物件導向程式設計

Case Study: Stack

Joseph Chuang-Chieh Lin
Dept. CSIE, Tamkang University

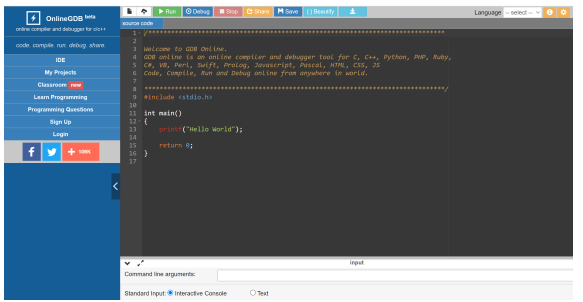
Platform

- Dev-C++

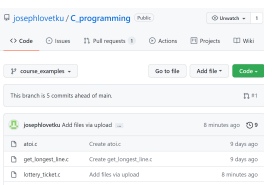
  Click here to download.

  Note: Please use this version otherwise you can't compile your programs/projects in Win10.

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



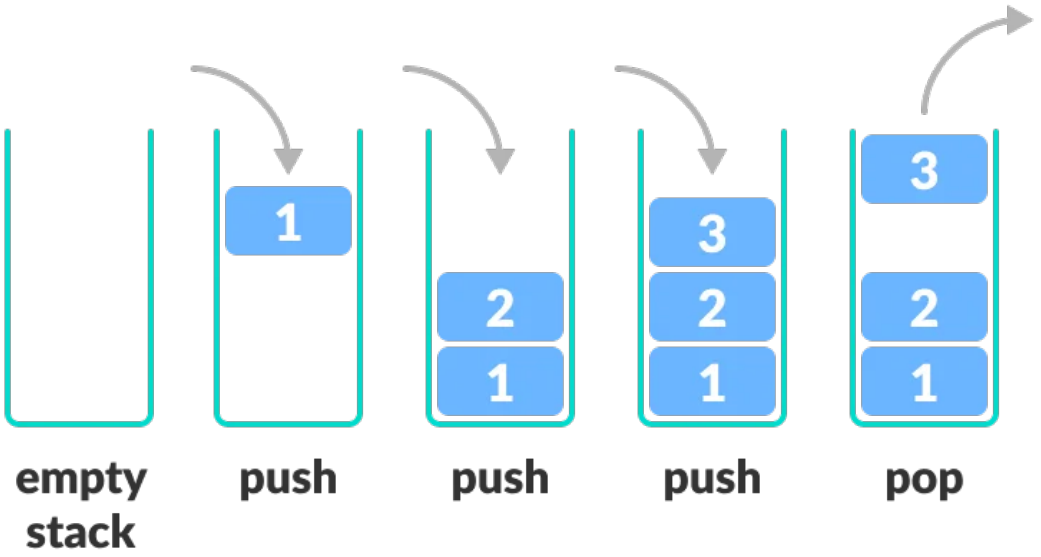
My GitHub page:
click the link here to visit.



- Other resources:
  - MIT OpenCourseWare - Introduction to C++ [link].
  - Learning C++ Programming [Programiz].
  - GeeksforGeeks [link]

Stack

- LIFO: Last In, First Out



Refer to https://www.programiz.com/dsa/stack

Implementation Using Linked List

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

```
 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/nERaZRsamB

```
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 absort '\n' newline input
87                  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

- Code example in geeksforgeeks.org

```cpp
#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  }
```

Exercise (3%)

```
struct Node {
    string stu_no;
    int math;
    int english;
    int computer;
    Node *next;
};
```

```
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(string s, int m, int e, int c);
    void pop();
    void display();
};
```

Main()

```
int main() {
    stack st;
    string stu_no;
    int math, eng, cs;
    char c;

    cout << "push an item: ";
         cin >> stu_no >> math >> eng >> cs;
    st.push(stu_no, math, eng, cs);
    cout << "push an item: ";
    cin >> stu_no >> math >> eng >> cs;
    st.push(stu_no, math, eng, cs);
    cout << "push an item: ";
    cin >> stu_no >> math >> eng >> cs;
    st.push(stu_no, math, eng, cs);
    //show the stack content
    st.display();

    cout << "Press any key to continue..."
         << endl;
    cin.ignore(1);
```

```
// pop an item
    cout << "pop out an item: ";
    cin.ignore(1);
    st.pop();
    //show the stack content
    st.display();
    // pop an item
    cout << "pop out an item: ";
    cin.ignore(1);
    st.pop();
    //show the stack content
    st.display();
    // pop an item
    cout << "pop out an item: ";
    cin.ignore(1);
    st.pop();
    st.display();
    Return 0;
}
```

Requirement

- Implement `stack::push()`, `stack::pop()`, and `stack::display()`.
- Whenever an item is popped, output the average grades of Math, English and Computer of the rest students in the stack.

Sample input & output

```
 # The stack is generated.
push an item: 001 100 90 80
push an item: 002 90 80 70
push an item: 003 70 60 50
003 70 60 50
002 90 80 70
001 100 90 80

Press any key to continue...
pop out an item:
math avg: 86.6667
enlish avg: 76.6667
computer avg: 66.6667
003 is removed.
002 90 80 70
001 100 90 80

pop out an item:
math avg: 95
enlish avg: 85
computer avg: 75
002 is removed.
001 100 90 80

pop out an item:
math avg: 100
enlish avg: 90
computer avg: 80
001 is removed.
List is empty!
 # The stack is deleted.
```

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.