C++

程式語言（二）

Introduction to Programming (II)

A Case Study: Stack

Joseph Chuang-Chieh Lin

Dept. CSE, NTOU

Platform/IDE

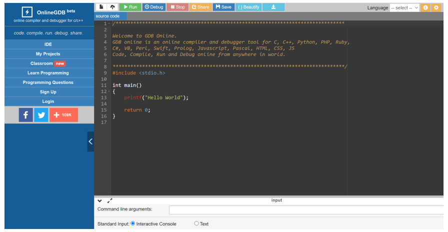
- Dev-C++

https://www.pngegg.com/en/search?q=Dev-C

- Codeblocks

https://icons8.com/icons/set/code-blocks

- OnlineGDB (https://www.onlinegdb.com/)

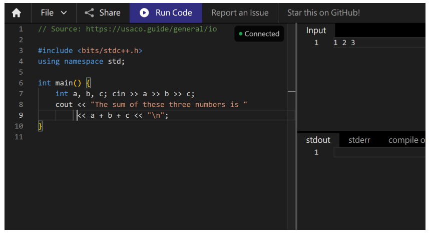


```
/******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,
C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

Command line arguments:

Standard Input: [✓ Interactive Console] Text

- Real-Time Collaborative Online IDE (https://ide.usaco.guide/)



```
// Source: https://usaco.guide/general/io

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

int main() {
    int a, b, c; cin >> a >> b >> c;
    cout << "The sum of these three numbers is "
        << a + b + c << "\n";
}
```

Input:

```
1 2 3
```

Textbooks (We focusing on C++11)

- ***Learn C++ Programming by Refactoring* (由重構學習 C++ 程式設計). Pang-Feng Liu (劉邦鋒). NTU Press. 2023.**
- ***C++ Primer. 5th Edition.* Stanley B. Lippman, Josée Lajoie, Barbara E. Moo. 2019.**
- *Effective C++*. Scott Meyers. O’Reilly. 2016.
- *Thinking in C++. Vol. 1: Introducing to Standard C++*. 2nd Edition. Bruce Eckel. Prentice Hall PTR. 2000.

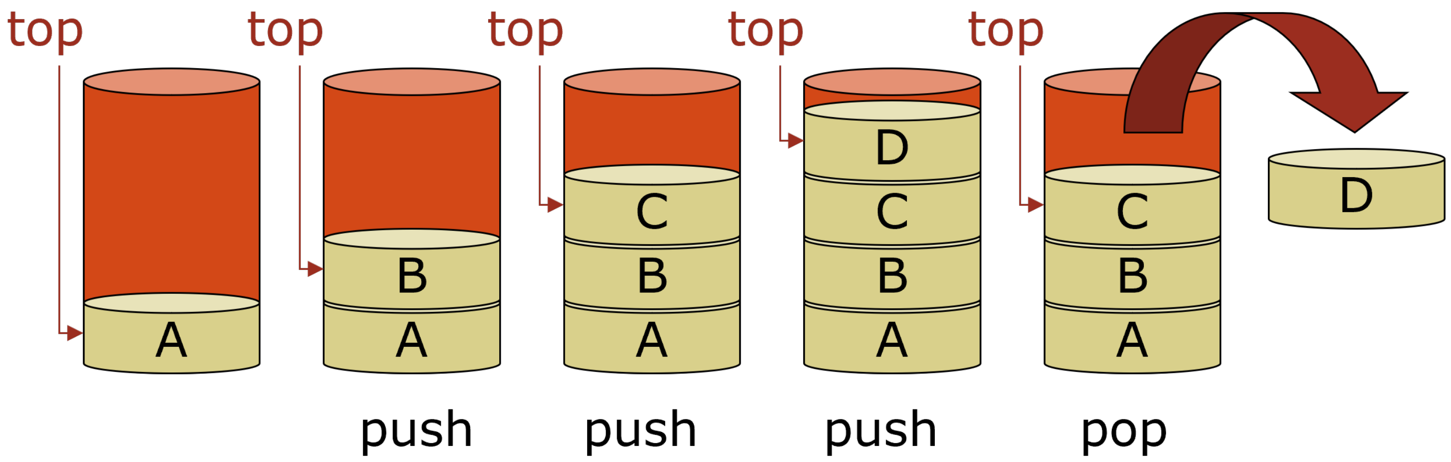
Useful Resources

- Tutorialspoint
  - https://www.tutorialspoint.com/cplusplus/index.htm
  - Online C++ Compiler
- Programiz
  - https://www.programiz.com/cpp-programming
- LEARN C++
  - https://www.learncpp.com/
- MIT OpenCourseWare - Introduction to C++
  - https://ocw.mit.edu/courses/6-096-introduction-to-c-january-iap-2011/pages/lecture-notes/
- Learning C++ Programming
  - https://www.programiz.com/cpp-programming
- GeeksforGeeks
  - https://www.geeksforgeeks.org/c-plus-plus/

Stack

Stack

- LIFO: Last In, First Out



Implementation Using Linked List

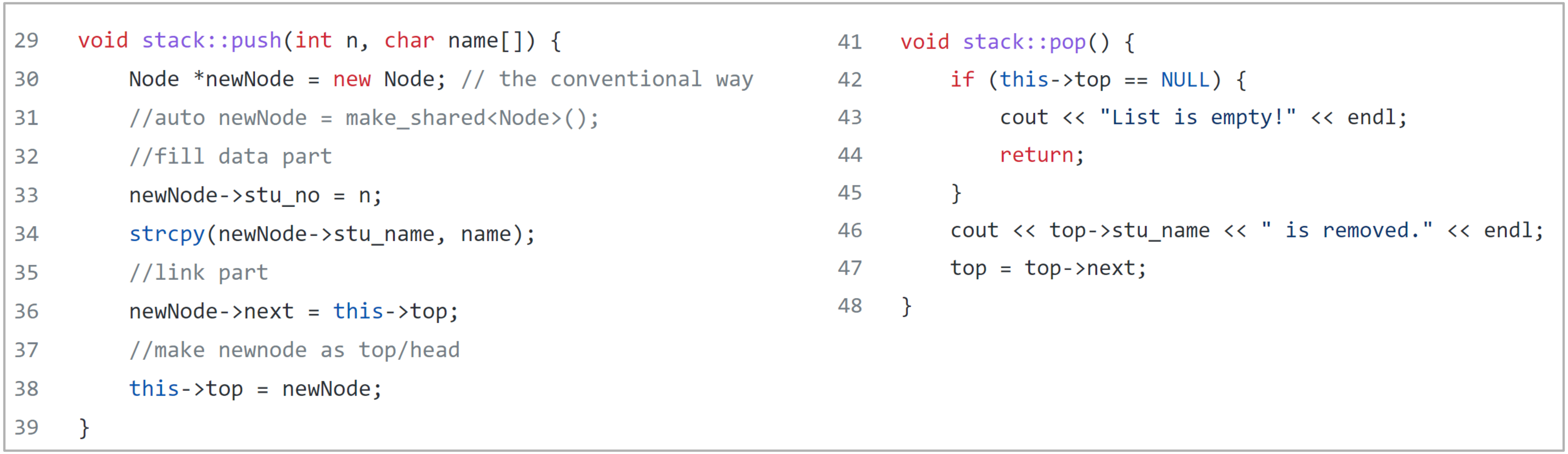
- The code in my GitHub page: link
- Code on OnlineGDB: https://onlinegdb.com/W0S_dJ_k26

```cpp
struct Node {
    int stu_no;
    char stu_name[50];
    //shared_ptr<Node> next;
    Node *next; // the conventional way
};
```

