University of Mosul

College of Computer Sciences and Mathematics

Department of Artificial Intelligence

Algorithms and Structured Programming (2)

First stage

Lecture 5

أ.م. بيداء سليمان بهنام

Assist.Prof. Baydaa Sulaiman Bahnam

Lecture 5 Outline

1. C++ Strings
2. C-Style Character Strings (Character Arrays)
3. C++ Style Strings

C++ Strings

A string is a variable that stores a sequence of characters, such as "Hello" or “Welcome to the 2nd course ".

There are two types of strings commonly used in C++ :

C++ Strings

1. C-style character strings (character arrays)
2. C++ Style String

1. C-Style Character Strings (Character Arrays)

The C-style character string originated within the C language and continues to be supported within C++. This string is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus, a null-terminated string contains the characters that comprise the string followed by a null. For example:

```
char str[] = "Hello";
```

Or:

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

Note: In C-style strings, the data type is **char**[], not string. The char type holds individual characters (e.g., 'H'), and a C-style string is a collection of such characters ending with a null character.

	Array members	str[0]	str[1]	str[2]	str[3]	str[4]	str[5]
char str	Array elements Variable	H العنصر الاول	e العنصر الثاني	l العنصر الثالث	l العنصر الرابع	o العنصر الخامس	\0 العنصر السادس
	Array indexes	0	1	2	3	4	5

I. C-Style Character Strings (Character Arrays)

In C++, C-style character strings (character arrays) can be created in various ways. Here are the different methods:

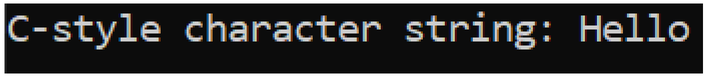
1. Direct Initialization

You can directly initialize a character array with a string literal. The compiler automatically adds the null terminator ('\0’).

Example 1:

```
#include<iostream>
 using namespace std;
 int main()
 {
     // Direct initialization with a string literal
     char str[] = "Hello";// or       char str[6] = “Hello";

     cout << "C-style character string: " << str << endl;
     return 0;
 }
```



```
C-style character string: Hello
```

Explanation:

The character array str is automatically sized to include the null terminator ('\0').

The string literal "Hello" is stored in the array str.

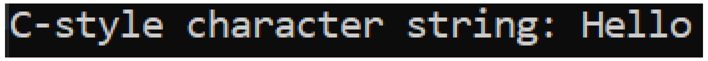
I. C-Style Character Strings (Character Arrays)

2. Partial Initialization

You can partially initialize a character array and leave the rest uninitialized. However, you must include the null terminator manually.

Example:

```
#include<iostream>
using namespace std;
int main() {
    // Partial initialization
    char str[10] = { 'H', 'e', 'l', 'l', 'o', '\0’ };
   //or  char str[] = { 'H', 'e', 'l', 'l', 'o', '\0' };
    cout << "C-style character string: " << str << endl;
    return 0;
}
```



```
C-style character string: Hello
```

Explanation:

The first five elements are initialized with characters, and the sixth element is the null terminator.

The remaining elements of the array remain uninitialized but are irrelevant because the string ends at the null terminator.

تبقى العناصر المتبقية من المصفوفة غير مهيأة ولكنها غير ذات صلة لأن السلسلة تنتهي عند نقطة النهاية الفارغة '\0’.

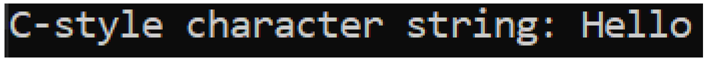
I. C-Style Character Strings (Character Arrays)

3. Character-by-Character Initialization

You can manually assign each character to specific positions in the array and ensure you include the null terminator at the end.

Example:

```
#include<iostream>
using namespace std;
int main()
{
    // Character-by-character initialization
    char str[6]; // Array size must be at least 6 (5 characters + 1 null terminator)
    str[0] = 'H';
    str[1] = 'e';
    str[2] = 'l';
    str[3] = 'l';
    str[4] = 'o';
    str[5] = '\0'; // Null terminator
    cout << "C-style character string: " << str << endl;
    return 0;
}
```



```
C-style character string: Hello
```

Explanation:

The array size must be specified, and it should be large enough to hold the characters and the null terminator.

