Stack

- · Last in First Out
- · Flements can be added or removed only only from one end
- · Gives access only be element at the top

 od data structure.

Definition

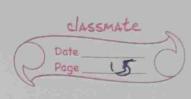
- -> An ordered collection of homogenous data items
- -> can be accessed at at only one end (the top)

Operations

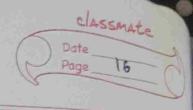
- -> Create an empty stack
- Check if it is empty
- -> Push: add an element to the top
- -> POP: remove the top element.
- -> peek : retrieve the top element (Not the deletion)
- -> Destroy: Remove all the elements one by one a destroy the data structure.

The Stack ADT Value definition:

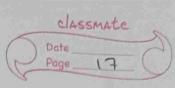
Abstract type ded StakkType (FlementType ele)



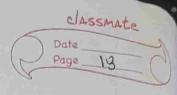
| | Operator definition: |
|----|--|
| 1) | Abstract Stack Type CreateStack() |
| -> | Precondition: None |
| -5 | Post condition: Empty Stack is created |
| | The state of the s |
| | Abstract Stack Type PushStack (Stack Type Stack, |
| 2) | Element Type Element) |
| | Precondition: Stack not full or Not full (Stack)= Free- |
| | Postcondition: Stack = Stack + Element at the |
| | top or Stack = Original stack with new flement. |
| | |
| | at the top. |
| | 2 51 th (Sirich Fudé Hack). |
| 3) | Abstract Stack Type PopStack (Stack Fype stack). |
| -> | Becondition: Stack not empty or NotEmpty (Stack) |
| | |
| -> | PARCICIO - PIPMENT OUTRE CIT |
| | a long lonent alto |
| | Original Stack without top Flement. |
| | Origina Store |
| 43 | Abstract Destroy Stack (Stake Type Stack) - |
| 4) | Abstract Destroy Stack (Stack Not Empty or Not Empty) Brecondition: Stack not empty or Not Empty |
| | Reconditions. |
| - | (stack) = True. Post condition: Elements from the stack Post condition: Elements from the stack |
| | Post condition: Elements from the are removed one by one starting from |
| | are removed one |
| | In P to bottom. |
| | Empty(stack) = True |
| | |



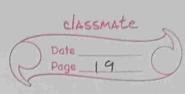
| 5) | Abstract Boolean NotFull (Stark Type Stack) |
|---------------|--|
| | |
| -> | Post condition: Not Full (Stack) - The |
| | 10 Stack is not full |
| | Not Full (Stack) = False it stack is dull |
| الأدداء ا | BA FURNISH THE STATE OF THE STA |
| 6) | Abstract Boolfan Not Empty (Stack Type Stack) |
| | Precondition : None |
| -7 | Post condition o Not Empty (Stack) = True 1) |
| | Stack is not empty |
| | Not Empty (Stack) = False id stack is emp |
| 7) | Abstract Element Type Peek & Stack Type Slack |
| 9 | Precondition: Stack not empty or Not Empty (Stack) |
| | = True. |
| -7 | Postcondition: Peek Stack = element at the top |
| ETT BAS | Stack = Orginal Stack. |
| | The test of the same of the sa |
| | |
| d 5 3 1 7 (5) | ATEMPANE Y STADIE WANTED A LOS MAN A RESTAR |
| ing think | A SHEET AND MAN AND RELEASE ABOUT THE RESERVENCE OF THE PARTY OF THE P |
| | |
| 0/0/0/1 | Total metallic de la final de la contraction del |
| C POLO CONTO | MIT (SIZ) Beautal bas towards and |



