数据结构

2022

实 验 报 告

实验项目名称：	数据结构期中实验
班级：	21 级 7 班
学号：	2021302181140
姓名：	应晓宇
指导教师：	沈志东
实验时间：	2022.4.30

实验一： 对于 1~n 的每一个整数 n 输出值

一、实验要求

（1）独立完成实验

（2）撰写实验报告

二、实验环境

硬件：CPU：AMD RYZEM 5900HX GPU：RTX 3070

操作系统：windows 10

软件：visual studio code

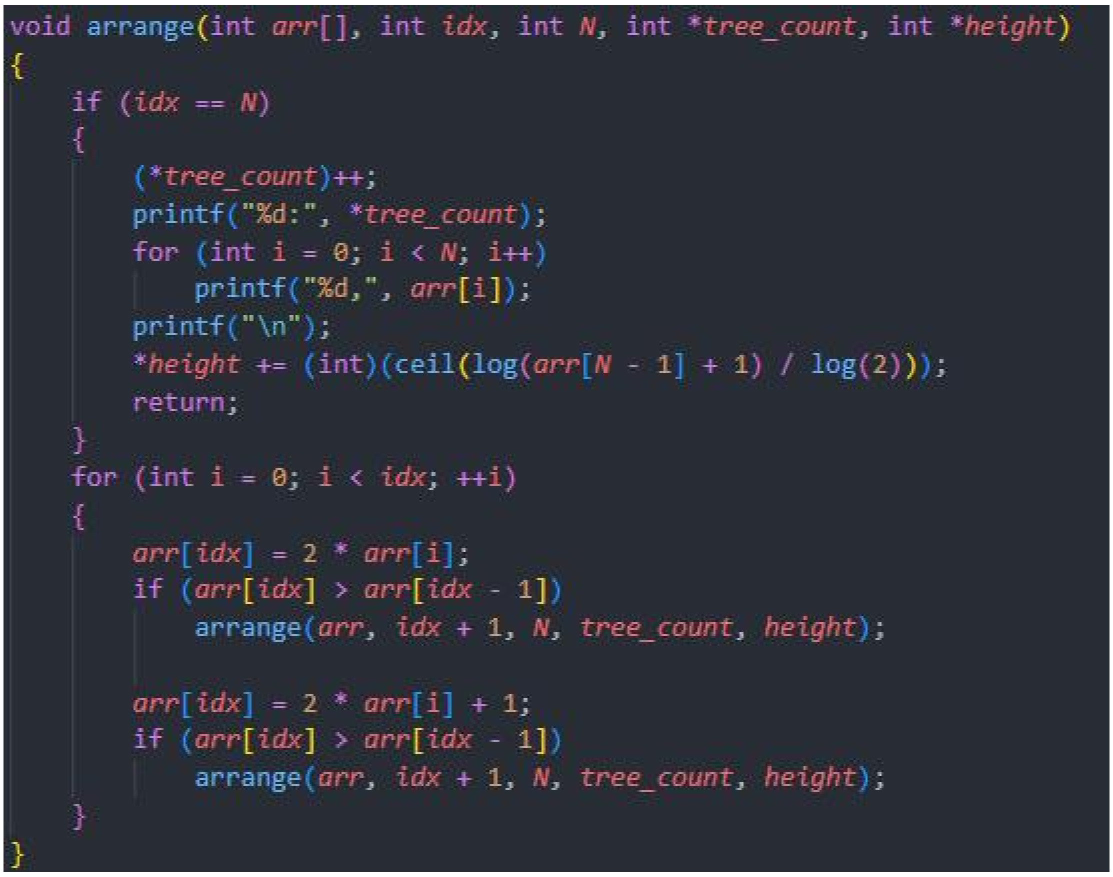
三、实验步骤及思路

题目分析：

题目已给定函数，则在函数中设计。

使用 idx 代表数组下标，用数组存储二叉树结点的值，数组长度为二叉树结点数。

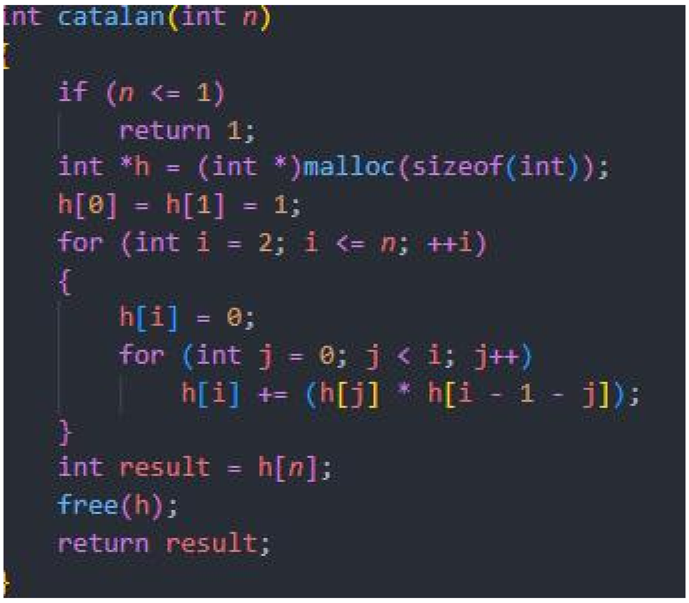
函数中，如果数组已满，则依次输出值，并使形状数+1，总高度增加。若未满，则数组分别填入左右元素，并检测是否合理，若合理，则递归调用用继续填入。



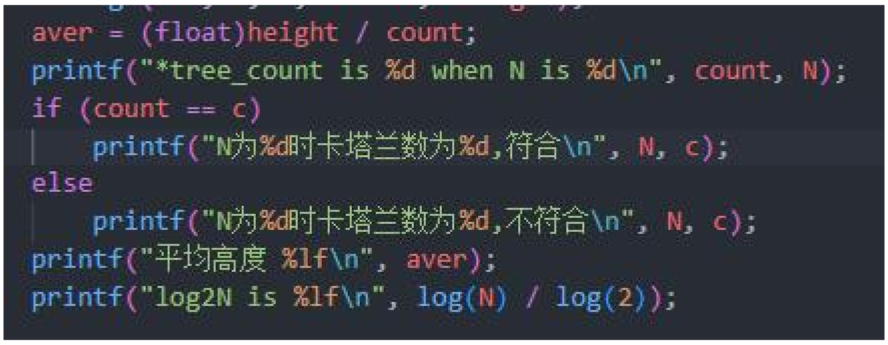
```c
void arrange(int arr[], int idx, int N, int *tree_count, int *height)
{
    if (idx == N)
    {
        (*tree_count)++;
        printf("%d:", *tree_count);
        for (int i = 0; i < N; i++)
            printf("%d,", arr[i]);
        printf("\n");
        *height += (int)(ceil(log(arr[N - 1] + 1) / log(2)));
        return;
    }
    for (int i = 0; i < idx; ++i)
    {
        arr[idx] = 2 * arr[i];
        if (arr[idx] > arr[idx - 1])
            arrange(arr, idx + 1, N, tree_count, height);

        arr[idx] = 2 * arr[i] + 1;
        if (arr[idx] > arr[idx - 1])
            arrange(arr, idx + 1, N, tree_count, height);
    }
}
```

题目还有其他要求，依次满足即可。

计算卡塔兰数：



```c
int catalan(int n)
{
    if (n <= 1)
        return 1;
    int *h = (int *)malloc(sizeof(int));
    h[0] = h[1] = 1;
    for (int i = 2; i <= n; ++i)
    {
        h[i] = 0;
        for (int j = 0; j < i; j++)
            h[i] += (h[j] * h[i - 1 - j]);
    }
    int result = h[n];
    free(h);
    return result;
}
```

