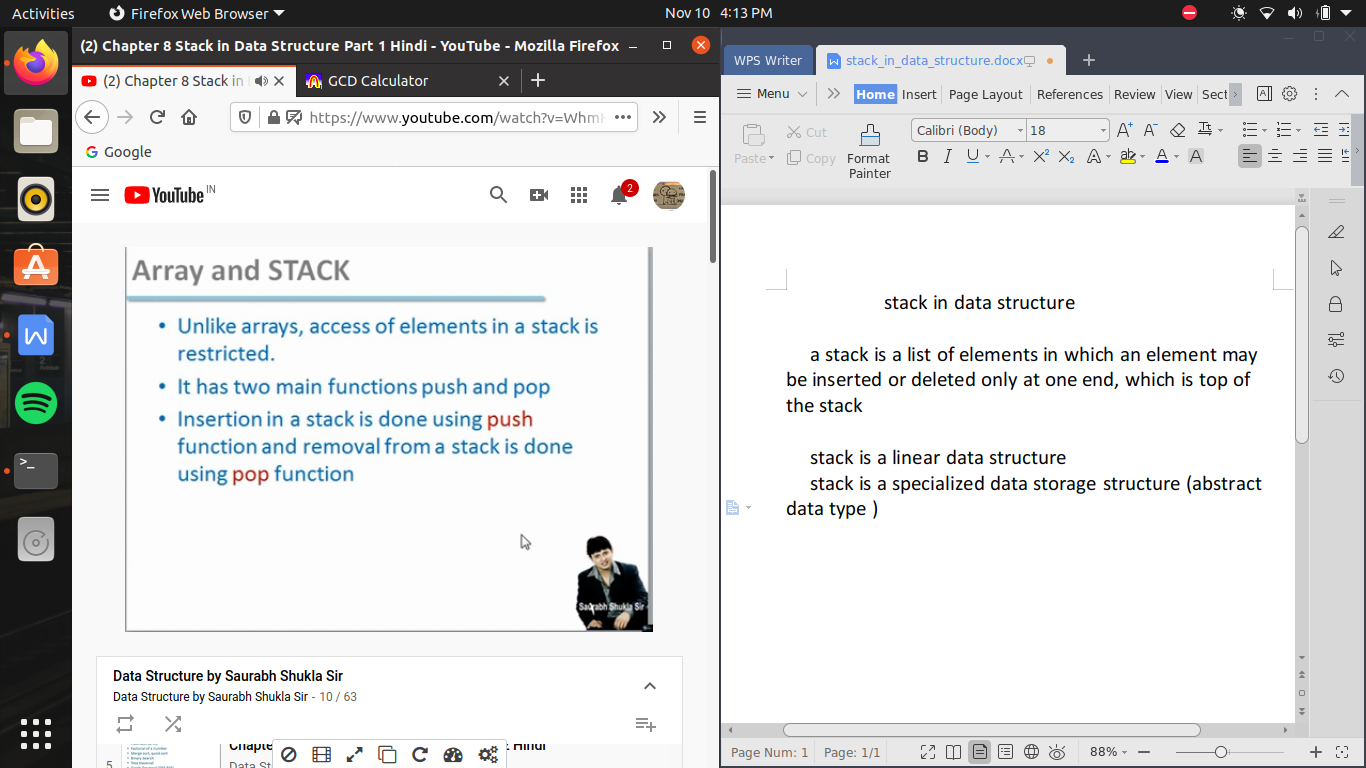
stack in data structure

a stack is a list of elements in which an element may be inserted or deleted only at one end, which is top of the stack

