

Lesson 8.1:

In addition to handling numerical data, programs are also required to deal with alphabetic data. Strings are arrays of characters. C libraries provide a number of functions for performing operations on strings. In this chapter, string constants and string variables are first introduced. The different commonly used string functions from C libraries are discussed.

OVERVIEW

The following are the coverage for Character Strings:

- String Declaration, Initialization and Operations
- String Input and Output
- String Functions
- The ctype.h Character Functions
- String to Number Conversions
- Arrays of Character Strings

Content Copyright Nanyang Technological University

2

There are 6 main sections to cover for Character Strings as shown. This video lesson focuses on the first part of String Declaration, Initialization and Operations.

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain strings are arrays of characters

After this lesson, you should be able to:

Explain strings are arrays of characters

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain strings are arrays of characters
- Define a string

Define a string

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain strings are arrays of characters
- Define a string
- Define string constant

Define string constant

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain strings are arrays of characters
- Define a string
- Define string constant
- Describe how to declare a string variable

Describe how to declare a string variable

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain strings are arrays of characters
- Define a string
- Define string constant
- Describe how to declare a string variable
- Declare strings using array notation

Content Copyright Nanyang Technological University

7

Declare strings using the array notation

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain strings are arrays of characters
- Define a string
- Define string constant
- Describe how to declare a string variable
- Declare strings using array notation
- Declare strings using pointer notation

Declare strings using pointer notation

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain strings are arrays of characters
- Define a string
- Define string constant
- Describe how to declare a string variable
- Declare strings using array notation
- Declare strings using pointer notation
- Explain the difference in the declaration of string variables that use array or pointer

Explain the difference in the declaration of string variables that use array or pointer

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain strings are arrays of characters
- Define a string
- Define string constant
- Describe how to declare a string variable
- Declare strings using array notation
- Declare strings using pointer notation
- Explain the difference in the declaration of string variables that use array or pointer
- Apply string declaration

Content Copyright Nanyang Technological University

10

Apply string declaration

CHARACTER STRINGS

Definition String

A **string** is an array of characters terminated by a **NULL** character ('\0').

Content Copyright Nanyang Technological University

11

A string is an array of characters terminated by a null character.

CHARACTER STRINGS

Definition String

A **string** is an array of characters terminated by a **NULL** character ('\0').

Example: String "abcdef12345OK"

The diagram illustrates a string as an array of characters. It shows a row of 13 empty squares, each representing a character cell in memory. An orange bracket underneath the first 12 squares is labeled 'array', indicating that these 12 squares together form the string "abcdef12345".

Content Copyright Nanyang Technological University 12

For example, the string double quote a b c d e f 1 2 3 4 5 O K double quote is actually stores as characters in array

CHARACTER STRINGS

Definition **String**

A **string** is an array of characters terminated by a **NULL** character ('\0').

Example: String "abcdef12345OK"

Null character terminates string

a | b | c | d | e | f | 1 | 2 | 3 | 4 | 5 | O | K |

array

Content Copyright Nanyang Technological University

13

terminated by a **NULL** character.

STRING CONSTANTS

String constant

String constant is a set of characters in double quotes.

Content Copyright Nanyang Technological University

14

String constant is a set of characters in double quotes.

STRING CONSTANTS

String constant

String constant is a set of characters in double quotes.

Example: "**C Programming**" - is an array of characters and automatically terminated with the **null** character '**\0**'

The diagram illustrates a string constant "C Programming" as an array of characters. The characters are represented in boxes: C, P, r, o, g, r, a, m, m, i, n, g, \0. A bracket below the first eleven characters is labeled "array". An orange arrow points to the character "\0", which is labeled "Null character terminates string".

Content Copyright Nanyang Technological University 15

For example, the string **double quote C Programming double quote** is an array of characters and automatically terminated with the **null** character.

When the compiler encounters a string constant, it allocates space for each individual character of the string and adds a terminating null character at the end of the string.

STRING CONSTANTS

String constant

Using `#define` to define a string constant.

Content Copyright Nanyang Technological University 16

We can use **pound define** to define string constants.

STRING CONSTANTS

String constant

Using #define to define a string constant.

Example: `#define NTU "Nanyang Technological Uni."`

Null character terminates string

N a n y a n g | T e c h n o l o g i c a l | U n i . \0

Content Copyright Nanyang Technological University 17

For example,

```
pound define NTU double quote Nanyang Technological University double
quote
```

STRING CONSTANTS

String constant

Using #define to define a string constant.

Example: `#define NTU "Nanyang Technological Uni."`

The diagram shows a horizontal array of characters representing the string "Nanyang Technological Uni.". The array consists of 19 cells, each containing a single character: N, a space, a n, a y, a n, a g, a space, T, e, c, h, n, o, l, o, g, i, c, a, l, a space, U, n, i, a space, and \0. An orange box labeled "NTU" has a red arrow pointing to the first cell of the array. A bracket below the array is labeled "array". An orange arrow points from the text "Null character terminates string" to the last cell containing "\0".

Content Copyright Nanyang Technological University 18

A pointer pointing to its first character is returned.

A string constant is essentially a pointer to the first character of an array of characters.

