

## Activity No. 4.1

### Hands-on Activity 4.1 Stacks

<b>Course Code:</b> CPE010	<b>Program:</b> Computer Engineering
<b>Course Title:</b> Data Structures and Algorithms	<b>Date Performed:</b> 8/24/2025
<b>Section:</b> CPE21S4	<b>Date Submitted:</b> 8/25/2025
<b>Name(s):</b> Avila, Vince Gabriel V.	<b>Instructor:</b> Engr. Jimlord Quejado

#### 6. Output

The screenshot shows a code editor interface with a dark theme. On the left, the code file 'main.cpp' is displayed with line numbers from 1 to 25. The code implements a stack using the standard library's `<stack>` container. It pushes three integers (3, 8, 15) onto the stack, then prints the stack's state: it's not empty, its size is 3, its top item is 15, and after popping, its size is 2. The output is displayed on the right side of the interface.

```
1 #include <iostream>
2 #include <stack>
3
4 using namespace std;
5
6 int main() {
7     stack<int> myStack;
8
9     myStack.push(3);
10    myStack.push(8);
11    myStack.push(15);
12
13    cout << "Is the Stack Empty? " << myStack.empty() << endl;
14
15    cout << "Current Stack Size: " << myStack.size() << endl;
16
17    cout << "Top Item on the Stack: " << myStack.top() << endl;
18
19    myStack.pop();
20
21    cout << "Top Item after Popping: " << myStack.top() << endl;
22    cout << "Updated Stack Size: " << myStack.size() << endl;
23
24    return 0;
25 }
```

Output:

```
Is the Stack Empty? 0
Current Stack Size: 3
Top Item on the Stack: 15
Top Item after Popping: 8
Updated Stack Size: 2
== Code Execution Successful ==
```

**Table 4.1**

```
main.cpp
1 #include <iostream>
2 using namespace std;
3
4 class Stack {
5 private:
6     int top;
7     int capacity;
8     int* arr;
9
10 public:
11     Stack(int size) {
12         capacity = size;
13         arr = new int[capacity];
14         top = -1;
15     }
16
17     ~Stack() {
18         delete[] arr;
19     }
20
21     void push(int value) {
22         if (top == capacity - 1) {
23             cout << "Stack Overflow!" << endl;
24         } else {
25             arr[++top] = value;
26         }
27     }
28
29     void pop() {
30         if (top == -1) {
31             cout << "Stack Underflow!" << endl;
32         } else {
33             top--;
34         }
35     }
36
37     void getTop() {
38         if (top == -1) {
39             cout << "The stack is empty!" << endl;
40         } else {
41             cout << "The element on the top of the stack is = " << arr[top] << endl;
42         }
43     }
44
45     void isEmpty() {
46         cout << ((top == -1) ? "Stack is EMPTY!" : "Stack is NOT EMPTY!") << endl;
```

Run

Output

```
Enter the size of the stack: 2
Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 1
Enter value to push: 10
Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 2
Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 3
Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 3
The stack is empty!
Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 4
Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 5
Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 0
*** Session Ended. Please Run the code again ***
```

```
45+     void isEmpty() {
46+         cout << ((top == -1) ? "Stack is EMPTY!" : "Stack is NOT EMPTY!") << endl;
47+     }
48+
49+     void display() {
50+         if (top == -1) {
51+             cout << "Stack is empty!" << endl;
52+             return;
53+         }
54+
55+         cout << "Stack elements (top to bottom):" << endl;
56+         for (int i = top; i >= 0; --i) {
57+             cout << arr[i] << endl;
58+         }
59+     }
60+ };
61+
62+ int main() {
63+     int size;
64+     cout << "Enter the size of the stack: ";
65+     cin >> size;
66+
67+     Stack myStack(size);
68+
69+     int choice, value;
70+     do {
71+         cout << "\nStack Operations Menu:\n";
72+         cout << "1. PUSH\n";
73+         cout << "2. POP\n";
74+         cout << "3. GET TOP\n";
75+         cout << "4. IS EMPTY\n";
76+         cout << "5. DISPLAY STACK\n";
77+         cout << "0. EXIT\n";
78+         cout << "Enter your choice: ";
79+         cin >> choice;
80+
81+         switch (choice) {
82+             case 1:
83+                 cout << "Enter value to push: ";
84+                 cin >> value;
85+                 myStack.push(value);
86+                 break;
87+             case 2:
88+                 myStack.pop();
89+                 break;
```

```

89         break;
90     case 3:
91         myStack.getTop();
92         break;
93     case 4:
94         myStack.isEmpty();
95         break;
96     case 5:
97         myStack.display();
98         break;
99     case 0:
100        cout << "Exiting program." << endl;
101        break;
102    default:
103        cout << "Invalid choice! Try again." << endl;
104    }
105 } while (choice != 0);
106
107 return 0;
108 }
109

