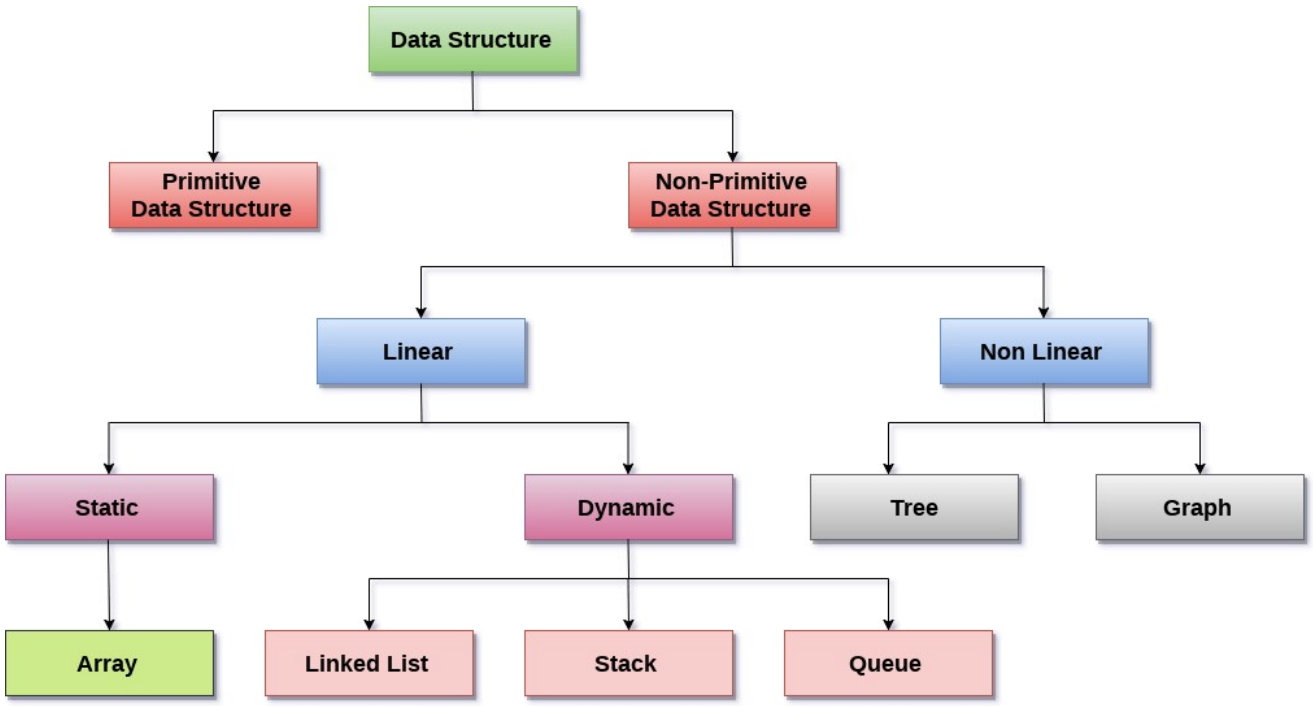
INTRODUCTION OF DATA STRUCTURE:

- Data Structure can be defined as the group of data elements which provides an efficient way of storing and organizing data in the computer so that it can be used efficiently.
- Some examples of Data Structures are arrays, Linked List, Stack, Queue, etc.
- Data Structures are widely used in almost every aspect of Computer Science i.e. operating System, Compiler Design, Artificial intelligence, Graphics and many more.
- It plays a vital role in enhancing the performance of software or a program as the main function of the software is to store and retrieve the user's data as fast as possible.

Data Structure Classification



Linear Data Structures:

- A data structure is called linear if all of its elements are arranged in the linear order.
- The implementation of linear data structures is easier as the elements are sequentially organized in memory.

- The data elements are traversed one after another and can access only one element at a time.
- The types of linear data structures are Array, Queue, Stack, Linked List.

Non Linear Data Structures:

- This data structure does not form a sequence i.e. each item or element is connected with other items in a non-linear arrangement.
- The data elements are not arranged in a contiguous manner.
- In the case of linear data structure, element is connected to two elements (previous and the next element), whereas, in the non-linear data structure, an element can be connected to more than two elements.

