

Folder SEM 3\Exp5

2 printable files

(file list disabled)

SEM 3\Exp5\Stack_ARR.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define MAX 100 // Maximum size of the stack
5
6  // Stack structure using arrays
7  struct StackArray
8  {
9      int top;
10     int arr[MAX];
11 };
12
13 // Function to create a stack
14 struct StackArray *createStack()
15 {
16     struct StackArray *stack = (struct StackArray *)malloc(sizeof(struct
StackArray));
17     stack->top = -1; // Initialize the top index
18     return stack;
19 }
20
21 // Check if the stack is full
22 int isFull(struct StackArray *stack)
23 {
24     return stack->top == MAX - 1;
25 }
26
27 // Check if the stack is empty
28 int isEmpty(struct StackArray *stack)
29 {
30     return stack->top == -1;
31 }
32
33 // Push an element onto the stack
34 void push(struct StackArray *stack, int value)
35 {
36     if (isFull(stack))
37     {
38         printf("Stack overflow!\n");
39         return;
40     }
41     stack->arr[++stack->top] = value;
42     printf("%d pushed onto stack\n", value);
43 }
44
```

```
45 // Pop an element from the stack
46 int pop(struct StackArray *stack)
47 {
48     if (isEmpty(stack))
49     {
50         printf("Stack underflow!\n");
51         return -1; // Return -1 for underflow
52     }
53     return stack->arr[stack->top--];
54 }
55
56 // Peek at the top element of the stack
57 int peek(struct StackArray *stack)
58 {
59     if (isEmpty(stack))
60     {
61         printf("Stack is empty!\n");
62         return -1; // Return -1 if empty
63     }
64     return stack->arr[stack->top];
65 }
66
67 // Display the stack
68 void display(struct StackArray *stack)
69 {
70     if (isEmpty(stack))
71     {
72         printf("Stack is empty!\n");
73         return;
74     }
75     printf("Stack elements: ");
76     for (int i = stack->top; i ≥ 0; i--)
77     {
78         printf("%d ", stack->arr[i]);
79     }
80     printf("\n");
81 }
82
83 int main()
84 {
85     struct StackArray *stack = createStack();
86     int choice, value;
87
88     while (1)
89     {
90         printf("\nStack Operations (Array Implementation):\n");
91         printf("1. Push\n");
92         printf("2. Pop\n");
93         printf("3. Peek\n");
94         printf("4. Display\n");
95         printf("5. Exit\n");
96         printf("Enter your choice: ");
97         scanf("%d", &choice);
98
```

```
99     switch (choice)
100     {
101     case 1:
102         printf("Enter the value to push: ");
103         scanf("%d", &value);
104         push(stack, value);
105         break;
106     case 2:
107         value = pop(stack);
108         if (value != -1)
109             printf("Popped value: %d\n", value);
110         break;
111     case 3:
112         value = peek(stack);
113         if (value != -1)
114             printf("Top value: %d\n", value);
115         break;
116     case 4:
117         display(stack);
118         break;
119     case 5:
120         free(stack);
121         exit(0);
122     default:
123         printf("Invalid choice!\n");
124     }
125 }
126
127 return 0;
128 }
129
```

SEM 3\Exp5\Stack_LL.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // Node structure for the linked list
5  struct Node
6  {
7      int data;
8      struct Node *next;
9  };
10
11 // Stack structure using linked lists
12 struct StackLinkedList
13 {
14     struct Node *top;
15 };
16
17 // Function to create a stack
18 struct StackLinkedList *createStack()
19 {
20     struct StackLinkedList *stack = (struct StackLinkedList *)malloc(sizeof(struct
StackLinkedList));
```

```
21     stack→top = NULL; // Initialize the top pointer
22     return stack;
23 }
24
25 // Check if the stack is empty
26 int isEmpty(struct StackLinkedList *stack)
27 {
28     return stack→top == NULL;
29 }
30
31 // Push an element onto the stack
32 void push(struct StackLinkedList *stack, int value)
33 {
34     struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
35     new_node→data = value;
36     new_node→next = stack→top;
37     stack→top = new_node;
38     printf("%d pushed onto stack\n", value);
39 }
40
41 // Pop an element from the stack
42 int pop(struct StackLinkedList *stack)
43 {
44     if (isEmpty(stack))
45     {
46         printf("Stack underflow!\n");
47         return -1; // Return -1 for underflow
48     }
49     struct Node *temp = stack→top;
50     int popped_value = temp→data;
51     stack→top = stack→top→next;
52     free(temp);
53     return popped_value;
54 }
55
56 // Peek at the top element of the stack
57 int peek(struct StackLinkedList *stack)
58 {
59     if (isEmpty(stack))
60     {
61         printf("Stack is empty!\n");
62         return -1; // Return -1 if empty
63     }
64     return stack→top→data;
65 }
66
67 // Display the stack
68 void display(struct StackLinkedList *stack)
69 {
70     if (isEmpty(stack))
71     {
72         printf("Stack is empty!\n");
73         return;
74     }
```

```
75     struct Node *temp = stack->top;
76     printf("Stack elements: ");
77     while (temp != NULL)
78     {
79         printf("%d ", temp->data);
80         temp = temp->next;
81     }
82     printf("\n");
83 }
84
85 int main()
86 {
87     struct StackLinkedList *stack = createStack();
88     int choice, value;
89
90     while (1)
91     {
92         printf("\nStack Operations (Linked List Implementation):\n");
93         printf("1. Push\n");
94         printf("2. Pop\n");
95         printf("3. Peek\n");
96         printf("4. Display\n");
97         printf("5. Exit\n");
98         printf("Enter your choice: ");
99         scanf("%d", &choice);
100
101         switch (choice)
102         {
103             case 1:
104                 printf("Enter the value to push: ");
105                 scanf("%d", &value);
106                 push(stack, value);
107                 break;
108             case 2:
109                 value = pop(stack);
110                 if (value != -1)
111                     printf("Popped value: %d\n", value);
112                 break;
113             case 3:
114                 value = peek(stack);
115                 if (value != -1)
116                     printf("Top value: %d\n", value);
117                 break;
118             case 4:
119                 display(stack);
120                 break;
121             case 5:
122                 // Free linked list nodes (cleanup)
123                 while (!isEmpty(stack))
124                 {
125                     pop(stack);
126                 }
127                 free(stack);
128                 exit(0);
```

```
129         default:
130             printf("Invalid choice!\n");
131         }
132     }
133
134     return 0;
135 }
136
```