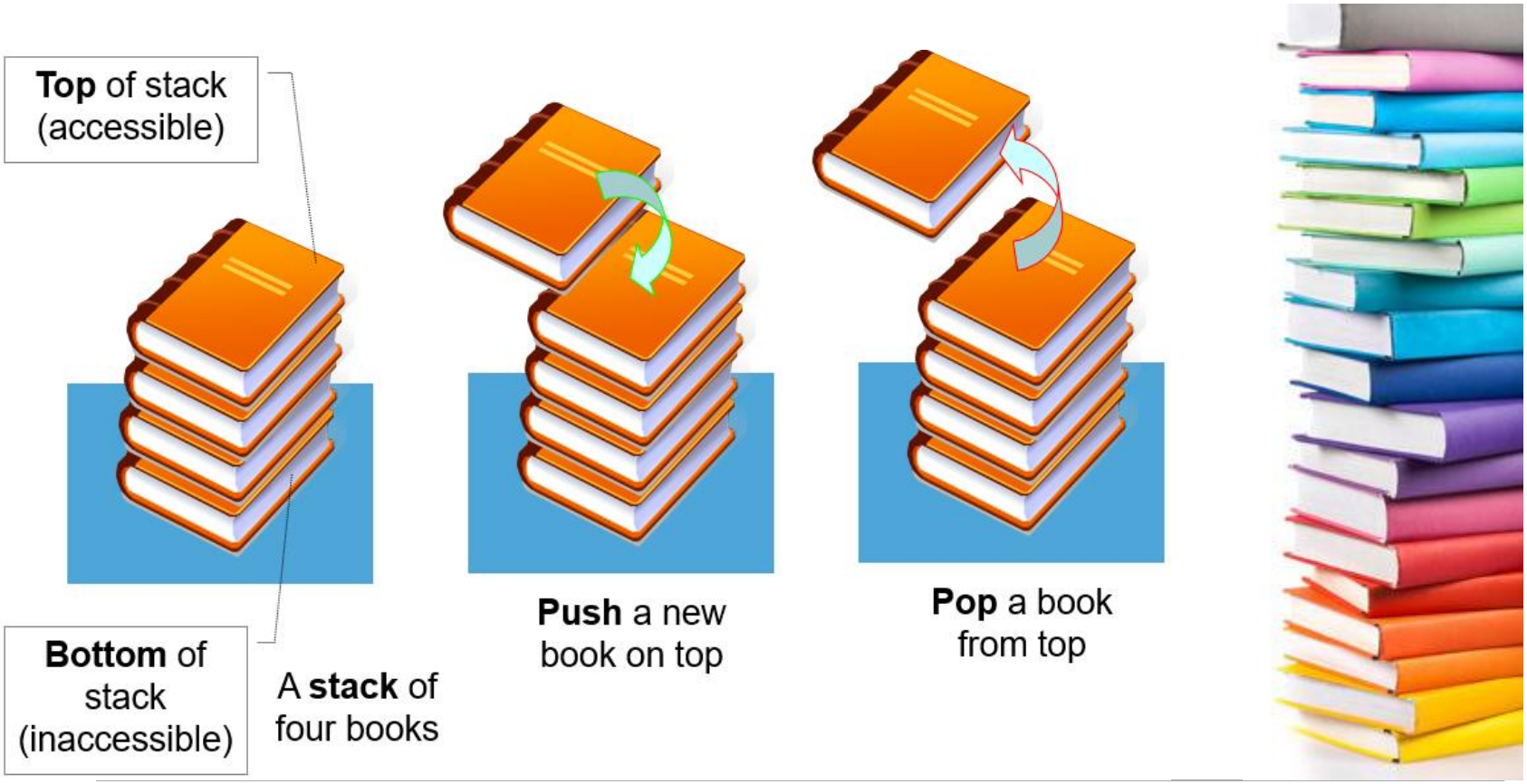
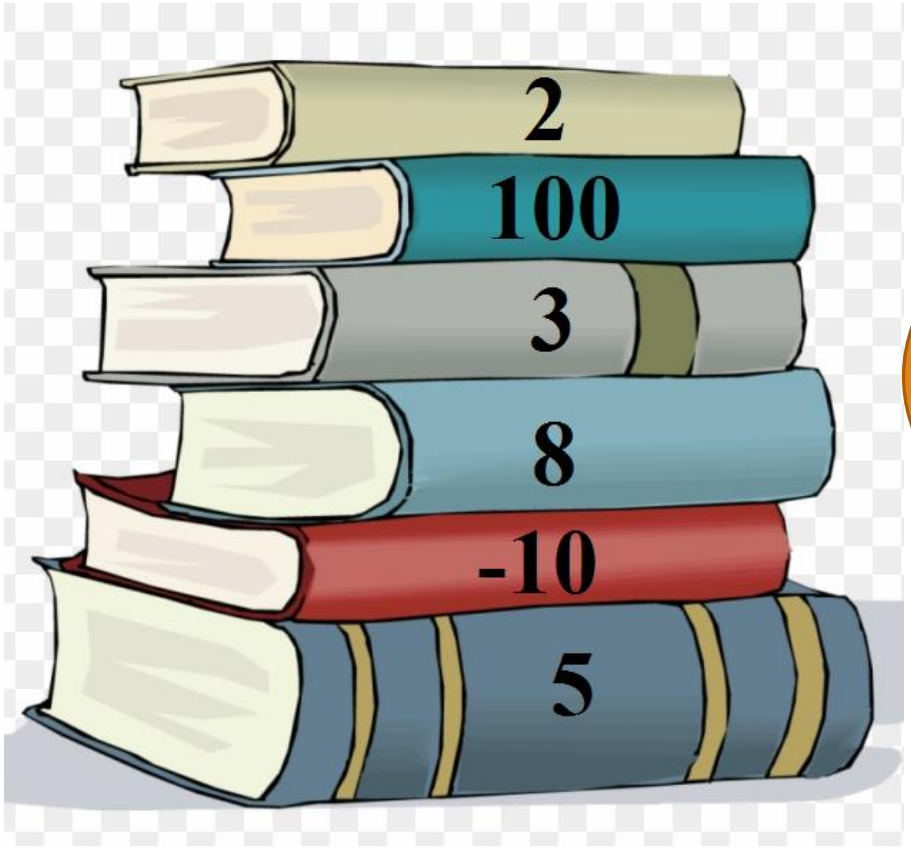
Data Structure

Lec 01 Stack


Top of stack (accessible)
Bottom of stack (inaccessible)
A **stack** of four books
Push a new book on top
Pop a book from top

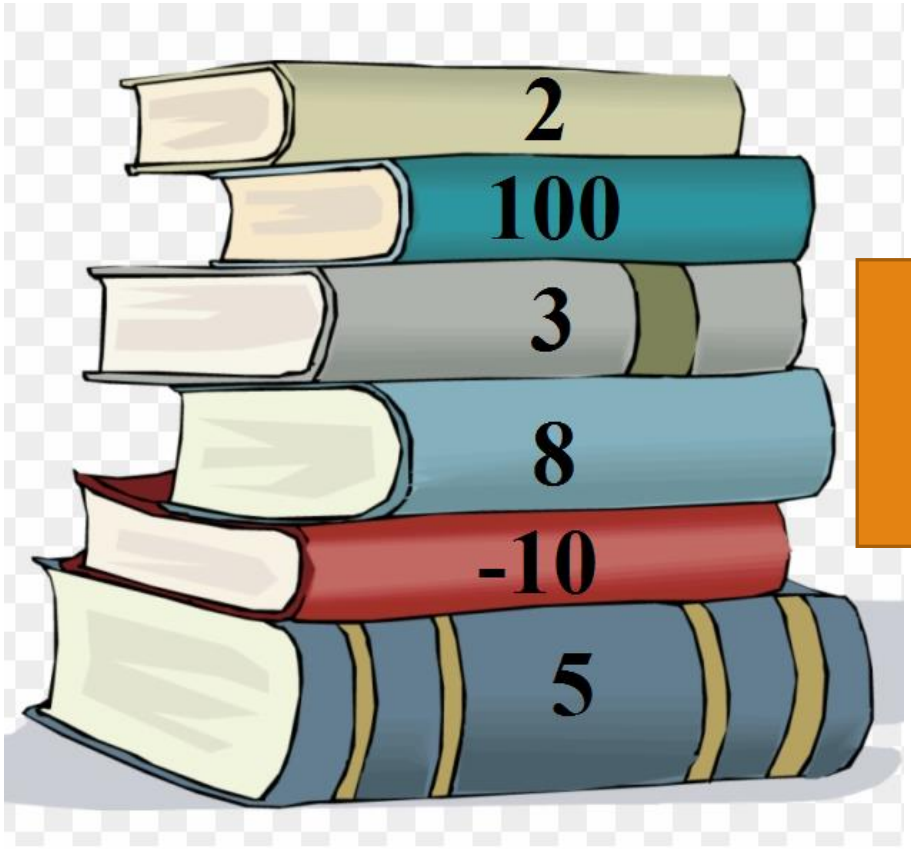
STACK

Introduction to Stack



How to store them in Memory?????

Introduction to Stack

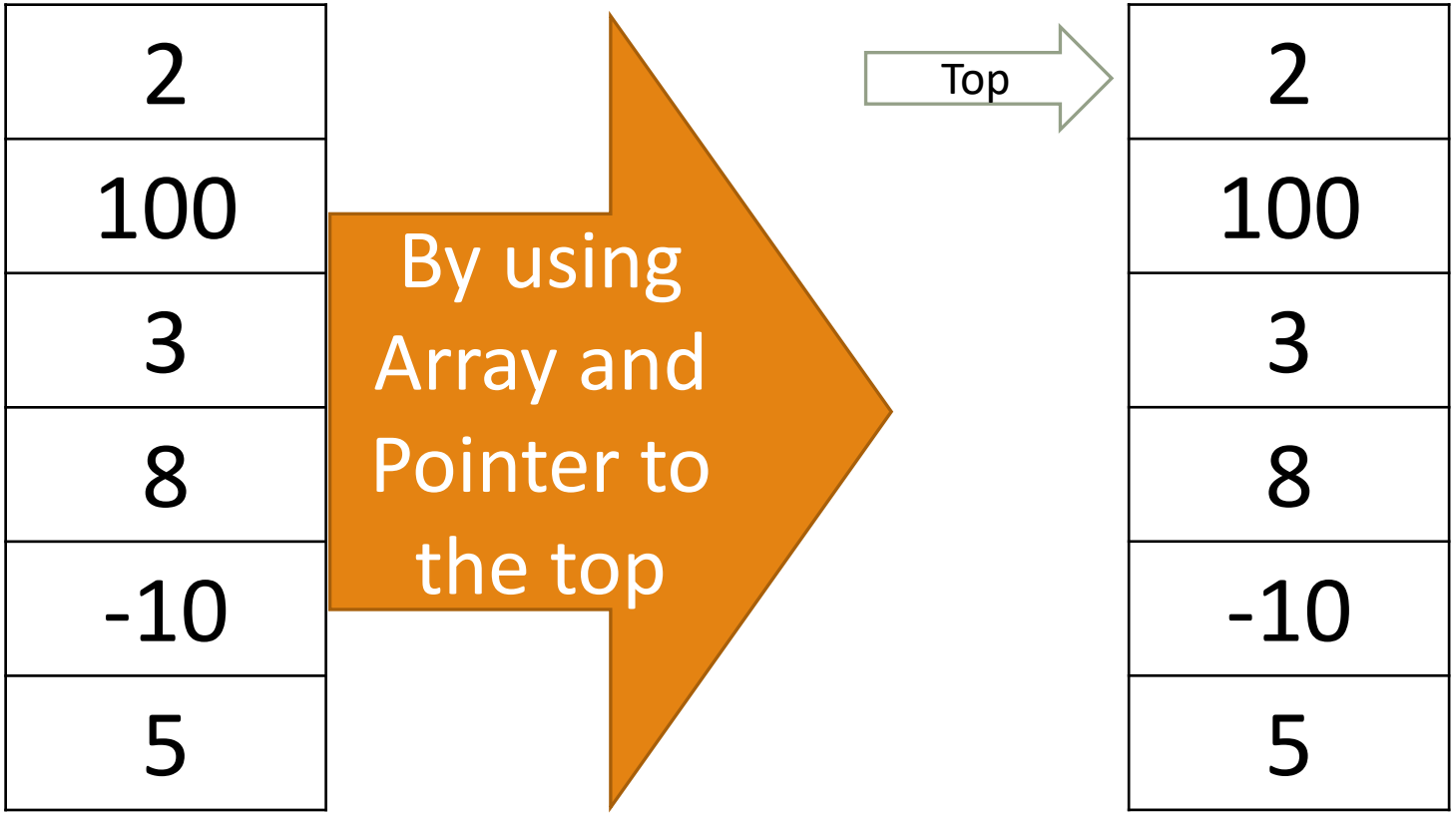


By using Array

2
100
3
8
-10
5

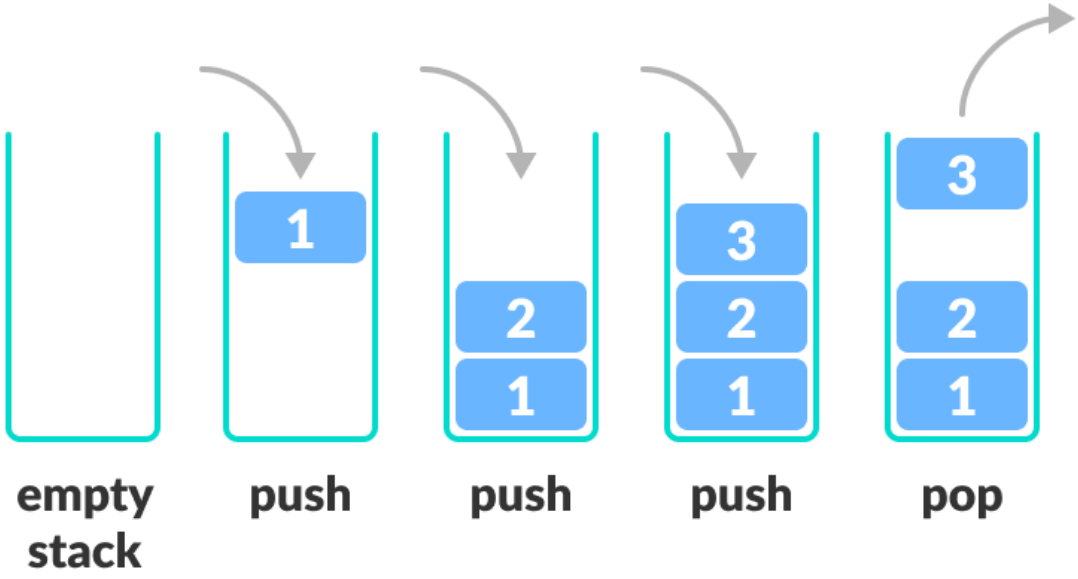
One-Dimension array

Introduction to Stack



Introduction to Stack

- Stack is an *ordered collection of items* in which *new data items* may be *added* to or *deleted* from *only one end*, called the *top of* the *stack*.
- All the *addition* and *deletion* in a stack is done from the *top of* the stack, the last added element will be first removed from the stack. That is why the stack is also called Last-in-First-out(LIFO).



Introduction to Stack

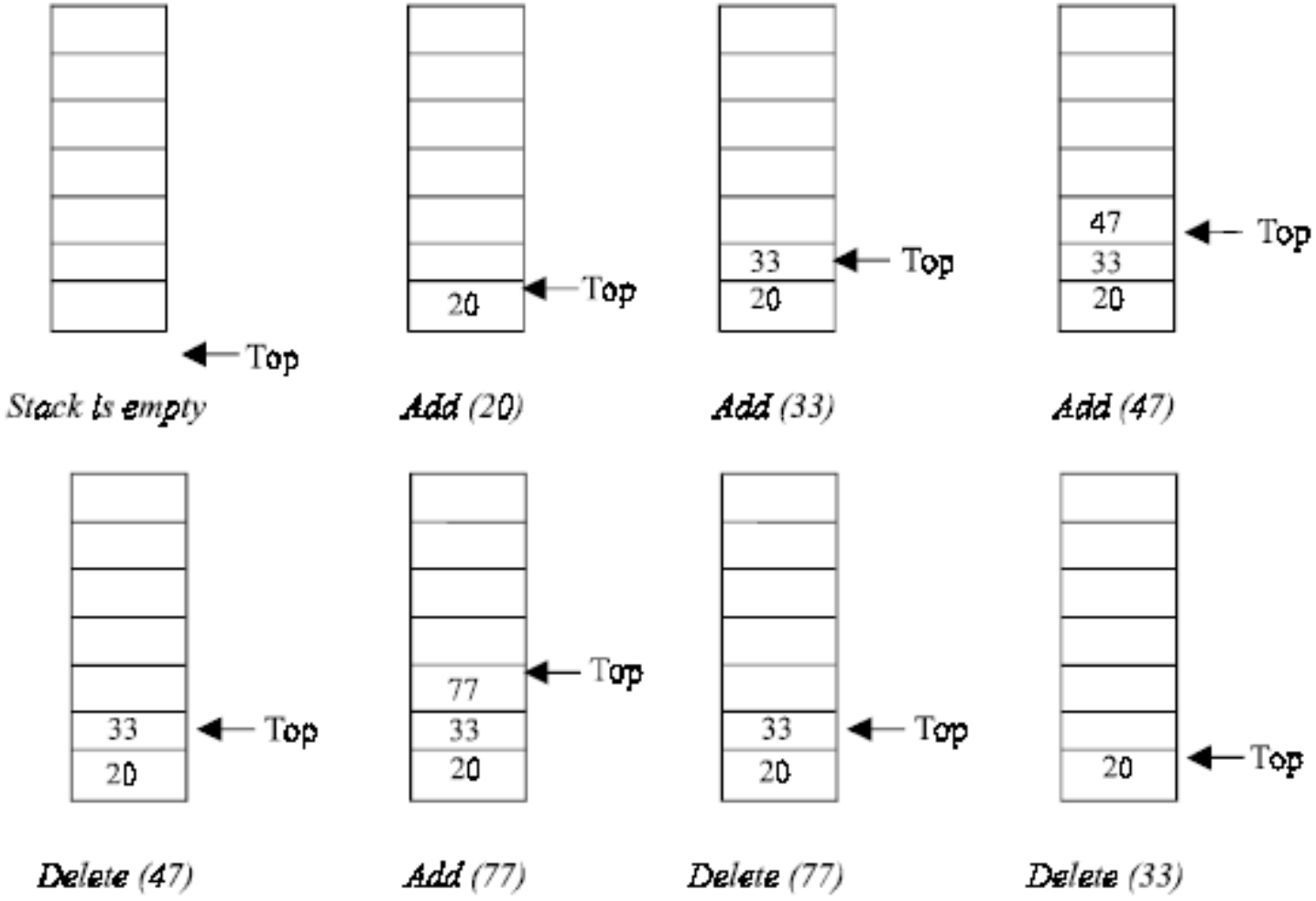
The *insertion* (or *addition*) operation is referred to as push.

The *deletion* (or *remove*) operation as pop.

A stack is said to be *empty* or *underflow*, if the stack contains no elements. At this point the top of the stack is present at *the bottom* of the stack.

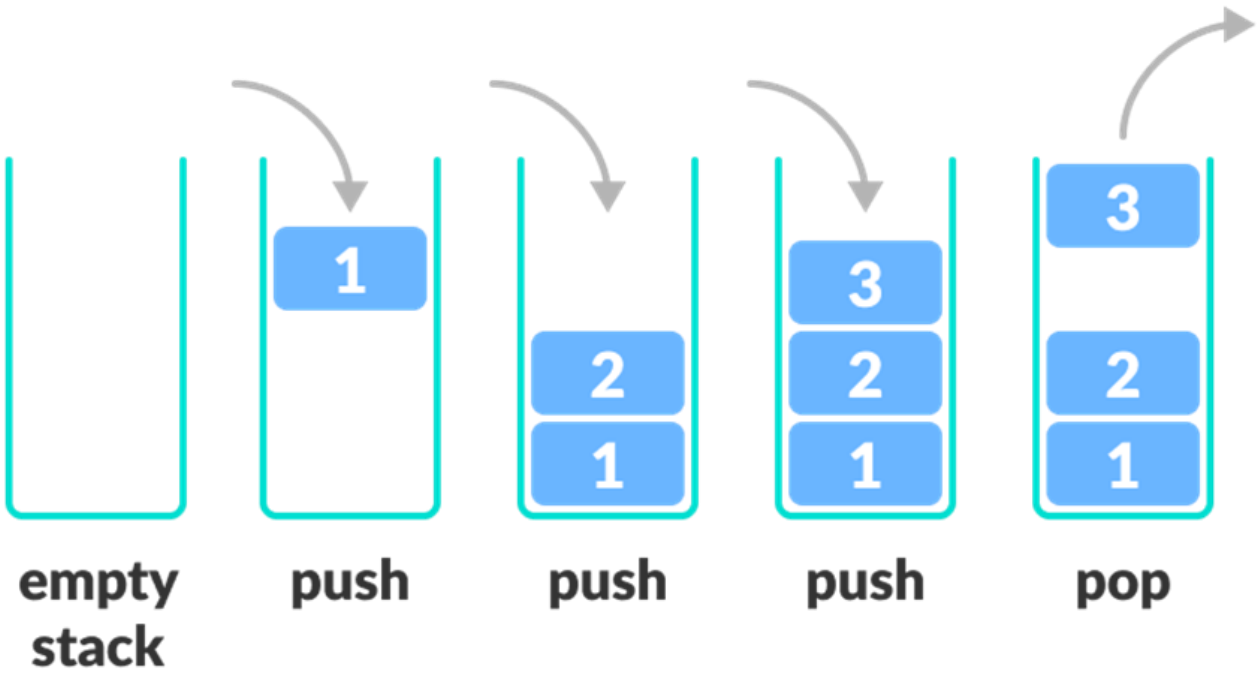
A stack is *overflow* when it becomes full, i.e., no other elements can be pushed onto the stack. At this point the top pointer is at *the highest* location of the stack.

Introduction to Stack

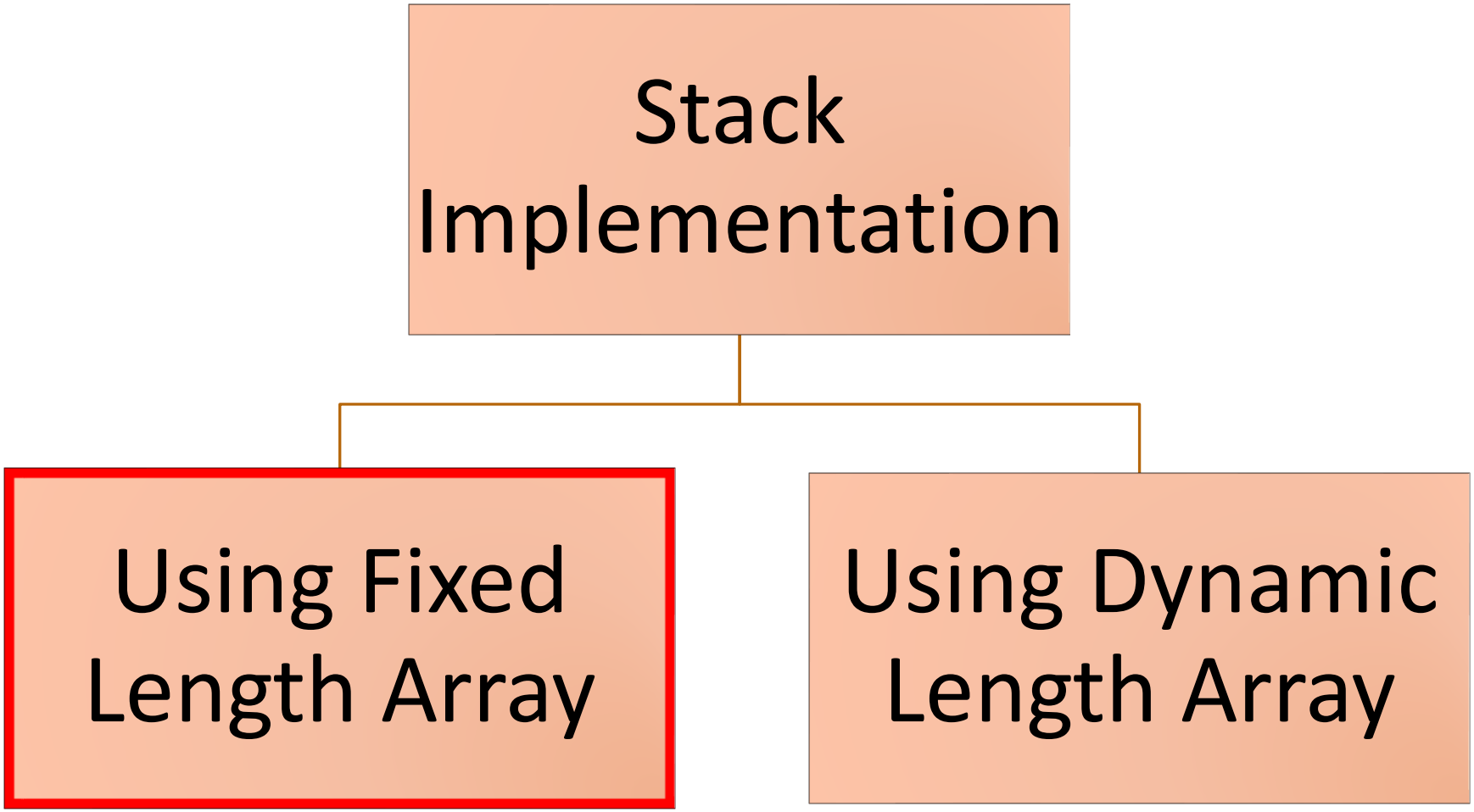


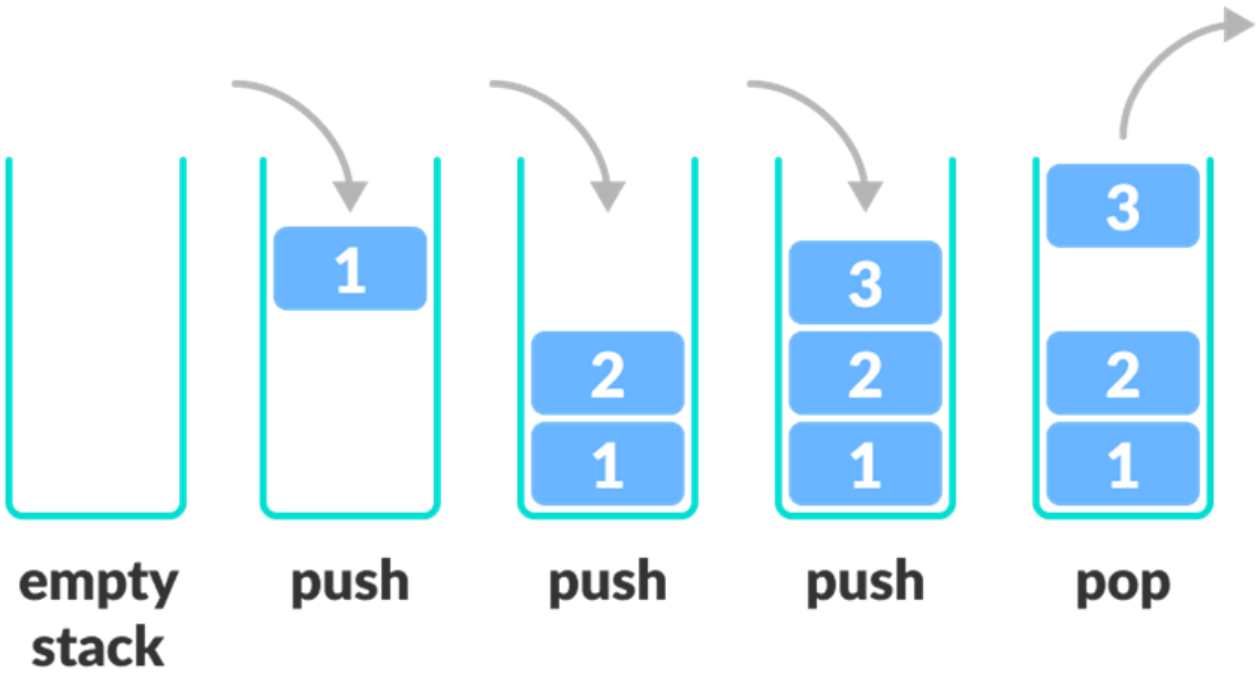
Operations Performed on Stack

- PUSH: The process of adding (or inserting) a new element to the top of the stack is called PUSH operation.
  - Pushing an element to a stack will add the new element at the top. After every push operation the top is incremented by one.
  - If the array is full and no new element can be accommodated, then the stack overflow condition occurs.
- POP: The process of deleting (or removing) an element from the top of stack is called POP operation.
  - After every pop operation the stack is decremented by one.
  - If there is no element in the stack and the pop operation is performed then the stack underflow condition occurs.



Stack Implementation


Stack Implementation
Using Fixed Length Array
Using Dynamic Length Array



Stack Using Fixed Length Array

Stack Implementation

```
#include <iostream>
#define max_size 100
using namespace std;

////////////////////////////////////
class stack
{
private:
        int items[max_size];
        int top;
public:
        stack() { top = -1; }
        void push(int x);
        int pop();
        int is_empty();
        int is_full();
        void print_all_elements();
};
```

Stack Implementation

```
////////////////////////////////////
// Is_empty Function
int stack::is_empty()
{
        if (top == -1)
                return 1;
        return 0;
}
////////////////////////////////////
//Is_full Function
int stack::is_full()
{
        if (top == max_size - 1)
                return 1;
        return 0;
}
```

Stack Implementation

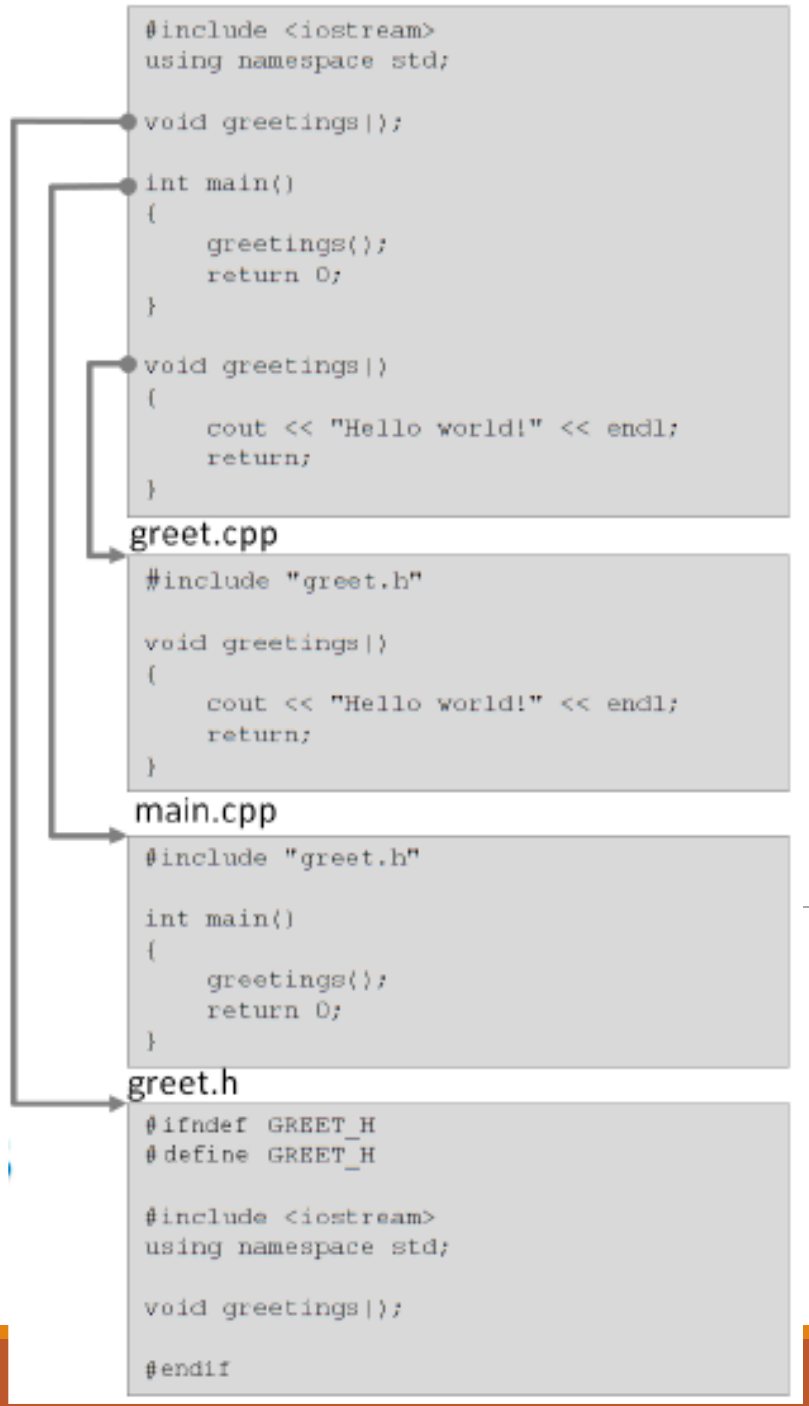
```
////////////////////////////////////////////////////////
// Push Function
void stack::push(int item)
{
        if (is_full())
        {
                cout << "Stack is overflow" << endl;
                return;
        }
        top++;
        items[top] = item;
}
```

Stack Implementation

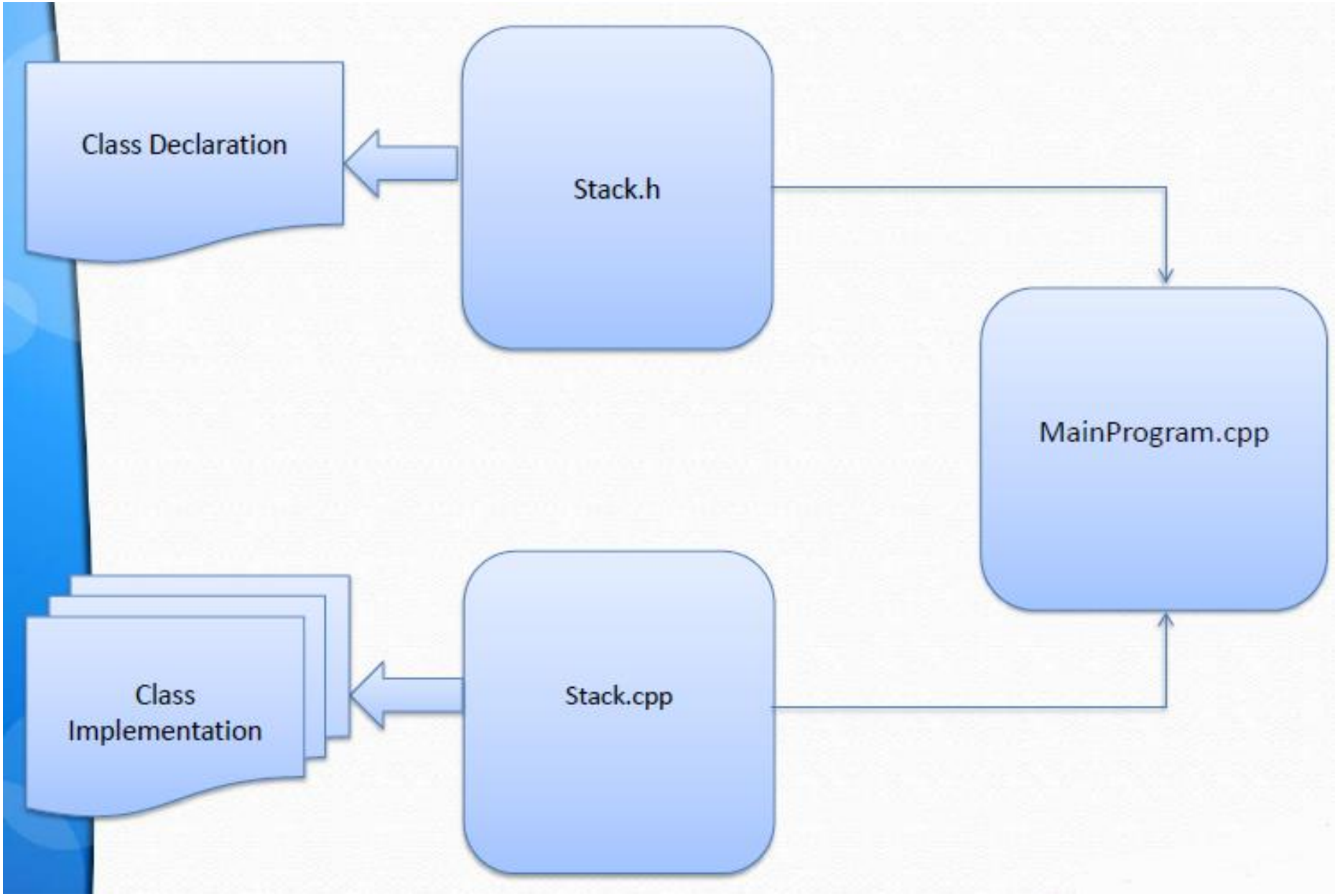
```
//////////////////////////////////////////////////////
// Pop Function
int stack::pop()
{
        if (is_empty())
        {
                cout << "Stack is underflow" << endl;
                return -1;
        }
        int item = items[top];
        top--;
        return item;
}
```

