جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الامن السيبراني
DEPARTMENT OF CYBER SECURITY

SUBJECT:

STRUCTURED PROGRAMMING

CLASS:

1ST STAGE

LECTURER:

DR. ABDULKADHEM A. ABDULKADHEM

LECTURE: (9)

Strings in C++ – C-style and C++ Strings

0. Header Files Used and Their Functions

Before working with strings, it's important to understand the header files involved:

Header File	Purpose & Key Functions
<iostream>	Input/output operations. → `cin`, `cout`, `endl`
<cstring>	C-style string manipulation. → `strlen()`, `strcpy()`, `strcat()`, `strcmp()`
<cstdlib>	String–numeric conversion. → `atoi()`, `atof()`, `itoa()` *(non-standard)*
<cctype>	Character checks and conversions. → `toupper()`, `tolower()`, `isdigit()`, `isspace()`
<string>	C++ `string` class and utilities. → `string`, `.length()`, `.append()`, `.substr()`, operators like +, ==, etc.

- When using `using namespace std;`, there's no need to prefix functions with `std::`.

1. C-Style Strings (Character Arrays)

In C++, a **C-style string** is a one-dimensional array of characters terminated with a special null character `'\0'`.

Syntax:

```
char stringName[size];
```

Examples:

```
char name[11] = "Abdulkadhem";      // Compiler adds '\0' automatically
char str[] = "ABCD";                // Size = 5 (4 characters + '\0')
```

Memory Representation:

Index	0	1	2	3	4
Value	A	B	C	D	\0

2. Reading, Writing, and Processing C-Style Strings

2.1 Reading Strings

- `cin` is used for reading single-word strings (stops at whitespace).
- `cin.getline()` is used for reading multi-word strings.

Example: Read and Print a String

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

int main() {
    char name[50];
    cout << "Enter your full name: ";
    cin.getline(name, 50);  // Reads a full line (including spaces)

    cout << "Hello, " << name << "!" << endl;
    return 0;
}
```

2.2 Print String Character by Character

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

int main() {
    char s[] = "ABCD";
    cout << "String: " << s << endl;

    for (int i = 0; s[i] != '\0'; i++) {
        cout << "s[" << i << "] = " << s[i] << endl;
    }
    return 0;
}
```

2.3 Modify String – Convert to Uppercase

```
#include <iostream>
#include <cctype>  // For toupper()
using namespace std;

int main() {
    char s[] = "abcd";
    cout << "Original: " << s << endl;

    for (int i = 0; s[i] != '\0'; i++) {
        s[i] = toupper(s[i]);
```

```
    }

    cout << "Modified: " << s << endl;
    return 0;
}
```

3. String Functions from <cstring>

The <cstring> library provides functions to operate on C-style strings.

Function	Description	Example
strlen(str)	Returns length (excluding '\0')	strlen("abcd") → 4
strcpy(dest, src)	Copies src into dest	strcpy(b, a) ⟶ b = a
strcat(dest, src)	Concatenates src at end of dest	strcat(a, b) ⟶ a = a + b
strcmp(a, b)	Compares a and b lexicographically	strcmp("A", "B") → -1

Example: Copy and Concatenate Strings

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

int main() {
    char a[] = "Hello";
    char b[20];

    strcpy(b, a);           // b becomes "Hello"
    strcat(b, " World");    // b becomes "Hello World"

    cout << "Concatenated String: " << b << endl;
    return 0;
}
```

4. Conversion Functions from <cstdlib>

The <cstdlib> header provides functions for converting between strings and numbers.

Function	Description	Example
atoi(str)	Converts string to integer	atoi("123") → 123
atof(str)	Converts string to double	atof("3.14") → 3.14
itoa(num, buffer, base)	Converts integer to string (non-standard)	itoa(123, buffer, 10)

Example: String to Integer

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

int main() {
    char numStr[] = "123";
    int num = atoi(numStr);

    cout << "Converted + 1: " << num + 1 << endl;  // Output: 124
    return 0;
}
```

5. C++ Strings (String Class from `<string>`)

The `string` class in C++ provides a more powerful, safe, and convenient way to handle text data.

