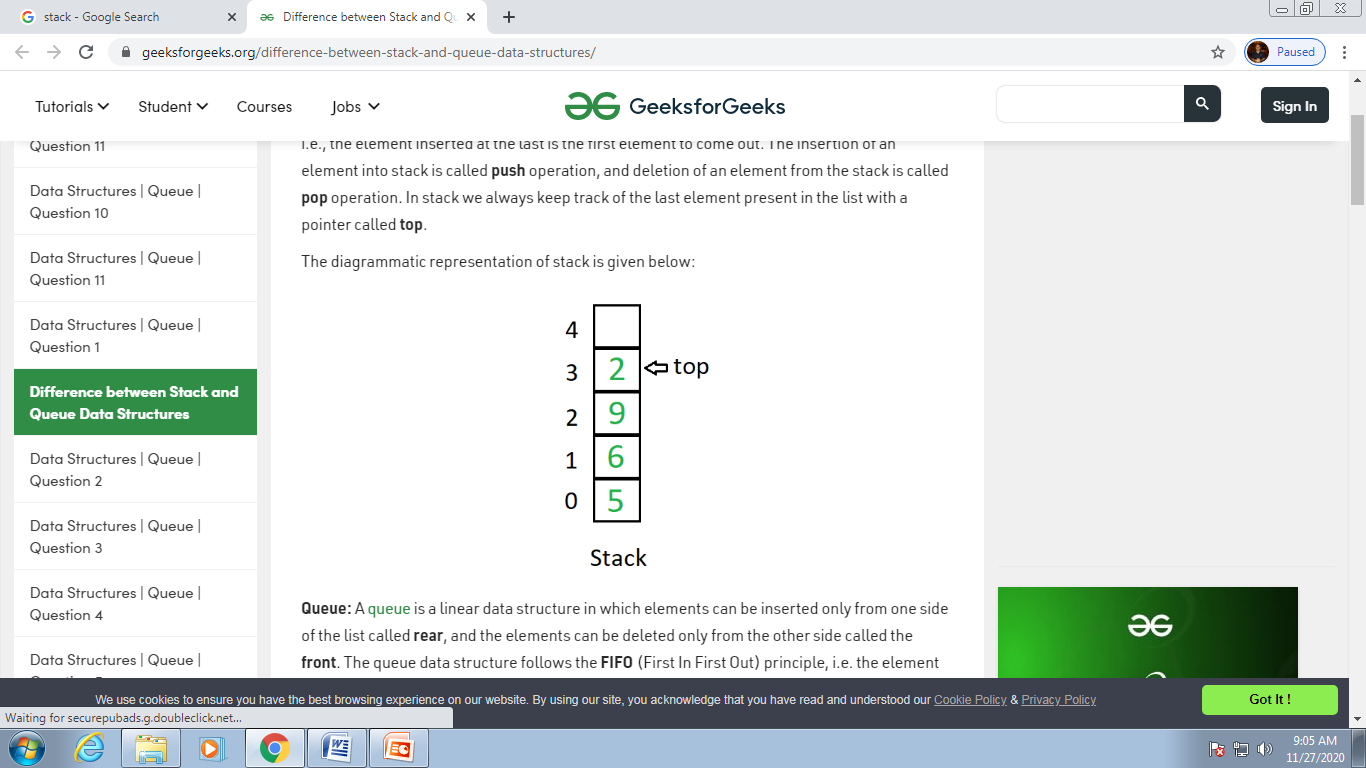
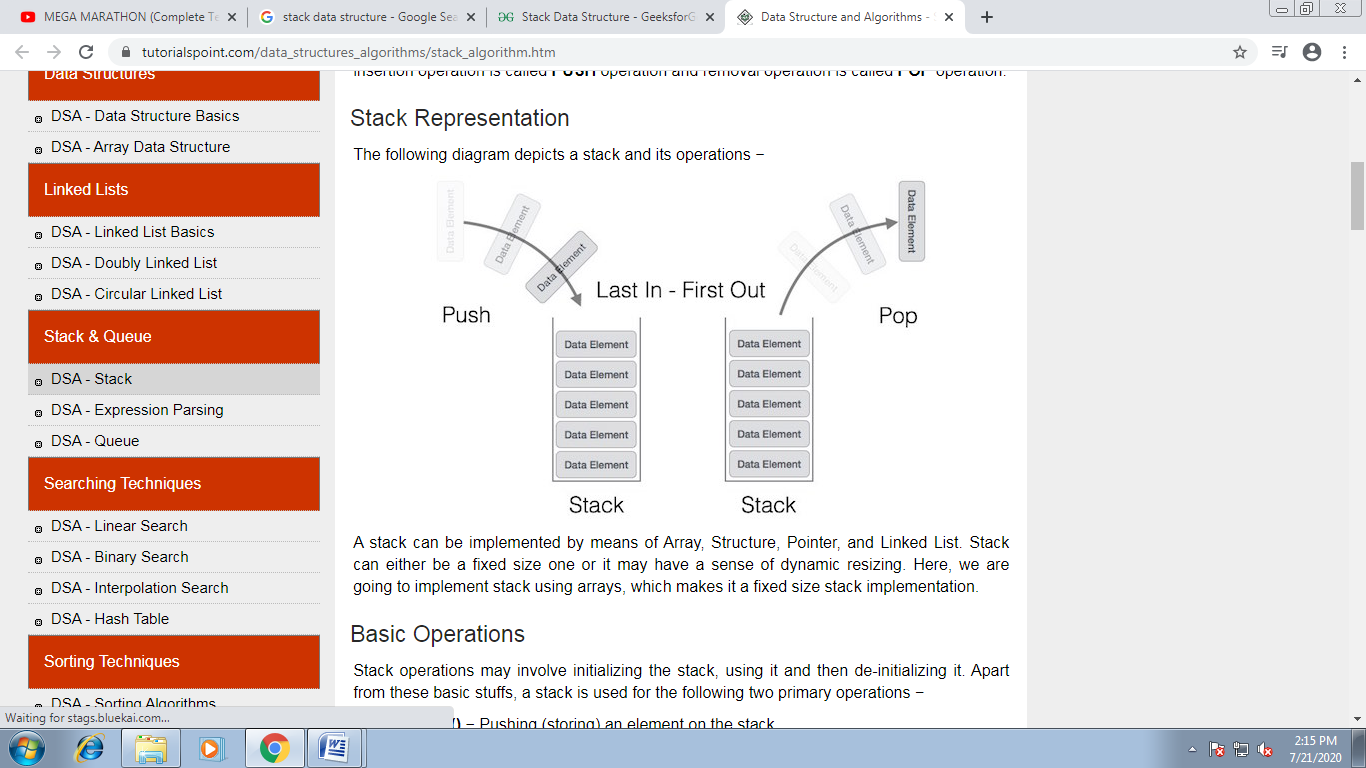
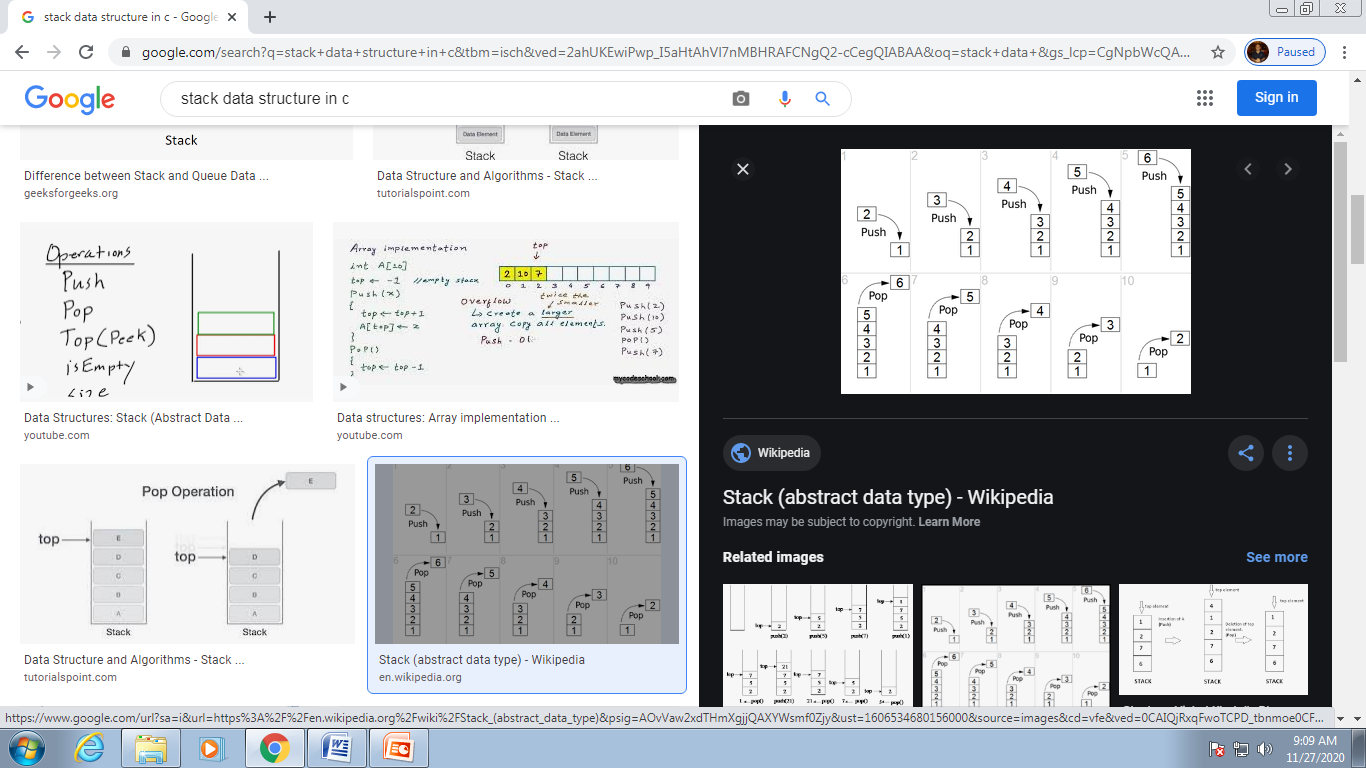
**Stack** is a data structure that allows adding and removing elements in a particular order. Every time an element is added, it goes on the **top** of the stack and the only element that can be removed is the element that is at the top of the stack, just like a pile of objects. It is known as LIFO data structure.





