

Activity No. <4.1>

<Hands-on Activity 4.1 Stacks>

Course Code: CPE010

Program: Computer Engineering

Course Title: Data Structures and Algorithms

Date Performed: 8/25/25

Section: CPE 010-CPE21S4

Date Submitted: 8/25/25

Name(s): Catungal Kerwin Jan B

Instructor: Jimlord Quejado

6. Output

ILO A: Create a stack using the C++ STL

```

1 #include <iostream>
2 #include <stack> // Calling Stack from the STL
3
4 using namespace std;
5
6 int main()
7 {
8     stack<int> newStack;
9
10    newStack.push(3); //Adds 3 to the stack
11    newStack.push(8);
12    newStack.push(15);
13
14    // returns a boolean response depending on if the stack is empty or not
15    cout << "Stack Empty? " << newStack.empty() << endl;
16
17    // returns the size of the stack itself
18    cout << "Stack Size: " << newStack.size() << endl;
19
20    // returns the topmost element of the stack
21    cout << "Top Element of the Stack: " << newStack.top() << endl;
22
23    // removes the topmost element of the stack
24    newStack.pop();
25
26    cout << "Top Element of the Stack: " << newStack.top() << endl;
27    cout << "Stack Size: " << newStack.size() << endl;
28
29    return 0;
30 }
    
```

```

C:\Users\Olaco\Downloads\ILOA1.exe
Stack Empty? 0
Stack Size: 3
Top Element of the Stack: 15
Top Element of the Stack: 8
Stack Size: 2

-----
Process exited after 0.3566 seconds with return value 0
Press any key to continue . . .
    
```

Table 4-2. Output of ILO B.1.

```

1 void getTop() {
2     if (top == -1) {
3         cout << "The stack is empty!" << endl;
4     }
5     else {
6         cout << "The element on the top of the stack is " << arr[top] << endl;
7     }
8 }
9
10 void isEmpty() {
11     cout << (top == -1 ? "Stack is EMPTY" : "Stack is NOT EMPTY") << endl;
12 }
13
14 void display() {
15     if (top == -1) {
16         cout << "Stack is empty!" << endl;
17         return;
18     }
19     cout << "Stack elements (top to bottom): ";
20     for (int i = top; i >= 0; --i) {
21         cout << arr[i] << " ";
22     }
23     cout << endl;
24     getTop();
25 }
26
27 int main() {
28     int size;
29     cout << "Enter number of max elements for new stack: ";
30     cin >> size;
31
32     Stack s(size);
33     int choice, value;
34
35     while (true) {
36         cout << "Stack Operations:\n";
37         cout << "1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY\n";
38         cin >> choice;
39
40         switch (choice) {
41             case 1:
42                 cout << "Enter value to push: ";
43                 cin >> value;
44                 s.push(value);
45                 break;
46             case 2:
47                 s.pop();
48                 break;
49             case 3:
50                 s.getTop();
51                 break;
52             case 4:
53                 s.isEmpty();
54                 break;
55             case 5:
56                 s.display();
57                 break;
58             default:
59                 break;
60         }
61     }
62 }
    
```

```

C:\Users\Olaco\Downloads\ILOB4.1.exe
Enter number of max elements for new stack: 20
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
1
New Value:
2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
4
Stack is NOT EMPTY
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
3
The element on the top of the stack is 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
5
Stack elements (top to bottom): 2
The element on the top of the stack is 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
    
```

Table 4-3. Output of ILO B.2.

```

29 }
30 }
31
32 void Top() {
33     if (head == nullptr) {
34         cout << "Stack is Empty." << endl;
35     } else {
36         cout << "Top of Stack: " << head->data << endl;
37     }
38 }
39
40 void display() {
41     cout << "Stack elements (top to bottom):" << endl;
42     Node *current = head;
43     while (current != nullptr) {
44         cout << current->data << " ";
45         current = current->next;
46     }
47     cout << endl;
48 }
49
50 int main() {
51     push(1);
52     cout << "After the first PUSH, top of stack is: ";
53     Top();
54
55     push(5);
56     cout << "After the second PUSH, top of stack is: ";
57     Top();
58
59     pop();
60     cout << "After the first POP operation, top of stack is: ";
61     Top();
62
63     pop();
64     cout << "After the second POP operation, top of stack is: ";
65     Top();
66 }
67
68

```

```

C:\Users\Olaco\Downloads\ILOB4.1.exe
Enter number of max elements for new stack: 20
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
1
New Value:
2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
4
Stack is NOT EMPTY
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
3
The element on the top of the stack is 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY
5
Stack elements (top to bottom): 2
The element on the top of the stack is 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY

```

7. Supplementary Activity

ILO C: SOLVE PROBLEMS USING AN IMPLEMENTATION OF STACK:

Table 4.3

a. Stack Using Arrays

```

0 }
1 };
2
3 bool isMatchingPair(char open, char close) {
4     return (open == '(' && close == ')') ||
5            (open == '{' && close == '}') ||
6            (open == '[' && close == ']');
7 }
8
9 bool checkBalancedArray(const string& expr) {
10     StackArray stack;
11     for (char ch : expr) {
12         if (ch == '(' || ch == '{' || ch == '[') {
13             stack.push(ch);
14         } else if (ch == ')' || ch == '}' || ch == ']') {
15             if (stack.isEmpty()) return false;
16             char open = stack.pop();
17             if (!isMatchingPair(open, ch)) return false;
18         }
19     }
20     return stack.isEmpty();
21 }
22
23 int main() {
24     string expr;
25     cout << "Enter expression: ";
26     getline(cin, expr); // Supports full-line input
27
28     if (checkBalancedArray(expr)) {
29         cout << "Balanced (Array)" << endl;
30     } else {
31         cout << "Not Balanced (Array)" << endl;
32     }
33
34     cout << "\n-----" << endl;
35
36     return 0;
37 }
38

```

```

C:\Users\Olaco\Downloads\Supplementary.exe
Enter expression: (A+B)+(C-D)
Balanced (Array)
-----
Process exited after 60.25 seconds with return value 0
Press any key to continue . . .

```

b. Stack using Linked Lists

The screenshot shows a C++ IDE with the following code in `Supplementaryb.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
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 }
```

The execution output for `Supplementaryb.exe` is as follows:

```
C:\Users\Olaco\Downloads\Supplementaryb.exe
Enter expression: (A+B)+(C-D)
Balanced (Linked List)

-----

Process exited after 127.1 seconds with return value 0
Press any key to continue . . .
```

Self-Checking:

Expression:

$(A+B)+(C-D)$

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

-----

-----

Process exited after 27.77 seconds with return value 0
Press any key to continue . . .
```

Expression:

$((A+B)+(C-D))$

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

-----

-----

Process exited after 29.23 seconds with return value 0
Press any key to continue . . .
```

((A+B)+[C-D])

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

```
-----
```

```
-----
```

```
Process exited after 4.137 seconds with return value 0
Press any key to continue . . .
```

((A+B)+[C-D])

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

```
-----
```

```
-----
```

```
Process exited after 3.992 seconds with return value 0
Press any key to continue . . .
```

8. Conclusion

In this activity we used push, pop, top, is empty, display. this process helped me learn how data is stored and viewed. It follows the rules of the Last In, First Out rule or LIFO. It took some logical thinking to make sure that each one was done correctly and fast. In conclusion, I learned a lot about the main ideas of stacks, but I still need to get better at putting my code in order.

9. Assessment Rubric