| | | |
|---|---|---|
| | PUNE INSTITUTE OF COMPUTER TECHNOLOGY PUNE - 411043 | |
| | Department of Electronics & Telecommunication | |
| | ASSESMENT YEAR: 2021-2022 | CLASS: SE-5 |
| | SUBJECT: DATA STRUCTURES | |
| **EXPT No:** | LAB Ref: SE/2021-22/ | Starting date: 29/11/2021 |
| | Roll No:22108 | Submission date:5/12/2021 |
| **Title:** | **Stack and Queue Operations using Singly Linked List** | |
| | | |
| **Problem statement** | Implement stack and queue using linked list. Perform push, pop, insert and operations on it. | |
| **Prerequisites:** | Basics of C programming | |
| | Decision making and loop controls | |
| | Choice based program | |
| | Functions, Stack and Queue, linked list | |
| **Objectives:** | Learn to create and display a stack and queue | |
| | Implement various operation on stack and queue to understand its effect on data. | |
| | | |
| **Theory:** | | |

| | |
|---|---|
| | 1) **Functions –**<br><br>• It is a self-contained block of statements that perform task of some kind.<br>• C program may have one or more functions<br>• C program must have at least one function i.e. main()<br>• There is no limit on number of functions<br>• Each function is called in a sequence specified by the function calls in main()<br>• After each function has done its job, control returns to next location from where it has been called.<br><br>2) **Array –**<br><br>• Arrays are used to store multiple values in a single variable. An array is a special variable, which can hold more than one value at a time.<br>• They are used to store similar type of elements as in the data type must be the same for all elements.<br>• Declared as – (datatype) (varname)[10], example – int arr[10]. Here, 10 is the limit of no. of elements predefined in the array.<br><br>3) **Stack –**<br><br>• It is an ordered group of homogeneous items of elements.<br>• Elements are added to and removed from the top of the stack (the most recently added items are at the top of the stack). |

| | |
|---|---|
| | • Elements in between the stack cannot be removed or element cannot be |

added between the stack, for that all the elements above it needs to be removed and stored separately and then finally we will be able to ‘pop’ (remove/delete permanently) or ‘push’ (add element),

- The last element to be added is the first to be removed (LIFO: Last In, First Out).

4) **Queue** –

- It is an ordered group of homogeneous items of elements.
- Elements are added at one and removed from the other (at the end).
- The last element to be added is the first to be removed (FIFO: First In, First Out).

5) **Linked List –**

- Linked lists are linear data structures, elements here are stored using pointers unlike arrays..
- Dynamic memory allocation is used in linked list, thus making insertion and deletion easier than arrays.
- Linked list has two parts, first is the data and second is the pointer, consider there are 3 nodes (nodes – building blocks of a linked list), in a linked list first node has pointer of second node, second node has pointer of third and third has no pointer data as there is no other node after it. The pointer of first node is contained by head variable.
- Initialization:

```
struct Node
{
    int data;
    struct Node *ptr;
};
struct Node *head;  (head node initialized)
```

| **Code** | Stack Operations:<br>#include<stdio.h><br>#include<stdlib.h> int<br>choice, i, checker = 0;<br><br>struct Node<br>{     int<br>val;<br>    struct Node *next;<br>};<br><br>struct Node* top; |
|---|---|

```
void end() {
    printf("\n==========This is the end of execution!==========");
}

void create() {
    top = NULL;
}
void display(struct Node *top)
{
    struct Node *ptr;

    ptr = top;

    if(ptr != NULL)
    {
        printf("\n The elements in Stack:\n");
while (ptr!=NULL)
        {
            printf("%d", ptr->val);
ptr = ptr->next;
```

```
        }
end();
}     else
    {
        printf("\n The Stack is empty");
        end();
    }
}

void push(struct Node *top)
{     int
ele, n;
    printf("\nEnter the number of elements in the stack:
");     scanf("%d", &n);     for (i = 0; i < n; i++)
    {
        printf("\nEnter the Value to be pushed in the stack: ");
scanf("%d", &ele);
        if (top == NULL)
        {
            top = (struct Node *)malloc(sizeof(struct
Node));             top->next = NULL;             top->val =
ele;
        }
else
```

```
        {
            struct Node *temp = (struct Node *)malloc(sizeof(struct
Node));             temp->next = top;             temp->val = ele;
            temp = top;
        }
    }

    printf("\nDo you wish the stack to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);
    if (checker == 1)
    {
display(top);
return;     }
else     {
return;
    }
}

void pop(struct Node *top)
{
    if(top == NULL)
    {
        printf("\n\t Stack is empty");
    }
else
{
        top = top->next;
        printf("Popped value = %d", top->val);
        free(top);
    }
    printf("\nDo you wish the stack to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);
    if (checker == 1)
    {
display(top);
return;     }
else     {
end();
return;
    }
}

void insert(struct Node *top)
{
    int ele;
    struct Node *new = (struct Node *)malloc(sizeof(struct Node));

    printf("\nEnter the new element to be inserted in the stack: 
");     scanf("%d", &ele);     new->val = ele;
    new->next = top;

    printf("\nDo you wish the stack to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);
    if (checker == 1)
    {
display(top);
end();
return;     }
else     {
end();
return;
    }
}

int main()
{
    printf("==============Stack          Operations==============");
printf("When the elements will be entered and push to the stack.\n Enter the
elements one by one-\n");
    push(top);

    printf("\nThere are total 3 operations:\n 1)Pop\n 2)Insert\n 3)Exit\nEnter
your choice: ");
    scanf("%d", &choice);

    switch (choice)
    {
    case 1: //Pop
    {
pop(top);
break;     }
    case 2: //Insert
    {
insert(top);
break;     }
    case 3: //Exit
    {
end();
return 0;
    }
default:
    {
        printf("\nPlease enter a valid choice between numbers 1 to 3! Try
again!!");         break;
    }     }
return 0;
}
```

