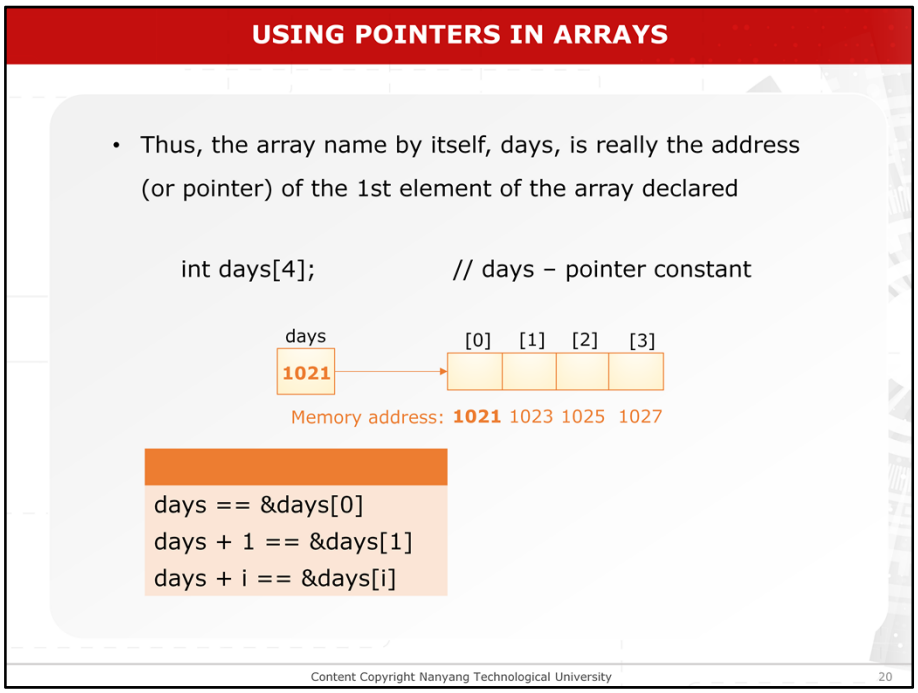
- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```




```
days == &days[0]
days + 1 == &days[1]
days + i == &days[i]
```


And in general, days plus I is equivalent to the address of the I plus 1 element.

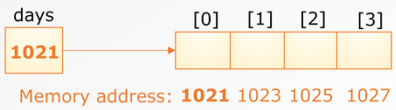


This is the array method using the pointer constant. **[This is not needed to spell out]**

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```



Memory address: **1021** 1023 1025 1027

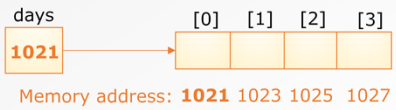
```
days == &days[0]
days + 1 == &days[1]
days + i == &days[i]
```


There is another way to retrieve the content of the element of the arrays. It is using the pointer notation.

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

```
int days[4];          // days – pointer constant
```



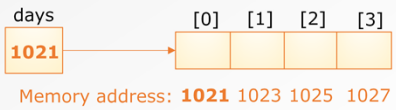

```
days == &days[0]          *days == days[0]
days + 1 == &days[1]
days + i == &days[i]
```


To access the first element which is array element days zero, we can use asterisk days.

USING POINTERS IN ARRAYS

- Thus, the array name by itself, days, is really the address (or pointer) of the 1st element of the array declared

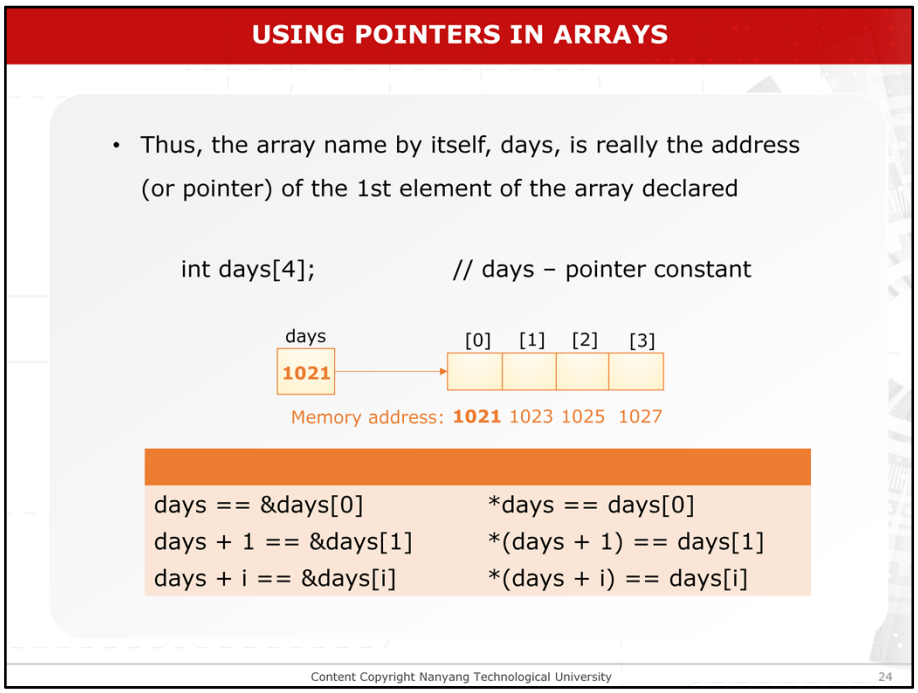
```
int days[4];          // days – pointer constant
```



Memory address: **1021** 1023 1025 1027

days == &days[0]	*days == days[0]
days + 1 == &days[1]	*(days + 1) == days[1]
days + i == &days[i]	

Similarly, we use asterisk days + 1 to access the array element days 1.

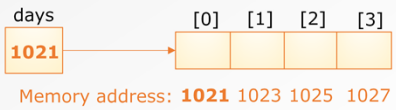


In general, asterisk days plus I access the array element days i.

However, it is important to note that the array name is a **pointer constant**, not a pointer variable. It means that the value stored in **days** cannot be changed by any statements.

USING POINTERS IN ARRAYS

- We **cannot change** the array **base pointer**



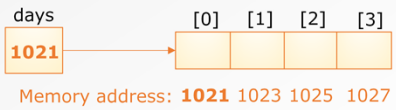
days == &days[0]	*days == days[0]
days + 1 == &days[1]	*(days + 1) == days[1]
days + i == &days[i]	*(days + i) == days[i]

We cannot change the array base pointer.

USING POINTERS IN ARRAYS

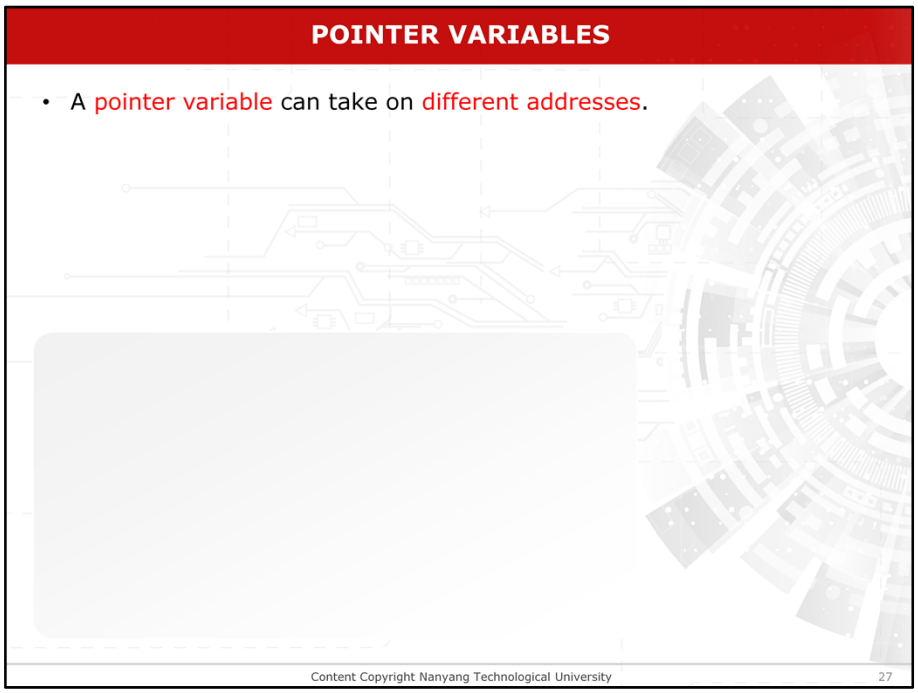
- We **cannot change** the array **base pointer**

```
days += 1;          // i.e. days = days + 1; not valid
days ++;            // i.e. days = days + 1; not valid
```



days == &days[0]	*days == days[0]
days + 1 == &days[1]	*(days + 1) == days[1]
days + i == &days[i]	*(days + i) == days[i]

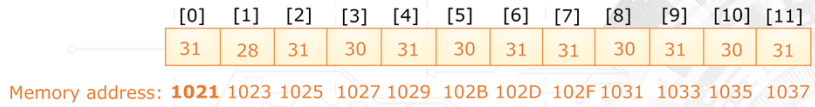
So days plus equals 1 or days plus plus which is the same as days increased by 1 is not a valid statement.



A *pointer variable* can take on different addresses.

POINTER VARIABLES

- A pointer variable can take on different addresses.

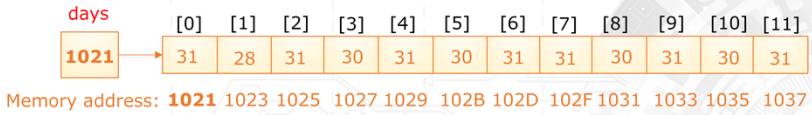



```
/* pointer arithmetic */
#define MTHS 12
int main()
{
    int days[MTHS]= {31,28,31,30,31,30,31,31,30,31,30,31};




}
```


In the program, we declare an array of 12 elements.

POINTER VARIABLES

- A pointer variable can take on different addresses.



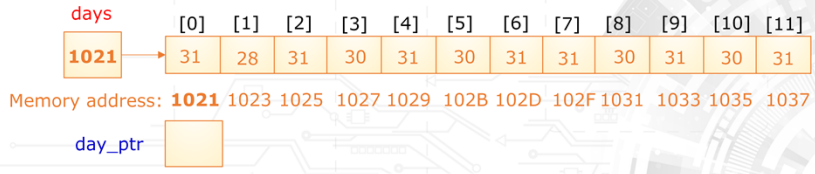

```
/* pointer arithmetic */
#define MTHS 12
int main()
{
    int days[MTHS]= {31,28,31,30,31,30,31,31,30,31,30,31};




}
```

days is a pointer constant which is declared as an array of 12 elements.

days is a pointer constant which is declared as an array of 12 elements.

POINTER VARIABLES

- A pointer variable can take on different addresses.



