

# LAB TASK



LAB TASK NO 6

SUBJECT: DATA STRUCTURE AND ALGORITHM

SUBMITTED TO: HIJAB DURRANI

SUBMITTED BY: MISBAH ULLAH JAN

CLASS CODE: BSSE-2024A

SEMESTER: 3RD

DATE: 11/27/2025

DEPARTMENT OF: SOFTWARE ENGINEERING



**CECOS UNIVERSITY HAYATABAD PESHAWAR**

## LAB TASK 1:

### 1. Modify your stack program so that:

1. Before inserting (enqueue), the program checks if the value already exists.
2. If it exists, do not insert and print:

**Duplicate value! Insertion not allowed.**

**Otherwise insert normally.**

### Algorithm + Modified C++ Program

Algorithm: Push with Duplicate Check

1. Start
2. Input value to push
3. IF stack is full
  - Print "Stack Overflow! Cannot push."
  - Stop
4. Traverse the stack from 0 to top
  - IF any element equals the new value
    - Print "Duplicate value! Insertion not allowed."
    - Stop
5. Increment top
6. Insert value at stackArr[top]
7. Print "{value} pushed successfully"
8. End

CODE:

```
main.cpp
1 #include <iostream>
2 using namespace std;
3
4 #define SIZE 5
5 int stackArr[SIZE];
6 int top = -1;
7
8 // Check empty
9 bool isEmpty() {
10     return top == -1;
11 }
12
13 // Check full
14 bool isFull() {
15     return top == SIZE - 1;
16 }
17
18 // Check duplicate
19 bool isDuplicate(int value) {
20     for (int i = 0; i <= top; i++) {
21         if (stackArr[i] == value)
22             return true;
23     }
24     return false;
25 }
26
27 // Push operation with duplicate check
28 void push(int value) {
```

```
main.cpp
26
27 // Push operation with duplicate check
28 void push(int value) {
29     if (isFull()) {
30         cout << "Stack Overflow! Cannot push.\n";
31         return;
32     }
33
34     if (isDuplicate(value)) {
35         cout << "Duplicate value! Insertion not allowed.\n";
36         return;
37     }
38
39     top++;
40     stackArr[top] = value;
41     cout << value << " pushed into stack.\n";
42 }
43
44 // Pop operation
45 void pop() {
46     if (isEmpty()) {
47         cout << "Stack Underflow! Nothing to pop.\n";
48         return;
49     }
50     cout << stackArr[top] << " popped from stack.\n";
51     top--;
52 }
53
```

```
main.cpp
54 // Peek operation
55 void peek() {
56     if (isEmpty()) {
57         cout << "Stack is empty.\n";
58     } else {
59         cout << "Top element: " << stackArr[top] << endl;
60     }
61 }
62
63 // Display elements
64 void display() {
65     if (isEmpty()) {
66         cout << "Stack is empty.\n";
67         return;
68     }
69     cout << "Stack elements: ";
70     for (int i = top; i >= 0; i--) {
71         cout << stackArr[i] << " ";
72     }
73     cout << endl;
74 }
75
76 int main() {
77     int choice, value;
78
79     while (true) {
80         cout << "\n=== Stack Menu (Duplicate Check Added) ===\n";
81         cout << "1. Push\n";
```

```
80         cout << "\n=== Stack Menu (Duplicate Check Added) ===\n";
81         cout << "1. Push\n";
82         cout << "2. Pop\n";
83         cout << "3. Peek\n";
84         cout << "4. Display\n";
85         cout << "5. Exit\n";
86         cout << "Enter choice: ";
87         cin >> choice;
88
89         switch (choice) {
90             case 1:
91                 cout << "Enter value to push: ";
92                 cin >> value;
93                 push(value);
94                 break;
95
96             case 2:
97                 pop();
98                 break;
99
100             case 3:
101                 peek();
102                 break;
103
104             case 4:
105                 display();
106                 break;
107
```

```

108         case 5:
109             cout << "Exiting...\n";
110             return 0;
111
112         default:
113             cout << "Invalid choice! Try again.\n";
114     }
115 }
116 }
117

```

CODE OUTPUT:

```

Output
Clear

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 20
20 pushed into stack.

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 30
30 pushed into stack.

