3.2 Character Arrays and Strings:

A **character array** is a collection of characters stored in contiguous memory locations.
Used to store text in older C++ programs or for low-level string manipulation.

Declaring a string in c-style

String can be declared like one-dimensional arrays

```
char str[30];
char text[80];
```

String initialization

char name[20] = “Hello";

Internally, it looks like this in memory:
‘H’ ‘e’ ‘l’ ‘l’ ‘o’ ‘\0’

C-style string is simply an **array of characters** that ends with a **null character** ('\0').

1. Compile Time Initialization

(iv) Array initialization with a string: - Consider the declaration with string initialization.

Ex:-

```
char b[]="COMPUTER";
```

The array b is initialized as shown in figure.

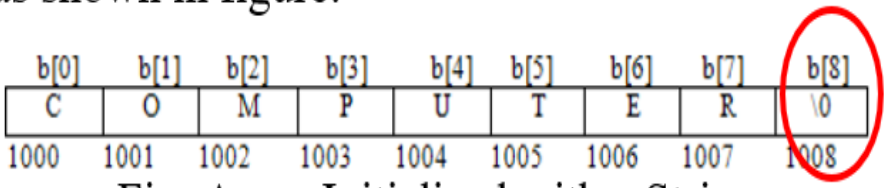


Fig: Array Initialized with a String

Eventhough the string "COMPUTER" contains 8 characters, because it is a string. It always ends with null character. So, the array size is 9 bytes (i.e., string length 1 byte for null character).

Ex:-

```
char b[9]="COMPUTER"; // correct
char b[8]="COMPUTER"; // wrong
```

- If we initialize char s[2] = "ab";
- Only 2 bytes of space — the null terminator doesn't fit.
- Compiler warning: initializer-string for char array is too long

Correct way:

- char s[3] = "ab"; // or char s[] = "ab";
- Compiler adds '\0' automatically.

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

int main() {
    char name[6] = {'A',’B’,’c',’d',’e','\0'};
    cout << "Name: " << name << endl;
    return 0;
}
```

Must include \0 at the end to indicate the string's termination.

Array size = number of characters + 1 (for \0).

Reading strings

There are two common ways to read strings in C++:

1. Using cin — reads a single word (stops at space).

```
char str[20];
cin >> str;  // Reads a word (stops at whitespace)
```

2. Using getline() — reads a full line including spaces.

```
cin.getline(str, 20); // Reads a full line including spaces
```

Reading a string using cin

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

int main() {
    char name[50];
    cout << "Enter a text: ";
    cin>>name;
    cout << "You entered: " << name;
}
```

```
Enter a text: Hello world
You entered: Hello
```

reading using getline

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

int main() {
    char name[50];
    cout << "Enter a text: ";
    cin.getline(name, 50);
    cout << "You entered: " << name;
}
```

getline reads the entire line including spaces, until:

- a newline ('\n') is found,

Or

- the specified size limit is reached (here 50)

```
Enter your full name: Hello world
You entered: Hello world
```

Program to count characters in text

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

int main() {
    char text[100];       // Array to store text (max 99 chars + '\0')
    int count = 0;

    cout << "Enter a line of text: ";
    cin.getline(text, 100);   // Reading  entire line including spaces

    // Manual counting loop
    for (int i = 0; text[i] != '\0'; i++) {
        count++;              // Incrementing  count for each character
    }

    cout << "Number of characters: " << count << endl;

    return 0;
}
```

```
Enter a line of text: this is a long text
Number of characters: 19
```

Modified program to exclude spaces

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

int main() {
    char text[100];        // Array to store text (max 99 chars + '\0')
    int count = 0;

    cout << "Enter a line of text: ";
    cin.getline(text, 100);   // Reading  entire line including spaces

    // Manual counting loop
    for (int i = 0; text[i] != '\0'; i++) {
      if(text[i]!=' ')   //ignoring spaces
          count++;              // Incrementing  count for each character
    }

    cout << "Number of characters: " << count << endl;

    return 0;
}
```

```
Enter a line of text: this is a long text
Number of characters: 15
```

WAP to read a string, copy it to another string, and display the copied string.

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

int main() {
    char source[100], destination[100];
    int i = 0;

    cout << "Enter source string: ";
    cin.getline(source, 100);

    while (source[i] != '\0') {
        destination[i] = source[i];
        i++;
    }
    destination[i] = '\0';

    cout << "Copied string: " << destination << endl;
    return 0;
}
```

```
Enter source string: Computer Science
Copied string: Computer Science
```

- Write a program to accept two strings and display the result after appending the second string to the end of the first string.

