

```
1 #include<iostream>
 2 using namespace std;
                                           带入小数据
 3 const int NUM=5;
 4 pint r(int n){
                                        输入
                                                   1
        int i;
 5
                                        输出
                                                   1
 6
        if(n<=NUM)</pre>
             return n;
 8
        for(i=1;i<=NUM;i++)</pre>
             if(r(n-i)<0)
                                        输入
                                                   5
                 return i;
10
                                        输出
                                                   5
11
        return -1;
12<sup>1</sup>}
13 pint main(){
                                        输入
        int n;
14
                                                   6
        cin>>n;
15
                                        输出
                                                   -1
        cout<<r(n)<<endl;</pre>
16
        return 0;
17
18<sup>1</sup>}
```

```
1 #include<iostream>
 2 using namespace std;
 3 const int NUM=5;
 4 int r(int n){
 5
        int i;
 6
         if(n<=NUM)</pre>
              return n;
         for(i=1;i<=NUM;i++)</pre>
 8
              if(r(n-i)<0)
10
                   return i;
        return -1;
11
12<sup>1</sup>}
13 pint main(){
        int n;
14
        cin>>n;
15
        cout<<r(n)<<endl;</pre>
16
        return 0;
17 |
18<sup>1</sup>}
```

确定递归边界

找出递归的状态 依赖关系

> 依赖i, r(n-i)<0返回i

```
1 #include<iostream>
                                                  n=6
 2 using namespace std;
 3 const int NUM=5;
                                                        r(n-i)
 4pint r(int n){
        int i;
 5
                                                          5
         if(n<=NUM)</pre>
 6
             return n;
                                             2
                                                          4
        for(i=1;i<=NUM;i++)</pre>
 8
              if(r(n-i)<0)
                                             3
                                                          3
10
                   return i;
                                                          2
                                             4
        return -1;
11
12<sup>1</sup>}
                                             5
13 pint main(){
        int n;
14
                                                 返回-1
15
        cin>>n;
16
        cout<<r(n)<<endl;</pre>
        return 0;
17
18<sup>1</sup>}
```

```
1 #include<iostream>
                                                  n=7
 2 using namespace std;
 3 const int NUM=5;
                                                        r(n-i)
 4pint r(int n){
        int i;
 5
                                                         -1
         if(n<=NUM)</pre>
 6
             return n;
        for(i=1;i<=NUM;i++)</pre>
                                                 返回1
 8
             if(r(n-i)<0)
 9
                  return i;
10
        return -1;
11
12<sup>1</sup>}
13 pint main(){
        int n;
14
15
        cin>>n;
16
        cout<<r(n)<<endl;</pre>
        return 0;
17
18<sup>1</sup>}
```

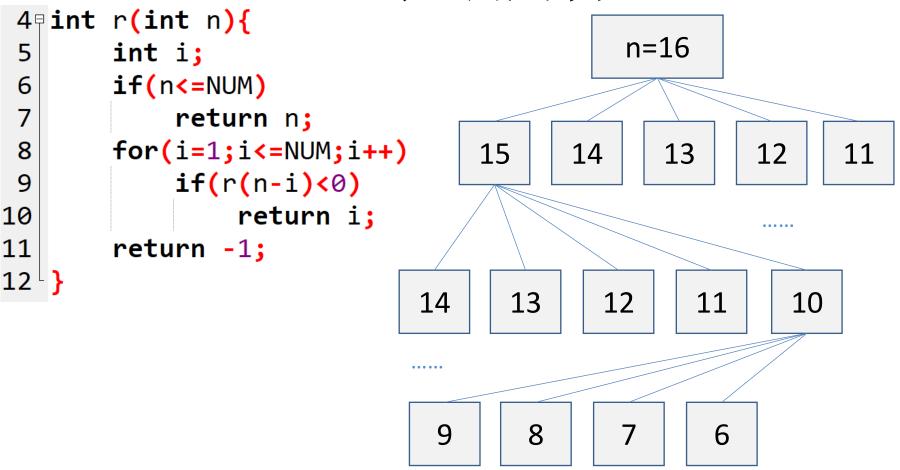
```
1 #include<iostream>
 2 using namespace std;
 3 const int NUM=5;
 4pint r(int n){
 5
         int i;
 6
         if(n<=NUM)</pre>
             return n;
 8
         for(i=1;i<=NUM;i++)</pre>
              if(r(n-i)<0)
                   return i;
10
11
        return -1;
12<sup>1</sup>}
13 pint main(){
         int n;
14
15
         cin>>n;
16
         cout<<r(n)<<endl;</pre>
        return 0;
17 |
18<sup>1</sup>}
```

请计算n=16的结果

返回4

以6为周期,返回 n%6的余数,如果 余数为0,返回-1

递归调用树



太多冗余结点,可以从递归边界反推结果

使用表格递推

n=16

n	r(n)	n	r(n)
1	1	9	3
2	2	10	4
3	3	11	5
4	4	12	-1
5	5	13	1
6	-1	14	2
7	1	15	3
8	2	16	4

```
1 #include<iostream>
 2 using namespace std;
 3 const int NUM=5;
 4pint r(int n){
 5
         int i;
 6
         if(n<=NUM)</pre>
             return n;
         for(i=1;i<=NUM;i++)</pre>
 8
             if(r(n-i)<0)
                  return i;
10
11
        return -1;
12<sup>1</sup>}
13 pint main(){
14
         int n;
15
        cin>>n;
16
        cout<<r(n)<<endl;</pre>
17
        return 0;
18<sup>1</sup>}
```

判断 若输入的数值为负,程序输入和输出一致()。

```
1 #include<iostream>
 2 using namespace std;
 3 const int NUM=5;
 4pint r(int n){
 5
        int i;
 6
         if(n<=NUM)</pre>
             return n;
 8
         for(i=1;i<=NUM;i++)</pre>
             if(r(n-i)<0)
                  return i;
10
11
        return -1;
12<sup>1</sup>}
13 pint main(){
14
         int n;
15
        cin>>n;
16
        cout<<r(n)<<endl;</pre>
17
        return 0;
18<sup>1</sup>}
```

```
1 #include<iostream>
 2 using namespace std;
 3 const int NUM=5;
 4pint r(int n){
 5
        int i;
 6
        if(n<=NUM)</pre>
             return n;
        for(i=1;i<=NUM;i++)</pre>
             if(r(n-i)<0)
                  return i;
10
11
        return -1;
12<sup>1</sup>}
13 pint main(){
14
        int n;
15
        cin>>n;
16
        cout<<r(n)<<endl;</pre>
17
        return 0;
18<sup>1</sup>}
```

