341. Flatten Nested List Iterator

题目描述：https://leetcode.com/problems/flatten-nested-list-iterator/

> 给定一个NestedInteger把它扁平化；
> bool hasNext() // 返回是否有值
> int next() //返回下一个整数值，并且下标加一
>
> 例如：
>
> ```
> given [[1,1],2,[1,1]]
> 依次调用hasNest 和 next
> ```

解题思路：

1. 初始化的时候把结果都存好
2. 每次hasNext现找

代码1：

```
    /**
     * // This is the interface that allows for creating nested lists.
     * // You should not implement it, or speculate about its implementation
     * class NestedInteger {
     *   public:
     *     // Return true if this NestedInteger holds a single integer, rather tha
n a nested list.
     *     bool isInteger() const;
     *
     *     // Return the single integer that this NestedInteger holds, if it holds
 a single integer
     *     // The result is undefined if this NestedInteger holds a nested list
     *     int getInteger() const;
     *
     *     // Return the nested list that this NestedInteger holds, if it holds a
nested list
     *     // The result is undefined if this NestedInteger holds a single integer
     *     const vector<NestedInteger> &getList() const;
     * };
     */
    class NestedIterator {
    public:
        vector<int> content;
```

```
        NestedIterator(vector<NestedInteger> &nestedList) {
            stack<NestedInteger> st;
            for(int i = 0; i < nestedList.size(); i++) {
                st.push(nestedList[i]);
            }
            while(!st.empty()) {
                NestedInteger ni = st.top();
                st.pop();
                if(ni.isInteger()) {
                    content.push_back(ni.getInteger());
                }
                else {
                    vector<NestedInteger> nini = ni.getList();
                    for(int i = 0; i < nini.size(); i++) {
                        st.push(nini[i]);
                    }
                }
            }
        }

        int next() {
            int i = content[content.size()-1];
            content.pop_back();
            return i;
        }

        bool hasNext() {
            return content.size() > 0;
        }
    };

    /**
     * Your NestedIterator object will be instantiated and called as such:
     * NestedIterator i(nestedList);
     * while (i.hasNext()) cout << i.next();
     */
###代码2：
/**
 * // This is the interface that allows for creating nested lists.
 * // You should not implement it, or speculate about its implementation
 * class NestedInteger {
 *   public:
 *     // Return true if this NestedInteger holds a single integer, rather than a
nested list.
 *     bool isInteger() const;
 *
 *     // Return the single integer that this NestedInteger holds, if it holds a s
ingle integer
```

```
 *     // The result is undefined if this NestedInteger holds a nested list
 *     int getInteger() const;
 *
 *     // Return the nested list that this NestedInteger holds, if it holds a nest
ed list
 *     // The result is undefined if this NestedInteger holds a single integer
 *     const vector<NestedInteger> &getList() const;
 * };
 */
class NestedIterator {
public:
    stack<NestedInteger> st;
    NestedIterator(vector<NestedInteger> &nestedList) {
        for(int i = nestedList.size() - 1; i >= 0; i--) {
            st.push(nestedList[i]);
        }
    }

    int next() {
        NestedInteger ni = st.top();
        st.pop();
        return ni.getInteger();
    }

    bool hasNext() {
        while(!st.empty()) {
            NestedInteger ni = st.top();
            if(ni.isInteger()) {
                return true;
            }
            st.pop();
            vector<NestedInteger> vni = ni.getList();
            for(int i = vni.size() - 1; i >= 0; i--) {
                st.push(vni[i]);
            }
        }
        return false;
    }
};

/**
 * Your NestedIterator object will be instantiated and called as such:
 * NestedIterator i(nestedList);
 * while (i.hasNext()) cout << i.next();
 */
```