cpp

Peking University
北京大学
ShuwenHe
何书文

2023 年 6 月 18 日

目录

Chapter

C++

1 环境搭建

1.1 linux 文件文件夹命令

创建文件夹
mkdir cpp 创建文件夹
mkdir cpp
进入文件夹
cd cpp
vi richard.cpp
i 进入 insert 模式
o 进入下一行
int 整型
main() 主函数 function
// 单行注释
“” 输入字符串
endl
; 一行语句结束需要用分号结束
return 0 一个函数正确执行完成之后需要用 return 0 来结束
esc 推出 vi 编辑器
:wq 保存并推出 write quit
ls 查看当前文件夹有什么文件 ls - list directory contents

1.2 g++ 编译器

g++ 编译器 C++

g++ richard.cpp -o richard

./richard 执行编译器编译产生的二进制文件

Chapter

for 循环

1

// 高斯求和公式求和 1+2+3+...+100

```
#include <iostream>
using namespace std;
int main() {
    int sum = 0;
    sum = (1+100)*100/2;
    cout << "gauss sum=" << sum << endl;
    return 0;
}
```

2

for 循环求和 1+2+3+...+100

```
int forSum(){
    int sum = 0;
    for (int i = 1; i <= 100; i++) {
        sum += i;
    }
    cout << "for sum=" << sum << endl;
    return 0;
}
```

3 X34 珠心算测验

题目描述【题目描述】珠心算是一种通过在脑中模拟算盘变化来完成快速运算的一种计算技术。珠心算训练，既能够开发智力，又能够为日常生活带来很多便利，因而在很多学校得到普及。某学校的珠心算老师采用一种快速考察珠心算加法能力的测验方法。他随机生成一个正整数集合，集合中的数各不相同，然后要求学生回答：其中有多少个数，恰好等于集合中另外两个（不同的）数之和？最近老师出了一些测验题，请你帮忙求出答案。输入格式共两行，第一行包含一个整数 n，表示测试题中给出的正整数个数。第二行有 n 个正整数，每两个正整数之间用一个空格隔开，表示测试题中给出的正整数。输出格式一个整数，表示测验题答案。【输入输出样例】输入复制 4 1 2 3 4 输出 1 复制 2 【样例解释】由 1 + 2 = 3 , 1 + 3 = 4 1+2=3,1+3=4，故满足测试要求的答案为 2 2。注意，加数和被加数必须是集合中的两个不同的数。数据范围对于 100 1003 100 3 n 100，测验题给出的正整数大小不超过 10 , 000 10,000。NOIP2014 普及组第 1 题

假设集合中有 n 个数，我们可以使用双重循环枚举集合中的每一对数，判断它们的和是否也在集合中出现过。这个方法的时间复杂度为 $O(n^2)$，对于较大的 n 可能会超时。

更高效的方法是，先将集合中的数按从小到大排序，然后对于每个数 x，使用双指针法在剩余的数中寻找两个数，使它们的和等于 x。具体地，我们可以将左指针指向 x 的下一个数，将右指针指向集合中最大的数，然后不断地将左指针右移或右指针左移，直到两个指针相遇为止。如果两个指针指向的数的和等于 x，则找到了一组符合条件的数对。

这个方法的时间复杂度为 $O(n \log n)$（排序的时间复杂度为 $O(n \log n)$，每个数最多被判断一次，因此双指针法的时间复杂度为 $O(n)$），可以通过本题。

以下是使用双指针法实现题目的 C++ 代码：

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> a(n);
```

```
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    sort(a.begin(), a.end());
    int cnt = 0;
    for (int i = 0; i < n; i++) {
        int left = i + 1, right = n - 1;
        while (left < right) {
            int sum = a[left] + a[right];
            if (sum == a[i]) {
                cnt++;
                left++;
                right--;
            } else if (sum < a[i]) {
                left++;
            } else {
                right--;
            }
        }
    }
    cout << cnt << endl;
    return 0;
}
```

代码中使用了 STL 的 ‘vector‘ 和 ‘sort‘，可以方便地实现数组的排序。双指针法的部分使用了 ‘while‘ 循环和条件语句进行实现。