Linear-Vs-Non-Linear-Data-Structure

	Linear Data structure	**Non-Linear Data structure**
Basic	In this structure, the elements are arranged sequentially or linearly and attached to one another.	In this structure, the elements are arranged hierarchically or non-linear manner.
Types	Arrays, linked list, stack, queue are the types of a linear data structure.	Trees and graphs are the types of a non-linear data structure.
Implementation	Due to the linear organization, they are easy to implement.	Due to the non-linear organization, they are difficult to implement.
Traversal	As linear data structure is a single level, so it requires a single run to traverse each data item.	The data items in a non-linear data structure cannot be accessed in a single run. It requires multiple runs to be traversed.
Arrangement	Each data item is attached to the previous and next items.	Each item is attached to many other items.
Levels	This data structure does not contain any hierarchy, and all the data elements are organized in a single level.	In this, the data elements are arranged in multiple levels.
Memory	In this, the memory utilization is	In this, memory is utilized in a very efficient

utilization	not efficient.	manner.
Time complexity	The time complexity of linear data structure increases with the increase in the input size.	The time complexity of non-linear data structure often remains same with the increase in the input size.
Applications	Linear data structures are mainly used for developing the software.	Non-linear data structures are used in image processing and Artificial Intelligence.

Ques: Explain Application areas of DS.

Ans: Some examples of how data structures are used include the following:

- **Storing data:**

  Data structures are used for efficient data storage, such as specifying the collection of attribute and corresponding structures used to store records in a database management system.

- **Managing resources and services.**

  Core operating system (OS) resources and services are enabled through the use of data structures such as linked lists for memory allocation, file directory management and file structure trees, as well as process scheduling queues.

- **Data exchange.**

  Data structures define the organization of information so that it can be shared between applications.

- **Ordering and sorting.**

  Data structures such as sorted binary tree -- provide efficient methods of sorting objects. With data structures such as priority queues, programmers can manage items organized according to a specific priority.

- **Indexing.**

Even more complicated data structures such as B-trees are used to index objects, such as those stored in a database.

- **Searching.**

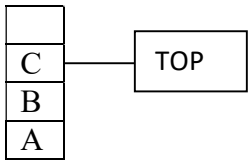
Indexes created using binary search trees, B-trees or hash tables speed the ability to find a specific data.

- **Scalability.**

Data structures used for allocating and managing data storage across distributed storage locations, ensuring scalability and performance.

Ques. What is stack? How it is implemented?

Definitions :- A stack is an ordered list of element, in which the element are added and removed from only one end called as top of the stack.i.e.,last in fist out (LIFO)the last element added to the stack will be first one to be removed.



Stack can be implemented using two techniques:

1) Pointer (Dynamic Stack)

Here memory is allocated dynamically and error occurs if no more space available in memory area.

2) Array (Static Stack)

Here fixed number is allocated at compilation time.

Ques: Write down the some applications of Stack. Also explain in short array based stack implementation.

Ans: Real life applications:

1. Pile of books

2. Plate trays

More applications related to computer science

1. Program execution stack
2. Recursive Function
3. Calling Function
4. Expression Evaluation
5. Expression Conversion
    a. Infix to Postfix
    b. Infix to Prefix
    c. Postfix to Infix
    d. Prefix to Infix

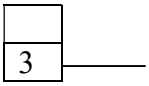
Array based stack implementation:

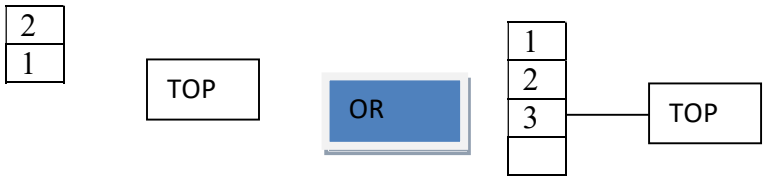
- Allocate an array of some size (pre-defined)
    - Maximum N elements in stack
- Bottom stack element stored at element 0
- last index in the array is the top
- Increment top when one element is pushed, decrement after pop

Ques: What is stack? Give its different operation?

A stack is a non-primitive linear data structure. It is defined as a list in which insertion of new data item and deletion of already existing data item is done from only one end called the top of stack (TOP).

Since, all the insertions and deletion in stack are made from top of the stack; the last insertion item will be the first to be removed from stack. That is why; stack is also called Last in First out (LIFO) type of list.

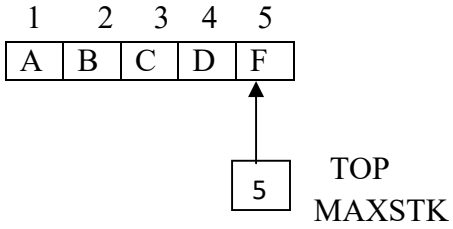




Ques: When stack overflow occur?

