Five Day Faculty Orientation Workshop on Data Structures 2019 Course

Prof. Digambar Padulkar Assistant Professor, VPKBIET, Baramati

Organized By

Hope Foundation's

International Institute of Information Technology, Pune

Under the Aegis of BoS (E&TC), SPPU, Pune SE E&TC/ Electronics) 2019 Course

22nd **to** 26th **June 2020**

Data Structures-Course Details

Course Code: 204184 **Course: Data Structures**

Program	Class	Academic Year
UG Program in EnTC/Electronics	SE (EnTC/Electronics)	2020-21 Sem-I

Corresponding Laboratory: Data Structures Laboratory(204188)

Data Structures-Teaching & Examination Scheme

Teaching Scheme		
Theory hrs/week	**Practical hrs/week**	**Tutorial hrs/week**
3hrs	**2hrs**	-

Examination Scheme					
Theory			**Lab**		
InSem	**EndSem**	**Sessional**	**TW**	**PR**	**OR**
30	70	–	–	–	25

IDE and OS

IDE

- CodeBlock
- Code
- Bluefish
- emac
- VI
- VIM
- geany
- lime

OS

- LINUX Based
  - **Elementary**
  - **Ubuntu**
  - **Fedora**
  - SuSE Linux
  - Kali Linux
  - Bharat OS
  - Mint Linux
  - Bodhi Linux
  - **Debian**
- Windows

Practical assignments

Group A(Compulsory)

1. Perform following String operations with and without pointers to arrays (without using the library functions):
    1. Substring
    2. Palindrome
    3. Compare
    4. Copy
    5. Reverse
2. Implement Database Management using array of structures with operations Create, Display, Modify, Append, Search and Sort. (For any database like Employee or Bank database with and without pointers to structures)
3. Implement Stack and Queue using arrays.
4. Create a singly linked list with options:
    1. Insert (at front, at end, in the middle),
    2. Delete (at front, at end, in the middle),
    3. Display,
    4. Display Reverse,
    5. Revert the SLL
5. Implement Binary search tree with operations Create, search, and recursive traversal.
6. Implement Graph using adjacency Matrix with BFS & DFS traversal.

Continued...

Group B(Any Three)

1. Implement stack and queue using linked list.
2. Implement assignment 2 using files.
3. Add two polynomials using linked list.
4. Reverse a doubly linked list.
5. Evaluate postfix expression (input will be postfix expression)
6. Reverse and Sort stack using recursion.
7. Implement inorder tree traversal without recursion.
8. To find inorder predecessor and successor of a given key in BST.
9. Implement Quicksort.

Continued...

Group C(Any One)

1. Implement merge sort for doubly linked list.
2. Construct a tree from given inorder and preorder traversal.
3. Implement Dijkstra's Algorithm.
4. Implement Circular Linked List with various operations.
5. Represent graph using adjacency list or matrix and generate minimum spanning tree using Prism's algorithm.

Group Assignments

- Make Group of 4 students in a batch (Batch of 20)
- Group will select any one topic as group assignment
- After completing the assignment, the respective group will present it during the practical slot.
- Distribution of work in a group during presentation may contain:
  - Algorithm / Flowchart
  - Program Explanation
  - Applications

Whats with Me

1. Perform following String operations with and without pointers to arrays (without using the library functions):
    1. Substring
    2. Palindrome
    3. Compare
    4. Copy
    5. Reverse
2. Implement Database Management using array of structures with operations Create, Display, Modify, Append, Search and Sort. (For any database like Employee or Bank database with and without pointers to structures)
3. Implement Stack and Queue using arrays.

Continued...