```

Output

Clear

```
=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 20
Duplicate value! Insertion not allowed.

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 4
Stack elements: 30 20

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 2
30 popped from stack.
```

Output

Clear

```
=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 20
Duplicate value! Insertion not allowed.

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 40
40 pushed into stack.
```

```

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 4
Stack elements: 40 20

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice:
=== Session Ended. Please Run the code again ===

```

## LAB TASK 2:

2. Write a program using a stack (array) that adds an additional function:

**countElements()**

This function returns the number of elements currently in the queue.

**Example:**

If queue contains: [10, 20, 30]

**countElements() → 3**

### Algorithm for countElements()

Algorithm: countElements()

1. Start
2. If stack is empty (top == -1):  
→ Return 0
3. Otherwise, number of elements = top + 1
4. Return top + 1
5. End

CODE :

```
main.cpp
1 #include <iostream>
2 using namespace std;
3
4 #define SIZE 5
5 int stackArr[SIZE];
6 int top = -1;
7
8 // Check empty
9 bool isEmpty() {
10     return top == -1;
11 }
12
13 // Check full
14 bool isFull() {
15     return top == SIZE - 1;
16 }
17
18 // Push
19 void push(int value) {
20     if (isFull()) {
21         cout << "Stack Overflow! Cannot push.\n";
22         return;
23     }
24     top++;
25     stackArr[top] = value;
26     cout << value << " pushed into stack.\n";
27 }
28
```

```
main.cpp
28
29 // Pop
30 void pop() {
31     if (isEmpty()) {
32         cout << "Stack Underflow! Nothing to pop.\n";
33         return;
34     }
35     cout << stackArr[top] << " popped from stack.\n";
36     top--;
37 }
38
39 // Peek
40 void peek() {
41     if (isEmpty()) {
42         cout << "Stack is empty.\n";
43     } else {
44         cout << "Top element: " << stackArr[top] << endl;
45     }
46 }
47
48 // Display stack
49 void display() {
50     if (isEmpty()) {
51         cout << "Stack is empty.\n";
52         return;
53     }
54
55     cout << "Stack elements: ";
```



```
main.cpp
56 ~   for (int i = top; i >= 0; i--) {
57       cout << stackArr[i] << " ";
58   }
59   cout << endl;
60 }
61
62 // New function: Count elements in stack
63 ~ int countElements() {
64     return top + 1; // because top starts at -1
65 }
66
67 ~ int main() {
68     int choice, value;
69
70 ~   while (true) {
71       cout << "\n=== Stack Menu (With countElements) ===\n";
72       cout << "1. Push\n";
73       cout << "2. Pop\n";
74       cout << "3. Peek\n";
75       cout << "4. Display\n";
76       cout << "5. Count Elements\n";
77       cout << "6. Exit\n";
78       cout << "Enter choice: ";
79       cin >> choice;
80
81 ~   switch (choice) {
82       case 1:
83           cout << "Enter value to push: ";
```

```
83           cout << "Enter value to push: ";
84           cin >> value;
85           push(value);
86           break;
87
88       case 2:
89           pop();
90           break;
91
92       case 3:
93           peek();
94           break;
95
96       case 4:
97           display();
98           break;
99
100      case 5:
101          cout << "Total elements in stack = " << countElements() << endl;
102          break;
103
104      case 6:
105          cout << "Exiting...\n";
106          return 0;
107
108      default:
109          cout << "Invalid choice! Try again.\n";
110      }
```

CODE OUTPUT:

Output

Clear

```
=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 1
Enter value to push: 10
10 pushed into stack.

=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 1
Enter value to push: 20
20 pushed into stack.
```

Output

Clear

```
=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 1
Enter value to push: 30
30 pushed into stack.

=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 4
Stack elements: 30 20 10
```

```
Output Clear
=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 2
30 popped from stack.

=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 4
Stack elements: 20 10
```

```
=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 5
Total elements in stack = 2

=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice:
```

### LAB TASK 3:

**3. Write a program to:**

- 1. Insert values into a queue using an array**
- 2. Reverse the stack without using any library functions**
- 3. Display the reversed stack**

**Example:**

**Input: 10 20 30 40**

**Output: 40 30 20 10**

Algorithm:

Step 1 — Insert values into queue (array-based queue)