STRING CONSTANTS

String constant

Used in function arguments, e.g. printf() and puts()

Examples:

```
printf("Hello, how are you?");  
printf("Welcome to Programming with C.\n");  
puts("Welcome to Programming with C.");
```

Content Copyright Nanyang Technological University

19

String constants can also be used as function arguments for **print f** and **put s** functions:

STRING CONSTANTS

String constant

Character constant

'X'

Content Copyright Nanyang Technological University 20

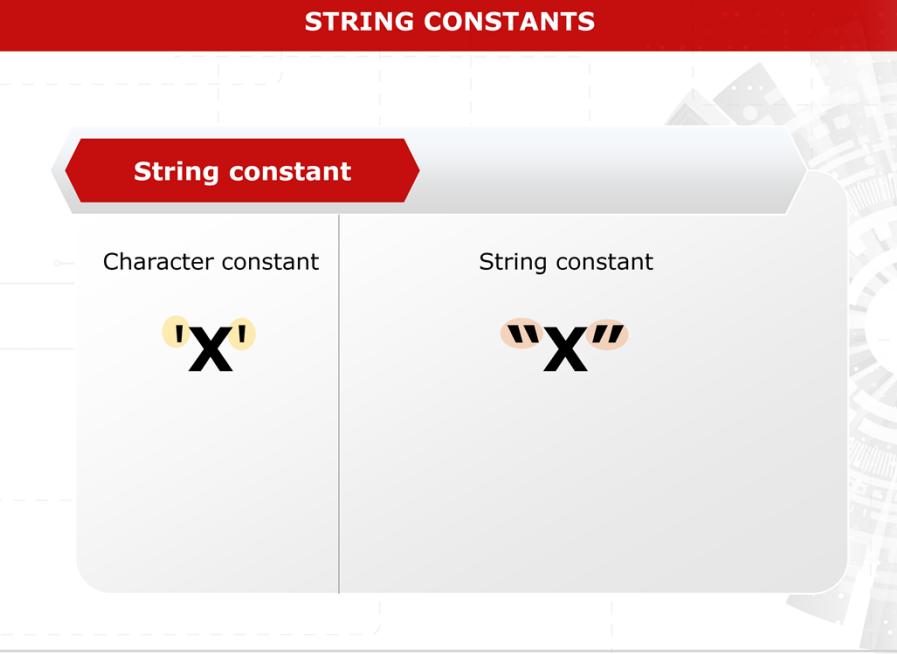
It is important to distinguish the difference between the character constant single quote X single quote

STRING CONSTANTS

String constant

Character constant <code>'X'</code>	String constant <code>"X"</code>
--	-------------------------------------

Content Copyright Nanyang Technological University 21

The diagram is titled "STRING CONSTANTS" at the top. Below it, a red arrow points right and contains the text "String constant". Underneath this, there are two columns. The left column is labeled "Character constant" and contains the code "X" enclosed in single quotes. The right column is labeled "String constant" and contains the code "X" enclosed in double quotes. The background of the slide features a faint watermark of a building.

and a string constant double quote X double quote

STRING CONSTANTS

String constant

Character constant 'X' single character of data type char	String constant "X"
---	-------------------------------

Content Copyright Nanyang Technological University 22

The character constant single quote X single quote consists of a single character of data type **character**

STRING CONSTANTS

String constant

Character constant	String constant
'X'	"X"
single character of data type char	character string constant "X" consists of two characters in an array of type char
[X]	[X] [\0]

Content Copyright Nanyang Technological University 23

while the character string constant double quote X double quote consists of two characters, that is, the character X and the null character and is an array of type **character**.

STRING VARIABLES: DECLARATION

String variable is declared using the **pointer notation**.

```
char *str = "C Programming";
```

String constant = array of char

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
C		P	r	o	g	r	a	m	m	i	n	g	\0

str

Null character terminates string

Content Copyright Nanyang Technological University

24

Typically, we can declare a string variable by assigning the pointer of the string constant to a **pointer variable**:

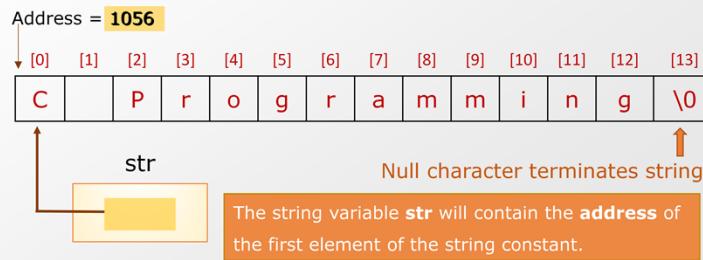
```
character asterisk string = double quote C Programming double quote;
```

STRING VARIABLES: DECLARATION

String variable is declared using the **pointer notation**.

```
char *str = "C Programming";
```

String constant = array of char



Content Copyright Nanyang Technological University

25

The string variable **string** will then contain the **address** of the first element of the string constant.

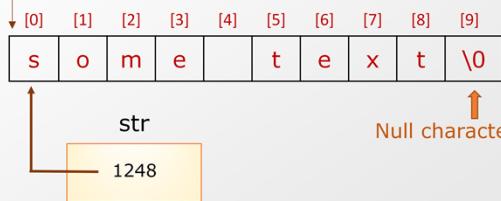
STRING VARIABLES: DECLARATION

String variable can also declared using the **array notation**.

Example 1: `char str[] = "some text"; // string`

String constant = array of char

Address = 1248



Content Copyright Nanyang Technological University

26

A string is an array of characters. Therefore, we can also define strings using the **array notation**, in addition to the pointer notation. To declare strings using array notation, we can use the following examples:

character array string = double quote some text double quote;

STRING VARIABLES: DECLARATION

String variable can also be declared using the **array notation**.

Example 1: `char str[] = "some text"; // string`

String constant = array of char

Address = 1248



str

1248

Null character terminates string

Null character '\0' is automatically added at end of the sequence of characters.

Content Copyright Nanyang Technological University

27

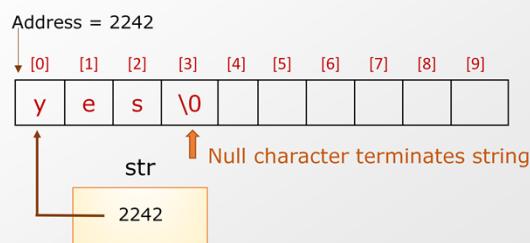
A null character is automatically added at end of the sequence of characters to make variable string a string.

