Strings

The most common use for one-dimensional arrays is to store **strings of characters**.

In C++, a **string** is defined as a character array terminated by a **null symbol** (**'\0'**).

To declare an array str that could hold a 10-character string, one would write:

```
char str[11];
```

Specifying the size as 11 makes room for the null at the end of the string.

Strings

Character Array Initialization:

Character arrays that will hold strings allow a shorthand initialization that takes this form:

char array-name[size] = “string”;

For example, the following code fragment initializes **str** to the phrase “hello”:

```
char str[6] = "hello";
```

Remember that one has to make sure to make the array long enough to include the null terminator.

This is the same as writing

```
char str[6] = { 'h', 'e', 'l', 'l', 'o', '\0' };
```

Why NULL character (‘\0’) is important?

This is how the compiler and (other libraries) knows where the string ends

Strings

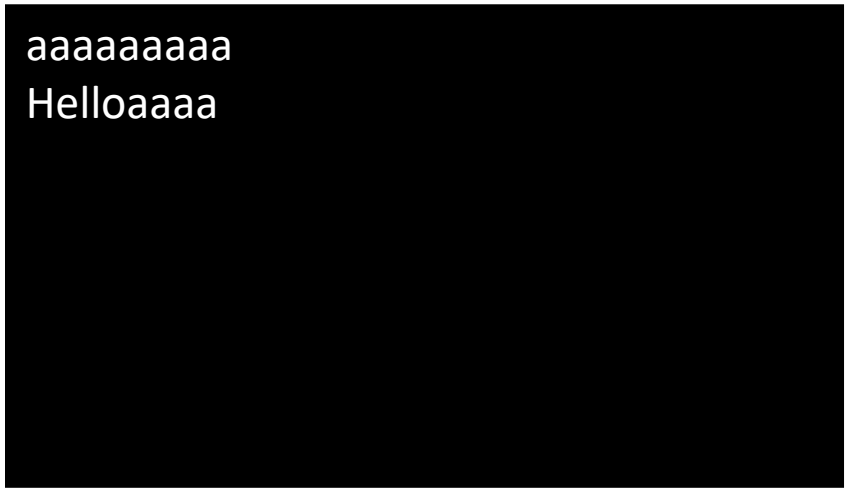
Example:

```
#include <iostream>
using namespace std;

int main()
{
    char sample[10] = {'a','a','a','a','a','a','a','a','a','\0'};

    cout << sample << endl;

    sample[0] = 'H';
    sample[1] = 'e';
    sample[2] = 'l';
    sample[3] = 'l';
    sample[4] = 'o';
    sample[5] = '\0';

    cout << sample << endl;

    return(0);
}
```

Strings

Example:

```
#include <iostream>
using namespace std;

int main()
{
    char sample[10] = {'a','a','a','a','a','a','a','a','a','\0'};

    cout << sample << endl;

    sample[0] = 'H';
    sample[1] = 'e';
    sample[2] = 'l';
    sample[3] = 'l';
    sample[4] = 'o';

    cout << sample << endl;

    return(0);
}
```



Reading a String from the Keyboard

- Make an array, that will receive the string
- The following program reads (part of) a string entered by the user:

```
#include <iostream>
using namespace std;

int main()
{
    char str[80];
    cout << "Enter a string : ";
    cin >> str; // read string from keyboard
    cout << "Here is your string : ";
    cout << str;
    return(0);
}
```

Problem: Entering the string “**This is a test**”, the above program only returns “**This**”, not the entire sentence.

Reason: The C++ input/output system stops reading a string when the first **whitespace** character is encountered.

Solution: Use another C++ library function, **gets_s()**.

Syntax: gets_s(char* **destination**)

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char str[80]; // long enough for user input?
    cout << "Enter a string : ";
    gets_s(str); // read a string from the keyboard
    cout << "Here is your string : ";
    cout << str << endl;
    return(0);
}
```

Some C++ Library Functions for Strings

C++ supports a range of string-manipulation functions.
The most common are:

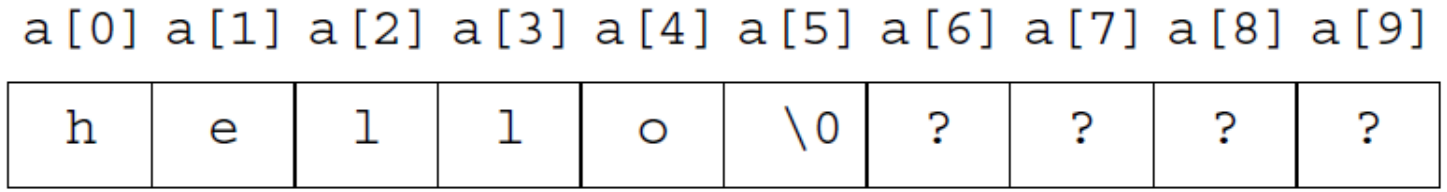
- **strcpy_s()** : copy characters from one string to another
- **strcat_s()** : concatenation of strings
- **strlen()** : length of a string
- **strcmp()** : comparison of strings

Some C++ Library Functions for Strings

Example of **strcpy_s()**:

Syntax: strcpy_s(char* **destination**, char* **source**)

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char a[10];
    strcpy_s(a, "hello");
    cout << a;
    return(0);
}
```



Some C++ Library Functions for Strings

Example of **strcpy_s()**:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char a[10]="Hi";
    char b[10] = "Hello";
    strcpy_s(a, b);
    cout << a;
    return(0);
}
```

Some C++ Library Functions for Strings

Example of **strlen()**:

Syntax: strlen(char* **source**)

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char str[80];
    cout << "Enter a string : ";
    gets_s(str);
    cout << "Length is : " << strlen(str);
    return(0);
}
```

Some C++ Library Functions for Strings

Example of **strcats_s()**:

Syntax: strcpy_s(char* **string1**, char* **string2**)

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char a[20]="Hi";
    char b[15] = " and Hello";
    strcat_s(a, b);
    cout << a;
    return(0);
}
```

Some C++ Library Functions for Strings

Example of **strcats_s()**:

Note: The **first string** array has to be large enough to hold both strings:

```
char a[20] = "Hi";
```

'H'	'i'	'\0'	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

```
char b[15] = " and Hello";
```

' '	'a'	'n'	'd'	' '	'H'	'e'	'l'	'l'	'o'	'\0'	?	?	?	?	?	?	?	?	?

```
strcat_s(a, b);
```

'H'	'i'	' '	'a'	'n'	'd'	' '	'H'	'e'	'l'	'l'	'o'	'\0'	?	?	?	?	?	?	?

Some C++ Library Functions for Strings

Example of **strcats_s()**:

To be on the safe side:

Size of first string >= strlen(s1) + strlen(s2) + 1

Some C++ Library Functions for Strings

Example of **strcmp()**:

Syntax: strcmp(char * **string1**, char* **string2**)

The strcmp(a, b) function compares two strings and returns the following result:

- str_1 == str_2 : 0
- str_1 > str_2 : positive number
- str_1 < str_2 : negative number

Some C++ Library Functions for Strings

Example of **strcpy_s()**:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char str[80];
    cout << "Enter password : ";
    gets_s(str);
    if (strcmp(str, "password") == 0)
    {
        cout << " Logged on.\n";
    }
    else
    {
        cout << "Invalid password.\n";
    }
    return(0);
}
```

Thanks a lot

If you are taking a Nap, **wake up**.....Lecture Over