LEARN DSA WITH C++

WEEK :: 08

LEARN DSA WITH C++

COURSE INSTRUCTOR BY
ROHIT NEGI

Check Profile

MADE BY-
PRADUM SINGHA

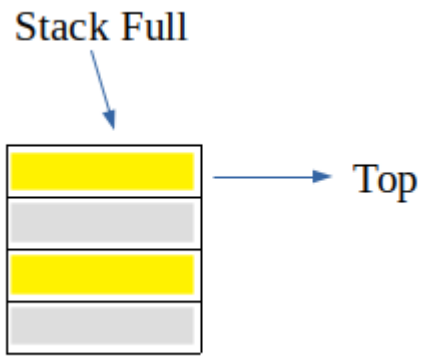
My profile

LEARN DSA WITH C++

WEEK :: 08 **DAY**: 01 **DATE**: 05-06-2023

STACK AND ITS IMPLEMENTATION

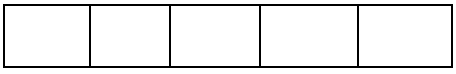
#STACK : A Stack is a linear data structure that follows the LIFO (Last-In-First-Out) principle.

	Put data one by one and take data from top.

When **data Insert** in memory	Use **Push()**
When **data take** from top	Use **Pop()**
When just **watch**	Use **top()**
Check elem, have elem = 1; no elem = 0	Use **empty()**

Create stack

Array



Index = 0 1 2 3 4

top = -1;

Push = 7; top = 0;

Push = 13; top= 1; [increase]

Same way .....

7	13	17	21	31

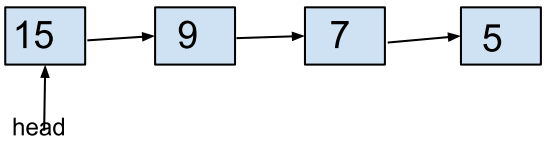
Stack overflow ; no space

Pop()

pop()

Linked list

Insert elem :: 5, 7, 9, 15



Insert elem at front..

Push and pop functions execute in front.

7	13	17		

top =2
If there are no elem top ==-1;
Stack underflow.

Create stack using Array ::

```
#include <iostream>
using namespace std;

class Stack
{
    int top;
    int *arr;
    int size;

public:
    Stack(int s)
    {
        arr = new int(s);
        top = -1;
        size = s;
    };
    // Push element in stack
    void push(int data)
    {
        if (top == size - 1)
        {
            cout << "Stack Overflow\n";
            return;
        }
    top++;
    arr[top] = data;
    }

    // pop element from the array
    void pop()
    {
        if(top ==-1)
    {
        cout<<"Stack underflow\n";
        return;
    }
    top--;
    return;
    }

    // check element in top
    int peek()
```

```
    {
        if(top ==-1)
        {
            cout<<"Stack Underflow\n";
            return -1;
        }
        return arr[top];
    }
    // check arr
    bool empty()
    {
        return top==-1;
    }
};
int main()
{
    Stack s(5);
    s.push(12);
    s.push(125);
    s.push(512);
    s.push(152);
    cout<<s.peek()<<endl;
    s.pop();
    cout<<s.peek()<<endl;
    cout<<s.empty();


    return 0;
};
```

Create Stack using liked list ::

```
#include <iostream>
using namespace std;
#include <stack>

class Node
{
public:
    int data;
    Node *next;
    Node(int data)
    {
        this->data = data;
        next = NULL;
    }
};

class Stack
{
    Node *top;
```

```
public:
    Stack()
    {
        top = NULL;
    }

    // push element
    void push(int data)
    {
        Node *temp = new Node(data);

        if (!temp)
        {
            cout << "Stack Underflow\n";
            return;
        }
        temp->next = top;
        top = temp;
    }

    // pop element
    void pop()
    {
        if (!top)
        {
            cout << "Satack Underflow\n";
            return;
        }
        Node *temp = top;
        top = top->next;
        delete temp;
    }
    // top element
    int peek()
    {
        if (!top)
        {
            cout << "Stack is Empty\n";
        }
        return top->data;
    };
    bool empty()
    {
        return top == NULL;
    }
};
int main()
{
    Stack s;
    s.push(10);
    s.push(30);
```

```
    cout<<s.peek()<<endl;
    cout << s.empty();
    return 0;
};
```

