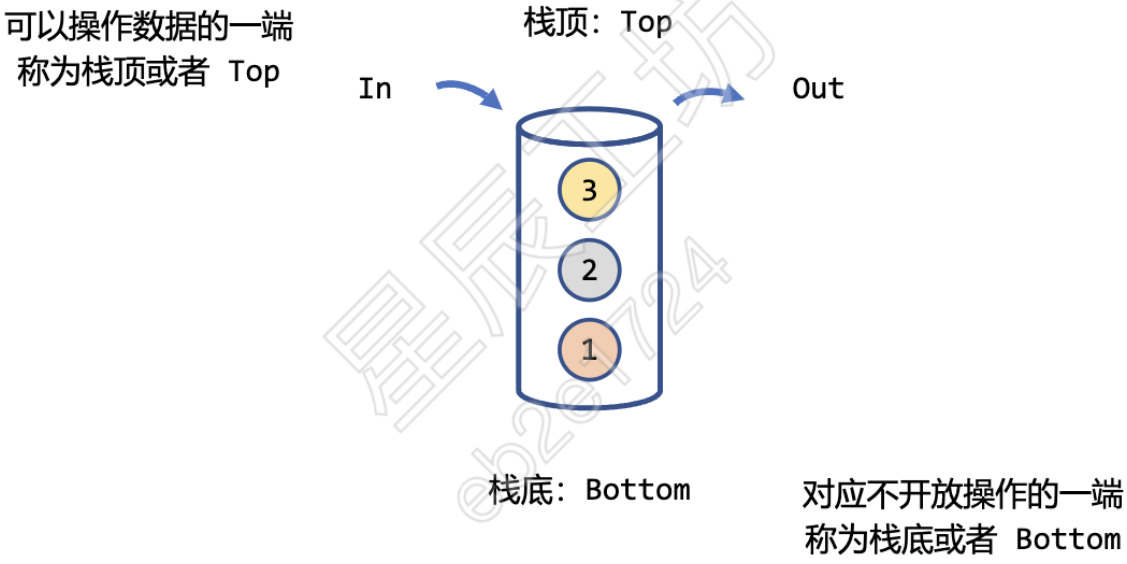
第十七章 栈与STL栈

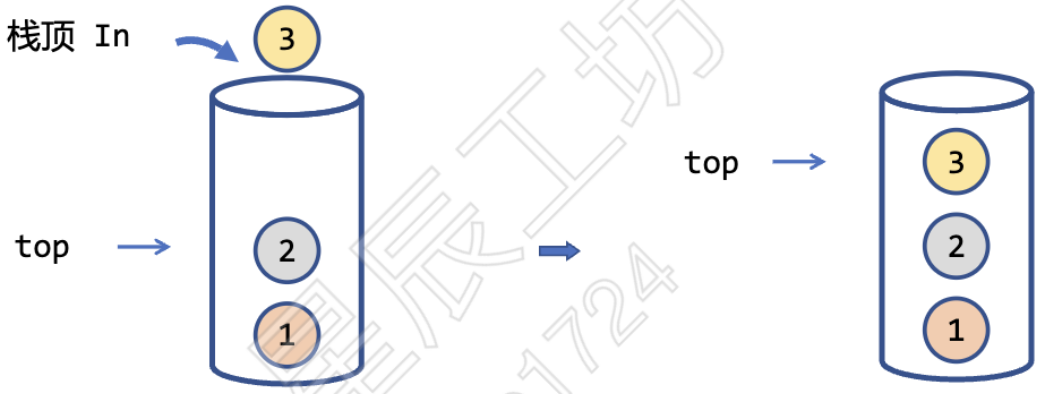
1. 栈

1.1栈的定义与模拟

- 栈的定义： 栈：是只能在一端进行数据操作的线性数据结构；
  栈具有先进后出的特点。 first in last out



- 栈的实现方式： 数组实现的栈（静态栈），链表栈，动态数组栈，使用库或者框架，循环数组实现的栈
- 入栈：



静态数组模拟入栈，可将 top + 1，top 指向数组下一个位置，表示新栈顶；
将获得的 x 保存在 top 位置，模拟栈顶入栈过程。

入栈操作：

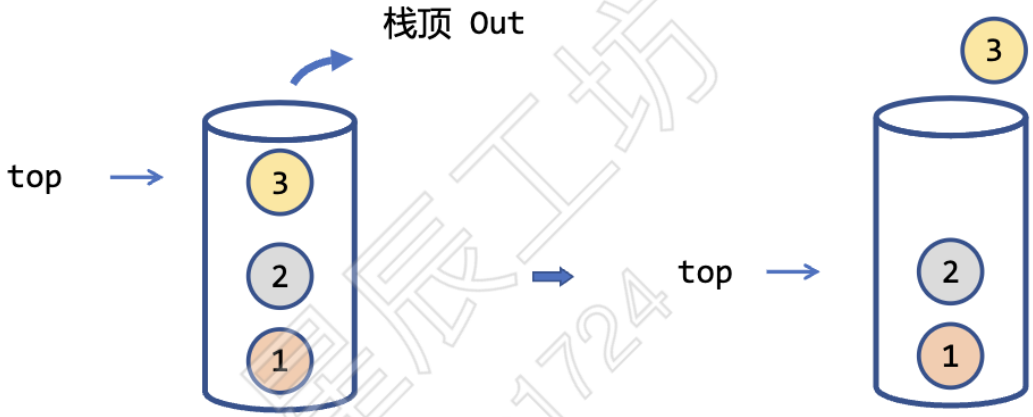
1. 在 top 小于 1000 时进行入栈；
2. 将 x 保存在新栈顶。

```
// 入栈
void push(int x)
{
    if (top < 1000) stack[++top] = x;
}
```

注：数组模拟栈时，要注意栈顶溢出问题。

栈顶不能超过初始定义的数组大小。

- 出栈：



静态数组模拟出栈，可将 top-1，top 指向数组上一个位置作为新栈顶。

出栈操作：

1. 在 top 不为 0 时执行出栈操作；
2. top--。

```
// 出栈
void pop()
{
    if (top != 0) top--;
}
```

注：数组模拟栈时，栈顶不能小于 0，防止栈底溢出

- 栈顶元素，大小，判空
  由于 top 初始为 ，++top 后才存储元素，数组模拟的过程中索引为的位置不存储元素，所以栈顶位置 = 大小 = top。关于判空，也只需考虑top是否为0。

1. 获取栈顶元素

```
int get_top()
{
    if(!empty()) return stack[top];
}
```

2. 大小

```
int size()
{
    return top;
}
```

3. 判空

```
bool empty()
{
    return top == 0;
}
```

- 遍历

  将栈内元素一一出栈，可依次获取栈内元素。

  遍历操作：

  1. top 不为 0 时，获取栈顶元素；
  2. 同时将栈顶元素出栈；
  3. 直至 top 为 0。

```
// 遍历
void print_stack()
{
    while (!empty())
    {
