项目说明文档

组合数学课程设计

——Catalan数

作 者 姓 名： 王星洲

学 号： 1652977

指 导 教 师： 冯巾松

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc531541899)

[1.1 题目需求分析 1](#_Toc531541900)

[1.2 功能分析 1](#_Toc531541901)

[2 设计 1](#_Toc531541902)

[2.1 数据结构设计 1](#_Toc531541903)

[2.2 流程设计 1](#_Toc531541904)

[3 实现 2](#_Toc531541905)

[3.1初始化 2](#_Toc531541906)

[3.1.1 主要代码 2](#_Toc531541907)

[3.1.2 注意事项 2](#_Toc531541908)

[3.2 接收用户的输入 3](#_Toc531541909)

[3.2.1用户输入过程流程图 3](#_Toc531541910)

[3.2.2核心代码 4](#_Toc531541911)

[3.2.3 人机交互优势及注意事项 5](#_Toc531541912)

[3.3 递推关系计算卡塔兰数功能的实现 6](#_Toc531541913)

[3.3.1 递推关系的计算 6](#_Toc531541914)

[3.3.2 递推关系流程图 7](#_Toc531541915)

[3.3.3 递推关系核心代码 8](#_Toc531541916)

[3.3.4 递推功能说明 8](#_Toc531541917)

[3.4 通项公式计算卡塔兰数功能的实现 8](#_Toc531541918)

[3.4.1 通项公式功能说明 8](#_Toc531541919)

[3.4.2 通项公式功能流程图 9](#_Toc531541920)

[3.4.3 通项公式核心代码 10](#_Toc531541921)

[3.4.4 通向公式功能亮点及注意事项 10](#_Toc531541922)

[3.5 计时功能的实现 10](#_Toc531541923)

[3.5.1 计时功能说明 10](#_Toc531541924)

[3.5.2 计时功能代码 10](#_Toc531541925)

[3.5.3 计时功能注意事项 11](#_Toc531541926)

[4 测试 12](#_Toc531541927)

[4.1 功能测试 12](#_Toc531541928)

[4.1.1 一般功能测试 12](#_Toc531541929)

[4.2 边界测试 12](#_Toc531541930)

[4.2.1 最大值测试 12](#_Toc531541931)

[4.2.2 最小值测试 13](#_Toc531541932)

[4.3 出错测试 13](#_Toc531541933)

[4.3.1 输入数据非法 13](#_Toc531541934)

[4.3.2 输入1或比1小的数 13](#_Toc531541935)

[4.3.3 输入比30大的数 14](#_Toc531541936)

[4.3.4 输入非整数 14](#_Toc531541937)

[5 两种核心算法时间消耗对比 14](#_Toc531541938)

[5.1 递推关系算法 14](#_Toc531541939)

[5.2 通项公式算法 14](#_Toc531541940)

[5.3 两种算法所需时间对比 14](#_Toc531541941)

[5.4 两种算法所需时间分析 15](#_Toc531541942)

# 1 分析

## 题目需求分析

输入1到30间某个数值x，显示出卡塔兰数列的前x项数值分别都是多少。并分别用递推关系和通向表达式两种方法计算卡塔兰数列前x项每项的时间开销。列出对比表格，并对结果进行分析。

## 1.2 功能分析

1. 输入：接收一个1-30间的正整数，由于书中的卡塔兰数定义，我们取大于1小于等于30的正整数。

2.计算卡塔兰数：需要用递推关系和通项表达式两种方法来计算，其中递推关系式为nCn+1=(4n-6)Cn。通项表达式为：Cn+1=1/n·C(2n-2,n-1)。

3.计算时间开销：将两种方法消耗的时间统计记录，并进行比较。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该程序使用int值来接收用户输入的x，两个算法消耗的时间用double数组time[30]来表示，其他运算所需数据用double类型变量。

## 2.2 流程设计

本程序首先创建接收用户输入的数据x，接下来用两种方法分别显示运算结果。然后再用clock函数分别计算两种方式的每一步的运行时间，把时间保存在time数组中，最后遍历数组，得到结果。