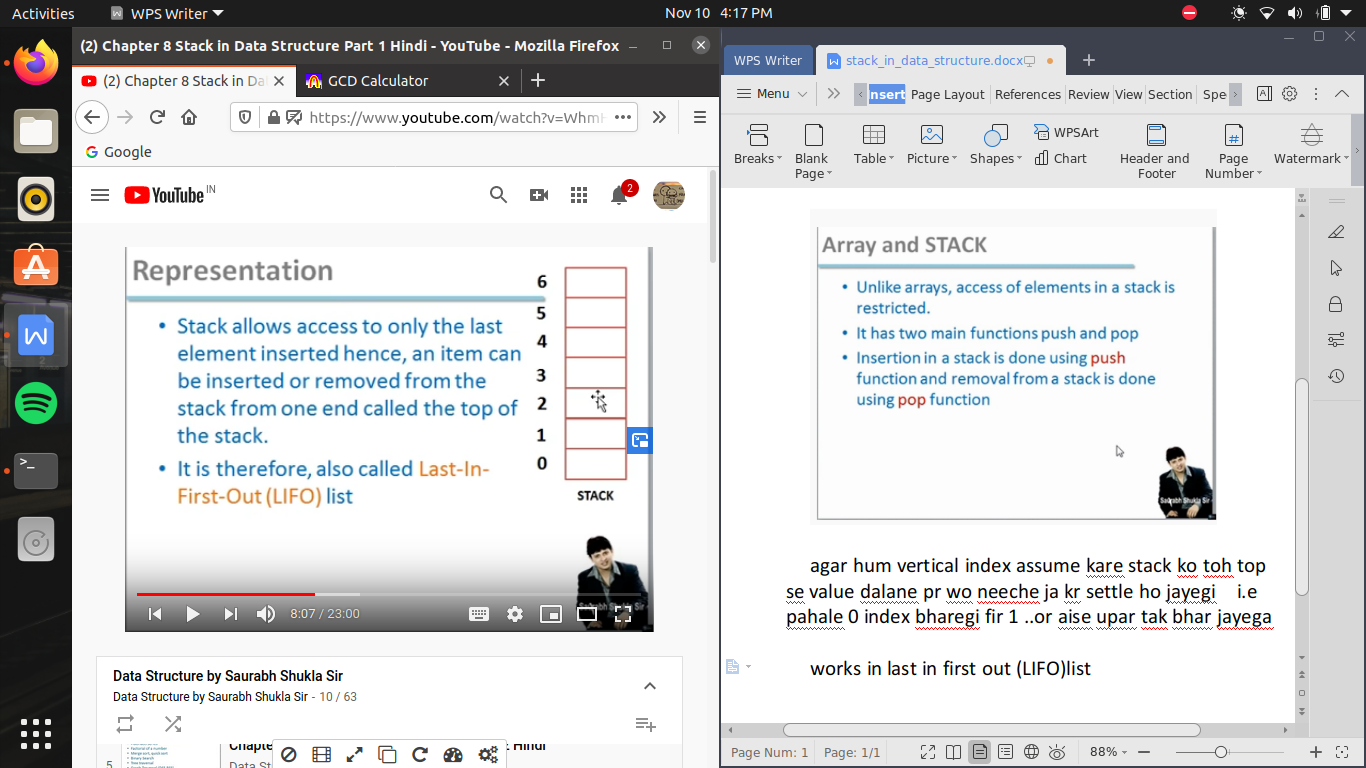
stack is a linear data structure

stack is a specialized data storage structure (abstract data type )



agar hum vertical index assume kare stack ko toh top se value dalane pr wo neeche ja kr settle ho jayegi i.e pahale 0 index bharegi fir 1 ..or aise upar tak bhar jayega