        cout << get_top() << " ";
        pop();
    }
    cout << endl;
}
```


1.2 习题演练（主要以选择题形式出现）

1. 合法出栈顺序。给定x、y、z3个字符，以下哪项不可能是x、y、z的合法入栈和出栈过程？ A zyzzyx
   B xxyzzy
   C xyyzzx
   D xyzxyz
   答案：D，D选项中，xyz依次入栈后在栈顶的元素应该是z，出栈过程不合法
2. 编程题
   题目描述：
   给定一个元素个数为n的整数数列，输出每个数左边第一个比它小的数，如果不存在则输出-1。 输入格式：
   共两行； 第一行一个整数n（5≤n50），表示元素个数
   第二行n个整数（1=<整数<=100），表示整数数列，整数之间以一个空格隔开
   输出格式
   一行，包含n个整数，其中第i个数表示第个数的左边比它小的第一个数，如果不存在，输出-1。 输入样例
   5
   6 3 9 7 2
   输出样例
   -1 -1 3 3 -1

分析：
单调栈：栈内元素是单调的，可分为单调递增栈，单调递减栈
由于需要找到左边第一个比它小的数，可以利用单调递增栈

1. 栈中有元素的情况下，将所有比x大的数都出栈
2. 如果栈不为空，那么栈顶元素就是第一个小于的元素；
3. 如果栈为空，那么没有找到比x小的元素，输出
4. 将当前元素入栈，栈仍保持单调递增

5. 直至所有元素都查找结束

代码：

```cpp
#include<iostream>
using namespace std;

int stk[1001], top;
int n;
int main()
{
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        int x;
        cin >> x;
        // 栈中有元素的情况下，将所有大于等于 x 的数都出栈
        while (top && stk[top] >= x) top--;
        // 如果栈不为空，栈顶元素就是第一个小于 x 的元素
        if (top) cout << stk[top] << ' ';
        // 栈为空，没有找到比 x 小的元素，输出 -1
        else cout << -1 << ' ';
        // 当前元素入栈，栈仍保持单调递增
        stk[++top] = x;
    }
    return 0;
}
```