```cpp
class stack {
private:
    //shared_ptr<Node> top;
    Node *top; // the conventional way

public:
    stack() {
        this->top = NULL;
        cout << " # The stack is generated. " << endl;
    }
    ~stack() { cout << " # The stack is deleted." << endl; }
    void push(int n, char name[]);
    void pop();
    void display();
};
```

Implementation Using Linked List

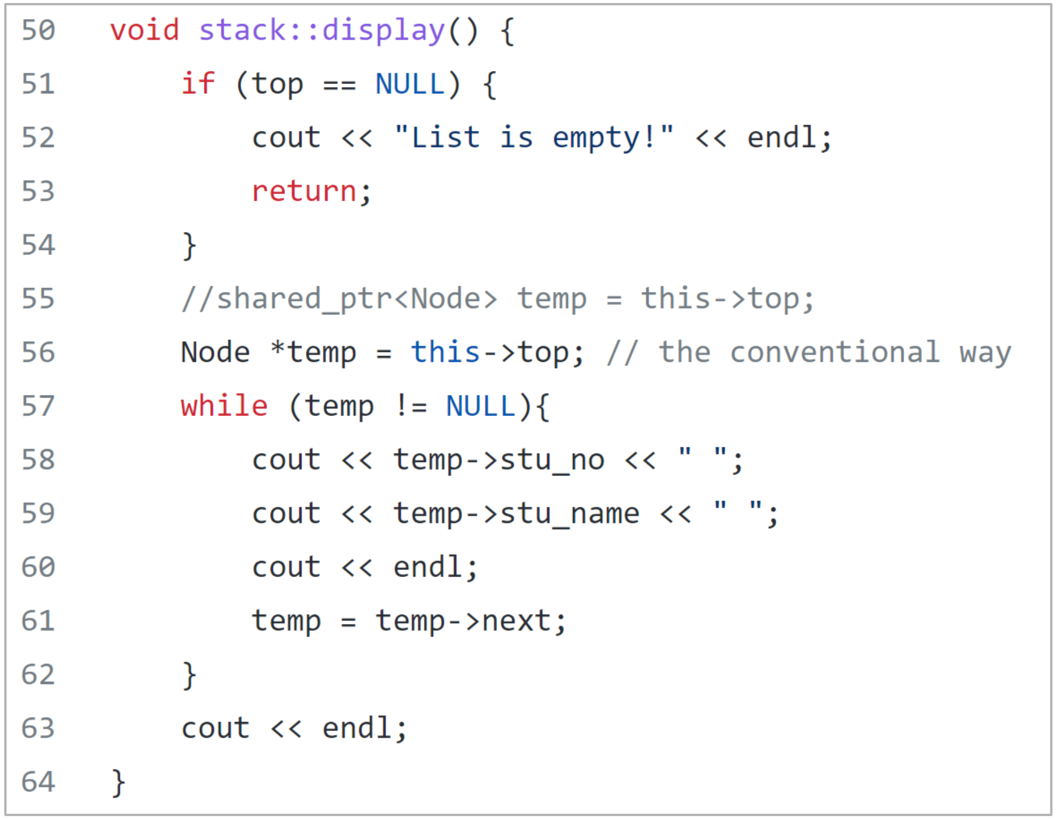
- The code in my GitHub page: link
- Code on OnlineGDB: https://onlinegdb.com/W0S_dJ_k26



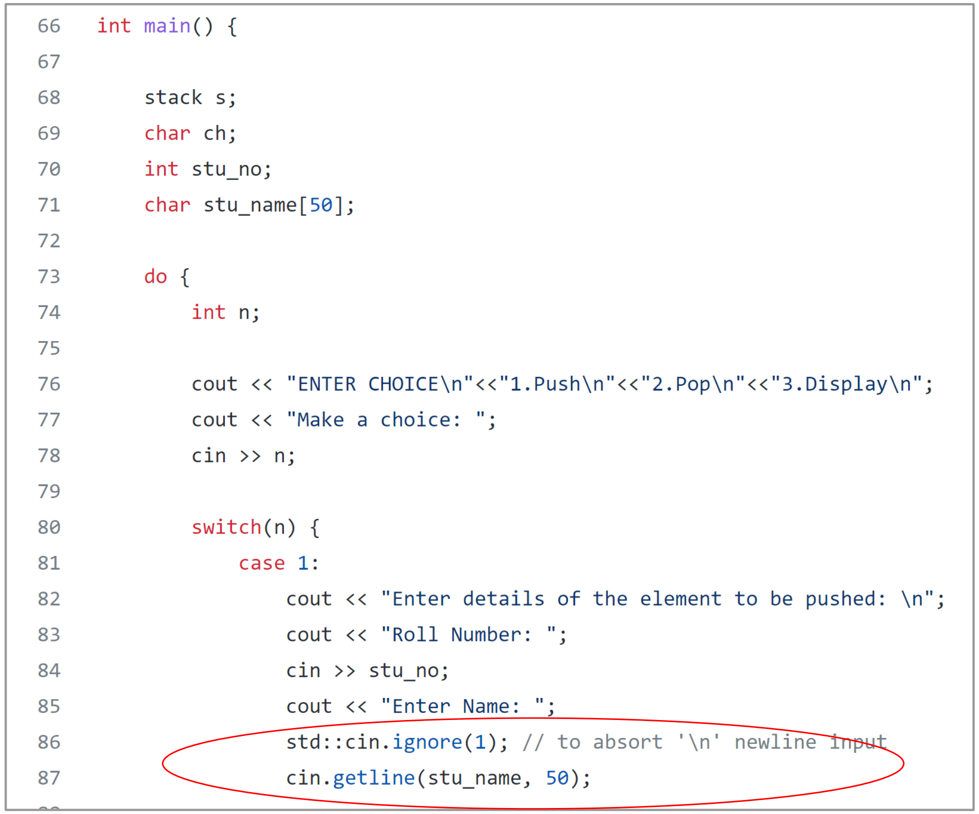
```cpp
void stack::push(int n, char name[]) {
    Node *newNode = new Node; // the conventional way
    //auto newNode = make_shared<Node>();
    //fill data part
    newNode->stu_no = n;
    strcpy(newNode->stu_name, name);
    //link part
    newNode->next = this->top;
    //make newnode as top/head
    this->top = newNode;
}
```

