C++

# 程式語言（二）
# Introduction to Programming (II)

## Case Studies & Supplementary

Joseph Chuang-Chieh Lin

Dept. CSE, NTOU

## Platform/IDE

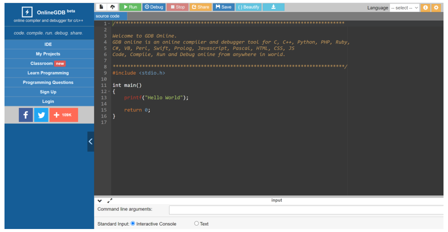
- Dev-C++

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

- Codeblocks

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

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

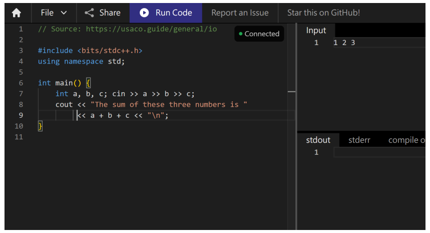


```
/******************************************************************************

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";
}

```

```
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/

## enum class

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

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

ourColor color = RED;
```

$\Rightarrow$

```
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/

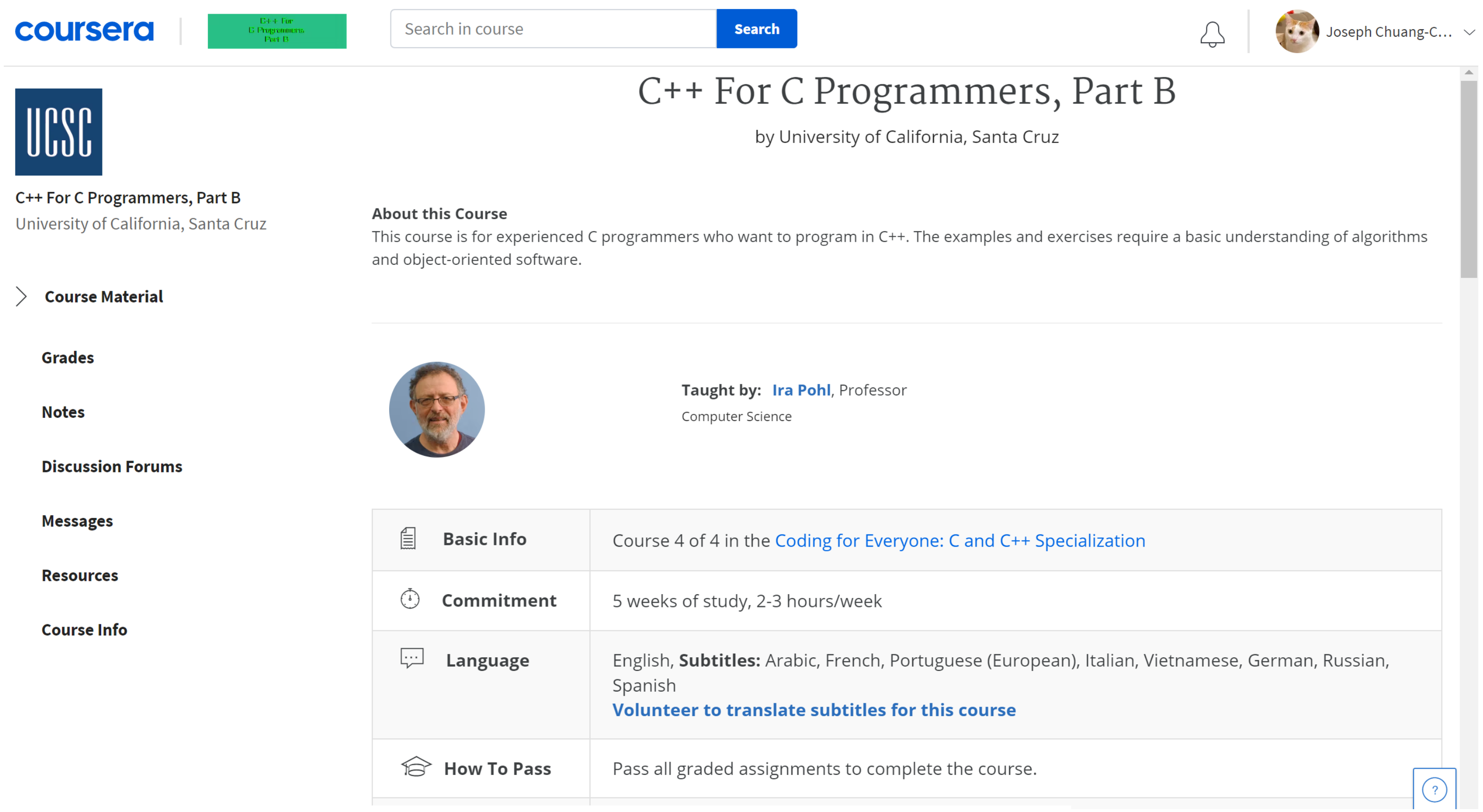
```
#include <iostream>

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)



C++ For C Programmers, Part B

University of California, Santa Cruz

- Course Material
- Grades
- Notes
- Discussion Forums
- Messages
- Resources
- Course Info

