Strings

Course Code: CSC1102 &1103 Course Title: Introduction to Programming

Dept. of Computer Science
Faculty of Science and Technology

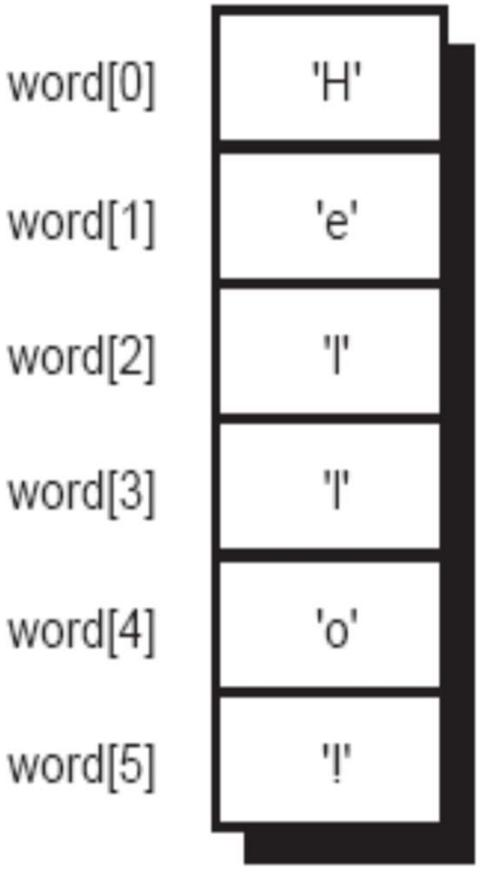
Lecturer No:	**6**	**Week No:**	**5 (1X1.5 hrs)**	**Semester:**	
Lecturer:	*Name & email*				

Lecture 6: Outline

- Strings
- Character Arrays/ Character Strings
  - Initializing Character Strings. The null string.
  - Escape Characters
  - Displaying Character Strings
  - Inputting Character Strings
  - String processing:
    - Testing Strings for Equality
    - Comparing   Strings
    - Copying Strings
  - Functions in <string.h>
  - String to number conversion functions
  - Character Strings, Structures, and Arrays
  - Example: Simple dictionary program
    - Sorting the dictionary
    - A better search in sorted arrays

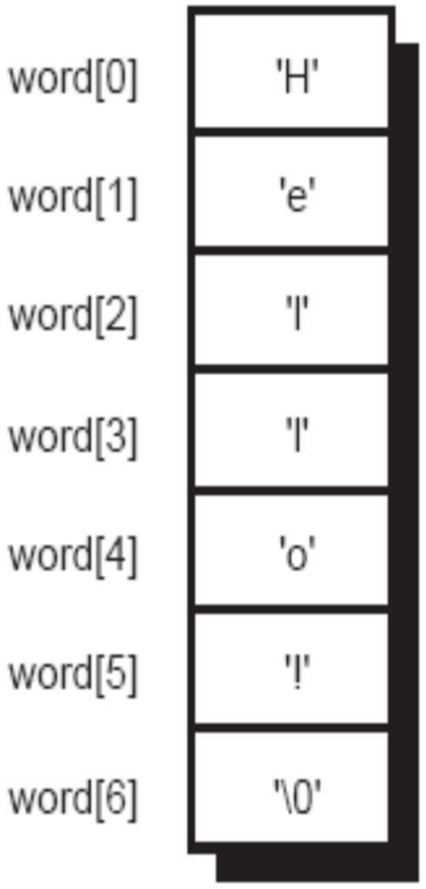
Arrays of characters

- `char word [] = { 'H', 'e', 'l', 'l', 'o', '!' };`
- To print out the contents of the array word, you run through each element in the array and display it.
- To do processing's of the word (copy, concatenate two words, etc.) you need to have the actual length of the character array in a separate variable !



Character strings

- A method for dealing with character arrays without having to worry about precisely how many characters you have stored in them:
- **Placing a special character at the end of every character string**. In this manner, the function can then determine for itself when it has reached the end of a character string after it encounters this special character.
- In the C language, the special character that is used to signal the end of a string is known as the *null* character and is written as **'\0'**.
- char word [] = { 'H', 'e', 'l', 'l', 'o', '!', '\0' };



Example: string length

```
// Function to count the number of characters in a string
#include <iostream>
using namespace std;
int stringLength (char string[]){
    int count = 0;
    while ( string[count] != '\0' )
        ++count;
    return count;
}
int main (void) {
    char word1[] = { 'a', 's', 't', 'e', 'r', '\0' };
    char word2[] = { 'a', 't', '\0' };
    char word3[] = { 'a', 'w', 'e', '\0' };
    cout<<stringLength (word1)<<endl;
    cout<<stringLength (word2)<<endl;
    cout<<stringLength (word3)<<endl;
    return 0;
}
```