# 3 实现

## 3.1初始化

### 3.1.1 主要代码

#include <iostream>

#include <time.h>

using namespace std;

double digui(int x);

double tongxiang(int x);

int main()

{

int x = 0;

double time[31];

int startTime, endTime;

return 0;

}

### 3.1.2 注意事项

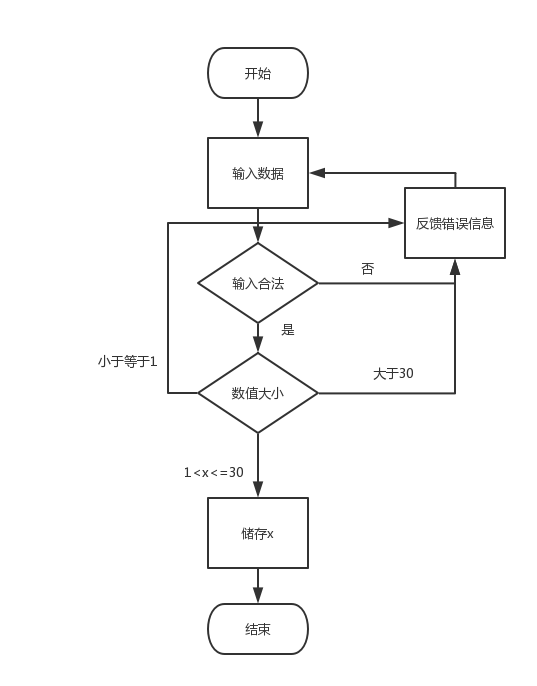
1.头文件<time.h>是为了之后使用clock函数计算运行时间。

2.digui(),tongxiang()两个函数分别对应用递归方式和通项公式方式计算卡塔兰数的算法。

3.x为接收用户输入的变量，time，startTime，endTime用于计算时间。

## 3.2 接收用户的输入

### 3.2.1用户输入过程流程图



### 3.2.2核心代码

cout << "请输入一个1~30之间的整数" << endl;

while (1)

{

cin >> x;

if (!cin >> x) //输入的不是int型

{

cout << "输入非法，请重新输入。" << endl;

cin.clear(); //清空输入

while (cin.get() != '\n')

{

continue;

}

}

else

{

if (x > 30) //输入越界

{

cout << "请输入30以下的整数。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

else if (x <= 1) //按照定义，1以下无卡塔兰数

{

cout << "请输入1以上的整数，不包括1。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

else

{

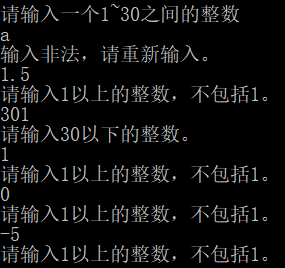
break;

}

}

}

3.2.3 人机交互截屏示例



### 3.2.3 人机交互优势及注意事项

1.我在输入中添加了合法检测

while (1)

{

cin >> x;

if (!cin >> x) //输入失败

{

cout << "输入非法，请重新输入。" << endl;

cin.clear(); //清空输入

while (cin.get() != '\n')

{

continue;

}

}

因为x是int类型，所以输入不合法的值会进入if语句，提示用户重新输入，同时清空输入缓存区，更改cin标志，为下次输入做准备，这样做大大提升了用户的体验，防止程序出现问题。

2.同时输入中添加了越界检测，如果用户没有保证输入的值处于题目范围，我们将给予提示并重新输入。

if (x > 30) //输入越界

{

cout << "请输入30以下的整数。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

else if (x <= 1) //按照定义，1以下无卡塔兰数

{

cout << "请输入1以上的整数，不包括1。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

else

{

break;

}

3.注意，代码中需要将输入流清空，否则将会在输出语句出现死循环

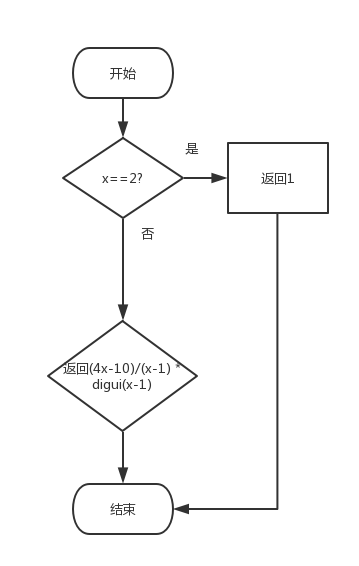
4.Cn在组合数学第五册书上定义域从2开始，1以下没有意义。其中C2=1，Cn表示了n边形被分割为三角形的可能分法数目。所以我们取n>=2。