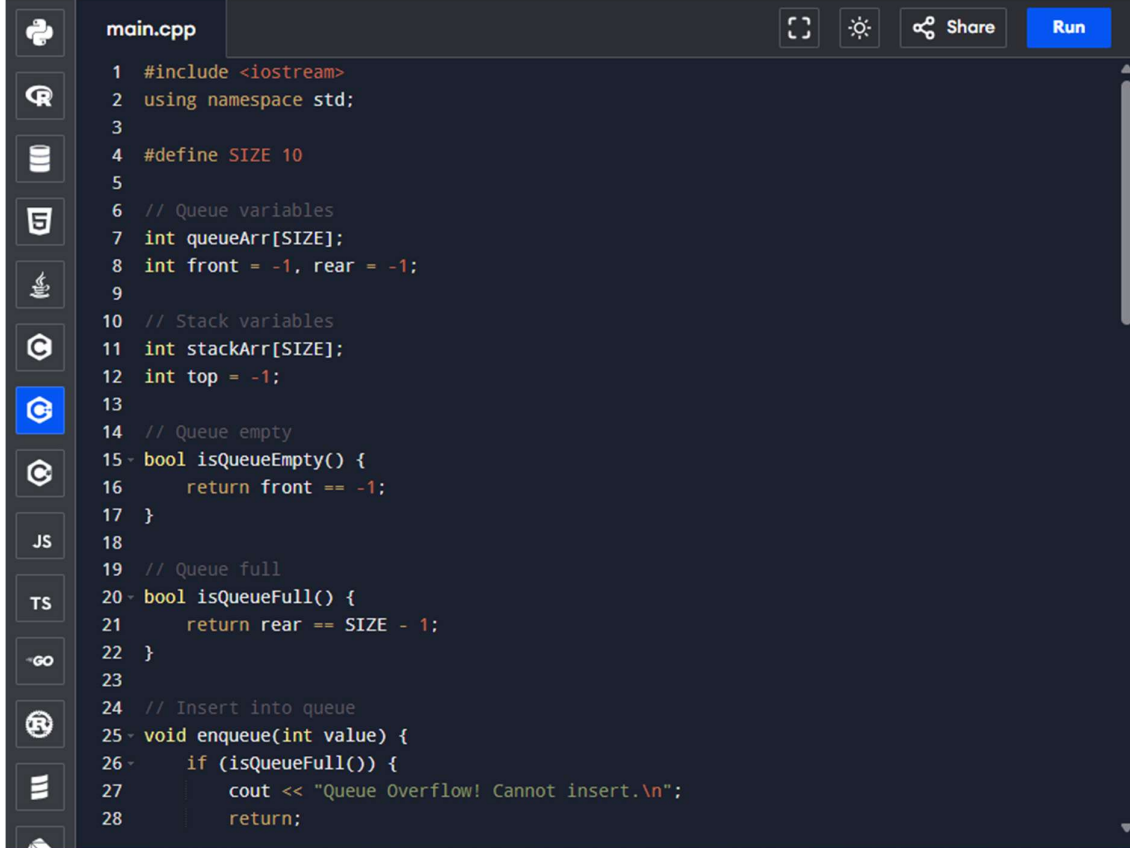
1. Start
2. For each input value:
  - If queue is full → print overflow
  - Else insert at rear and increment rear

Step 2 — Reverse using a stack (array)

1. Create an empty stack
2. While queue is not empty:
  - Pop each queue element and push it onto the stack
3. Now pop elements from stack to get reversed order