You need to manually place the null terminator ('\0') to indicate the end of the string.

Functions of C-Style Character Strings (Character Arrays)

When using C-style character strings declared as char[], you can work without including the **<cstring>** header. However, if you use standard C string functions such as:strlen(), strcpy(), etc.; you must include the <cstring> header, which provides declarations for these functions as part of the C standard library.

```
#include <cstring> // Include the cstring library, Needed for strlen, strcpy, etc.
```

Sr.No	Function & Purpose
1	**strcpy(s1, s2);** Copies string s2 into string s1.
2	**strcat(s1, s2);** Concatenates string s2 onto the end of string s1.
3	**strlen(s1);** Returns the length of string s1.
4	**strcmp(s1, s2);** Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	**strchr(s1, ch);** Returns a pointer to the first occurrence of character ch in string s1.
6	**strstr(s1, s2);** Returns a pointer to the first occurrence of string s2 in string s1.

2. C++ Style Strings

Initializing a String

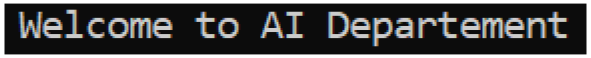
Initializing means assigning an initial value to a string variable. This is done using the assignment operator (=) along with the text enclosed in double quotes ("").

Note: The data type used for strings is string, not char. The char type is used to store a single character enclosed in single quotes ('a'), while string is used for a sequence of characters enclosed in double quotes ("Welcome").

```
        string str = "Some Text here";
```

Example:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
string str = " Welcome to AI Departement";
cout << str << endl;
return 0;
}
```



```
Welcome to AI Departement
```

Reading Strings

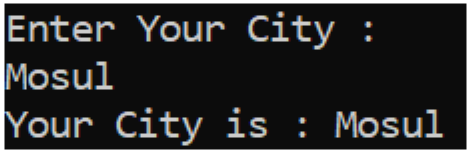
There are three methods to take a string as an input, and these methods are given as follows:

- **cin**
- **getline**
- **stringstream**

- **cin Method:** This is the simplest way to take a string as an input. The syntax is given as follows:

Syntax: **cin>>string_name;**

```
#include <iostream>
using namespace std;
int main() {
string s;
cout << "Enter Your City : " << endl;
//enter the string here
cin >> s;
cout << "Your City is : " << s;
return 0;}
```



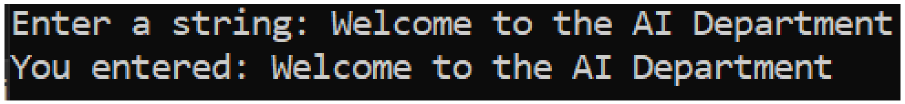
```
Enter Your City :
Mosul
Your City is : Mosul
```

- **getline Method**

In C++, the **getline function** is a way to read an entire line of console input into a variable.

Syntax: **getline(cin, string_name);**

```
#include <iostream>
#include <string>  // Include the string library
using namespace std;
int main() {
    string s;  // Declare an empty string
    cout << "Enter a string: ";  // Prompt the user
    getline(cin, s);  // Read the entire line from user input
    cout << "You entered: " << s << endl;  // Print the entered string
    return 0;}
```



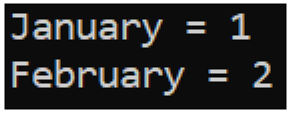
```
Enter a string: Welcome to the AI Department
You entered: Welcome to the AI Department
```

2. C++ Style Strings

Updating String

The string variable can be updated store a new string literal in a similar way it is initialized.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
string str = " January = 1";
cout << str << endl;
str = " February = 2";
cout << str << endl;
return 0;}
```

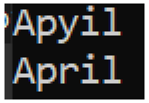


```
January = 1
February = 2
```

Updating a character in the String

A single character can also be changed by first accessing the character and then using assignment operator to assign value.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
string str = "Apyil";
cout << str << endl;
// Accessing 3rd character
str[2]= 'r';
cout << str << endl;
return 0;}
```



```
Apyil
April
```

Functions of C++ Style Strings

When using C++-style strings declared as std::string, you must include the <string> header. This header provides the class definition and necessary member functions as part of the C++ Standard Library.

If you use standard C++ string functions such as: length(), substr(), append(), find(), etc., you must include the <string> header to access their declarations and functionality.