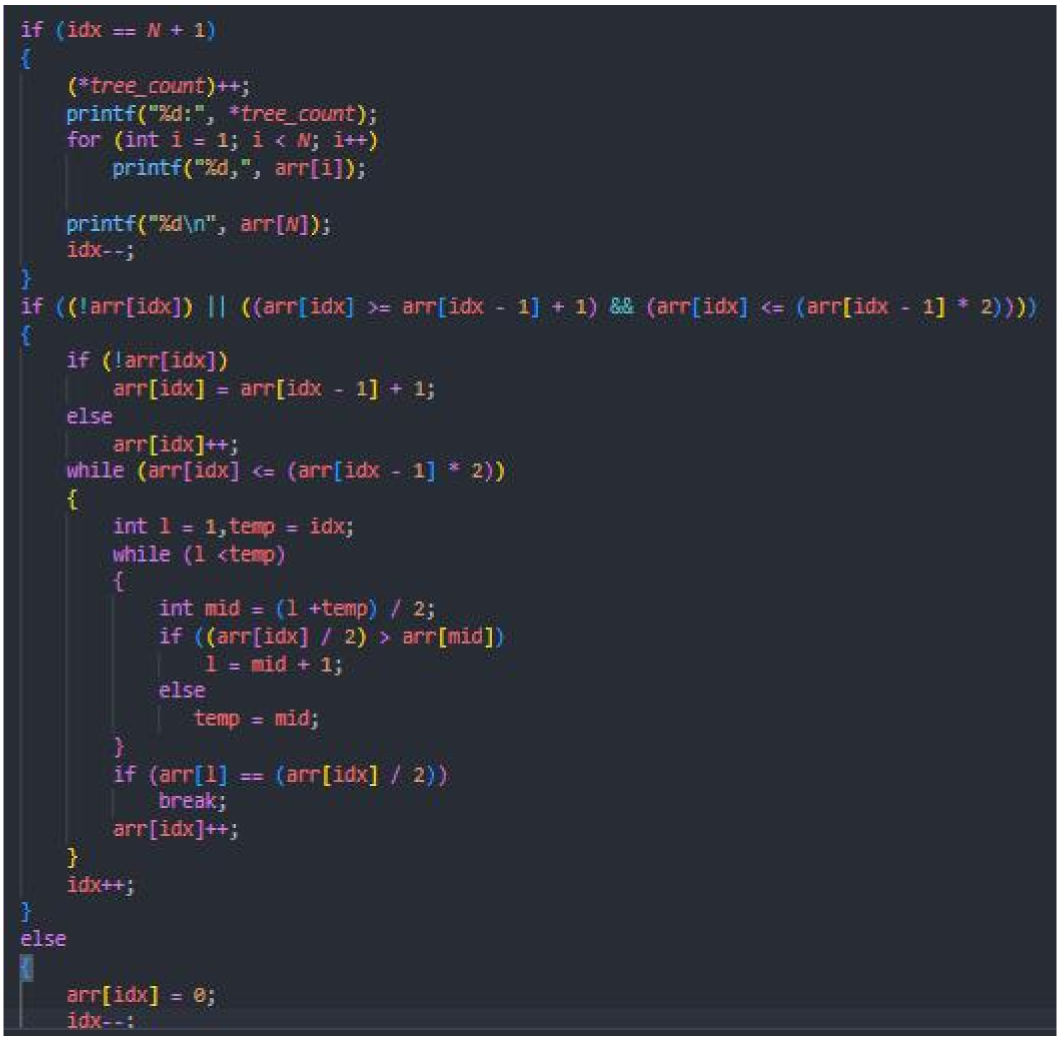


```c
aver = (float)height / count;
printf("*tree_count is %d when N is %d\n", count, N);
if (count == c)
    printf("N为%d时卡塔兰数为%d,符合\n", N, c);
else
    printf("N为%d时卡塔兰数为%d,不符合\n", N, c);
printf("平均高度 %lf\n", aver);
printf("log2N is %lf\n", log(N) / log(2));
```

