Implementation

In this lesson, we'll see how to implement a stack.

We'll cover the following

Implementation

A stack can be implemented using an array or a singly linked list, we will have the same time complexity.

Array

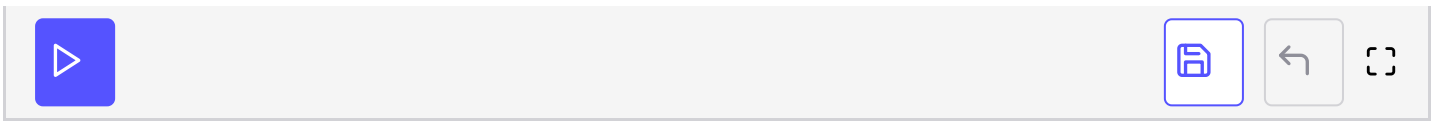
> The limitation in using an array is that the maximum size of the stack is limited.

We'll keep the `top` pointer and `A[0]` will be the bottommost element.

The below code is pretty self-explanatory. Run the code to see the printed stack at every step.

```
#include <bits/stdc++.h>
using namespace std;

struct Stack {
  static const int SZ = 4;
  int arr[SZ];
  int top;

  Stack() {
    top = -1;
  }

  bool isEmpty() {
    return (top < 0);
  }
```

```
  void push(int x) {
    if (top == SZ - 1) {

      cout << "Stack Overflow";
      return;
    }
    arr[++top] = x;
  }

  int peek() {
      if (top < 0) {
        cout << "Empty Stack";
        return -1;
      }
      return arr[top];
  }

  int pop() {
    if (top < 0) {
      cout << "Stack Underflow";
      return - 1;
    }
    return arr[top--];
  }

  void print_stack() {
    for (int i = 0; i <= top; i++)
      cout << arr[i] << " <- ";
    cout << "\n";
  }
};

int main() {
  Stack stack;
  stack.print_stack();
  stack.push(1);  stack.print_stack();
  stack.push(2);  stack.print_stack();
  stack.push(3);  stack.print_stack();
  stack.pop();  stack.print_stack();
  stack.pop();  stack.print_stack();
  return 0;
}
```



Linked list

Stacking using a singly linked list is very similar and we'll skip the code for that as an exercise.

Hint: Treat `head` as the top of the stack.

Push => inserts a node at the beginning.

Pop => deletes first node.

In the next lesson, we'll discuss how to use C++ STL stack.