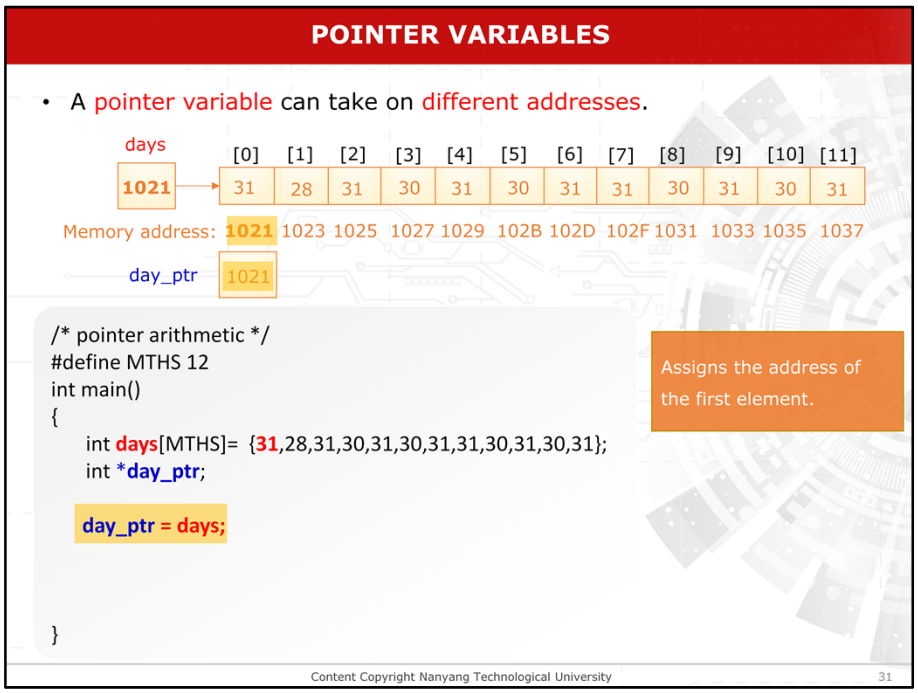

```
/* pointer arithmetic */
#define MTHS 12
int main()
{
    int days[MTHS]= {31,28,31,30,31,30,31,31,30,31,30,31};
    int *day_ptr;




}
```

day_ptr is a pointer variable.

If we declare

integer asterisk **day pointer**;

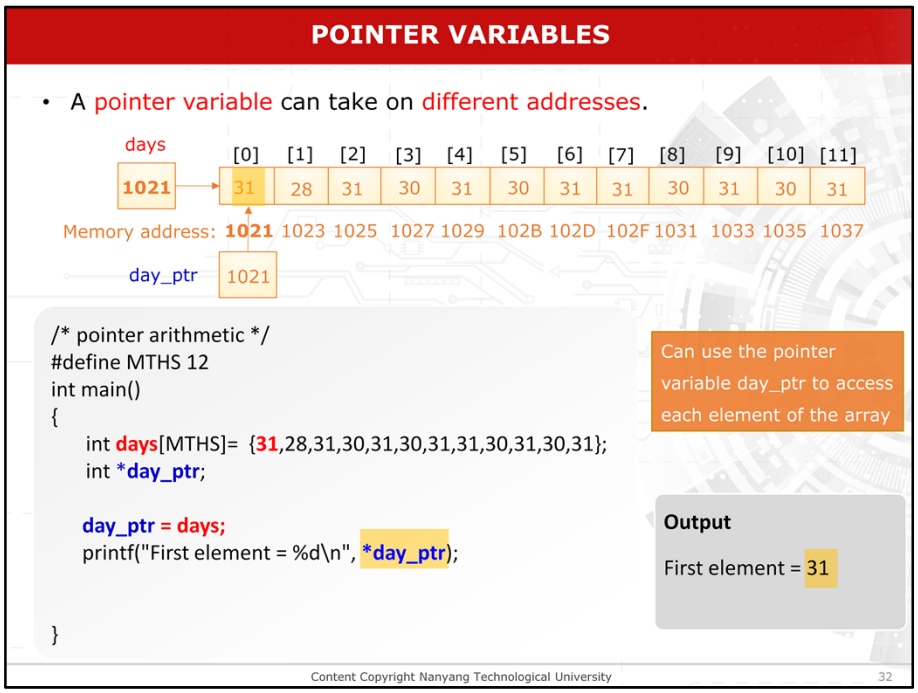
day pointer is a pointer variable.



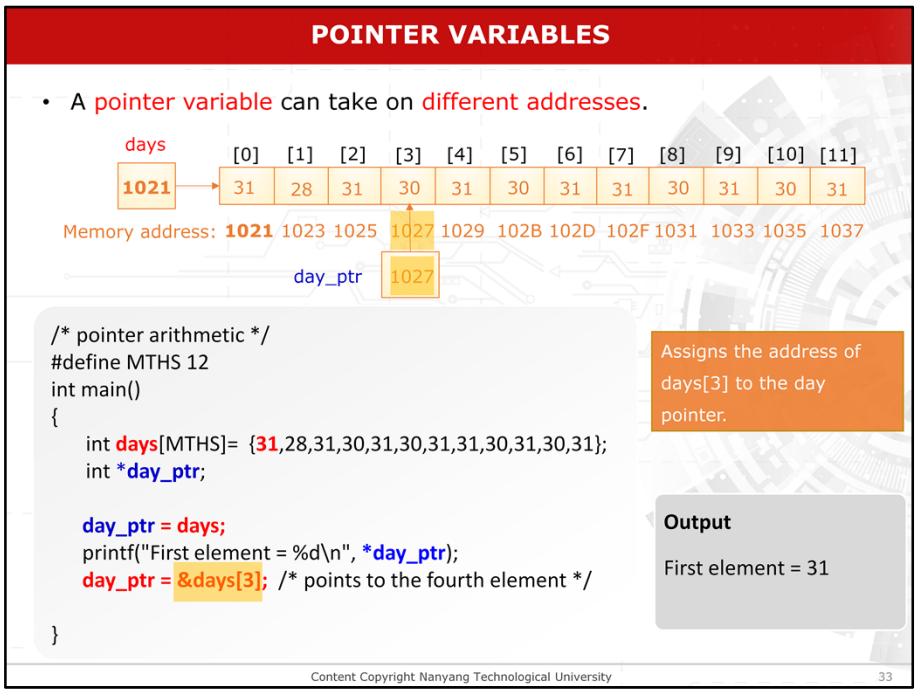
The statement

day pointer equals days;

assigns the address of the first element which is of value 1021 stored in **days** to the pointer variable **day pointer**.



This causes the pointer variable to point to the first element of the array.
Thus the value in the first element is printed.
After that, we can use the pointer variable **day pointer** to access each element of the array.



The statement

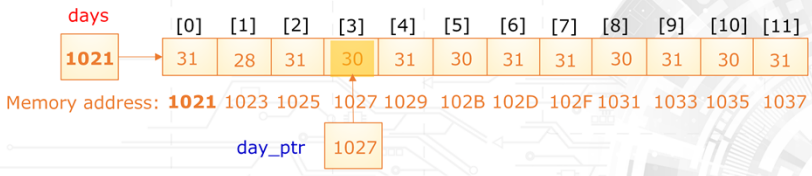
day pointer equals ampersands days[3];

assigns the address of **days[3]** to the **day pointer**.

It updates the **day pointer** to point to the fourth element of the array. The value stored in **day pointer** becomes 1027.

POINTER VARIABLES

- A pointer variable can take on different addresses.



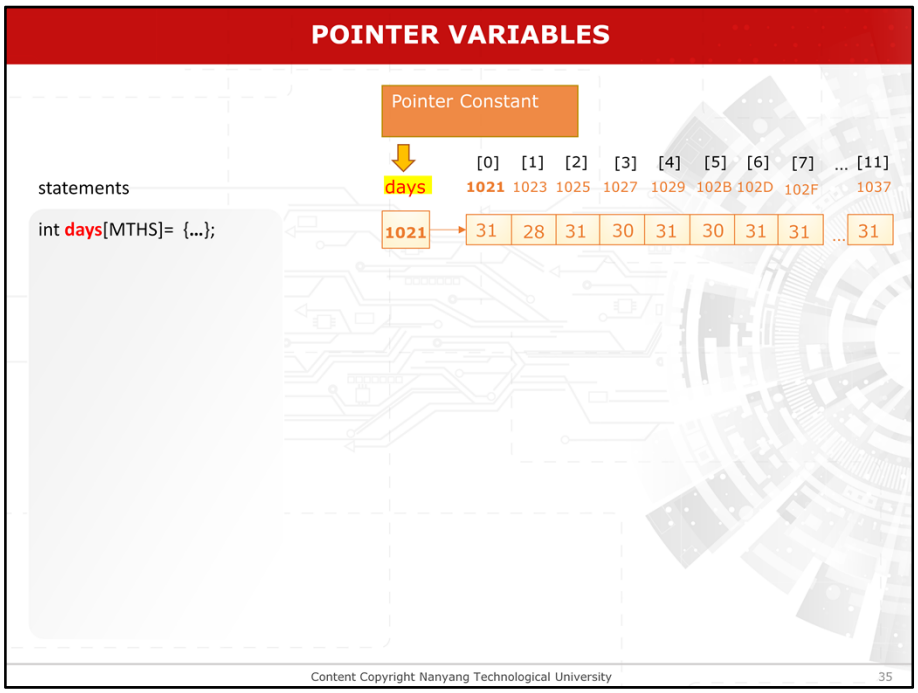

```
/* pointer arithmetic */
#define MTHS 12
int main()
{
    int days[MTHS]= {31,28,31,30,31,30,31,31,30,31,30,31};
    int *day_ptr;

    day_ptr = days;
    printf("First element = %d\n", *day_ptr);
    day_ptr = &days[3];  /* points to the fourth element */
    printf("Fourth element = %d\n", *day_ptr);
}
```

Thus the value in the fourth element is printed.