Step 3 — Display reversed stack

1. Print stack elements from top down to 0

The image shows a web-based C++ compiler interface. On the left is a vertical sidebar with icons for various programming languages: Python, R, Java, C#, JavaScript, TypeScript, Go, C++, PHP, Ruby, and Swift. The C++ icon is highlighted in blue. The main area is a code editor with a dark background, showing a C++ program named 'main.cpp'. The code implements a simple queue using an array. It includes headers, defines a size of 10, and initializes front and rear pointers to -1. It contains functions for checking if the queue is empty or full, and an enqueue function that prints an overflow message if the queue is full. At the top right of the editor are buttons for 'Share' and 'Run'.

```
1 #include <iostream>
2 using namespace std;
3
4 #define SIZE 10
5
6 // Queue variables
7 int queueArr[SIZE];
8 int front = -1, rear = -1;
9
10 // Stack variables
11 int stackArr[SIZE];
12 int top = -1;
13
14 // Queue empty
15 bool isEmpty() {
16     return front == -1;
17 }
18
19 // Queue full
20 bool isQueueFull() {
21     return rear == SIZE - 1;
22 }
23
24 // Insert into queue
25 void enqueue(int value) {
26     if (isQueueFull()) {
27         cout << "Queue Overflow! Cannot insert.\n";
28     }
29     return;
30 }
```

```
main.cpp [Full Screen] [Settings] [Share] [Run]

29     }
30
31     if (front == -1) front = 0; // first element
32
33     rear++;
34     queueArr[rear] = value;
35 }
36
37 // Stack push
38 void push(int value) {
39     if (top == SIZE - 1) {
40         cout << "Stack Overflow!\n";
41         return;
42     }
43     top++;
44     stackArr[top] = value;
45 }
46
47 // Stack pop
48 int pop() {
49     if (top == -1) {
50         cout << "Stack Underflow!\n";
51         return -1;
52     }
53     int value = stackArr[top];
54     top--;
55     return value;
56 }
```

```
main.cpp [Full Screen] [Settings] [Share] [Run]

57
58 int main() {
59     int n, value;
60
61     // Input
62     cout << "Enter number of elements: ";
63     cin >> n;
64
65     cout << "Enter " << n << " values:\n";
66     for (int i = 0; i < n; i++) {
67         cin >> value;
68         enqueue(value);
69     }
70
71     // Step 2: Move queue → stack (to reverse)
72     for (int i = front; i <= rear; i++) {
73         push(queueArr[i]);
74     }
75
76     // Step 3: Display reversed stack
77     cout << "Reversed stack: ";
78     while (top != -1) {
79         cout << pop() << " ";
80     }
81     cout << endl;
82
83     return 0;
84 }
```

## CODE OUTPUT:

```
Output Clear
Enter number of elements: 4
Enter 4 values:
10 20 30 40
Reversed stack: 40 30 20 10

=== Code Execution Successful ===
```

## LAB TASK 4:















4. Write a program to sort the elements of a stack.

### Algorithm:





#### SortStack Algorithm (Using Temporary Stack)

1. Create an empty temporary stack tempStack.
2. While original stack is NOT empty:
  - o Pop the top element → call it temp
  - o While tempStack is NOT empty AND tempStack.top() > temp  
→ Pop from tempStack and push back to original stack
  - o Push temp into tempStack
3. Copy all elements from tempStack back to original stack
4. Now the stack is sorted

### CODE:



main.cpp

 Share 

```
1 #include <iostream>
2 using namespace std;
3
4 #define SIZE 20
5 int stackArr[SIZE];
6 int top = -1;
7
8 // Push
9 void push(int value) {
10     if (top == SIZE - 1) {
11         cout << "Stack Overflow!\n";
12         return;
13     }
14     stackArr[++top] = value;
15 }
16
17 // Pop
18 int pop() {
19     if (top == -1) {
20         cout << "Stack Underflow!\n";
21         return -1;
22     }
23     return stackArr[top--];
24 }
25
26 // Peek
27 int peek() {
28     return stackArr[top];
```



```
main.cpp
29 }
30
31 // Check if empty
32 bool isEmpty() {
33     return top == -1;
34 }
35
36 // Sort stack using a temporary stack
37 void sortStack() {
38     int tempStack[SIZE];
39     int tempTop = -1;
40
41     while (!isEmpty()) {
42         int temp = pop();
43
44         while (tempTop != -1 && tempStack[tempTop] > temp) {
45             push(tempStack[tempTop--]);
46         }
47
48         tempStack[++tempTop] = temp;
49     }
50
51     // Move sorted values back to original stack
52     while (tempTop != -1) {
53         push(tempStack[tempTop--]);
54     }
55 }
56
```

```
main.cpp
57 // Display stack
58 void display() {
59     if (top == -1) {
60         cout << "Stack is empty.\n";
61         return;
62     }
63
64     cout << "Stack: ";
65     for (int i = top; i >= 0; i--) {
66         cout << stackArr[i] << " ";
67     }
68     cout << endl;
69 }
70
71 int main() {
72     int n, value;
73
74     cout << "Enter number of elements: ";
75     cin >> n;
76
77     cout << "Enter " << n << " values:\n";
78     for (int i = 0; i < n; i++) {
79         cin >> value;
80         push(value);
81     }
82
83     cout << "\nOriginal stack: ";
84     display();
85
```

```
85  
86     sortStack();  
87  
88     cout << "Sorted stack: ";  
89     display();  
90  
91     return 0;  
92 }  
93
```

