

C++

# 程式語言（二）

Introduction to Programming (II)

Case Studies & Supplementary

Joseph Chuang-Chieh Lin

Dept. CSE, NTOU

# Platform/IDE

- Dev-C++



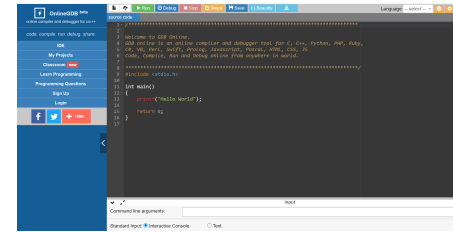
<https://www.pngegg.com/en/search?q=Dev-C>

- Codeblocks

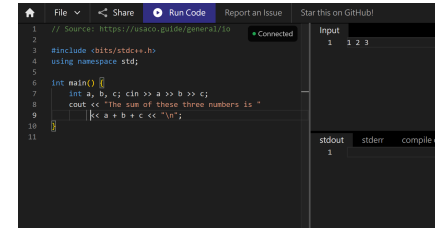


<https://icons8.com/icons/set/code-blocks>

- 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/>

# enum class

- Use `enum class` instead of `enum`.
  - A new feature in C++11.

```
enum ourColor {  
    RED,  
    GREEN,  
    BLUE  
};  
  
ourColor color = RED;
```



```
enum class ourColor {  
    RED,  
    GREEN,  
    BLUE  
};  
  
ourColor color = ourColor::RED;
```

# Previous Issue (I)

<https://kheresy.wordpress.com/2019/03/27/using-enum-class/>

```
#include <iostream>

enum ourColor {
    RED,
    GREEN,
    BLUE
};

enum ourFruit {
    APPLE,
    BANANA
};
```

```
int main() {
    ourColor c1 = RED;
    ourFruit f1 = APPLE;

    if (c1 == f1) {
        cout << "c1 equals f1" << endl;
    } else {
        cout << "c1 and f1 are not equal"

                << endl;
    }
    return 0;
}
```

<https://ide.usaco.guide/OPjyFv9QfAOstBp18V5>

# Previous Issue (I): Compile error

<https://kheresy.wordpress.com/2019/03/27/using-enum-class/>

```
#include <iostream>

enum class ourColor {
    RED,
    GREEN,
    BLUE
};

enum class ourFruit {
    APPLE,
    BANANA
};
```

```
int main() {
    ourColor c1 = ourColor::RED;
    ourFruit f1 = ourFruit::APPLE;

    if (c1 == f1) {
        cout << "c1 equals f1" << endl;
    } else {
        cout << "c1 and f1 are not equal"

                << endl;
    }
    return 0;
}
```

# Previous Issue (II)

<https://kheresy.wordpress.com/2019/03/27/using-enum-class/>

```
#include <iostream>

enum ourColor {
    RED,
    GREEN,
    BLUE
};

enum tLight {
    RED,
    YELLOW,
    Green
};
```

```
int main() {
    ourColor c1 = RED; // redefine; error!
    return 0;
}
```

error: 'RED' conflicts with a previous declaration



# Previous Issue (II)

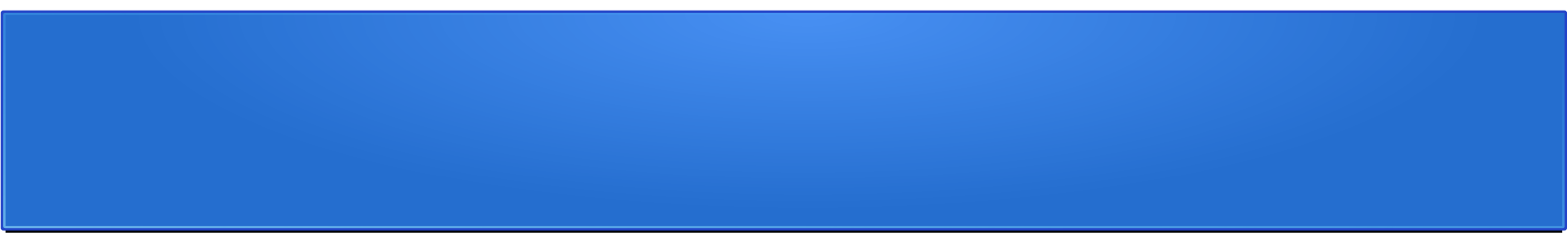
<https://kheresy.wordpress.com/2019/03/27/using-enum-class/>

```
#include <iostream>

enum class ourColor {
    RED,
    GREEN,
    BLUE
};

enum class tLight {
    RED,
    YELLOW,
    Green
};
```

```
int main() {
    ourColor c1 = ourColor::RED; // safe!
    return 0;
}
```