## 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.

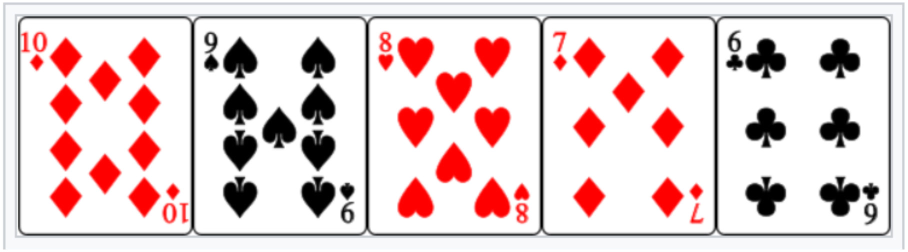
**Taught by:** Ira Pohl, Professor
Computer Science

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

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

## Project: Card Probability

- flush
- straight



https://en.wikipedia.org/wiki/List_of_poker_hands

**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

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;
};
```

Reference for 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

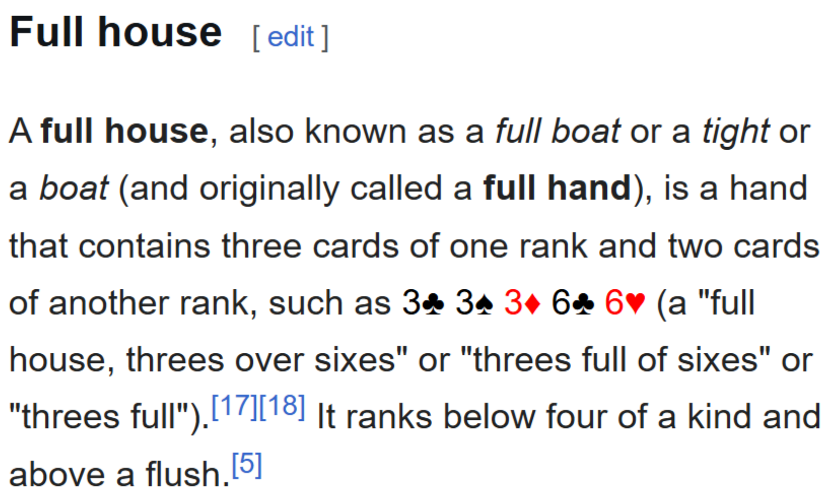
  to compute the number of fullhouses.

- Sample output:

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

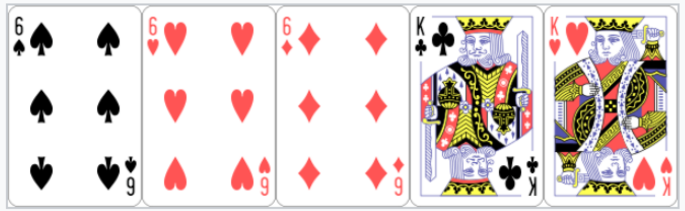
## Full House

- From Wikipedia:

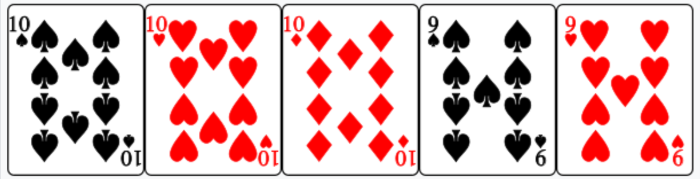


**Full house** [ edit ]

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").[17][18] It ranks below four of a kind and above a flush.[5]



A full house, sixes over kings

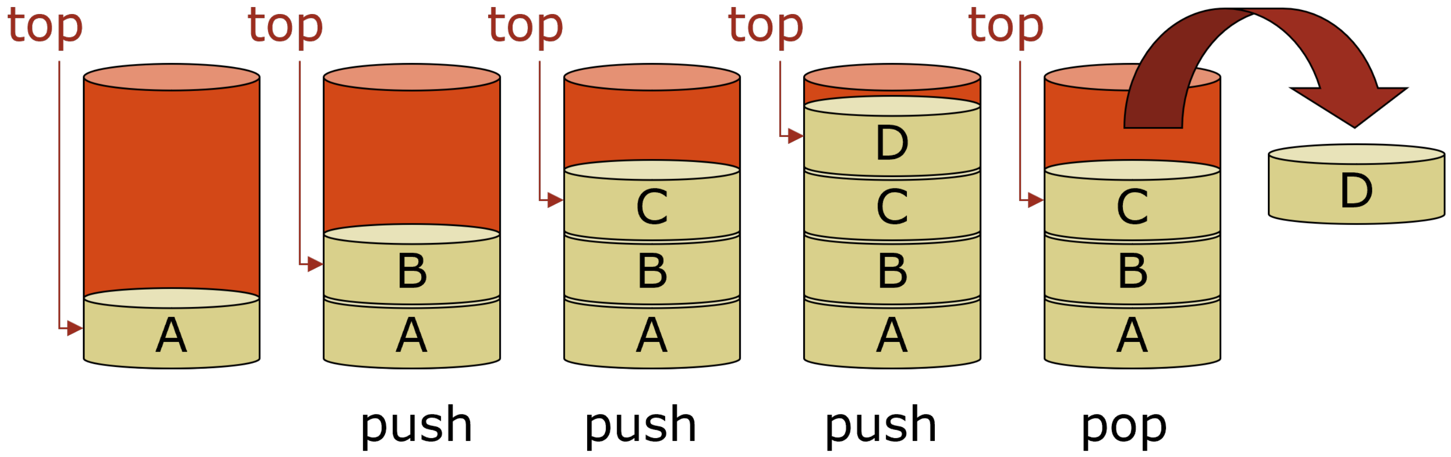


