**程序设计1\_2**

1. **实验目的和要求**

设计一个函数，计算S = 1 – 2 + 3 – 4 + 5 – 6 + … + / - N的值。要求时间复杂度为O(1) 。

**二、实验内容**

1. 实验准备
   1. 理论知识介绍
2. 时间复杂度计算

通常采用下属办法来估计算法在给定输入下的计算量：

1. 根据问题的特点合理地选择一种或几种操作作为“标准操作”，将标准操作作为一个抽象的运算单位。
2. 确定每个算法在给定输入下共执行了多少次标准操作，并将它作为算法的计算量。
3. 利用求和定理和求积定理相结合可以计算出最终的时间复杂度
4. 实验项目
   1. 分析
      1. 要计算的S = 1 – 2 + 3 – 4 + 5 – 6 + … + / - N， 一般来说是这里将一个加法运算作为一个标准操作，所以如果函数直接计算S = 1 – 2 + 3 – 4 + 5 – 6 + … + / - N的时间复杂度为O(N), 所以不能直接计算
      2. 为了时间复杂度为O(1), 我们就要简化算法，最直接简单的办法就是直接用数学的方法计算出S的通项公式。
   2. 方案
      1. 直接用数学的方法我们可以计算出S表达式的通项为s= -N / 2(N为偶数)，S= (N + 1) / 2(N为奇数)。所以我们直接可以将函数设计成分类讨论返回这两个值的函数
      2. 为了测试函数的正确性，我们在main函数里采用循环的方式测试数据的正确性
   3. 测试数据

#include<iostream>

using namespace std;

int sum(int n) {

if (n % 2 == 0) {

return -n / 2;

}

else

{

return (n + 1) / 2;

}

}

int main() {

for (int i = 1; i < 11; i++) {

cout << sum(i) << endl;

}

return 0;

}

* 1. 运行结果

1

-1

2

-2

3

-3

4

-4

5

-5

**三、实验小结**

1. 重点

时间复杂度的计算。

1. 难点

简化程序 的时间复杂度。

1. 解决方法

利用数学的方法直接计算出函数的通项公式。

1. 收获、体会

时间复杂度对于程序来说是一个很重要的部分，通常情况下最直接的方法的时间复杂度比较大，所以这种办法并不是最优的，我们可以通过各种简化的方法来减少我们程序运行所用的时间。