# Case Study (I): Poker Probabilities

# Material refer to C++ For C Programmers (Coursera)

coursera

C++ For  
C Programmers  
Part B

Search in course

Search



Joseph Chuang-C... ▾



C++ For C Programmers, Part B  
University of California, Santa Cruz

## C++ For C Programmers, Part B

by University of California, Santa Cruz

### About this Course

This course is for experienced C programmers who want to program in C++. The examples and exercises require a basic understanding of algorithms and object-oriented software.

### > Course Material

Grades

Notes

Discussion Forums

Messages

Resources

Course Info



**Taught by:** [Ira Pohl](#), Professor  
Computer Science

	<b>Basic Info</b>	Course 4 of 4 in the <a href="#">Coding for Everyone: C and C++ Specialization</a>
	<b>Commitment</b>	5 weeks of study, 2-3 hours/week
	<b>Language</b>	English, <b>Subtitles:</b> Arabic, French, Portuguese (European), Italian, Vietnamese, German, Russian, Spanish <a href="#">Volunteer to translate subtitles for this course</a>
	<b>How To Pass</b>	Pass all graded assignments to complete the course.

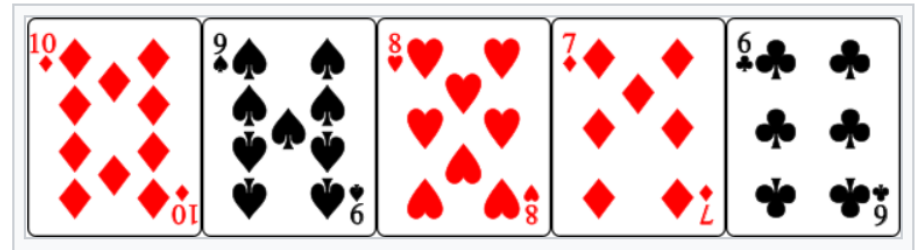


<https://www.coursera.org/learn/c-plus-plus-b/home/info>

wan

# Project: Card Probability

- flush
- straight



**What is the probability of a random shuffle having a flush, straight or a straight-flush?**

# The Refined Code on OnlineGDB

- [https://onlinegdb.com/D5mm\\_YxfA](https://onlinegdb.com/D5mm_YxfA)

## Header files:

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <assert.h>
#include <vector>
#include <algorithm>
```

# Suits & Pips

```
// suits
enum class suit: short {
    SPADE, HEART, DIAMOND, CLUB
};
```

```
class pips {
public:
    pips(int val): v(val) { assert(v>0 && v<14); }
    friend ostream& operator<<(ostream& out, const pips& p);
    int get_pips() { return v; }
private:
    int v;
};
```

# Card

```
class card {  
public:  
    card(): s(suit::SPADE), v(1) {}  
    card(suit st, pips pv): s(st), v(pv) {}  
    friend ostream& operator<<(ostream& out, const card& c);  
    suit get_suit() { return s; }  
    pips get_pips() { return v; }  
private:  
    suit s;  
    pips v;  
};
```

# Ostream << overloading

```
ostream& operator<<(ostream& os, const suit& s) {  
    os << static_cast<std::underlying_type<suit>::type>(s);  
    return os;  
}  
  
ostream& operator<<(ostream& os, const pips& p) {  
    os << p.v;  
    return os;  
}  
  
ostream& operator<<(ostream& os, const card& c) {  
    os << "pips: " << c.v << "suit: " << c.s << endl;  
    return os;  
}
```

**Note:** enum class doesn't allow implicit conversion to integer, you must `static_cast` it.



# Initialization of the deck & Print