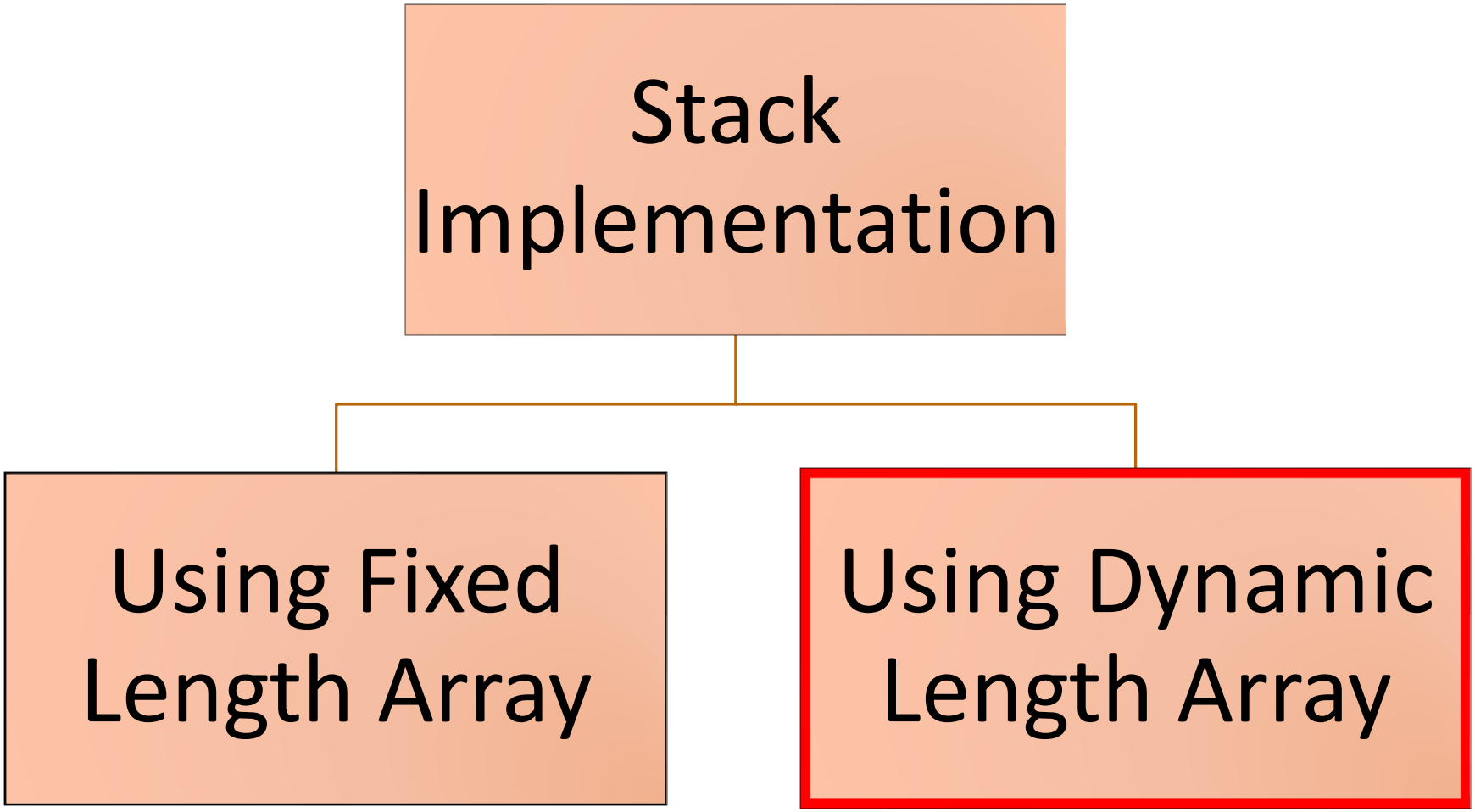
Stack Implementation

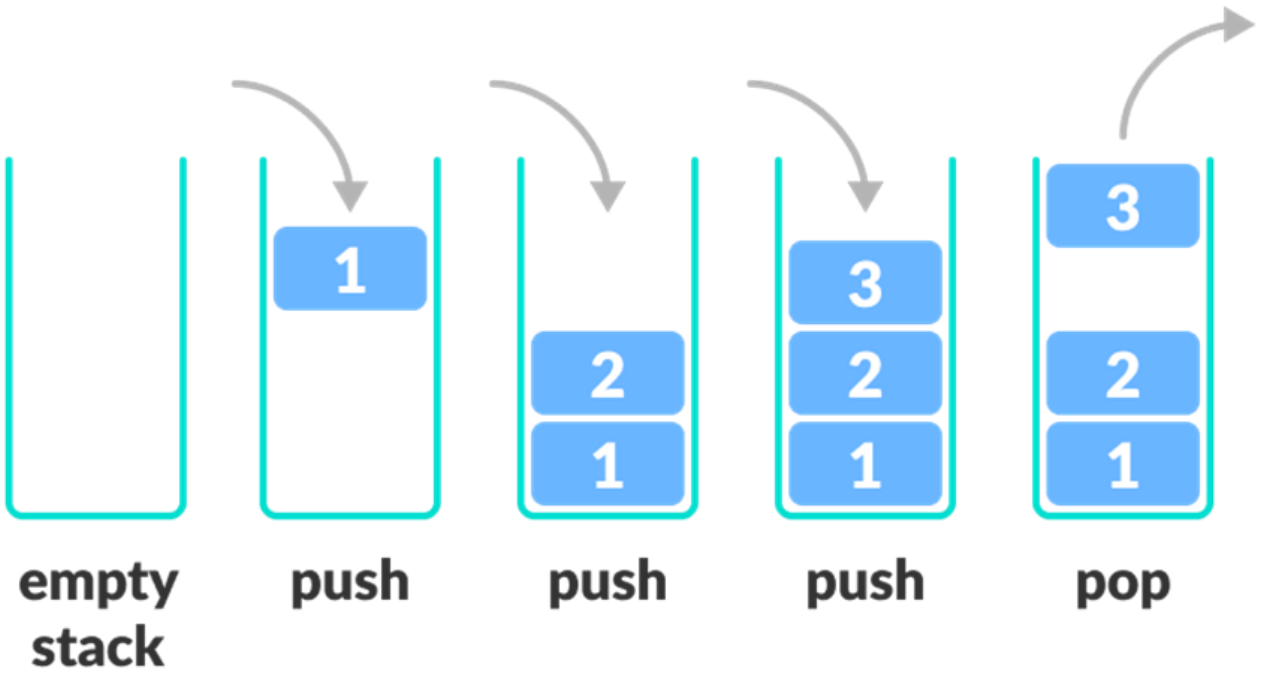
```
void stack::print_all_elements()
{

        for (int i = 0; i <= top ; i++)
                cout << items[i] << " ";
        cout << endl;
}
void main()
{
        stack s;
        s.push(5);
        s.push(3);
        s.push(500);
        s.print_all_elements();
        cout << s.pop() << endl;
        cout << s.pop() << endl;
        s.print_all_elements();
        system("pause");
}
```



