

Activity No. 4	
STACKS	
Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: OCTOBER 4, 2024
Section: CPE21S1	Date Submitted: OCTOBER 7, 2024
Name(s): GASPAR, AARON ROWEN O.	Instructor: MS. MARIA RIZETTE SAYO

6. Output

ILO A:

VoidFunction Limited

Run

main.cpp x +

main.cpp > f main

1 #include <iostream>

2 #include <stack> // Calling Stack from the STL

3 using namespace std;

4 int main() {

5 stack<int> newStack;

6 newStack.push(3); //Adds 3 to the stack

7 newStack.push(8);

8 newStack.push(15);

9 // returns a boolean response depending on if the stack is empty or not

10 cout << "Stack Empty? " << newStack.empty() << endl;

11 // returns the size of the stack itself

12 cout << "Stack Size: " << newStack.size() << endl;

13 // returns the topmost element of the stack

14 cout << "Top Element of the Stack: " << newStack.top() << endl;

15 // removes the topmost element of the stack

16 newStack.pop();

17 cout << "Top Element of the Stack: " << newStack.top() << endl;

18 cout << "Stack Size: " << newStack.size() << endl;

19 return 0;

20 }

21

Console x Shell +

Run

Stack Empty? 0

Stack Size: 3

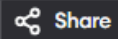
Top Element of the Stack: 15

Top Element of the Stack: 8

Stack Size: 2

ILO B.1:

main.cpp



Run

```
1 #include<iostream>
2 using namespace std;
3 const size_t maxCap= 100;
4 int stack[maxCap]; //stack with max of 100 elements
5 int top = -1, i, newData;
6 void push();
7 void pop();
8 void Top();
9 void ShowAll();
10 bool isEmpty();
11 int main(){
12     int choice;
13     cout << "Enter number of max elements for new stack: ";
14     cin >> i;
15     while(true){
16         cout << "Stack Operations: " << endl;
17         cout << "1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. ShowAll" << endl;
18         cin >> choice;
19         switch(choice){
20             case 1: push();
21             break;
22             case 2: pop();
23             break;
24             case 3: Top();
25             break;
26             case 4: cout << isEmpty() << endl;
27             break;
28             case 5: ShowAll();
29             break;
30             default: cout << "Invalid Choice." << endl;
31             break;
32         }
33     }
34     return 0;
35 }
```

```

36 ~ bool isEmpty(){
37   if(top== -1) return true;
38   return false;
39 }
40 ~ void push(){
41   //check if full -> if yes, return error
42 ~ if(top == i-1){
43   cout << "Stack Overflow." << endl;
44   return;
45 }
46 cout << "New Value: " << endl;
47 cin >> newData;
48 stack[++top] = newData;
49 }
50 ~ void pop(){
51   //check if empty -> if yes, return error
52 ~ if(isEmpty()){
53   cout << "Stack Underflow." << endl;
54   return;
55 }
56 //display the top value
57 cout << "Popping: " << stack[top] << endl;
58 //decrement top value from stack
59 top--;
60 }
61 ~ void Top(){
62 ~ if(isEmpty()) {
63   cout << "Stack is Empty." << endl;
64   return;
65 }
66 cout << "The element on the top of the stack is " << stack[top] <<
67 endl;
68 }

```

```

69 ~ void ShowAll() {
70 ~ if(isEmpty()) {
71   cout << "Stack is Empty." << endl;
72   return;
73 }
74 cout << "Elements in the Stack: ";
75 ~ for (int j = top; j >= 0; j--) {
76   cout << stack[j] << " ";
77 }
78 cout << endl;
79 }

```

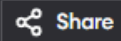
Output

[Clear](#)

```
/tmp/GEBb1wiYfP.o
Enter number of max elements for new stack: 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. ShowAll
1
New Value:
12
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. ShowAll
1
New Value:
13
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. ShowAll
5
Elements in the Stack: 13 12
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty, 5. ShowAll
|
```

ILO B.2:

main.cpp



Run

```
1  #include<iostream>
2  using namespace std;
3
4  class Node {
5  public:
6      int data;
7      Node *next;
8  };
9
10 Node *head = NULL, *tail = NULL;
11
12 void push(int newData) {
13     Node *newNode = new Node;
14     newNode->data = newData;
15     newNode->next = head;
16     if (head == NULL) {
17         head = tail = newNode;
18     } else {
19         head = newNode;
20     }
21 }
22
23 int pop() {
24     int tempVal;
25     Node *temp;
26     if (head == NULL) {
27         std::cout << "Stack Underflow." << std::endl;
28         return -1;
29     } else {
30         temp = head;
31         tempVal = temp->data;
32         head = head->next;
33         delete(temp);
34         return tempVal;
35     }
36 }
37
```

```
38 void Top() {
39     if (head == NULL) {
40         std::cout << "Stack is Empty." << std::endl;
41     } else {
42         std::cout << "Top of Stack: " << head->data << std::endl;
43     }
44 }
45
46 void ShowAll() {
47     if (head == NULL) {
48         cout << "Stack is Empty." << endl;
49         return;
50     }
51     cout << "Elements in the Stack: ";
52     Node *temp = head;
53     while (temp != NULL) {
54         cout << temp->data << " ";
55         temp = temp->next;
56     }
57     cout << endl;
58 }
59
60 int main() {
61     push(1);
62     std::cout << "After the first PUSH, top of stack is: ";
63     Top();
64     push(5);
65     std::cout << "After the second PUSH, top of stack is: ";
66     Top();
67     pop();
68     std::cout << "After the first POP operation, top of stack is: ";
69     Top();
70     pop();
71     std::cout << "After the second POP operation, top of stack is: ";
72     Top();
73     ShowAll();
74     return 0;
75 }
```

Output

[Clear](#)

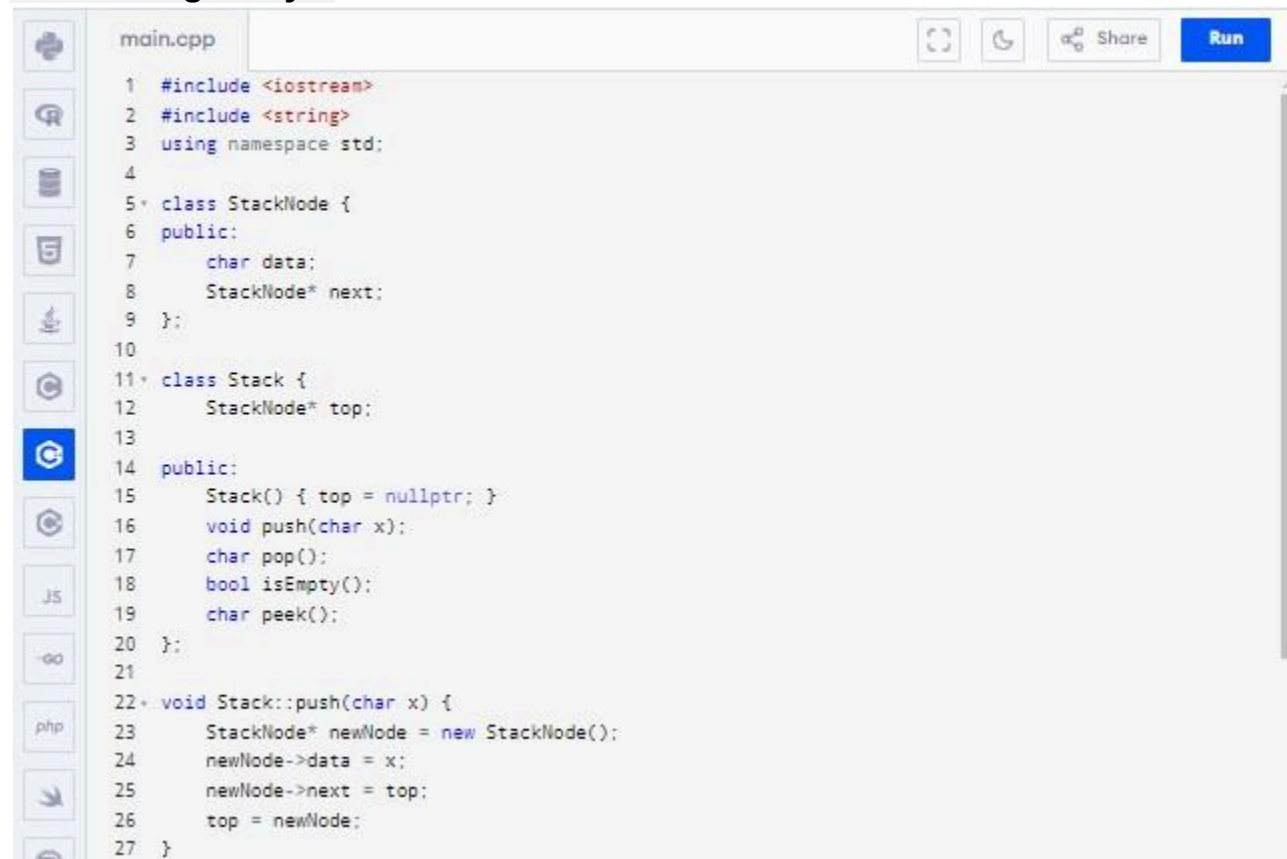
/tmp/LLKmxLLMwI.o

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 first POP operation, top of stack is: Top of Stack: 1
After the second POP operation, top of stack is: Stack is Empty.
Stack is Empty.