Create stack using STL ::

```
#include<iostream>
#include<stack>
using namespace std;

int main()
{
    stack<int>s;
    s.push(55);
    s.push(11);
    s.push(33);
    s.push(44);
    s.pop();
    cout<<s.top()<<endl;
    cout<<s.empty();

return 0;
};
```

Delete middle element of a stack **<<GeeksforGeeks >>**

```
class Solution
{
    public:
    //Function to delete the middle element of a stack.
    void deleteMid(stack<int>&s, int size)
    {
        // code here..
        stack<int>temp;
        int count = size/2;

        while(count--)
        {
            temp.push(s.top());
            s.pop();
        };
        s.pop();

        while(temp.size())
        {
            s.push(temp.top());
            temp.pop();
        }
    }
};
```

Redundant Braces :: < Interviewbit >

```
int Solution::braces(string A) {

    stack<char>s;
    int count, i=0, n=A.size();

    while(i<n)
    {
        if(A[i]!=')')
        s.push(A[i]);
        else
        {
            count =0;
            while(s.size() && s.top() != '(')
            count++, s.pop();

            if(count<3)
            return 1;
            s.pop();
            s.push('a');
        }
        i++;
    }
    return 0;
}
```

LEARN DSA WITH C++

WEEK :: 08 **DAY**: 02 **DATE**: 06-06-2023

STACK MEDIUM LEVEL QUESTION

Get minimum element from stack :: < GeeksForGeek >>

Logic:: (Original Value + mini-element *101) = value
For Original Value -> value % size and mini-element -> value / size

```
class Solution{
    int minEle;
    stack<int> s;
    public:
       /*returns min element from stack*/
       int getMin(){

           //Write your code here
           if(s.size() ==0)
           return -1;
           else
           return minEle;
       }
       /*returns popped element from stack*/
       int pop(){

           //Write your code here
           if(s.size()==0)
           return -1;

           int ans =s.top();
           ans = ans%101;
           s.pop();
           if(s.size())
           minEle = s.top()/101;
           return ans;
       }
      /*push element x into the stack*/
       void push(int x){

           //Write your code here
           if(s.size()==0)
           {
               minEle = x;
               s.push(x+x*101);
           }
           else
           {
               minEle = min(x,s.top()/101);
               s.push(x+minEle*101);
           }
       }
};
```

String Manipulation << GeeksForGeeks >>

```
class Solution{
    public:
    int removeConsecutiveSame(vector <string > v)
    {
        // Your code goes here
        stack<string>s;
        int i=0;
        while(i<v.size())
        {
            if(s.size()==0)
            s.push(v[i]);
            else
            {
                if(s.top()==v[i])
                s.pop();
                else
                s.push(v[i]);
            }
            i++;
        }
        return s.size();
    }
};
```

Next Greater Element << GeeksforGeeks >>

```
class Solution
{
    public:
    //Function to find the next greater element for each element of the array.
    vector<long long> nextLargerElement(vector<long long> arr, int n){
        // Your code here
        stack<int>s;
        vector<long long >ans(n);
        int i=0;
        while(i<n)
        {
            if(s.size()==0)
            s.push(i);
            else
            {
                if(arr[s.top()]>= arr[i])
                s.push(i);
                else
                {
                    while(s.size() && arr[s.top()]< arr[i])
                    {
                        ans[s.top()] = arr[i];
                        s.pop();
                    };
                    s.push(i);
                }
            }
```

```
            i++;
        }
        while(s.size())
        {
            ans[s.top()]=-1;
            s.pop();
        };
        return ans;
    }
};
```

