

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

Pravara Rural Education Society’s

**Pravara Rural Engineering College, Loni**

Tal: - Rahata, Dist: - Ahmednagar, Loni-413736 (M.S.)



**Course File**

Data Structures Lab (210242)

#### Second Year Engineering (Course 2019)

**Prof. Kote G.B. (MTech CSE)**

***(Assistant Professor)***

**Department of Computer Engineering**



***Data Structure and Lab (DSL)***

***SE Computer Engineering***

##### DATA STRUCTURES LAB

**(210246)**

|  |  |  |
| --- | --- | --- |
| **Teaching Scheme** | **Credit** | **Examination Scheme** |
| PR: 04  Hours/Week | 02 | TW: 25 Marks  PR: 50 Marks |

Guidelines for Instructor's Manual

The instructor’s manual is to be developed as a hands-on resource and reference. The instructor's manual need to include prologue (about University/program/ institute/ department/foreword/ preface etc), University syllabus, conduction & Assessment guidelines, topics under consideration- concept, objectives, outcomes, set of typical applications/practicals/ guidelines, and references.

Guidelines for Student Journal

The laboratory practicals are to be submitted by student in the form of journal. Journal consists ofprologue, Certificate, table of contents, and handwritten write-up of each practical (Title, Objectives, Problem Statement, Outcomes, software & Hardware requirements, Date of Completion, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, conclusion/analysis. Program codes with sample output of all perform practical‟s are to be submitted as softcopy.

As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal may be avoided. Use of DVD containing students programs maintained by lab In-charge is highly encouraged. For referenceone or two journals may be maintained with program prints at Laboratory.

Guidelines for Assessment

Continuous assessment of laboratory work is done based on overall performance and lab practicals performance of student. Each lab practical assessment will assign grade/marks based on parameters with appropriate weightage. Suggested parameters for overall assessment as well as each lab practical assessment include- timely completion, performance, innovation, efficient codes, punctuality and neatness.

Guidelines for Practical Examination

Both internal and external examiners should jointly set problem statements. During practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement. The supplementary and relevant questions may be asked at the time of evaluation to test the student’s for advanced learning, understanding of the fundamentals, effective and efficient implementation. So encouraging efforts, transparent evaluation and fair approach of the evaluator will not create any uncertainty or doubt in the minds of the students. So adhering to these principles will consummate our team efforts to the

pr***D***o***a***m***ta***is***S***i***t***n***ru***g***c***s***tu***ta***r***r***e***t***a***o***n***f***d***t***L***h***a***e***b***s***(***t***D***ud***SL***en***)*** t's academics.



***SE Computer Engineering***

Guidelines for Laboratory Conduction

The instructor is expected to frame the practicals by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The practical framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. The instructor may set multiple sets of practicals and distribute among batches of students. It is appreciated if the practicals are based on real world problems/applications. Encourage students for

Appropriate use of Hungarian notation, proper indentation and comments. Use of open source software is to be encouraged.

In addition to these, instructor may assign one real life application in the form of a mini- project based on the concepts learned. Instructor may also set one practical or mini-project that is suitableto respective branch beyond the scope of syllabus.

Set of suggested practical list is provided in groups- A, B, C, D, and E. Each student must perform at least 13 practicals as at least 3 from group A, 3 from group B, 2 from group C, 2 from group D and 3 from group E.

**Group A and B assignments should be implemented in Python without using built-in methods** for major functionality of assignment. Use List data structure of Python as array**. Group C, D and E assignments should be implemented in C++ language.**

**Operating System recommended**:- 64-bit Open source Linux or its derivative

**Programming tools recommended: -** Open Source Python,

**Programming tool** like Jupiter, Notebook, Pycharm, Spyder, G++/GCC.

|  |  |
| --- | --- |
| **ractical No.** | **Laboratory Assignments** |
| **GROUP - A** | |
| 1 | In second year computer engineering class, group A student’s play cricket, group B students play badminton and group C students play football.  Write a Python program using functions to compute following: -   1. List of students who play either cricket or badminton but not both 2. Number of students who play neither cricket nor badminton 3. Number of students who play cricket and football but not badminton. 4. List of students who play both cricket and badminton |

***Data Structure and Lab (DSL) SE Computer Engineering***



2 Write a Python program to store marks scored in subject “Fundamental ofData Structure” by N students in the class. Write functions to compute following:

1. The average score of class
2. Highest score and lowest score of class
3. Count of students who were absent for the test
4. Display mark with highest frequency

Write a **Python** program for department library which has N books, write functions for following :-

3

Delete the duplicate entries

Display books in ascending order based on cost of books Count number of books with cost more than 500

##### GROUP - B

1. Write a **Python** program to store roll numbers of student in array who attended training program in random order. Write function for searching whether particular student
2. attended training program or not, using linear search and Sentinel search.
3. Write a **Python** program to store roll numbers of student array who attended training Program in sorted order. Write function for searching whether particular student attended training program or not, using Binary search and Fibonacci search for writing PL/SQL block in line with above statement.

a) Write a **Python** program to store names and mobile numbers of your friends in sorted order on names. Search your friend from list using binary search (recursive and non- recursive). Insert friend if not present in phonebook

1. b) Write a **Python** program to store names and mobile numbers of your friends in sorted order

on names. Search your friend from list using Fibonacci search. Insert friend if not Present in phonebook

Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using

6

* 1. SelectionSort
  2. Bubble sort and display top five scores

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | |  |  |  |
|  |  | |  |  |  |
| **GROUP - C** | | | | | |
| 7 | | Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third and final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write C++ program to maintain club member‘s information using singly linked list. Store student PRN and Name. Write functions to:   1. Add and delete the members as well as president or even secretary. 2. Compute total number of members of club 3. Display members 4. Two linked lists exists for two divisions. Concatenate two lists. | | | |
| 8. | | The ticket booking system of Cinemax theater has to be implemented using C++ program. There are 10 rows and 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Headpointer) to each row. On demand  1)The list of available seats is to be displayed 2)The seats are to be booked  3)The booking can be cancelled | | | |

***Data Structure and Lab (DSL) SE Computer Engineering***



***Data Structure and Lab (DSL) SE Computer Engineering***

##### GROUP - D

In any language program mostly syntax error occurs due to unbalancingdelimiter such as (),{},[].

1. Write C++ program using stack to check whether given expression is well parenthesized or not.

A palindrome is a string of character that‘s the same forward and backward. Typically, punctuation, capitalization, and spaces are ignored. For example, “Poor Dan is in a droop” is a palindrome, as can be seen by examining the characters “poor danisina droop” and observing that they are the same forward and backward. One way to check for a palindrome is to reverse the characters in the string and then compare with them the original-in a palindrome, the sequence will be identical. Write C++ program with functions

* 1. To print original string followed by reversed string using stack
  2. To check whether given string is palindrome or not.