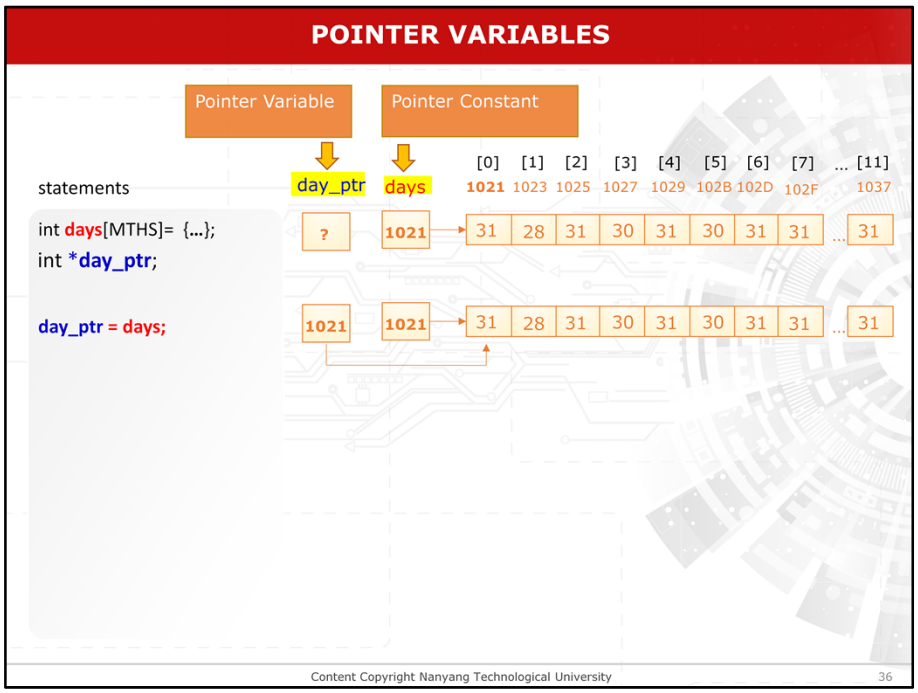
Output

First element = 31
Fourth element = 30

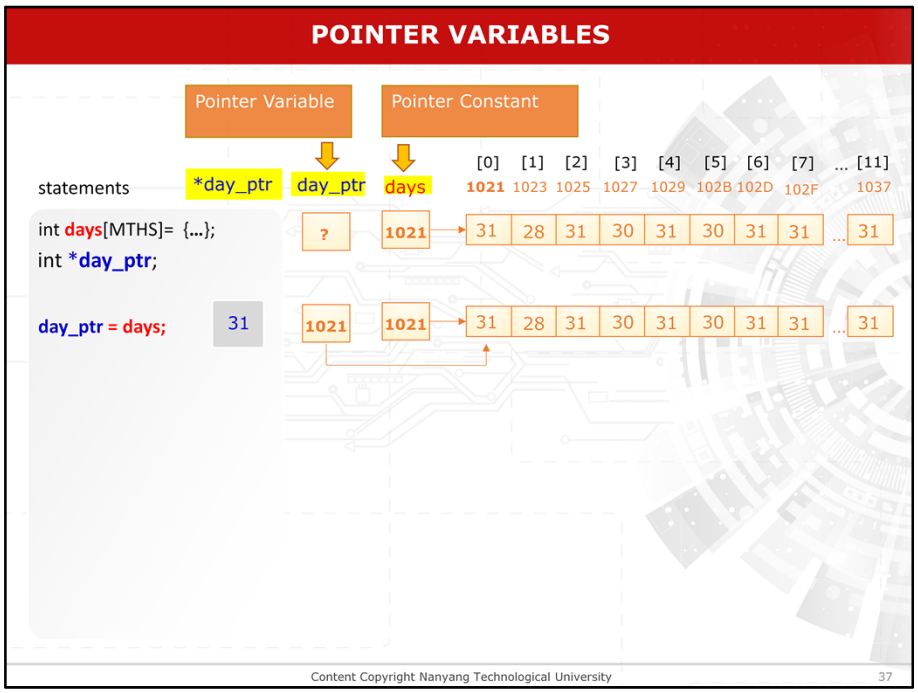
Thus the value in the fourth element is printed.



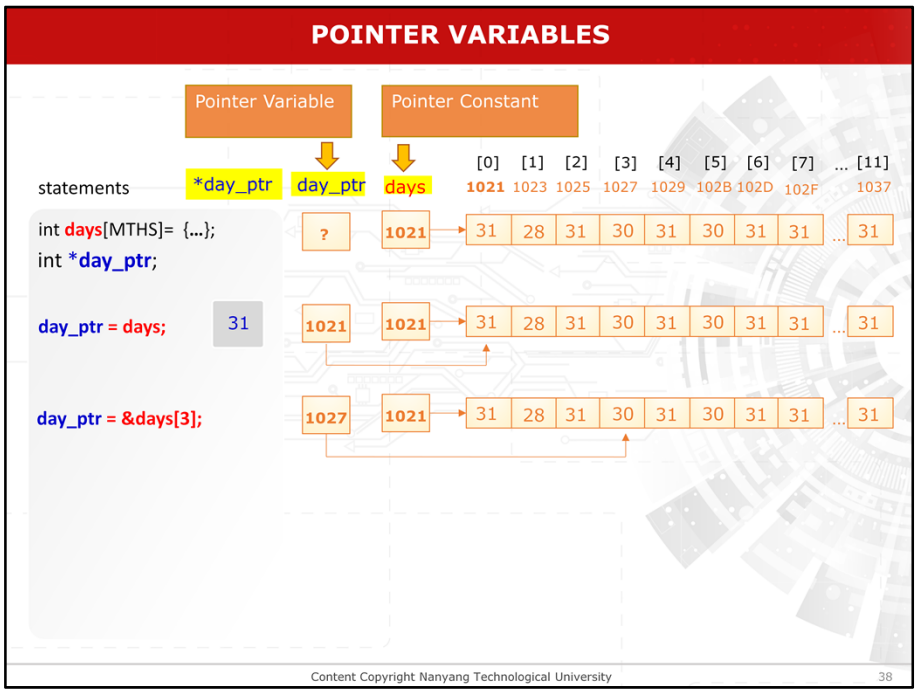
So far in this example, we have define days as a pointer constant which is declared as an array of 12 elements.



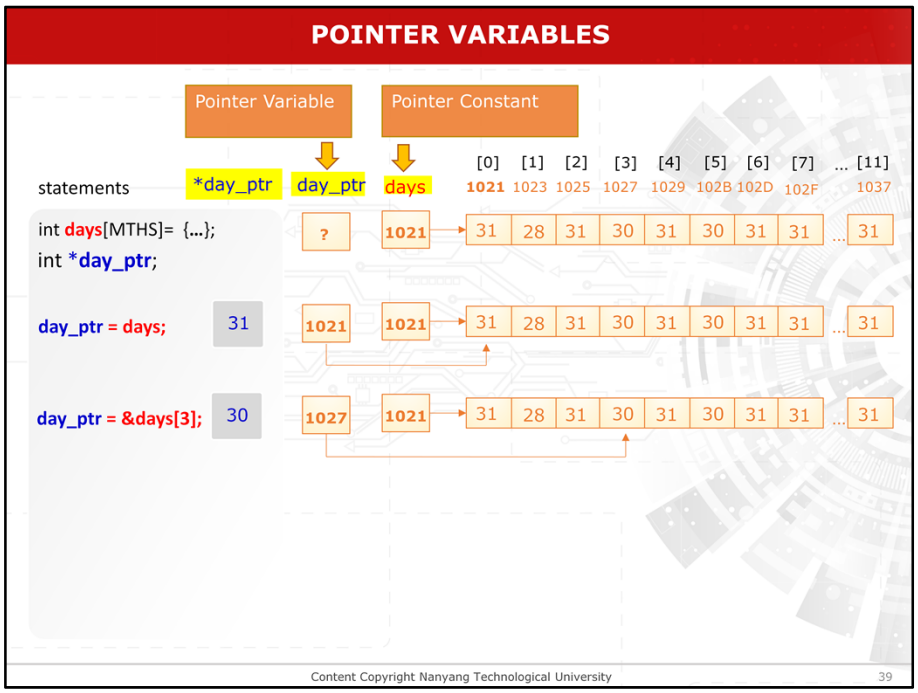
The statement day pointer equals days define day pointer as a pointer variable that points to the first element of the array.



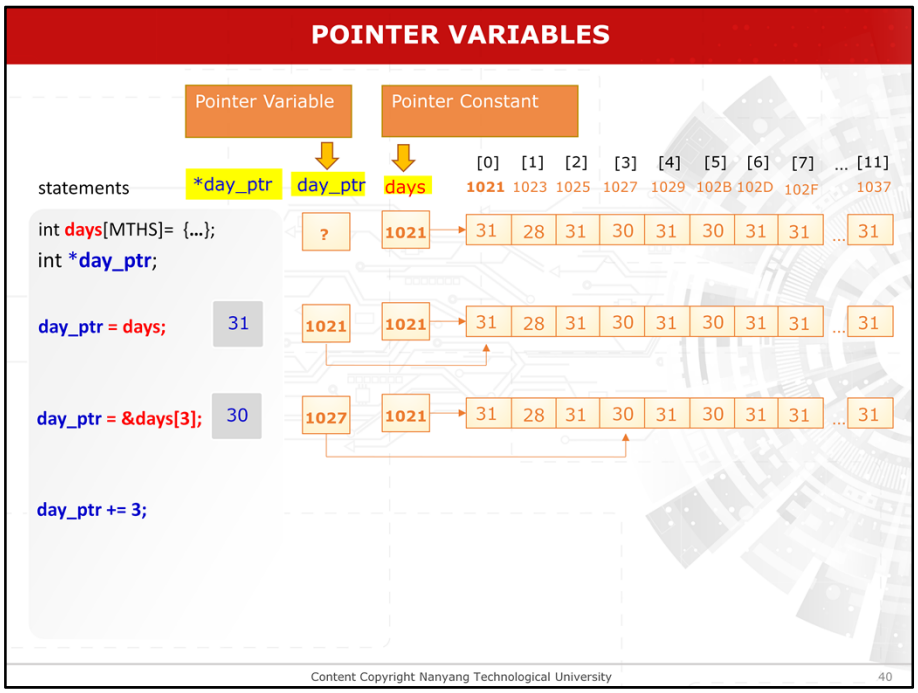
Thus asterisk day pointer will give the value of the first element, that is, 31.



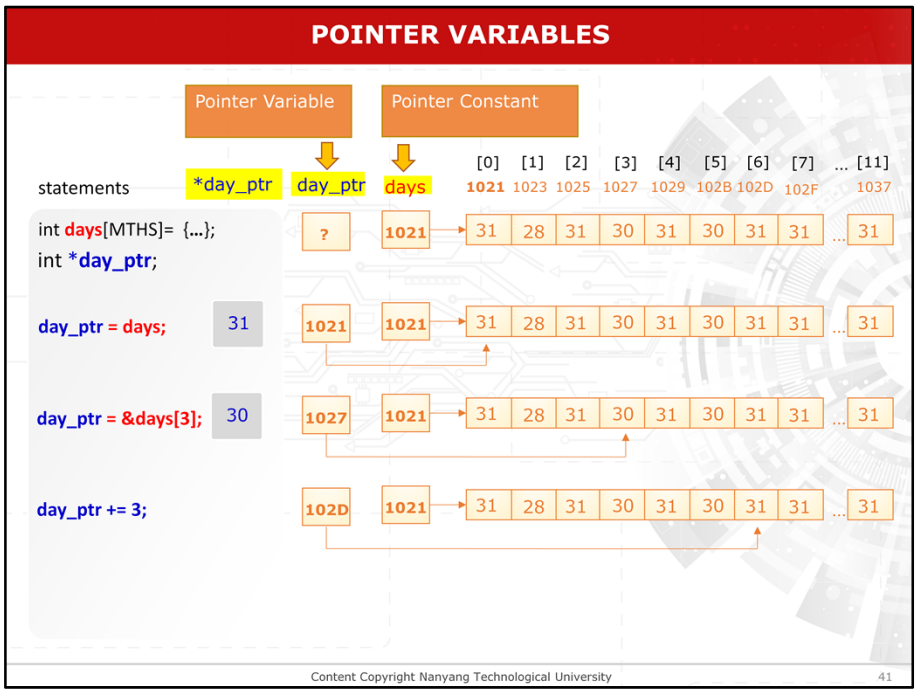
The statement day pointer equals ampersands days square bracket 3 updates the day pointer to point to the fourth element of the element. The value stored in day pointer becomes 1027.