LEARN DSA WITH C++

WEEK :: 08 **DAY**: 03 **DATE**: 07-06-2023

STACK HARD LEVEL QUESTION

Remove K Digits << GeeksforGeeks >>

```
class Solution {
  public:
    string removeKdigits(string S, int K)
    {
        stack<int>st;
        int i= 0, num;
        while(i<S.size())
        {
            num = S[i] -'0';
            while(st.size() && st.top()>num && K)
            {
                st.pop();
                K--;
            };
            st.push(num);
            i++;
        };

        while(K--)
        st.pop();

        string ans;
        char c;
        while(st.size())
        {
            c = '0' + st.top();
            st.pop();
            ans+= c;
        };

        i = ans.size() -1;
        while(i>=0 && ans[i] == '0')
        {
            ans.pop_back();
            i--;
        };

        reverse(ans.begin(), ans.end());

        if(ans.size()==0)
        return "0";
        else
        return ans;
    }
};
```

Clumsy Factorial << LeetCode>>

```
class Solution {
public:
    int clumsy(int n) {

        stack<int>s;
        int num, i=0;
        s.push(n);
        n--;
        while(n)
        {
            if(i==0)
            {
                num = s.top();
                s.pop();
                s.push(num*n);
            }
            else if(i==1)
            {
                num = s.top();
                s.pop();
                s.push(num/n);
            }
            else
            s.push(n);

            i =(i+1)%4;
            n--;
        };

        stack<int>ans;
        while(s.size())
        {
            ans.push(s.top());
            s.pop();
        };
        int sum = ans.top();
        ans.pop();
        bool flag =0;
        while(ans.size())
        {
            if(flag==0)
            sum+= ans.top();
            else
            sum-= ans.top();
            ans.pop();
            flag =!flag;
        }

        return sum;
    }
};
```

Maximum Rectangular Area in a Histogram << GeeksforGeeks >>

```
class Solution {
public:
    void push_right(long long arr[], int n, int right[]) {
        stack<int> s;
        int i = 0;

        while (i < n) {
            if (s.empty() || arr[i] >= arr[s.top()]) {
                s.push(i);
                i++;
            } else {
                while (!s.empty() && arr[i] < arr[s.top()]) {
                    right[s.top()] = i;
                    s.pop();
                }
            }
        }

        while (!s.empty()) {
            right[s.top()] = n;
            s.pop();
        }
    }

    void push_left(long long arr[], int n, int left[]) {
        stack<int> s;
        int i = n - 1;

        while (i >= 0) {
            if (s.empty() || arr[i] >= arr[s.top()]) {
                s.push(i);
                i--;
            } else {
                while (!s.empty() && arr[i] < arr[s.top()]) {
                    left[s.top()] = i;
                    s.pop();
                }
            }
        }

        while (!s.empty()) {
            left[s.top()] = -1;
            s.pop();
        }
    }

    long long getMaxArea(long long arr[], int n) {
        int* right = new int[n];
        int* left = new int[n];

        push_right(arr, n, right);
        push_left(arr, n, left);
```

```
        long long area = 0;

        for (int i = 0; i < n; i++) {
            long long ans = right[i] - left[i] - 1;
            ans *= arr[i];
            area = max(ans, area);
        }

        delete[] right;
        delete[] left;

        return area;
    }
};
```