```

**Table 4.2**

main.cpp	Run	Output
<pre> 1 #include &lt;iostream&gt; 2 using namespace std; 3 4 class StackNode { 5 public: 6     int value; 7     StackNode* next; 8 }; 9 10 StackNode* topNode = nullptr; 11 12 // Push a new element onto the stack 13 void push(int data) { 14     StackNode* newElement = new StackNode; 15     newElement-&gt;value = data; 16     newElement-&gt;next = topNode; 17     topNode = newElement; 18 } 19 20 // Pop the top element off the stack 21 int pop() { 22     if (topNode == nullptr) { 23         cout &lt;&lt; "Stack Underflow." &lt;&lt; endl; 24         return -1; 25     } else { 26         StackNode* temp = topNode; 27         int val = temp-&gt;value; 28         topNode = topNode-&gt;next; 29         delete temp; 30         return val; 31     } 32 } 33 34 // View the top element of the stack 35 void showTop() { 36     if (topNode == nullptr) { 37         cout &lt;&lt; "Stack is Empty." &lt;&lt; endl; 38     } else { 39         cout &lt;&lt; "Top of Stack: " &lt;&lt; topNode-&gt;value &lt;&lt; endl; 40     } 41 } 42 43 // Display all elements from top to bottom 44 void displayStack() { 45     cout &lt;&lt; "Stack elements (top to bottom):" &lt;&lt; endl; 46     StackNode* current = topNode; </pre>		<pre> After the first PUSH, top of stack is: Top of Stack: 1 After the second PUSH, top of stack is: Top of Stack: 5 After the third PUSH, top of stack is: Top of Stack: 10 Stack after pushing 3 elements: Stack elements (top to bottom): 10 5 1  Popping one element... Top after POP: Top of Stack: 5  Final Stack Content: Stack elements (top to bottom): 5 1  *** Code Execution Successful *** </pre>

```
47  while (current != nullptr) {
48      cout << current->value << " ";
49      current = current->next;
50  }
51  cout << endl;
52 }
53
54 // Main function
55 int main() {
56     push(1);
57     cout << "After the first PUSH, top of stack is: ";
58     showTop();
59
60     push(5);
61     cout << "After the second PUSH, top of stack is: ";
62     showTop();
63
64     push(10);
65     cout << "After the third PUSH, top of stack is: ";
66     showTop();
67
68     cout << "Stack after pushing 3 elements: " << endl;
69     displayStack();
70
71     cout << "\nPopping one element..." << endl;
72     pop();
73     cout << "Top after POP: ";
74     showTop();
75
76     cout << "\nFinal Stack Content:" << endl;
77     displayStack();
78
79     return 0;
80 }
```

## 7. Supplementary Activity

**ILO C:** SOLVE PROBLEMS USING AN IMPLEMENTATION OF STACK:

Table 4.3

### a. Stack Using Arrays

File  
main.cpp (Ctrl+M)

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 #define LIMIT 100
6
7 class BracketStack {
8 private:
9     char data[LIMIT];
10    int topIndex;
11
12 public:
13    BracketStack() {
14        topIndex = -1;
15    }
16
17    bool isEmpty() {
18        return topIndex == -1;
19    }
20
21    bool isFull() {
22        return topIndex == LIMIT - 1;
23    }
24
25    void pushChar(char symbol) {
26        if (!isFull()) {
27            data[++topIndex] = symbol;
28        }
29    }
30
31    char popChar() {
32        if (isEmpty()) return '\0';
33    }
34}
```

Enter expression: (A+B)+(C-D)  
Balanced (Array)

-----  
...Program finished with exit code 0  
Press ENTER to exit console. █