```cpp
void stack::pop() {
    if (this->top == NULL) {
        cout << "List is empty!" << endl;
        return;
    }
    cout << top->stu_name << " is removed." << endl;
    top = top->next;
}
```

Implementation Using Linked List



```cpp
void stack::display() {
    if (top == NULL) {
        cout << "List is empty!" << endl;
        return;
    }
    //shared_ptr<Node> temp = this->top;
    Node *temp = this->top; // the conventional way
    while (temp != NULL){
        cout << temp->stu_no << " ";
        cout << temp->stu_name << " ";
        cout << endl;
        temp = temp->next;
    }
    cout << endl;
}
```



```cpp
int main() {

    stack s;
    char ch;
    int stu_no;
    char stu_name[50];

    do {
        int n;

        cout << "ENTER CHOICE\n"<<"1.Push\n"<<"2.Pop\n"<<"3.Display\n";
        cout << "Make a choice: ";
        cin >> n;

        switch(n) {
            case 1:
                cout << "Enter details of the element to be pushed: \n";
                cout << "Roll Number: ";
                cin >> stu_no;
                cout << "Enter Name: ";
                std::cin.ignore(1); // to absort '\n' newline input
                cin.getline(stu_name, 50);
```

A Simplified Version

- https://onlinegdb.com/rQ1j_k3Fiz
- The code in my GitHub page: link

