

Experiment 4 Lab 1: Stack Implementation Using Arrays

main.c	Output
<pre>1 #include <stdio.h> 2 #include <stdlib.h> 3 4 #define MAX 5 // Define the maximum size of the stack 5 6 // Stack structure 7 struct Stack { 8 int arr[MAX]; 9 int top; 10 }; 11 12 // Function prototypes 13 void push(struct Stack *stack, int value); 14 int pop(struct Stack *stack); 15 int peek(struct Stack *stack); 16 int isEmpty(struct Stack *stack); 17 int isFull(struct Stack *stack); 18 19 int main() { 20 struct Stack stack; 21 stack.top = -1; // Initialize top to -1 indicating an empty stack 22 23 int choice, value; 24 25 while (1) { 26 // Menu 27 printf("\nStack Operations:\n"); 28 printf("1. Push\n"); 29 printf("2. Pop\n"); 30 printf("3. Peek\n"); 31 printf("4. Check if Stack is Empty\n"); 32 printf("5. Check if Stack is Full\n"); 33 printf("6. Exit\n"); 34 printf("Enter your choice: "); 35 scanf("%d", &choice); 36 37 switch (choice) { 38 case 1: 39 // Push operation 40 printf("Enter the value to push: "); 41 scanf("%d", &value); 42 push(&stack, value); 43 break; 44 45 case 2: 46 // Pop operation 47 value = pop(&stack); 48 if (value != -1) { 49 printf("Popped value: %d\n", value); 50 } 51 break;</pre>	<pre>/tmp/uboQvRS1B0.o Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Check if Stack is Full 6. Exit Enter your choice: 3 Stack is empty. Nothing to peek. Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Check if Stack is Full 6. Exit Enter your choice: 4 The stack is empty. Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Check if Stack is Full 6. Exit Enter your choice: </pre>
<pre>50 } 51 break; 52 53 case 3: 54 // Peek operation 55 value = peek(&stack); 56 if (value != -1) { 57 printf("Top value: %d\n", value); 58 } 59 break; 60 61 case 4: 62 // Check if stack is empty 63 if (isEmpty(&stack)) { 64 printf("The stack is empty.\n"); 65 } else { 66 printf("The stack is not empty.\n"); 67 } 68 break; 69 70 case 5: 71 // Check if stack is full 72 if (isFull(&stack)) { 73 printf("The stack is full.\n"); 74 } else { 75 printf("The stack is not full.\n"); 76 } 77 break; 78 79 case 6: 80 // Exit program 81 printf("Exiting...\n"); 82 exit(0); 83 84 default: 85 printf("Invalid choice! Please try again.\n"); 86 } 87 } 88 89 return 0; 90 } 91 92 // Function to push an element onto the stack 93 void push(struct Stack *stack, int value) { 94 if (isFull(stack)) { 95 printf("Stack Overflow! Cannot push %d.\n", value); 96 } else { 97 stack->top++; 98 stack->arr[stack->top] = value; 99 printf("Pushed %d onto the stack.\n", value); 100 }</pre>	<pre>/tmp/uboQvRS1B0.o Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Check if Stack is Full 6. Exit Enter your choice: 3 Stack is empty. Nothing to peek. Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Check if Stack is Full 6. Exit Enter your choice: 4 The stack is empty. Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Check if Stack is Full 6. Exit Enter your choice: </pre>

main.c

Share

Run

```
04         default:
05             printf("Invalid choice! Please try again.\n");
06         }
07     }
08 }
09 return 0;
10 }
11
12 // Function to push an element onto the stack
13 void push(struct Stack *stack, int value) {
14     if (isFull(stack)) {
15         printf("Stack Overflow! Cannot push %d.\n", value);
16     } else {
17         stack->top++;
18         stack->arr[stack->top] = value;
19         printf("Pushed %d onto the stack.\n", value);
20     }
21 }
22
23 // Function to pop an element from the stack
24 int pop(struct Stack *stack) {
25     if (isEmpty(stack)) {
26         printf("Stack Underflow! Cannot pop.\n");
27         return -1;
28     } else {
29         int value = stack->arr[stack->top];
30         stack->top--;
31         return value;
32     }
33 }
34
35 // Function to return the top element without removing it
36 int peek(struct Stack *stack) {
37     if (isEmpty(stack)) {
38         printf("Stack is empty. Nothing to peek.\n");
39         return -1;
40     } else {
41         return stack->arr[stack->top];
42     }
43 }
44
45 // Function to check if the stack is empty
46 int isEmpty(struct Stack *stack) {
47     return stack->top == -1;
48 }
49
50 // Function to check if the stack is full
51 int isFull(struct Stack *stack) {
52     return stack->top == MAX - 1;
53 }
```

/tmp/uboQvRS1BD.o

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Stack is Empty
5. Check if Stack is Full
6. Exit
Enter your choice: 3
Stack is empty. Nothing to peek.

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Stack is Empty
5. Check if Stack is Full
6. Exit
Enter your choice: 4
The stack is empty.