Max rectangle << GeeksforGeeks >>

```
class Solution{
  public:
  void push_right(vector<int>&arr, int n, int right[])
  {
      stack<int>s;
      int i=0;

      while(i<n)
      {
          if(s.size()==0)
          s.push(i);
          else
          {
              if(arr[i]>= arr[s.top()])
              s.push(i);
              else
              {
                  while(s.size() && arr[i]< arr[s.top()])
                  {
                      right[s.top()] = i;
                      s.pop();
                  };
                  s.push(i);
              }
          }
          i++;
      }
      while(s.size())
      {
          right[s.top()] = n;
          s.pop();
      }
  }

  void push_left(vector<int> &arr, int n, int left[])
  {
      stack<int>s;
      int i= n-1;
      while(i>=0)
```

```
        {
            if(s.size() ==0)
            s.push(i);
            else
            {
                if(arr[i]>= arr[s.top()])
                s.push(i);
                else
                {
                    while(s.size() && arr[i]<arr[s.top()])
                    {
                        left[s.top()] = i;
                        s.pop();
                    }
                    s.push(i);
                }
            }
            i--;
        }
        while(s.size())
        {
            left[s.top()] =-1;
            s.pop();
        }
    };

    int maxArea(int M[MAX][MAX], int n, int m) {
        // Your code here
        vector<int>sum(m, 0);
        int area = 0, ans;
        int *left = new int[m];
        int *right = new int[m];
        for(int i=0; i<n; i++)
        {
            for(int j=0; j<m; j++)
            {
                if(M[i][j])
                sum[j]++;
                else
                sum[j] = 0;
            }

            push_right(sum, m, right);
            push_left(sum, m, left);

            for(int i=0; i<m; i++)
            {
                ans = right[i] - left[i]-1;
                ans = ans*sum[i];
                area = max(ans, area);
            };
        }

        return area;
    }
};
```

The Celebrity Problem << GeeksforGeeks >>

```
class Solution
{
    public:
    //Function to find if there is a celebrity in the party or not.
    int celebrity(vector<vector<int> >& M, int n)
    {
        // code here
        stack<int>s;
        for(int i=0; i<n; i++)
        s.push(i);
        int first, second;

        while(s.size()>1)
        {
            first = s.top();
            s.pop();
            second = s.top();
            s.pop();
            if(M[first][second] && !M[second][first])
            s.push(second);
            else if(M[second][first] && !M[first][second])
            s.push(first);
        };
        if(s.size() ==0)
        return -1;

        int check = s.top();
        int count_row=0, count_col =0;
        for(int i=0; i<n; i++)
        {
            count_col += M[i][check];
            count_row += M[check][i];
        };

        if(count_row==0 && count_col==n-1)
        return check;
        else
        return -1;
    }
};
```

LEARN DSA WITH C++

WEEK :: 08 **DAY**: 04 **DATE**: 08-06-2023

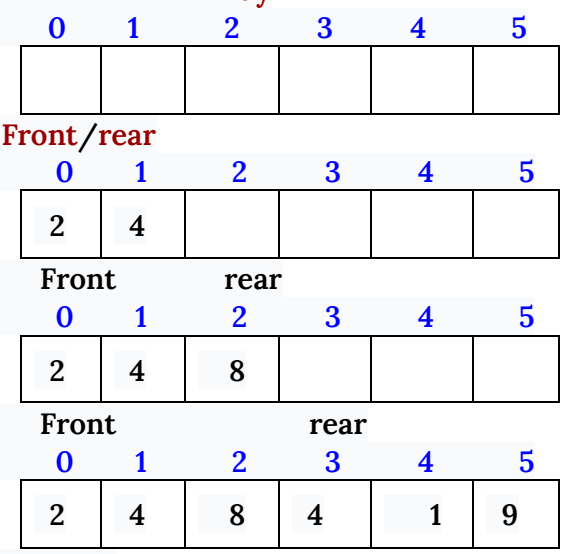
QUEUE

#Queue :: Stand in line.
FIFO:: First in First Out.

The queue data structure follows the FIFO (First In First Out) principle where elements that are added first will be removed first.