```cpp
struct Node {
    int stu_no;
    Node *next; // the conventional way
};
```

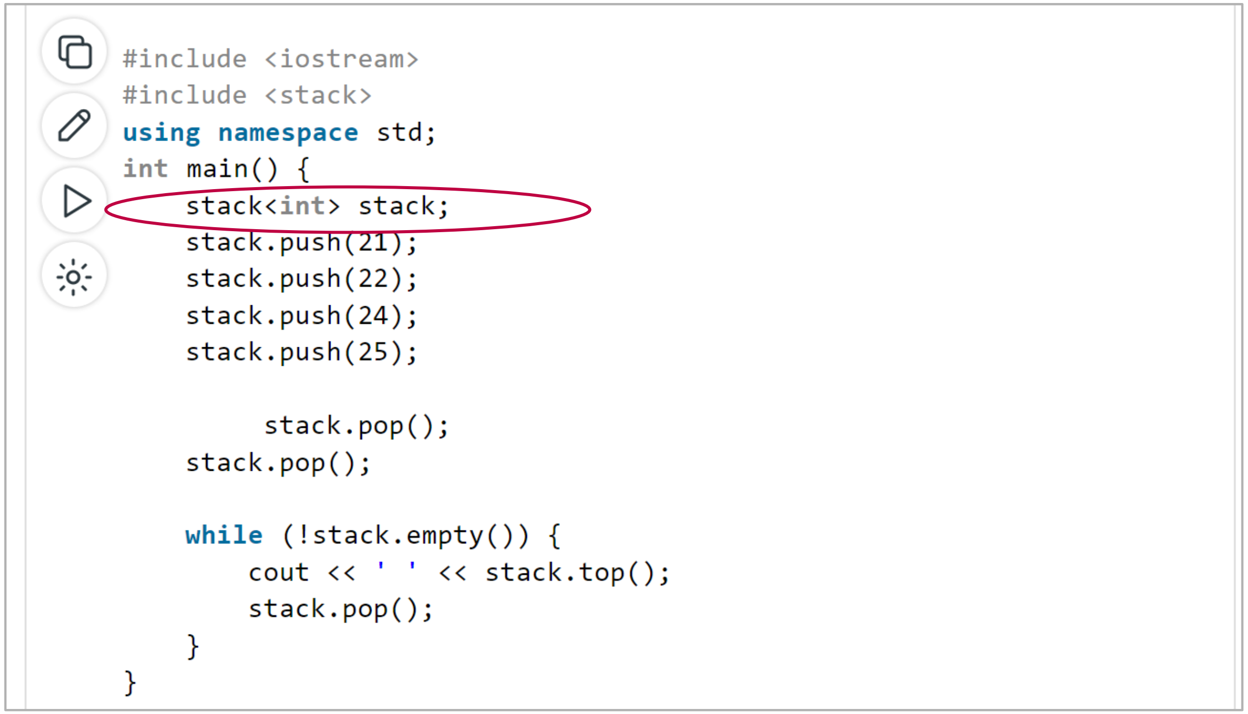
```cpp
class stack {
private:
    Node *top; // the conventional way

public:
    stack() {
        this->top = NULL;
        cout << " # The stack is generated. " << endl;
    }
    ~stack() { cout << " # The stack is deleted." << endl; }
    void push(int n);
    void pop();
    void display();
};
```

```cpp
void stack::push(int n) {
    Node *newNode = new Node; // the conventional way
    //fill data part
    newNode->stu_no = n;
    //link part
    newNode->next = this->top;
    //make newnode as top/head
    this->top = newNode;
}
```