1. In any language program mostly syntax error occurs due to unbalancingdelimiter such as (),{},[]. Write C++ program using stack to check whether given expression is well parenthesized or not

##### GROUP - E

Queues are frequently used in computer programming, and a typical example is the creation of a

1. job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enterthe system. Write C++ program for simulating job queue. Write

functions to addjob and delete job from queue.

Write program to implement a priority queue in C++ using an inorder list to store the items in the

1. queue. Create a class that includes the data items (which should be template) and the priority (which should be int). The inorder list should contain these objects, with operator <= overloaded so that the items with highest priority appear at the start of the list (which will make it relatively easy to retrieve the highest item.).

A double-ended queue (deque) is a linear list in which additions and deletions may

1. be made at either end. Obtain a data representation mapping a deque into a one-dimensional



***Data Structure and Lab (DSL)***

***SE Computer Engineering***

#### Practical No: 01(A)

**TITLE**:- In second year computer engineering class, group A student’s play cricket, Group B students play badminton and group C students play football. Write a Python program using functions to compute following: -

* 1. List of students who play both cricket and badminton
  2. List of students who play either cricket or badminton but not both
  3. Number of students who play neither cricket nor badminton
  4. Number of students who play cricket and football but not badminton.

OBJECTIVES:-

1. To know the basics of set.
2. To perform operation on array.
3. To implement set operation using array

**THEORY**:-

Set Theory:

No restriction is placed on the nature of the objects in a set. They can be anything: points, lines, numbers, people, countries, etc. Thus the mathematical meaning of the word set is the same as the regular, nontechnical meaning of the word.

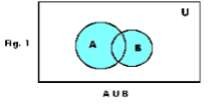
Examples of sets. –

-All points in a given line segment

* Lines through a given point in space
* The set of all rational numbers
* Solutions of the equation 3x2 + 2y2 - 1 = 0
* Citizens of England
* Rivers of Mexico

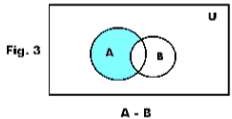
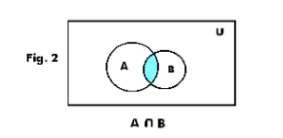
Union of sets:

The union of two sets A and B is the set consisting of all elements in A plus all elements in B and is denoted by A∪B or A + B.



Example: If A = {a, b, c, d} and B = {b, c, e, f, g} then A∪B = {a, b, c, d, e, f, g}.

**Intersection of sets**: The intersection of two sets A and B is the set consisting of all elements that occur in both A and B (i.e. all elements common to both) and is denoted by A∩B, A · B or AB.



***Data Structure and Lab (DSL)***

***SE Computer Engineering***

Example: If A = {a, b, c, d} and B = {b, c, e, f, g} then A∩B = {b, c}.

**Difference of two sets**: The set consisting of all elements of a set A that do not belong to a set B is called the difference of A and B and denoted by A - B.

Example: If A = {a, b, c, d} and B = {b, c, e, f, g} then A - B = {a, d}.

**Universal set U**: Often a discussion involves subsets of some particular set called the universe of discourse (or briefly universe), universal set or space. The elements of a space are often called the points of the space. We denote the universal set by U.

Example: The set of all even integers could be considered a subset of a universal set consisting of all the integers. Or they could be considered a subset of a universal set consisting of all the rational numbers. Or of all the real numbers.

ALGORITHM:-

1. Start
2. Declare variable and string
3. Accept input from user
4. Define all functions and perform all operations
5. Display the output
6. Stop

QUESTIONS:-

1. What a set and how to represent it? 2. What is Universal set?

1. Explain various ways of representing set
2. Explain various operations which can be performed on set?
3. Explain 2D Character Array?



***Data Structure and Lab (DSL)***

***SE Computer Engineering***

#### Practical No: 02(A)

**Practical Title:** Write a python program to store marks for N students.

**Aim:** - Write a Python program to store marks scored in subject “Fundamental of Data Structure” byN students in the class. Write functions to compute following:

* 1. The average score of class
  2. Highest score and lowest score of class
  3. Count of students who were absent for the test
  4. Display mark with highest frequency

Prerequisite:

* Python Programming

Objectives:

To understand the use functions for N student’s record.

**Input:** N number of students.

**Outcome:** Resulting average, highest and lowest marks operation.

Theory:

**ARRAYS**

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of anarray as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

**Declaring** Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

type arrayName [ arraySize ];

This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called

**bal*D*an*at*c*a*e*S***o***tr***f***u***t***c***y***t***p***ur***e***e***d***a***o***n***u***d***b***L***le***ab***, u***(***s***D***e***S***t***L***h***)***is statement −



***SE Computer Engineering***

Double balance [10];

**Initializing Arrays**

You can initialize an array in C either one by one or using a single statement as follows −

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

###### Accessing Array Elements

An element is accessed by indexing the array name. This is done by placing the index of the elementwithin square brackets after the name of the array. For example −

double salary = balance[9];

Functions Used:

**Write algorithm/pseudo code for each function.**

1. The average score of class
2. Highest score and lowest score of class
3. Count of students who were absent for the test
4. Display mark with highest frequency

Algorithm:

WRITE ALGO.

**Flowchart:**

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

Draw flowchart of above algorithm.

**Conclusion:**

By this way, we can store the marks of N student’s successfully.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **P** | **J** | **Total** | **Dated Sign** |
| 3 | 4 | 3 | 10 |
|  |  |  |  |  |

**Questions:**

* 1. Basics of python Programming.
  2. What are functions?
  3. What are the features of python Programming?
  4. What are array?

### Practical No: 3(A)

**Practical Title:** Write a python program for department of library.

**Aim:** - Write a **Python** program for department library which has N books, write functions for following :- a)Delete the duplicate entries

1. Display books in ascending order based on cost of books
2. Count number of books with cost more than 500

**Prerequisite:**

* + Python Programming

**Objectives:**

To understand the use functions for library.

**Input:** N number of books. **Outcome:** Books name,price **Theory:**

Method 1: Using \*set()

This is the fastest and smallest method to achieve a particular task. It first removes the duplicates and returns a dictionary which has to be converted to list.

Method 2: Using list comprehension

Th***D***is***at***m***a S***et***tr***h***u***o***c***d***tu***h***re***as***an***w***d***o***L***r***a***k***b***in***(D***g ***S***si***L***m***)*** ilarly to the above method, but this***S***is***E***j***C***us***o***t***m***a***pu***o***t***n***e***e***r*** -***E***li***n***n***g***e***i***r***ne***s***e***h***r***o***in***r***g***thand of a longer method done with the help of list comprehension.

Method 3: Using set()

This is the most popular way by which the duplicates are removed from the list. But the main and notable drawback of this approach is that the ordering of the element is lost in this particular method.