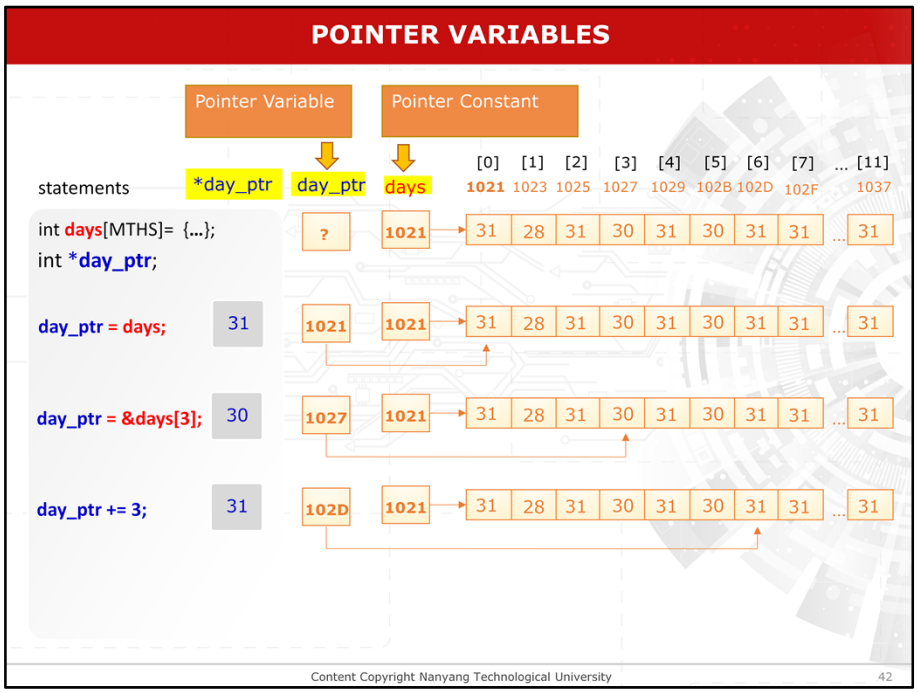
So asterisk day pointer will give the value of the fourth element, that is, 30.



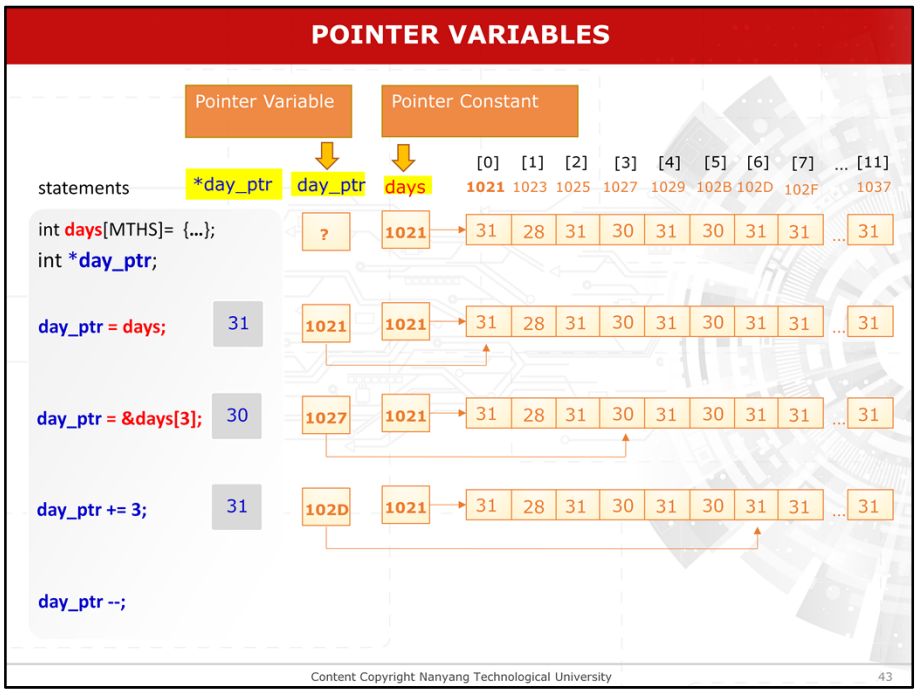
We can also add an integer value of 3 to the pointer variable day pointer as shown.



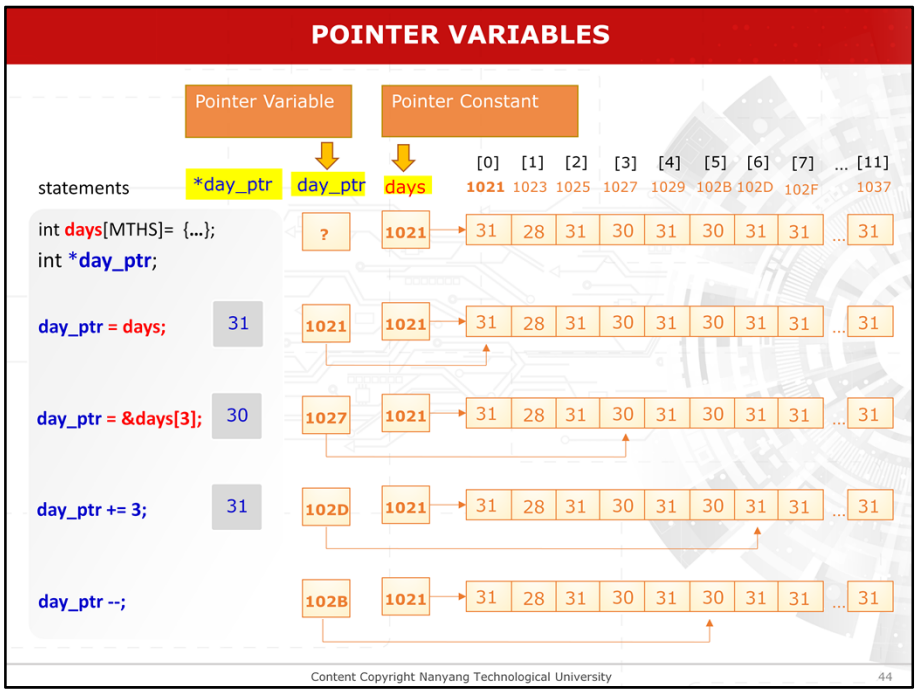
The day pointer will move forward three elements. The day pointer contains the value of 102D which is the address of the seventh element of the array.



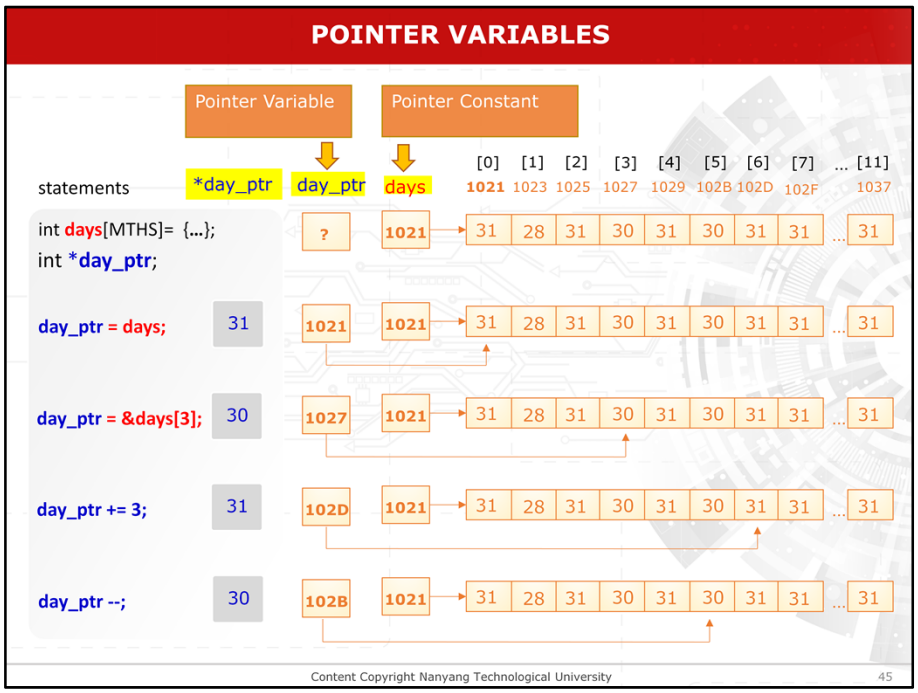
Asterisk day pointer will thus give the value of the seventh element, that is, 31.



The pointer variable can also be decremented as shown.



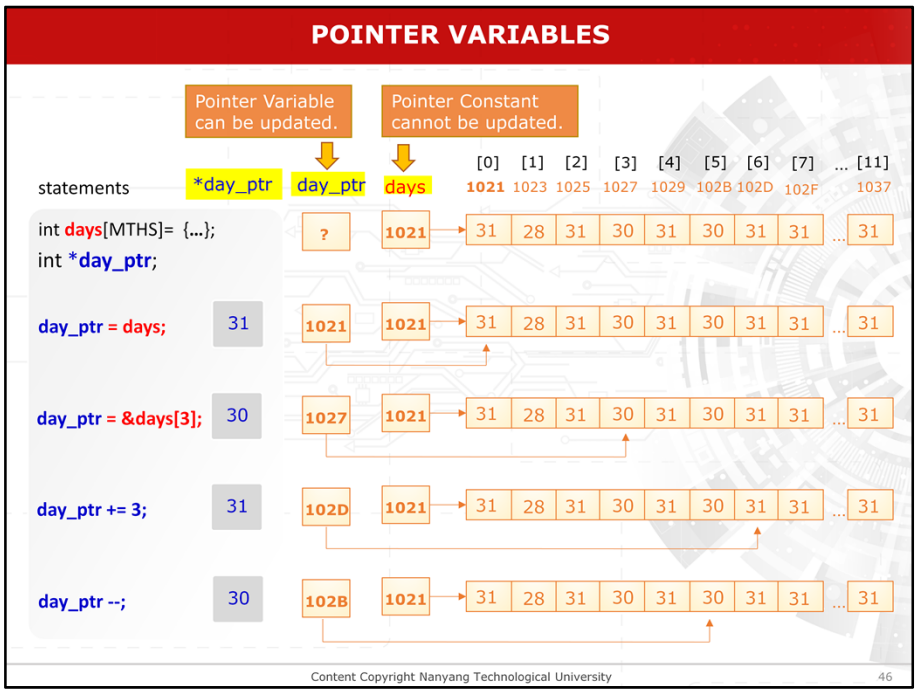
The day pointer move back 1 element in the array. It points to the sixth element of the array.



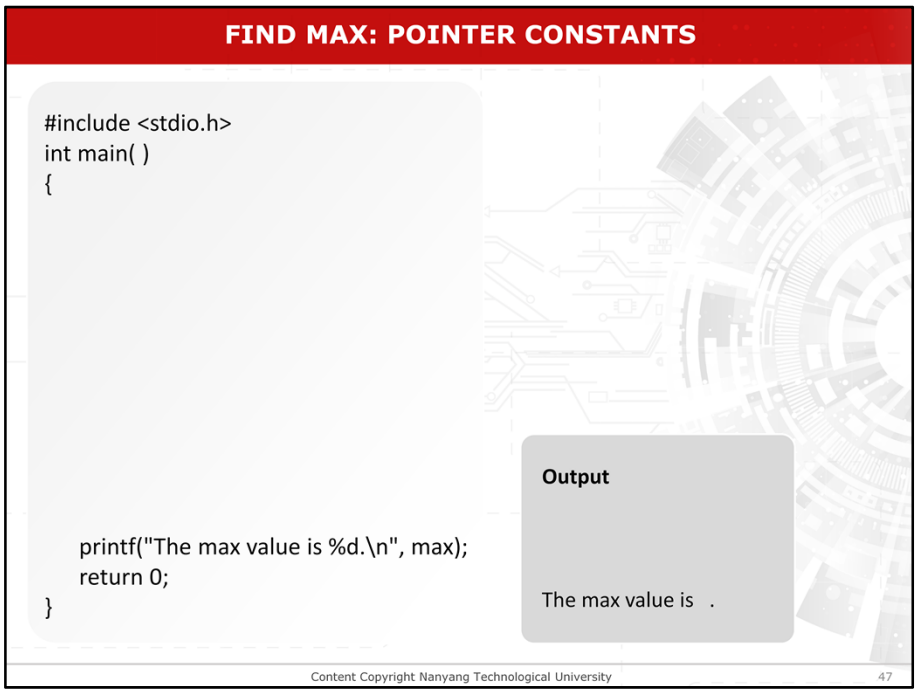
Asterisk day pointer will give the value of the sixth element, that is, 30.