```
void init_deck(vector<card> & d) {  
    int i;  
    for (i=1; i<14; i++) {  
        card c(suit::SPADE, i);  
        d[i-1] = c;  
    }  
    for (i=1; i<14; i++) {  
        card c(suit::HEART, i);  
        d[i+12] = c;  
    }  
    for (i=1; i<14; i++) {  
        card c(suit::DIAMOND, i);  
        d[i+25] = c;  
    }  
    for (i=1; i<14; i++) {  
        card c(suit::CLUB, i);  
        d[i+38] = c;  
    }  
}
```

```
void print(vector<card> &deck) {  
    for (auto p=deck.begin(); p!=deck.end(); ++p) {  
        // for (auto card_val: deck) cout << card_val  
        cout << *p;  
    }  
    cout << endl;  
}
```

# Check if the deck is a flush

```
bool is_flush(vector<card> &hand) {  
    suit s = hand[0].get_suit();  
    for (auto p=hand.begin(); p!=hand.end(); ++p) {  
        if (s != p->get_suit()) {  
            return false;  
        }  
    }  
    return true;  
}
```

# Check if the deck is a straight

```
bool is_straight(vector<card> &hand) {  
    int pips_v[5];  
    int i = 0;  
    for (auto p=hand.begin(); p!=hand.end(); ++p) {  
        pips_v[i++] = (p->get_pips()).get_pips();  
    }  
    sort(pips_v, pips_v+5); // feed the range for the iterator  
    if (pips_v[0] != 1) { // not ACE  
        return (pips_v[0] == pips_v[1]-1 && pips_v[1] == pips_v[2]-1)  
            && (pips_v[2] == pips_v[3]-1 && pips_v[3] == pips_v[4]-1);  
    } else {  
        return (pips_v[0] == pips_v[1]-1 && pips_v[1] == pips_v[2]-1)  
            && (pips_v[2] == pips_v[3]-1 && pips_v[3] == pips_v[4]-1)  
            || (pips_v[1] == 10) && (pips_v[2] == 11) && (pips_v[3] == 12)  
            && (pips_v[4] == 13);  
    }  
}
```

# Straight and Flush

```
bool is_straight_flush(vector<card> &hand) {  
    return is_flush(hand) && is_straight(hand);  
}
```

# Main Function

```
vector<card> deck(52);
srand(time(0));
init_deck(deck);
int num_shuffles;
int flush_count = 0;
int str_count = 0;
int str_flush_count = 0;
cout << "How many shuffles? ";
cin >> num_shuffles;
```

```
for (int loop=0; loop<num_shuffles; ++loop) {
    random_shuffle(deck.begin(), deck.end());
    vector<card> hand(5);
    int i=0;
    for (auto p=deck.begin(); i<5; ++p) {
        hand[i++] = *p;
    }
    if (is_flush(hand)) {
        flush_count++;
    }
    if (is_straight(hand)) {
        str_count++;
    }
    if (is_straight_flush(hand)) {
        str_flush_count++;
    }
}
```

```
cout << "Flushes: " << flush_count << " out of " << num_shuffles << endl;
cout << "Straights: " << str_count << " out of " << num_shuffles << endl;
cout << "Straight Flushes: " << str_flush_count
    << " out of " << num_shuffles << endl;
```

# C++ algorithms library

- <https://en.cppreference.com/w/cpp/algorithm>

# Notices about `sort()`

- Arrange the elements in the range from `XXX.begin()` up to but not including `XXX.end()` in ascending order.
- The `sort()` algorithm requires its two iterator arguments to be random-access iterators.
  - Available data types or containers: built-in arrays and STL containers **array**, **vector** and **deque**.

# Revisit to `is_straight()`

```
bool is_straight(vector<card> &hand) {  
    int pips_v[5];  
    int i = 0;  
    for (auto p=hand.begin(); p!=hand.end(); ++p) {  
        pips_v[i++] = (p->get_pips()).get_pips();  
    }  
    sort(pips_v, pips_v+5); // feed the range for the iterator  
    if (pips_v[0] != 1) { // not ACE  
        return (pips_v[0] == pips_v[1]-1 && pips_v[1] == pips_v[2]-1)  
            && (pips_v[2] == pips_v[3]-1 && pips_v[3] == pips_v[4]-1);  
    } else {  
        return (pips_v[0] == pips_v[1]-1 && pips_v[1] == pips_v[2]-1)  
            && (pips_v[2] == pips_v[3]-1 && pips_v[3] == pips_v[4]-1)  
            || (pips_v[1] == 10) && (pips_v[2] == 11) && (pips_v[3] == 12)  
            && (pips_v[4] == 13);  
    }  
}
```



# Using a lambda expression