works in last in first out (LIFO)list



stack can be implemented as

1. array
2. dynamic array
3. linked list

array implementation->

1. push -> insert data
2. pop -> pull out data

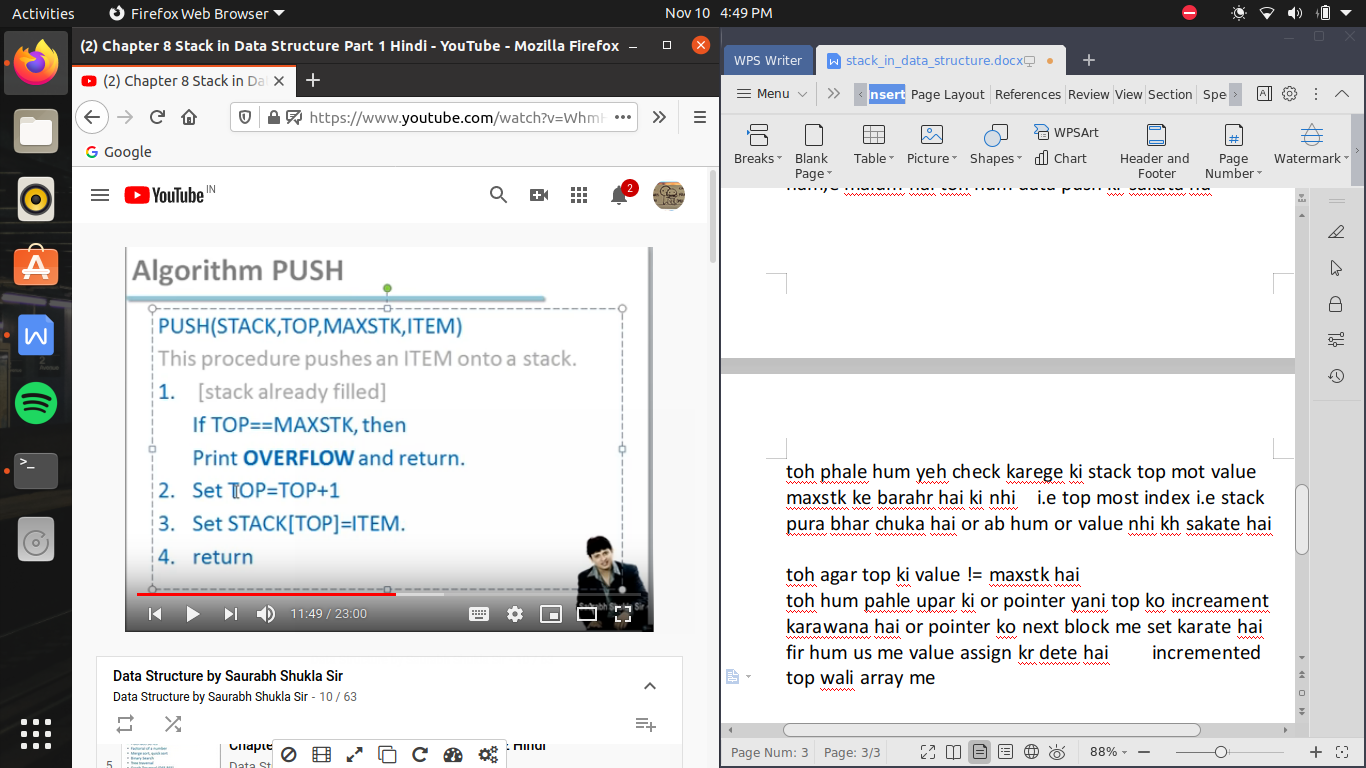
algorithm PUSH ->

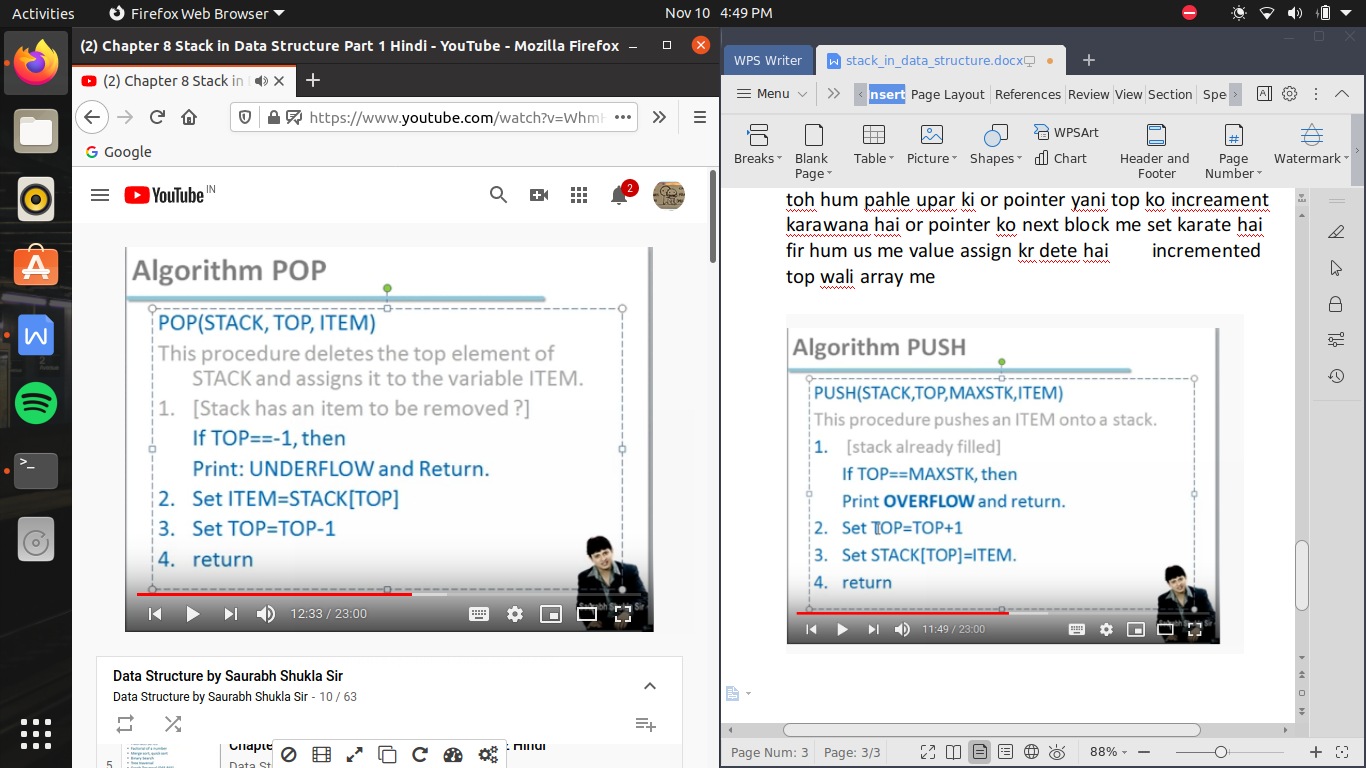
agr hame pata ho ki stack kon si hai, top most element kaha pe rakha hai yani index pata chl jaye , or hame yeh pata chl jaye ki max kitane elements rkh sakate hai , or hame kon sa item rakhana hai toh agar yeh 4 bate agar ham,e malum hai toh hum data push kr sakata hu