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

int main() {
    char str1[100], str2[100];
    int i = 0, j = 0;

    cout << "Enter first string: ";
    cin.getline(str1, 100);
    cout << "Enter second string: ";
    cin.getline(str2, 100);

    while (str1[i] != '\0')
        i++;

    while (str2[j] != '\0') {
        str1[i] = str2[j];
        i++;
        j++;
    }
    str1[i] = '\0';

    cout << "Concatenated string: " << str1 << endl;
    return 0;
}
```

```
Enter first string: Hello
Enter second string: World
Concatenated string: HelloWorld
```

WAP to find and display the position of the first occurrence of a given character in a string.

```
Enter a string: Computer Engineering
Enter a character to search: e
Character 'e' found at position 7
```

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

int main() {
    char str[100], ch;
    int i = 0, pos = -1;

    cout << "Enter a string: ";
    cin.getline(str, 100);
    cout << "Enter a character to search: ";
    cin >> ch;

    while (str[i] != '\0') {
        if (str[i] == ch) {
            pos = i;
            break;
        }
        i++;
    }

    if (pos != -1)
        cout << "Character '" << ch << "' found at position " << pos + 1 << endl;
    else
        cout << "Character '" << ch << "' not found." << endl;

    return 0;
}
```

WAP to compare two strings and display 0 if they are equal, a negative value if the first string appears earlier, and a positive value if it appears later in dictionary order

Sample Input-output:

Case 1:

Enter first string: apple

Enter second string: apple

Comparison result: 0

-> Both strings are equal.

Case 2:

Enter first string: apple

Enter second string: banana

Comparison result: -1

→ First string appears earlier in dictionary order.

Case 3:

Enter first string: mango

Enter second string: grapes

Comparison result: 6

→ First string appears later in dictionary order.

```
cout << "Enter first string: ";
cin.getline(str1, 100);

cout << "Enter second string: ";
cin.getline(str2, 100);

// Compare character by character
while (str1[i] != '\0' && str2[i] != '\0') {
    if (str1[i] != str2[i]) {
        result = str1[i] - str2[i];  // difference of ASCII values
        break;
    }
    i++;
}
// If one string ended but other didn't
if (result == 0)
    result = str1[i] - str2[i];

cout << "Comparison result: " << result << endl;

if (result == 0)
    cout << "Both strings are equal." << endl;
else if (result < 0)
    cout << "First string appears earlier in dictionary order." << endl;
else
    cout << "First string appears later in dictionary order." << endl;
```

<cstring> header file

- **<cstring>** is a **C++ standard header file** that provides a set of **functions for manipulating C-style strings**

Few string function of <cstring> header file

Function	Purpose / Description	Example	Sample Output / Meaning
strlen(str)	Returns number of characters in the string (excluding '\0')	strlen("Hello")	5
strcpy(dest, src)	Copies one string into another	strcpy(b, a)	b = "Hello"
strncpy(dest, src, n)	Copies first n characters only	strncpy(b, a, 3)	b = "Hel"
strcat(dest, src)	Appends one string at the end of another	strcat(a, b)	a = "HelloWorld"
strncat(dest, src, n)	Appends first n characters of one string to another	strncat(a, b, 3)	a = "HelloWor"
strcmp(s1, s2)	Compares two strings lexicographically	strcmp("abc", "abd")	< 0 (since "abc" < "abd")
strchr(str, ch)	Finds the first occurrence of a character in a string	strchr("Hello", "l")	Returns "llo" (address of first l)
strstr(str1, str2)	Finds the first occurrence of a substring in another string	strstr("HelloWorld", "World")	Returns "World" (pointer to substring)