=== Code Execution Successful ===

7. Supplementary Activity

Stack using Arrays:



```
main.cpp
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class StackNode {
6  public:
7      char data;
8      StackNode* next;
9  };
10
11 class Stack {
12     StackNode* top;
13
14 public:
15     Stack() { top = nullptr; }
16     void push(char x);
17     char pop();
18     bool isEmpty();
19     char peek();
20 };
21
22 void Stack::push(char x) {
23     StackNode* newNode = new StackNode();
24     newNode->data = x;
25     newNode->next = top;
26     top = newNode;
27 }
```

```

28
29 char Stack::pop() {
30     if (top == nullptr) {
31         cout << "Stack Underflow";
32         return 0;
33     } else {
34         StackNode* temp = top;
35         top = top->next;
36         char popped = temp->data;
37         delete temp;
38         return popped;
39     }
40 }
41
42 bool Stack::isEmpty() {
43     return top == nullptr;
44 }
45
46 char Stack::peek() {
47     if (top == nullptr) {
48         cout << "Stack is Empty";
49         return 0;
50     } else {
51         return top->data;
52     }
53 }
54

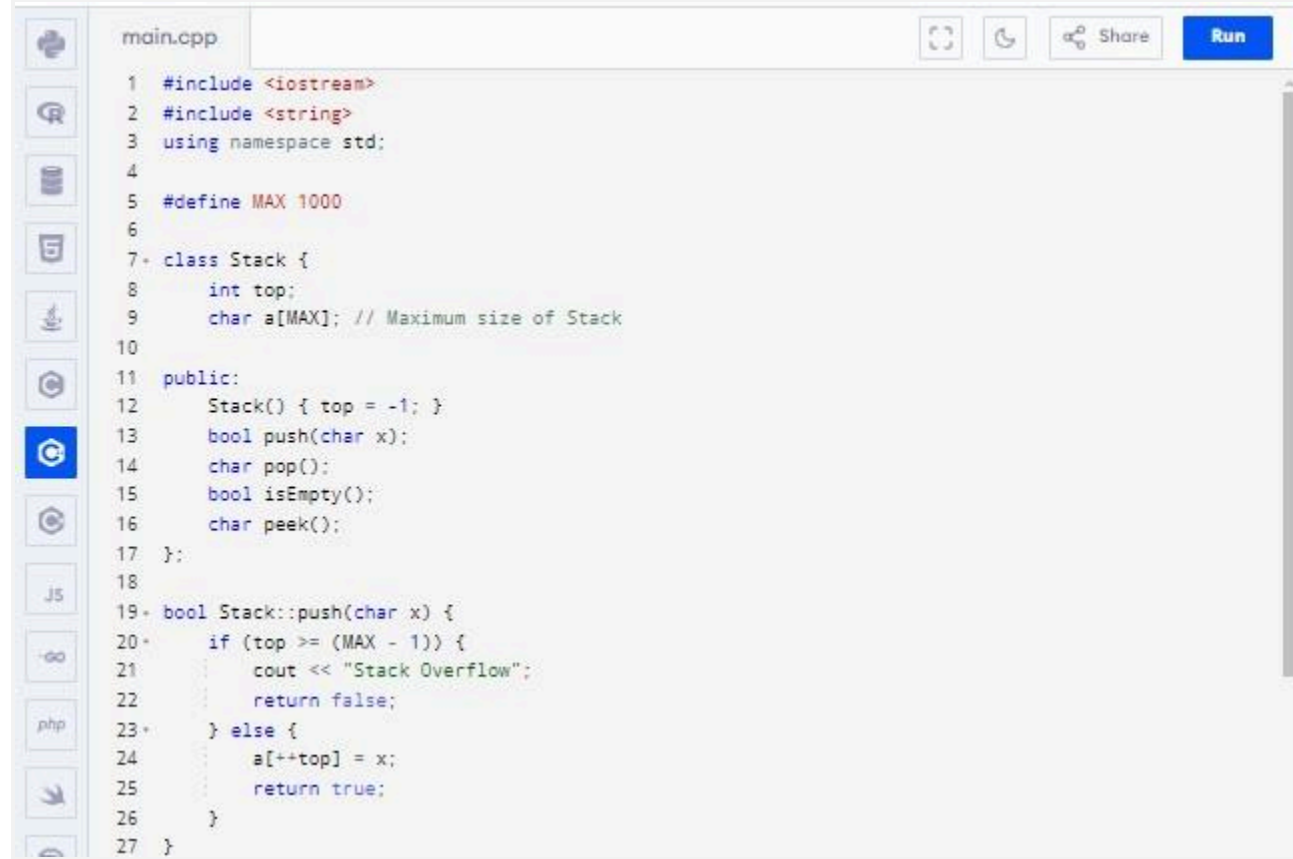
```

```

55 bool isMatchingPair(char character1, char character2) {
56     if (character1 == '(' && character2 == ')')
57         return true;
58     else if (character1 == '{' && character2 == '}')
59         return true;
60     else if (character1 == '[' && character2 == ']')
61         return true;
62     else
63         return false;
64 }
65
66 bool areParenthesesBalanced(string expr) {
67     Stack stack;
68     for (int i = 0; i < expr.length(); i++) {
69         if (expr[i] == '{' || expr[i] == '(' || expr[i] == '[')
70             stack.push(expr[i]);
71         if (expr[i] == '}' || expr[i] == ')' || expr[i] == ']') {
72             if (stack.isEmpty() || !isMatchingPair(stack.pop(), expr[i]))
73                 return false;
74         }
75     }
76     return stack.isEmpty();
77 }
78
79 int main() {
80     string expr = "{()}[]";
81     if (areParenthesesBalanced(expr))
82         cout << "Balanced";
83     else
84         cout << "Not Balanced";
85     return 0;
86 }