```
bool is_straight(vector<card> &hand) {  
    sort(pips_v, pips_v+5, [](const card& a, const card& b)  
        { return (a.get_pips()).get_pips() < (b.get_pips()).get_pips(); } );  
    int pips_v[5];  
    int i = 0;  
    for (auto p=hand.begin(); p!=hand.end(); ++p) {  
        pips_v[i++] = (p->get_pips()).get_pips();  
    }  
    if (pips_v[0] != 1) { // not ACE  
        return (pips_v[0] == pips_v[1]-1 && pips_v[1] == pips_v[2]-1)  
            && (pips_v[2] == pips_v[3]-1 && pips_v[3] == pips_v[4]-1);  
    } else {  
        return (pips_v[0] == pips_v[1]-1 && pips_v[1] == pips_v[2]-1)  
            && (pips_v[2] == pips_v[3]-1 && pips_v[3] == pips_v[4]-1)  
            || (pips_v[1] == 10) && (pips_v[2] == 11) && (pips_v[3] == 12)  
            && (pips_v[4] == 13);  
    }  
}
```

# Recall: Lambda Expression in C++

- In C++11 and later, a **lambda expression** is a convenient way of defining an anonymous function object *right at the location* where it's invoked or passed as an argument to a function.
  - Especially when it's not going to be reuse and not worth naming.

```
[ capture clause ] (parameters) -> return-type  
{  
    definition of method  
}
```

<https://www.geeksforgeeks.org/lambda-expression-in-c/>

<https://blog.gtwang.org/programming/lambda-expression-in-c11/>

# Modified Poker Probability Code

- <https://onlinegdb.com/vdGeAd2QIg>

# Card (Modified...)

```
class card {  
public:  
    card(): s(suit::SPADE), v(1) {}  
    card(suit st, pips pv): s(st), v(pv) {}  
    friend ostream& operator<<(ostream& out, const card& c);  
    suit get_suit() const { return s; }  
    pips get_pips() const { return v; }  
private:  
    suit s;  
    pips v;  
};
```

Where did I find the solution? => <https://tinyurl.com/cdhm48af>

# Exercise

- Add a function:

```
bool is_straight_flush(vector<card> &hand)
```

to the program <https://www.onlinegdb.com/vdGeAd2QIg>  
to compute the number of fullhouses.

- Sample output:

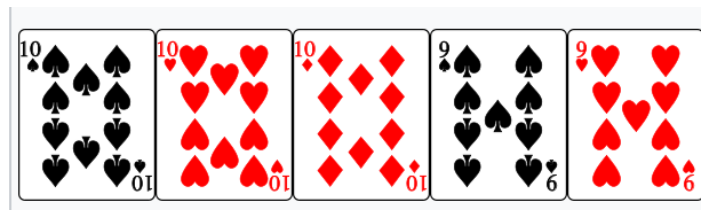
```
Flushes: 3 out of 1000  
Straights: 6 out of 1000  
Straight Flushes: 0 out of 1000  
Fullhouses: 4 out of 1000
```

# Full House

- From Wikipedia:

## Full house [\[ edit \]](#)

A **full house**, also known as a *full boat* or a *tight* or a *boat* (and originally called a **full hand**), is a hand that contains three cards of one rank and two cards of another rank, such as 3♣ 3♠ 3♦ 6♣ 6♥ (a "full house, threes over sixes" or "threes full of sixes" or "threes full").<sup>[17][18]</sup> It ranks below four of a kind and above a flush.<sup>[5]</sup>

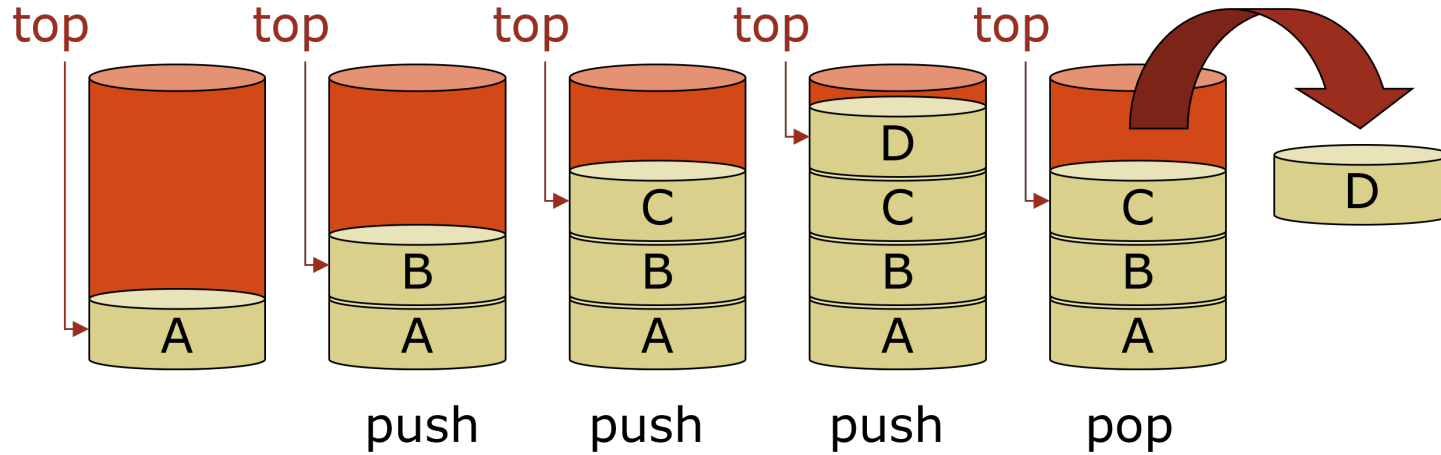




# Case Study (II): Stack

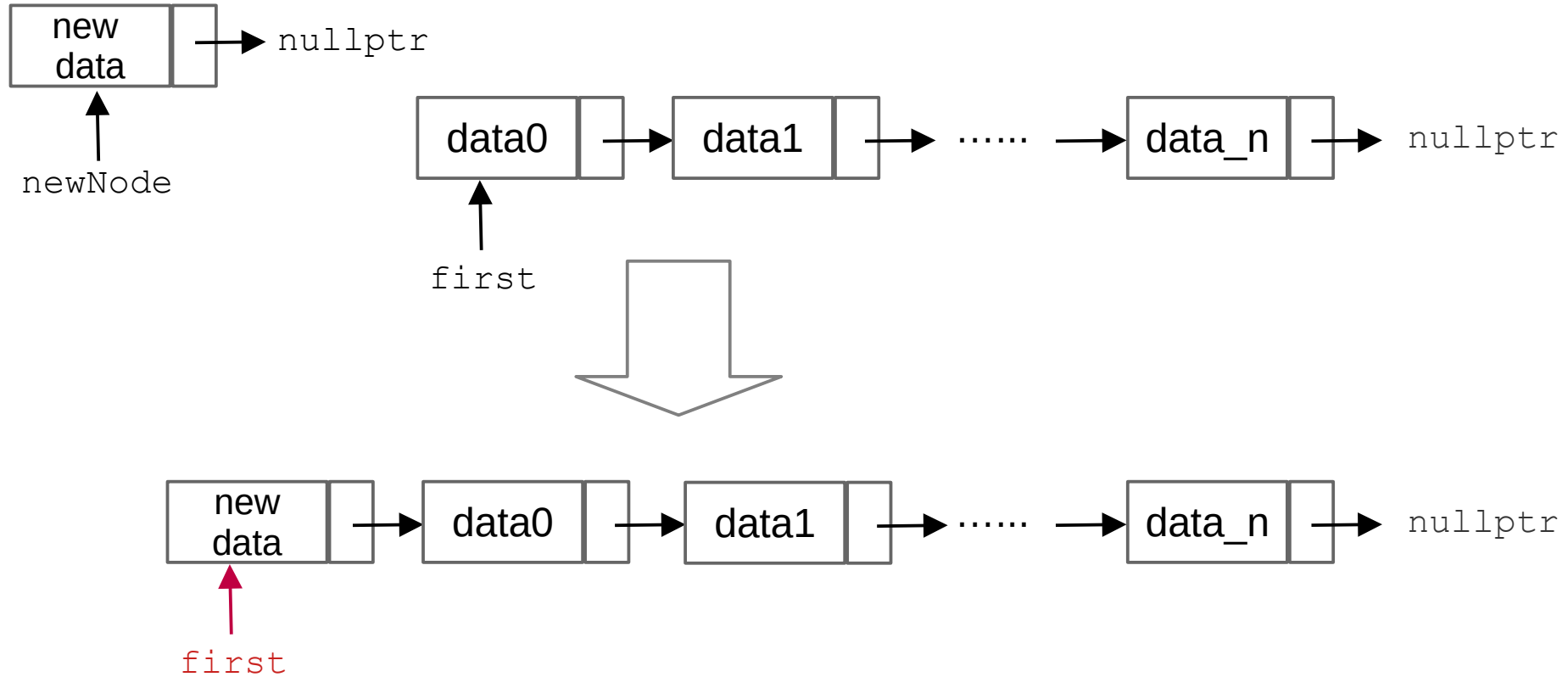
# Stack

- LIFO: Last In, First Out

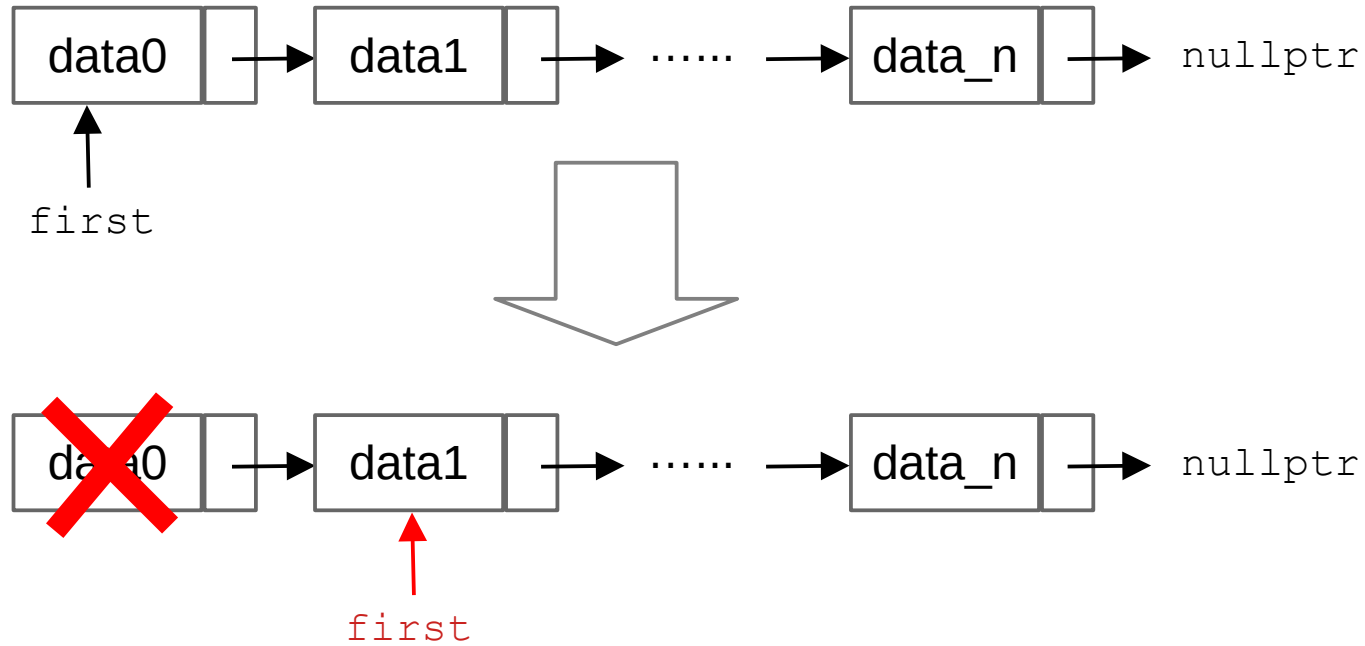




# Push ()



# Pop ()



# Implementation Using Linked List

- The code in my GitHub page: [link](#)
- Code on OnlineGDB: <https://onlinegdb.com/7itcr--jP>