Stack will maintained by a linear array. It has a fixe size that is a maximum size of stack, when maximum number of element is present in stack. Then it indicted overflow of stack.

TOP:-it contains the location of the top element of the stack.
MAXSTK:-it is maximum size of elements of stack.



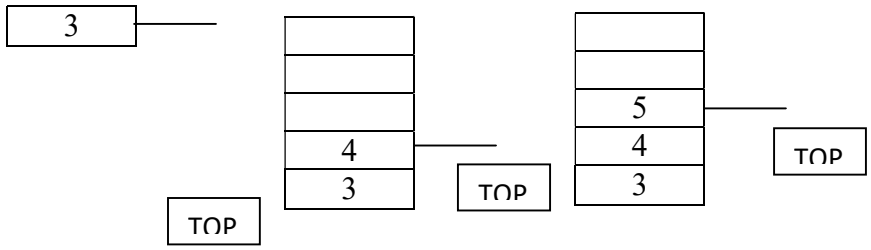
The condition TOP=MAXSTK, it indicate that stack is full. Then overflow occur.

Basic Operations Performed on Stack:

1. Push
2. Pop
3. Display(Peep)

1. **Push**: -

The process of inserting element to the top of stack is called push operation. Each time a new element is inserted in the stack, top is incremented by one before the element is placed on the stack.



Assumption:

Top: is a variable that indicates position of topmost element in stack.

Max: indicates the maximum limit of stack.

Item: indicates the value of item to be inserted.

Stack: is a 1 D array of size max.

Push Algorithm:

1.if top==MAX-1 then print stack is full (overflow)

2.else accept data items from the user

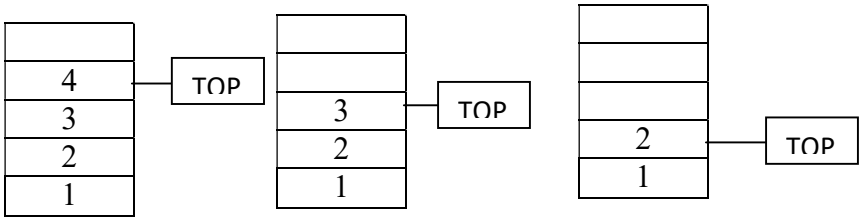
3.set top=top+1

4.stack[top] = item

5.stop

2**. Pop**: -

The process of deleting an element from the top of stack is called pop operation. After every pop operation, the top is decremented by one.



Assumption:

Same as push.

Pop Algorithm:

1.if top== -1 then print stack is empty (underflow)

2.else set item= stack[top]

3.set top=top-1

4.stop

3. Peep (display):-

Peep is used to display element of a specific position in a stack. In this operation we simply move logically to desired location and then fetch the information associated with that location.

Assumption:

I is a variable that indicates element number in the stack from top of the stack.other assumptions are same as push.

Algorithm:

1. if(i<=0 || i>max) then

   Print " invalid location".

2. else return(stack[top-i+1])

3. stop.

C++ Program to Implement Stack using array

Example

```
#include <iostream>

int stack[100], n=100, top=-1;

void push(int val) {
```

```
  if(top>=n-1)
  cout<<"Stack Overflow"<<endl;
  else {
    top++;
    stack[top]=val;
  }
}
void pop() {
  if(top<=-1)
  cout<<"Stack Underflow"<<endl;
  else {
    cout<<"The popped element is "<< stack[top] <<endl;
    top--;
  }
}
void display() {
  if(top>=0) {
    cout<<"Stack elements are:";
    for(int i=top; i>=0; i--)
    cout<<stack[i]<<" ";
    cout<<endl;
  } else
  cout<<"Stack is empty";
}
int main() {
```

```
int ch, val;
cout<<"1) Push in stack"<<endl;
cout<<"2) Pop from stack"<<endl;
cout<<"3) Display stack"<<endl;
cout<<"4) Exit"<<endl;
do {
  cout<<"Enter choice: "<<endl;
  cin>>ch;
  switch(ch) {
    case 1: {
      cout<<"Enter value to be pushed:"<<endl;
      cin>>val;
      push(val);
      break;
    }
    case 2: {
      pop();
      break;
    }
    case 3: {
      display();
      break;
    }
    case 4: {
      cout<<"Exit"<<endl;
```

```
            break;
        }
        default: {
            cout<<"Invalid Choice"<<endl;
        }
      }
   }while(ch!=4);
   return 0;
}
```

Output

```
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 1
Enter value to be pushed: 2
Enter choice: 1
Enter value to be pushed: 6
Enter choice: 1
Enter value to be pushed: 8
Enter choice: 1
Enter value to be pushed: 7
Enter choice: 2
The popped element is 7
Enter choice: 3
Stack elements are:8 6 2
Enter choice: 5
Invalid Choice
Enter choice: 4
Exit
```

In the above program, the push() function takes argument val i.e. value to be pushed into the stack. If a top is greater than or equal to n, there is no space in a stack and overflow is printed. Otherwise, val is pushed into the stack.