How to create QUEUE ::

Array

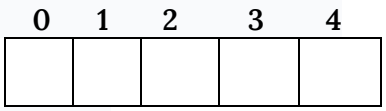



```
Void Push (int x)
{
   if(rear == size)
    {
       cout<<"Queue is full";
       return;
     arr[rear] = x;
        rear++;
}
Void pop( )
{
   if(rear == font)
   {      front = rear =0;
     cout<<"Queue is Empty";
      return ; }
}
```

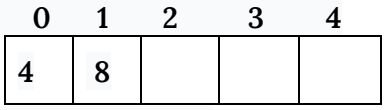
Linked List

```
front = null;
Rear = null;   {Queue = empty}
```

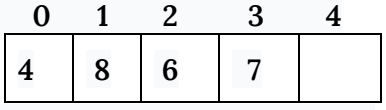
Circular Queue ::



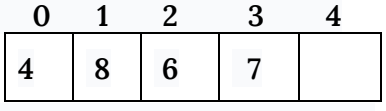
front/rear



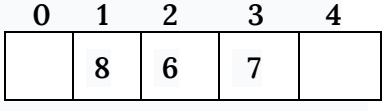
Front rear



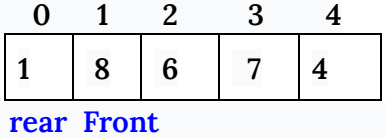
Front rear



Front rear

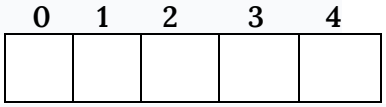


Front rear



rear Front front==rear -> Queue is full

So we initialize rear and front = -1;



rear /Front front = rear =-1 Queue is Empty

```
Void push(int x)
{
  if(Empty() )
   {
    front == rear =0;
    arr front(x);
   }
}
Else if (Full() )
{
   cout<<"Queue is Full";
}
Else
```

case ::

Queue is Empty
Queue is Full
Normal push

```
{
  Rear = (rear + 1)  % size;
arr [rear] = data;
return;
};
```

Pop Operation ::

```
Void pop()
{
       if(Empty() )
       {
        Cout <<"Queue is empty";
        }
       Else if (front =rear)
        {
           Front =rear = -1;
         }
        Else
          {
             Front = (front + 1) % size;
             return;
}
```

#Function::

```
 bool Empty()
{
 return front ==-1 && rear == -1;
}
bool Full()
{
   return front == (rear +1) % size;
}
```

Create Queue Using array:-

```
#include<iostream>
using namespace std;

class CircularQueue
{
    int *arr;
    int front;
    int rear;
    int size;

    public:

    CircularQueue(int size)
    {
        front = rear = -1;
```

```
        arr = new int[size];
        this->size = size;
    }

    bool empty()
    {
        return front ==-1 && rear == -1;
    }

    bool full()
    {
        return front == (rear+1) % size;
    }

    void push(int data)
    {
        if(empty())
        {
            front=rear =0;
            arr[rear]=data;
            return;
        }
        else if(full())
        {
            cout<<"Queue is full\n";
            return;
        }
        else
        {
            rear =(rear+1) % size;
            arr[rear] =data;
            return;
        }
    }

    void pop()
    {
        if(empty())
        {
            cout<<"Queue is Empty\n";
            return;
        }
        else if(front==rear)
        {
            cout<<"Element is popped "<<arr[front]<<endl;
            front = rear = -1;
            return;
        }
        else
        {
            cout<<"Element is popped "<<arr[front]<<endl;
            front = (front +1)%size;
```

```
            return;
        }
    }
};
int main()
{
    CircularQueue q(5);
    q.push(10);
    q.push(11);
    q.push(12);
    q.push(13);
    q.push(14);
    q.push(15);
    q.pop();
    q.push(17);
    q.push(18);
    return 0;
}
```

Create Queue using STL : -

```
#include<iostream>
#include<queue>
using namespace std;

int main()
{
    queue<int>q;
    q.push(10);
    q.push(11);
    q.push(12);
    q.push(13);
    q.push(14);
    q.push(15);
    q.pop();
    q.push(17);
    q.push(18);
    cout<<q.front()<<endl;
    cout<<q.size()<<endl;
    cout<<q.empty()<<endl;

return 0;
};
```

