**Batch:\_\_\_\_B1\_\_\_\_ Roll No.:\_16010122109\_**

**Experiment No. 1**

**Grade: AA / AB / BB / BC / CC / CD /DD**

|  |
| --- |
| **Title:**  Implementation of Abstract Data Type |

**Objective:** Implementation of ADT without using any standard library function

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| **CO 1** | Explain the different data structures used in problem solving. |

**Books/ Journals/ Websites referred:**

**Abstract**:-

*(Define ADT. Why are they important in data structures?)*

Abstract Data Types (ADTs) stores data and allow various operations on the data to access and change it.A mathematical model, together with various operations defined on the model. An ADT is a collection of data and associated operations for manipulating that data.ADTs support abstraction, encapsulation, and information hiding.ADT is a data type that focuses on what it does by ignoring how it does.

ADT in data structures are important:

Specification: The supported operations of the ADT

Implementation: Data structures and actual coding to meet the specification

It helps in abstraction and encapsulation.

Encapsulation and Information Hiding: ADTs encapsulate the data and operations related to a data structure. The implementation details are hidden from the user, allowing developers to use the data structure without worrying about how it is internally organized. This information hiding enhances security, modularity, and maintainability.

Abstraction: ADTs abstract away the complexity of data structures. Users only need to understand how to use the operations, making it easier to work with complex data structures without getting bogged down in implementation details

It is important for large-scale programming.

Enables code reusability as the same data structure can be used in multiple programs with the same interface.

**Abstract Data Type for \_\_\_Strings\_\_\_\_**

*[for chosen data type write value definition and operator definition)*

Chosen Data Type:

The chosen data type for representing strings is string.

Value Definitions:

In the given code, the values for the string str1 and str2are taken as input from the user using scanf.

Operator Definitions:

The following operations are defined for the String data type:

int stringLength(const char \*str) {

int length = 0;

while (str[length] != '\0') {

length++;

}

return length;

}

Value Definition:

abstract Integer StringLength (StringType String)

Precondition: None (A string may contain n characters where n=>0)

Postcondition: Stringlength = NumberOfCharacters(String)

The length function takes a string str as input and returns the length .

void stringCopy(char \*dest, const char \*src) {

int i = 0;

while (src[i] != '\0') {

dest[i] = src[i];

i++;

}

dest[i] = '\0';

}

Value Definition:

abstract StringType StringConcat(StringType String1, StringType String2)

Precondition: None

Postcondition: StringConcat=String1+String2

The copy function takes a string str as input and returns the copied value of the given string.

int stringCompare(const char \*str1, const char \*str2) {

int i = 0;

while (str1[i] == str2[i]) {

if (str1[i] == '\0')

break;

i++;

}

char cmp=str1[i] - str2[i];

if (cmp == 0)

printf("The two strings are equal.\n");

else if (cmp < 0)

printf("The string1 is less than the string2.\n");

else

printf("The string1 is greater than the string2.\n");

return cmp;

}

Value Definition:

abstract StringType StringCompare( StringType String1, StringType String2)

Precondition: None

Postcondition: StringCompare= Same if strings are equal,

StringCompare= Greater/ Lesser if they are unequal (Depending upon String1 and String2 ) .

The compare function takes two strings str1 and str2 as input and returns whether the str1 is greater , lesser or equal to the str2 .

void stringConcat(char \*dest, const char \*src) {

int dest\_len = stringLength(dest);

int i = 0;

while (src[i] != '\0') {

dest[dest\_len + i] = src[i];

i++;

}

dest[dest\_len + i] = '\0';

}

Value Definition:

abstract StringType StringCopy(StringType String, StringType String1)

Precondition: None

Postcondition: StringCopy: String= String1

The concat function takes two strings str1 and str2 as input and returns the concat(adding two strings) of the string .

**Implementation Details:**

**1. Explain the Importance of the approach followed by you**

The above c-program code deals with string functions: StringLength, StringCopy, StringCompare, and StringConcat.The importance of each of these functions and the approach :

Importance:

The approach followed in the code is important because it demonstrates how to use strings manually without relying on the built-in string functions provided by the C Standard Library (e.g: strlen, strcpy, strcmp, strcat). While these standard functions are efficient and widely used, implementing the basic string manipulation functions from scratch helps in understanding the underlying concepts and algorithms involved.Having a basic understanding of how strings work and implementing these functions can be helpful.

Approach:

Input: The program prompts the user to enter two strings using scanf. This allows the user to interact with the program and provide input.

StringLength: The stringLength function calculates the length of a given string by counting the number of characters till it encounters the null terminator '\0'. The null terminator indicates the end of a C-style string. This function is allows us to determine the length of a string, which is essential for various string operations and memory management.