STRING VARIABLES: DECLARATION

String variable can also be declared using the **array notation**.

Example 2: `char str[10] = "yes"; // legal`

String constant = array of char



Content Copyright Nanyang Technological University

28

Example 2

character array of size 10, string = double quote yes double quote;

The array **string** is allocated space to hold up to 10 elements.

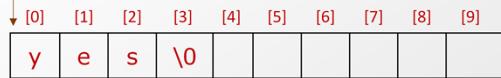
STRING VARIABLES: DECLARATION

String variable can also be declared using the **array notation**.

Example 2: `char str[10] = "yes"; // legal`

String constant = array of char

Address = 2242



Content Copyright Nanyang Technological University

29

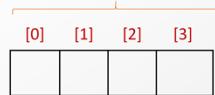
The array has sufficient space to hold the characters y, e, s and a null character. So this is a legal declaration.

STRING VARIABLES: DECLARATION

String variable can also declared using the **array notation**.

Example 3: `char str[4] = "four"; // incorrect`

Array declared can only hold up to 4 elements



Content Copyright Nanyang Technological University

30

Example 3

character array of size 4, string = double quote four double quote;

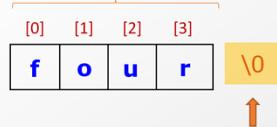
The array **string** is allocated space to hold up to only 4 elements.

STRING VARIABLES: DECLARATION

String variable can also declared using the **array notation**.

Example 3: `char str[4] = "four"; // incorrect`

Array declared can only hold up to 4 elements



No null character included means no string declared.

Null character **cannot** be included in declared array due to insufficient space declared therefore array is not able to contain string

Content Copyright Nanyang Technological University

31

However, the null character **cannot** be included in declared array due to insufficient space declared. Therefore, array is not able to contain string.

The **string** array only allocates 4 elements to hold its data, which is not enough as it needs an additional element to hold the null character that is automatically added at the end of the sequence of characters.

STRING VARIABLES: DECLARATION

String variable can also declared using the **array notation**.

Example 4: `char str[] = {'a', 'b', 'c', '\0'}; // string`

Array of 4 elements of type char is created



Content Copyright Nanyang Technological University

32

We can also create strings as follows:

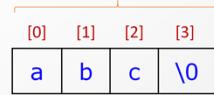
character array string = open curly bracket, single quote a single quote, single quote b single quote, single quote c single quote and single quote null character single quote, close curly bracket;

STRING VARIABLES: DECLARATION

String variable can also be declared using the **array notation**.

Example 4: `char str[] = {"a", "b", "c", "\0"}; // string`

Array of 4 elements of type char is created



Tedious to initialise array by listing all the characters in curly brackets and single quote each character.

Content Copyright Nanyang Technological University

33

However, it is tedious to initialize array by listing all the characters in curly brackets and single quote each character

STRING VARIABLES: DECLARATION

String variable can also be declared using the **array notation**.

Example 4: `char str[] = {"a", "b", "c", '\0'}; // string`

Array of 4 elements of type char is created

[0]	[1]	[2]	[3]
a	b	c	\0

Can simply enclose the characters
using double quotes as if they were
a string constant.

Content Copyright Nanyang Technological University

34

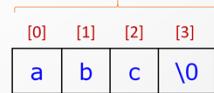
We can simply enclose the characters using double quotes as if they were a string constant.

STRING VARIABLES: DECLARATION

String variable can also declared using the **array notation**.

Example 4: `char str[] = { 'a', 'b', 'c', '\0' } ; // string`

Array of 4 elements of type char is created



Equivalent to

`char str[] = "abc";`

Content Copyright Nanyang Technological University

35

This is equivalent to

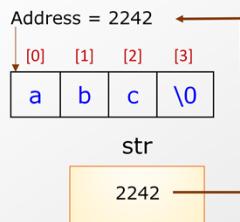
character array string = double quote a b c double quote;

STRING VARIABLES

Therefore, just like other kinds of arrays, the array name **str** gives the **address** of the 1st element of the array.

Example: `char str[] = "abc"; // string`

Array of 4 elements of type char is created



Content Copyright Nanyang Technological University

36

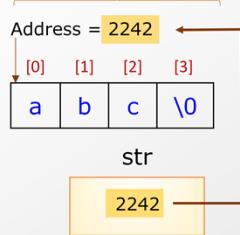
As a string constant returns a pointer to its first character, the array name **string** contains the address of the first element of the array as other kinds of arrays

STRING VARIABLES

Therefore, just like other kinds of arrays, the array name **str** gives the **address** of the 1st element of the array:

Example: `char str[] = "abc"; // string`

Array of 4 elements of type char is created



`str == &str[0] == 2242`

Content Copyright Nanyang Technological University

37

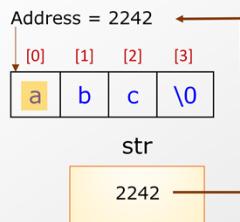
the **string** contains the address of the first element of the array of characters.

STRING VARIABLES

Therefore, just like other kinds of arrays, the array name **str** gives the **address** of the 1st element of the array:

Example: `char str[] = "abc"; // string`

Array of 4 elements of type char is created



```
str == &str[0] == 2242  
*str == str[0] == 'a'
```

Content Copyright Nanyang Technological University

38

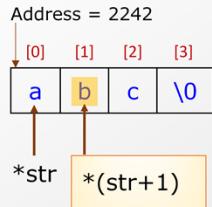
we can use **asterisk string** to retrieve the first element of the string.

STRING VARIABLES

Therefore, just like other kinds of arrays, the array name **str** gives the **address** of the 1st element of the array:

Example: `char str[] = "abc"; // string`

Array of 4 elements of type char is created



```
str == &str[0] == 2242  
*str == str[0] == 'a'  
*(str+1) == str[1] == 'b'
```

Content Copyright Nanyang Technological University

39

similarly, we can use **asterisk string +1** to retrieve the second element of the character string.