Demonstration of string functions

```
Enter first string: Hello
Enter second string: world

---- STRING FUNCTION RESULTS ----
1. Length of first string (strlen): 5
   Length of second string: 5
2. After strcpy(result, str1): Hello
3. After strncpy(result, str2, 3): wor
4. After strcat(str1, str2): Helloworld
5. After strncat(str1, str2, 3): Hellowor
6. Result of strcmp(str1, str2): -1
```

```
cout << "Enter first string: ";
cin.getline(str1, 50);

cout << "Enter second string: ";
cin.getline(str2, 50);

cout << "\n---- STRING FUNCTION RESULTS ----\n";

cout << "1. Length of first string (strlen): " << strlen(str1) << endl;
cout << "   Length of second string: " << strlen(str2) << endl;

strcpy(result, str1);
cout << "2. After strcpy(result, str1): " << result << endl;

strncpy(result, str2, 3);
result[3] = '\0';  // Ensuring string ends properly
cout << "3. After strncpy(result, str2, 3): " << result << endl;

strcpy(result, str1);        // Copy str1 into result first
strcat(result, str2);        // Append str2
cout << "4. After strcat(str1, str2): " << result << endl;

strcpy(result, str1);        // Copy str1 again
strncat(result, str2, 3);    // Append first 3 chars of str2
cout << "5. After strncat(str1, str2, 3): " << result << endl;


int cmp = strcmp(str1, str2);
cout << "6. Result of strcmp(str1, str2): " << cmp << endl;
```

Demo of strchr function

```
char text[100];
char ch;

cout << "Enter a string: ";
cin.getline(text, 100);

cout << "Enter a character to search: ";
cin >> ch;

char* pos = strchr(text, ch);

if (pos != NULL)
    cout << "Character '" << ch << "' found at position "
         << (pos - text + 1) << " in \"" << text << "\"" << endl;

else
   cout << "Character '" << ch << "' not found in the string." << endl;
```

```
Enter a string: computer
Enter a character to search: o
Character 'o' found at position 2 in "computer"
```

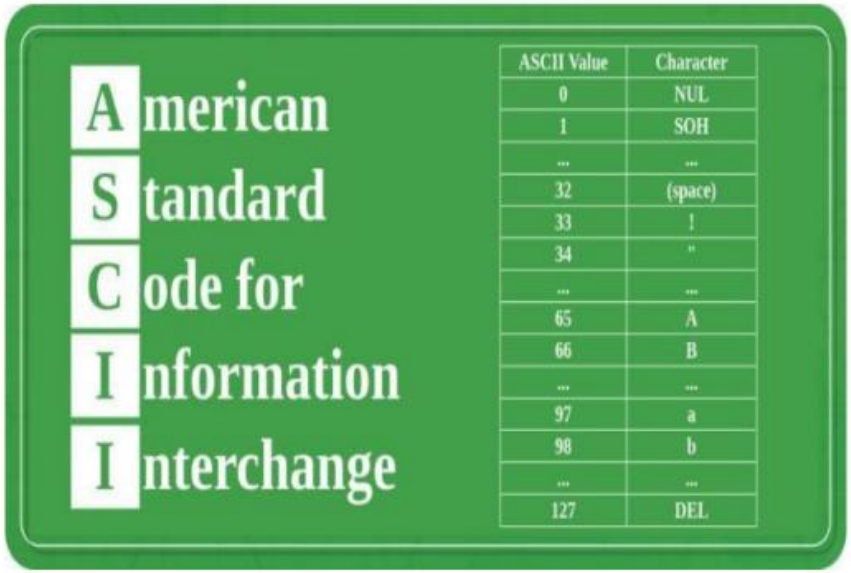
Demo of strstr function

```
char text[100], sub[50];

cout << "Enter the main string: ";
cin.getline(text, 100);

cout << "Enter the substring to search: ";
cin.getline(sub, 50);

char* pos = strstr(text, sub);

if (pos != NULL)
    cout << "Substring \"" << sub << "\" found at position "
         << (pos - text + 1) << " in \"" << text << "\"" << endl;
else
    cout << "Substring \"" << sub << "\" not found in the string." << endl;
```

```
Enter the main string: Computer
Enter the substring to search: pu
Substring "pu" found at position 4 in "Computer"
```