Basic Operations

Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for the following two primary operations −

* **push()** − Pushing (storing) an element on the stack.
* **pop()** − Removing (accessing) an element from the stack.

When data is PUSHed onto stack.

To use a stack efficiently, we need to check the status of stack as well. For the same purpose, the following functionality is added to stacks −

* **peek()** − get the top data element of the stack, without removing it.
* **isFull()** − check if stack is full.
* **isEmpty()** − check if stack is empty.

## Push Operation

The process of putting a new data element onto stack is known as a Push Operation. Push operation involves a series of steps −

* **Step 1** − Checks if the stack is full.
* **Step 2** − If the stack is full, produces an error and exit.
* **Step 3** − If the stack is not full, increments **top** to point next empty space.
* **Step 4** − Adds data element to the stack location, where top is pointing.
* **Step 5** − Returns success.

## Algorithm for PUSH Operation

A simple algorithm for Push operation can be derived as follows −

begin procedure push: stack, data

if stack is full

return null

endif

else

top ← top + 1

stack[top] ← data

end procedure

## Pop Operation

Accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead **top** is decremented to a lower position in the stack to point to the next value. But in linked-list implementation, pop() actually removes data element and deallocates memory space.

A Pop operation may involve the following steps −

* **Step 1** − Checks if the stack is empty.
* **Step 2** − If the stack is empty, produces an error and exit.
* **Step 3** − If the stack is not empty, accesses the data element at which **top** is pointing.
* **Step 4** − Decreases the value of top by 1.
* **Step 5** − Returns success.

## Algorithm for Pop Operation

A simple algorithm for Pop operation can be derived as follows −

begin procedure pop: stack

if stack is empty

return null

endif

data ← stack[top]

top ← top - 1

return data

end procedure

#include<stdio.h>

#include<conio.h>

#define MAX 100 // Macro Template & Macro Expansion.

int push(int stack[], int top);

int pop(int stack[], int top)

void display(int stack[], int top);

void main()

{

int stack[MAX], i, top=-1;

while(1)

{

clrscr();

printf("\n\n\t\tWhat operation you want to perform on stack -- ");

printf("\n\t\t(STACK Limit = 100 elements)");

printf("\n\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t\t1. Push the elements in the stack OR");

printf("\n\t\t2. Pop the elements from the stack OR");

printf("\n\t\t3. Display the elements of the stack OR");

printf("\n\t\t4. Wanna exit.");

printf("\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\n\t\t\tEnter your choice = ");

fflush(stdin);

scanf("%d", &i);

switch(i)