## Case Study (II):

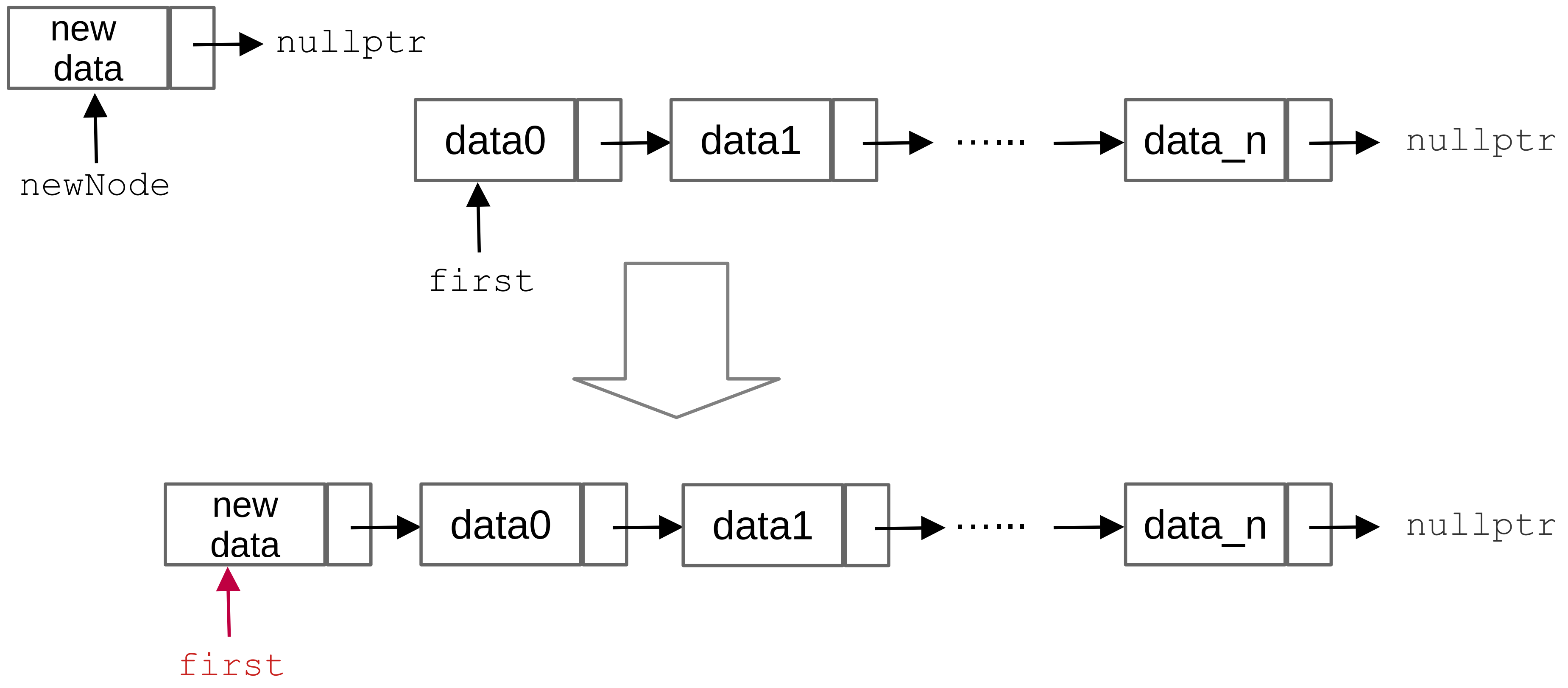
## Stack

## Stack

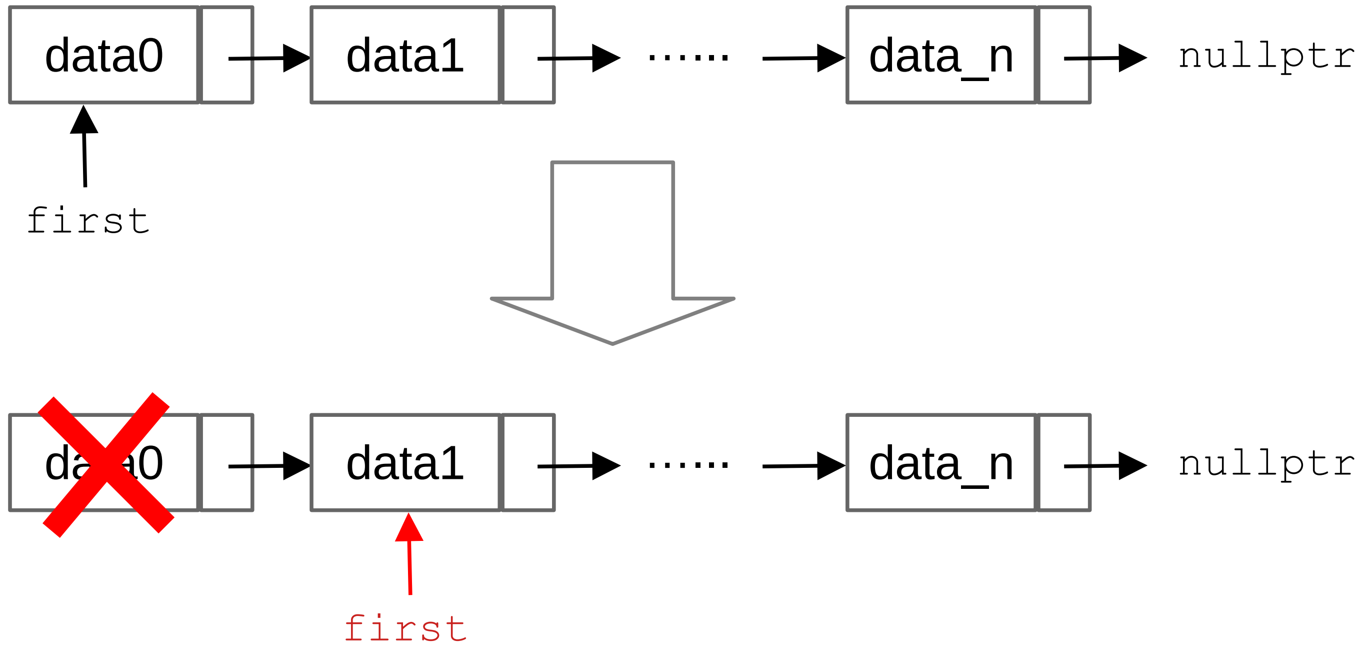
- LIFO: Last In, First Out



## Push()

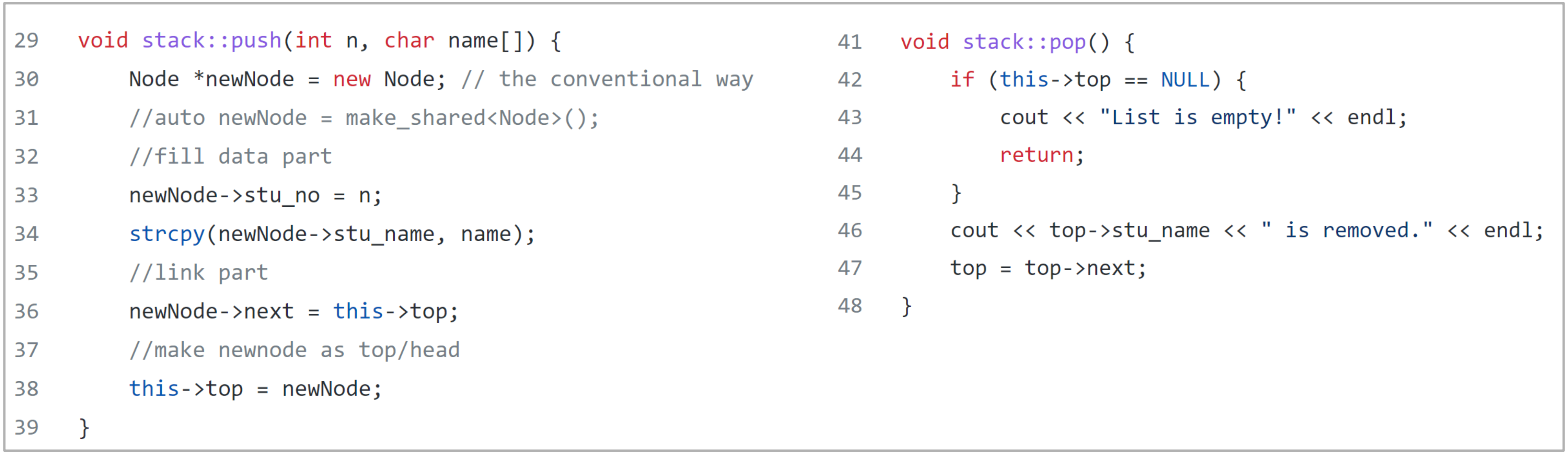

new data
nullptr
newNode
data0
data1
……
data_n
nullptr
first
new data
data0
data1
……
data_n
nullptr
first

# Pop()


data0
data1
……
data_n
nullptr
first
data0
data1
……
data_n
nullptr
first

## Implementation Using Linked List

- The code in my GitHub page: link
- Code on OnlineGDB: https://onlinegdb.com/7itcr--jP