When we perform pointer arithmetic, it is carried out according to the size of the data object that the pointer refers to. If **day pointer** is declared as a pointer variable of type **integer**, then every two bytes (assuming that **integer** takes 2 bytes) will be added for every increment of one.

Therefore, after assigning the array variable to the pointer variable, we can either use the array variable **days** to access each element of the array, or we can use the pointer variable **day pointer** to access each element. As such, there are two possible ways to process an array: (1) use the array variable directly, or (2) use a pointer variable and assign the array variable to the pointer variable.



However, note that the array variable cannot be changes as it is a pointer constant. Pointer Constant cannot be updated but pointer variable can be updated.



In this program, it shows the use of array variable (that is, pointer constant) to access each element of the array to find the maximum number from an array.

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf(“Enter 10 numbers: \n”);




    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

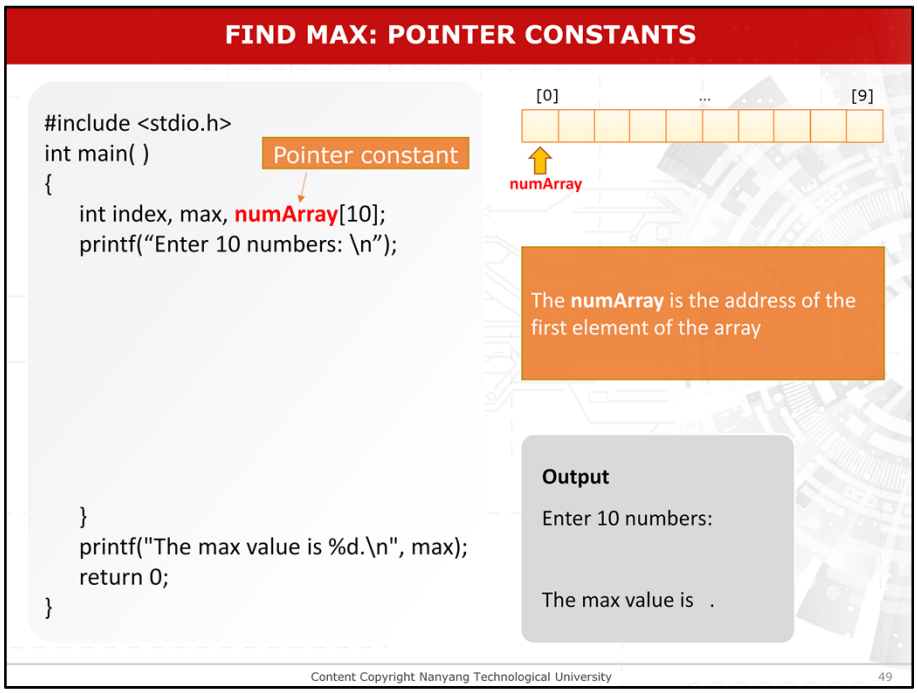
The program first reads in 10 integers from the user and stores them into the array variable **numArray**.

Output

Enter 10 numbers:

The max value is .

The program first reads in 10 integers from the user and stores them into the array variable **num Array**.



The **num Array** is the address of the first element of the array

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf("Enter 10 numbers: \n");
    for (index = 0;  index < 10;  index++)
        scanf("%d", numArray + index);




    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

numArray + index is the address of element **numArray[index]**

Output

Enter 10 numbers:

The max value is .

Num Array + index is the address of element **num Array [index]**

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf("Enter 10 numbers: \n");
    for (index = 0;  index < 10;  index++)
        scanf("%d", numArray + index);




    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

numArray + index is the address of element **numArray[index]**

Output

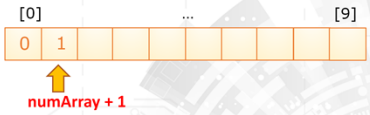
Enter 10 numbers:
0

The max value is .

As the user enter the numbers, they are stored in the num Array accordingly.

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf("Enter 10 numbers: \n");
    for (index = 0;  index < 10;  index++)
        scanf("%d", numArray + index);




    }
    printf("The max value is %d.\n", max);
    return 0;
}
```



numArray + index is the address of element **numArray[index]**

Output

Enter 10 numbers:
0 1

The max value is .

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf(“Enter 10 numbers: \n”);
    for (index = 0;   index < 10;   index++)
          scanf("%d", numArray + index);




    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

numArray + index is the address of element **numArray[index]**

Output

Enter 10 numbers:
0 1 2

The max value is .

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf("Enter 10 numbers: \n");
    for (index = 0;  index < 10;  index++)
        scanf("%d", numArray + index);




    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

numArray + index is the address of element **numArray[index]**

Output

Enter 10 numbers:
0 1 2 39

The max value is .

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf(“Enter 10 numbers: \n”);
    for (index = 0;  index < 10;  index++)
          scanf("%d", numArray + index);




    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

numArray + index is the address of element **numArray[index]**

Output

Enter 10 numbers:
0 1 2 39 4 5 6 7 8 9

The max value is .

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf(“Enter 10 numbers: \n”);
    for (index = 0;   index < 10;   index++)
          scanf("%d", numArray + index);




    printf("The max value is %d.\n", max);
    return 0;
}
```


```
for (index = 0; index < 10; index++)
    scanf(“%d”, &numArray[index]);
```

In addition, you may also use the array notation such as **numArray[index]** to access directly each element of the array.

In addition, you may also use the array notation such as **num Array [index]** to access directly each element of the array.

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf(“Enter 10 numbers: \n”);
    for (index = 0;   index < 10;   index++)
          scanf("%d", numArray + index);




    printf("The max value is %d.\n", max);
    return 0;
}
```


```
for (index = 0; index < 10; index++)
    scanf(“%d”, &numArray[index]);
```

In addition, you may also use the array notation such as **numArray[index]** to access directly each element of the array.

Note that the address operator (**&**) is needed in the **scanf()** statement.

Note that the address operator (**ampersand**) is needed in the **scanf()** statement.

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf(“Enter 10 numbers: \n”);
    for (index = 0;  index < 10;  index++)
          scanf("%d", numArray + index);
    // Find maximum from array data
    max = *numArray;

    printf("The max value is %d.\n", max);
    return 0;
}
```

Output

Enter 10 numbers:
0 1 2 39 4 5 6 7 8 9

The max value is .

Variable max is first define with the value of the first element in num Array. This value will then be systematically compared with the value in the rest of the array to determine the maximum from the array data.

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf("Enter 10 numbers: \n");
    for (index = 0;  index < 10;  index++)
        scanf("%d", numArray + index);
    // Find maximum from array data
    max = *numArray;
    for (index = 1;  index < 10;  index++)
    {
        if (*(numArray + index) > max)
        max = *(numArray + index);
    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

The **for** loop accesses each element of the array, and compares it with **max** in order to determine the maximum value. The maximum value is then assigned to **max**.

Output

Enter 10 numbers:
0 1 2 39 4 5 6 7 8 9

The max value is 39.

The **for** loop accesses each element of the array, and compares it with **max** in order to determine the maximum value. The maximum value is then assigned to **max**.

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf("Enter 10 numbers: \n");
    for (index = 0;  index < 10;  index++)
        scanf("%d", numArray + index);
    // Find maximum from array data
    max = *numArray;
    for (index = 1;  index < 10;  index++)
    {
        if (*(numArray + index) > max)
        max = *(numArray + index);
    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

***(numArray+index)** is the value of the element **numArray[index]**.

Output

Enter 10 numbers:
0 1 2 39 4 5 6 7 8 9

The max value is 39.

Asterisk (num Array + index) is the value of the element **num Array [index]**.

FIND MAX: POINTER CONSTANTS

```
#include <stdio.h>
int main( )
{
    int index, max, numArray[10];
    printf("Enter 10 numbers: \n");
    for (index = 0;  index < 10;  index++)
        scanf("%d", numArray + index);
    // Find maximum from array data
    max = *numArray;
    for (index = 1;  index < 10;  index++)
    {
        if (*(numArray + index) > max)
        max = *(numArray + index);
    }
    printf("The max value is %d.\n", max);
    return 0;
}
```

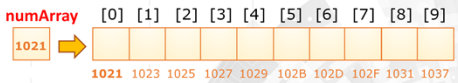
In addition, you may also use the array notation such as **numArray[index]** to access directly each element of the array.

```
max = numArray[0];
for (index = 1;  index < 10; index++)
{
    if (numArray[index] > max)
        max = numArray[index];
}
```

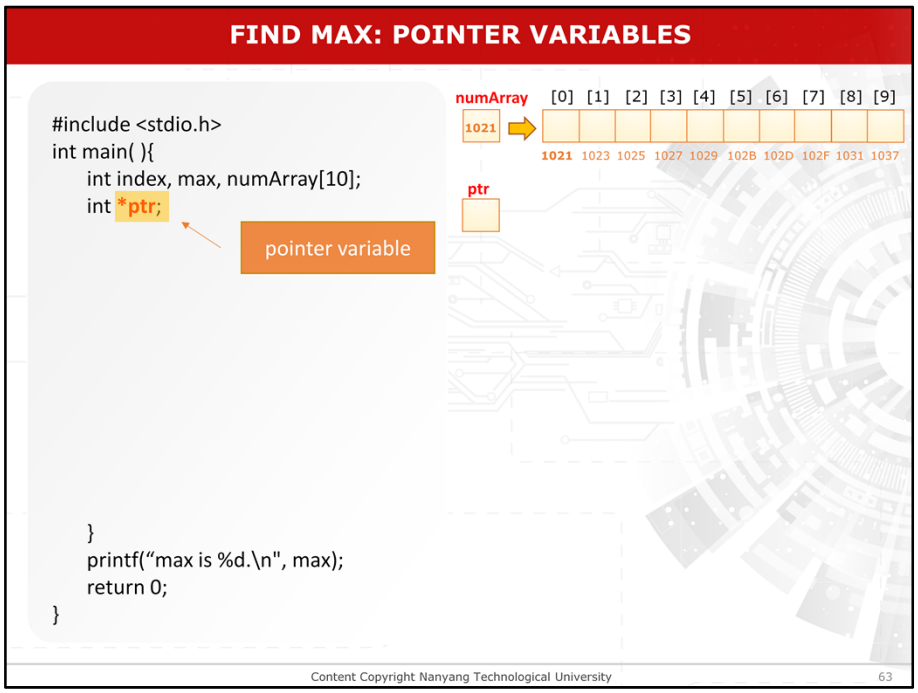

In addition, you may also use the array notation such as **num Array [index]** to access directly each element of the array.