StringCopy: The stringCopy function copies the contents of one string (source) to another string (destination). It is commonly known as the strcpy function in the Standard C Library. This function is used when you need to duplicate a string .

StringCompare: The stringCompare function compares two strings character by character and determines whether they are equal or which one is greater or lesser. It returns an integer that indicates the difference between the first differing characters in the strings. This function is used for sorting strings, searching for strings in a collection. The approach used here is it that it compares characters one by one until it finds a mismatch or reaches the end of one of the strings.

StringConcat: The stringConcat function concatenates (appends) one string to the end of another string. It is commonly known as the strcat function in the C Standard Library. This function is used when you need to combine strings. The approach followed here calculates the length of the destination string and then appends the source string after it.

Output: The program displays the results of each operation using printf, providing the user with the computed string functions.

**Program code and Output screenshots:**

**Code:**

**#include <stdio.h>**

**int stringLength(const char \*str) {**

**int length = 0;**

**while (str[length] != '\0') {**

**length++;**

**}**

**return length;**

**}**

**void stringCopy(char \*dest, const char \*src) {**

**int i = 0;**

**while (src[i] != '\0') {**

**dest[i] = src[i];**

**i++;**

**}**

**dest[i] = '\0';**

**}**

**int stringCompare(const char \*str1, const char \*str2) {**

**int i = 0;**

**while (str1[i] == str2[i]) {**

**if (str1[i] == '\0')**

**break;**

**i++;**

**}**

**char cmp=str1[i] - str2[i];**

**if (cmp == 0)**

**printf("The two strings are equal.\n");**

**else if (cmp < 0)**

**printf("The string1 is less than the string2.\n");**

**else**

**printf("The string1 is greater than the string2.\n");**

**return cmp;**

**}**

**void stringConcat(char \*dest, const char \*src) {**

**int dest\_len = stringLength(dest);**

**int i = 0;**

**while (src[i] != '\0') {**

**dest[dest\_len + i] = src[i];**

**i++;**

**}**

**dest[dest\_len + i] = '\0';**

**}**

**int main() {**

**char str1[100], str2[100], result[200];**

**printf("Enter the string1: ");**

**scanf("%s", str1);**

**printf("Enter the string2: ");**

**scanf("%s", str2);**

**int length = stringLength(str1);**

**printf("Length of the first string: %d\n", length);**

**stringCopy(result, str1);**

**printf("Copied string (string 1): %s\n", result);**

**stringCompare(str1, str2);**

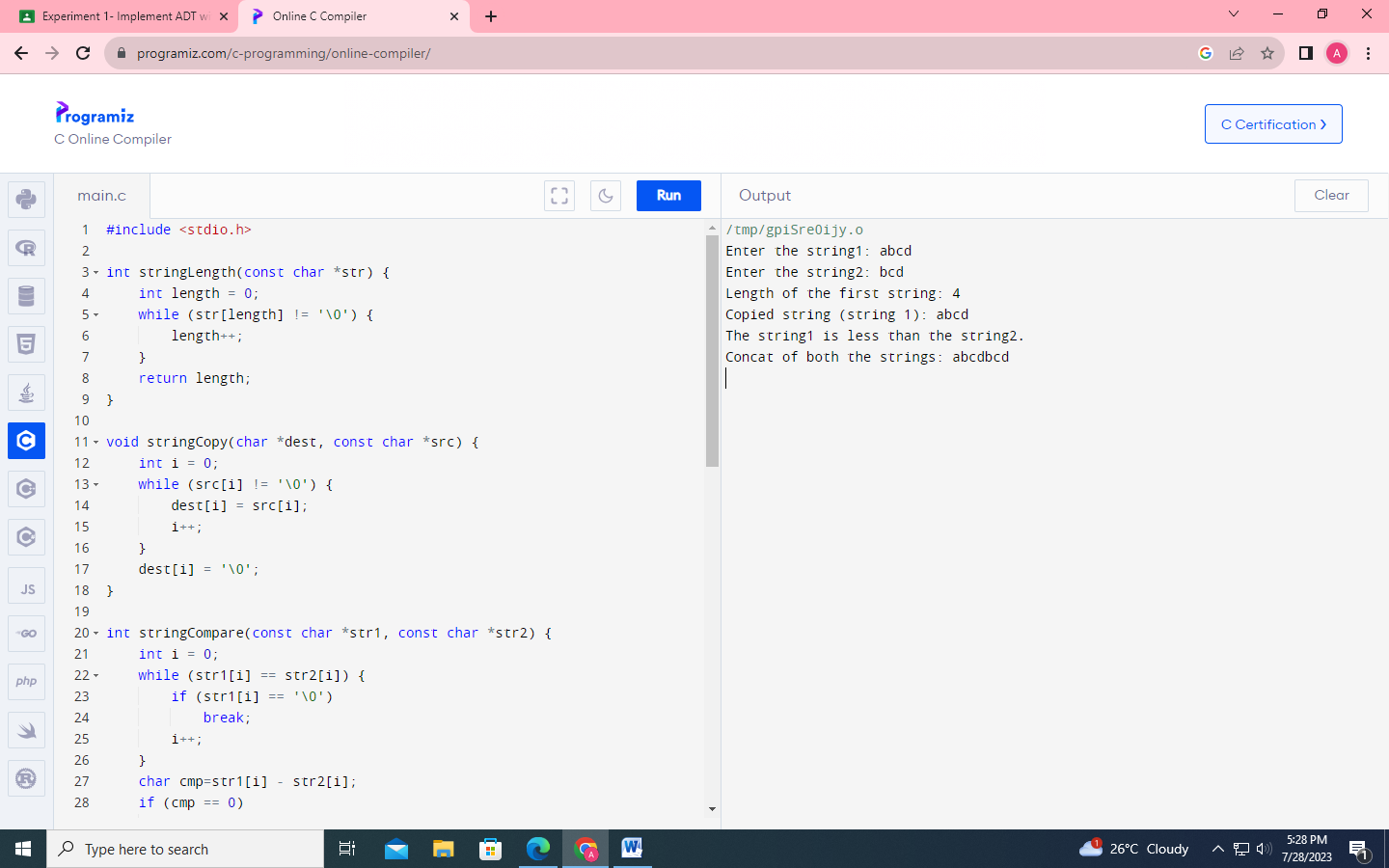
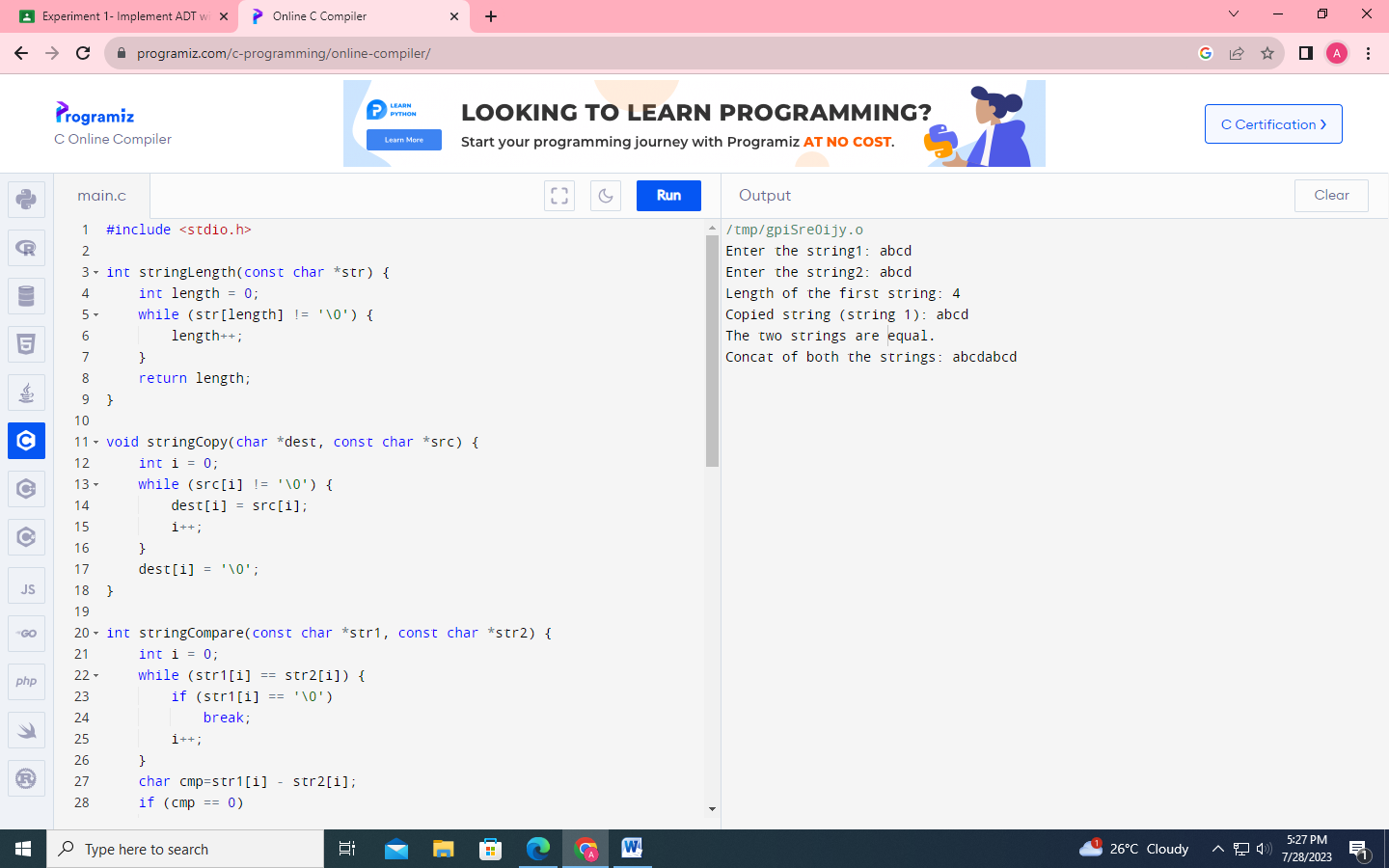
**stringConcat(str1, str2);**