```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

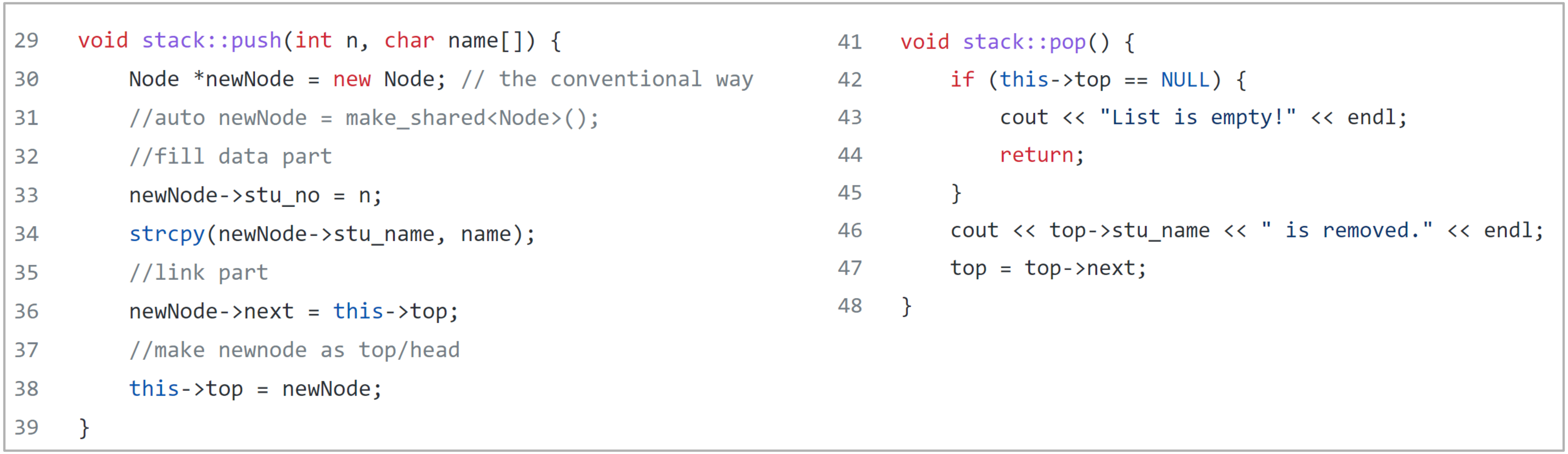
- The code in my GitHub page: link
- Code on OnlineGDB: https://onlinegdb.com/7itcr--jP

```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

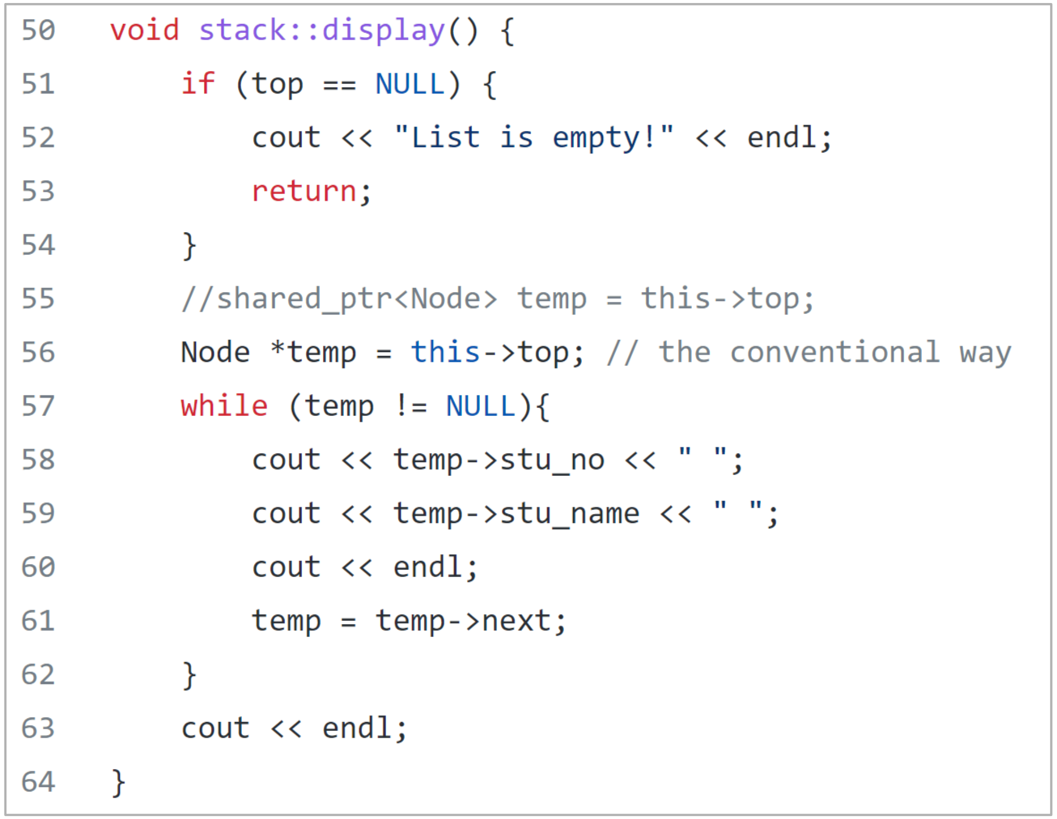
- The code in my GitHub page: link
- Code on OnlineGDB: https://onlinegdb.com/7itcr--jP



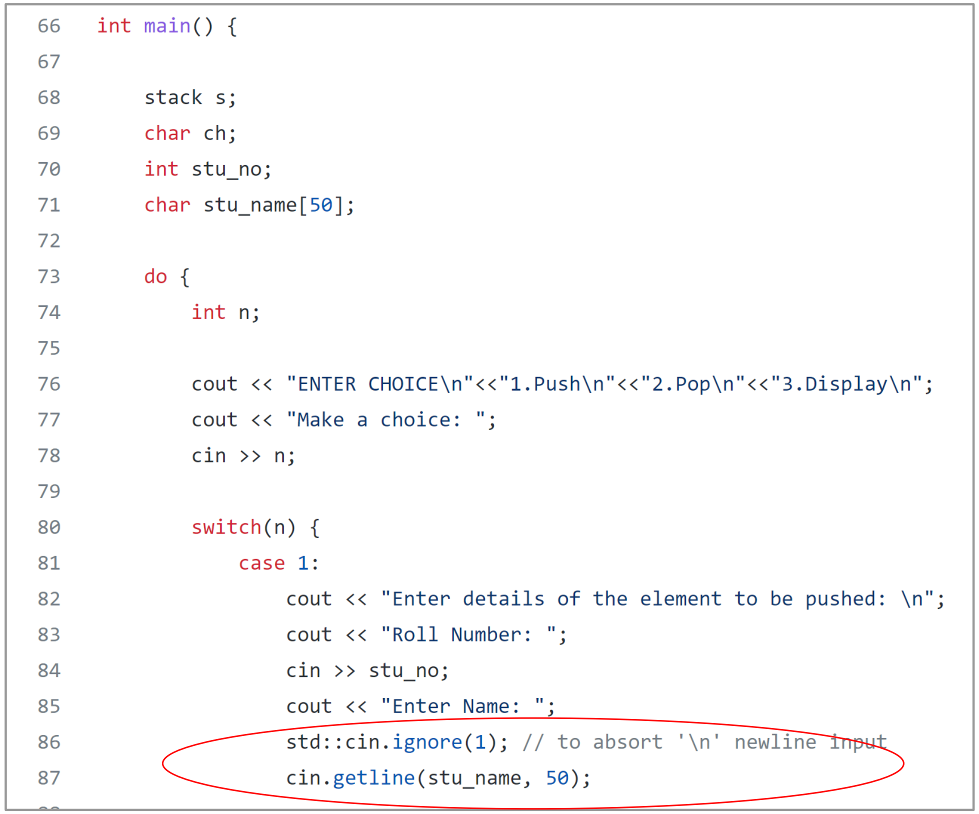
```cpp
void stack::push(int n, char name[]) {
    Node *newNode = new Node; // the conventional way
    //auto newNode = make_shared<Node>();
    //fill data part
    newNode->stu_no = n;
    strcpy(newNode->stu_name, name);
    //link part
    newNode->next = this->top;
    //make newnode as top/head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

