**DAY-17(19-11-2024)**

1)Question: what if we delete the node at the beginning in the circular linked list then which node is the head node

1. Free the allocated memory (for structures and poinnters )

Ans) iterate through the head node to NULL ,we take while loop and consider two pointers same like a node deletion.

1. Count the number of nodes in a linked list.

**STACKS:**

* LIFO (last in first out)
* Refer notes
* The insertion and deletion is done from the top
* Stack is grow from bottom to top
* If we can see the bottom means the stack is empty ,if we can see the top means it reachs the limit.
* If TOP==BOTTOM🡺 stack is empty
* Top==limit 🡺 stack is full
* In stack or container the size/limit/max should be fixed.
* Insertion/push: the condition is check if the stack is full 🡺 the top is growing from the bottom to top
* Deletion/pop/pull/remove: the condition is to check if the stack is empty🡺 the top is moving(shrank) from the top to bottom
* HOW TO AVOID MULTIPLE DEFINITIONS OF VARIABLES?
* WHY THE GLOBAL DECLARATION IN HEADER FILE IS WORKING BUT NOT IN THE .C FILE?
* Program for the stack :

#include <stdio.h>

#include<stdlib.h>

#define MAX 5

int bottom=-1;

int top=-1;

int push(int);

int pop();

int stackCont[MAX];

int push(int v)

{

if(top==MAX-1)

{

printf("\n stack is full");

return 0;

}

top++;

stackCont[top]=v;

}

int pop()

{

int v;

if(top==bottom)

{

printf("\n stack is empty");

return 0;

}

v= stackCont[top];

top--;

return v;

}

void displayStack()

{

int iter;

if(top==bottom)

{

printf("\n stack is empty");

return;

}

printf("\n stack elements are:");

for(iter=top;iter>bottom;iter--)

{

printf("\n%d",stackCont[iter]);

}

printf("\n\n");

}

int main() {

int v;

// v=pop();

//printf("\n %d element is deleted from the stack",v);

push(10);

push(20);

push(30);

push(40);

push(50);

push(60);

displayStack();

int i;

for(i=0;i<MAX;i++)

{

v=pop();

printf("%d element is deleted",v);

displayStack();

}

printf("\n\n");

return 0;

}

Output:

stack is full

stack elements are:

50

40

30

20

10

50 element is deleted

stack elements are:

40

30

20

10

40 element is deleted

stack elements are:

30

20

10

30 element is deleted

stack elements are:

20

10

20 element is deleted

stack elements are:

10

10 element is deleted

stack is empty

stack with array program:

#include <stdio.h>

#include <stdlib.h>

#define CAP 5

int top, bottom;

int push(int [], int);

void dispStack(int []);

void pop();

int main()

{

int arr[CAP];

int ch=1, ele;

top = 0;

bottom = 0;

while(ch)

{

printf("\n\tEnter the Value to be insert in stack: ");

scanf("%d",&ele);

push(arr, ele);

printf("\nDo you want to add/insert new element to the stack (0/1): ");

scanf("%d",&ch);

}

dispStack(arr);

pop(arr);

dispStack(arr);

pop(arr);

dispStack(arr);

pop(arr);

dispStack(arr);

pop(arr);

dispStack(arr);

pop(arr);

dispStack(arr);

printf("\n\n");

return 0;

}

int push(int stackArr[], int ele)

{

if(top > (CAP-1))

{

printf("\n\tStack is full\n");

return 0;

}

stackArr[top] = ele;

top++;

return 1;

}

void pop(int a[])

{

if(top == bottom)

{

printf("\n\tStack is Empty\n");

return ;

}

else

{

printf("\n\t%d element is popped out successfully",a[top-1]);

top--;

}

}

void dispStack(int stackArr[])

{

int i;

if(top == bottom)

{

printf("\n\tStack is Empty\n");

return ;

}

printf("\n\tELements in the stack are\n");

printf("\t\t");

for(i=top-1;i>=0;i--)

printf("%d ",stackArr[i]);

printf("\n\n");

}

Stack program using menu based:

#include <stdio.h>

#include <stdlib.h>

#define CAP 5

int top, bottom;

int push(int [], int);

void dispStack(int []);

void pop();

int main()

{

int arr[CAP];

int ch=1, ele;

top = 0;

bottom = 0;

while(ch)