Method 4: Using list comprehension + enumerate()

list comprehension coupled with enumerate function can also achieve this task. It basically looks for already occurred elements and skips adding them. It preserves the list ordering.

**Method 5: Using collections.OrderedDict.fromkeys()**

This is the fastest method to achieve a particular task. It first removes the duplicates and returns a dictionary which has to be converted to list. This works well in the case of strings also.

**Python list sort() function** can be used to sort a List in ascending, descending, or user-defined order. In each case, the time complexity is ***O(nlogn)*** in Python***.***

Syntax of sort() function

**Syntax:** List\_name.sort(reverse=True/False, key=myFunc)

**Parameter:**

* **reverse:** (Optional), reverse=True will sort the list descending. Default is reverse=False
* **key** Optional. A function to specify the sorting criteria(s)

**Sort the List in Descending Order**

Here, we are sorting the list of numbers in Descending order

**Algorithm:**

WRITE ALGORITHM

**Flowchart:**

Draw flowchart of above algorithm

**Conclusion:**

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

### Practical No: 04(B)

**AIM:-** Implement python program for searching data using various searching algorithm.

**TITLE**:-

1. Write a Python program to store roll numbers of student in array who attended training program in random order. Write function for searching whether particular student attended training program or not, using Linear search and Sentinel search.
2. Write a Python program to store roll numbers of student array who attended training program in sorted order. Write function for searching whether particular student attended training program or not, using Binary search and Fibonacci search

**OBJECTIVES:-**

1. To learn basic concepts of different searching techniques.
2. To learn Linear search, Sentinel search, Binary search and Fibonacci search concept.
3. To get result of operations.

**THEORY**:-

Search techniques Depending on the way data is scanned for searching a particular record, the search techniques are categorized as follows:

1. Sequential (Linear) search
2. Binary search
3. Fibonacci search
4. Index sequential search
5. Hashed search

The performance of a searching algorithm can be computed by counting the number of comparisons to find a given value. We shall study these algorithms with respect to arrays. For sequential search, the same concept applies for searching data in linked lists as well as files.

**1. Linear search**- Linear search is rarely used practically because other algorithms such as the binary search and hash tables allow significantly faster searching, compared to linear search. The time complexity of linear search is O(n). Figure shows a sample linear unordered data and traces the search for the target data of 89.

**ALGORITHM**:

1. Initialize the integer variables.
2. Get the target number from user.
3. Get the list of numbers to be searched.
4. If the counter is equal to target the print the location.
5. If it is not equal to then print that the location is not found end.
6. Exit

Average complexity is the sum of comparisons for each position of the target data divided by n. Hence, average number of comparisons is (1 + 2 + 3 + … + n)/n= (n + 1)/2 The worst-case complexity = n The best-case complexity = 1 if element to be found is at first position.

**Variations of Sequential Search is**

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**2. Sentinel Search-** To reduce overhead of checking the list‟s length, the value to be searched can be appended to the list at the end (or beginning in case of Reverse Search) as a “sentinel value”. A sentinel value is one whose presence guarantees the termination of a loop that processes structured (or sequential) data. Thus on encountering a matching value, its index is returned. The calling function can then determine if the returned index is a valid one or not. Though the optimization resulted in isn‟t much, it reduces the overhead of checking if the index is within limit in each step.

**ALGORITHM:**

1. Set i = 0
2. list[n] = target {add sentinel}
3. Compare key[i] and target Set location = i and goto step 6
4. Move to next data element
5. goto step 3
6. if(location < n) then return location as position of target
7. else report as „Target not found‟ and return −1
8. stop

**3. Binary search-**

Binary search is a very fast and efficient searching technique. It requires the list to be in sorted order. In this method, to search an element there is need to compare it with the present element at the center of the list. If it matches, then the search is successful otherwise the list is divided into two halves: one from the 0thelement till middle element which is the center element (first half) another from the center element to the last element (which is the 2nd half) where all values are greater than the center element.

The searching mechanism proceeds from either of the two halves depending upon whether the target element is greater or smaller than the central element. If the element is smaller than the central element, then searching is done in the first half, otherwise searching is done in the second half.

**ALGORITHM:**

1. Initialize the integer variables.
2. Get the values from the list.
3. Sorted List of numbers is got as input.
4. Get the target number from the user.
5. Initialize first value as zero and last as n-1;
6. The mid value is found.
7. If the target is greater than mid then first is mid + 1 if not last is mid -1.
8. First is last +1.
9. The target is found, if its equal to A(mid) otherwise target is not found.

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**4. Fibonacci search**-

The Fibonacci search also searches from a sorted array. The Fibonacci search technique uses a divide-and-conquer mechanism that helps decrease the possible locations by using Fibonacci numbers.

The Fibonacci search works like the binary search but with a few modifications. In binary search, we have low, high, and mid positions for the sub-list. Here, we have mid = n - Fk−1 + 1, F1 = Fk−2, and F2 = Fk−3. The target to be searched is compared with A[mid], mid is computed as follows:

Case 1 if equal the search terminates;

Case 2 if the target is greater and F1 is 1, then the search terminates with an unsuccessful search; else the search continues at the right of the list with new values of low, high, and mid as mid = mid

+ F2, F1 = Fk−4, and F2 = Fk−5

Case 3 if the target is smaller and F2 is 0, then the search terminates with an unsuccessful search; else the search continues at the left of the list with new values of low, high, and mid as mid = mid

- F2, F1 = Fk−3 and F2 = Fk−4

The search continues by either searching at the left of mid or at the right of mid in the list. Fibonacci search is more efficient than binary search for large- sized lists. However, it is inefficient in case of small lists. The number of comparisons is of the order of n, and the time complexity is O(log(n)).

**CONCLUSION:- QUESTIONS:-**

1. What is need of searching technique?
2. What is difference between linear search and sentinel search?
3. What is difference between binary search and Fibonacci search?
4. What is divide and conquer strategy?
5. Why best case complexity of searching algorithm is 1, explain with an example?

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

### Practical No: 5(B)

**AIM:-** Implement python program for searching data using various searching algorithm.

**TITLE**:-

1. Write a **Python** program to store names and mobile numbers of your friends in sorted order on names. Search your friend from list using binary search (recursive and non- recursive). Insert friend if not present in phonebook
2. Write a **Python** program to store names and mobile numbers of your friends in sorted order on names. Search your friend from list using Fibonacci search. Insert friend if not Present in phonebook

**OBJECTIVES:-**

1. To learn basic concepts of different searching techniques.
2. To learn, Binary search and Fibonacci search concept.

**THEORY**:-

# Binary Search

* **Difficulty Level :** Easy
* **Last Updated :** 13 Oct, 2022

**Problem:** Given a sorted array **arr[]** of **n** elements, write a function to search a given element **x** in **arr[]** and return the index of x in the array.

Consider array is 0 base index.