```
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

- The code in my GitHub page: [link](#)
- Code on OnlineGDB: <https://onlinegdb.com/7itcr--jP>

```
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/7itcr--jP>

```
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](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

- A code example:

```
#include <iostream>
#include <stack>    // Include the stack container
using namespace std;

int main() {
    // Create a stack of integers
    stack<int> myStack;

    // Push elements onto the stack
    myStack.push(21);
    myStack.push(22);
    myStack.push(24);
    myStack.push(25);

    // Pop the top two elements
    myStack.pop();
    myStack.pop();

    // Print and pop each remaining element until the stack is empty
    while (!myStack.empty()) {
        cout << ' ' << myStack.top();
        myStack.pop();
    }

    cout << endl;
    return 0;
}
```



# 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](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](https://onlinegdb.com/tYB_-1RTR)

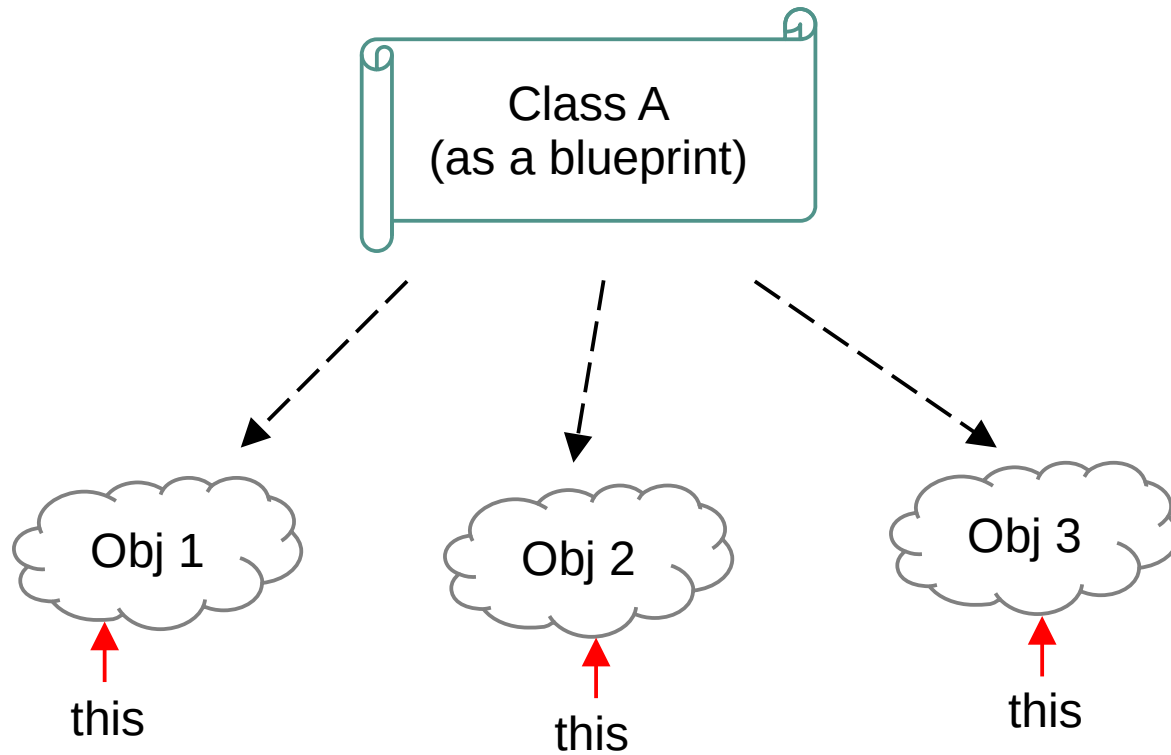
```
void stack::pop() {
    if (this->top == NULL) {
        cout << "List is empty!"
            << endl;
        return;
    }
    Node *temp;
    cout << top->stu_name << " is removed."
        << endl;
    temp = top;
    top = top->next;
    delete temp;
}
```

## 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;  
}
```

# 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;  
}
```





# Discussions & Questions