Example: const strings

```
// Function to count the number of characters in a string
#include <iostream>
using namespace std;
int stringLength (const char string[]){
    int count = 0;
    while ( string[count] != '\0' )
        ++count;
    return count;
}
int main (void) {
    const char word1[] = { 'a', 's', 't', 'e', 'r', '\0' };
    const char word2[] = { 'a', 't', '\0' };
    const char word3[] = { 'a', 'w', 'e', '\0' };
    cout<<stringLength (word1)<<endl;
    cout<<stringLength (word2)<<endl;
    cout<<stringLength (word3)<<endl;
    return 0;
}
```

The function declares its argument as a const array of characters because it is not making any changes to the array

Initializing character strings

- **Initializing a string:**

```
char word[] = "Hello!";
```

- **Is equivalent with:**

```
char word[] = { 'H', 'e', 'l', 'l', 'o', '!', '\0' };
```

- **The null string: A character string that contains no characters other than the null character**

```
char empty[]= "";
char buf[100]= "";
```

- **Initializing a very long string over several lines:**

```
char letters[] =
{ "abcdefghijklmnopqrstuvwxyz\
ABCDEFGHIJKLMNOPQRSTUVWXYZ" };
```

- **Adjacent strings are concatenated:**

```
char letters[] =
{ "abcdefghijklmnopqrstuvwxyz"
"ABCDEFGHIJKLMNOPQRSTUVWXYZ" };
cout<<"Programming" " in C is fun";
```

Strings vs Characters

- The string constant "x“
- The character constant 'x‘
- Differences:
  1. 'x' is a basic type (char) but "x" is a derived type, an array of char
  2. "x" really consists of two characters, 'x' and '\0', the null character

Escape characters

- the backslash character has a special significance
- other characters can be combined with the backslash character to perform special functions. These are referred to as *escape characters*.

\a	Audible alert
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\\\\	Backslash
\\"	Double quotation mark
\\'	Single quotation mark
\\?	Question mark
\nnn	Octal character value *nnn*
\unnnn	Universal character name
\Unnnnnnnn	Universal character name
\xnn	Hexadecimal character value *nn*

String functions

- The C++ library supplies several string-handling functions; You don't have to re-write them from scratch !
- C++ uses the <string.h> header file to provide the prototypes.
- Most frequently used functions: strlen(), strcat(), strncat(), strcmp(), strncmp(), strcpy(), and strncpy().

- #include <string.h>
- strcat (*s1, s2*)
  - Concatenates the character string *s2* to the end of *s1*, placing a null character at the end of the final string.The function also returns *s1*.
- strcmp (*s1, s2*)
  - Compares strings *s1* and *s2* and returns a value less than zero if *s1* is less than *s2*, equal to zero if *s1* is equal to *s2*, and greater than zero if *s1* is greater than *s2*.
- strcpy (*s1, s2*)
  - Copies the string *s2* to *s1*, also returning *s1*.
- strlen (*s*)
  - Returns the number of characters in *s*, excluding the null character.

String functions (cont.)

- **strncat (*s1, s2, n*)**
  - Copies *s2* to the *end* of *s1* until either the null character is reached or *n* characters have been copied, whichever occurs first. Returns *s1*.
- **strncmp (*s1, s2, n*)**
  - Performs the same function as strcmp, except that at most *n* characters from the strings are compared.
- **strncpy (*s1, s2, n*)**
  - Copies *s2* to *s1* until either the null character is reached or *n* characters have been copied, whichever occurs first. Returns *s1*.
- **strchr (*s, c*)**
  - Searches the string *s* for the last occurrence of the character *c*. If found, a pointer to the character in *s* is returned; otherwise, the null pointer is returned.
- **strstr (*s1, s2*)**
  - Searches the string *s1* for the first occurrence of the string *s2*. If found, a pointer to the start of where *s2* is located inside *s1* is returned; otherwise, if *s2* is not located inside *s1*, the null pointer is returned.

Example: String functions

```
#include <iostream>
using namespace std;
#include <string.h> /* provides strlen() prototype */

#define PRAISE " What a super marvelous name!"

int main(void) {
char name[40];
cout<<"What's your First Name? "<<endl;
cin>>name;
cout<<"Hello "<< name<< PRAISE<<endl;
cout<<"Your name of "<<strlen(name)<<" letters occupies
"<<sizeof name<<" memory"<<endl;
return 0;
}
```

Example: String functions

```
#include <iostream>
#include <string.h>
using namespace std;
int main(void) {
char string1[] = "this is";
char string2[] = "a test";
char string3[20] = "Hello, ";
char string4[] = "world!";
cout<< string3<<endl;
strcat(string3, string4);
cout<<string3<<endl;
if(strcmp(string1, string2) == 0)
       cout<<"strings are equal"<<endl;
else cout<<"strings are different"<<endl;
return 0;
}
```