```cpp
void stack::pop() {
    if (this->top == NULL) {
        cout << "List is empty!" << endl;
        return;
    }
    Node *temp;
    cout << top->stu_no << " is removed." << endl;
    temp = top;
    top = top->next;
    delete temp;
}
```

The Easiest Way Using STL

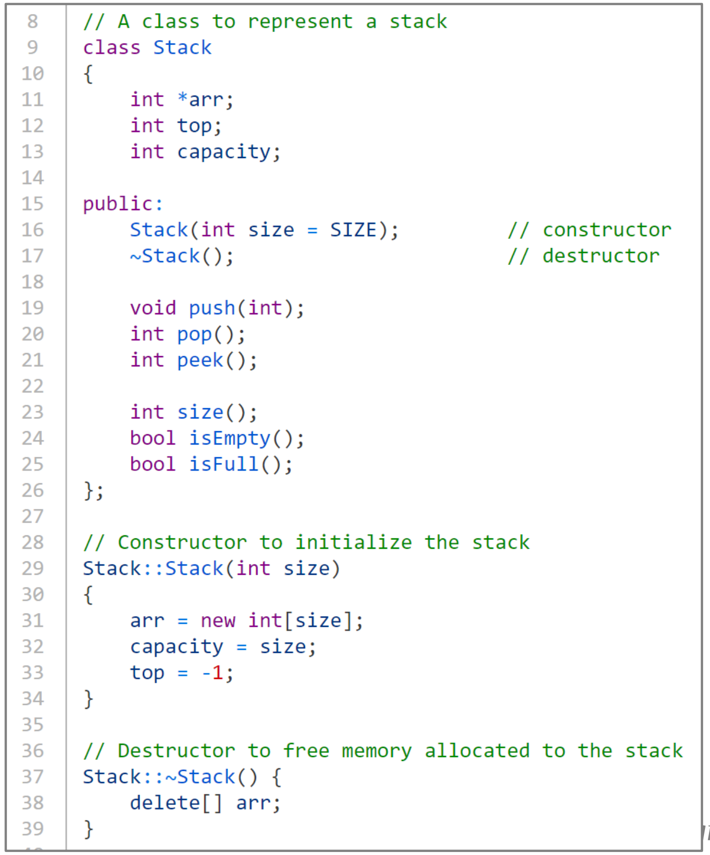
- Code example in geeksforgeeks.org



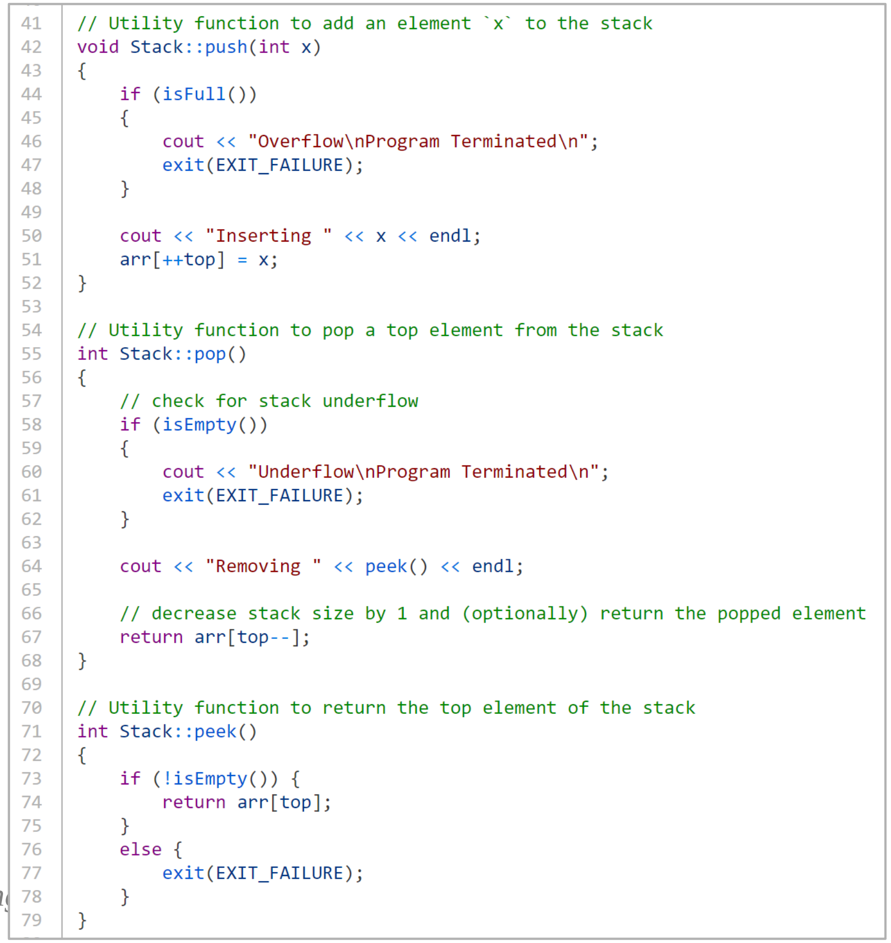
```
#include <iostream>
#include <stack>
using namespace std;
int main() {
    stack<int> stack;
    stack.push(21);
    stack.push(22);
    stack.push(24);
    stack.push(25);