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

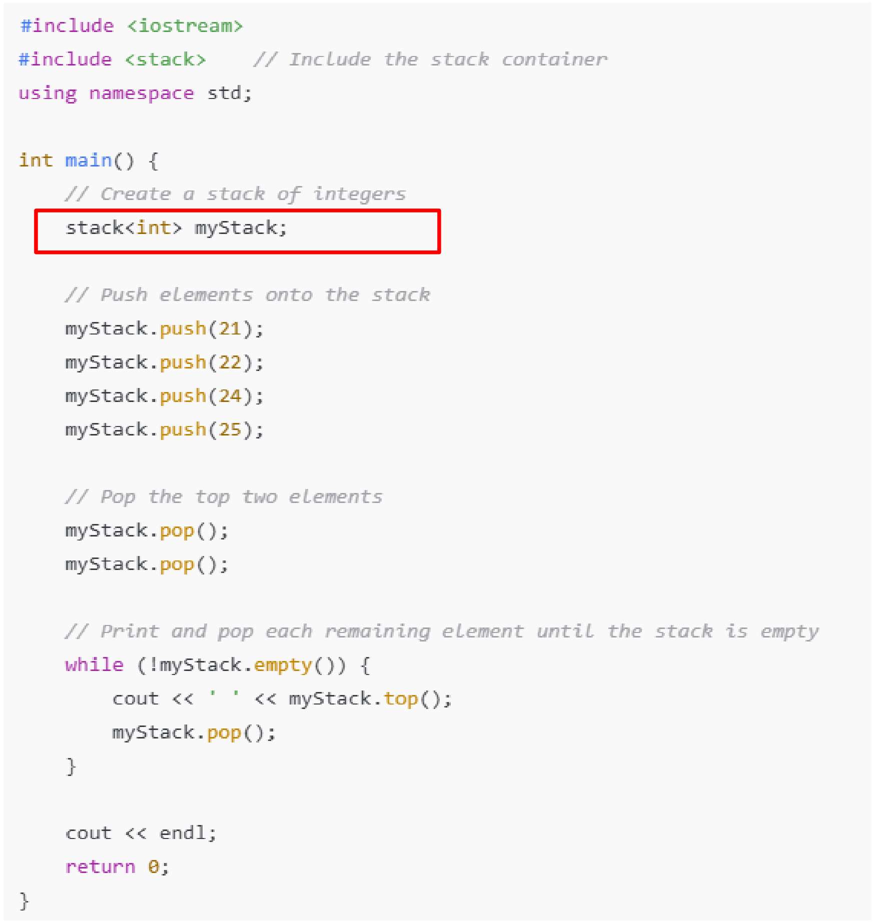
```
class stack {
private:
    Node *top; // the conventional way

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

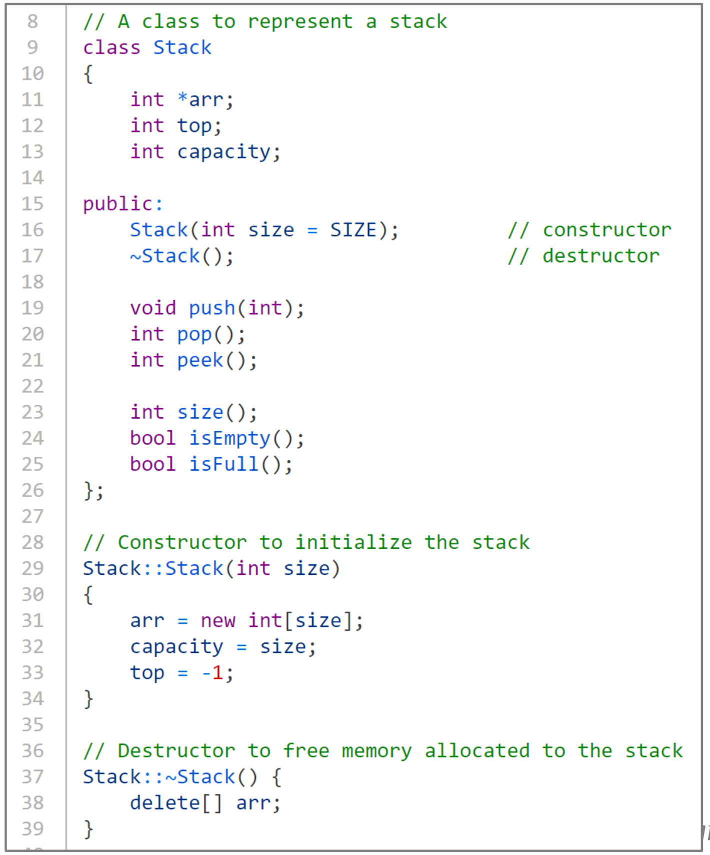
- A code example:



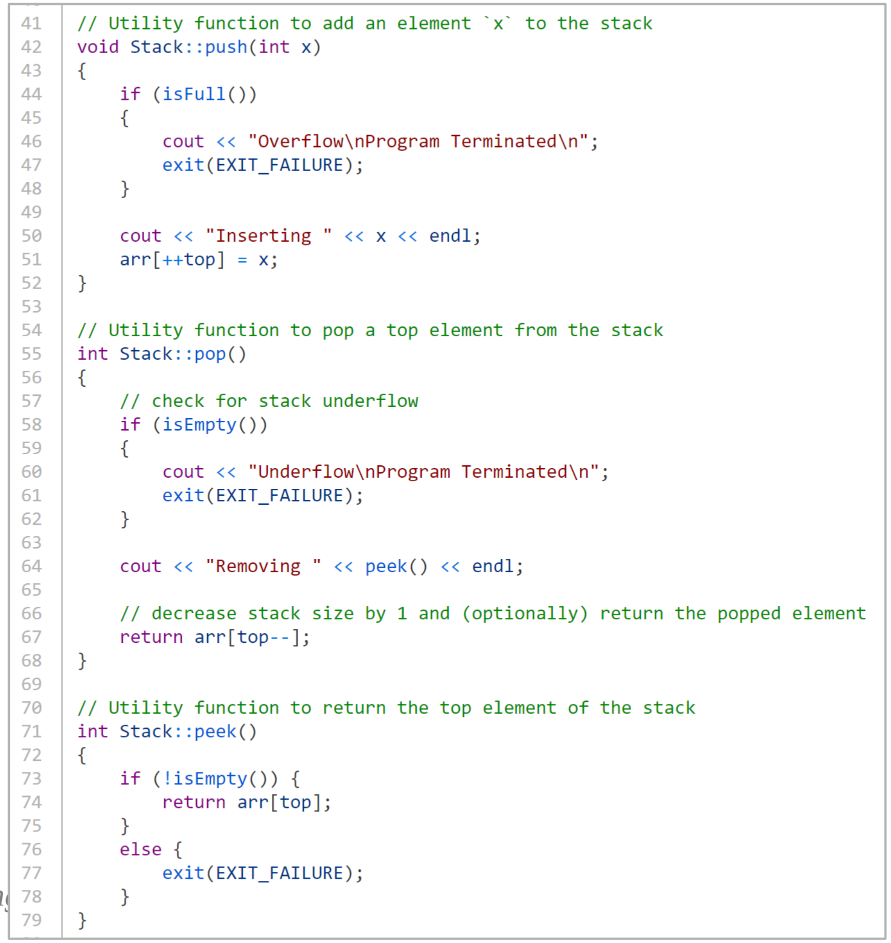
```
#include <iostream>
#include <stack>    // Include the stack container
using namespace std;