{

printf("\n\tPress,");

printf("\n\t1. Push Operation");

printf("\n\t2. Pop Operation");

printf("\n\t3. Display Stack");

printf("\n\t4. Exit");

printf("\n\tChoice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\n\tEnter the Value to be insert in stack: ");

scanf("%d",&ele);

push(arr, ele);

break;

case 2:

pop(arr);

break;

case 3:

printf("\n");

dispStack(arr);

break;

case 4:

printf("\n\tThank You for using Stack app\n");

exit(EXIT\_SUCCESS);

default:

printf("\n\tEnter the correct choice");

}

}

printf("\n\n");

return 0;

}

int push(int stackArr[], int ele)

{

if(top > (CAP-1))

{

printf("\n\tStack is full\n");

return 0;

}

stackArr[top] = ele;

top++;

return 1;

}

void pop(int a[])

{

if(top == bottom)

{

printf("\n\tStack is Empty\n");

return ;

}

else

{

printf("\n\t%d element is popped out successfully",a[top-1]);

top--;

}

}

void dispStack(int stackArr[])

{

int i;

if(top == bottom)

{

printf("\n\tStack is Empty\n");

return ;

}

printf("\n\tELements in the stack are\n");

printf("\t\t");

for(i=top-1;i>=0;i--)

printf("%d ",stackArr[i]);

printf("\n\n");

}

Using DMA(stack):

#include <stdio.h>

#include <stdlib.h>

#define INITIAL\_CAPACITY 5

int top, bottom;

int push(int \*, int, int\*);

void dispStack(int \*, int);

void pop(int \*, int\*);

int main()

{

int \*arr;

int ch = 1, ele, capacity = INITIAL\_CAPACITY;

arr = (int \*)malloc(capacity \* sizeof(int));

if (arr == NULL)

{

printf("Memory allocation failed!\n");

return 1;

}

top = 0;

bottom = 0;

while (ch)

{

printf("\n\tEnter the value to be inserted in stack: ");

scanf("%d", &ele);

if (!push(arr, ele, &capacity))

{

printf("\n\tError: Stack is full.\n");

break;

}

printf("\nDo you want to add/insert a new element to the stack (0/1): ");

scanf("%d", &ch);

}

dispStack(arr, capacity);

pop(arr, &capacity);

dispStack(arr, capacity);

pop(arr, &capacity);

dispStack(arr, capacity);

pop(arr, &capacity);

dispStack(arr, capacity);

pop(arr, &capacity);

dispStack(arr, capacity);

pop(arr, &capacity);

dispStack(arr, capacity);

// Free dynamically allocated memory

free(arr);

return 0;

}

int push(int stackArr[], int ele, int \*capacity)

{

if (top >= \*capacity)

{

\*capacity \*= 2;

stackArr = (int \*)realloc(stackArr, (\*capacity) \* sizeof(int));

if (stackArr == NULL)

{

printf("\nMemory reallocation failed!\n");

return 0;

}

}

stackArr[top] = ele;

top++;

return 1;

}

void pop(int stackArr[], int \*capacity)

{

if (top == bottom)

{

printf("\n\tStack is Empty\n");

return;

}

else

{

printf("\n\t%d element is popped out successfully\n", stackArr[top - 1]);

top--;

}

if (top < \*capacity / 4 && \*capacity > INITIAL\_CAPACITY)

{

\*capacity /= 2;

stackArr = (int \*)realloc(stackArr, (\*capacity) \* sizeof(int));

if (stackArr == NULL)

{

printf("\nMemory reallocation failed!\n");

return;

}

}

}

void dispStack(int stackArr[], int capacity)

{

if (top == bottom)

{

printf("\n\tStack is Empty\n");

return;

}

printf("\n\tElements in the stack are:\n");

printf("\t\t");

for (int i = top - 1; i >= 0; i--)

{

printf("%d ", stackArr[i]);

}

printf("\n\n");

}

**QUEUE:**

* Refer notes
* Create the employee records dynamically display it using stack operation?

CUNIT:

* It is a unit testing framework for c
* Interface is name of the function/library
* The name of the function is preponed with test word

Int max(){

}

Void test\_max(void)

{

CU\_ASSERT( our output == expected output);🡺true;

}

* The cunit assertion is used to test the logical conditions
* Refer the documentation sent in whatsapp group(systemwith linux)
* **Expression handling for validate the phone numbers**
* There are two types of test cases

1. Sunny test case 🡪positive
2. Rainy 🡪negative