**Examples:**

***Input:*** *arr[] = {10, 20, 30, 50, 60, 80, 110, 130, 140, 170}, x = 110*

***Output:*** *6*

***Explanation:*** *Element x is present at index 6.*

***Input:*** *arr[] = {10, 20, 30, 40, 60, 110, 120, 130, 170}, x = 175*

***Output:*** *-1*

***Explanation:*** *Element x is not present in arr[].*

**Linear Search Approach**: A simple approach is to do a linear search**.** The time complexity of the Linear search is O(n). Another approach to perform the same task is using *Binary Search*.

**Binary Search Approach:**

***Binary Search*** *is a searching algorithm used in a sorted array by* ***repeatedly dividing the search interval in half****. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).*

***Data Structure and Lab (DSL) SE Computer Engineering***

**Binary Search Algorithm:** The basic steps to perform Binary Search are:

* + Begin with the mid element of the whole array as a search key.
  + If the value of the search key is equal to the item then return an index of the search key.
  + Or if the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
  + Otherwise, narrow it to the upper half.
  + Repeatedly check from the second point until the value is found or the interval is empty.

Binary Search Algorithm can be implemented in the following two ways

1. Iterative Method
2. Recursive Method

1. Iteration Method

binarySearch(arr, x, low, high)

repeat till low = high

mid = (low + high)/2

if (x == arr[mid])

return mid

else if (x > arr[mid]) // x is on the right side

low = mid + 1

else // x is on the left side

high = mid - 1

2. Recursive Method (The recursive method follows the divide and conquer approach)***eering***

binarySearch(arr, x, low, high)

***Data Structure and Lab (DSL) SE Computer Engin***

if low > high

return False

else

mid = (low + high) / 2

if x == arr[mid]

return mid

else if x > arr[mid] // x is on the right side

return binarySearch(arr, x, mid + 1, high)

else // x is on the left side

return binarySearch(arr, x, low, mid - 1)

# Fibonacci Search

* **Difficulty Level :** Medium
* **Last Updated :** 26 Oct, 2022

Given a sorted array arr[] of size n and an element x to be searched in it. Return index of x if it is present in array else return -1.

**Examples:**

***Data Structure and Lab (DSL)***

***Input:*** *arr[] = {2, 3, 4, 10, 40}, x = 10*

***SE Computer Engineering***

***Output:*** *3*

*Element x is present at index 3.*

***Input:*** *arr[] = {2, 3, 4, 10, 40}, x = 11*

***Output:*** *-1*

*Element x is not present.*

Fibonacci Search is a comparison-based technique that uses Fibonacci numbers to search an element in a sorted array.

**Similarities with Binary Search:**

1. Works for sorted arrays
2. A Divide and Conquer Algorithm.
3. Has Log n time complexity.

**Differences with Binary Search**:

1. Fibonacci Search divides given array into unequal parts
2. Binary Search uses a division operator to divide range. Fibonacci Search doesn’t use /, but uses + and -. The division operator may be costly on some CPUs.
3. Fibonacci Search examines relatively closer elements in subsequent steps. So when the input array is big that cannot fit in CPU cache or even in RAM, Fibonacci Search can be useful.

**Background:**

Fibonacci Numbers are recursively defined as F(n) = F(n-1) + F(n-2), F(0) = 0, F(1) = 1. First few Fibonacci Numbers are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

**Observations:**

Below observation is used for range elimination, and hence for the O(log(n)) complexity.

F(n - 2) &approx; (1/3)\*F(n) and

F(n - 1) &approx; (2/3)\*F(n).

**Algorithm:**

Let the searched element be x.

The idea is to first find the smallest Fibonacci number that is greater than or equal to the length of the given array. Let the found Fibonacci number be fib (m’th Fibonacci number). We use (m-2)’th Fibonacci number as the index (If it is a valid index). Let (m-2)’th Fibonacci Number be i, we compare arr[i] with x, if x is same, we return i. Else if x is greater, we recur for subarray after i, else we recur for subarray before i.

Below is the complete algorithm

Let arr[0..n-1] be the input array and the element to be searched be x.

1. Find the smallest Fibonacci Number greater than or equal to n. Let this number be fibM [m’th Fibonacci Number]. Let the two Fibonacci numbers preceding it be fibMm1 [(m-1)’th Fibonacci Number] and fibMm2 [(m-2)’th Fibonacci Number].
2. While the array has elements to be inspected:

1. Compare x with the last element of the range covered by fibMm2

2.

***SE Computer Engineering***

***Data Structure and Lab (DSL)***

1. **If** x matches, return index
2. **Else If** x is less than the element, move the three Fibonacci variables two Fibonacci down, indicating elimination of approximately rear two-third of the remaining array.
3. **Else** x is greater than the element, move the three Fibonacci variables one Fibonacci down. Reset offset to index. Together these indicate the elimination of approximately front one-third of the remaining array.
4. Since there might be a single element remaining for comparison, check if fibMm1 is 1. If Yes, compare x with that remaining element. If match, return index.

**ALGORITHM**: **CONCLUSION:- QUESTIONS:-**

***SE Computer Engineering***

***Data Structure and Lab (DSL)***

**Practical No: 06(B)**

**Practical Title:** Sorting of an array using **selection and bubble sort.**

**Aim:** Write a Python program to store first year percentage of students in array. Write function forsorting array of floating point numbers in ascending orderusing

* 1. Selection Sort
  2. Bubble sort and display top five scores. of club**.**

**Pre-requisite:**

Knowledge of sorting techniques

**Objective:**

Input:

***SE Computer Engineering***

***Data Structure and Lab (DSL)***

Size of array Elements of array

Theory :

- **Write short theory of sorting with its advantages and disadvantages.**

* Explain selection and bubble sort with example

**Algorithm:**

def bubbleSort(alist):

for passnum in range(len(alist)-1,0,- 1):for i in range(passnum):

if alist[i]>alist[i+1]: temp = alist[i] alist[i] = alist[i+1] alist[i+1] = temp

alist = [54,26,93,17,77,31,44,55,20]

bubbleSort(alist

)print(alist)

def selectionSort(alist):

for fillslot in range(len(alist)-1,0,- 1):positionOfMax=0

for location in range(1,fillslot+1): if

alist[location]>alist[positionOfMax

]:positionOfMax = location

temp = alist[fillslot] alist[fillslot] = alist[positionOfMax] alist[positionOfMax] = temp

alist = [54,26,93,17,77,31,44,55,20]

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

selectionSort(alist

)

print(alist)

Flowchart :

Draw flowchart for above algorithms**.**

Conclusion:

By this way, we can perform sorting of an array using selection and bubble sort.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **P** | **J** | **Total** | **Dated Sign** |
| 3 | 4 | 3 | 10 |
|  |  |  |  |  |

Question Bank:

1. Explain the sorting?
2. What are the different types of sorts in data structures3.Define the bubble sort?
3. Define the selection sort?
4. How many passes are required in selection sort?