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



```cpp
// 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;
}
```



```cpp
// 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

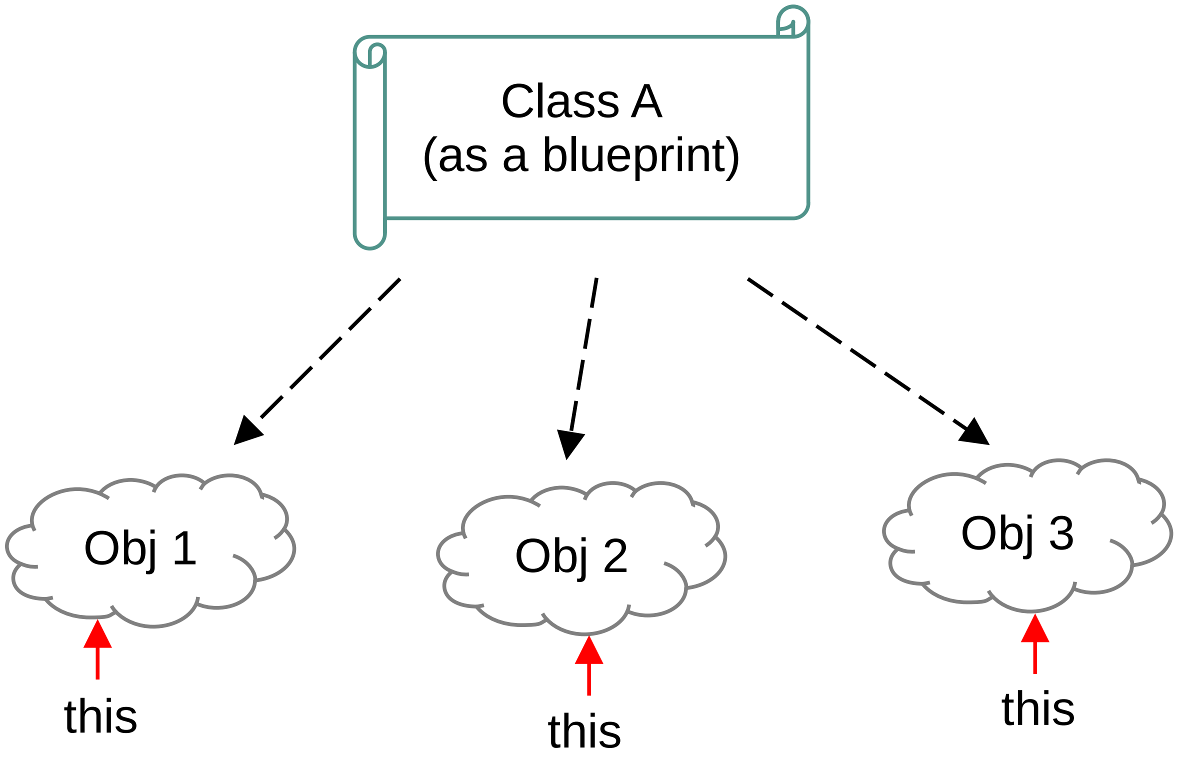
https://onlinegdb.com/tYB_-1RTR