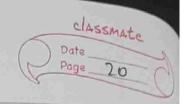
| | Date Page 17 |
|--|--|
| | |
| | Implementation od Stack |
| 3 | Three ways |
| -> | Array |
| 13 | Vector 2 A VAR MALLES A HORIER A A MALLES |
| The same of the sa | linked List |
| 41 | SUATE (1855/2) W to mad told fill i rait! |
| I | Disc Array (30 310) pla 3 1 + 09) slides |
| | Daluan to as (30340.) 300 12 907 43 1391 |
| 9 | Best performance |
| | 1 -1.10 12 12 10 P |
| -) | Fixed 5128 for 1 33 long to made |
| | 0100 |
| | |
| 1 | Algorithm Stack Type CreateStack () |
| | 6 |
| | integer Stack tope = -1; |
| | |
| | Return Stack; |
| 2) | A. III - Ctc.ck Type roshsiae |
| 11111 | Element Type Element) |
| | |
| 1118 | if (Not Full(stack)) = Frue |
| | if (Not Full(stack)) = Element; Stack (++ Stanck TOMP) = Element; |
| | IIE. VELTON'I |
| 0 | Else Terror |
| 2 | Alcothen Flement 191 |
| | Stack) Stack) = True Stack) = True |
| | S Pd (NO+ Empty (Stack)) |
| | · · · · · · · · · · · · · · · · · · · |



| | Return Stack [Stack Top=-] |
|--------------|---|
| | Return Stack [Stack Top=-]; Else "Error"; |
| | 2 PANAG |
| 4) | Abstract DestroyStack CStack Type Stack |
| | & 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
| | id (Not Empty (Stack)) = True |
| | while (Not Empty (stack)) |
| | print Popstack (Stack); |
| | Else "Errorli, STADMONTONO PESSE |
| | 3 - Same - Sportalov Coszill |
| 5) | Abstract Boolean Not Full CStack Type |
| | Stack) |
| | & ? of NOTFULI (Stack) |
| | return True; 91) shotz misropial |
| | CIER |
| | retun Fuder: sant asstance |
| lita gazaria | Petern Stack |
| | Abstract Boolean Not Empty (StackTypest |
| | 2) Algorithm Stack Type Push Stack (Step) |
| | id Not Empty (Stack) |
| | return True; |
| | else surficted by |
| | return False; Caron stable 14 Januaries |
| | 2 |
| 7) | Abstract Element Type Beek (Stack Type Stal |
| TROPE | of (Not Empty (Stack) = True 1 1001 |
| | Retorn Stack & Stack Top); |
| 1 | Else 4 Ertot 11 ; sold to the land to the 161 ? |



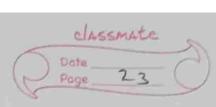
| 1 | Vector |
|-------|---|
| | any amount occurre |
| | In a common of cutton policy of color |
| -) | of the check when data size a vector size |
| | |
| | Slowest when data size ? vector size |
| 3 | |
| > | can grow to unlithited size |
| 3 | |
| 9 | Algo |
| | Same as parray |
| 1912) | 10 517,9 (N) 29 VI TO |
| 3 | Linned List |
| -4 | Adva-Nfage Adva-Nfage Constant + 1 mne to push or pop |
| | Adva-Nfage Always constant + 1 me to push or pop |
| | Can grow to our |
| | Disadvantage |
| | s Slowest method |
| | Struct Node Type & Element Type Flement; |
| 9 | Struct Node Type & Element Type Flement; |
| | Node Type Next; |
| | Louis (C) |
| | 3 Stack Type (xeatestac |
| 1+1 | Algorithm Stack Type (reatestack () |
| | 2 10 (TOP); |
| | Create Node (TOP); |
| | TOP=NULL; |