WAP to capitalize all the letters of the string

ASCII codes



```
char str[100];
cout << "Enter a string: ";
cin.getline(str, 100);
// Convert each character to uppercase
for (int i = 0; i < strlen(str); i++) {
    if (str[i] >= 'a' && str[i] <= 'z')    // check if lowercase
        str[i] = str[i] - 32;              // convert to uppercase
}

cout << "String in uppercase: " << str << endl;
```

```
Enter a string: Hello World
String in uppercase: HELLO WORLD
```

<cctype> header file

<cctype> header provides predefined functions like isalpha(), isdigit(), toupper(), etc.,to check type of character and to convert characters between cases.

<cctype> header file

Function	Purpose / Description	Example	Output / Meaning
isalpha(ch)	Checks if character is an alphabet (A–Z or a–z)	isalpha('A')	1 (true)
isdigit(ch)	Checks if character is a digit (0–9)	isdigit('9')	1 (true)
isalnum(ch)	Checks if character is alphanumeric (A–Z, a–z, 0–9)	isalnum('#')	0 (false)
isspace(ch)	Checks if character is a whitespace (space, tab, newline)	isspace(' ')	1 (true)
islower(ch)	Checks if character is lowercase	islower('g')	1 (true)
isupper(ch)	Checks if character is uppercase	isupper('G')	1 (true)
toupper(ch)	Converts lowercase to uppercase	toupper('b')	'B'
tolower(ch)	Converts uppercase to lowercase	tolower('H')	'h'
ispunct(ch)	Checks if character is punctuation (!, ?, etc.)	ispunct('!')	1 (true)
isprint(ch)	Checks if character is printable	isprint('\n')	0 (false)
iscntrl(ch)	Checks if character is a control character	iscntrl('\n')	1 (true)
isxdigit(ch)	Checks if character is a hexadecimal digit (0–9, A–F, a–f)	isxdigit('F')	1 (true)

WAP to reverse the string and check whether it a palindrome (use strlen, strcmp)

WAP to Count Digits, Alphabets, and Special Characters (use isalpha(), isdigit() function)

Input: Hello123@C++
Output:

Alphabets: 6

Digits: 3

Special characters: 3

C++ String

- Strings in C++ are no longer just a sequence of characters but an object of the std::string class from the Standard Template Library (STL).

Declare and initializing string

Include the header file

#include <string>

//This brings in the definition of the ***std::string*** *class*
//from the ***Standard Template Library (STL***

string name; //declaring the string

string text="Hello" //declaring and initializing string

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

int main() {

    // Initializing a string directly
    string str = "Hello World";

    // Printing the string
    cout << str << endl;

    return 0;
}
```

Reading string using getline() of string class

- **getline()** is a **C++ input function** that reads an entire line of text — including spaces — into a **std::string** variable.
- It is defined in the **<string>** header and belongs to the **std namespace**.

Syntax:

getline(cin, string_variable);

C style vs C++ string

Feature	C-style String (char[])	C++ std::string
Type	Array of characters ending with '\0'	Object (class) from the Standard Library
Header file	<cstring> or <string.h>	<string>
Memory	Fixed size (must be declared)	Grows/shrinks automatically
Operations	Done using library functions (strcpy, strlen, etc.)	Done using operators (+, =, ==) and built-in methods
Safety	Prone to overflow and manual errors	Memory-safe and flexible
Input	cin.getline(name, size)	getline(cin, name)
Example declaration	char name[20] = "Alice";	string name = "Alice";

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

int main() {
    string name;
    cout << "Enter your full name: ";
    getline(cin, name);       // reads full line including spaces
    cout << "You entered: " << name << endl;
    return 0;
}
```

String operations

String length

- The number of characters in a string can be found using size() or length().