判断 可以将第16行改写为cout<<r(n+6)<<endl;{ ,程序结果不变()。

```
1 #include<iostream>
 2 using namespace std;
 3 const int NUM=5;
 4pint r(int n){
 5
        int i;
 6
        if(n<=NUM)</pre>
             return n;
 8
        for(i=1;i<=NUM;i++)</pre>
             if(r(n-i)<0)
 9
                  return i;
10
11
        return -1;
12<sup>\[\]</sup>
13 pint main(){
14
        int n;
15
        cin>>n;
16
        cout<<r(n)<<endl;</pre>
17
        return 0;
18<sup>1</sup>}
```

```
选择 输入7,输出是( )。
A.7 B.1 C.4 D.5
```

```
1 #include<iostream>
 2 using namespace std;
 3 const int NUM=5;
 4pint r(int n){
        int i;
 6
        if(n<=NUM)</pre>
             return n;
 8
        for(i=1;i<=NUM;i++)</pre>
 9
             if(r(n-i)<0)
                  return i;
10
11
        return -1;
12<sup>\[\]</sup>
13 pint main(){
14
        int n;
15
        cin>>n;
16
        cout<<r(n)<<endl;</pre>
17
        return 0;
18<sup>1</sup>}
```

```
选择 输入16,程序输出为( )。
A.2 B.-1 C.4 D.16
```

```
1 #include<iostream>
 2 using namespace std;
 3 const int NUM=5;
 4pint r(int n){
        int i;
 6
        if(n<=NUM)</pre>
             return n;
 8
        for(i=1;i<=NUM;i++)</pre>
             if(r(n-i)<0)
 9
                  return i;
10
11
        return -1;
12<sup>\[\]</sup>
13 pint main(){
14
        int n;
15
        cin>>n;
16
        cout<<r(n)<<endl;</pre>
17
        return 0;
18<sup>1</sup>}
```

```
选择 程序第11行改写成"return -2;",程序输出结果改变有()。

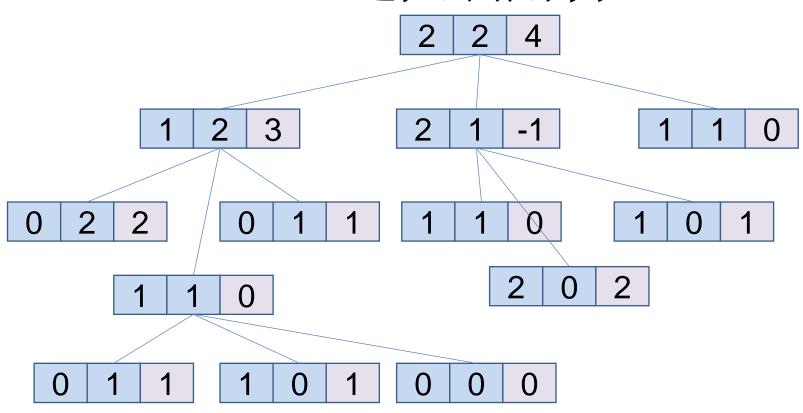
A.7 B.5 C.12 D.17
```

```
带入小数据
   #include<iostream>
   #include<iomanip>
                                                       输入
                                                                     00
    using namespace std;
    int n,m;
 5 pint findans(int n, int m) {
                                                       输出
                                                                     0
        if(n==0) return m;
 6
 7
        if(m==0) return n\%3;
 8
        return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
 9
                                                       输入
                                                                     10
10 int main() {
11
        cin>>n>>m;
                                                       输出
        for(int i=0;i<=n;i++) {</pre>
12 □
                                                                     0
13 申
            for(int j=0;j<=m;j++) {</pre>
                cout<<setw(3)<<findans(i,j);</pre>
14
15
16
            cout<<endl;
17
                                                       输入
                                                                     0 1
18
        return 0;
19 <sup>L</sup> }
                                                       输出
                                                                     0 1
```

```
带入小数据
   #include<iostream>
 2 #include<iomanip>
                                                      输入
                                                                   1 1
   using namespace std;
   int n,m;
 5 pint findans(int n, int m) {
                                                      输出
                                                                   0 1
       if(n==0) return m;
 6
                                                                   10
 7
       if(m==0) return n\%3;
        return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
 9
10 int main() {
11
        cin>>n>>m;
                                                      输入
                                                                   22
        for(int i=0;i<=n;i++) {</pre>
12 □
13 □
            for(int j=0;j<=m;j++) {</pre>
                                                      输出
                cout<<setw(3)<<findans(i,j);</pre>
14
15
                                                                   1 0 3
16
            cout<<endl;
17
18
        return 0;
```