The pop() function pops the topmost value of the stack, if there is any value. If the stack is empty then underflow is printed.

The display() function displays all the elements in the stack. It uses a for loop to do so. If there are no elements in the stack, then Stack is empty is printed.

The function main() provides a choice to the user if they want to push, pop or display the stack. According to the user response, the appropriate function is called using switch. If the user enters an invalid response, then that is printed.

Ques: Explain the difference between stack and queue with their functionality.

Stack	Queue
A linear list which allows insertion or deletion of an element at one end only is called stack.	A linear list which allows insertion and one end and deletion at another end is called as Queue.
Since insertion and deletion of an element are performed at one end of the stack, the elements can only be removed in the opposite order of insertion.	Since insertion and deletion of an element are performed at opposite end of the queue, the element can only be removed in the same order of insertion.
Stack is called as Last In First Out (LIFO) list.	Queue is called as First In First Out (FIFO) List.
The most and least accessible elements are called as TOP.	Insertion of element is performed at FRONT end and deletion is performed from REAR end.
Example of Stack is arranging coins in one above one.	Example is ordinary queue in Electricity bill payment.
Insertion operation is referred as PUSH and deletion operation is referred as POP	Insertion operation is referred as INSERT and deletion operation is referred as DELETE.
Function calling in any languages uses Stack.	Task Scheduling by Operating System uses queue.

Ques: Explain recursion. Give its advantages and disadvantages write recursive function to display first N Fibonacci number.

Recursive function: - function that calls itself, a function that is part of a cycle in the sequence of function calls.

Recursion function is a process of calling the function repeatedly in terms of itself. The repeated calling is terminated by the specified condition.

- ❖ The recursion function must satisfy following condition.
  - → The function must have a stopping condition.i.e; it should not continue to call indefinitely.

→ Each time function calls itself, it must be in recursive form,i.e., must be nearer to a solution.

Types of recursions

I. Direct
II. Indirect

Direct:- in the direct recursion type, the function calls itself repeatedly under certain condition is satisfied.

Indirect:-in the indirect recursion,the function calls another function which eventually causes the first procursion to be called the function the function indirectly calls itself through another function.

Advantages of recursion

1. The code may be easier to write.
2. Recursion reduces the complexity
3. The recursion technique is more natural and compact.
4. The recursion programs can have any number of nesting levels.
5. Recursion reduces the length of code.
6. It is very useful in solving the data structure problem.
7. Useful for Stack operations and infix, prefix, postfix evaluations etc.

Disadvantages of recursion

1. Recursive functions are generally slower than non-recursive function.
2. It may require a lot of memory space to hold intermediate results on the system stacks.
3. Hard to analyze or understand the code.
4. It is not more efficient in terms of space and time complexity.
5. The computer may run out of memory if the recursive calls are not properly checked.

Fibonacci series using recursion in C++

In case of Fibonacci series, next number is the sum of previous two numbers for example 0, 1, 1, 2, 3, 5, 8, 13, 21 etc. The first two numbers of Fibonacci series are 0 and 1.

Let's see the Fibonacci series program in C++ using recursion.

```
#include<iostream>

void printFibonacci(int n){
   static int n1=0, n2=1, n3;

   if(n>0){
       n3 = n1 + n2;
       n1 = n2;
       n2 = n3;
       cout<<n3<<" ";
       printFibonacci(n-1);
   }
}
int main(){
   int n;
   cout<<"Enter the number of elements: ";
   cin>>n;
   cout<<"Fibonacci Series: ";
   cout<<"0 "<<"1 ";
   printFibonacci(n-2);  //n-2 because 2 numbers are already printed
    return 0;
}
```

Output:

Enter the number of elements: 15

Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Ques: What is recursion? Which conditions is necessary in recursion?

Recursion is a powerful programming tool that allows executing the function repeatedly in terms of itself. The recursion reduces the complexity of program.

- ❖ The recursion function must satisfy following condition:

  → The function must have a stopping condition.i.e; it should not continue to call indefinitely.

→ Each time function calls itself, it must be in recursive form, i.e., must be nearer to a solution.