Queue Reversal :: <GeeksforGeeks>

```
class Solution
{
    public:
    queue<int> rev(queue<int> q)
    {
        // add code here.
        stack<int>s;
        while(q.size())
        {
            s.push(q.front());
            q.pop();
        }
        while(s.size())
        {
            q.push(s.top());
            s.pop();
        };
        return q;
    }
};
```

Queue using two Stacks :: < GeeksforGeeks >

```
void StackQueue :: push(int x)
{
    // Your Code
    s1.push(x);
}

//Function to pop an element from the queue by using 2 stacks.
int StackQueue :: pop()
{
        // Your Code
        if(s1.empty() && s2.empty())
        return -1;

        if(s2.size())
        {
            int data = s2.top();
            s2.pop();
            return data;
        };

        while(s1.size())
        {
            s2.push(s1.top());
            s1.pop();
        };
        int data = s2.top();
        s2.pop();
        return data;
}
```

LEARN DSA WITH C++

WEEK :: 08 **DAY**: 05 **DATE**: 12-06-2023

QUEUE ADVANCE

Circular tour << GeeksforGeeks >>

```
class Solution{
  public:

    //Function to find starting point where the truck can start to get through
    //the complete circle without exhausting its petrol in between.
    int tour(petrolPump p[],int n)
    {
       //Your code here
       int deficit =0;
       int balance = 0;
       int start = 0;

       for(int i=0; i<n; i++)
       {
           balance+= p[i].petrol - p[i].distance;
           if(balance <0)
           {
               deficit += balance;
               balance =0;
               start = i+1;
           }
       };

       if(balance + deficit >= 0)
       return start;

       else
       return -1;
    }
};
```

DEQUE

We can insert the element both side (Front and End);
Elm =5;
Elm =6;

4	7	6	1

5	4	7	6	1

5	4	7	6	1	6

Operation ::

```
d.push_back(6);
d.push_front(5);
d.pop_back(5);
d.pop_front();
d.front();
d.back();
```

```cpp
#include<iostream>
#include<deque>
using namespace std;

int main()
{
    deque<int>q;
    q.push_back(5);
    q.push_front(11);

    cout<<q.front()<<endl;
    cout<<q.back()<<endl;
    cout<<q.size()<<endl;
    q.pop_back();
    q.pop_front();
    cout<<q.size()<<endl;

return 0;
};
```

First negative integer in every window of size k << GeeksforGeeks >>

```cpp
vector<long long> printFirstNegativeInteger(long long int A[],
                                            long long int N, long long int K) {

    vector<long long int>ans;
    queue<long long int>q;

    for(int i=0; i<K-1; i++)
    {
        if(A[i]<0)
        q.push(i);
    };

    for(int i=K-1; i<N; i++)
    {
        if(A[i]<0)
```

```
        q.push(i);

        if(q.empty())
        ans.push_back(0);
        else
        ans.push_back(A[q.front()]);

        if((!q.empty()) && q.front() <= i-K+1)
        q.pop();
    };
    return ans;
}
```

Sliding Window Maximum << InterviewBit >>

```
vector<int> Solution::slidingMaximum(const vector<int> &A, int B) {
    vector<int> C;
    if (A.empty() || B <= 0) {
        return C;
    }

    deque<int> window;
    int n = A.size();

    // Process the first window separately
    for (int i = 0; i < B; i++) {
        // Remove elements smaller than the current element from the back of the
deque
        while (!window.empty() && A[i] >= A[window.back()]) {
            window.pop_back();
        }
        window.push_back(i);
    }

    // Maximum of the first window
    C.push_back(A[window.front()]);

    // Process the remaining windows
    for (int i = B; i < n; i++) {
        // Remove elements outside the current window from the front of the deque
        while (!window.empty() && window.front() <= (i - B)) {
            window.pop_front();
        }

        // Remove elements smaller than the current element from the back of the
deque
```

```
        while (!window.empty() && A[i] >= A[window.back()]) {
            window.pop_back();
        }

        window.push_back(i);
        C.push_back(A[window.front()]);
    }

    return C;
}
```