         stack.pop();
    stack.pop();

    while (!stack.empty()) {
        cout << ' ' << stack.top();
        stack.pop();
    }
}
```

Implementation Using an Array

- Example: link



```
// A class to represent a stack
class Stack
{
    int *arr;
    int top;
    int capacity;

public:
    Stack(int size = SIZE);         // constructor
    ~Stack();                       // destructor

    void push(int);
    int pop();
    int peek();

    int size();
    bool isEmpty();
    bool isFull();
};

// Constructor to initialize the stack
Stack::Stack(int size)
{
    arr = new int[size];
    capacity = size;
    top = -1;
}

// Destructor to free memory allocated to the stack
Stack::~Stack() {
    delete[] arr;
}
```



```
// Utility function to add an element `x` to the stack
void Stack::push(int x)
{
    if (isFull())
    {
        cout << "Overflow\nProgram Terminated\n";
        exit(EXIT_FAILURE);
    }

    cout << "Inserting " << x << endl;
    arr[++top] = x;
}

// Utility function to pop a top element from the stack
int Stack::pop()
{
    // check for stack underflow
    if (isEmpty())
    {
        cout << "Underflow\nProgram Terminated\n";
        exit(EXIT_FAILURE);
    }

    cout << "Removing " << peek() << endl;

    // decrease stack size by 1 and (optionally) return the popped element
    return arr[top--];
}

// Utility function to return the top element of the stack
int Stack::peek()
{
    if (!isEmpty()) {
        return arr[top];
    }
    else {
        exit(EXIT_FAILURE);
    }
}
```

A Refined Stack Class

https://onlinegdb.com/tYB_-1RTR

```
struct Node {
    int stu_no;
    char stu_name[50];
    //shared_ptr<Node> next;
    Node *next;
    Node() {
        cout << "A node is created."
             << endl;
    }
    ~Node() {
        cout << "A node is deleted."
             << endl;
    }
};
```

Add a constructor and a destructor of structure `Node`.

Add a constructor and a destructor of class `stack`.

```
class stack {
private:
    Node *top;

