### OOP第一周作业文档

2019010175 孔瑞阳 土木92

### 功能说明

输入一个正整数n。

当整数n不是正整数的时候，提示输出的不是正整数，并要求重新输入。

输出不超过n 的所有正偶数之和。

### 模型

##### 计算方法1

从2开始枚举偶数，2,4,6,...。每枚举到一个数之后加到和里面去。

当枚举到的偶数比n大的时候就退出循环。

##### 计算方法2

1. 当n是偶数的时候，答案就是2+4+...+n。

运用等差数列公式求和，答案为 (n+2) \* (n / 2) / 2。

1. 当n是奇数的时候，答案就是2+4+...+(n-1)。

运用等差数列公式求和，答案为 (n+1) \* ((n - 1) / 2) / 2。

因为n是奇数，所以这里的 (n-1)/2也可以写成 n/2。

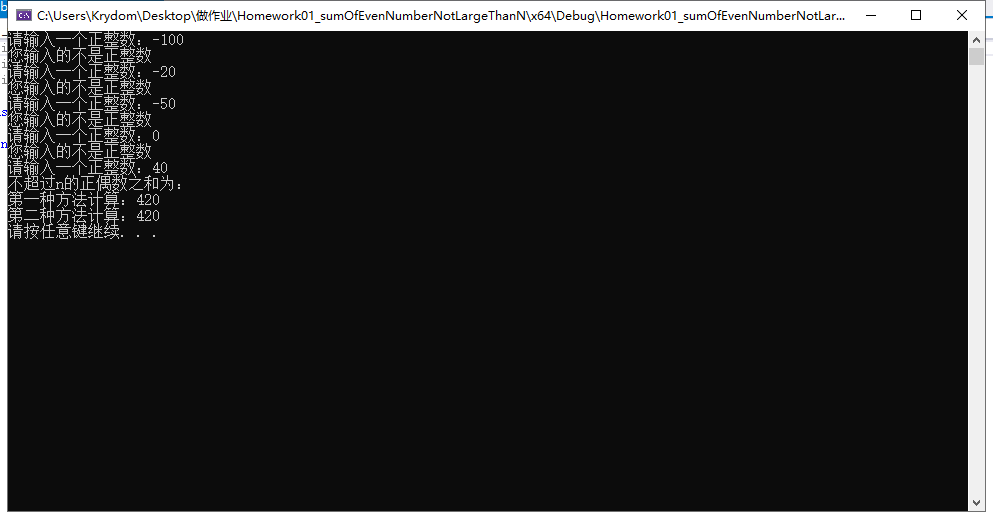
### 验证

##### 关于输入内容的验证

如果输入的是负数或者0的话，那么应该给出提示，并且重新输入。

如果输入的是一个正整数，那么进行下面的操作。

验证结果如下：



可以看出，程序运行和输入的要求符合，对于负数举的几个例子是不失一般性的。

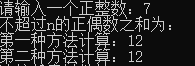
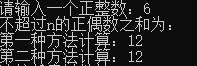
\*但是当输入的内容超过int的范围或者不是整数时，会出现错误。

##### 关于程序内容正确性的验证

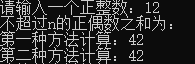
第一步：对于比较小的数据进行手算。

并且可以看成，但n是一个偶数的时候，n和n+1的答案应该是一样的。

n=6/7，则ans = 2 + 4 + 6 = 12



N=12/13，则ans = 2 + 4 + 6 + 8 + 10 + 12 = 42



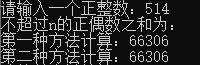
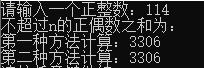
符合预测情况。

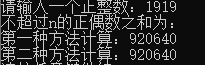
当n比较大的时候，手算不能直接算出来。

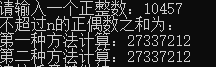
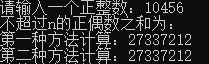
但是因为有两种计算方法，并且第一种计算方法由于自身的简单性质，经过肉眼观察和在比较小的数据下经过了验证，可以基本上认为是正确的。并且当两种计算方法得出同样的结果之后，也就是不出现矛盾，也可以基本说算法是正确的。

注意第二种方法因为有分类讨论，所以要对奇数偶数也分别验证。

下面给出部分验证过程：







经过多组数据验证，两种计算方法都没有产生矛盾。并且和使用计算器进行计算也没有矛盾，可以认为程序是正确的。

\*但是当答案超过int的范围是会出现错误。

### 关于两种计算方法的比较

第一种计算方法不需要对于n进行讨论，只运用了最基础的循环，所以思维难度比较小，并且不需要分类讨论。缺点是时间复杂度为O(n)，比较高。

第二种方法运用了等差数列的计算公式，将时间复杂度优化到了O(1)，更加快速。缺点是因为计算过程中要对于奇数和偶数分类讨论。