```


Stack using Linked Lists:



The image shows a code editor window with a file named 'main.cpp'. The editor has a sidebar on the left with various icons for different programming languages and tools. The code is written in C++ and implements a Stack using an array. The code includes headers for `<iostream>` and `<string>`, uses the `std` namespace, and defines a constant `MAX` as 1000. A `Stack` class is defined with a `top` integer and a `char` array `a` of size `MAX`. The class has public methods: `Stack()` (constructor), `push(char x)`, `pop()`, `isEmpty()`, and `peek()`. The `push` method checks for stack overflow (when `top` is greater than or equal to `MAX - 1`) and prints "Stack Overflow" if it occurs. Otherwise, it pushes the character `x` onto the stack by incrementing `top` and storing `x` in `a[top]`. The `pop` method returns the character at the top of the stack and decrements `top`. The `isEmpty` method returns `true` if `top` is -1, and `peek` returns the character at the top of the stack.

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 #define MAX 1000
6
7 class Stack {
8     int top;
9     char a[MAX]; // Maximum size of Stack
10
11 public:
12     Stack() { top = -1; }
13     bool push(char x);
14     char pop();
15     bool isEmpty();
16     char peek();
17 };
18
19 bool Stack::push(char x) {
20     if (top >= (MAX - 1)) {
21         cout << "Stack Overflow";
22         return false;
23     } else {
24         a[++top] = x;
25         return true;
26     }
27 }
```

```

29 char Stack::pop() {
30     if (top < 0) {
31         cout << "Stack Underflow";
32         return 0;
33     } else {
34         char x = a[top--];
35         return x;
36     }
37 }
38
39 bool Stack::isEmpty() {
40     return (top < 0);
41 }
42
43 char Stack::peek() {
44     if (top < 0) {
45         cout << "Stack is Empty";
46         return 0;
47     } else {
48         return a[top];
49     }
50 }
51
52 bool isMatchingPair(char character1, char character2) {
53     if (character1 == '(' && character2 == ')')
54         return true;
55     else if (character1 == '{' && character2 == '}')
56         return true;
57     else if (character1 == '[' && character2 == ']')
58         return true;
59     else
60         return false;
61 }

```

```

63 bool areParenthesesBalanced(string expr) {
64     Stack stack;
65     for (int i = 0; i < expr.length(); i++) {
66         if (expr[i] == '{' || expr[i] == '(' || expr[i] == '[')
67             stack.push(expr[i]);
68         if (expr[i] == '}' || expr[i] == ')' || expr[i] == ']') {
69             if (stack.isEmpty() || !isMatchingPair(stack.pop(), expr[i]))
70                 return false;
71         }
72     }
73     return stack.isEmpty();
74 }
75
76 int main() {
77     string expr = "((A+B)+(C-D))";
78     if (areParenthesesBalanced(expr))
79         cout << "Balanced";
80     else
81         cout << "Not Balanced";
82     return 0;
83 }

```

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

8. Conclusion

Implementing a stack in C++ has provided a solid grasp of this fundamental data structure, which operates on a Last In, First Out (LIFO) basis. You've learned key operations like push, pop, and peek, which are essential for managing data efficiently. This experience not only enhances your problem-solving skills but also prepares you for more complex data structures and algorithms. Mastering stack implementation is a significant milestone in your programming journey, equipping you to handle various computational tasks effectively.

9. Assessment Rubric