Declaration and Initialization:

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

int main() {
    string name = "Abdulkadhem";
    cout << "My Name is: " << name << endl;
    return 0;
}
```

Key Features:

- Automatic memory management.
- Overloaded operators (`+`, `=`, `==`, `[]`).
- Built-in member functions for manipulation.

6. Comparing C-Style Strings and C++ Strings

Feature	C-Style String	C++ String (`string`)
Header File	`<cstring>`	`<string>`
Initialization	`char name[] = "Ali";`	`string name = "Ali";`
Memory Management	Manual	Automatic
String Length	`strlen(str)`	`str.length()`

Feature	C-Style String	C++ String (`string`)
Concatenation	`strcat(a, b)`	`a + b` or `a.append(b)`
Comparison	`strcmp(a, b)`	`a == b`, `a < b`, etc.
Safer & Easier	☐	☐

Example: Concatenating and Comparing C++ Strings

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

int main() {
    string first = "Hello";
    string second = "World";
    string message = first + " " + second;

    cout << "Message: " << message << endl;

    if (first == "Hello") {
        cout << "Greeting matched!" << endl;
    }

    return 0;
}
```

7. Summary and Best Practices

- Use **C-style strings** when working with embedded systems or legacy(قديم) C code.
- Use **C++ strings** for modern, safer, and more expressive programming.
- Always ensure space for the null terminator `'\0'` in C-style strings.
- Prefer `cin.getline()` over `cin` when reading strings with spaces.
- Include proper headers: `<cstring>`, `<cstdlib>`, or `<string>` based on usage.

Appendix ملحق

C++ String-Related Functions by Header

1. <cstring> – C-Style String Functions

Function	Description	Example
strlen(str)	Returns length of the string (not including \0)	strlen("ABC") → 3
strcpy(dest, src)	Copies src into dest	strcpy(b, "Hi");
strncpy(dest, src, n)	Copies up to n chars	strncpy(b, a, 3);
strcat(dest, src)	Appends src to dest	strcat(a, b);
strncat(dest, src, n)	Appends up to n chars	strncat(a, b, 2);
strcmp(a, b)	Compares strings, returns 0 if equal	strcmp("Hi", "Hi") → 0
strncmp(a, b, n)	Compares up to n characters	strncmp("Hi", "Ho", 1) → 0
strchr(str, ch)	Finds first occurrence of ch in str	strchr("Hello", 'l') → "llo"
strrchr(str, ch)	Finds last occurrence of ch	strrchr("Hello", 'l') → "lo"
strstr(hay, needle)	Finds substring needle in hay	strstr("Hello", "lo") → "lo"

2. <string> – C++ string Class Functions

Method	Description	Example
s.length()	Returns number of characters	string s = "abc"; s.length() → 3
s.size()	Same as length()	s.size()
s.empty()	Checks if string is empty	s.empty() → false
s.at(i)	Returns character at index i with bounds check	s.at(1) → 'b'
s[i]	Returns character at index i	s[1] → 'b'
s.append(str)	Appends string	s.append("xyz")
s += str	Concatenates	s += "xyz"
s.substr(pos, len)	Extracts substring	s.substr(1, 2) → "bc"
s.find(str)	Finds position of substring	s.find("lo") → 3

Method	Description	Example
`s.replace(pos, len, str)`	Replaces substring	`s.replace(0, 3, "Hi")`
`s.erase(pos, len)`	Removes part of string	`s.erase(1, 2)`
`s.insert(pos, str)`	Inserts string	`s.insert(1, "xyz")`
`s.compare(str)`	Compares with `str`, returns 0 if equal	`s.compare("abc")` → `0`

3. `<cctype>` – Character Functions

Function	Description	Example
`toupper(ch)`	Converts to uppercase if alphabet	`toupper('a')` → `'A'`
`tolower(ch)`	Converts to lowercase	`tolower('A')` → `'a'`
`isalpha(ch)`	Checks if letter	`isalpha('A')` → `true`
`isdigit(ch)`	Checks if digit	`isdigit('9')` → `true`
`isalnum(ch)`	Checks if letter or digit	`isalnum('5')` → `true`
`isspace(ch)`	Checks for whitespace (space, tab, etc.)	`isspace(' ')` → `true`
`isupper(ch)`	Checks if uppercase	`isupper('Z')` → `true`
`islower(ch)`	Checks if lowercase	`islower('z')` → `true`

4. `<cstdlib>` – String <--> Number Conversion

Function	Description	Example
`atoi(str)`	Converts C-string to integer	`atoi("123")` → `123`
`atof(str)`	Converts C-string to double	`atof("3.14")` → `3.14`
`atol(str)`	Converts C-string to long	`atol("999999")`
`itoa(val, buf, base)`	Converts int to C-string *(non-standard)*	`itoa(123, buf, 10)` → `"123"`

Notes:

- Functions in `<string>` are used with **`string` objects** (modern C++).
- Functions in `<cstring>`, `<cstdlib>`, and `<cctype>` are for **C-style strings** (`char[]`).
- `itoa()` is **not standard**, but available in many compilers (like Visual C++ and GCC).