FIND MAX: POINTER VARIABLES

```
#include <stdio.h>
int main( ){
    int index, max, numArray[10];









    }
    printf("max is %d.\n", max);
    return 0;
}
```



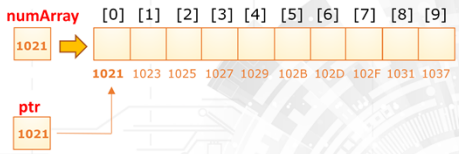
The previous program uses the pointer constant **num Array** to access all the elements of the array. Another way to access the elements of an array is to use a pointer variable.



This program gives an example using a pointer variable to find the maximum element of the array.

FIND MAX: POINTER VARIABLES

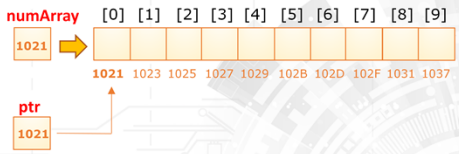
```
#include <stdio.h>
int main( ){
    int index, max, numArray[10];
    int *ptr;
    ptr = numArray;

    }
    printf("max is %d.\n", max);
    return 0;
}
```



To achieve this, it is important to assign **num Array** to **pointer** with the statement **pointer = num Array;**

FIND MAX: POINTER VARIABLES

```
#include <stdio.h>
int main( ){
      int index, max, numArray[10];
      int *ptr;
      ptr = numArray;
      printf("Enter 10 numbers: \n");
      for (index = 0; index < 10; index++)
            scanf("%d", ptr++);




      }
      printf("max is %d.\n", max);
      return 0;
}
```



Output

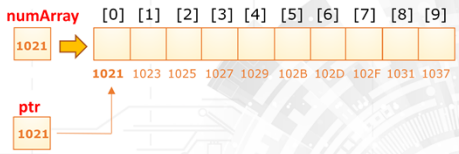
Enter 10 numbers:

The max value is .

After that, we can read in the array data via the pointer variable **pointer**.

FIND MAX: POINTER VARIABLES

```
#include <stdio.h>
int main( ){
      int index, max, numArray[10];
      int *ptr;
      ptr = numArray;
      printf("Enter 10 numbers: \n");
      for (index = 0; index < 10; index++)
            scanf("%d", ptr++);




      }
      printf("max is %d.\n", max);
      return 0;
}
```



Output

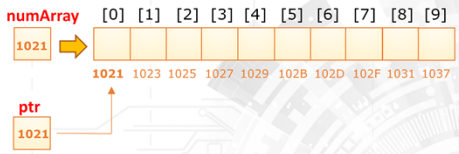
Enter 10 numbers:

The max value is .

In the **for** loop, we use **scanf()** to read in user input.

FIND MAX: POINTER VARIABLES

```
#include <stdio.h>
int main( ){
      int index, max, numArray[10];
      int *ptr;
      ptr = numArray;
      printf("Enter 10 numbers: \n");
      for (index = 0; index < 10; index++)
            scanf("%d", ptr++);




      }
      printf("max is %d.\n", max);
      return 0;
}
```



Output

Enter 10 numbers:

The max value is .

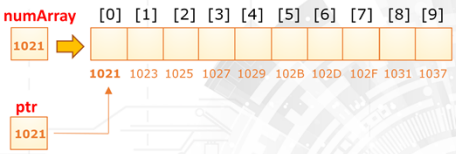
We increment the **pointer** as

pointer + +;

to access each element of the array in order to store the input integer into the corresponding index location of the array.

FIND MAX: POINTER VARIABLES

```
#include <stdio.h>
int main( ){
      int index, max, numArray[10];
      int *ptr;
      ptr = numArray;
      printf(“Enter 10 numbers: \n”);
      for (index = 0; index < 10; index++)
            scanf("%d", ptr++);




      }
      printf(“max is %d.\n", max);
      return 0;
}
```




```
for (index = 0; index < 10; index++)
    scanf(“%d”, &numArray[index]);
```

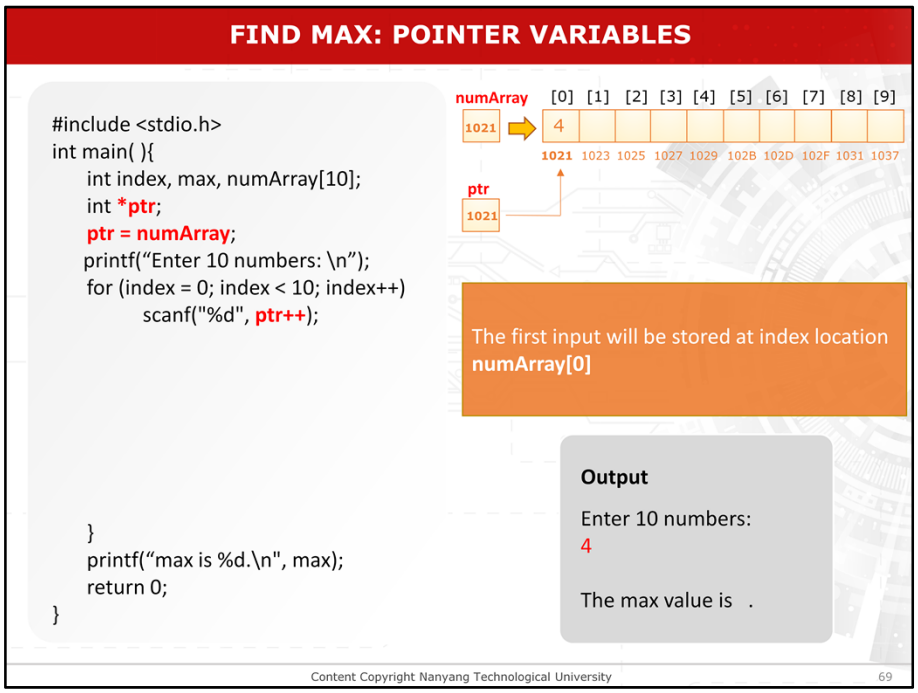
Using index for reading input

Output

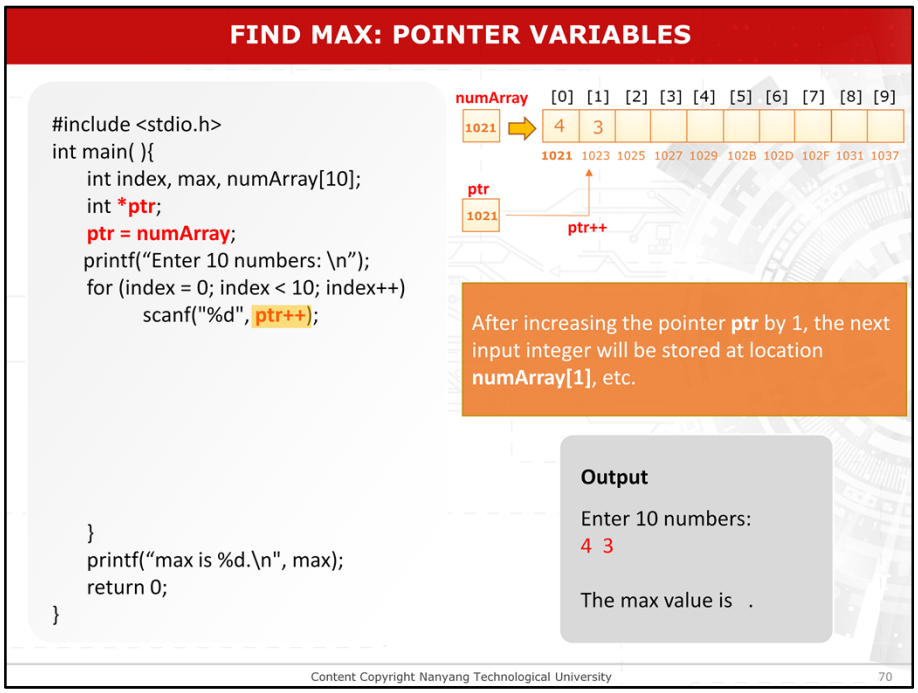
Enter 10 numbers:

The max value is .

The equivalent statements of using index for reading input are as shown.



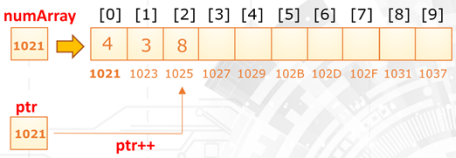
The first input will be stored at index location **num Array [0]**



after increasing the pointer **pointer** by 1, the next input integer will be stored at location **num Array [1]**, etc.

FIND MAX: POINTER VARIABLES

```
#include <stdio.h>
int main( ){
      int index, max, numArray[10];
      int *ptr;
      ptr = numArray;
      printf("Enter 10 numbers: \n");
      for (index = 0; index < 10; index++)
            scanf("%d", ptr++);




      }
      printf("max is %d.\n", max);
      return 0;
}
```



After increasing the pointer **ptr** by 1, the next input integer will be stored at location **numArray[1]**, etc.

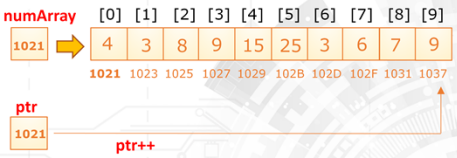
Output

Enter 10 numbers:
4 3 8

The max value is .

FIND MAX: POINTER VARIABLES

```
#include <stdio.h>
int main( ){
      int index, max, numArray[10];
      int *ptr;
      ptr = numArray;
      printf("Enter 10 numbers: \n");
      for (index = 0; index < 10; index++)
            scanf("%d", ptr++);




      }
      printf("max is %d.\n", max);
      return 0;
}
```

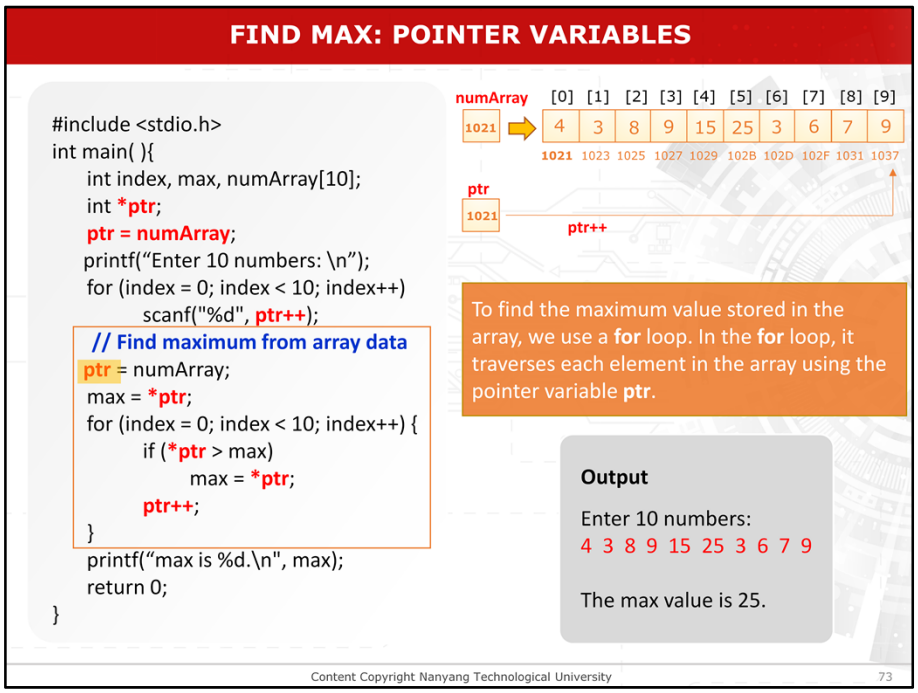


After increasing the pointer **ptr** by 1, the next input integer will be stored at location **numArray[1]**, etc.

