**作业 PA3 实验报告**

姓名：何正潇 学号：1950095日期：2021年11月1日

1. **涉及数据结构和相关背景**

**本题涉及的数据结构为栈，该题模拟栈的实际运算。**

**2. 实验内容**

## 运用栈模拟阶乘函数的调用过程

**2.1.1 问题描述**

以n的阶乘为例，递归是一种函数调用自身的方法，代码可以如此实现：

public long f(int n){

    if(n==1) return 1; //停止调用

    return n\*f(n-1); //调用自身

}

当调用一个函数时，编译器会将参数和返回地址入栈；当函数返回时，这些值出栈。

递归通常有两个过程：

（1）递归过程：不断递归入栈push，直到停止调用n=1

（2）回溯过程：不断回溯出栈pop, 计算n\*f(n-1)，直到栈空，结束计算。

**基本来说就是用栈模仿编译器运行递归程序的实验。**

**2.1.2 基本要求**

使用栈来模拟编译器运行递归程序

**2.1.3 数据结构设计**

class Data {

private:

long n; //函数的输入参数

//int returnAddress; //函数的返回地址(这题显然不需要)

public:

void set(int number);

int get\_number();

};

//设置参数n

void Data::set(int number)

{

n = number;

}

//取得参数n

int Data::get\_number()

{

return n;;

}

**2.1.4功能说明（函数、类）**

**/\*递归程序的压栈过程程序\*/**

void Recursive(stack<Data>& myStack)

{

Data digit;

int n;

cin >> n;

while (n > 1) { //挺简单的，就一直把整个数据给他嗯压进去就行

digit.set(n);

myStack.push(digit);

--n;

}

}

**/\*回溯程序的出栈过程程序\*/**

//开始回导

long backtracking(stack<Data>& myStack)

{

Data digit;

long sum = 1;

while (!myStack.empty()) {

digit = myStack.top();

myStack.pop();

sum \*= digit.get\_number();

}

return sum;

}

主要算法的时间复杂度:

进栈的时间复杂度O(1)

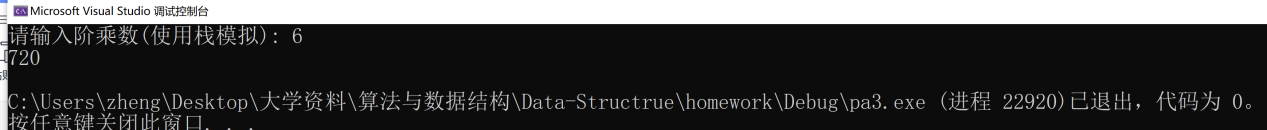
出栈的时间复杂度O(1)

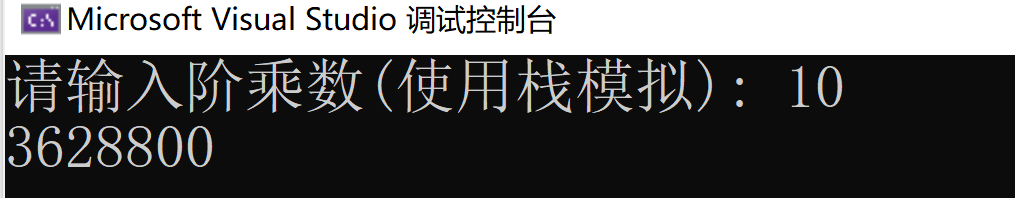
完成整个回溯过程的时间复杂度O(n)

本题的时间复杂度较低，因为程序基本运用的都是比较简单的栈操作。

**2.1.5 调试分析（遇到的问题和解决方法）**

基本来说进行了简单的测试，结果都正确，因为该题的思路较为清晰，所以没有遇到什么问题。





**2.1.6 总结和体会**

这道题总体来说我认为还是比较好玩的，虽然代码比较简单，但是让我们体会到了栈可以模拟递归程序运行的思想。进一步理解了栈的多用。对未来进一步的学习打下了坚实的基础。