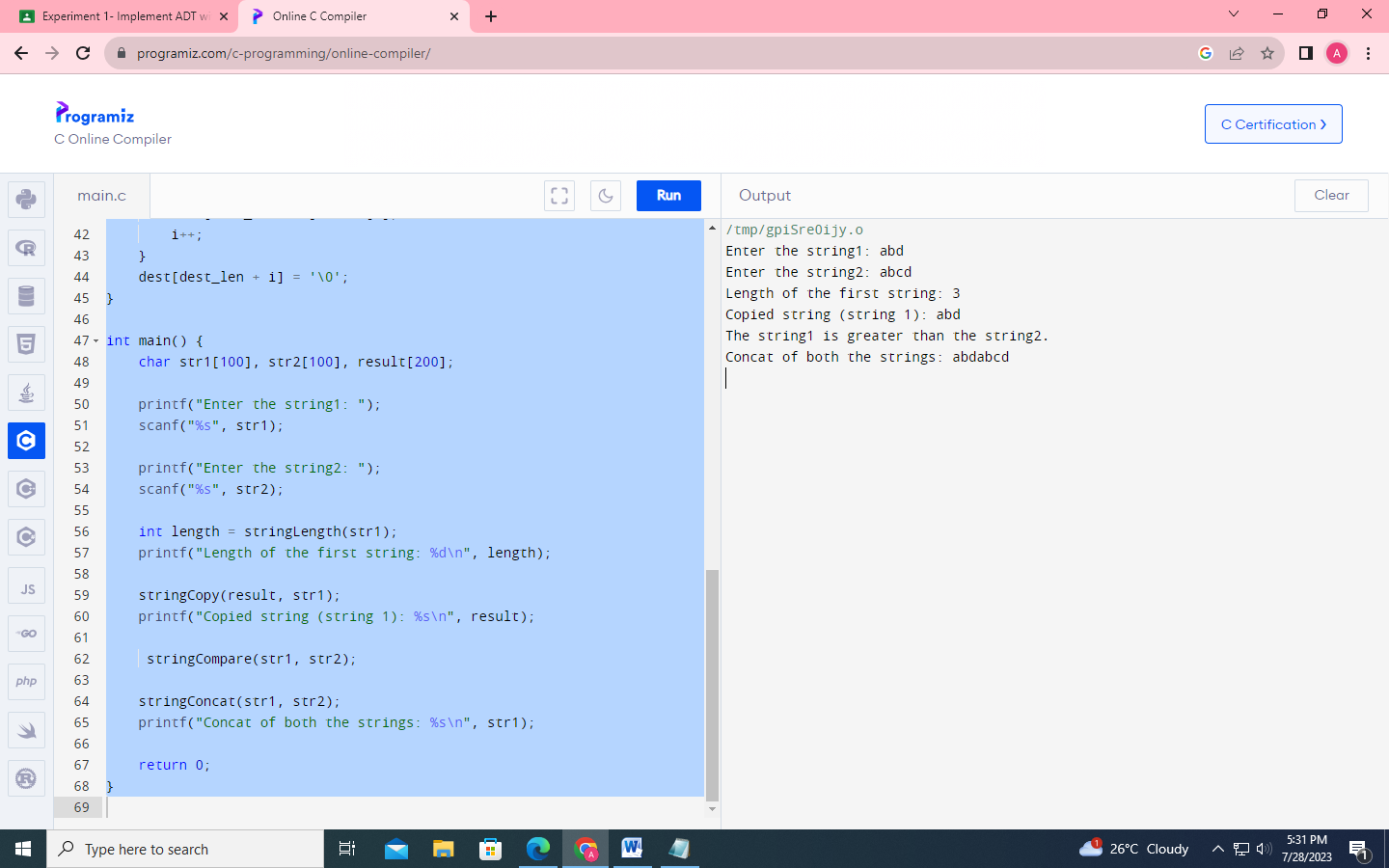
**printf("Concat of both the strings: %s\n", str1);**

**return 0;**

**}**

**Output:**

****

****

**Conclusion:-**

**Therefore, we learned how to implement abstract data structures.**