# 数组

# 可变长度数组

但是有的时候数组要开多大比较难以计算，无法确定。

在 C++ 的 STL 中，给我们提供了一个可变长度数组。可变长度数组的头文件是 <vector>。vector 有几个常用功能：

| 功能 | 说明 |
| --- | --- |
| vector<int> v(N,i) | 建立一个可变长度的 int数组 v，且初始有 N 个为 i的元素。N,i 可以省略。 |
| v.push_back(a) | 将元素 a插入 v的末尾。 |
| v.size() | 返回元素个数。 |
| v.resize(n,m) | 重新调整数组大小为 n。如果 n比原来大，则新增的部分都初始化为 m。 |
| v[a] | 访问下标为 a的元素。 |

# 栈

古代的货栈只有一个门用于运入或运出货物。当货栈堆满时，为了从中运出货物，往往是最近放的货物最先拿出，以前放的货物会后拿出。

# 栈

C++ STL 提供了栈。它的头文件是 <stack>，有以下的方法：

| 功能 | 说明 |
| --- | --- |
| stack<int> s | 建立一个栈 s，其内部元素类型是 int。 |
| s.push(a) | 将元素 a压进栈 s。 |
| s.pop() | 将 s的栈顶元素弹出。 |
| s.top() | 查询 s的栈顶元素。 |
| s.empty() | 查询 s是否为空。 |
| s.size() | 查询 s的元素个数。 |

# 栈

在使用栈的时候，需要防止栈因存储内容过多而导致溢出，也需 要防止对空栈弹出元素。

C++ STL 已经做了一部分处理，但是查询空栈的栈头也是会导致 Runtime Error 的 。

对于手写栈可以在操作之前进行判断。例如弹出操作：

```
void pop(){
    if (p == 0) printf("Stack is empty"); else
    p -= 1;
}
```

注意：STL 虽然提供了许多方便的功能，但是如果使用 STL 时不打 开 -O2 优化开关，就有一点慢（常数大）。在需要追求运行速度 的情况下，往往需要自己手写栈。

# 栈

记录一个栈顶指针 p，指向下一个待插入的数组位置。

push 操作：直接把 x 赋值给 stack[p]，并且更新指针加一即可。

```
void push(int x){
    stack[p] = x; p += 1;
}
```

pop 操作：直接让栈顶指针后退一位。

```
void pop(){
    p -= 1;
}
```

top 操作：因为 p 指向下一个待插入的数组位置，所以栈顶的数组位置是 p-1。

```
int top(){
    return stack[p - 1];
}
```

# C++ 读入字符串

C++ 中处理字符串问题的一个注意点：

使用 cin 读入一个独占的数字后，其读入指针在这一行的末尾。 使用 getline 读入一行字符串时，只会读到空串（第一行）。如果希望读到第二行，则必须要假装读入这一行，可以使用 getline，也可以使用 getchar 等。

思考：如果没有第三行的 getline 会有什么后果？

```
cin >> num;
string p;
getline(cin, p); //  假装换行符，读入空行
while (num--) {
    getline(cin, p);
    // 读入一行，这样这个 p 是正常的。
```

```
3
([])
(([()]))
([()[]()])()
```

# 队列

在超市种，收银员会给排在队伍最前面的顾客买单，然后服务队伍中下一个顾客。而队伍的末尾也一直会有更多的顾客依次加入队列。

# 队列

C++ STL 提供了队列。它的头文件是 <queue>，有以下的方法：

| 功能 | 说明 |
| --- | --- |
| queue<int> q | 建立一个队列 q，其内部元素类型是 int。 |
| q.push(a) | 将元素 a插入队列 q 的末尾。 |
| q.pop() | 将 q 的队首元素删除。 |
| q.front() | 查询 q 的队首元素。 |
| q.end() | 查询 q 的队尾元素。 |
| q.size() | 查询 q 的元素个数。 |
| q.empty() | 查询 q 是否为空。 |

# 队列

队列是有头（head）尾（tail）的。需要记录队列的头和尾在哪里。

push 操作：直接把 x 赋给队尾，再让队尾 +1。

```
void push(int x){
    queue[tail] = x; tail += 1;
}
```

pop 操作：不必让元素全部往前进一位，只需让头指针前进。

```
void pop(){ 。
    head += 1;
}
```

front 操作：直接获取 q[head] 即可。

```
int front(){
    return queue[head];
}
```