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**Practical No :7(C)**

**Title:** Singly Linked List.

Objectives:

1. To understand Dynamic memory allocation.
2. To understand implementation of Singly Linked List like Display, Insert and Delete etc.

Problem Statement:

Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of Second, third

and final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write C++ program to maintain club member‘s information using singly linked list. Store student

PRN and Name. Write functions to

 Add and delete the members as well as president or even secretary. ,

 Compute total number of members of club

 Display members

 Display list in reverse order using recursion

 Two linked lists exist for two divisions. Concatenate two lists.

Outcomes:

On completion of this assignment students will be able to-

* Implement Singly Linked list and perform various operations on it.
* Solve real world problem logically using Singly Linked List.

Software & Hardware requirements:

* Open Source C Programming tools like G++/GCC or Eclipse.
* 64-bit Open source Linux or its derivative.

Theory- Concept:

**MEMORY ALLOCATION IN C:**

 **Memory allocation** is the procedure of assigning the computer memory for the execution of programs and processes.

 There are two types of memory allocation in C.

* 1. Static Memory Allocation

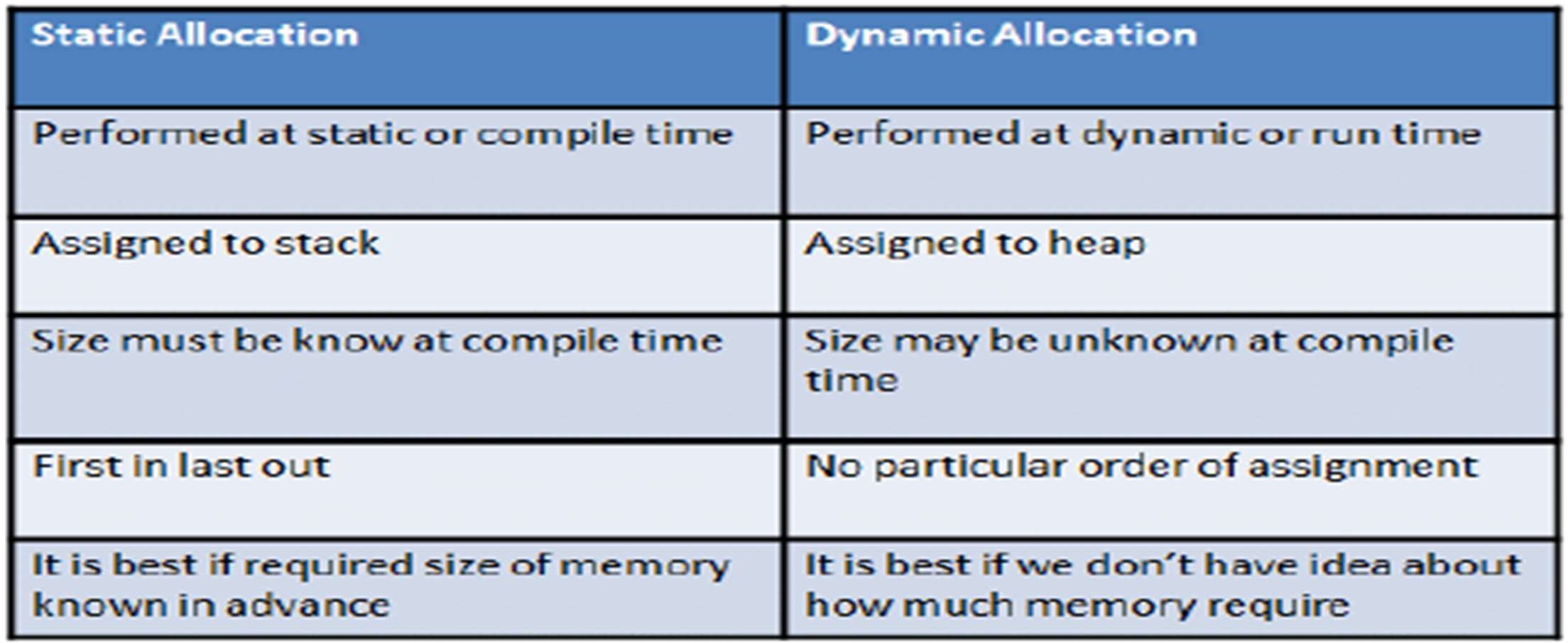
The process of allocating memory before program execution (i.e. During Compilation) is called static memory allocation.

* 1. Dynamic Memory Allocation

The process of allocating memory during program execution is called dynamic memory allocation.

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

DIFFERENCE BETWEEN STATIC MEMORY ALLOCATION AND DYNAMIC MEMORY ALLOCATION IN C:

**DYNAMIC MEMORY ALLOCATION IN C:**

 We use dynamic allocation techniques when it is not known prior how much of memory space is needed for the program and process.

 Dynamic memory allocation arises due to the problems associated with static memory allocation such as if fewer elements are stored, then the rest of the memory is unnecessarily wasted. Therefore, it overcomes the problems of static memory allocation where memory is allocated only when it is required.

 Typical C programs use the functions malloc / calloc and free to allocate and deallocate heap memory.

 The modified versions of the same in C++, is the built-in operators new and delete to allocate and deallocate objects in heap memory.

 When you use new and delete instead of malloc and free, you are able to take advantage of the class library's memory-management enhancements, which can be useful in detecting memory leaks.

 The standard versions of the new and delete operators provide an efficient way to allocate and deallocate memory

DYNAMIC MEMORY ALLOCATION FUNCTIONS IN C:

The concept of dynamic memory allocation in c language enables the C programmer to allocate memory at runtime. Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.

1. malloc()
2. calloc()
3. realloc()
4. free()

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

1. MALLOC() FUNCTION IN C:

 malloc () function is used to allocate space in memory during the execution of the program.

 malloc () does not initialize the memory allocated during execution. It carries garbage value.

 malloc () function returns null pointer if it couldn’t able to allocate requested amount of memory.

 The syntax of malloc() function is given below:

ptr=(cast-type\*)malloc(byte-size);

**Example:**

int main()

{

int \*ptr;

printf("Enter number of elements: "); scanf("%d",&n); ptr=(int\*)malloc(n\*sizeof(int));

}

1. CALLOC() FUNCTION IN C:

 calloc () function is also like malloc () function. But calloc () initializes the allocated memory to zero. But, malloc() doesn’t.

 The syntax of calloc() function is given below:

ptr=(cast-type\*) calloc (number, byte-size);

**Example:**

int main()

{

int \*ptr;

printf("Enter number of elements: "); scanf("%d",&n); ptr=(int\*)calloc(n,sizeof(int));

}

1. REALLOC() FUNCTION IN C:

 realloc () function modifies the allocated memory size by malloc () and calloc () functions to new size.  If enough space doesn’t exist in memory of current block to extend, new block is allocated for the full size of reallocation, then copies the existing data to new block and then frees the old block.