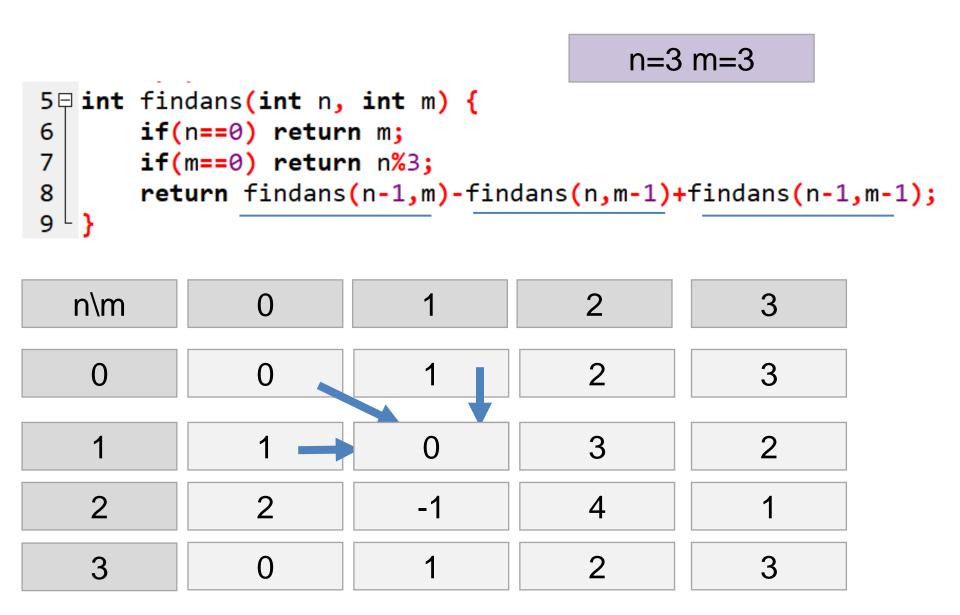
19 ^L }

递归调用树



```
int findans(int n, int m) {
   if(n==0) return m;
   if(m==0) return n%3;
   return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
}
```

使用表格递推



```
1 #include<iostream>
 2 #include<iomanip>
 3 using namespace std;
 4 int n,m;
 5 pint findans(int n, int m) {
        if(n==0) return m;
 6
 7
         if(m==0) return n\%3;
 8
         return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
 9
10 □ int main() {
        cin>>n>>m;
11
12 □
         for(int i=0;i<=n;i++) {</pre>
13 □
             for(int j=0;j<=m;j++) {</pre>
14
                  cout<<setw(3)<<findans(i,j);</pre>
15
16
             cout<<endl;
17
18
        return 0;
19 <sup>L</sup> }
```

判断 输入的n和m如果是负数,输出结果是0()

```
#include<iostream>
 2 #include<iomanip>
 3 using namespace std;
 4 int n,m;
 5 pint findans(int n, int m) {
        if(n==0) return m;
 6
 7
         if(m==0) return n\%3;
 8
         return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
 9
10 □ int main() {
        cin>>n>>m;
11
12 □
         for(int i=0;i<=n;i++) {</pre>
13 □
             for(int j=0;j<=m;j++) {</pre>
14
                  cout<<setw(3)<<findans(i,j);</pre>
15
16
             cout<<endl;
17
18
        return 0;
19 <sup>L</sup> }
```

判断

当m=0时,无论输入n的数值是多大,findans(n,m)的返回值不会超过2()。

```
#include<iostream>
 2 #include<iomanip>
 3 using namespace std;
 4 int n,m;
 5 pint findans(int n, int m) {
        if(n==0) return m;
 6
 7
         if(m==0) return n\%3;
 8
         return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
 9
10 □ int main() {
        cin>>n>>m;
11
12 □
         for(int i=0;i<=n;i++) {</pre>
13 □
             for(int j=0;j<=m;j++) {</pre>
14
                  cout<<setw(3)<<findans(i,j);</pre>
15
16
             cout<<endl;
17
18
        return 0;
19 <sup>L</sup> }
```

判断

当n=0时,无论输入m的数值是多大,findans(n,m)的返回值不会超过2()。

```
#include<iostream>
 2 #include<iomanip>
 3 using namespace std;
 4 int n,m;
 5 pint findans(int n, int m) {
        if(n==0) return m;
 6
 7
         if(m==0) return n\%3;
 8
         return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
 9
10 □ int main() {
        cin>>n>>m;
11
12 □
         for(int i=0;i<=n;i++) {</pre>
13 □
             for(int j=0;j<=m;j++) {</pre>
14
                  cout<<setw(3)<<findans(i,j);</pre>
15
16
             cout<<endl;
17
18
        return 0;
19 <sup>L</sup> }
```

