DSA-Assignment-03

Page :
Date :

Experiment No (3)

AIM – Implementation of stack using a linked list. Use this stack to perform evaluation of a postfix expression.

Objective ⟶

1. To understand the concept of abstract data type
2. How different data structures such as arrays and stacks are represented as an ADT.

Theory ⟶

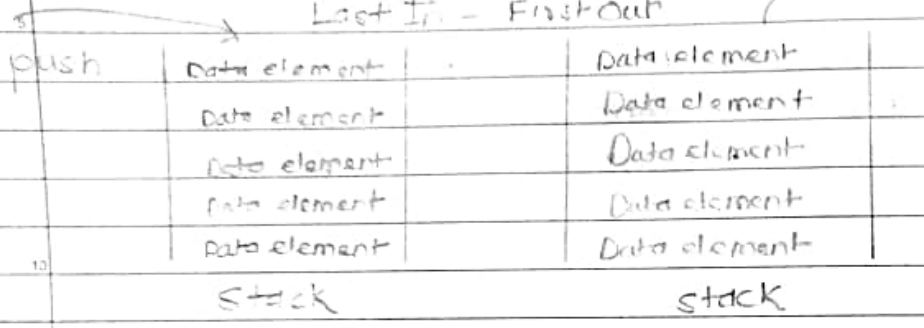
- A stack is an abstract data type (ADT), commonly used in most programming languages. It is named as stack as it behaves like real world stack, for example a deck of cards or a pile of plates, etc.
- A real world stack allows operations at one end only. For example, we can place or remove a card or plate from the top of stack only. Likewise stack ADT allows all the data operation at one end only.
- At any given time, we can only access the top element of a stack. This feature makes it a LIFO Data structure. (Lifo stands for last in first out).
- Here the element which is added or inserted at last will be first accessed.
- In stack terminology ~~iteration~~ insertion operation is called as push operation and deletion is known as pop opera(n).

rudraksh.karpe.cs@ghrcem.raisoni.net

Page :
Date :

(*) Stack. Representation →

The following diagram depicts a stack and its operations —



A stack can be implemented by means of Array, structure, pointer and linked list. Stack can either be a fixed size one or it may have a sense of dynamic resizing. Here, we are going to implement stack using arrays, which makes it fixed size stack implementation.

Basic operation →

Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for following two primary operations →

- push () → Pushing (storing) an element on the stack.
- pop () → Removing (accessing) an element from the stack.

rudraksh.karpe.cs@ghrcem.raisoni.net

Page :
Date :

— When data is pushed onto stack.
— To use a stack efficiently, we need to check the status of stack as well. For the same purpose the below functions are used.

- peek() — get the top element of the stack without removing it.
- isFull() — check if stack is full.
- isEmpty() — check if stack is empty.

At all the times, we main maintain a pointer to the last pushed data on the stack. As this pointer always represents the top of the stack, hence we named it as top. The pointer provides the top values of the stack without actually remaing it.

rudraksh.karpe.cs@ghrcem.raisoni.net

Implementation of Stack using array:

Program code:

```
#include <iostream>
using namespace std;
int stack[100], n = 100, top = -1;
void push(int val)
{
   if (top >= n - 1)
      cout << "Stack Overflow" << endl;
   else
   {
      top++;
      stack[top] = val;
   }
}
void pop()
{
   if (top <= -1)
      cout << "Stack Underflow" << endl;
   else
   {
      cout << "The popped element is " << stack[top] << endl;
      top--;
   }
}
void display()
```

```
{
   if (top >= 0)
   {
      cout << "Stack elements are:";
      for (int i = top; i >= 0; i--)
         cout << stack[i] << " ";
      cout << endl;
   }
   else
      cout << "Stack is empty";
}
int main()
{
   int ch, val;
   cout << "1) Push in stack" << endl;
   cout << "2) Pop from stack" << endl;
   cout << "3) Display stack" << endl;
   cout << "4) Exit" << endl;
   do
   {
      cout << "Enter choice: " << endl;
      cin >> ch;
      switch (ch)
      {
      case 1:
      {
         cout << "Enter value to be pushed:" << endl;
         cin >> val;
         push(val);
         break;
      }
```

```
        case 2:
        {
           pop();
           break;
        }
        case 3:
        {
           display();
           break;
        }
        case 4:
        {
           cout << "Exit" << endl;
           break;
        }
        default:
        {
           cout << "Invalid Choice" << endl;
        }
        }
    } while (ch != 4);
    return 0;
}
```

rudraksh.karpe.cs@ghrcem.raisoni.net

Output:

```
PS R:\GHRCEM\DSA new lab> cd "r:\GHRCEM\DSA new lab\" ; if ($?) { g++ Stack_array.cpp -o Stack_array } ; if ($?) { .\Stack_array }
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
```

```
Enter choice:
1
Enter value to be pushed:
4
Enter choice:
1
Enter value to be pushed:
5
Enter choice:
3
Stack elements are:5 4 2
Enter choice:
2
The popped element is 5
Enter choice:
```