## 3.3 递推关系计算卡塔兰数功能的实现

### 3.3.1 递推关系的计算

由nCn+1=(4n-6)Cn可求得Cn = (4n-10)/(n-1)\*Cn-1

### 3.3.2 递推关系流程图



### 3.3.3 递推关系核心代码

double digui(int x)

{

if (x == 2)

{

return 1;

}

else

{

return double(4 \* x - 10) / (x - 1)\*digui(x - 1);

}

}

### 3.3.4 递推功能说明

1.本函数使用递归方式实现

2.当x = 2时递归结束，开始回推

3.当x>2时，按照公式计算结果

## 3.4 通项公式计算卡塔兰数功能的实现

### 3.4.1 通项公式功能说明

Cn+1=1/n·C(2n-2,n-1) 可以求出Cn = 1/(n-1)·C(2n-4,n-2)

其中n>=3

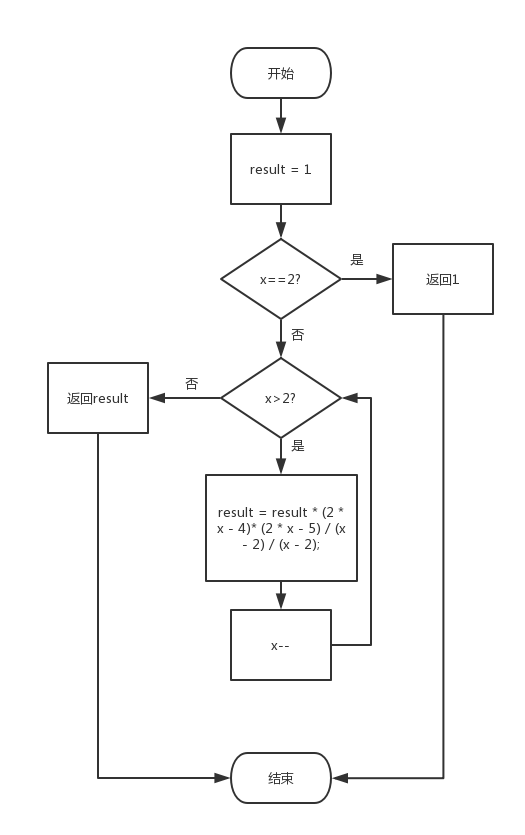
而C(2n-4,n-2)可以化为(2n-4)!/(n-2)!/(n-2)!

即((2n-4)(2n-5)(2n-6)…\*2\*1) / ((n-2)(n-2)(n-3)(n-3)…1)

而直接计算(2n-4)!达到了E74大小，不适合直接运算，所以可以将乘除法拆分开来，以便于快捷计算。即(2n-4)(2n-5) / (n-2)(n-2) \* (2n-6) (2n-7) / (n-3)(n-3)…

由此得出以下算法

### 3.4.2 通项公式功能流程图



### 3.4.3 通项公式核心代码

double tongxiang(int x)

{

double result = 1; //结果

if (x == 2)

{

return result;

}

else

{

double t = x - 1; //事先记录，以免x改变后无法计算

while (x > 2)

{

result = result \* (2 \* x - 4)\* (2 \* x - 5) / (x - 2) / (x - 2);

x--;

}

result = result / t;

return result;

}

}

### 3.4.4 通向公式功能亮点及注意事项

1.事先储存x的值，以免x改变后无法将result/(x-1)