判断

当m=x时,无论输入n的数值是多大,findans(n,m)的返回值不会超过x()。

```
1 #include<iostream>
 2 #include<iomanip>
    using namespace std;
    int n,m;
 5 pint findans(int n, int m) {
        if(n==0) return m;
 6
 7
        if(m==0) return n%3;
        return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
 8
 9
10 □ int main() {
11
        cin>>n>>m;
        for(int i=0;i<=n;i++) {</pre>
12 □
13 □
             for(int j=0; j<=m; j++) {</pre>
14
                 cout<<setw(3)<<findans(i,j);</pre>
15
             cout<<endl;
16
17
18
        return 0;
19 <sup>L</sup> }
```

选择 输入n=3, m=5时,输出共有()个数字。

A. 15 B. 20 C. 25 D. 24

```
1 #include<iostream>
     2 #include<iomanip>
        using namespace std;
        int n,m;
     5 p int findans(int n, int m) {
            if(n==0) return m;
     6
     7
            if(m==0) return n%3;
            return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
     8
     9
    10 □ int main() {
    11
            cin>>n>>m;
    12 □
            for(int i=0;i<=n;i++) {</pre>
    13 □
                for(int j=0; j<=m; j++) {</pre>
    14
                    cout<<setw(3)<<findans(i,j);</pre>
    15
                cout<<endl;
    16
    17
    18
            return 0;
    19 <sup>L</sup> }
       输入n=3, m=5时,输出的数字中最大值是()。
选择
        A. -1
                       B. 3
                                     C. 5
                                                      D. 6
```

```
阅读程序
   #include<iostream>
2
   using namespace std;
                                         带入小数据
3
   int n,i,j,a[100][100];
                                       输入
   int solve(int x,int y){
5
        int u, v;
6
        if(x==n) return a[x][y];
        u=solve(x+1,y);
                                       输出
8
        v=solve(x+1,y+1);
        if(u>v) return a[x][y]+u;
9
                                       输入
        else return a[x][y]+v;
10
11
12 pint main(){
        cin>>n;
13
                                       输出
                                                3
14
        for(i=1;i<=n;i++)
15
            for(j=1;j<=i;j++) cin>>a[i][j];
16
        cout<<solve(1,1)<<endl;</pre>
17
        return 0;
18
```

```
阅读程序
   #include<iostream>
                                     i代表二维数组行号
   using namespace std;
2
   int n,i,j,a[100][100]; ←
3
                                     i代表二维数组列号
   int solve(int x,int y){
5
       int u, v;
                                      n是输入的总行数
6
       if(x==n) return a[x][y];
       u=solve(x+1,y); \leftarrow
                                     a是输入的二维数组
       v=solve(x+1,y+1); ←
8
                                    u是x+1行y列solve值
       if(u>v) return a[x][y]+u;
9
       else return a[x][y]+v;
10
                                   v是x+1行y+1列solve值
11
12 □
   int
       main(){
       cin>>n;
13
14
       for(i=1;i<=n;i++)
15
           for(j=1;j<=i;j++) cin>>a[i][j];←
16
       cout<<solve(1,1)<<endl;</pre>
17
       return 0;
18
```

```
阅读程序
   #include<iostream
   using namespace std;
3
   int n,i,j,a[100][100];
   int solve(int x,int y){
5
        int u, v;
6
        if(x==n) return a[x][y];
        u=solve(x+1,y);
8
        v=solve(x+1,y+1);
                                        注意内层for循环
        if(u>v) return a[x][y]+u;
9
                                         列数i <= 行数i
        else return a[x][y]+v;
10
11
12□ int main(){
13
        cin>>n;
        for(i=1;i<=n;i++) <
14
            for(j=1;j<=i;j++) cin>>a[i][j];
15
16
        cout<<solve(1,1)<<endl;</pre>
17
        return 0;
18
```

```
阅读程序
   #include<iostream>
   using namespace std;
2
   int n,i,j,a[100][100];
3
                                      solve(x,y)是递归函数
   int solve(int x, int y) \{\leftarrow
5
        int u, v;
                                      当行数=n 返回a[x][y]
        if(x==n) return a[x][y];
6
       u=solve(x+1,y);
                                         否则递归调用
8
        v=solve(x+1,y+1);
                                         solve(x+1,y)
        if(u>v) return a[x][y]+u;
9
                                        solve(x+1,y+1)
       else return a[x][y]+v;
10
11
                                     并将两次递归返回值中
12 □
       main(){
   int
                                      的较大者与a[x][y]之和
13
        cin>>n;
                                             返回
        for(i=1;i<=n;i++)
14
15
            for(j=1;j<=i;j++) cin>>a[i][j];
16
        cout<<solve(1,1)<<endl;</pre>
17
        return 0;
18
```

```
阅读程序
   #include<iostream>
   using namespace std;
 2
   int n,i,j,a[100][100];
3
                                            找到入口
   int solve(int x,int y){
5
        int u, v;
                                   solve(1,1)依赖哪几个函数
6
        if(x==n) return a[x][y];
                                     solve(2,1) solve(2,2)
        u=solve(x+1,y);
8
        v=solve(x+1,y+1);
                                   solve(2,1)依赖哪几个函数
        if(u>v) return a[x][y]+u;
9
        else return a[x][y]+v;
10
                                     solve(3,1) solve(3,2)
11
                                   solve(3,1)依赖哪几个函数
   int main(){
12 □
        cin>>n;
13
                                     solve(4,1) solve(4,2)
14
        for(i=1;i<=n;i++)
15
            for(j=1;j<=i;j++) cin>>a[i][j];
        cout<<solve(1,1)<<endl;</pre>
16
17
        return 0;
18
```