2) Stack Type Push Stack (Stack Type Stack Node Type New Mode) i) TOP == NULL & NewNode -> Next = NULL; Top=NewNode; ? F1862 NewNode -> Next=TOP; Top=New Node, ? ODA 3) Algorithm Flement Type Popstack (Stack Type Stock) Laid Hornell if TOD = = NULL Error Print 'Underflow'; Else Create Node (Temp); Temp=top; 2 3911 Mois 10012 5 TOP 2 TOP > Next; Retorm (Temp -> Dafa); 4) Abstracte Destroy Stack (Stack Type Stack) if Top == Nole
Print "Underflow"; FIEC

| | create Node (Temp); | |
|----|---|-----|
| | While (Not Empty (Stack)) | |
| | C I of corolled railing and give | |
| | Temp= Top; | |
| | Top= Top > Next; | |
| | Return (Temp -) Data); | |
| 1 | 2 | |
| -1 | Abstract Element Type Peep (Stack Type Stack | 6 |
| 5) | Abstract Element Green | 7 |
| | el Tanana Nolli | |
| | if Top==NULL | |
| | Print " Frror" | |
| | Elée | |
| | Return Top -> Datas | |
| | | - |
| | 3 | 2 |
| 6 | 3 Abstract Display Stack (Stack Type Stack | (z) |
| 6 | \$ | (z) |
| 6 | S 2 TOO == NULL | الم |
| 6, | \$ | (z) |
| 6) | S ? Top == NULL Print "Error" Else | 2) |
| 6, | S ? Top == NULL Print "Error" Else 2 | 2) |
| 6 | S ? Top == NULL Print "Error" Else ? Create Node Type (Temp); | (2) |
| 6) | ? ? Top == NULL Print "Error" Else ? Create Node Tope (Temp); Temp=1Top; | (2) |
| 6 | ? ? Top == NULL Print "Error" Else 2 Create Node Type (Temp); Temp: 1Top; 12) & le (Temp! = NULL) { | (2) |
| 6) | Print "Error" Else 2 Create Node Tope (Temp); Temp=!Top; While (Temp! = NULL) { Print (Temp > classe); | (2) |
| 6, | 2 Create Node Tope (Temp); Temp=1Top; Whele (Temp) = NULL) { Print (Femp > clada); Tempi> Next; | (2) |
| 6 | Print "Error" Else 2 Create Node Fore (Temp); Temp=1Top; While (Temp1 = NULL) { Print (Femp > clasta); Temp1> Next; 3 | (2) |
| 6, | 2 Create Node Tope (Temp); Temp=1Top; Whele (Temp) = NULL) { Print (Femp > clada); Tempi> Next; | |

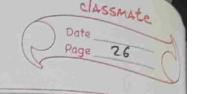
| A section of | | | Date Page 22 |
|---------------|--------------|-------------|--|
| | | | |
| + | Application | onlamar | 1 about at up man |
|) | | | g 7 fuer 31.40 |
| Ey | 1/P = String | g = d()(| 2 |
| | Input | Stach | & Basically |
| | - 9 | 2 | Once bopo |
| | | \$0 | The Parently |
| [dnot2.sq.,] | | 2 | mat stack is For |
| | | § (-) | in the out put |
| | 2 | Timolia | |
| |) | | 100 = = 90 [] |
| 2) (1 | fix to Posa | 6. | novy a track |
| -> Ps | ecedence | MAN CO | Elde |
| 17 1 | | | OT COLORS |
| () × | 100000 | 01-0 en/g | end imported to |
| (1)+, | | | |
| | | | 09 8 8 90T 69 |
| E9 M | *N+TAQ | 1F # A+B | Normal Printerior |
| | | | 6186 |
| | put 5 | tack | Output |
| () N | | TOTAL TOTAL | |
| 2) * | | * | 401 Minas |
| 3) N | 3 | * 1011= | MN |
| (e) + | | (toka)sc | 1 MW* |
| 5) 7 | of Many and | + 1100 | MNXTI |
| 6) 1 | 4 | ^ | Wh*1 |
| 7) 9 | + | ^ | MN*TQ |
| 8) / | + | / | MN*TRN |
| | | | Mark Control of the C |