 The syntax of realloc() function is given below:

ptr=realloc(ptr, new-size);

**Example**:

int main()

{

int \*ptr;

printf("Enter number of elements: "); scanf("%d",&n); ptr=(int\*)calloc(n,sizeof(int)); ptr=realloc(ptr,n+1);

}

1. FREE() FUNCTION IN C:

 free () function frees the allocated memory by malloc (), calloc (), realloc () functions and returns the memory to the system.

***SE Computer Engineering***

***Data Structure and Lab (DSL)***

 Otherwise, it will consume memory until program exit.

 The syntax of free() function is given below:

ptr=free(ptr);

**Example**:

int main()

{

int \*ptr;

printf("Enter number of elements: "); scanf("%d",&n); ptr=(int\*)calloc(n,sizeof(int)); free(ptr);

}

WHAT IS LINKED LIST?

 When we want to work with unknown number of data values, we use a linked list data structure to organize that data.

 Linked list is a linear data structure that contains sequence of elements such that each element links to its next element in the sequence.

 Each element in a linked list is called as "Node".

 Each node holds its own data and the address of the next node hence forming a chain like structure.

 Linked Lists are used to create trees and graphs.

. TYPES OF LINKED LISTS:

 There are 3 different implementations of Linked List available, they are: o Singly Linked List

* Doubly Linked List
* Circular Linked List

SINGLY LINKED LIST:

 Singly linked lists contain nodes which have a data part as well as an address part i.e. next, which points to the next node in the sequence of nodes. Single linked list is a sequence of elements in which every element has link to its next element in the sequence.

 The graphical representation of a node in a single linked list is as follows...

 In a single linked list, the address of the first node is always stored in a reference node known as "front" (Sometimes it is also known as "head").

 Always next part (reference part) of the last node must be NULL.

Algorithm:

 Before we implement actual operations, first we need to setup empty list. First perform the following steps before implementing actual operations.

– **Step 1:** Include all the header files which are used in the program.

* **Step 2**: Declare all the user defined functions.

***SE Computer Engineering***

***Data Structure and Lab (DSL)***

* **Step 3:** Define a Node structure with two members data and next
* **Step 4:** Define a Node pointer 'head' and set it to NULL.
* **Step 5:** Implement the main method by displaying operations menu and make suitable function calls in the main method to perform user selected operation.

INSERTION ALGORITHM

 In a single linked list, the insertion operation can be performed in three ways.

 They are as follows...

* + Inserting At Beginning of the list
  + Inserting At End of the list
  + Inserting At Specific location in the list

INSERTING AT BEGINNING OF THE LIST

 We can use the following steps to insert a new node at beginning of the single linked list...

* Step 1: Create a newNode with given value.
  + Step 2: Check whether list is Empty (head == NULL)
  + Step 3: If it is Empty then, set newNode→next = NULL and head = newNode.
  + Step 4: If it is Not Empty then, set newNode→next = head and head = newNode.

INSERTING AT END OF THE LIST

 We can use the following steps to insert a new node at end of the single linked list...

* + - Step 1: Create a newNode with given value and newNode → next as NULL.
    - Step 2: Check whether list is Empty (head == NULL).
    - Step 3: If it is Empty then, set head = newNode.
    - Step 4: If it is Not Empty then, define a node pointer temp and initialize with head.
    - Step 5: Keep moving the temp to its next node until it reaches to the last node in the list (until temp → next is equal to NULL).
    - Step 6: Set temp → next = newNode.

INSERTING AT SPECIFIC LOCATION IN THE LIST (AFTER A NODE)

 We can use the following steps to insert a new node after a node in the single linked list...

* + - * Step 1: Create a newNode with given value.
      * Step 2: Check whether list is Empty (head == NULL)
      * Step 3: If it is Empty then, set newNode → next = NULL and head = newNode.
      * Step 4: If it is Not Empty then, define a node pointer temp and initialize with head.
      * Step 5: Keep moving the temp to its next node until it reaches to the node after which we want to insert the newNode (until temp1 → data is equal to location, here location is the node value after which we want to insert the newNode).
      * Step 6: Every time check whether temp is reached to last node or not. If it is reached to last node then display 'Given node is not found in the list!!! Insertion not possible!!!' and

***Data Structure and Lab (DSL) SE Computer Engineering***

terminate the function. Otherwise move the temp to next node.

* + - * Step 7: Finally, Set 'newNode → next = temp → next' and 'temp → next = newNode'

DISPLAYING A SINGLE LINKED LIST

 We can use the following steps to display the elements of a single linked list...

* + - * + Step 1: Check whether list is Empty (head == NULL)
        + Step 2: If it is Empty then, display 'List is Empty!!!' and terminate the function.
        + Step 3: If it is Not Empty then, define a Node pointer 'temp' and initialize with head.
        + Step 4: Keep displaying temp → data with an arrow (--->) until temp reaches to the last node
        + Step 5: Finally display temp → data with arrow pointing to NULL (temp → data ---> NULL).

Conclusion:

Hence we have learned Singly Linked Listand how to perform various operations on it. Program Codes with sample output:

---

Assignment Questions:

1. Write down Advantages, Disadvantages and Applications of Linked List.
2. Write down difference between Malloc and Calloc function.

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

## Practical No: 08(C)

**Practical Title:** The ticket booking system of Cinemax Theater has to be implemented using C++ program. There are 10 rows and 7 seats in each row. Doubly circular linked list has to bemaintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to each row. On demand

* 1. The list of available seats is to be displayed
  2. The seats are to be booked
  3. The booking can be cancelled.

Pre-requisite:

* + - Knowledge of Doubly Circular Linked List
    - Representation of Circular Linked list
    - Knowledge of ticket booking

Objective:

* + - To perform Doubly Circular linked list for cinemax ticket booking.
    - To display available seats.
    - To book and cancel seats

Input:

Row no and seat no to book seat

Outcome:

* + - Display available seats to book movie ticket.
    - Display status of Booked seat/ cancel seat.

Theory:

**Circular Doubly linked list: (write CDLL theory in details i.e. definition, concepts, advantages,disadvantages.)**

* Doubly Circular linked list as an ADT: (write pseudo code for each operation)
  + **Algorithms :**

(Write your own algorithms for your program)

* + **Flowchart :**

(Draw flowchart for above algorithms)

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

Conclusion:

By this way, we can book or cancel movie ticket using doubly Circular linked lists.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **P** | **J** | **Total** | **Dated Sign** |
| 3 | 4 | 3 | 10 |
|  |  |  |  |  |

Questions:

1. What is Doubly Circular Linked List?
2. How to represent doubly circular linked list?
3. How to book or cancel seat?
4. What is doubly linked list?
5. How to insert and delete elements from doubly circulars linked list?

