

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

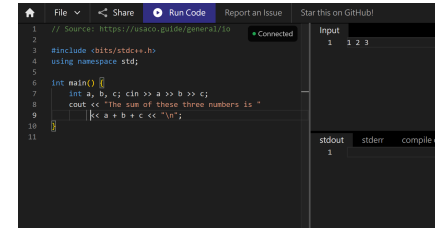


<https://icons8.com/icons/set/code-blocks>

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



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



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

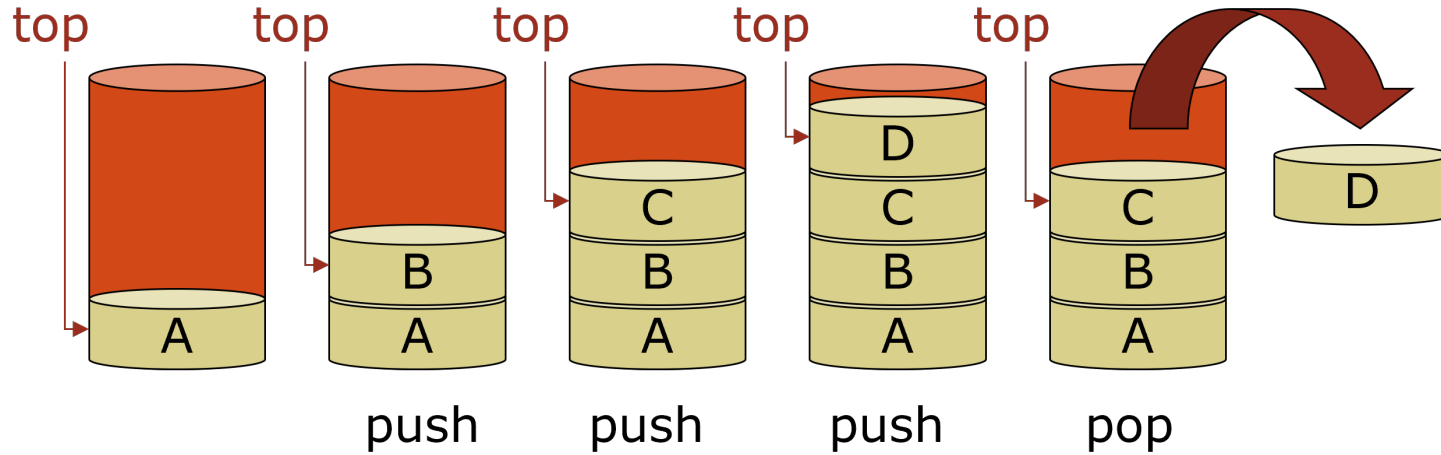
- Tutorialspoint
 - <https://www.tutorialspoint.com/cplusplus/index.htm>
 - Online C++ Compiler
- Programiz
 - <https://www.programiz.com/cpp-programming>
- LEARN C++
 - <https://www.learncpp.com/>
- MIT OpenCourseWare - Introduction to C++
 - <https://ocw.mit.edu/courses/6-096-introduction-to-c-january-iap-2011/pages/lecture-notes/>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

```
6  struct Node {
7      int stu_no;
8      char stu_name[50];
9      //shared_ptr<Node> next;
10     Node *next; // the conventional way
11 };
```

```
13 class stack {
14 private:
15     //shared_ptr<Node> top;
16     Node *top; // the conventional way
17
18 public:
19     stack() {
20         this->top = NULL;
21         cout << " # The stack is generated. " << endl;
22     }
23     ~stack() { cout << " # The stack is deleted." << endl; }
24     void push(int n, char name[]);
25     void pop();
26     void display();
27 };
```

Implementation Using Linked List

- The code in my GitHub page: [link](#)
- Code on OnlineGDB: https://onlinegdb.com/W0S_dJ_k26

```
29 void stack::push(int n, char name[]) {
30     Node *newNode = new Node; // the conventional way
31     //auto newNode = make_shared<Node>();
32     //fill data part
33     newNode->stu_no = n;
34     strcpy(newNode->stu_name, name);
35     //link part
36     newNode->next = this->top;
37     //make newnode as top/head
38     this->top = newNode;
39 }
```

```
41 void stack::pop() {
42     if (this->top == NULL) {
43         cout << "List is empty!" << endl;
44         return;
45     }
46     cout << top->stu_name << " is removed." << endl;
47     top = top->next;
48 }
```


Implementation Using Linked List

```
50 void stack::display() {
51     if (top == NULL) {
52         cout << "List is empty!" << endl;
53         return;
54     }
55     //shared_ptr<Node> temp = this->top;
56     Node *temp = this->top; // the conventional way
57     while (temp != NULL){
58         cout << temp->stu_no << " ";
59         cout << temp->stu_name << " ";
60         cout << endl;
61         temp = temp->next;
62     }
63     cout << endl;
64 }
```

```
66 int main() {
67
68     stack s;
69     char ch;
70     int stu_no;
71     char stu_name[50];
72
73     do {
74         int n;
75
76         cout << "ENTER CHOICE\n"<<"1.Push\n"<<"2.Pop\n"<<"3.Display\n";
77         cout << "Make a choice: ";
78         cin >> n;
79
80         switch(n) {
81             case 1:
82                 cout << "Enter details of the element to be pushed: \n";
83                 cout << "Roll Number: ";
84                 cin >> stu_no;
85                 cout << "Enter Name: ";
86                 std::cin.ignore(1); // to absorb '\n' newline input
87                 cin.getline(stu_name, 50);
88         }
```