关键语句

二维递推

solve(x,y)|y=1y=2 | y=3 y=4 x=1x=2x=3x=4x=5

输入: 5 2 -1 4 2 -1 -2 -1 6 4 0 3 2 -1 5 8

请填写solve()二维表格

限时8分钟

递归问题填表顺序:

从"归"的一层(x==n那一层)往回退 逐行往上填表 最后得出solve(1,1)

```
阅读程序
   #include<iostream>
   using namespace std;
   int n,i,j,a[100][100];
                                        递归函数solve
4 p int solve(int x,int y){
                                      ① 在各层 solve(x,y)
 5
        int u, v;
                                  是solve(x+1,y)和solve(x+1,y+1)
        if(x==n) return a[x][y];
 6
                                      中较大者与a[x][y]的和
        u=solve(x+1,y);
                                    ② 当x==n 返回a[x][y]本身
        v=solve(x+1,y+1);
 8
        if(u>v) return a[x][y]+u;
        else return a[x][y]+v;
10
        main(){
13
        cin>>n;
                                     读入三角形数字阵列
        for(i=1;i<=n;i++)
14
            for(j=1;j<=i;j++) cin>>a[i][j];
15
        cout<<solve(1,1)<<endl;</pre>
16
                                  从solve(1,1)开始递归调用
17
        return 0;
18
```

```
using namespace std;阅读程序
    int n,i,j,a[100][100];
4 pint solve(int x,int y){
5
        int u, v;
6
        if(x==n) return a[x][y];
        u=solve(x+1,y);
8
        v=solve(x+1,y+1);
        if(u>v) return a[x][y]+u;
        else return a[x][y]+v;
10
12p int main(){
        cin>>n;
13
        for(i=1;i<=n;i++)</pre>
14
15
            for(j=1;j<=i;j++) cin>>a[i][j];
        cout<<solve(1,1)<<endl;</pre>
16
17
        return 0;
18 <sup>∟</sup> }
```