Dealing With Multiple Files


Class Declaration
Stack.h
MainProgram.cpp
Class Implementation
Stack.cpp


Stack Implementation
Using Fixed Length Array
Using Dynamic Length Array



Stack Using Dynamic Length Array

Stack Implementation

```
#ifndef STACK_H
#define STACK_H

class stack{
private:
        int max_size;
        int* items;
        int top;
public:
        stack();
        ~stack();
        void push(int);
        int pop();
        bool is_full();
        bool is_empty();
        int return_top();
        void print_all_elements();
};
#endif
```

Stack Implementation

```
#include <iostream>
using namespace std;
#include "stack.h"

stack::stack()
{
        cout << "Enter stack size : ";
        cin >> max_size;
        items = new int[max_size];
        top = -1;
}
//////////////////////////////////////
stack::~stack()
{
        delete items;
        items = NULL;
        top = -1;
        max_size = 0;
}
```

Stack Implementation

```
//////////////////////////////////////////////////////////
bool stack::is_full()
{
        return (top == (max_size - 1));
}
//////////////////////////////////////////////////////////
bool stack::is_empty()
{
        return (top == -1);
}
//////////////////////////////////////////////////////////
```

Stack Implementation

```
////////////////////////////////////////////////////////
void stack::push(int item)
{
        if (is_full())
        {
                cout << "Error : Stack is overflow\n";
                return;
        }
        top++;
        items[top] = item;
}
```

Stack Implementation

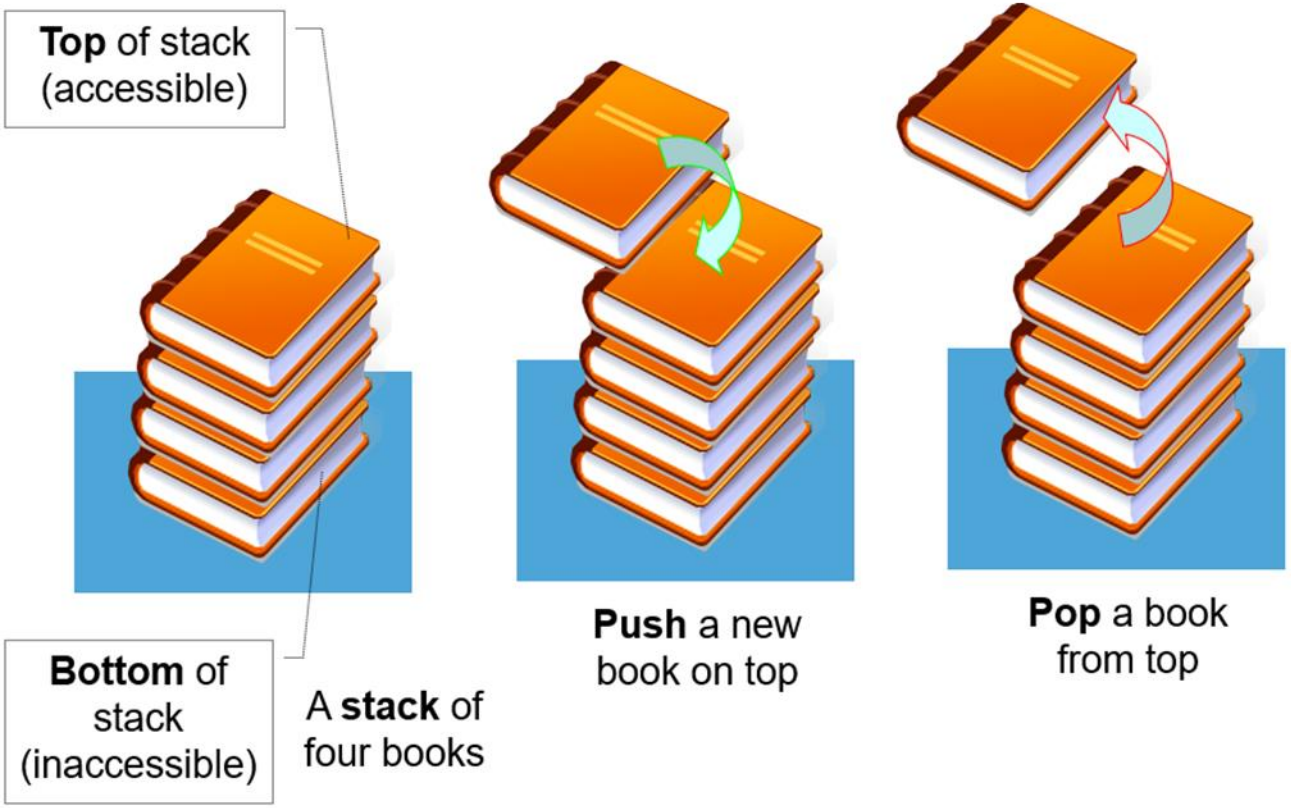
```
//////////////////////////////////////////////////////////
int stack::pop()
{
        if (is_empty())
        {
                cout << "Error : Stack is underflow\n";
                return -1;
        }
        int item = items[top];
        top--;
        return item;
}
```

Stack Implementation

```
////////////////////////////////////////////////////////
int stack::return_top()
{
        if (is_empty())
        {
                cout << "Error : Stack is underflow\n";
                return -1;
        }
        return items[top];;
}
////////////////////////////////////////////////////////
void stack::print_all_elements()
{
        cout << "Stack : | ";
        for (int i = 0; i <= top; i++)
                cout << items[i] << " | ";
        cout << endl;
}
////////////////////////////////////////////////////////
```

Stack Implementation

```
////////////////////////////////////
////////////////////////////////////
#include <iostream>
using namespace std;
#include "stack.h"

void main()
{
        stack s;
        s.push(5);
        s.push(3);
        s.push(500);
        s.print_all_elements();
        cout << s.pop() << endl;
        cout << s.pop() << endl;
        s.print_all_elements();
        system("pause");
}
```



Stack Applications

Some Stack Applications

- Run-time stack used in function calls.
- Page-visited history in a Web browser.
- Undo sequence in a text editor.
- Removal of recursion.
- Conversion of Infix to Postfix notation.
- Evaluation of Postfix expressions.
- Reversal of sequences.
- Checking for balanced symbols.