C++方向编程题答案

答案说明：

> 大家如果对本次题目或者答案有问题，可以联系下方的出题老师答疑。
>
> **出题老师：**
>
> 选择题：张文超　qq：3627274478
>
> 代码题：时亮益　qq：569334855

第七周

day40

> 771发邮件
>
> 链接：https://www.nowcoder.com/questionTerminal/95e35e7f6ad34821bc2958e37c08918b
>
> **【题目解析】**
>
> 该题为经典的问题：**错排问题**
>
> **【解题思路】**
>
> 用A、B、C……表示写着n位友人名字的信封，a、b、c……表示n份相应的写好的信纸。把错装的总数为记作D(n)。假设把**a错装进B里了**，包含着这个错误的一切错装法分两类：
>
> - b装入A里，这时每种错装的其余部分都与A、B、a、b无关，应有D(n - 2)种错装法。
> - b装入A、B之外的一个信封，这时的装信工作实际是把（除a之外的）n - 1份信纸b、c ……装入（除B以外的）n - 1个信封A、C……，显然这时装错的方法有D(n - 1)种。
>
> 总之在a装入B的错误之下，共有错装法D(n - 2) + D(n - 1)种。
>
> a装入C，装入D……的n - 2种错误之下，同样都有D(n - 1) + D(n - 2)种错装法，因此D(n) = (n - 1)[D(n - 1) + D(n - 2)]
>
> D(n) = (n-1) [D(n-2) + D(n-1)]
>
> 特殊地，D(1) = 0, D(2) = 1.

```
#include <iostream>
using namespace std;

int main()
{
 long long D[21] = {0,0,1};
 for(int i = 3; i < 21; ++i)
 {
     D[i] = (i-1)*(D[i-1] + D[i-2]);
 }

 int n;
 while(cin >> n)
 {
     cout<<D[n]<<endl;
 }
```

```
 return 0;
}
```

805 最长上升子序列

https://www.nowcoder.com/questionTerminal/d83721575bd4418eae76c916483493de

【题目解析】

在一个序列中找最长递增子序列，动态规划的典型应用，详细见解题思路。

【解题思路】

动态规划的难点在于定义数组和创建“状态转移方程”。

1. 定义height来存储数据，**f[i]为以height[i]结尾的元素的最长上升子序列元素个数**，初始时将f所有内容全部初始化成1，因为子序列中至少包含一个元素。
2. 定义"状态转移方程"

一开始先将f中的数据全部置为1，因为最小的子序列长度为1

然后**对于每个height[i]，通过遍历height[0]~~height[i-1]之间的数据，如果在该区间中找到一个height[j]比height[i]小的元素，开始比较f[j]+1和f[i]的大小，如果f[j]+1>f[j]则更新f[i]**,因此：

- 当height[i] > height[j]: **f[i] = max(f[i], f[j]+1)**
- 当height[i] <= height[j]：继续取下一个数据

```
#include <iostream>
using namespace std;

#include <vector>


int LIS(vector<int>& array, int n)
{
 // 起始状态：每个自取件中都包含有一个元素，则默认最长上升子序列就是1
 vector<int> dp(n, 1);

 // 求以array[i]元素结尾的最长子序列
 int result = 1;
 for(int i = 1; i < n; ++i)
 {
     // 将array[0] ~ array[i-1]之间的每一个元素与array[i]比较
     // array[i] > array[j]: dp[i] = max(dp[i], dp[j]+1)
     for(int j = 0; j < i; ++j)
     {
         if(array[i] > array[j])
         {
             dp[i] = max(dp[i], dp[j]+1);
         }
     }

     result = max(result, dp[i]);
 }

 // 求状态的最大值
 return result;
```

```
}

int main()
{
 int n;
 while(cin>>n)
 {
     // 接收被测试数组中的每一个数据
     vector<int> array(n);
     for(int i = 0; i < n; ++i)
     {
         cin>>array[i];
     }

     // 在array数组中找最长上升子序列
     cout<<LIS(array, n)<<endl;
 }
 return 0;
}
```