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**Practical No:9(D)**

**Aim:** A palindrome is a string of character that‘s the same forward and backward. Typically, punctuation, capitalization, and spaces are ignored. For example, “Poor Dan is in a droop” is a palindrome, as can be seen by examining the characters “poor danisina droop” and observing that they are the same forward and backward. One way to check for a palindrome is to reverse the characters in the string and then compare with them the original-in a palindrome, the sequence will be identical. Write C++ program with functions

* 1. To print original string followed by reversed string using stack
  2. To check whether given string is palindrome or not.

**Pre-requisite:**

* + - Basics of string
    - Palindrome related cocepts

**Objective**:

To check whether given string is palindrome or not.

**Input:**

Take a string

**Outcome:**

string is palindrome or not.

**Theory:**

**Palindrome:-**

A string is said to be palindrome if the reverse of the string is the same as string. For example, “radar” is a palindrome, but “radix” is not a palindrome.

**Examples:**

**Input :** malayalam

**Output :** Yes

**using the inbuilt function to reverse a string:**

In this method, predefined function **‘ ‘.join(reversed(string))** is used to reverse string.

**Method using flag:**

In this method, the user compares each character from starting and ending in a for loop and if the character does not match then it will change the status of the flag. Then it will check the status of the flag and accordingly and print whether it is a palindrome or not.

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**Method using recursion**:

This method compares the first and the last element of the string and gives the rest of the substring to a recursive call to itself.

**Method :**

Using extend() and reverse() methods

**Method using one extra variable:**

In this method, the user takes a character of string one by one and store it in an empty variable. After storing all the characters user will compare both the string and check whether it is palindrome or not.

**Algorithms:**

Write your own algorithms

**Flowchart:**

Draw flowchart for above algorithms.

**Conclusion:**

### Practical No:10(D)

***Data Structure and Lab (DSL) SE Computer Engineering***

**Practical Title:** Write C++ program to check well formedness of parenthesis using stack.

**Aim:** In any language program mostly syntax error occurs due to unbalancing delimiter such as (),

{},[]. Write C++ program using stack to check whether given expression is well parenthesized ornot.

**Pre-requisite:**

* + - Basics of stack.
    - Different operations that can be performed on stack

**Objective**:

* + - To check whether the given expression is well parenthesized or not.

**Input:**

Expression using {}, (),[].

**Outcome:**

* + - Result of checking well forkedness of parenthesis.

**Theory:**

- **Write short theory of stack.**

- **Write concept of well form parenthesis.**

- **Example of well form parenthesis.**

**Algorithms:**

**Write your own algorithms**

**Flowchart:**

**Draw flowchart for above algorithms.**

**Conclusion:**

By this way, we can check well formedness of parenthesis using stack.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **P** | **J** | **Total** | **Dated Sign** |
| 3 | 4 | 3 | 10 |
|  |  |  |  |  |

**Question Bank:**

1. What is Stack?
2. Which are the different operations that can be performed on stack?
3. Explain PUSH, POP operations on stack
4. What are the applications of stack?

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

### Practical No:11 (E)

**Practical Title:** Perform different operations on Queue.

**Aim:** Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue.Writefunctions to add job and delete job from queue.

**Pre-requisite:**

* + Basics of Queue
  + Different operations that can be performed on queue

**Objective**:

* + To perform addition and deletion operations on queue.

**Input:**

Size of queue Elements in queue

**Outcome:**

* + Result of addition of job operation on queue.
  + Result of deletion of job operation on queue.

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**Theory:**

* **Write theory of queue (definition, concepts, types, advantages, disadvantages)**
* **Explain queue as an ADT. (write pseudo code)Algorithms:**

**Step 1**: Include all the header files which are used in the program and define a constant 'SIZE' withspecific value.

**Step 2**: Declare all the user defined functions which are used in queue implementation.

**Step 3**: Create a one dimensional array with above defined SIZE (int queue[SIZE])

**Step 4**: Define two integer variables 'front' and 'rear' and initialize both with '-1'. (int front = -1, rear

= -1)

**Step 5**: Then implement main method by displaying menu of operations list and make suitablefunction calls to perform operation selected by the user on queue.

**enQueue (value) - Inserting value into the queue:**

In a queue data structure, enQueue() is a function used to insert a new element into the queue. In a queue, the new element is always inserted at rear position. The enQueue () function takes one integer value as parameter and inserts that value into the queue. We can use the following steps to insert an element into the queue...

**Step 1**: Check whether queue is FULL. (rear == SIZE-1)

**Step 2**: If it is FULL, then display "Queue is FULL!!! Insertion is not possible!!!" and terminate thefunction.

**Step 3**: If it is NOT FULL, then increment rear value by one (rear++) and set queue[rear] = value.

**deQueue() - Deleting a value from the Queue:**

In a queue data structure, deQueue() is a function used to delete an element from the queue. In a queue, the element is always deleted from front position. The deQueue() function does not take any value as parameter. We can use the following steps to delete an element from the queue...

**Step 1**: Check whether queue is EMPTY. (front == rear)

**Step 2**: If it is EMPTY, then display "Queue is EMPTY!!! Deletion is not possible!!!" and terminatethe function.

**Step 3**: If it is NOT EMPTY, then increment the front value by one (front ++). Then display queue[front] as deleted element. Then check whether both front and rear are equal (front == rear), ifit TRUE, then set both front and rear to '-1' (front = rear = -1).

**display() - Displays the elements of a Queue:**

We can use the following steps to display the elements of a queue...

**Step 1**: Check whether queue is EMPTY. (front == rear)

**Step 2**: If it is EMPTY, then display "Queue is EMPTY!!!" and terminate the function.

**Step 3**: If it is NOT EMPTY, then define an integer variable 'i' and set 'i = front+1'.

**Step 3**: Display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until 'i' valueis equal to rear (i <= rear)

**Flowchart :**

**Draw flowchart for above algorithms**

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**Conclusion:**

By this way, we can perform different operations on queue

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **P** | **J** | **Total** | **Dated Sign** |
| 3 | 4 | 3 | 10 |
|  |  |  |  |  |

**Question Bank:**

1. What is Queue?
2. What are the different operations that can be performed on queue?
3. Explain all the operations on queue

##### Practical No: 12 (E)

**Title :** To implement priority queue.

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**Aim:**Write program to implement a priority queue in C++ using an inorder list to store the items in the queue. Create a class that includes the data items (which should be template) and the priority (which should be int). The inorder list should contain these objects, with operator <= overloaded so that the items with highest priority appear at the start of the list (which will make it relatively easy to retrieve the highest item.).

**Pre-requisite: •**

* + Basics of priority Queue
  + Different operations that can be performed on priority queue

**Objective:**

Make a relatively easy to retrieve the highest item

**Outcome:**

* arrange your data according to your priority
* make your tasks easy.