```
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;
}
```

## Case Study (III):

## Random Number Generation

## `rand()`

- Required header: `cstdlib`
- It generates an unsigned integer between 0 and `RAND_MAX`.
  - `RAND_MAX` is defined in `cstdlib`.
  - Every number in the range is chosen with equal chance when `rand` is called each time.
- `rand()` actually generates pseudo-random numbers.

# Rolling a Six-Sided Die

```
#include <iostream>
#include <iomanip> // for setw(); setting width of output
#include <cstdlib>

using namespace std;

int main() {
    for (unsigned int counter = 1; counter <= 20; ++counter) {
        // pick random number from 1 to 6 and output it
        cout << setw(10) << (1+rand()%6);
        // if counter is divisible by 5, start a new line
        if ( counter % 5 == 0 )
            cout << endl;
    } // end for
}
```

Try to play around the code by yourself.

## Adding 'Seeds'

- `srand()`:
  - In `cstdlib` header.
  - A function takes an unsigned integer argument and seeds the rand function to produce **different** sequence of random numbers for each execution.
- Why "seed" matters?
  - **Reproducibility**: replay the same result of `rand()`.
  - **Variability:** `srand(time(nullptr));` => your seed changes every second.

# Rolling a Six-Sided Die **with Seeds**

```
#include <iostream>
#include <iomanip> // for setw(); setting width of output
#include <cstdlib>

using namespace std;

int main() {
    unsigned int seed = 0;
    cout << "Enter seed: ";
    cin >> seed;
    srand(seed); // seed random number generator
    for (unsigned int counter = 1; counter <= 20; ++counter) {
        // pick random number from 1 to 6 and output it
        cout << setw(10) << (1+rand()%6);
        // if counter is divisible by 5, start a new line
        if (counter%5 == 0)
            cout << endl;
    }
}
```

Try to play around the code by yourself.

## Supplementary:
### Sketch of the mechanism in the hindsight

```
static unsigned long next = 1;  // default seed if you never call srand()

void srand(unsigned int seed) { // set the hidden next = seed
    next = seed;                // re-initialize the state
}

int rand(void) {
    next = next * 1103515245 + 12345;
    // return the high-order bits, in [0, RAND_MAX]
    return (unsigned int)(next / 65536) % 32768;
}
```

## C++ 11 Random Numbers

- According CERT, `rand()` does not have good statistical properties and can be predictable.
- C++11 provides a more secure library of random-number capabilities that can’t predicted.
  - Located in the **`random`** header.
- C++11 provides many classes that represent various random number generation *engines* and *distributions*.
  - An engine implements a random-number generation **algorithm** that produce pseudo-random numbers.
  - A distribution controls the **range** of values produced by an engine, the **types** of those values and the **statistical properties** of the values.

## Example

- We consider the default engine and uniform distribution as the example.
  - `default_random_engine`
  - `uniform_int_distribution`

## Rolling a Six-Sided Die (C++11 random)

```
#include <iostream>
#include <iomanip> // for setw(); setting width of output
#include <random> // C++11 random number generation features
#include <ctime> // for time() function

using namespace std;

int main() {
    default_random_engine engine(static_cast<unsigned int>(time(0)));
    uniform_int_distribution<unsigned int> randomInt(1,6);

    for (unsigned int counter = 1; counter <= 20; ++counter) {
        cout << setw(10) << randomInt(engine);
        if (counter%5 == 0)
            cout << endl;
    }
}
```

Try to play around the code by yourself.

# Normal Distribution

```
#include <iostream>
#include <random> // C++11 random number generation features
#include <cmath> // for using lround; rounding a real number
#include <vector>

using namespace std;

int main() {
    default_random_engine e;
    normal_distribution<> normal(4, 1.5); // using default type: double
    vector<unsigned> vals(9); // a vector of 9 0's
    for (size_t i=0; i<100; i++) {
        unsigned v = lround(normal(e));
        if (v < vals.size())
            ++vals[v];
    }
    for (size_t j=0; j<vals.size(); j++)
        cout << j << ": " << string(vals[j], '*') << endl;

    return 0;
}
```

```
0: **
1: ***
2: *********
3: *****************
4: *********************************
5: ********************
6: *************
7: ***
8:
```

Discussions & Questions