计算平静高度和 log2n 并输出

迭代法：

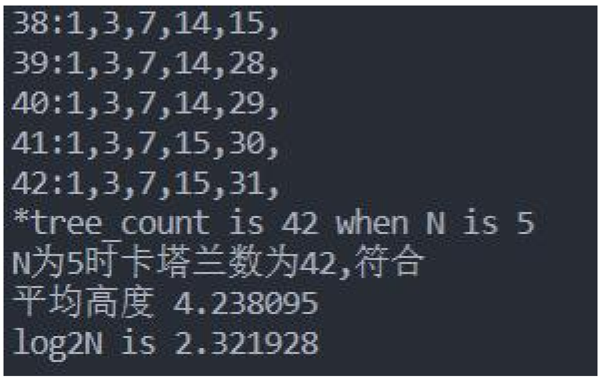
让 idx 循环递增，若填满则使 idx 递减，重新填入。

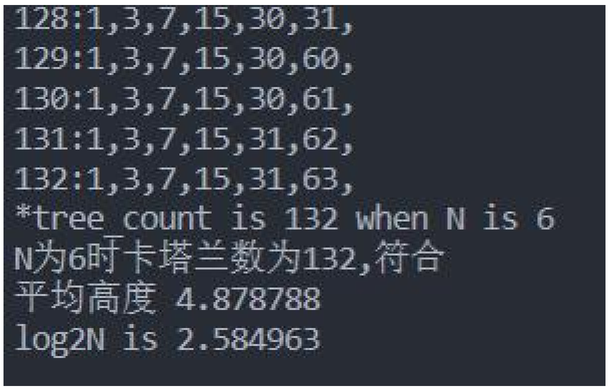


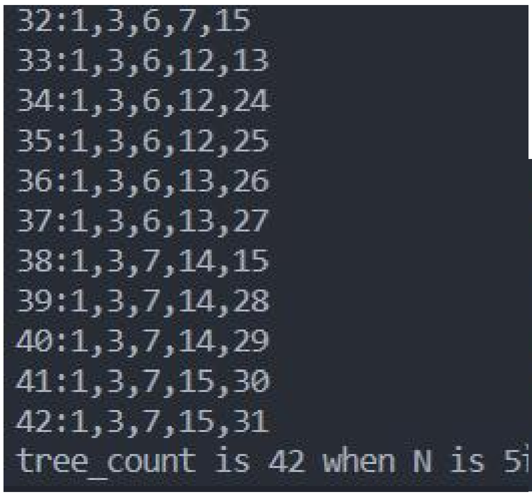
```c
if (idx == N + 1)
{
    (*tree_count)++;
    printf("%d:", *tree_count);
    for (int i = 1; i < N; i++)
        printf("%d,", arr[i]);

    printf("%d\n", arr[N]);
    idx--;
}
if ((!arr[idx]) || ((arr[idx] >= arr[idx - 1] + 1) && (arr[idx] <= (arr[idx - 1] * 2))))
{
    if (!arr[idx])
        arr[idx] = arr[idx - 1] + 1;
    else
        arr[idx]++;
    while (arr[idx] <= (arr[idx - 1] * 2))
    {
        int l = 1,temp = idx;
        while (l <temp)
        {
            int mid = (l +temp) / 2;
            if ((arr[idx] / 2) > arr[mid])
                l = mid + 1;
            else
               temp = mid;
        }
        if (arr[l] == (arr[idx] / 2))
            break;
        arr[idx]++;
    }
    idx++;
}
else
{
    arr[idx] = 0;
    idx--;
```