toh phale hum yeh check karege ki stack top mot value maxstk ke barahr hai ki nhi i.e top most index i.e stack pura bhar chuka hai or ab hum or value nhi kh sakate hai

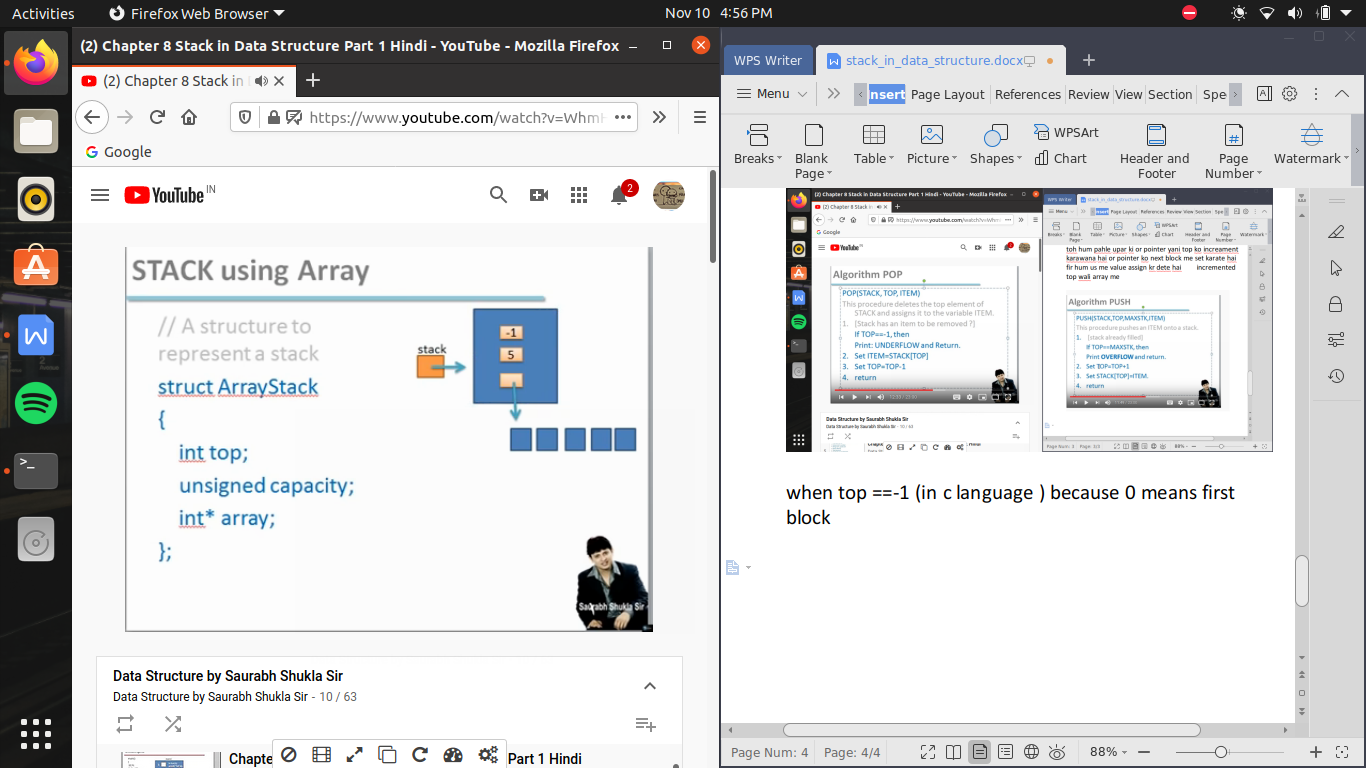
toh agar top ki value != maxstk hai

toh hum pahle upar ki or pointer yani top ko increament karawana hai or pointer ko next block me set karate hai fir hum us me value assign kr dete hai incremented top wali array me





when top ==-1 (in c language ) because 0 means first block



/\* stack implementation through array \*/

#include<stdio.h>

#include<stdlib.h> //malloc function defined in this lib

struct arraystack

{

int top;

int capacity; //agar hum unsigned likhate hai toh es ka mtlb hai ki hum hamesha positive valuse rakhana chahate hai

int \*array;

};

struct arraystack \* createstack(int cap)

{

struct arraystack \*stack;

//yaha hum ne ek pointer banaya hai stack nam ka or kyu ki es ka data type arraystack hai toh yeh kisi arraystack ke variable ka address rkh sakata hai

stack= malloc(sizeof(struct arraystack)); //here we are passing the address of the a variable of arraystack as a whole

stack->capacity=cap;

//here we specifed the size of stack

//here stack is a pointer point a varaible made by struct -> (arrow)

//and a object i.e normal variable point modified variable by struct so we use . (dot)

stack->top=-1;

//-1 here showing stack is fully empty

stack->array= malloc (sizeof(int)\*stack->capacity);

// here we make array let say onle variable contain 4 bytes and we need to make 10 varibles in array thus

//malloc make a array of 40 bytes in coontinuous form

//ptr i.e pointer pass the address of stack in which memory is allocated by malloc function because here in programing we don't know the size of stack to be made, it is based upon user

return(stack);

//here by passing the address of pointer of arraystack we can access all the data by this address

}

int is\_full(struct arraystack \*stack)

{

if (stack->top==stack->capacity-1)

//-1 because stack index start from 0

return(1);

else

return(0);

}

int is\_empty(struct arraystack \*stack)

{

if (stack->top==-1)

return (1);

else

return(0);

}

void push(struct arraystack \*stack, int item)

{

if (!is\_full(stack))

{

stack->top++;

stack->array[stack->top]=item;

}

}

int pop(struct arraystack \*stack)

{

int item;

if (!is\_empty(stack))

{

item=stack->array[stack->top];

stack->top--;

return (item);

}

return(-1);

}

void main()

{

struct arraystack \*stack;

int choice,cap,item;

printf("enter the capacity of stack\n");

scanf("%d",&cap);

stack=createstack(cap);

while(1)

{

printf("\n 1.push");

printf("\n 2.pop");

printf("\n 3.exit");

printf("\n enter your choice \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\n enter a number \n");

scanf("%d",&item);

push(stack,item);

break;

case 2:

item=pop(stack);

if (item==-1)

printf("\n stack is empty \n");

else

printf("\n poped value is %d \n",item);

break;

case 3:

exit(0);

}

}

}

//malloc is the short name for memory allocation and is used to dynamically allocate the single block of contiguous memory according to the size specified

// malloc function simply allocate the memory block according to size specified and on success it returns a pointer (specifically a void pointer because it doesn't know which type of memory it is pointing to as because input is done by user)to the first byte of allocated memory else return NULL

//unsigned int is the positive value of integer

//and thus is your resposibility to type cast the void pointer into an appropriate type