```

32     if (isEmpty()) return '\0';
33     return data[topIndex--];
34   }
35
36   char peekChar() {
37     if (isEmpty()) return '\0';
38     return data[topIndex];
39   }
40 };
41
42 bool matchBrackets(char open, char close) {
43   return (open == '(' && close == ')') ||
44          (open == '{' && close == '}') ||
45          (open == '[' && close == ']');
46 }
47
48 bool isExpressionBalanced(const string& input) {
49   BracketStack stack;
50   for (char ch : input) {
51     if (ch == '(' || ch == '{' || ch == '[') {
52       stack.pushChar(ch);
53     } else if (ch == ')' || ch == '}' || ch == ']') {
54       if (stack.isEmpty()) return false;
55       char lastOpen = stack.popChar();
56       if (!matchBrackets(lastOpen, ch)) return false;
57     }
58   }
59   return stack.isEmpty();
60 }
61
62 int main() {
63   string expression;
64   cout << "Enter expression: ";
65   getline(cin, expression); // Full-line input with spaces
66
67   if (isExpressionBalanced(expression)) {
68     cout << "Balanced (Array)" << endl;
69   } else {
70     cout << "Not Balanced (Array)" << endl;
71   }
72
73   cout << "\n-----" << endl;
74
75   return 0;
76 }
77

```

## b. Stack Using Linked Lists

**main.cpp**

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct Node {
6     char data;
7     Node* next;
8 };
9
10 class StackLinkedList {
11 private:
12     Node* top;
13
14 public:
15     StackLinkedList() { top = nullptr; }
16
17     bool isEmpty() { return top == nullptr; }
18
19     void push(char ch) {
20         Node* newNode = new Node{ch, top};
21         top = newNode;
22     }
23
24     char pop() {
25         if (isEmpty()) return '\0';
26         char ch = top->data;
27         Node* temp = top;
28         top = top->next;
29         delete temp;
30         return ch;
31     }
32 }
```



Enter expression: ((A+B)+(C-D)  
Not Balanced (Linked List)

-----  
...Program finished with exit code 0  
Press ENTER to exit console.

```

30     return cu,
31 }
32
33 ~StackLinkedList() {
34     while (!isEmpty()) pop();
35 }
36 };
37
38 bool isMatchingPair(char open, char close) {
39     return (open == '(' && close == ')') ||
40            (open == '{' && close == '}') ||
41            (open == '[' && close == ']');
42 }
43
44 bool checkBalancedLinkedList(const string& expr) {
45     StackLinkedList stack;
46     for (char ch : expr) {
47         if (ch == '(' || ch == '{' || ch == '[') {
48             stack.push(ch);
49         } else if (ch == ')' || ch == '}' || ch == ']') {
50             if (stack.isEmpty()) return false;
51             char open = stack.pop();
52             if (!isMatchingPair(open, ch)) return false;
53         }
54     }
55     return stack.isEmpty();
56 }
57
58 int main() {
59     string expr;
60     cout << "Enter expression: ";
61     getline(cin, expr);

```

```

    if (checkBalancedLinkedList(expr)) {
        cout << "Balanced (Linked List)" << endl;
    } else {
        cout << "Not Balanced (Linked List)" << endl;
    }

```

```

    cout << "\n-----" << endl;

```

```

    return 0;
}

```

**Self-Checking: Expression: (A+B)+(C-D)**

```
Enter expression: (A+B)+(C-D)
Balanced (Array)

-----
...Program finished with exit code 0
Press ENTER to exit console.█
```

#### Self-Checking: Expression: ((A+B)+(C-D))

```
Enter expression: ((A+B)+(C-D))
Not Balanced (Linked List)

-----
...Program finished with exit code 0
Press ENTER to exit console.
```

#### Self-Checking: Expression: ((A+B)+(C-D))

```
Enter expression: ((A+B]+[C-D])
Not Balanced (Linked List)

-----
...Program finished with exit code 0
Press ENTER to exit console.
```

#### Self-Checking: Expression: ((A+B)+[C-D])

```
Enter expression: ((A+B)+[C-D])
Balanced (Linked List)

-----
...Program finished with exit code 0
Press ENTER to exit console.
```

#### Self-Checking: Expression: ((A+B]+[C-D])}

```
Enter expression: ((A+B]+[C-D])  
Not Balanced (Linked List)  
-----  
...Program finished with exit code 0  
Press ENTER to exit console.
```

## 8. Conclusion

In conclusion, I am still learning about how a stack works by following the Last In, First Out rule and how the different operations push, pop, top, isEmpty, and display are used. I am beginning to understand how data is stored and removed in the order it was put in. I still have to work on organizing and writing better code. The activities did help me better understand how stacks work in a program.

## 9. Assessment Rubric