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

int main() {
    string str = "Hello World";

    // Using size()
    cout << "Length using size(): " << str.size() << endl;

    // Using length()
    cout << "Length using length(): " << str.length() << endl;

    return 0;
}
```

```
Length using size(): 11
Length using length(): 11
```

Concatenation of Strings

- Strings can be joined using the + operator or the append() function.
- The + operator creates a new string, while append() modifies the existing string in place.

```
int main() {
    string str1 = "Hello";
    string str2 = "World";

    string result1 = str1 + " " + str2;  // joins strings into a new string
    cout << "Using + operator: " << result1 << endl;


    string str3 = "Good";
    string str4 = "Morning";

    str3.append(" ");       // add space
    str3.append(str4);      // append str4 to str3 (modifies str3 itself)

    cout << "Using append() function: " << str3 << endl;


    return 0;
}
```

Modifying a String

- Characters of a string can be added with .push_back(), removed with .pop_back(), or altered using .insert() and .erase().

Function	Purpose	Action
.push_back(ch)	Adds a single character at the end of the string	Append one character
.pop_back()	Removes the last character of the string	Deletes last character
.insert(pos, str)	Inserts one or more characters at a specific position	Adds characters in the middle
.erase(pos, len)	Removes one or more characters starting from a given position	Deletes part of the string

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

int main() {
    string str = "Hello World";
    // Adding a character at the end
    str.push_back('!');
        cout << "After push_back: " << str << endl;

    // Removing the last character
    str.pop_back();
    cout << "After pop_back: " << str << endl;

    // Inserting a substring
    str.insert(5, " C++");
    cout << "After insert: " << str << endl;

    // Erasing part of the string
    str.erase(5, 4);
    cout << "After erase: " << str << endl;

    return 0;
}
```

```
After push_back: Hello World!
After pop_back: Hello World
After insert: Hello C++ World
After erase: Hello World
```

Substring Extraction

- The .substr(pos,len) is used to extract a part of a string, where pos means the starting position and len means how many characters you want to copy.
- This function creates a new string containing the selected portion, starting at pos and copying len characters.

Substring Extraction

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

int main() {
    string str = "Hello World";

    // Extract "Hello"
    string sub1 = str.substr(0, 5);
    cout << "Substring 1: " << sub1 << endl;

    // Extract "World"
    string sub2 = str.substr(6, 5);
    cout << "Substring 2: " << sub2 << endl;

    return 0;
}
```

Searching substring in a String

- The find() function is used to search for a substring inside a string. If found, it returns the index (position) where the substring starts; if not, it returns a special value (npos).

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

int main() {
    string str = "Hello World";

    size_t pos = str.find("Wow");

    if (pos != string::npos){
        cout << "\"World \" found at index: " << pos << endl;
    }
    else
        cout<<"not found"<<endl;

    return 0;
}
```

```
"World " found at index: 6
```

- npos is a **constant value** defined inside the C++ std::string class.
- It represents **“no position**

Substr

substr() function is used to extract a substring from the given string. It generates a new string with its value initialized to a copy of a sub-string of the given string.

```
string s = "World";

string sub = s.substr(3, 2);
```

Output: `ld`

String compare

```
int main() {
    string s1 = "Apple";
    string s2 = "Banana";
    string s3 = "Apple";

    cout << "Comparing s1 and s2: " << s1.compare(s2) << endl;
    cout << "Comparing s1 and s3: " << s1.compare(s3) << endl;
    cout << "Comparing s2 and s1: " << s2.compare(s1) << endl;

    return 0;
}
```

```
Comparing s1 and s2: -1
Comparing s1 and s3: 0
Comparing s2 and s1: 1
```

String compare

```
string s1 = "Apple";
string s2 = "Banana";
string s3 = "Apple";

if( s1==s3)
    cout<< "s1 and s3 both are equal"<<endl;

if( s1<s2)
    cout<< "s1 < s2"<<endl;

if( s2 > s3)
    cout<< "s2 > s3 "<<endl;
```

```
s1 and s3 both are equal
s1 < s2
s2 > s3
```

String insert

- **string insert()** function is used to insert characters or a string at the given position of the string.

