K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent college of SomaiyaVidyavihar University)

Batch:___C1_______	**Roll No.:_16010122257_____________**
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:

1. https://www.comp.nus.edu.sg/~stevenha/cs1020e/lectures/L5%20-%20ADT.pdf
2. *Fundamentals Of Data Structures In C* – Ellis Horowitz, SatrajSahni, Susan Anderson-Fred
3. *An Introduction to data structures with applications* – Jean Paul Tremblay, Paul G. Sorenson
4. *Data Structures A Pseudo Approach with C* – Richard F. Gilberg& Behrouz A. Forouzan
5. *https://ece.uwaterloo.ca/~dwharder/aads/Abstract_data_types/Linear_ordering/String/#:~:text=The%20most%20common%20and%20basic,has%20the%20bit%20representation%2000000000).*

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.Itis a type (or class) for objects whose behavior is defined by a set of values and a set of operations. The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented.They are important for large-scale programming. They package data structures and operations on them, hiding internal details.

Abstract Data Type for string

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

ADT-string

valuedefinition

sequence of characters

Operations:

– StringLength

– StringCompare

– StringConcat

– StringCopy

Example ADT : String

• Value Definition

Abstract TypedefStringType<<Chars>>

Condition: None (A string may contain n

characters where n=>0)

Example ADT : String

Operator Definition

1. abstract Integer StringLength (StringType

String)

Precondition: None (A string may contain n

characters where n=>0)

Postcondition: Stringlength=

NumberOfCharacters(String)

Example ADT : String

Operator Definition

2. abstractStringTypeStringConcat(

StringType String1, StringType String2)

Precondition: None

Postcondition: StringConcat=

String1+String2 / All the characters in

Strings1 immediately followed by all the

characters in String2 are returned as result.

Example ADT : String

Operator Definition

3. abstract Boolean StringCompare(StringType

String1, StringType String2)

Precondition: None

Postcondition: StringCompare= True if

strings are equal, StringCompare= False if

they are unequal . (Function returns 1 if

strings are same, otherwise zero)

Example ADT : String

Operator Definition

4. abstractStringTypeStringCopy(

StringType String1, StringType String2)

Precondition: None

Postcondition: StringCopy: String1= String2 /

All the characters in Strings2 are

copied/overwritten into String1.

Implementation Details:

1. **Enlist all the Steps followed and various options explored.**
   - String Structure:Defined a struct ‘String’ with a character array ‘data’ to hold the string content and an integer ‘length’ to track the string’s length.
   - Initialization:Implemented a function ‘initializeString’ for initializing ‘String’ struct with a given text.This function copies characters from the input text to the ‘data’ array until the end of the text or until the maximum string length is reached.
   - Concatenation:Create a function ‘concatenateStrings’ to concatenate the contents of one String struct onto another. This function appends characters from the source string to the end of the destination string until the end of the source string is reached or until the destination string's maximum length is nearly reached.
   - Printing: Implemented a function printString for printing the content of a String struct.

- Clearing: Created a function clearString for resetting the content and length of String struct by filling the data array with null characters and setting the length to zero.

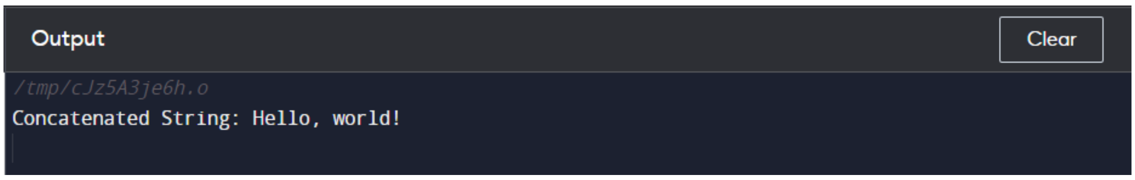
2. **Explain your program logic and methods used.**
    - The program starts by defining a structure ‘String’ for holding string data and length information.It uses a fixed-size character array to store the string content and an integer to track the string’s length.
    - The ‘initializeString’ function initializes a String struct with a given text. It iterates through the input text and copies characters to the data array until the end of the text or the maximum string length is reached. It also sets the length member of the struct.
    - The concatenateStrings function takes two String structs: a source and a destination. It appends the content of the source string to the end of the destination string. The function iterates through the source string's characters and appends them to the destination string's data array, updating the destination string's length accordingly.
    - The ‘printString’ function prints the characters of a String struct's data array until it encounters a null character.
    - The clearString function resets the content of a String struct by filling its data array with null characters and setting the length to zero.

3. **Explain the Importance of the approach followed by you.**
    - Modular design:The approach of using functions for initialization , concatenation, printing, and clearing makes the code modular and easy to understand. Each function serves a specific purpose, enhancing code readability and maintainability.
    - Data Encapsulation: The use of a String structure encapsulates both the string content and its length, providing a convenient and organized way to manage strings.
    - Dynamic length handling:By tracking the length of each string, the program can manage strings of varying lengths without memory wastage.

- Avoiding Buffer Overflows: The approach ensures that strings are not copied or concatenated beyond the allocated buffer size, preventing buffer overflows and potential security vulnerabilities.
- Reusable functions:The functions created for initialization, concatenation, printing, and clearing can be reused in other parts of the program or in different projects, promoting code reusability.
- Structured Memory Management: The program efficiently manages memory by using a fixed-size array for string data, making it predictable and preventing memory fragmentation.

Program code and Output screenshots:

```
#include <stdio.h>


#define MAX_STRING_LENGTH 100


struct String {
    char data[MAX_STRING_LENGTH];
    int length;
};


void initializeString(struct String *str, const char *text) {
    int i = 0;
    while (text[i] != '\0' && i < MAX_STRING_LENGTH - 1) {
        str->data[i] = text[i];
        i++;
    }
    str->data[i] = '\0';

    str->length = i;

}


void concatenateStrings(struct String *dest, const struct String *src) {

    int i = 0;

    while (src->data[i] != '\0' && dest->length < MAX_STRING_LENGTH - 1) {

        dest->data[dest->length + i] = src->data[i];

        i++;

    }

    dest->data[dest->length + i] = '\0';

    dest->length += i;

}


void printString(const struct String *str) {

    int i = 0;

    while (str->data[i] != '\0') {

        printf("%c", str->data[i]);

        i++;

    }

    printf("\n");

}


void clearString(struct String *str) {

    int i;
    for (i = 0; i < MAX_STRING_LENGTH; i++) {
        str->data[i] = '\0';
    }
    str->length = 0;
}

int main() {
    struct String s1, s2;
    initializeString(&s1, "Hello, ");
    initializeString(&s2, "world!");

    concatenateStrings(&s1, &s2);

    printf("Concatenated String: ");
    printString(&s1);

    clearString(&s1);
    clearString(&s2);

    return 0;
}
```



```
/tmp/cJz5A3je6h.o
Concatenated String: Hello, world!
```

Conclusion:-

Implemented a program based on string ADT successfully,after understanding the rules,theory of ADT and the various operations and examples of it,along with some web resources as reference.I used C language for this purpose,without using any standard library function as mentioned,and successfully completed the experiment.