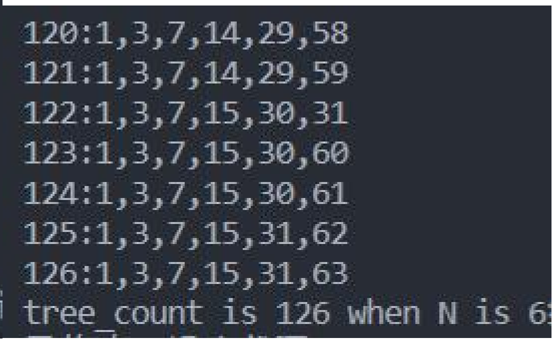
其余详见源码。

四、实验结果及分析

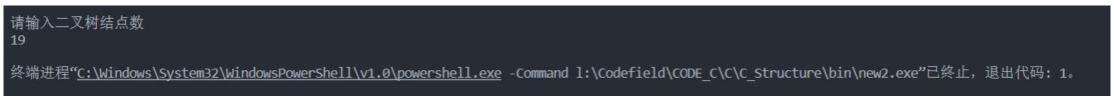

```
38:1,3,7,14,15,
39:1,3,7,14,28,
40:1,3,7,14,29,
41:1,3,7,15,30,
42:1,3,7,15,31,
*tree_count is 42 when N is 5
N为5时卡塔兰数为42,符合
平均高度 4.238095
log2N is 2.321928
```


```
128:1,3,7,15,30,31,
129:1,3,7,15,30,60,
130:1,3,7,15,30,61,
131:1,3,7,15,31,62,
132:1,3,7,15,31,63,
*tree_count is 132 when N is 6
N为6时卡塔兰数为132,符合
平均高度 4.878788
log2N is 2.584963
```
递归分别输入 5,6 得

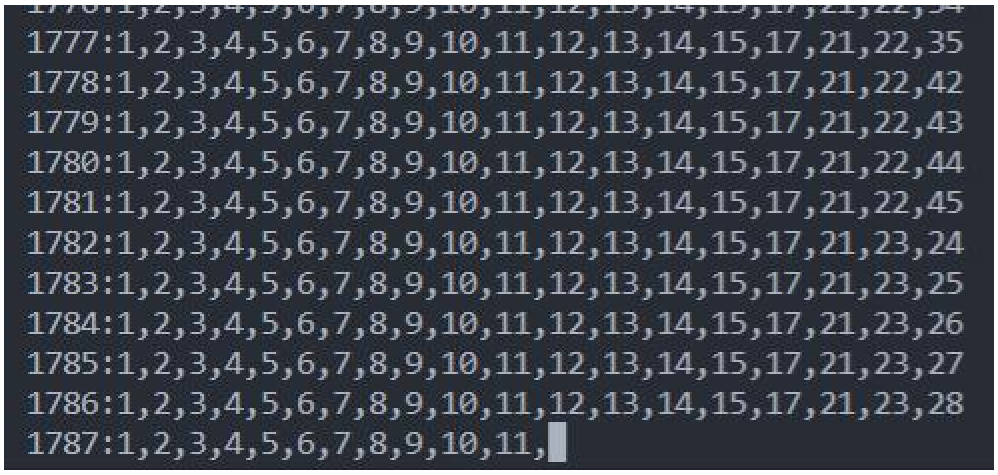

```
32:1,3,6,7,15
33:1,3,6,12,13
34:1,3,6,12,24
35:1,3,6,12,25
36:1,3,6,13,26
37:1,3,6,13,27
38:1,3,7,14,15
39:1,3,7,14,28
40:1,3,7,14,29
41:1,3,7,15,30
42:1,3,7,15,31
tree_count is 42 when N is 5
```


```
120:1,3,7,14,29,58
121:1,3,7,14,29,59
122:1,3,7,15,30,31
123:1,3,7,15,30,60
124:1,3,7,15,30,61
125:1,3,7,15,31,62
126:1,3,7,15,31,63
tree_count is 126 when N is 6
```
迭代输入 5,6

五、总结


```
请输入二叉树结点数
19

终端进程“C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -Command l:\Codefield\CODE_C\C\C_Structure\bin\new2.exe”已终止，退出代码: 1。
```

递归算法 n 比较大时，数组形状数过多导致内存溢出



```
1776:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,22,34
1777:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,22,35
1778:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,22,42
1779:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,22,43
1780:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,22,44
1781:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,22,45
1782:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,23,24
1783:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,23,25
1784:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,23,26
1785:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,23,27
1786:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,21,23,28
1787:1,2,3,4,5,6,7,8,9,10,11,
```

迭代算法会不停计算

实验已完成。

结果准确。