| | | Page 23 | | | |
|----|--|------------|----------------------|--|--|
| | | | | | |
| 9) | E 200 | +/ | MNXTQ1F | | |
| - | * | (+*) *(2) | MN*TQ^F/ | | |
| 1 | A | 0+* | MN*TQNF/A | | |
| | + | + | MN*TQ1F1A*+ | | |
| | В | Empty | MN* TQ^F/A*+B+ | | |
| | | | | | |
| | If inside the stack is higher precendent. sign in Step(2) There is a lower precendent sign in There is a lower precendent sign in | | | | |
| | | | | | |
| | | | | | |
| | Ma : ni | I I POPS | the Higger precedent | | |
| | | Ga Multou | + in Step (9) | | |
| | | and is a | Omber 1, collagour | | |
| -> | when there is a Lower precedent sign when there is a Lower precedent sign in the Stack & higher in input the in the Stack & higher in input the | | | | |
| | higher is Pushed int the stack without popping the Lower precendent sign in | | | | |
| | | | | | |
| | poppingt | re power | | | |
| | |) A | 0 1 101 17 | | |
| -> | If equa | precena | ent pop he one in | | |
| | Stackar | nd push to | | | |
| | (10) | 3/4 | | | |

| | | 300 | |
|-------------------|------------------------------|------------------|--|
| | Infix tog | post-fly with Pe | arenthesis |
| - Eg | (((A+B)*C)) + ((PFE)*(F+G))) | | |
| | | Stach | |
| +* | MOLDE | +46 | A. Hear |
| 4.8 | CHARLE | CC + | 431 |
| 4 | | CCC | |
| रहा करतावर् | Athan | 600 | - Alizai II - |
| -1 | + | CCC+ | AAIAMA |
| PALLADI | B | (C C + s - | AB |
| Personal T | 3 | CC 2049 | PB+ |
| - | * | (C* 940) | AB+ |
| The state of | Caranta | CC* | AB+C |
| | 1 - POH - 1 | MC wada'nd & | A B+C* |
| 1000 | S.C. (1775) | MC - 1 hads | A B+C* |
| 1 | (cetas) | C-C and | A B+C* 909 |
| | (| C-CC | A B+C* |
| 019 9 | D | C-CC | AB+C+D |
| [6] | FULL N. A. | C-CC- | A B+C*D |
| - | E | C-CC- | AB+C*DE |
| |) | C-C | AB+C*DE- |
| | * contraction | C-C* | AB+C*DE- |
| | F | C-C*C | AB+CKDE- |
| DE ROLL | | C-C*C | AB+C*DE-F |
| 4 | | C-C*C+ | AB+C* DE-F |
| G | | C-C*(+ \$ | AB+C*DE-FG |
|) | | (-C* | ABIC*DE+F6+ |
| | | C- | AB+C*DE-FG+* |
| | | EMPTY | AB+C*DE-FG+*- |
| The second second | | | and the second s |

a so basically once you short the pareathosis whatever was in the the parenthesis it gets porged a) Postfix evaluation of Create a stack for storing operands - scan the input expression from Left to right. S) Reverse a string using stack -> Crente a Stack (empty) -> One by one Push the all the characters -> One byone pop all the characters from Stack & put them back to string. 6) Check if a string is Palidrome -) 00 (5) -> Check if equal hentrup else false 7) Recursion -> Calling the same function directly or inderect > Represente a problem in terms of one or more smaller Problems, and add one or more base conditions that stop the records - The maximal no od nested calls is called recursion depth



Recursive function Call

- -) Correct function is paused
- pis remembered in a special data DS called execution context stack.
- -> The nested call executes
- Junction is resumed from where it stopped
- > Each recursive call, needso to save
 - · Current value
 - · local variable
 - . Reform Address
- -) Also, as a function calls to another function first its arguments, then the return address I finally local variable is proshed.

Backtracking

Jt is an algorithmic - technique for solving problems recursively by trying to build a solution in creanentally, one piece at a time

Removing those solutions that fail to sodies by the constraints of the problem at any point of time