2.本函数也可以用递归方式实现

3.为了避免数值过大，我们采用乘除混合方式运算，而不是计算所有阶乘再一起做乘除。

## 3.5 计时功能的实现

### 3.5.1 计时功能说明

由于单步实验所需时间很少，我们将其进行1000000次，再计算他们的时间，从2开始，到30，每一步用时都会被记录到数组中，最后再遍历数组。

### 3.5.2 计时功能代码

cout << "递归方法时间：" << endl;

for (int j = 2; j <= 30; j++)

{

startTime = clock(); //计时开始

for (int i = 0; i < 1000000; i++) //运行一百万次

{

digui(j);

}

endTime = clock(); //计时结束

time[j] = (double)(endTime - startTime) / CLOCKS\_PER\_SEC; //存入数组time，单位秒

}

for (int i = 2; i <= 30; i++)

{

cout << time[i] << " ";

}

cout << endl;

cout << "通项公式方法时间：" << endl;

for (int j = 2; j <= 30; j++)

{

startTime = clock();

for (int i = 0; i < 1000000; i++)

{

tongxiang(j);

}

endTime = clock();

time[j] = (double)(endTime - startTime) / CLOCKS\_PER\_SEC;

}

for (int i = 2; i <= 30; i++)

{

cout << time[i] << " ";