```
#include <string>  // Include the string library
```

Function	Description
`length()` or `size()`	Returns the number of characters in the string.
`empty()`	Returns `true` if the string is empty.
`at(index)`	Returns the character at the specified position with bounds checking.
`[]` (indexing)	Accesses the character at the given index (no bounds checking).
`append(str)`	Appends `str` to the end of the string.
`+`	Concatenates two strings.
`substr(pos, len)`	Returns a substring starting at `pos` with length `len`.

Functions of C++ Style Strings

If you use standard C++ string functions such as: length(), substr(), append(), find(), etc., you must include the <string> header to access their declarations and functionality.

```
#include <string>  // Include the string library
```

Function	Description
`find(str)`	Returns the position of the first occurrence of `str`, or `npos` if not found.
`compare(str)`	Compares two strings; returns 0 if equal, <0 or >0 otherwise.
`insert(pos, str)`	Inserts `str` at position `pos`.
`erase(pos, len)`	Removes `len` characters starting from `pos`.
`replace(pos, len, str)`	Replaces part of the string with `str`.
`clear()`	Removes all characters from the string.
`c_str()`	Returns a C-style (`const char*`) version of the string for use with C APIs.

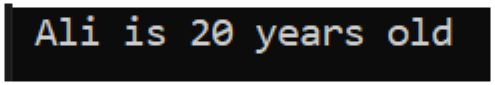
String Concatenation

String concatenation is a way to add two strings together. This can be done using two ways :

1. Addition Operator +

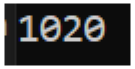
The addition operator is used to add two elements. In case of strings, the addition operator concatenates the two strings. This is clearly explained in the following example :

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
string x = " Ali is ";
string y = "20 years old";
cout << x + y << endl;
return 0;
}
```



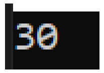
```
Ali is 20 years old
```

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
string x = "10";
string y = "20";
cout << x + y << endl;
return 0;
}
```



```
1020
```

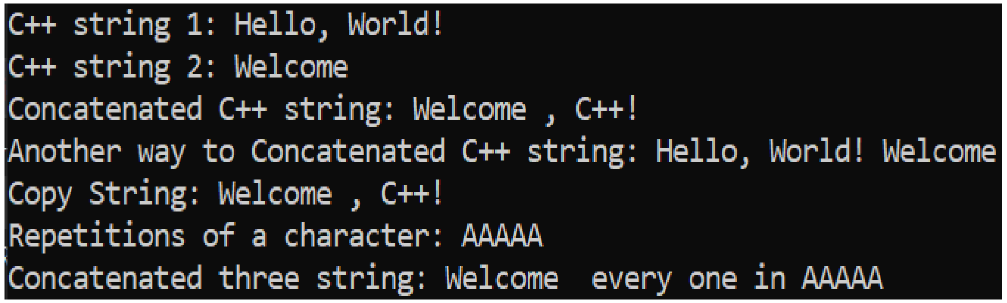
```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
int x = 10;
int y = 20;
cout << x + y << endl;
return 0;
}
```



```
30
```

String Concatenation

```cpp
#include<iostream>
#include <string>  // Include the string library
using namespace std;
int main() {
    // Creating a C++ string
    string str1 = "Hello, World! ";
    // You can also declare an empty string and assign a value later
    string str2;
    str2 = "Welcome ";
    // Concatenating strings
    string str3 = str2 + ", C++!";
    // Concatenating two strings
    string str4 = str1 + str2;
    //Creates a copy of another string
    string str5 = str3;
    //Creates a string with multiple repetitions of a character
    string str6(5, 'A');  // "AAAAA"
    // Concatenating three strings
    string str7 = str2 + " every one in " + str6;
    // Printing C++ strings
    cout << "C++ string 1: " << str1 << endl;
    cout << "C++ string 2: " << str2 << endl;
    cout << "Concatenated C++ string: " << str3 << endl;
    cout << "Another way to Concatenated C++ string: " << str4 << endl;
    cout << "Copy String: " << str5 << endl;
    cout << "Repetitions of a character: " << str6 << endl;
    cout << "Concatenated three string: " << str7 << endl;
    return 0;}
```