{

case 1: top=push(stack, top);

break;

case 2: top=pop(stack, top);

break;

case 3: display(stack, top);

break;

case 4: printf("\n\n\t\*\*You have been successfully terminated from the program.\*\*");

getch();

exit(0);

default:printf("\n\n\n\ invalid choice.");

printf("\n\t Press any key to continue.......");

getch();

continue;

}

}

}

int push(int stack[], int top)

{

int item;

clrscr();

if(top==MAX - 1)

{

printf("\n\t\Stack is full ");

getch();

exit(0);

return top;

}

else

{

printf("\n\n\tEnter number to insert ");

scanf(“%d”, &item);

top=top+1;

stack[top]=item;

return top;

}

}

int pop(int stack[], int top)

{

int item;

clrscr();

if(top==-1)

{

printf("\n\t\t Stack is underflow(no element in stack) ");

getch();

exit(0);

return top;

}

else

{

item=stack[top];

printf("\n\n\t\tDeleted element is-%d", item);

top=top – 1;

getch();

return top;

}

}

void display(int stack[], int top)

{

int i;

clrscr();

if(top==-1)

{

printf("\n\n\t\tThere is no element in the list");

getch();

exit(0);

}

else

{

for(i=top; i>=0; i--)

{

printf(“\n\n\t Element is- ”);

printf(“\n%d”,stack[i]);

getch();

}

} }

**struct stack**

**{**

**int info;**

**struct stack \*next;**

**};**

**typedef struct stack stack;**

**stack\* push(stack \*top);**

**stack\* pop(stack \*top);**

**void display(stack \*top);**

**void main()**

**{**

**int i;**

**stack \*top;**

**top=NULL;**

**while(1)**

**{**

**clrscr();**

**printf("\n\n\t\tWhat operation you want to perform on STACK -- ");**

**printf("\n\n\t\t1. Push the elements in the stack OR");**

**printf("\n\n\t\t2. Pop the elements from the stack OR");**

**printf("\n\n\t\t3. Display the elements of the stack OR");**

**printf("\n\n\t\t4. Wanna to exit.");**

**printf("\n\n\n\n\t\t\t Enter your choice = ");**

**scanf("%d", &i);**

**switch(i)**

**{**

**case 1: top=push(top);**

**break;**

**case 2: top=pop(top);**

**break;**

**case 3: display(top);**

**break;**

**case 4: printf("\n\n\t\*\*You have been successfully terminated from the program.\*\*");**

**getch();**

**exit(0);**

**default:printf("\n\n\t\t WRONG Press any key to choose again .......");**

**getch();**

**continue;**

**} } }**

**stack\* push(stack \*top)**

**{**

**stack \*tmp;**

**clrscr();**

**tmp=(stack\*)malloc(sizeof(stack));**

**printf("\n\n\t\tEnter the number to push into stack =");**

**scanf("%d", &tmp->info);**

**tmp->next=top;**

**top=tmp;**

**printf("\n\n\t\t\*\*Your element has been successfully pushed in.\*\*");**

**getch();**

**return top;**

**}**

**stack\* pop(stack \*top)**

**{**

**clrscr();**

**if(top==NULL)**

**{**

**printf("\n\n\t\t\*\*There is no element in the list to pop out.\*\*"**

**"\n\n\t\tPlease insert some element first then try again!!!");**

**getch();**

**return top;**

**}**

**else**

**{**

**printf("\n\n\t\t\*\*The top element has been popped out.\*\*");**

**printf(“\n\n\t\t\*\*The poped element is %d ”,top->info);**

**getch();**

**return top->next;**

**}**

**}**

**void display(stack \*top)**

**{**

**stack \*tmp;**

**clrscr();**

**if(tmp==NULL)**

**{**

**printf("\n\n\t\t\*\*There is no element in the stack to display.\*\*");**

**getch();**

**}**

**else**

**{**

**printf("\n\n\t\tYour elements are as follows --\n\n");**

**for(tmp=top;tmp!=NULL;tmp=tmp->next)**

**printf("\n\n\t\t\t\t%d",tmp->info);**

**getch();**

**}**

**}**