STRING VARIABLES

- As discussed earlier, there are two ways to declare a string:

```
char str1[ ] = "How are you?"; // using array declaration  
char *str2 = "How are you?"; // using pointer declaration
```

- Can you tell the difference between the two declarations?

It is important to distinguish the declaration of string variables that use array or pointer. For example, the following strings are defined as:

character array string 1 = double quote How are you? double quote;

character asterisk string 2 = double quote How are you? double quote;

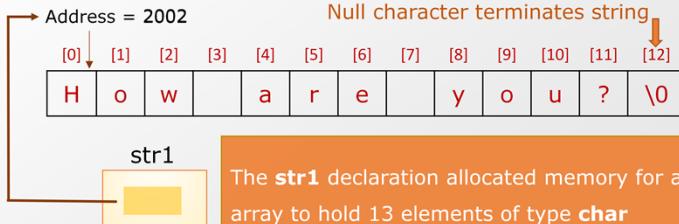
Question: Can you tell the difference between the two declarations?

STRING VARIABLES

Two ways to declare a string:

```
char str1[ ] = "How are you?"; // using array declaration  
char *str2 = "How are you?"; // using pointer declaration
```

Array has been allocated with memory to hold 13 elements



The **str1** declaration allocated memory for an array to hold 13 elements of type **char**

Content Copyright Nanyang Technological University

41

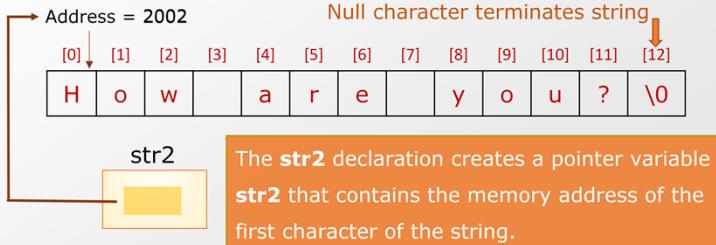
The **string 1** declaration creates an array of type **character**. The array has been allocated with memory to hold 13 elements including the null character. The C compiler also creates a pointer constant that is initialized to point to the first element of the array.

STRING VARIABLES

Two ways to declare a string:

```
char str1[ ] = "How are you?"; // using array declaration  
char *str2 = "How are you?"; // using pointer declaration
```

Array has been allocated with memory to hold 13 elements



Content Copyright Nanyang Technological University

42

In the **string 2** declaration, the C compiler creates a pointer variable **string 2** that points to the first character of the string. It contains the memory address of the first character of the string 'H'.

STRING VARIABLES

Two ways to declare a string:

Content Copyright Nanyang Technological University

43

As discussed earlier, there are two ways to declare a string

STRING VARIABLES

Two ways to declare a string:

```
// using array declaration  
char str1[ ] = "abc";
```

```
// using pointer declaration  
char *str2 = "abc";
```

Content Copyright Nanyang Technological University

44

We can declare a string by using a array declaration or pointer declaration

STRING VARIABLES

Two ways to declare a string:

```
// using array declaration  
char str1[ ] = "abc";
```

```
// using pointer declaration  
char *str2 = "abc";
```

Content Copyright Nanyang Technological University

45

Both **string 1** and **string 2** are pointers.

STRING VARIABLES

Two ways to declare a string:

```
// using array declaration  
char str1[ ] = "abc";
```

```
// using pointer declaration  
char *str2 = "abc";
```

str1 is a **pointer (or address) constant**

Content Copyright Nanyang Technological University

46

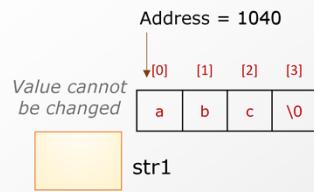
However, there is a difference between the two declarations.

String 1 is a pointer (or address) constant,

STRING VARIABLES

Two ways to declare a string:

```
// using array declaration  
char str1[ ] = "abc";
```



```
// using pointer declaration  
char *str2 = "abc";
```

str1 is a **pointer (or address) constant**

Content Copyright Nanyang Technological University

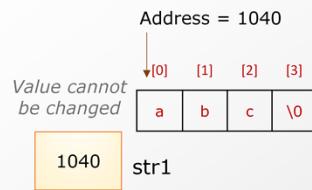
47

Pointer constant means that the value cannot be changed

STRING VARIABLES

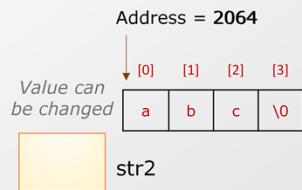
Two ways to declare a string:

```
// using array declaration  
char str1[ ] = "abc";
```



str1 is a **pointer (or address) constant**

```
// using pointer declaration  
char *str2 = "abc";
```



str2 is a **pointer variable**

Content Copyright Nanyang Technological University

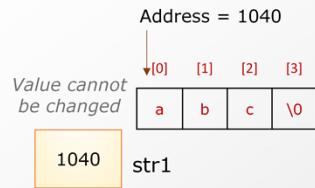
48

String 2 is a pointer variable.

STRING VARIABLES

Two ways to declare a string:

```
// using array declaration  
char str1[ ] = "abc";
```



As **str1** is a pointer (or address) constant, we cannot change its value.

str1 is a **pointer (or address) constant**

Content Copyright Nanyang Technological University

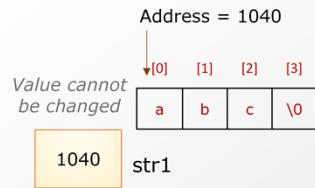
49

As **string1** is a pointer (or address) constant, we cannot change its value.

STRING VARIABLES

Two ways to declare a string:

```
// using array declaration  
char str1[ ] = "abc";
```



As **str1** is a pointer (or address) constant, we cannot change its value.