Syntax:

- str.insert(pos, c);

```
int main() {
    string s = "I love programming";
    s.insert(7, "C++ ");
    cout << "After insert: " << s << endl;
    return 0;
}
```

```
After insert: I love C++ programming
```

String erase

- The string erase() function is a built-in function of the string class that is used to erase the whole or part of the string, shortening its length.
- Syntax: str.erase(pos, len)

  erases len characters from postion

```
int main() {
    string s = "I love C++ programming";
    // Erase "C++ "
    s.erase(7, 4); // start at index 7, remove 4 chars
    cout << "After erase: " << s << endl;
    return 0;
}
```

```
After erase: I love programming
```

String replace

- The **replace()** function of string class function is used to replace a single or multiple characters from a given index
- Syntax: str1.**replace**(pos, n, str2
- // Replaces len characters starting from pos with the contents of str.

```
int main() {
    string s = "I love Java programming";
    // Replace "Java" with "C++"
    s.replace(7, 4, "C++");
    cout << "After replace: " << s << endl;
    return 0;
}
```

```
After replace: I love C++ programming
```

1. Concatenation:

String concatenation is the process of **joining two or more strings together** to form a single continuous string.

```
string a = "Data";
string b = "Science";
string c = a + " " + b;
```

2. Length of String

The *length of a string* refers to the **number of characters** it contains, including letters, digits, symbols, and spaces, but **excluding the null terminator ('\0')** in C-style strings.

```
cout << a.length();
```

3. **Substring Extraction**

A **substring** is a **portion or part of a larger string**, obtained by extracting a sequence of consecutive characters from it.

```
string s = "Artificial";
string sub = s.substr(0, 5);  // "Artif"
```

Sample Program

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

int main() {
    // String Declaration
    string str1 = "Data";
    string str2 = "Science";

    string result = str1 + " " + str2;
    cout << "Concatenated String: " << result << endl;

    cout << "Length of Concatenated String: " <<
result.length() << endl;

    string sub = result.substr(0, 4);
    cout << "Substring (first 4 characters): " << sub << endl;

    return 0;
}
```

4. Comparison

String comparison determines the **lexicographical (dictionary) order** or equality between two strings.

In C++, the ==, <, >, <=, >=, and != operators can be used directly with the string class.

```
string x = "abc", y = "abd";
if (x == y) cout << "Equal";
else if (x < y) cout << "x comes before y";
```

5. Insertion and Deletion

Insertion means **adding new characters or a substring** at a specific position within an existing string.

Deletion means **removing characters** from a string starting at a given position.

```
string s = "Machine";
s.insert(7, " Learning");  // "Machine Learning"
s.erase(7, 9);             // removes " Learning“
```

6. Finding Substring

```
string s = "Data Science";
int pos = s.find("Science");
if (pos != string::npos)
    cout << "Found at " << pos;
```

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Computer";
    string s2 = "Compute";
    if (s1 == s2)
        cout << "Strings are equal" << endl;
    else if (s1 > s2)
        cout << s1 << " is greater than " << s2 << endl;
    else
        cout << s2 << " is greater than " << s1 << endl;
    string text = "Programming in C++ is interesting";
    string key = "C++";
    size_t pos = text.find(key);
    if (pos != string::npos)
        cout << "Substring '" << key << "' found at position: " << pos << endl;
    else
        cout << "Substring not found" << endl;
    text.insert(12, "and Java ");
    cout << "After insertion: " << text << endl;
    text.erase(12, 8); // remove "and Java "
    cout << "After deletion: " << text << endl;
    return 0;
}
```

Searching in a String

The find() function is used to search for a substring inside a string. If found, it returns the index (position) where the substring starts; if not, it returns a special value (npos)

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

int main() {
    string str = "Hello students;

    size_t pos = str.find(“students");

    if (pos < str.size()) {
        cout << "\“students\" found at index: " << pos << endl;
    }

    return 0;
}
```