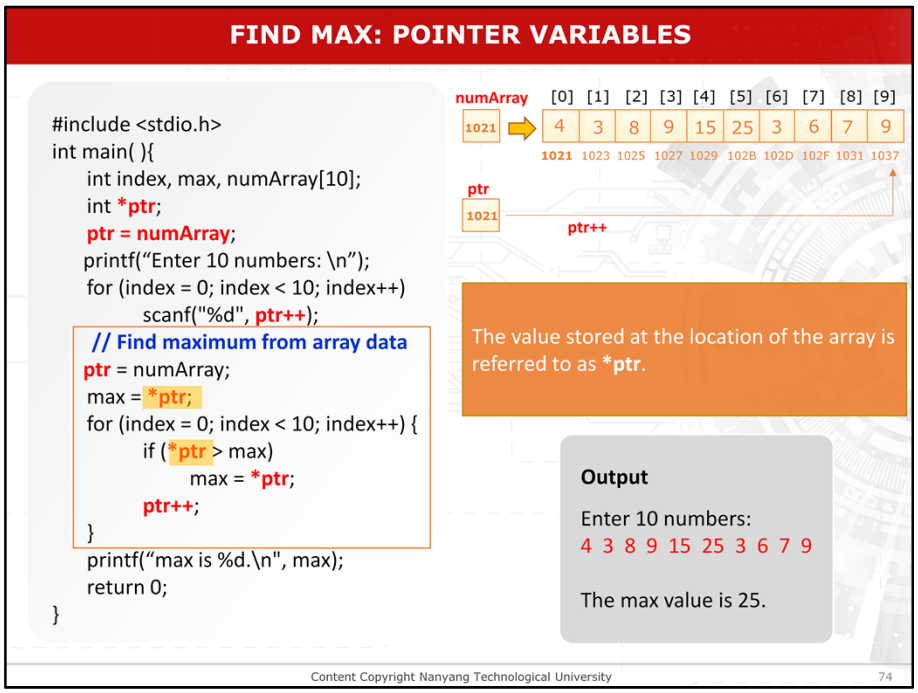
Output

Enter 10 numbers:
4 3 8 9 15 25 3 6 7 9

The max value is .



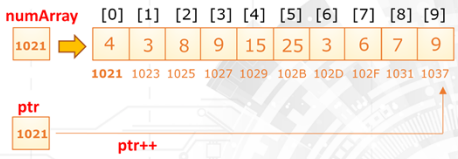
To find the maximum value stored in the array, we use a **for** loop. In the **for** loop, it traverses each element in the array using the pointer variable **pointer**.



The value stored at the location of the array is referred to as asterisk pointer

FIND MAX: POINTER VARIABLES

```
#include <stdio.h>
int main( ){
    int index, max, numArray[10];
    int *ptr;
    ptr = numArray;
    printf("Enter 10 numbers: \n");
    for (index = 0; index < 10; index++)
        scanf("%d", ptr++);
    // Find maximum from array data
    ptr = numArray;
    max = *ptr;
    for (index = 0; index < 10; index++) {
        if (*ptr > max)
            max = *ptr;
        ptr++;
    }
    printf("max is %d.\n", max);
    return 0;
}
```



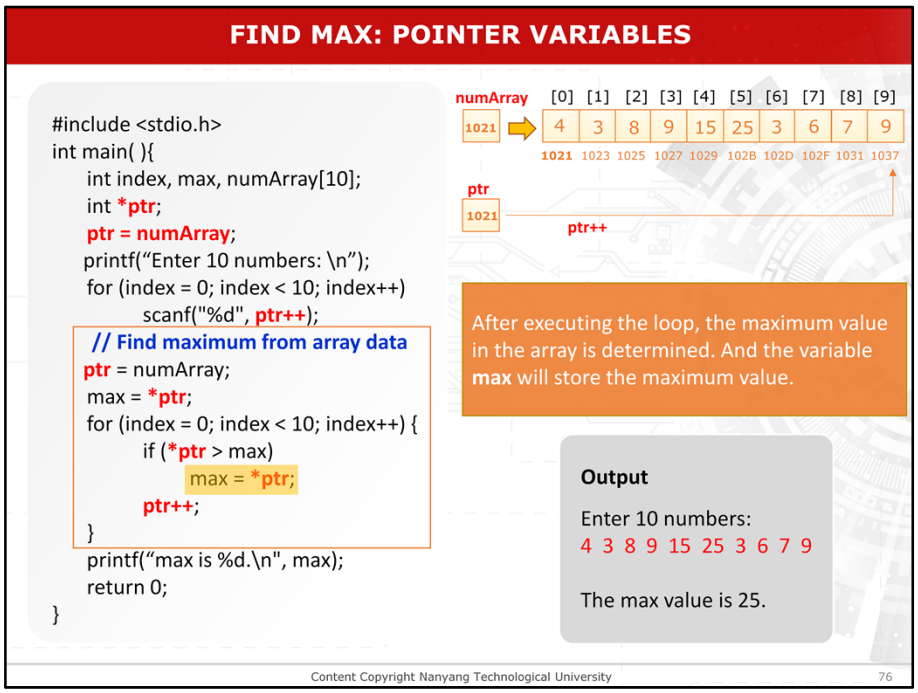
The content of each element of the array is compared with the current maximum value.

Output

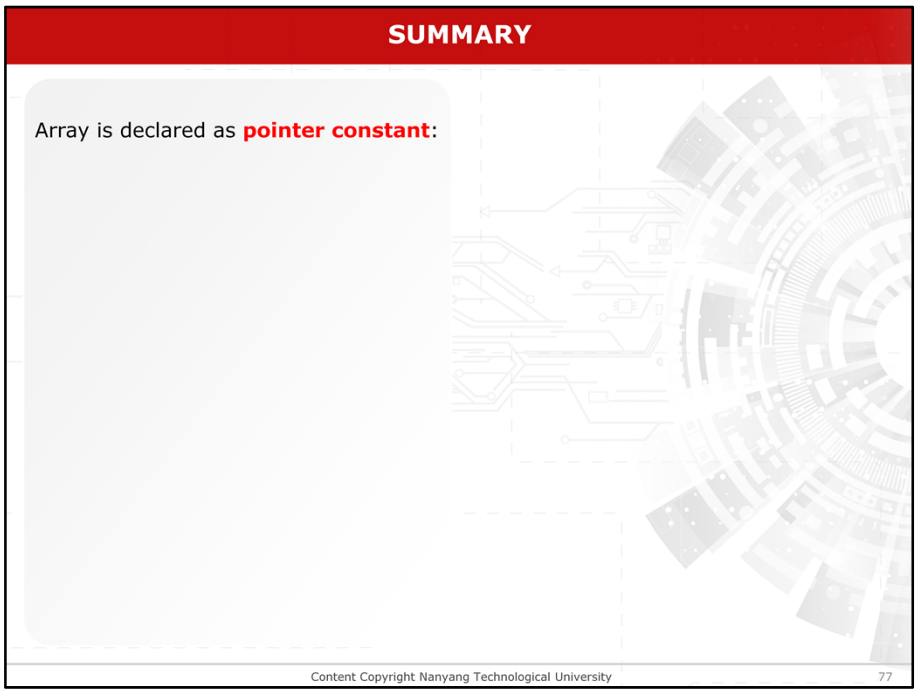
Enter 10 numbers:
4 3 8 9 15 25 3 6 7 9

The max value is 25.

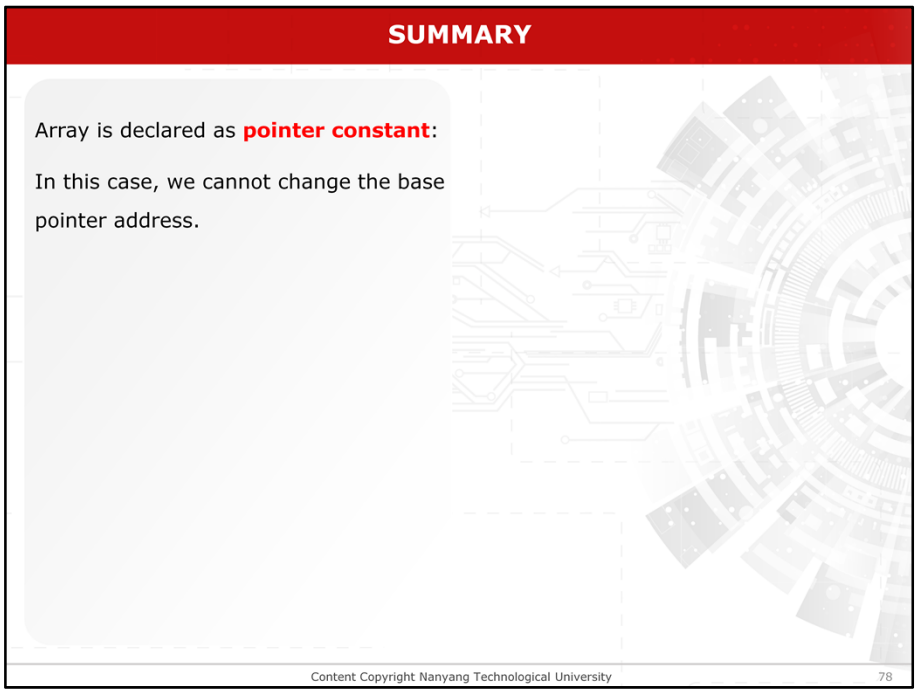
The content of each element of the array is compared with the current maximum value.



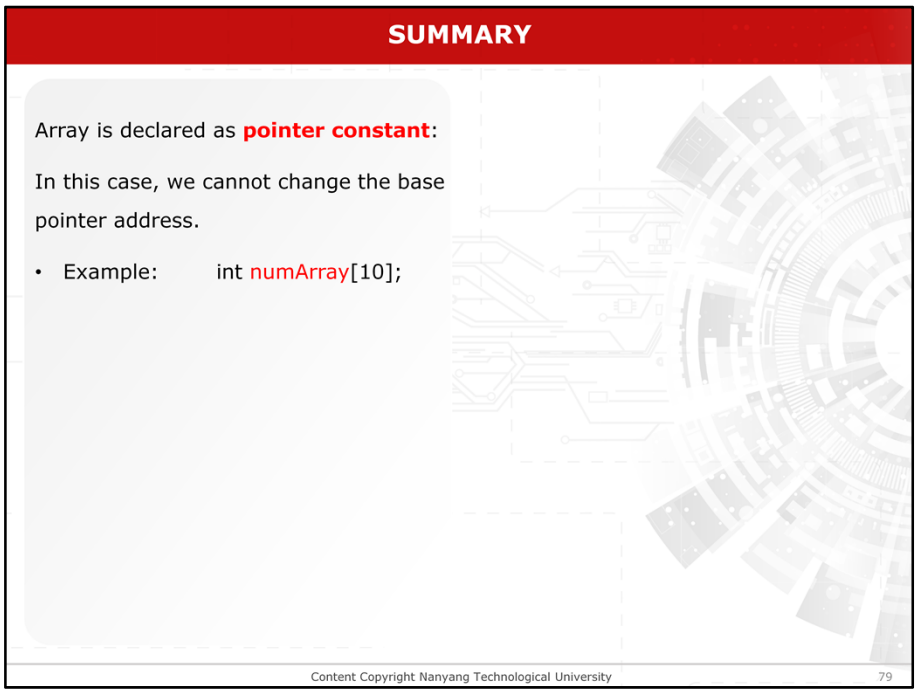
After executing the loop, the maximum value in the array is determined. And the variable **max** will store the maximum value.



