

Activity No. 4.2

4.2 Stacks

Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 8/28/2025
Section: CPE21S4	Date Submitted: 8/28/2025
Name(s): Crishen Luper S. Pulgadp	Instructor: Engr. Jimlord Quejado

6. Output



The screenshot shows the Code::Blocks IDE interface with the file `stack.h` open in the code editor. The code defines a stack class with private members `top` and `arr`, and public methods `isempty()`, `push(T value)`, `pop()`, `top()`, and `size()`. The `push` method checks if the stack is full before pushing. The `pop` method returns the top element and decrements the top pointer. The `top` method returns the element at the top without changing the stack. The `size` method returns the current size of the stack.

```
#include <iostream>
#include <vector>
using namespace std;

class Stack {
private:
    int top = -1;
    vector<int> arr;
public:
    void isEmpty() {
        if (top == -1)
            cout << "the stack is empty";
        else
            cout << "the value of the stack is " << arr[top];
    }
    void push(T value) {
        if (top == arr.size() - 1)
            cout << "STACK OVERFLOW" << endl;
        else
            arr[top + 1] = value;
            cout << "Successfully pushed " << value << endl;
    }
    T pop() {
        bool isEmpty();
        return arr[top];
    }
    T top() {
        bool isEmpty();
        return arr[top + 1];
    }
    int size() {
        void isEmpty();
        if (top == -1)
            cout << "the stack is empty ";
        else
            cout << "Successfully pop " << arr[top - 1] << endl;
    }
};
```

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** File, Edit, View, Insert, Length, TMR, Insert, Stack, Stack pointer, Stack pointer, Stack pointer.
- Toolbar:** File, Edit, Search, New, Direct, Backup, Disk, Help, Windows, Data.
- Menu Bar:** Project, Classes, Debug, Stack, Stack pointer, Stack pointer.
- Code Editor:** The main area contains the following C code for a stack implementation:

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 100

int top = -1;
int arr[MAX];

void push(int value) {
    if (top == MAX - 1) {
        printf("Stack Overflow");
        exit(1);
    } else {
        arr[++top] = value;
        printf("Successfully pushed %d\n", arr[top]);
    }
}

int pop() {
    if (top == -1) {
        printf("Stack Underflow");
        exit(1);
    } else {
        int value = arr[top];
        arr[top] = 0;
        top--;
        return value;
    }
}

int isEmpty() {
    if (top == -1)
        return 1;
    else
        return 0;
}

int peek() {
    if (top == -1)
        return -1;
    else
        return arr[top];
}

int main() {
    int choice;
    while (1) {
        printf("\n1. Push\n2. Pop\n3. Peek\n4. IsEmpty\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                push(45);
                break;
            case 2:
                printf("Popped element: %d\n", pop());
                break;
            case 3:
                printf("Top element: %d\n", peek());
                break;
            case 4:
                if (isEmpty())
                    printf("Stack is empty\n");
                else
                    printf("Stack is not empty\n");
                break;
            case 5:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
}
```

Compiler Status Bar: Compiler (green), Notifications (yellow), Compiling (green), Settings (grey), Find Results (grey), Clean (grey).

Message Bar: In file included from C:\Users\HP\Desktop\CP1010\st\stack\practice.cpp: In function 'main': [Warning] control reaches end of non-void function [-Wreturn-type].

The screenshot shows a C++ development environment with three windows:

- Code Editor:** Displays the `stack.cpp` file containing the `Stack` class definition and its usage.
- Compiler Log:** Shows the build process and the command used to run the program.
- Terminal:** Displays the execution of the program, showing the stack operations and their results.

Code Editor Content (`stack.cpp`):

```
1 // Stack.h
2 #include <iostream>
3 #include <vector>
4
5 class Stack {
6 public:
7     void push(int);
8     int pop();
9     int peek();
10    bool isEmpty();
11    void display();
12 };
13
14 int main() {
15     Stack s;
16     s.push(10);
17     s.push(5);
18     s.push(1);
19     std::cout << "The value of the stack is: " << s.peek() << std::endl;
20     s.pop();
21     std::cout << "Successfully pop 1" << std::endl;
22     std::cout << "The value of the stack is: " << s.peek() << std::endl;
23     s.display();
24     return 0;
25 }
```

Compiler Log:

```
1 Compiler (0) Resources (0) Compile Log (0) Debug (0) Run Results (0) Clean
2
3 Use: Ctrl F5
4
5 C:\Users\TIPQC\Desktop\CP1110\stack\stack.cpp
6 C:\Users\TIPQC\Desktop\CP1110\stack\stack.exe
7
8 In file included from C:\Users\TIPQC\Desktop\CP1110\stack\stack.cpp:1:
9 | [Running] clang++ stack.cpp -o stack.exe [stack.exe] finished by exit code 0
```

Terminal Output:

```
C:\Users\TIPQC\Desktop\CP1110>
1 The stack is empty
2 Succesfully pushed 10
3 Succesfully pushed 5
4 Succesfully pushed 1
5 The value of the stack is: 1
6 Successfully pop 1
7 The value of the stack is: 5
8 5
9 10
10
11 -----
12 Process exited after 0.01545 seconds with return value 0
13 Press any key to continue . . . |
```

7. Supplementary Activity

8. Conclusion

To conclude, we can use stacks in arrays with declaring its fixed size. Through using the operations in stack, we can change the elements inside via adding or removing by following the properties of LIFO. In this activity, I think I had learned well how does stack work and arrays.

9. Assessment Rubric