Queue Operations

```
#include <stdio.h>
#include<stdlib.h>
int choice, i, checker = 0;

struct Node {     int
val;     struct Node
*next;
}; struct Node*
rear;
struct Node *front;

void end()
{
    printf("\n==========This is the end of execution!==========");
return;
}

void create()
{    front =
NULL;     rear =
NULL;
}

void display(struct Node *front, struct Node *rear)
{
    struct Node *ptr;

    ptr = front;

    if (ptr != NULL)
    {
        printf("\n The elements in Queue:\n");
while (ptr!=NULL)
        {
            printf("%d", ptr->val);
ptr = ptr->next;
        }
end();

}
else
    {
        printf("\n The Queue is empty");
end();
    }
}

void dequeue(struct Node *front, struct Node *rear)
{
    struct Node *ptr = front;

    if (front == NULL)
    {
        printf("\nQueue is empty.");

return;
}    else
    {
        printf("\n\t The popped element is %d", front->val);
        front = front->next;
free(front);
    }
    printf("\nDo you wish the Queue to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);
    if (checker == 1)
    {
        display(front, rear);

return;
}    else
{
end();
return;
    }
}
```

|  |  |
|---|---|
|  | <pre>void enqueue(struct Node *front, struct Node *rear)<br>{     int<br>ele;<br>    struct Node *n = (struct Node *)malloc(sizeof(struct Node));<br><br>    printf("Enter the element which needs to be pushed to the back of Queue:<br>");     scanf("%d", &ele);<br><br>    if (n == NULL)<br>    {<br>        printf("The Queue is full!");<br>        return;<br>    }<br><br>    n->val = ele;     n-<br>>next = NULL;<br>    if (front == NULL)<br>    {<br>        front = rear = n;<br>    }<br>else<br>{<br>        rear->next = n;<br>rear = n;<br>    }<br><br><br>}<br><br>int main()<br>{    int<br>ele, n;<br>    rear = (struct Node *)malloc(sizeof(struct Node));<br>    printf("==============Queue  Operations==============\nEnter<br>the number of elements in the Queue (max = 100): ");<br>scanf("%d", &n);<br>    printf("When the elements will be entered and push to the Queue.\n Enter<br>the elements one by one-\n");<br><br>    create();<br><br>    for (i = 0; i < n; i++)</pre> |

```
    {
        enqueue(front, rear);
    }

    printf("There are total 3 operations:\n 1)Insert\n 2)Delete\n 3)Exit\nEnter
your choice: ");
    scanf("%d", &choice);

    switch (choice)
    {
    case 1: //Enqueue
    {
        enqueue(front, rear);
        printf("\nDo you wish the Queue to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);          if (checker == 1)
```

```
        {
            display(front, rear);
            return 0;
        }
else
{
            return 0;
        }
break;
    }
    case 2: //Dequeue
    {
        dequeue(front, rear);
        break;
    }
    case 3: //Exit
    {
end();
return 0;
    }
default:
    {
        printf("\nPlease enter a valid choice between numbers 1 to 3! Try again!!");         break;
    }
}
    return 0;
}
```

**CONCLUSION:**

**In this experiment, we are able to implement stack and queue operations using SLL. We applied POP, Push and Delete in Stack and Insert and Delete in Queue using SLL.**

**REFERENCES:**

Seymour Lipschutz, Data Structure with C, Schaum's Outlines, Tata McGrawHill

E Balgurusamy - Programming in ANSI C, Tata McGraw-Hill (Third Edition)

Yashavant Kanetkar- Let Us C, BPB Publication, 8th Edition.

| | |
|---|---|
| | |
| | |

**Continuous Assessment for DS AY 2021-22**

| **RPP (5)** | **SPO (5)** | **Total (10)** | **Signature:** |
|---|---|---|---|
| | | | **Assessed By: Mr. V. B. Vaijapurkar** |
| **Start date** | **Submission date** | | **Date:** |
| **29/11/2021** | **5/12/2021** | | **Roll. No.22108** |
| ***Regularity, Punctuality, performance**<br>***Submission, Presentation, orals** | | | |

Page