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Stack is Empty
5. Check if Stack is Full
6. Exit
Enter your choice:

Experiment 4 Lab 2: Stack Implementation Using Arrays

main.c	Output
<pre>1 #include <stdio.h> 2 #include <stdlib.h> 3 4 // Define a node structure for the linked list 5 struct Node { 6 int data; 7 struct Node* next; 8 }; 9 10 // Function prototypes 11 void push(struct Node** top, int value); 12 int pop(struct Node** top); 13 int peek(struct Node* top); 14 int isEmpty(struct Node* top); 15 16 int main() { 17 struct Node* stack = NULL; // Initialize stack as empty 18 19 int choice, value; 20 21 while (1) { 22 // Menu 23 printf("\nStack Operations:\n"); 24 printf("1. Push\n"); 25 printf("2. Pop\n"); 26 printf("3. Peek\n"); 27 printf("4. Check if Stack is Empty\n"); 28 printf("5. Exit\n"); 29 printf("Enter your choice: "); 30 scanf("%d", &choice); 31 32 switch (choice) { 33 case 1: 34 // Push operation 35 printf("Enter the value to push: "); 36 scanf("%d", &value); 37 push(&stack, value); 38 break; 39 40 case 2: 41 // Pop operation 42 value = pop(&stack); 43 if (value != -1) { 44 printf("Popped value: %d\n", value); 45 } 46 break; 47 48 case 3: 49 // Peek operation 50 value = peek(stack); 51 if (value != -1) {</pre>	<pre>/tmp/yTmJP3g60W.o Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Exit Enter your choice: 3 Stack is empty. Nothing to peek. Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Exit Enter your choice: 4 The stack is empty. Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Exit Enter your choice: </pre>
<pre>50 value = peek(stack); 51 if (value != -1) { 52 printf("Top value: %d\n", value); 53 } 54 break; 55 56 case 4: 57 // Check if stack is empty 58 if (isEmpty(stack)) { 59 printf("The stack is empty.\n"); 60 } else { 61 printf("The stack is not empty.\n"); 62 } 63 break; 64 65 case 5: 66 // Exit program 67 printf("Exiting...\n"); 68 exit(0); 69 70 default: 71 printf("Invalid choice! Please try again.\n"); 72 } 73 } 74 75 return 0; 76 } 77 78 // Function to push an element onto the stack 79 void push(struct Node** top, int value) { 80 struct Node* newNode = (struct Node*)malloc(sizeof(struct Node)); 81 if (!newNode) { 82 printf("Heap overflow. Unable to push %d.\n", value); 83 return; 84 } 85 newNode->data = value; 86 newNode->next = *top; 87 *top = newNode; 88 printf("Pushed %d onto the stack.\n", value); 89 } 90 91 // Function to pop an element from the stack 92 int pop(struct Node** top) { 93 if (isEmpty(*top)) { 94 printf("Stack Underflow! Cannot pop.\n"); 95 return -1; 96 } 97 struct Node* temp = *top; 98 int poppedValue = temp->data; 99 *top = (*top)->next;</pre>	<pre>/tmp/yTmJP3g60W.o Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Exit Enter your choice: 3 Stack is empty. Nothing to peek. Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Exit Enter your choice: 4 The stack is empty. Stack Operations: 1. Push 2. Pop 3. Peek 4. Check if Stack is Empty 5. Exit Enter your choice: </pre>

main.c

Share

Run

```
68         exit(0);
69
70         default:
71             printf("Invalid choice! Please try again.\n");
72     }
73 }
74
75 return 0;
76 }
77
78 // Function to push an element onto the stack
79 void push(struct Node** top, int value) {
80     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
81     if (!newNode) {
82         printf("Heap overflow. Unable to push %d.\n", value);
83         return;
84     }
85     newNode->data = value;
86     newNode->next = *top;
87     *top = newNode;
88     printf("Pushed %d onto the stack.\n", value);
89 }
90
91 // Function to pop an element from the stack
92 int pop(struct Node** top) {
93     if (isEmpty(*top)) {
94         printf("Stack Underflow! Cannot pop.\n");
95         return -1;
96     }
97     struct Node* temp = *top;
98     int poppedValue = temp->data;
99     *top = (*top)->next;
100     free(temp);
101     return poppedValue;
102 }
103
104 // Function to return the top element without removing it
105 int peek(struct Node* top) {
106     if (isEmpty(top)) {
107         printf("Stack is empty. Nothing to peek.\n");
108         return -1;
109     } else {
110         return top->data;
111     }
112 }
113
114 // Function to check if the stack is empty
115 int isEmpty(struct Node* top) {
116     return top == NULL;
117 }
```

/tmp/yTmJP3g60M.o

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Stack is Empty
5. Exit
Enter your choice: 3
Stack is empty. Nothing to peek.

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Stack is Empty
5. Exit
Enter your choice: 4
The stack is empty.

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Stack is Empty
5. Exit
Enter your choice: