

C++方向编程题答案

第一周

day3

题目ID: 69385--字符串中找出连续最长的数字串

链接: <https://www.nowcoder.com/practice/bd891093881d4ddf9e56e7cc8416562d?tpId=85&&tqId=29864&rp=1&ru=/activity/oj&qu=/ta/2017test/question-ranking>

【题目解析】：

本题是一个很简单的题目，这里就不解析了。

【解题思路】：

遍历字符串，使用cur去记录连续的数字串，如果遇到不是数字字符，则表示一个连续的数字串结束了，则将数字串跟之前的数字串比较，如果更长，则更新更长的数字串更新到res。

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str,res,cur;
    cin>>str;
    for(int i=0;i<=str.length();i++)
    {
        // 数字+=到cur
        if(str[i]>='0' && str[i]<='9')
        {
            cur+=str[i];
        }
        else
        {
            // 找出更长的字符串，则更新字符串
            if(res.size() < cur.size())
                res=cur;
            else
                cur.clear();
        }
    }
    cout<<res;
    return 0;
}
```

23271-数组中出现次数超过一半的数字

<https://www.nowcoder.com/practice/e8a1b01a2df14cb2b228b30ee6a92163?tpId=13&tqId=11181&tPage=2&rp=2&ru=/ta/coding-interviews&qu=/ta/coding-interviews/question-ranking>

【题目解析】：

本题题意很简单，需要找出超过一半的那个数字。需要注意这个题是一个往年面试的热门题型

【解题思路1】：

思路一：数组排序后，如果符合条件的数存在，则一定是数组中间那个数。这种方法虽然容易理解，但由于涉及到快排sort，其时间复杂度为 $O(N\log N)$ 并非最优；

```
class Solution {
public:
    int MoreThanHalfNum_Solution(vector<int> numbers)
    {
        // 因为用到了sort，时间复杂度 $O(N\log N)$ ，并非最优
        if(numbers.empty()) return 0;

        sort(numbers.begin(), numbers.end()); // 排序，取数组中间那个数
        int middle = numbers[numbers.size()/2];

        int count=0; // 出现次数
        for(int i=0; i<numbers.size(); ++i)
        {
            if(numbers[i]==middle) ++count;
        }

        return (count>numbers.size()/2) ? middle : 0;
    }
};
```

【解题思路2】：

如果有符合条件的数字，则它出现的次数比其他所有数字出现的次数和还要多。在遍历数组时保存两个值：一是数组中一个数字，一是次数。遍历下一个数字时，若它与之前保存的数字相同，则次数加1，否则次数减1；若次数为0，则保存下一个数字，并将次数置为1。遍历结束后，所保存的数字即为所求。然后再判断它是否符合条件即可。

```
class Solution {
public:
    int MoreThanHalfNum_Solution(vector<int> numbers)
    {
        if(numbers.empty()) return 0;

        // 遍历每个元素，并记录次数；若与前一个元素相同，则次数加1，否则次数减1
        int result = numbers[0];
        int times = 1; // 次数

        for(int i=1; i<numbers.size(); ++i)
        {
            if(times == 0)
            {
                // 更新result的值为当前元素，并置次数为1
                result = numbers[i];
            }
            if(result == numbers[i]) ++times;
            else --times;
        }

        return result;
    }
};
```

```
        times = 1;
    }
    else if(numbers[i] == result)
    {
        ++times; // 相同则加1
    }
    else
    {
        --times; // 不同则减1
    }
}

// 判断result是否符合条件，即出现次数大于数组长度的一半
times = 0;
for(int i=0;i<numbers.size();++i)
{
    if(numbers[i] == result) ++times;
}

return (times > numbers.size()/2) ? result : 0;
}
};
```