public:
    stack() {
        this->top = NULL;
        cout << " # The stack is generated. "
             << endl;
    }
    ~stack() {
        while (this->top != NULL) {
            pop();
        }
        cout << " # The stack is deleted."
             << endl;
    }
    void push(int n, char name[]);
    void pop();
    void display();
};
```

A Refined Stack Class

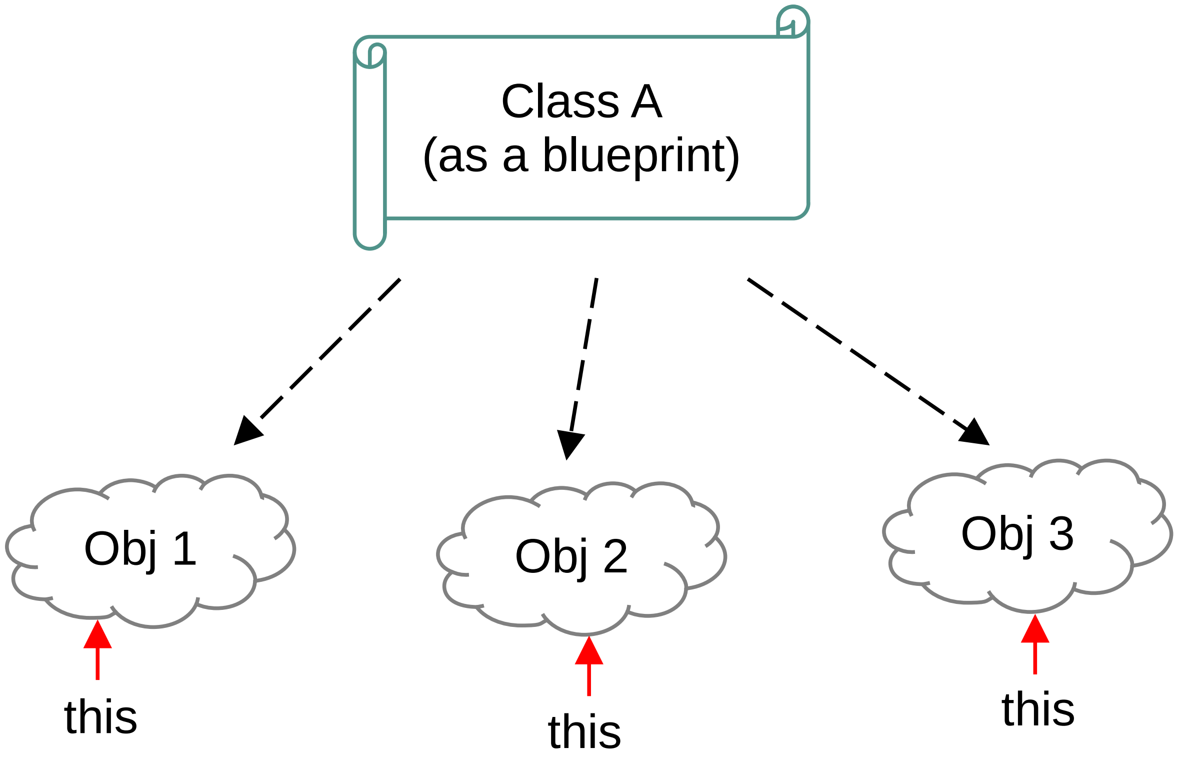
https://onlinegdb.com/tYB_-1RTR

```
void stack::pop() {
    if (this->top == NULL) {
        cout << "List is empty!"
             << endl;
        return;
    }
    Node *temp;
    cout << top->stu_name << " is removed."
        << endl;
    temp = top;
    top = top->next;
    delete temp;
}
```

Note:

We delete each popped element in a stack, and hence the destructor of `Node` is activated.

Some notes on “this pointer”


Class A
(as a blueprint)
Obj 1
Obj 2
Obj 3
this
this
this

Example 1

```
class Demo {
private:
    int value;
public:
    Demo(int value) {
        this->value = value;
    // Using this pointer to refer to
    // the current object
    }
    void display() {
        cout << "Value: " << this->value << endl;
    }
};
```

```
int main() {
    Demo obj(10);
    obj.display(); // 10
    return 0;
}
```

Example 2

```
class Number {
private:
    int num;
public:
    Number(int num) {
        this->num = num;
    }

    Number& setValue(int num) {
        this->num = num;
        return *this; // Returning current object
    }

    void display() {
        cout << "Number: " << num << endl;
    }
};
```

```
int main() {
    Number obj(5);
    obj.setValue(10).display();
    return 0;
}
```

Example 3

```
class Employee {
private:
    string name;
    int age;
public:
    Employee(string name, int age) {
        this->name = name;
        this->age = age;
        // Resolving conflicts!
    }

    void show() {
        cout << "Employee Name: "
              << this->name << "("
              << this->age << ")" << endl;
    }
};
```

```
int main() {
    Employee emp("Alice", 30);
    emp.show(); // Alice(30)
    return 0;
}
```

More examples…

- We will see that in operator overloading again.