Therefore, the following statements are invalid:

Content Copyright Nanyang Technological University

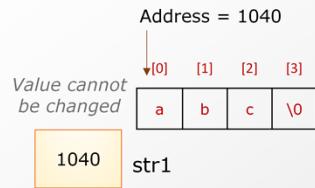
50

Therefore, the following statements are invalid:

STRING VARIABLES

Two ways to declare a string:

```
// using array declaration  
char str1[ ] = "abc";
```



As **str1** is a pointer (or address) constant, we cannot change its value.

Therefore, the following statements are invalid:

```
str1++;  
++str1;  
str1 = str2;
```

Content Copyright Nanyang Technological University

51

String 1 plus plus.

Or plus plus string 1.

Or string 1 equal string 2.

These are all invalid statements.

STRING VARIABLES

Two ways to declare a string:

Pointer variable allows its value to be changed.

```
// using pointer declaration  
char *str2 = "abc";
```

Address = 2064

Value can be changed [0] [1] [2] [3]

2064 str2

str2 is a pointer variable

STRING VARIABLES

Two ways to declare a string:

Pointer variable allows its value to be changed.

Therefore the following statements are valid:

```
// using pointer declaration  
char *str2 = "abc";
```

Address = 2064

Value can be changed [0] [1] [2] [3]

2064 str2

str2 is a pointer variable

STRING VARIABLES

Two ways to declare a string:

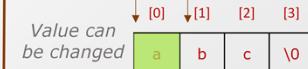
Pointer variable allows its value to be changed.

Therefore the following statements are valid:

```
str2++;
++str2;
str2 = str1;
```

```
// using pointer declaration
char *str2 = "abc";
```

Address = 2065



2065

str2++

str2 is a pointer variable

STRING VARIABLES

When declaring string variable using pointer notation, we assign a **string constant** to a **pointer** that points to char:

```
char *ptr;  
ptr = "This is a string";  
or char *ptr = "This is a string";
```

Content Copyright Nanyang Technological University

55

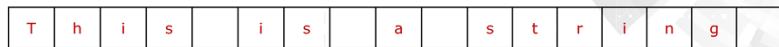
We can assign a string constant to a pointer that points to type **character**:
character asterisk pointer = double quote This is a string double quote;

STRING VARIABLES

```
char *ptr = "This is a string";
```

When a **string constant** is assigned to a **pointer**, C compiler will:

1. Allocate **memory space** to hold the string constant



Content Copyright Nanyang Technological University

56

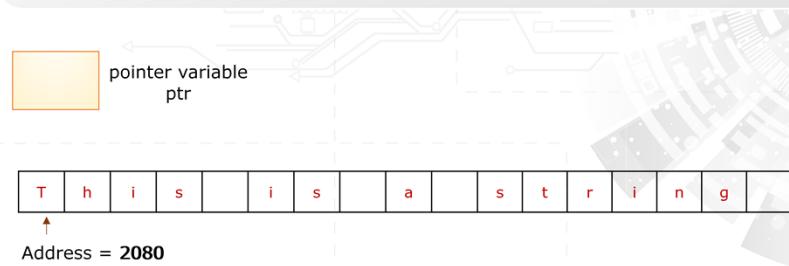
When a string constant is assigned to a pointer, the C compiler will allocate memory space to hold the string constant

STRING VARIABLES

```
char *ptr = "This is a string";
```

When a **string constant** is assigned to a **pointer**, C compiler will:

1. Allocate **memory space** to hold the string constant
2. Store the starting address of the string in the pointer variable



Content Copyright Nanyang Technological University

57

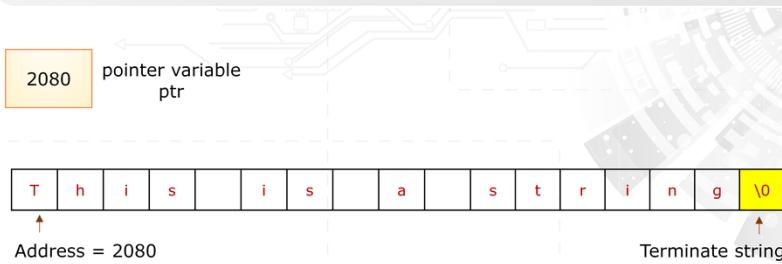
To store the starting address of the string (that is the address of the character T) in the pointer

STRING VARIABLES

```
char *ptr = "This is a string";
```

When a **string constant** is assigned to a **pointer**, C compiler will:

1. Allocate **memory space** to hold the string constant
2. Store the starting address of the string in the pointer variable
3. Terminate the string with **NULL** ('\0') character



Content Copyright Nanyang Technological University

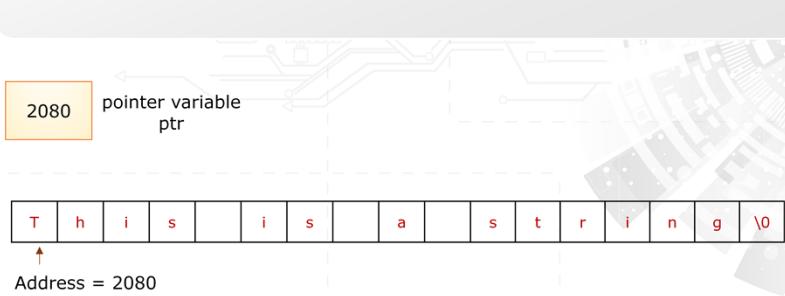
58

And terminate the string with a null character.

STRING VARIABLES

```
char *ptr = "This is a string";
```

However, ptr is a pointer variable that can be changed.



Content Copyright Nanyang Technological University

59

However, pointer is a pointer variable that can be changed.