```
C++ string 1: Hello, World!
C++ string 2: Welcome
Concatenated C++ string: Welcome , C++!
Another way to Concatenated C++ string: Hello, World! Welcome
Copy String: Welcome , C++!
Repetitions of a character: AAAAA
Concatenated three string: Welcome  every one in AAAAA
```

String Concatenation

2. Using string append() function

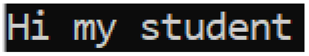
C++ is an object-oriented programming language, and hence a string is actually an object, which contain functions that can perform certain operations on strings. We can use string append() method to append one string to another. The syntax of this operation is as follows:

Syntax:

string_1.append(string_2);

The append() function adds the contents of string_2 to the end of string_1.

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
string x = "Hi ";
string y = "my student";
x.append(y);
cout << x << endl;
return 0;
}
```



```
Hi my student
```

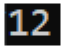
Length of a String

The length of a string is the number of characters present in the string. This can be accessed using the length() method in <string> header file.

Syntax: **int len = string_1.length();**

It means that the variable len will hold the length of the string string_1, which is the number of characters it contains without null.

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
string x = "Hi every one";
cout << x.length() << endl;
return 0;
}
```



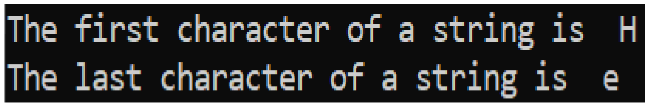
```
12
```

at() function

The at() function returns an indexed character from a string_1.

Syntax: **string_1.at(index);**

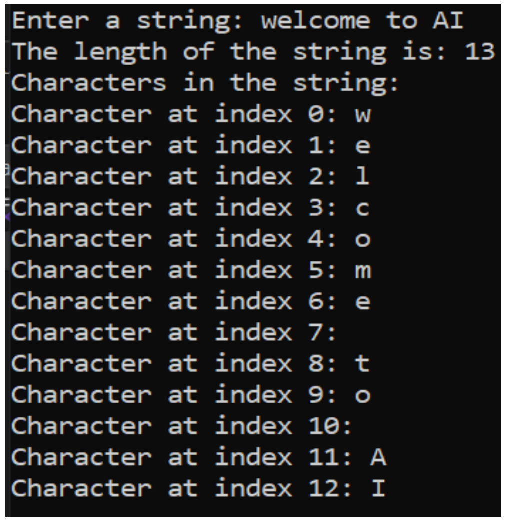
```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
string str = "Hi every one";
//Get the first character of a string :
cout << "The first character of a string is  "<<str.at(0) << endl;
//Get the last character of a string :
cout <<"The last character of a string is  " <<str.at(str.length() - 1);
return 0;}
```



```
The first character of a string is  H
The last character of a string is  e
```

Write a C++ program that reads a string input from the user, displays the total length of the string, and then prints each character along with its corresponding index.

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    string text;
    // Ask the user to enter a string
    cout << "Enter a string: ";
    getline(cin, text);
    // Print the length of the string
    cout << "The length of the string is: " << text.length() << endl;
    // Print each character using at()
    cout << "Characters in the string:" << endl;
    for (size_t i = 0; i < text.length(); ++i) {
        cout << "Character at index " << i << ": " << text.at(i) << endl;
    }
    return 0;
}
```

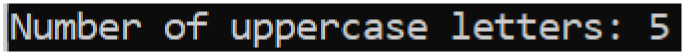


```
Enter a string: welcome to AI
The length of the string is: 13
Characters in the string:
Character at index 0: w
Character at index 1: e
Character at index 2: l
Character at index 3: c
Character at index 4: o
Character at index 5: m
Character at index 6: e
Character at index 7:
Character at index 8: t
Character at index 9: o
Character at index 10:
Character at index 11: A
Character at index 12: I
```

Write a c++ program that uses a function named (`countUppercase`) with a const reference parameter to return the number of uppercase letters in a string.
The string is: Hello World CPP

```
#include <iostream>
#include <string>
using namespace std;
// Function to count the number of uppercase letters in a string
int countUppercase(const string& str) {
    int count = 0;
    for (int i = 0; i < str.length(); i++) {
        if (isupper(str[i])) {                       or  if (str[i] >= 'A' && str[i] <= 'Z') {
            count++;
        }
    }
    return count;
}
int main() {
    string s = "Hello World CPP";
    int result = countUppercase(s);
    cout << "Number of uppercase letters: " << result << endl;
    return 0;
}
```

