Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ22ECSBL1)

Experiment no: 2 Date: 08/08/2024

Aim: Write a program in C/Java to implement stack

Write a program in C/Java to reverse a string using stack.

Software Language: C

Theory: A stack is a linear data structure that follows the principle of **Last In First Out (LIFO)**. This means the last element inserted inside the stack is removed first.

Basic Operations of Stack:

There are some basic operations that allow us to perform different actions on a stack.

- Push: Add an element to the top of a stack
  - IsFull: Check if the stack is full(overflow condition)
- Pop: Remove an element from the top of a stack
  - IsEmpty: Check if the stack is empty(underflow condition)
- Peek: Get the value of the top element without removing it

Algorithm for push operation:

Step 1 − Checks if the stack is full.(top==max-1) where max is size of array.

Step 2 − If the stack is full, then display “overflow” 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 pop operation:

Step 1 − Checks if the stack is empty. (top == -1)

Step 2 − If the stack is empty, then display “underflow” and exit.

Step 3 − If the stack is not empty, access the data element at which top is pointing.

Step 4 − Decrease the value of top by 1.

Step 5 − Returns success.

Algorithm for peek operation:

Step 1 - Check whether stack is EMPTY. (top == -1)

Step 2 - If it is EMPTY, then display "Stack is EMPTY!!!" and terminate the function.

Step 3 - If it is NOT EMPTY, then display top element of the stack.

Step 4 - Return success

Algorithm to reverse a string:

Step 1 – Create an empty stack.

Step 2 - Pick the characters from the string one by one and put them to the stack, so that the last character of the string comes at the top of the stack.

Step 3 - Pop the stack and put the popped characters back in the empty string.

Code:

Code and output for reverse a string :

```c
#include <stdio.h>
#include <string.h>
#define size 20

char stack[size];
char stack2[size];
int top = -1;
int top2 = -1;

int push(char a) {
  if (top == size - 1) {
    printf("Stack Overflow");
  }
  top++;
  stack[top] = a;
  return 0;
}

char pop() {
  char c = stack[top];
  stack[top] = 0;
  top--;
  return c;
}

int main() {
  char a[20];
  printf("Enter the string\n");
  scanf("%[^\n]s", &a);
  for (int i = 0; i < strlen(a); i++) {
    push(a[i]);
  }
  for (int i = 0; i < strlen(a); i++) {
    stack2[i] = pop();
  }
  printf("%s", stack2);
  return 0;
}
```

```
Enter the string
Everything is a joke
ekoj a si gnihtyrevE
```

Code and output for push, pop, peek and display algorithm :

```c

#include <stdio.h>

#define size 4
int top = -1;
int stack[size];

int push(int a) {
  if (top == size - 1) {
    printf("stack is full\n");
    return 0;
  }
  top++;
  stack[top] = a;
  printf("Inserted number is %d\n", a);
  return 0;
}

int pop() {
  if (top == -1) {
    printf("stack is empty\n");
  }
  printf("Number removed is %d\n", stack[top]);
  stack[top] = 0;
  top--;
  return 0;
}
```

```
28
29  int peek() {
30    if (top == -1) {
31      printf("There is no element in the stack\n");
32      return 0;
33    }
34    printf("The last entered element is %d\n", stack[top]);
35  }
36
37  int display() {
38    if (top == -1) {
39      printf("There is no element in the stack\n");
40      return 0;
41    }
42    for (int i = 0; i <= top; i++) {
43      printf("\n%d\n", stack[i]);
44    }
45    return 0;
46  }
47
```

```
47
48  int main() {
49    int choice, a;
50    int exit = 1;
51    while (exit) {
52      printf("1.push\n2.pop\n3.peek\n4.display\n");
53      scanf("%d", &choice);
54      switch (choice) {
55      case 1:
56        printf("Enter the number to be inserted\n");
57        scanf("%d", &a);
58        push(a);
59        break;
60
61      case 2:
62        pop();
63        break;
64
65      case 3:
66        peek();
67        break;
68
69      case 4:
70        display();
71        break;
72
73      default:
74        break;
75      }
76      printf("Do you wanna exit ? If yes then press 0\n");
77      scanf("%d", &exit);
78    }
79
80    return 0;
81  }
```

```
1.push
2.pop
3.peek
4.display
1
Enter the number to be inserted
35
Inserted number is 35
Do you wanna exit ? If yes then press 0
1
1.push
2.pop
3.peek
4.display
1
Enter the number to be inserted
56
Inserted number is 56
Do you wanna exit ? If yes then press 0
1
1.push
2.pop
3.peek
4.display
1
Enter the number to be inserted
98
Inserted number is 98
Do you wanna exit ? If yes then press 0
1
1.push
2.pop
3.peek
4.display
1
Enter the number to be inserted
90
Inserted number is 90
Do you wanna exit ? If yes then press 0
1
1.push
2.pop
3.peek
4.display
1
Enter the number to be inserted
56
stack is full
```

jay

```
Do you wanna exit ? If yes then press 0
1
1.push
2.pop
3.peek
4.display
4
35
56
98
90
Do you wanna exit ? If yes then press 0
1
1.push
2.pop
3.peek
4.display
2
Number removed is 90
Do you wanna exit ? If yes then press 0
1
1.push
2.pop
3.peek
4.display
4
35
56
98
Do you wanna exit ? If yes then press 0
1
1.push
2.pop
3.peek
4.display
3
The last entered element is 98
Do you wanna exit ? If yes then press 0
```

Result and Conclusion:

From the following practical, we learned how the stack works and what are different algorithms or methods used in in stack data structure and how to implement in C.