#### CODE OUTPUT:

```
Output  
Enter number of elements: 8  
Enter 8 values:  
40 10 50 20 5 60 30 15  
  
Original stack: Stack: 15 30 60 5 20 50 10 40  
Sorted stack: Stack: 5 10 15 20 30 40 50 60  
  
=== Code Execution Successful ===
```

**LAB TASK 5: 5. Write a program to find the mean, variance and standard deviation of all elements of a stack**

#### Algorithm

Algorithm: Mean / Variance / Standard Deviation

1. Start
2. Insert values into stack
3. Compute sum of all stack elements
4. Compute mean:

$$\text{mean} = \frac{\text{sum}}{\text{number of elements}}$$

5. For variance:

- For each value  $x$ , compute

$$(x - \text{mean})^2$$

- Sum all of these

- Divide by total count
- 6. Standard deviation =  $\sqrt{\text{variance}}$
- 7. Display mean, variance, standard deviation
- 8. End

**CODE:**



The image shows a screenshot of a C++ code editor with a dark theme. The editor has a sidebar on the left with icons for Python, R, Java, C++, JavaScript, TypeScript, and others. The main area displays a file named 'main.cpp' with the following code:

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 #define SIZE 50
6 int stackArr[SIZE];
7 int top = -1;
8
9 // Push into stack
10 void push(int value) {
11     if (top == SIZE - 1) {
12         cout << "Stack Overflow!\n";
13         return;
14     }
15     stackArr[++top] = value;
16 }
17
18 // Display stack
19 void display() {
20     if (top == -1) {
21         cout << "Stack is empty.\n";
22         return;
23     }
24     cout << "Stack elements: ";
25     for (int i = top; i >= 0; i--) {
26         cout << stackArr[i] << " ";
27     }
28 }
```

The code implements a stack with a fixed size of 50. It includes functions for pushing elements and displaying the stack. The stack is represented by an array 'stackArr' and a pointer 'top' to the current top element. The code is written in C++ and uses standard library headers and namespace.

```
main.cpp
27     cout << stackArr[i] << " ";
28 }
29 cout << endl;
30 }
31
32 // Function to calculate mean
33 double findMean() {
34     if (top == -1) return 0;
35
36     double sum = 0;
37     for (int i = 0; i <= top; i++)
38         sum += stackArr[i];
39
40     return sum / (top + 1);
41 }
42
43 // Function to calculate variance
44 double findVariance(double mean) {
45     if (top == -1) return 0;
46
47     double sumSquares = 0;
48
49     for (int i = 0; i <= top; i++) {
50         sumSquares += pow(stackArr[i] - mean, 2);
51     }
52
53     return sumSquares / (top + 1);
54 }
55
```

```
main.cpp
55
56 // Function to calculate standard deviation
57 double findStdDev(double variance) {
58     return sqrt(variance);
59 }
60
61 int main() {
62     int n, value;
63
64     cout << "Enter number of elements: ";
65     cin >> n;
66
67     cout << "Enter " << n << " values:\n";
68     for (int i = 0; i < n; i++) {
69         cin >> value;
70         push(value);
71     }
72
73     display();
74
75     double mean = findMean();
76     double variance = findVariance(mean);
77     double stdDev = findStdDev(variance);
78
79     cout << "\nMean = " << mean << endl;
80     cout << "Variance = " << variance << endl;
81     cout << "Standard Deviation = " << stdDev << endl;
82
```

```
-GO 79 cout << "\nMean = " << mean << endl;
80 cout << "Variance = " << variance << endl;
81 cout << "Standard Deviation = " << stdDev << endl;
82
83 return 0;
84 }
85
```

#### CODE OUTPUT:

```
Output
Enter number of elements: 5
Enter 5 values:
10 20 30 40 50
Stack elements: 50 40 30 20 10

Mean = 30
Variance = 200
Standard Deviation = 14.1421

=== Code Execution Successful ===
```