Implementation of stack using LinkedList:

Program code:

```
#include <iostream>
using namespace std;
struct Node
{
   int data;
   struct Node *next;
};
```

rudraksh.karpe.cs@ghrcem.raisoni.net

```
struct Node *top = NULL;
void push(int val)
{
   struct Node *newnode = (struct Node *)malloc(sizeof(struct Node));
   newnode->data = val;
   newnode->next = top;
   top = newnode;
}
void pop()
{
   if (top == NULL)
      cout << "Stack Underflow" << endl;
   else
   {
      cout << "The popped element is " << top->data << endl;
      top = top->next;
   }
}

void display()
{
   struct Node *ptr;
   if (top == NULL)
      cout << "stack is empty";
   else
   {
      ptr = top;
      cout << "Stack elements are: ";
```

```
        while (ptr != NULL)
        {
            cout << ptr->data << " ";
            ptr = ptr->next;
        }
    }
    cout << endl;
}
int main()
{
    int ch, val;
    cout << "1) Push in stack" << endl;
    cout << "2) Pop from stack" << endl;
    cout << "3) Display stack" << endl;
    cout << "4) Exit" << endl;
    do
    {
        cout << "Enter choice: " << endl;
        cin >> ch;
        switch (ch)
        {
        case 1:
        {
            cout << "Enter value to be pushed:" << endl;
            cin >> val;
            push(val);
            break;
        }
        case 2:
        {
            pop();
```

```
            break;
        }
        case 3:
        {
            display();
            break;
        }
        case 4:
        {
            cout << "Exit" << endl;
            break;
        }
        default:
        {
            cout << "Invalid Choice" << endl;
        }
        }
    } while (ch != 4);
    return 0;
}
```

rudraksh.karpe.cs@ghrcem.raisoni.net

Output:

```
PS R:\GHRCEM\DSA new lab> cd "r:\GHRCEM\DSA new lab\" ; if ($?) { g++ Stack_Linkedlist.cpp -o Stack_Linkedlist } ; if ($?) { .\Stack_Linkedlist }
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
3
Enter choice:
1
Enter value to be pushed:
7
Enter choice:
1
Enter value to be pushed:
9
Enter choice:
3
Stack elements are: 9 7 3
Enter choice:
2
The popped element is 9
Enter choice:
2
The popped element is 7
Enter choice:
3
Stack elements are: 3
Enter choice:
4
Exit
PS R:\GHRCEM\DSA new lab>
```

Postfix expression:

Program code:

```
#include <iostream>
#include <string.h>

using namespace std;

struct Stack
{
   int top;
   unsigned capacity;
```

rudraksh.karpe.cs@ghrcem.raisoni.net

```
    int *array;
};

// Stack Operations
struct Stack *createStack(unsigned capacity)
{
    struct Stack *stack = (struct Stack *)malloc(sizeof(struct Stack));

    if (!stack)
        return NULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (int *)malloc(stack->capacity * sizeof(int));

    if (!stack->array)
        return NULL;

    return stack;
}

int isEmpty(struct Stack *stack)
{
    return stack->top == -1;
}

char peek(struct Stack *stack)
{
    return stack->array[stack->top];
}
```

```
char pop(struct Stack *stack)
{
    if (!isEmpty(stack))
        return stack->array[stack->top--];
    return '$';
}

void push(struct Stack *stack, char op)
{
    stack->array[++stack->top] = op;
}

int evaluatePostfix(char *exp)
{

    struct Stack *stack = createStack(strlen(exp));
    int i;
    if (!stack)
        return -1;


    for (i = 0; exp[i]; ++i)
    {

        if (isdigit(exp[i]))
            push(stack, exp[i] - '0');


        else
        {
            int val1 = pop(stack);
```

rudraksh.karpe.cs@ghrcem.raisoni.net

```
            int val2 = pop(stack);
            switch (exp[i])
            {
            case '+':
               push(stack, val2 + val1);
               break;
            case '-':
               push(stack, val2 - val1);
               break;
            case '*':
               push(stack, val2 * val1);
               break;
            case '/':
               push(stack, val2 / val1);
               break;
            }
         }
      }
      return pop(stack);
}

int main()
{
   char exp[] = "10 20 * 30 40 10 / - +";
   cout << "postfix evaluation: " << evaluatePostfix(exp);
   return 0;
}
```

Output:


PS R:\GHRCEM\DSA new lab> cd "r:\GHRCEM\DSA new lab\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRu
postfix evaluation: 72
PS R:\GHRCEM\DSA new lab>