A Simplified Version

- https://onlinegdb.com/rQ1j_k3Fiz
- The code in my GitHub page: [link](#)

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

```
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();  
};
```

```
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;  
}
```

```
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](#)

```
8 // A class to represent a stack
9 class Stack
10 {
11     int *arr;
12     int top;
13     int capacity;
14
15 public:
16     Stack(int size = SIZE);           // constructor
17     ~Stack();                         // destructor
18
19     void push(int);
20     int pop();
21     int peek();
22
23     int size();
24     bool isEmpty();
25     bool isFull();
26 };
27
28 // Constructor to initialize the stack
29 Stack::Stack(int size)
30 {
31     arr = new int[size];
32     capacity = size;
33     top = -1;
34 }
35
36 // Destructor to free memory allocated to the stack
37 Stack::~Stack() {
38     delete[] arr;
39 }
```

```
41 // Utility function to add an element `x` to the stack
42 void Stack::push(int x)
43 {
44     if (isFull())
45     {
46         cout << "Overflow\nProgram Terminated\n";
47         exit(EXIT_FAILURE);
48     }
49
50     cout << "Inserting " << x << endl;
51     arr[++top] = x;
52 }
53
54 // Utility function to pop a top element from the stack
55 int Stack::pop()
56 {
57     // check for stack underflow
58     if (isEmpty())
59     {
60         cout << "Underflow\nProgram Terminated\n";
61         exit(EXIT_FAILURE);
62     }
63
64     cout << "Removing " << peek() << endl;
65
66     // decrease stack size by 1 and (optionally) return the popped element
67     return arr[top--];
68 }
69
70 // Utility function to return the top element of the stack
71 int Stack::peek()
72 {
73     if (!isEmpty()) {
74         return arr[top];
75     }
76     else {
77         exit(EXIT_FAILURE);
78     }
79 }
```

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.

C++ Programmin };

```
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

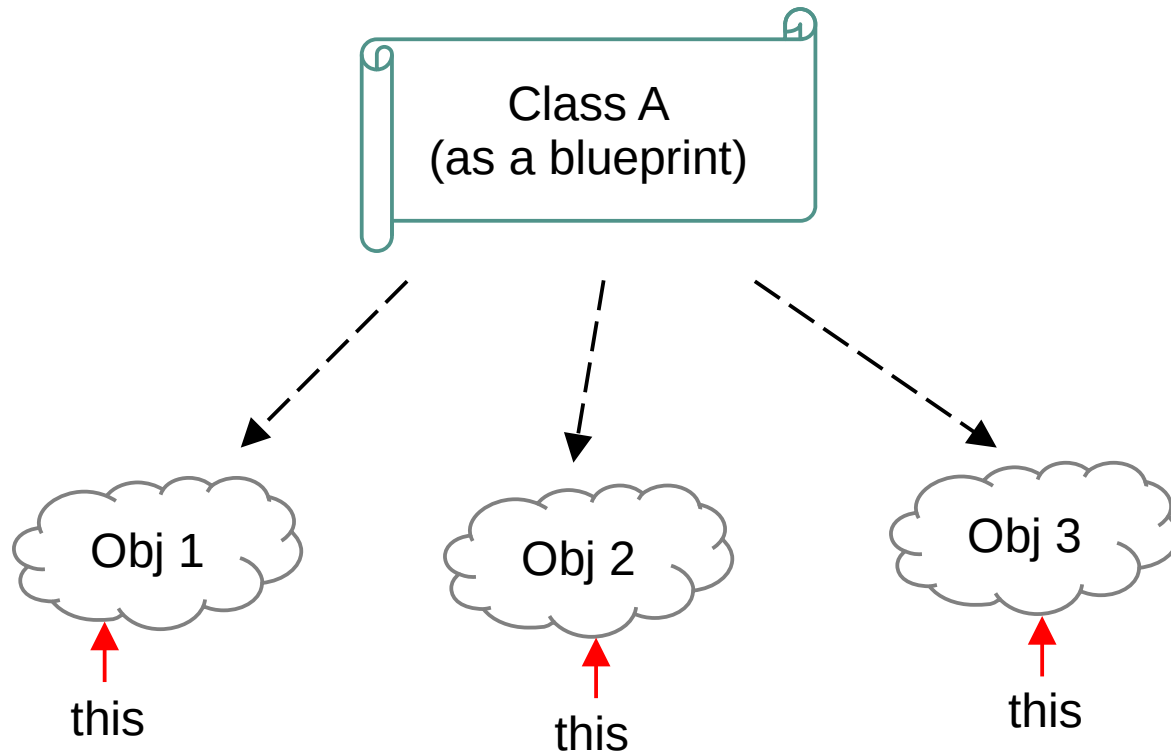
```
void stack::pop() {
    if (this->top == NULL) {
        cout << "List is empty!"
            << endl;
        return;
    }
    Node *temp;
    cout << top->stu_name << " is removed."
        << endl;
    temp = top;
    top = top->next;
    delete temp;
}
```

Note:

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



Some notes on “this pointer”



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;  
}
```

More examples...

- We will see that in operator overloading again.