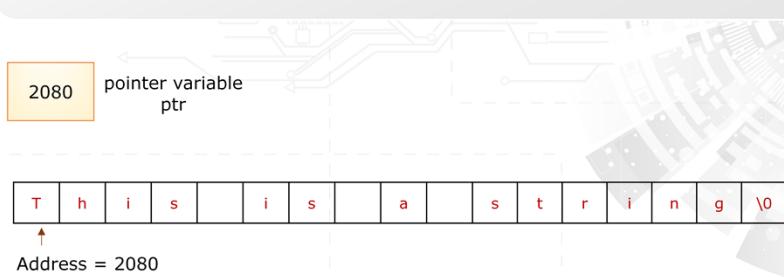
STRING VARIABLES

```
char *ptr = "This is a string";
```

However, ptr is a pointer variable that can be changed.

For example, the statement

```
ptr++;
```



Content Copyright Nanyang Technological University

60

For example, the statement `pointer++`

STRING VARIABLES

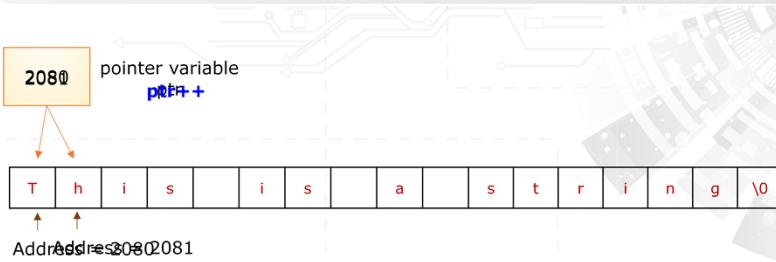
```
char *ptr = "This is a string";
```

However, ptr is a pointer variable that can be changed.

For example, the statement

```
ptr++;
```

changes the ptr variable to point to the next character (i.e. 'h').



Content Copyright Nanyang Technological University

61

changes the **pointer** variable to point to the next character (that is 'h') in the string "**This is a string**".

STRING OPERATION: EXAMPLE

Content Copyright Nanyang Technological University

62

String operation: Example

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
}
```

The diagram illustrates the relationship between a character array and a pointer. On the left, a code snippet in C declares a character array named 'array' and initializes it with the string 'pointer'. On the right, a pointer variable 'array' is shown pointing to the first element of the array. Below the array, its elements are displayed in a row of boxes: p, o, i, n, t, e, r, \0. A callout box labeled 'Using array notation' points to the declaration 'char array[] = "pointer";'.

Content Copyright Nanyang Technological University

63

The declaration

Character array = double quote pointer double quote;

declares a string variable **array** using array notation which is initialized with 7 characters plus the null character.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer
}
```

array → pointer\0

ptr1 → 10 spaces\0

1	0		s	p	a	c	e	s	\0
---	---	--	---	---	---	---	---	---	----

Using pointer notation

Content Copyright Nanyang Technological University

64

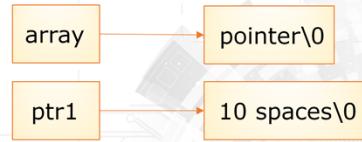
The declaration

Character asterisk pointer 1 = double quote 10 spaces double quote;
 declares a string variable **pointer 1** using pointer notation which is initialized and pointed to the string "**10 spaces**".

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
}
```



Output

```
ptr1 = 10 spaces
```

Content Copyright Nanyang Technological University

65

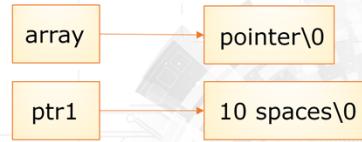
When **pointer1** is printed, the string "**10 spaces**" will be displayed.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);

}
```



Output

```
array = pointer
```

Content Copyright Nanyang Technological University

66

When **array** is printed, the string "**pointer**" will be displayed.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
}
```

The diagram illustrates the state of memory after the assignment `array[5] = 'A';` is executed. It shows two pointers: `array` pointing to the start of the string "pointer\0" and `ptr1` pointing to the start of the string "10 spaces\0". Below is a memory representation with indices 0 through 7. The array contains the characters p, o, i, n, t, A, r, \0. An arrow from the code highlights the assignment `array[5] = 'A'`, which is shown in green, pointing to the character 'A' at index 5 in the array.

Output

Content Copyright Nanyang Technological University

67

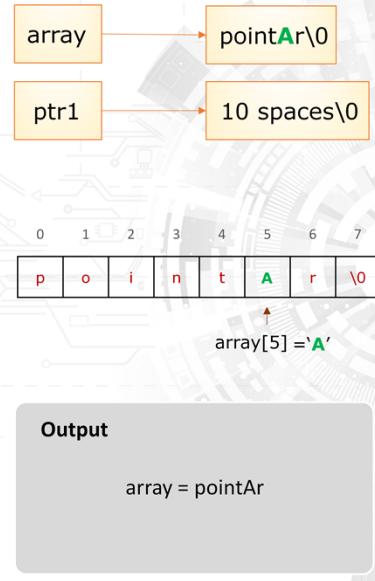
The statement
Array 5 = single quote A single quote;
is valid and the string is updated accordingly.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);

}
```