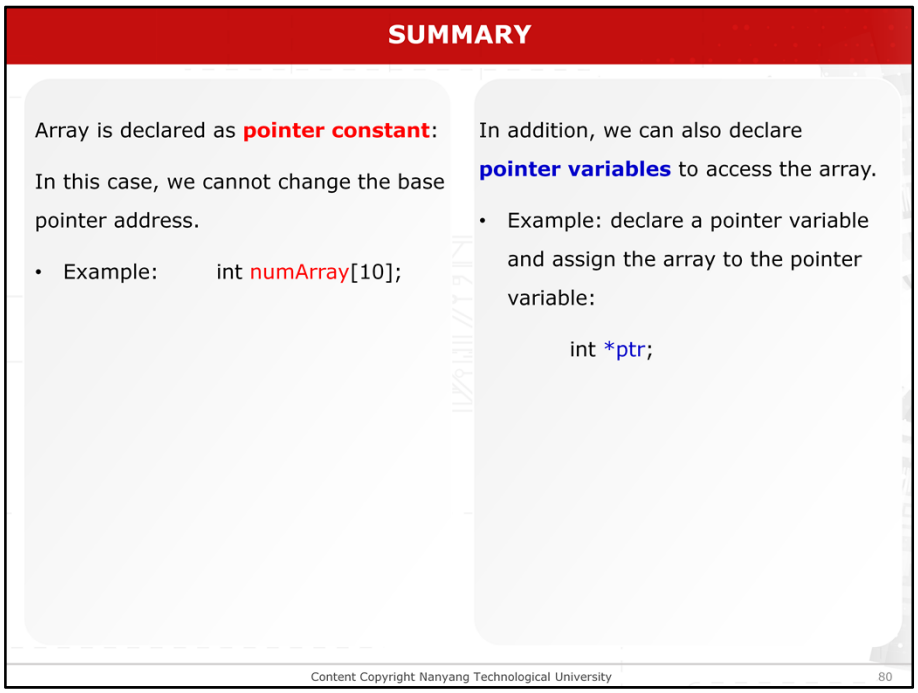
Array is declared as pointer constant.



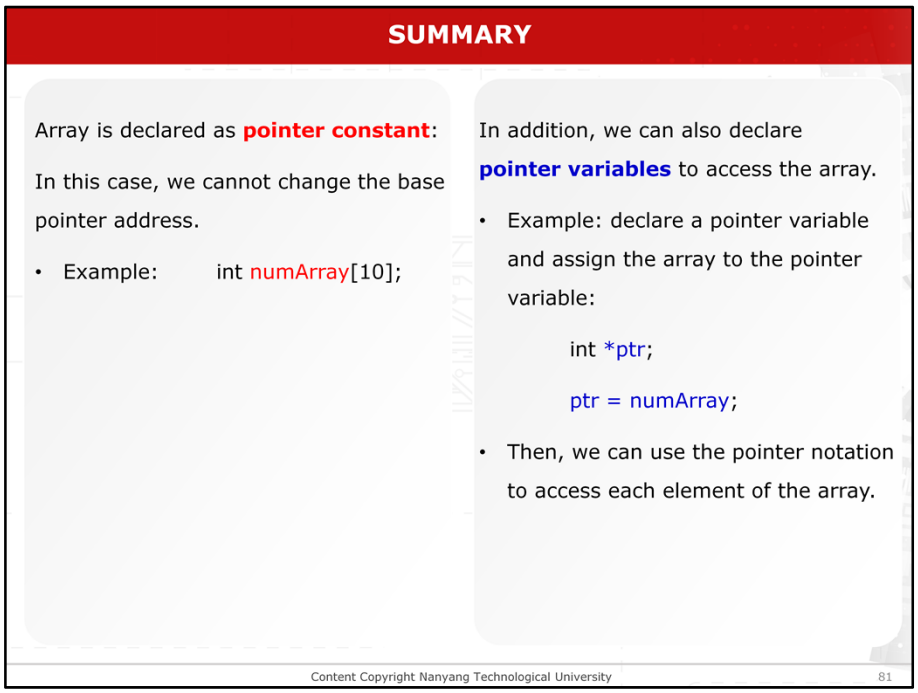
For pointer constant declaration, the base pointer address stored in the array variable cannot be changed.



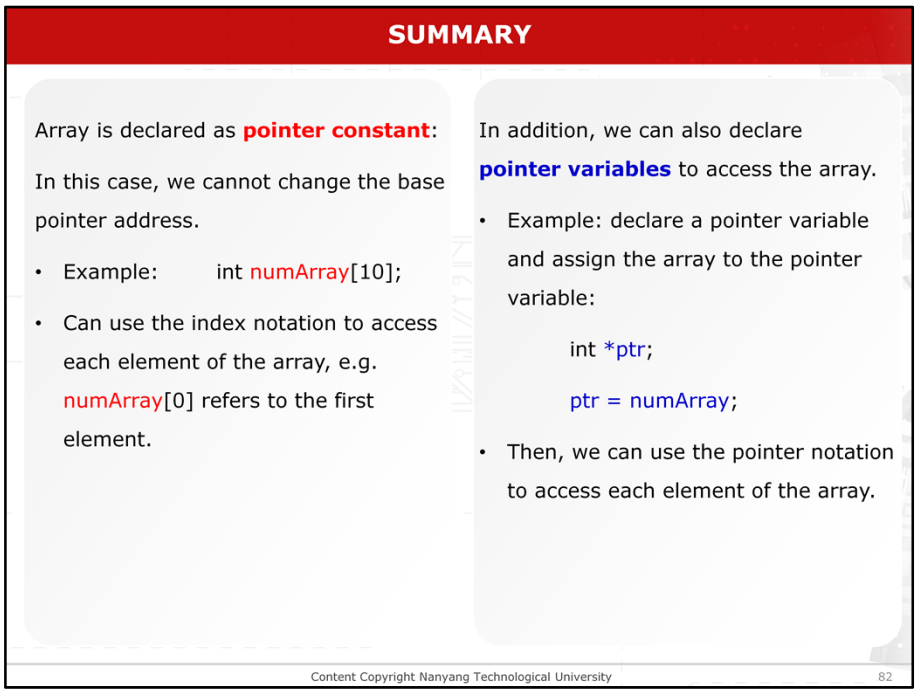
For example, integer **num Array** [10]



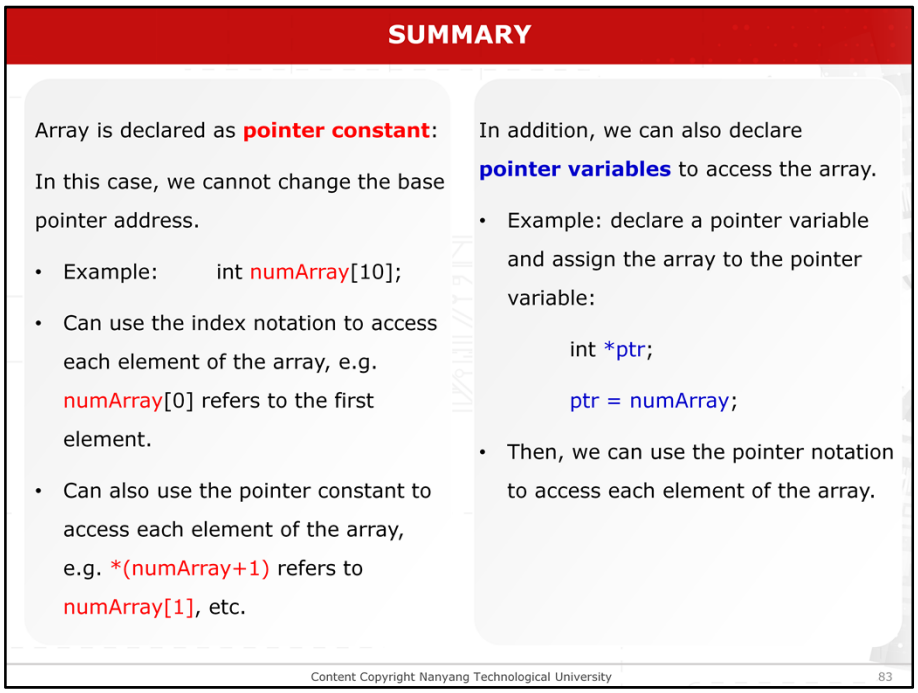
In addition, we can also declare **pointer variables** to access the array. For example, integer asterisk **pointer**;



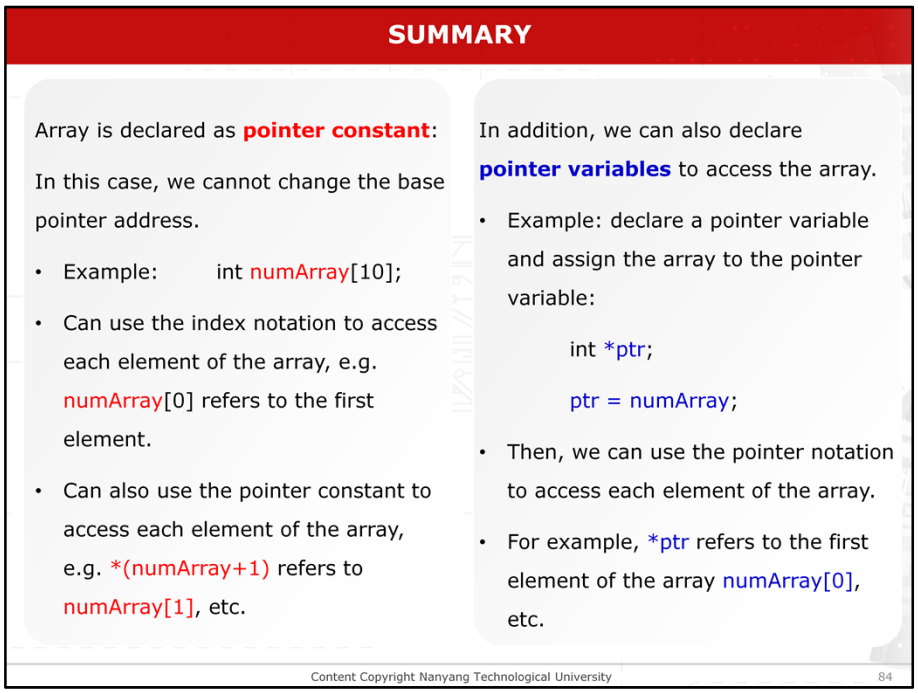
we can then assign the pointer variable with the array address, that is, **pointer = num Array**; we can then use the pointer variable to access each element of the array.
both the use of array notation and pointer variable can be adopted for accessing individual elements of an array.



Using array notation: for example, **num Array [index]**



Using pointer constant: for example asterisk **(num Array + index)**



Using pointer variable: for example **asterisk pointer**
However, the use of pointer variable will be more efficient than the array notation, and it is also more convenient when working with strings.