تمرير string بواسطة المرجع reference لا يتم نسخ string، بل يُستخدم نفس المتغير الموجود في main
اما const تعني لا يُسمح بتعديل string داخل الدالة.



```
Number of uppercase letters: 5
```

A.P. Baydaa Sulaiman

Array of Strings in C++

In C++, a string is sequence of characters that is used to store textual information. Internally, it is implemented as a dynamic array of characters. Array of strings is the array in which each element is a string.

The general syntax of array of strings is:

string array_name[size]

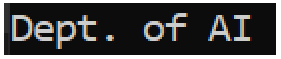
where **array_name** is the name assigned to the array and size is the desired size.

What will be the output of the following C++ program when executed?

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

int main() {
    // Array of C++ style strings
    string arr[3] = { "Dept.", "of", "AI" };

    for (int i = 0; i < 3; i++)
        cout << arr[i] << " ";
    return 0;
}
```



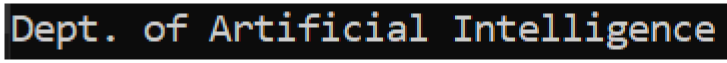
```
Dept. of AI
```

Array of Strings in C++

What will be the output of the following C++ program when executed?

```cpp
#include <iostream>
using namespace std;
int main() {
    // Array of C++ style strings
    string arr[3] = { "Dept.", "of", "AI" };
    // Modifying strings
    arr[2] = "Artificial Intelligence";

    for (int i = 0; i < 3; i++)
        cout << arr[i] << " ";
    return 0;
}
```



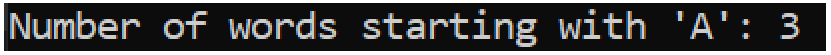
```
Dept. of Artificial Intelligence
```

Write a C++ program to count how many words in a string array named words start with the letter ‘A’.

```
Ex: words[5] = { "AI", "Algorithm", "Data", "Apple", "Code" };
Number of words starting with 'A': 3
```

```cpp
#include <iostream>
using namespace std;
int main() {
    string words[5] = { "AI", "Algorithm", "Data", "Apple", "Code" };
    char target = 'A';
    int count = 0;

    for (int i = 0; i < 5; i++) {
        if (words[i][0] == target)
            count++;
    }
    cout << "Number of words starting with 'A': " << count << endl;
    return 0;
}
```



```
Number of words starting with 'A': 3
```

A.P. Baydaa Sulaiman

Array of Strings in C++

Write a C++ program to count how many words in an array contain the letter 'o'.

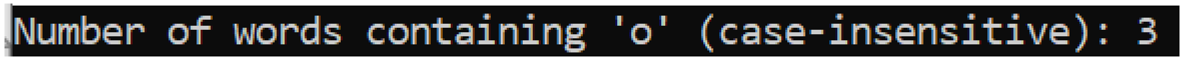
```
Ex: words[5] = { "CODE", "Loop", "AI", "C++", "ROBOT" }
Number of words containing 'o': 4
```

```cpp
#include <iostream>
#include <cctype>  // We need to include this library to use tolower()
using namespace std;

int main() {
    // Define an array of words
    string words[5] = { "CODE", "Loop", "AI", "C++", "ROBOT" };
    // Define the target character (the one we are looking for)
    char target = 'o';
    // Convert the target character to lowercase to ensure case-insensitive comparison
    target = tolower(target);
    // Initialize a counter to store the number of words containing the target character
    int count = 0;
    // Loop through each word in the array
    for (int i = 0; i < 5; i++) {
        // Inner loop to check each character in the current word
        for (int j = 0; j < words[i].length(); j++) {
            // Convert each character of the word to lowercase using tolower()
            if (tolower(words[i][j]) == target) {
                // If the character matches the target (after conversion)
                count++;
                break; // No need to check the rest of the characters in this word
            }
        }
    }

    // Output the result
    cout << "Number of words containing 'o' (case-insensitive): " << count << endl;

    return 0;
}
```



```
Number of words containing 'o' (case-insensitive): 3
```

A.P. Baydaa Sulaiman