

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

The screenshot shows a code editor with two tabs: ILOA1.cpp and ILOB4.1.cpp. The ILOA1.cpp tab is active, displaying the following C++ code:

```

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 }

```

To the right of the code editor is a terminal window titled 'C:\Users\Olaco\Downloads\ILOA1.exe'. It displays the following output:

```

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.

The screenshot shows a code editor with two tabs: LOA1.cpp and ILOB4.1.cpp. The ILOB4.1.cpp tab is active, displaying the following C++ code:

```

void getTop() {
    if (top == -1) {
        cout << "The stack is empty!" << endl;
    } else {
        cout << "The element on the top of the stack is " << arr[top] << endl;
    }
}

void isEmpty() {
    cout << (top == -1 ? "Stack is EMPTY" : "Stack is NOT EMPTY") << endl;
}

void display() {
    if (top == -1) {
        cout << "Stack is empty!" << endl;
        return;
    }

    cout << "Stack elements (top to bottom): ";
    for (int i = top; i >= 0; --i) {
        cout << arr[i] << " ";
    }
    cout << endl;
    getTop();
}

int main() {
    int size;
    cout << "Enter number of max elements for new stack: ";
    cin >> size;

    Stack s(size);
    int choice, value;

    while (true) {
        cout << "Stack Operations:\n";
        cout << "1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. DISPLAY\n";
        cin >> choice;
    }
}

```

To the right of the code editor is a terminal window titled 'C:\Users\Olaco\Downloads\ILOB4.1.exe'. It displays the following output:

```

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.

The screenshot shows a Windows-based IDE interface. On the left, three tabs are visible: ILOA1.cpp, ILOB4.1.cpp, and ILOB4.2.cpp. The ILOB4.2.cpp tab is active, displaying C++ code for a stack implementation. The code includes functions for pushing elements onto the stack, popping them off, checking if the stack is empty, and displaying the top element. It also contains a main() function that demonstrates these operations. On the right, the execution window shows the command-line interface of the application. It prompts the user to enter the maximum number of elements for the new stack (20). It then lists the stack operations: PUSH, POP, TOP, isEmpty, and DISPLAY. The application performs several operations, including pushing values 1 and 5 onto the stack, popping them off, and displaying the top element (which is 2). The output concludes with a message indicating the stack is NOT EMPTY.

```

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 }

```

Compilation results...

- Errors: 0

7. Supplementary Activity

ILO C: SOLVE PROBLEMS USING AN IMPLEMENTATION OF STACK:

Table 4.3

a. Stack Using Arrays

The screenshot shows a Windows-based IDE interface. On the left, four tabs are visible: ILOA1.cpp, ILOB4.1.cpp, ILOB4.2.cpp, and Supplementary.cpp. The Supplementary.cpp tab is active, displaying C++ code for a stack implemented using an array. The code includes a helper function to check if a pair of parentheses is matching, and a main function that uses a stack array to validate the balance of characters in an input expression. The main function reads an expression from standard input and prints "Balanced (Array)" if the expression is balanced, or "Not Balanced (Array)" otherwise. On the right, the execution window shows the command-line interface of the application. It prompts the user to enter an expression: "(A+B)+(C-D)". The application processes the expression and outputs "Balanced (Array)". At the end, it displays a message indicating the process exited after 60.25 seconds with a return value of 0.

```

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 }

```

Compilation results...

Errors: 0
Warnings: 0

b. Stack using Linked Lists

```
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 public:
14     StackLinkedList() { top = nullptr; }
15     bool isEmpty() { return top == nullptr; }
16     void push(char ch) {
17         Node* newNode = new Node(ch, top);
18         top = newNode;
19     }
20     char pop() {
21         if (isEmpty()) return '\0';
22         char ch = top->data;
23         Node* temp = top;
24         top = top->next;
25         delete temp;
26         return ch;
27     }
28     ~StackLinkedList() {
29         while (!isEmpty()) pop();
30     }
31 };
32
33 bool isMatchingPair(char open, char close) {
34     return (open == '(' && close == ')') ||
35            (open == '[' && close == ']') ||
36            (open == '{' && close == '}');
37
38 bool isBalanced(string expression) {
39     StackLinkedList stack;
40     for (char ch : expression) {
41         if (ch == '(' || ch == '[' || ch == '{') {
42             stack.push(ch);
43         } else if (ch == ')' || ch == ']' || ch == '}') {
44             if (stack.isEmpty() || !isMatchingPair(stack.pop(), ch)) {
45                 return false;
46             }
47         }
48     }
49     return stack.isEmpty();
50 }
51
52 int main() {
53     string expression;
54     cout << "Enter expression: ";
55     cin >> expression;
56     if (isBalanced(expression)) {
57         cout << "Balanced (Linked List)" << endl;
58     } else {
59         cout << "Not Balanced (Linked List)" << endl;
60     }
61     system("PAUSE");
62     return 0;
63 }
```

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