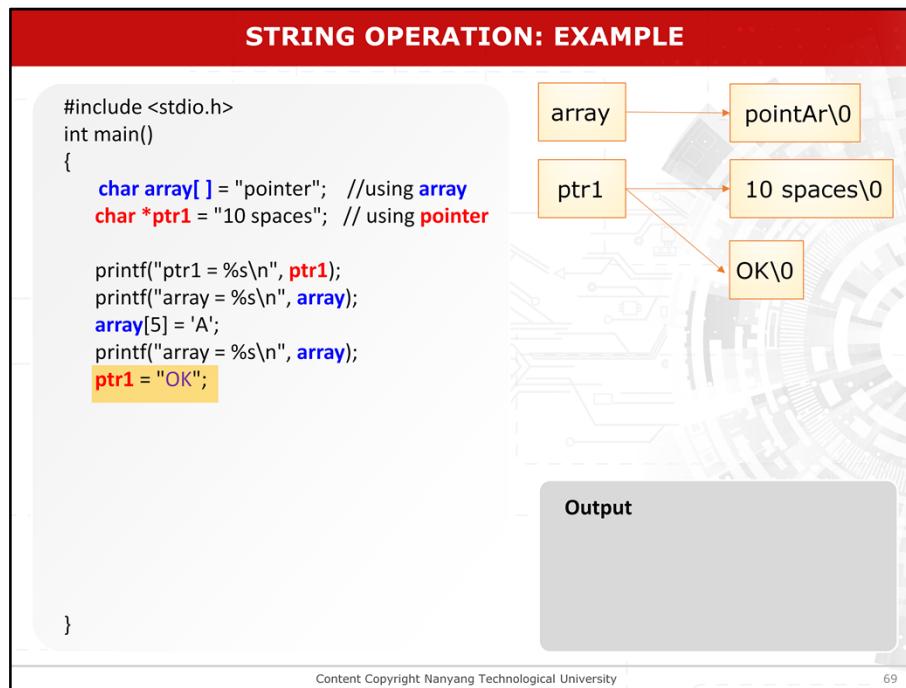
Output

```
array = pointAr
```

Content Copyright Nanyang Technological University

68

When `array` is printed, the string `p o i n t A r` will be displayed.



The statement

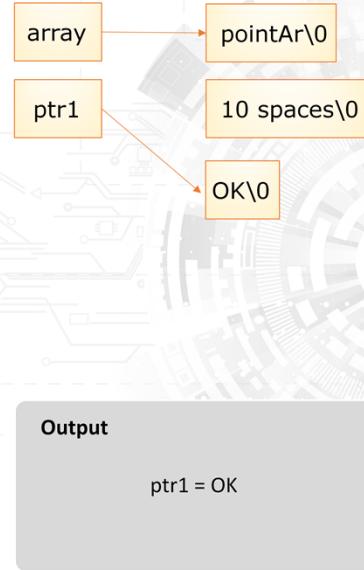
Pointer 1 = double quote OK double quote;
will update **pointer 1** to point to a new string "**OK**".

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);

}
```



Content Copyright Nanyang Technological University

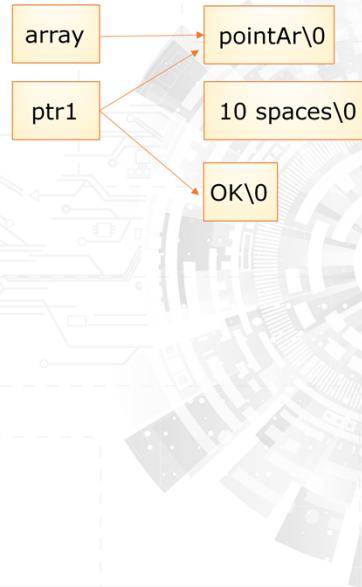
70

When **pointer1** is printed, the string "**OK**" will be displayed.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);
    ptr1 = array;
```



Content Copyright Nanyang Technological University

71

The statement

Pointer 1 = array;

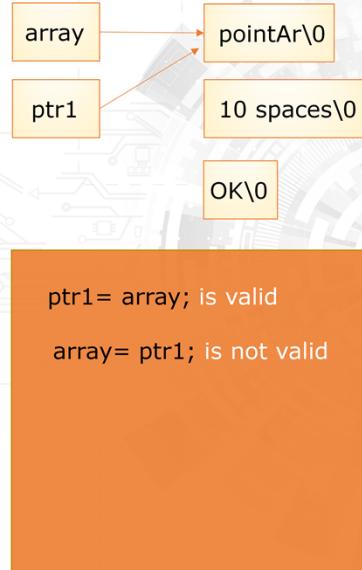
changes the pointer variable **pointer 1** to point to the same address contained in **array**.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);
    ptr1 = array;

}
```



Content Copyright Nanyang Technological University

72

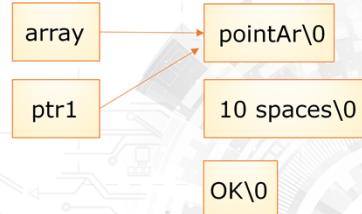
However, if we have
array = pointer 1;
which will be invalid

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);
    ptr1 = array;

}
```



ptr1 = array; is valid
array = ptr1; is not valid

- Error on type mismatch will occur

Content Copyright Nanyang Technological University

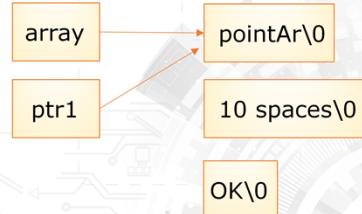
73

because an error on type mismatch will occur.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);
    ptr1 = array;
}
```



- ptr1 = array; is valid
- array = ptr1; is not valid
 - Error on type mismatch will occur
 - Not allowed to change the base address of array

Content Copyright Nanyang Technological University

74

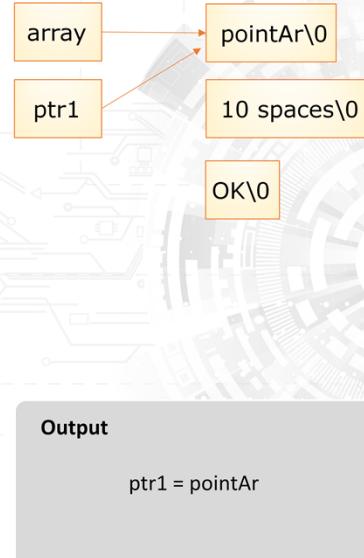
Also, we are not allowed to change the base address of **array**.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);
    ptr1 = array;
    printf("ptr1 = %s\n", ptr1);

}
```



