Assignment No. 3

Arrays and Strings

Aim
Write a menu-driven program to perform various string operations such as copy, length, reversing, palindrome, concatenation and, to find occurrence of a sub-string using and without using library functions.

Objective(s)	
1	To study basics of linear data structures
2	To learn the concept of array and types of array
3	To understand use of array
4	To understand string operations without using library functions

Theory

1. What is array? How are arrays declared? How are the elements of an array stored in memory?
2. Define the terms: a) ADT b) Persistent data structures.
3. In what respect linear data structures differ from non-linear data structures?
4. What are advantages of array data structure?
5. What are disadvantages of array data structure?
6. State and explain applications of arrays.
7. What are the properties of abstract data types?
8. How are arrays declared?

Algorithms:

1. Algorithm for finding the string length

Algorithm mystrlen (char str[])

This algorithm reads a Source String character by character and counts characters till end of the string and returns length of the String.

Pre-condition : String should be accepted
Post-condition : Length of the String should be calculated.
Return : Length of the String.

1 Initialize index = 0

2 while (src[index] != '\0') repeat step 3 & 4

3 len = len +1

4 index = index + 1

5 end

6 return len

2. Algorithm for String Copy operation

Algorithm mystrcpy (char des[] , char src[])

This algorithm reads a Source String character by character and copies it to the destination string.

Pre-condition : Source string should be accepted
Post-condition : copy of source should be present in destination string.
Return : reference to Copied String

1 index = 0
2 while (src[index] != '\0') repeat step 3 & 4
3 dest[index] = src[index]
4 index = index + 1
5 end
6 dest[index] ='\0'
7 return reference to the Copied String

3. Algorithm for Sting concatenation Operation

Algorithm mystrcat (char des[] , char src[])

This algorithm reads the source string character by character and appends each character at the end of the destination string and returns the reference to the destination string.

Pre-condition : Both the Strings should be accepted
Post-condition : Source String should be appended at the end of destination string
Return : reference to destination String

1 index1 = 0
2 while(not end of the des)
3 index1 = index1 + 1
4 end
5 index2 = 0
6 while (not end of src)
7 des[index1] = src[index2]
8 index1 = index1 +1
9 index2 = index2 + 1
10 end
11 terminate destination string (des) by '\0' character
12 return reference to destination string (des)

4. Algorithm for String Reverse Operation

Algorithm mystrrev (char src[])

This algorithm reads the source string character by character and adds each character to the source string in reverse order and returns the reference to the destination string.

Pre-condition : The source string should be accepted
Post-condition : Reverse of the Source string should be present in source string.
Return : reference to source string

```
1   start_index = 0
2   end_index  = 0
3   while ( src [ end_index ] != '\0' )
4       end_index = end_index + 1
5   end
6   end_index = end_index – 1
7   while ( start_index < end_index )
8       Swap src[start_index] & src[end_index]
9       start_index = start_index + 1
10      end_index = end_index – 1
11  end
10  return reference to source string
```

5. Algorithm for Palindrome check

Algorithm mypalindrome (char src[])

This algorithm reads the String character by character and compares first character to last character, second to second last and continues till middle character (checks if string is palindrome).

Pre-condition : Source String should be accepted
Post-condition : Result as palindrome or not a palindrome.
Return : Boolean value as a result (1/0)

```
1   start_index = 0
2   end_index = strlen(src)-1
3   while ( start_index < end_index)
4       if src[start_index] == src[end_index]
5           start_index =  start_index + 1
6           end_index = end_index  - 1
7       else
8           goto step 10
9   end
10  if (start_index < end_index )
11      return 0
12  else
13      return 1
```

6. Algorithm for String compare operations

Algorithm mystrcmp(char str1[],char str2[])

This algorithm reads both the Strings character by character and compares characters till end of the string and returns length of the String.

Pre-condition : Both the Strings should be accepted
Post-condition : Equality of the Strings should be checked.
Return : Result of comparison (+ve if str1 > str2, -ve if str1 < str2 , Zero if equal) (ascii difference)

```
1   index= 0;
2   while( str1[index] != '\0' || str2[index ]!= '\0')
3       if (str1[index] == str2[index])
4           index =  index + 1
5       else
6          break;
7       end
8   end
9   diff = str1[index]  - str2[index]
10  return diff
```

7. Algorithm for substring operations

Algorithm SubString (char str1[],char str2[])

This algorithm reads both the Strings character by character and compares characters to check occurrence of str2 in str1 till end of the string and returns boolean result (true/false).

Pre-condition : Both the Strings should be accepted
Post-condition : Check for occurrence of string2 in string1.
Return : No of occurrence of the substring in the main string

```
1   L1 = string length of str1
2   L2 = string length of str2
3   if L2 > L1
4       Return 0  // not a substring i.e. 0 occurrences
5   end
6   Count = 0 // occurrence count
7   for i = 0 to L1 – L2
8       for j = 0 to L2-1
9           if str1[i+j] is not equal to str2[j]
10              break;
11          end if
12      end for
13      if j == L2
14          Increment  count // Increment substring count
15      end if
16  end for
17  return Count
```

Test Conditions:

1. Input two strings with same length.
2. Input two strings without same length.
3. Input such string whose reverse will be equal as original.
4. Input empty strings.

Sample Input Output

***** MENU FOR STRING OPERATIONS *****

1. String Length
2. String Copy
3. String Concatenation
4. String Reverse
5. String Palindrome
6. String Compare
7. Substring

Input :

Let String 1 = " Fundamentals"
String 2 = "ment"
Choose one of the operations to be performed.

Output :

1. StringLength (string1)
   Length of string1 i.e. 12 will be returned
2. StringCopy (string1,string2)
   String 2 will also consist " Fundamentals"
3. StringConcat (string1,string2)
   String2 will get appended to string1
   String2 will have " FundamentalsFundamentals"
4. StringReverse (string1,string2)
   String2 will consist reverse of string1 i.e. "slatnemadnuF"
5. palindrome(string1)
   if string1 consists " Fundamentals"
6. StringCompare (string1,string2)
   String 1 is greater than string 2
7. SubString (string1,string2)
   If string1 = " Fundamentals" and string2 ="ment"
   String 2 is the substring of string1.
   The string is not a pallindrome.

Conclusion