Theory & Algorithm:

**What is Priority Queue?**

A priority queue in C++ is a derived class in STL(standard template library) that processes only

the highest priority element. It means that the first element of the queue is greatest from the rest of the elements of the queue. Later all the elements are arranged in non-increasing order, i.e., you can see each element of the queue has fixed priority.

The queue data structure follows the FIFO strategy, i.e., first in, first out, whereas the priority queue only checks the priority of the elements for retrieving the elements. Priority queues are implemented as container adapters. It is a class that uses an encapsulated object of a specific container class and deliver a specific set of member function to access the queue elements.

**Algorithm for insertion operation in the priority queue**

1. **if** no node is present, create a newNode.
2. **else** (the node is present) insert the newNode at the end
3. perform heapify

**C++ program to insert the new element in the priority queue**

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

/\* insertion operation in priority queue\*/

#include#include //Header-file **using namespace** std;

**int main**()

{

priority\_queue<**int**> p;

p.push(**5**); // inserting elements

p.push(**4**);

p.push(**3**);**while** (!p.empty())

{

cout << ' ' << p.top(); //printing elements p.pop();

}

}

**2) Accessing elements in Priority Queue**

Priority Queue always retrieves the highest priority element i.e., the greatest element from the max heap without deleting the node.

**Algorithm for accessing an element in the priority queue**

1) **return** rootNode

**C++ program to access elements from the priority queue**

/\* Program to access an element in priority queue\*/ #include#include //Header-file **using namespace** std;

**int main**()

{

priority\_queue<**int**> p; p.push(**5**);

p.push(**4**);

p.push(**3**);

p.push(**2**);

cout<<p.top(); //fetch element priority(maximum element)

}

**3) Deletion in Priority Queue**

For deleting a certain element from the priority queue(max-heap), you can follow below steps:

S ***Data Structure and Lab (DSL)*** leted

electing the element to be de

***SE Computer Engineering***

Now, you will swap it with the last element of the priority tree

Later, remove the last element as shown in the below image:

**Algorithm for deletion operation in the priority queue**

1. **if** nodeToBeDeleted is the leafNode then

**delete** the node

1. **else** swap lastLeafNode with the nodeToBeDeleted

**delete** nodeToBeDeleted

1. perform heapify

**C++ program to delete the elements in the priority queue**

/\* Program to delete elements in priority queue\*/ #include#include //Header-file **using namespace** std; **int main**()

{

priority\_queue<**int**> p; p.push(**5**);

p.push(**4**);

p.push(**3**);

p.push(**2**);

p.pop();

p.pop();

**while** (!p.empty())

{

cout << ' ' << p.top(); p.pop();

}

}

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

**Algorithms :**

**Write your own algorithms**

**Flowchart :**

Draw flowchart for above algorithms

# Conclusion

Identifying the priorities and doing tasks accordingly allows you with faster and effective results. Priority queue is one such data structure that helps you arrange your data according to your priority and make your tasks easy. Therefore, it is highly recommended to learn priority queue in C++ and take high advantage of all its applications.

##### Practical No: 13 (E)

**Practical Title:** Perform operations on Double ended queue.

**Aim:** A double-ended queue(deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one-dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque.

**Pre-requisite:**

***Data Structure and Lab (DSL)***

***SE Computer Engineering***

* Knowledge of Queue
* Types of queue
* Knowledge of double ended queue and different operations that can be performed on it

**Objective**:

* To simulate deque with functions to add and delete elements from either end of the deque.

**Input:**

Size of array Elements in the queue

**Outcome:**

* Result of deque with functions to add and delete elements from either end of the deque.

**Theory:**

**Double-Ended Queue**

A double-ended queue is an abstract data type similar to an simple queue, it allows you to insert anddelete from both sides means items can be added or deleted from the front or rear end.

**Algorithm for Insertion at rear end**

Step -1: [Check for overflow] if(rear==MAX) Print("Queue is Overflow”);return;

Step-2: [Insert element]else rear=rear+1;

q[rear]=no;

[Set rear and front pointer]

if rear=0

rear=1

;if front=0

front=1; Step-3:

return

**Implementation of Insertion at rear end**

void add\_item\_rear()

{

int num;

printf("\n Enter Item to insert : ");scanf("%d",&num); if(rear==MAX)

{

printf("\n Queue is Overflow");return;

}

else

{

rear++; q[rear]=nu m; if(rear==0) rear=1; if(front==0) front=1;

}

}

**Algorithm for Insertion at font end**

Step-1 : [Check for the front position]if(front<=1)

Print (“Cannot add item at front end”);return;

Step-2 : [Insert at front]else front=front-1; q[front]=no;

Step-3 :

Return

**Implementation of Insertion at font end**

void add\_item\_front()

{

int num;

printf("\n Enter item to insert:");scanf("%d",&num);

if(front<=1)

{

printf("\n Cannot add item at front end");return;

}

else

{

front--; q[front]=nu m;

}

}

**Algorithm for Deletion from front end**

Step-1 [ Check for front pointer]if front=0 print(" Queue is Underflow”);return; Step-2 [Perform deletion]else

no=q[front]; print(“Deleted element is”,no);[Set front and rear pointer]

if front=rear front=0; rear=0; else

front=front+1

; Step-3 :

Return

**Implementation of Deletion from front end**

void delete\_item\_front()

{

int num; if(front==0

)

{

printf("\n Queue is Underflow\n");return;

}

else

{

num=q[front];

printf("\n Deleted item is

%d\n",num);if(front==rear)

{

front=0

;

rear=0;

}

else

{

front++

}

}

}

**Algorithm for Deletion from rear end**Step-1 : [Check for the rear pointer] if rear=0

print(“Cannot delete value at rear end”);return;

Step-2: [ perform deletion]else no=q[rear];

[Check for the front and rear pointer]if front= rear front=0

;

rear=0; else

rear=rear-1; print(“Deleted element is”,no);Step-3 : Return

**Implementation of Deletion from rear end**

void delete\_item\_rear()

{

int num; if(rear==0

)

{

printf("\n Cannot delete item at rear end\n");return;

}

else

{

num=q[rear]; if(front==rea r)

{

front=0

;

rear=0;

}

else

{

rear--;

printf("\n Deleted item is %d\n",num);

}

}

}

**Flowchart :**

Draw flowchart for above algorithms

**Conclusion:**

By this way, we can perform operations on double ended queue

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **P** | **J** | **Total** | **Dated Sign** |
| 3 | 4 | 3 | 10 |
|  |  |  |  |  |

**Question Bank:**

1. What is queue?
2. Types of queue
3. What is double ended queue?
4. How to insert the new node in Doubly Link List?
5. How to delete the node from front of Doubly Link List ?
6. How to delete the node from end of Doubly Link List ?
7. How to delete the node in between of Doubly Link List