Content Copyright Nanyang Technological University

75

When **pointer 1** is printed, the string **p o i n t A r** will be displayed.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);
    ptr1 = array;
    printf("ptr1 = %s\n", ptr1); );
    ptr1[5] = 'C';

}
```

array → pointer\0
ptr1 → 10 spaces\0
OK\0

0	1	2	3	4	5	6	7
p	o	i	n	t	e	r	\0

array[5] = 'C'

Output

Content Copyright Nanyang Technological University

76

The statement

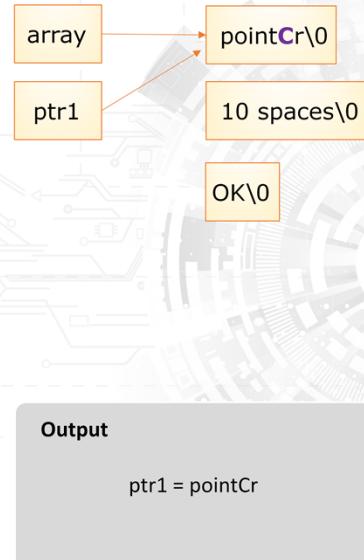
Pointer 1 5 = single quote C single quote;
is valid and the string is updated accordingly.

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);
    ptr1 = array;
    printf("ptr1 = %s\n", ptr1);
    ptr1[5] = 'C';
    printf("ptr1 = %s\n", ptr1);

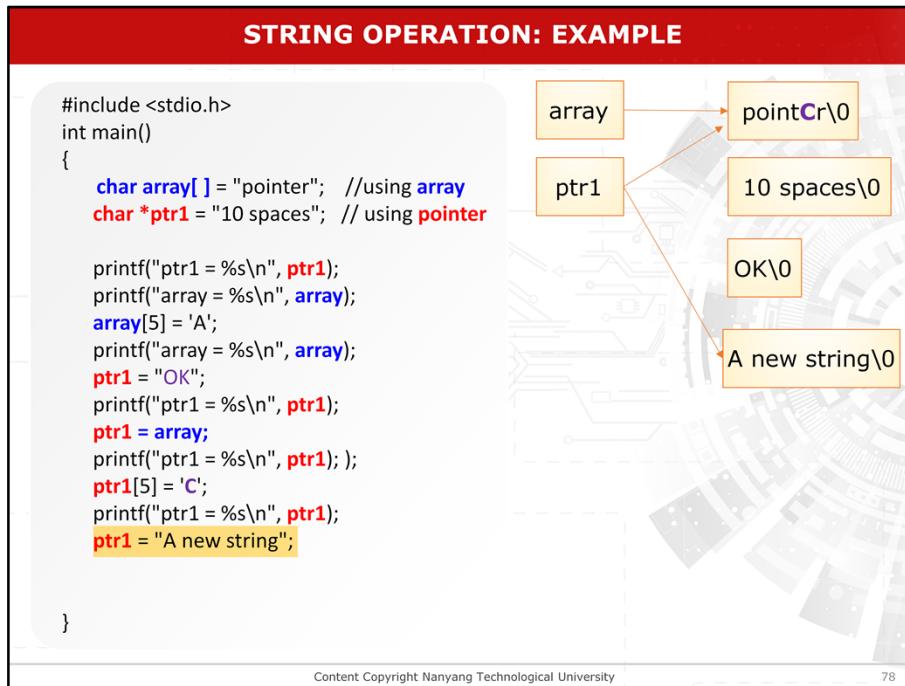
}
```



Content Copyright Nanyang Technological University

77

When **pointer 1** is printed, the string with **p o i n t c r** will be displayed.



The statement

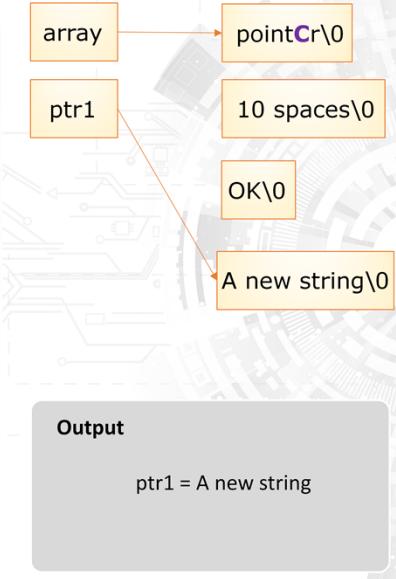
Pointer 1 = double quote A new string double quote;

is valid which changes the pointer variable **pointer 1** to point to the new string "**A new string**".

STRING OPERATION: EXAMPLE

```
#include <stdio.h>
int main()
{
    char array[ ] = "pointer"; //using array
    char *ptr1 = "10 spaces"; // using pointer

    printf("ptr1 = %s\n", ptr1);
    printf("array = %s\n", array);
    array[5] = 'A';
    printf("array = %s\n", array);
    ptr1 = "OK";
    printf("ptr1 = %s\n", ptr1);
    ptr1 = array;
    printf("ptr1 = %s\n", ptr1);
    ptr1[5] = 'C';
    printf("ptr1 = %s\n", ptr1);
    ptr1 = "A new string";
    printf("ptr1 = %s\n", ptr1);
    return 0;
}
```



Content Copyright Nanyang Technological University

79

When **pointer 1** is printed, the string "**A new string**" will be displayed.

SUMMARY

After this lesson, you should be able to:

- Explain strings are arrays of characters
- Define a string
- Define string constant
- Describe how to declare a string variable
- Declare strings using the array notation
- Declare strings using pointer notation
- Explain the difference in the declaration of string variables that use array or pointer
- Apply string declaration

Content Copyright Nanyang Technological University

80

In summary, after viewing this video lesson, you should be able to do the points listed.