1

Perform following String operations with and without pointers to arrays (without using the library functions):

1. Substring
2. Palindrome
3. Compare
4. Copy
5. Reverse

```
1: procedure SUBSTRING($str, substr$)    ▷ substr is string whose
   appearence is checked
2:     Enter main string $str$ and Substring $substr$
3:     Collect the strings
4:     $i = 0$, $j = 0$
5:     while $End\ \ of\ \ string$ do
6:         if $str[i] == substr[j]$ then
7:             $i++$, $j++$
8:             if $End\ \ of\ \ Substring$ then
9:                 Substring Found
10:            end if
11:        end if
12:        $i++$, $j = 0$
13:    end while
14: end procedure
```

algorithm 1: Finding Substring

```
1: procedure PALINDROME($str$)    ▷ str is a string to be checked as palindrome or not
2:     Enter the string
3:     Collect the strings
4:     $i = 0, j = n - 1$
5:     while $i! = \frac{lenght(str)}{2}$ do
6:         if $str[i] == str[j]$ then
7:             $i++, j--$
8:             String is Palindrome
9:         end if
10:        String is not Palindrome
11:    end while
12: end procedure
```

algorithm 2: Checking string is Palindrome or not

```
1: procedure STRING-COMPARE($str_1, str_2$) ▷ $str_1$ and $str_2$ are the strings
   to be compared
2:     Enter $sre_1$ and $str_2$
3:     Collect the strings
4:     $i = 0, j = 0$
5:     while $End \ \ of \ \ string$ do
6:         if $str_1[i] == str_2[j]$ then
7:             $i++, j++$
8:             String matching
9:         end if
10:        strings are not matching
11:    end while
12: end procedure
```

algorithm 3: Compare Strings

```
procedure REVERSE-STRING($str$)        ▷ $str$ is a string to be reversed
    Enter string $str$
    Collect the strings
    $i = 0, j = lenght(str) - 1$
    while $i != \frac{length(str)}{2}$ do
        $tmp = str[i]$
        $str[i] = str[j]$
        $i++, j--$
    end while
    Print $str$
end procedure
```

algorithm 4: Reverse String

```
procedure SUBSTRING($str, substr$)            ▷ substr is string whose appearance is checked
    Enter main string $str$ and Substring $substr$
    Collect the strings
    *ptr1, *ptr2
    $ptr1 = str$
    $ptr2 = substr$
    while $End\ \ of\ \ string$ do
        if $*ptr1 = *ptr2$ then
            $ptr++, ptr2++$
        end if
        $ptr2 = substr$
        if $ptr2 = eos$ then
            Substring Found
        end if
        $ptr1++, ptr2 = sustring$
    end while
end procedure
```

algorithm 5: Finding Substring

```
1: procedure PALINDROME($str$)    ▷ str is a string to be checked as palindrome or not
2:     Enter the string
3:     Collect the strings
4:     $i = 0$, $j = n - 1$
5:     $ptr1 = str$, $ptr2 = str[n-1]$
6:     while $i != \frac{lenght(str)}{2}$ do
7:         if $*ptr1 == *ptr2$ then
8:             $ptr++$, $ptr2--$
9:             String is Palindrome
10:        end if
11:        String is not Palindrome
12:    end while
13: end procedure
```

algorithm 6: Checking string is Palindrome or not

```
1: procedure STRING-COMPARE($str_1, str_2$) ▷ $str_1$ and $str_2$ are the strings to be compared
2:     Enter $str_1$ and $str_2$
3:     Collect the strings
4:     $ptr_1 = str_1$, $ptr_2 = str_2$
5:     while $End\ \ of\ \ string$ do
6:         if $*ptr_1 == *ptr_2$ then
7:             $ptr_1 + +$, $ptr_2 + +$
8:             String matching
9:         end if
10:        strings are not matching
11:    end while
12: end procedure
```

algorithm 7: Compare Strings

```
procedure REVERSE-STRING($str$)        ▷ $str$ is a string to be reversed
    Enter string $str$
    $*ptr$
    $ptr = str$
    Collect the strings
    $i = 0, j = lenght(str) - 1$
    while $i != \frac{length(str)}{2}$ do
        $tmp = ptr$
        $ptr[i] = ptr[j]$
        $i++, j--$
    end while
    Print $str$
end procedure
```

algorithm 8: Reverse String