模拟

瞿绍军
湖南师范大学

Speed Limit-poj2017

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

int main(){
	int n,s[11],t[11];
	int i,ret;
	while(cin>>n, n != -1){
		ret = 0;
		memset(s, 0, sizeof(s));
		memset(t, 0, sizeof(t));
		for(i = 1; i <= n; i++){
			cin>>s[i]>>t[i];
			ret += s[i] * (t[i] - t[i - 1]);
		}
		cout<<ret<<" miles"<<endl;
	}
	return 0;
}
```

Counterfeit Dollar POJ 1013

- 题意：
- 某人有12枚硬币，确定其中有且仅有一枚为假的，且不知较轻或较重，已知条件有
- 假币又且仅有一枚
- 真币刚好11枚
- 每个case保证有且仅有唯一解
- 每个case只称量三次
- 每次称量都保证天平左右两边硬币数量一致
- 硬币编号用大写字母 A-L 表示
- even表示天平两头平衡
- down表示天平右侧下沉
- up表示天平右侧上升

Counterfeit Dollar POJ 1013

- 输入：
- 第一行为case总数量
- 接下去每三行代表一个case
- 每个case中的每一行分三部分
- 第一部分表示天平左边
- 第二部分表示天平右边
- 第三部分表示天片右边平衡状态
- 输出：
- 输出假硬币的编号
- 输出假硬币是轻是重

Counterfeit Dollar POJ 1013

- 核心思想是根据每次称量来增加假币的被怀疑程度
- 1.初始化所有硬币有可能为真，用0表示
- 2.遇到even，标记两边硬币确实为真，用99999表示
- 3.遇到up，处理已经标记确实为真之外的硬币
- 左盘：硬币怀疑为重币，值++
- 右盘：硬币怀疑为轻币，值--

Counterfeit Dollar POJ 1013

- 4.遇到down，处理已经标记确实为真之外的硬币
- 左盘：硬币怀疑为轻币，值- -
- 右盘：硬币怀疑为重币，值++
- 5.由于硬币有且仅有一个是假的，且输入保证有解
- 遍历12枚硬币的值，不处理99999确实为真的硬币
- 取出值与0的绝对值最大的那枚硬币，代表被怀疑程度最大，此币为假。值如为负，则是轻币；值如为正，则是重币。

Counterfeit Dollar POJ 1013

- 思考如下：
- 输入保证了解的唯一性
- 如果假币为轻的，无论放在哪一边去称量，这枚硬币始终会被- -
- 如果假币为重的，无论放在哪一边去称量，这枚硬币始终会被++
- 由于这三点，所以假币被称量的次数越多，他的值越会偏离初始值0，所以最终答案就是那枚绝对值最大的硬币

Web Navigation Poj1028

- The following commands need to be supported:
  BACK: Push the current page on the top of the forward stack. Pop the page from the top of the backward stack, making it the new current page. If the backward stack is empty, the command is ignored.
  FORWARD: Push the current page on the top of the backward stack. Pop the page from the top of the forward stack, making it the new current page. If the forward stack is empty, the command is ignored.
  VISIT : Push the current page on the top of the backward stack, and make the URL specified the new current page. The forward stack is emptied.
  QUIT: Quit the browser.
  Assume that the browser initially loads the web page at the URL http://www.acm.org/

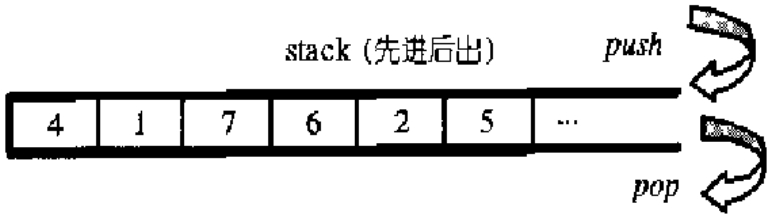
Web Navigation Poj1028

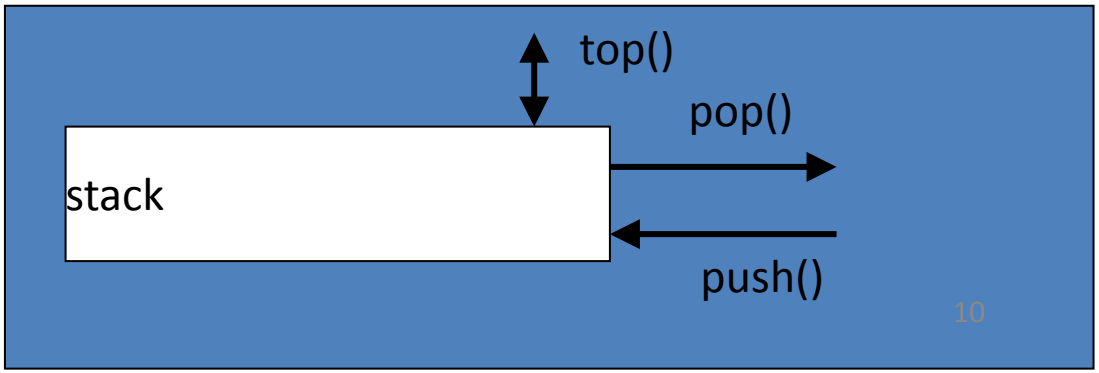
- 一道栈模拟题目，要注意地方只有一个，就是当在栈中某处访问新地址后，不能再访问栈此处之后的地址，只能访问之前的。

- 直接用vector来实现栈，将vector的尾端看做栈的顶端，在这里就行pop和push操作。。。push就直接用vector的push_back方法，pop就用vector的back方法，并用erase删除顶端元素。注意：end操作返回的是最后一个元素的下一个位置，所以要减去1。

STL容器：堆栈

- stack
  - 后进先出（LIFO）
  - #include <stack>
  - 核心接口
    - empty() -- 返回bool型，表示栈内是否为空 (s.empty())
    - size() -- 返回栈内元素个数 (s.size())
    - push(value)—将元素压栈
    - top()—返回栈顶元素，但不移除
    - pop()—从栈中移除栈顶元素，但不返回
    - 不提供迭代器
  - 实例：stack





- TOJ Problem 1312　五子棋问题
- 题目要求：
- 　　判断白棋赢还是黑棋赢，如果黑/白的赢了，就是有5个连续的黑/白棋，则输出1 / 2，然后输出这5个棋中的最左边的那一个，如果这5个棋是竖直排列的，则输出最上边的一个。

- 题意理解：
- 1 黑棋和白棋不可能同时赢。
- 2 任何一方如果有多于5个连续的棋相连不算赢。
- 3 棋盘是19*19的，因此模拟数组只要开到[21][21]
- 就可以。
- 问题求解：
- 只要按照从左上到右下的顺序遍历一遍棋盘就可以了，顺序是：先左后右，先上后下。如果遇上是棋子就判断是否赢。

- 伪代码：
- for(j=1;j<=19;j++)
- {
- for(i=1;i<=19;i++)
- {
- if(any one wins)
- break;
- }
- if(any one wins)
- break;
- }

- Hunnu11316