}

### 3.5.3 计时功能注意事项

1.每次测量计算1000000步重复运算的时间

2.再运算时间时转化为double 类型

# 4 测试

## 4.1 功能测试

### 4.1.1 一般功能测试

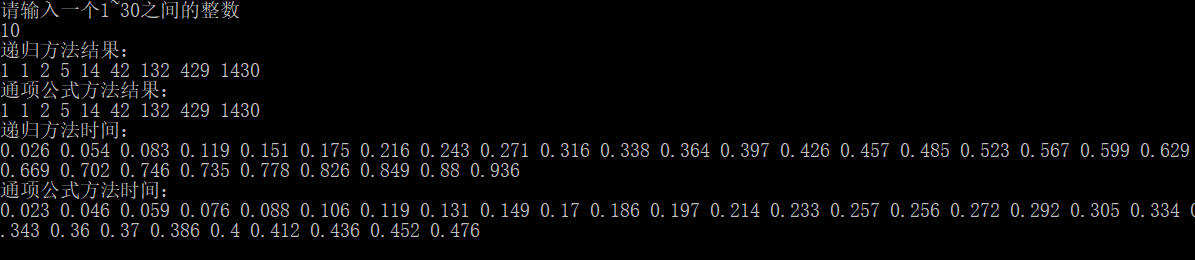
**测试用例**：

输入10

**预期结果**：

输出1 1 2 5 14 42 132 429 1430 以及两种方法所需的时间

**实验结果**



## 4.2 边界测试

### 4.2.1 最大值测试

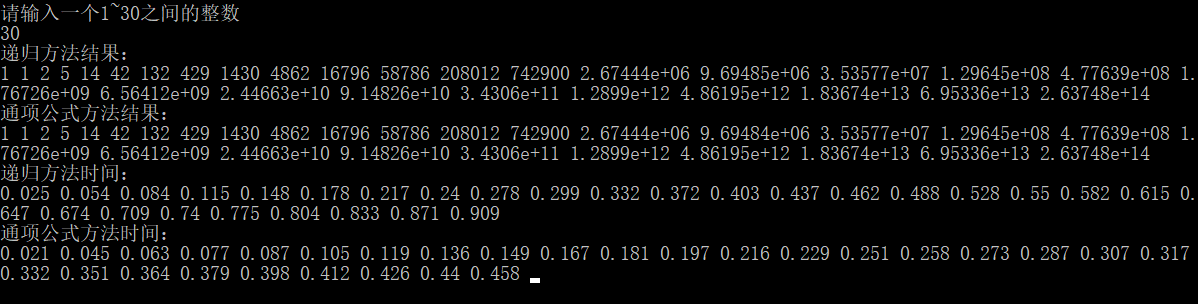
**测试用例：**

输入30

**预期结果：**

2-30所有卡塔兰数以及他们用两种算法所需的时间。

**实验结果：**



### 4.2.2 最小值测试

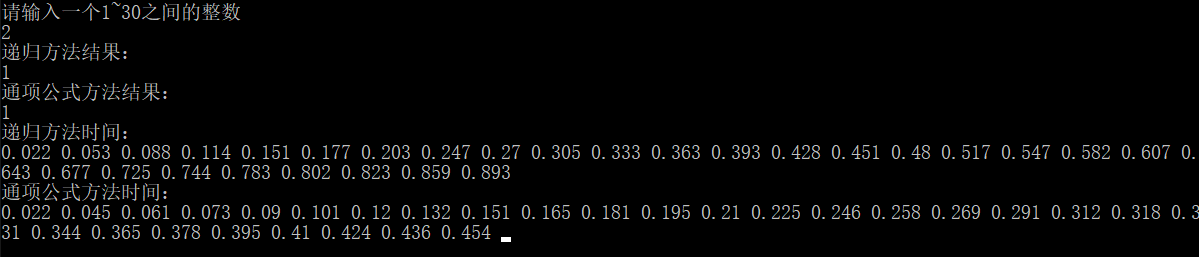
**测试用例：**

输入2

**预期结果：**

输出1，以及两种算法计算2卡塔兰数的时间

**实验结果：**



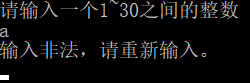
## 4.3 出错测试

### 4.3.1 输入数据非法

**测试用例：**输入字符而非数据

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

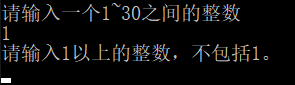


### 4.3.2 输入1或比1小的数

**测试用例：**1

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

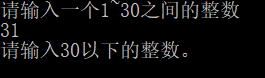


### 4.3.3 输入比30大的数

**测试用例：**31

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

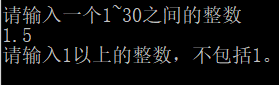


### 4.3.4 输入非整数

**测试用例：**1.5

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**



# 5 两种核心算法时间消耗对比

### 5.1 递推关系算法

参考3.3

运用递归函数，通过Cn-1来计算Cn的值，n=2时程序开始回推得出结果。

### 5.2 通项公式算法

参考3.4

通过通项公式，直接使用n通过循环计算两个阶乘的值，再进行运算得出结果。

### 5.3 两种算法所需时间对比

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 算法\时间 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 递推关系 | 0.022 | 0.053 | 0.088 | 0.114 | 0.151 | 0.177 | 0.203 | 0.247 | 0.27 |
| 通项公式 | 0.022 | 0.045 | 0.061 | 0.073 | 0.09 | 0.101 | 0.12 | 0.132 | 0.151 |
| 算法\时间 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 递推关系 | 0.305 | 0.333 | 0.363 | 0.393 | 0.428 | 0.451 | 0.48 | 0.517 | 0.547 |
| 通项公式 | 0.165 | 0.181 | 0.195 | 0.21 | 0.225 | 0.246 | 0.258 | 0.269 | 0.291 |
| 算法\时间 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 递推关系 | 0.582 | 0.607 | 0.643 | 0.677 | 0.725 | 0.744 | 0.783 | 0.802 | 0.823 |
| 通项公式 | 0.312 | 0.318 | 0.331 | 0.344 | 0.365 | 0.378 | 0.395 | 0.41 | 0.424 |

单位(s/1000000次)

|  |  |  |
| --- | --- | --- |
| 算法\时间 | 29 | 30 |
| 递推关系 | 0.859 | 0.893 |
| 通项公式 | 0.436 | 0.454 |

### 5.4 两种算法所需时间分析

可见，使用通项公式算法所需时间更低，在x=2时，两者都是进行了一个if判断，时间消耗大致相同，之后递推关系的算法x每增加1，时间就增加约0.3秒/100万次；而通项关系式的算法则大约增加0.15秒/100万次，约为递推关系算法的一半时间。

我认为，由于递推关系的算法必须要通过递归的思想实现，每一个新的结果都需要上一个结果，所以函数经历了从x到2，再从2到x的过程，一去一回；而通向关系算法使用循环的思想计算阶乘，只有从x到2的阶乘的过程，而无需从2到x，这样也就节省了这一半的时间。

也因此，x每增加1，递推算法所消耗的时间就增加了通项公式算法的两倍。