判断 二维数组a通过cin输入了n²次()。

```
using namespace std; 阅读程序
    int n,i,j,a[100][100];
4 pint solve(int x,int y){
5
        int u, v;
6
        if(x==n) return a[x][y];
        u=solve(x+1,y);
8
        v=solve(x+1,y+1);
        if(u>v) return a[x][y]+u;
        else return a[x][y]+v;
10
12p int main(){
13
        cin>>n;
        for(i=1;i<=n;i++)</pre>
14
15
             for(j=1;j<=i;j++) cin>>a[i][j];
        cout<<solve(1,1)<<endl;</pre>
16
17
        return 0;
18 <sup>∟</sup> }
```

判断

当solve函数递归调用到第n行各列的数字时,会直接返回该数字到上一层()。

```
using namespace std;阅读程序
    int n,i,j,a[100][100];
4 pint solve(int x,int y){
5
        int u, v;
6
        if(x==n) return a[x][y];
        u=solve(x+1,y);
8
        v=solve(x+1,y+1);
        if(u>v) return a[x][y]+u;
        else return a[x][y]+v;
10
12 pint main(){
13
        cin>>n;
        for(i=1;i<=n;i++)</pre>
14
15
            for(j=1;j<=i;j++) cin>>a[i][j];
        cout<<solve(1,1)<<endl;</pre>
16
17
        return 0;
18 <sup>∟</sup> }
```

判断

程序7-10行 solve(x, y)返回给上一层是a[x+1][y]和a[x+1][y+1]两数中较小者与a[x][y]的和的结果()

```
using namespace std;阅读程序
   int n,i,j,a[100][100];
4 pint solve(int x,int y){
       int u, v;
5
6
       if(x==n) return a[x][y];
       u=solve(x+1,y);
8
       v=solve(x+1,y+1);
       if(u>v) return a[x][y]+u;
       else return a[x][y]+v;
10
12p int main(){
       cin>>n;
13
       for(i=1;i<=n;i++)</pre>
14
15
            for(j=1;j<=i;j++) cin>>a[i][j];
16
        cout<<solve(1,1)<<endl;</pre>
17
        return 0;
18
      n可以输入的最大值()。
```

选择 A. 100 B. 99 C. 10000 D. 9801

```
using namespace std;阅读程序
   int n,i,j,a[100][100];
4 pint solve(int x,int y){
5
       int u, v;
                                    输入:
6
       if(x==n) return a[x][y];
       u=solve(x+1,y);
8
       v=solve(x+1,y+1);
       if(u>v) return a[x][y]+u;
                                       13
       else return a[x][y]+v;
10
                                    输出结果是()。
12 pint main(){
       cin>>n;
13
       for(i=1;i<=n;i++)</pre>
14
15
            for(j=1;j<=i;j++) cin>>a[i][j];
16
        cout<<solve(1,1)<<endl;</pre>
17
        return 0;
18
      A. 6
           B. 5
```

```
using namespace std;阅读程序
   int n,i,j,a[100][100];
4 pint solve(int x,int y){
                                    输入:
5
        int u, v;
                                        5
6
        if(x==n) return a[x][y];
        u=solve(x+1,y);
8
        v=solve(x+1,y+1);
                                        -14
        if(u>v) return a[x][y]+u;
                                        2 -1 -2
        else return a[x][y]+v;
10
                                        -1640
                                        3 2 - 1 5 8
12 int main(){
                                    输出结果是(
        cin>>n;
13
14
        for(i=1;i<=n;i++)</pre>
15
            for(j=1;j<=i;j++) cin>>a[i][j];
16
        cout<<solve(1,1)<<endl;</pre>
17
        return 0;
18
       A. 14
              B. 2
```

选择 C. 1 D. 5

最大公约数之和

下列程序想要求解整数n的所有约数两两之间最大公约数的和对10007求余后的值,试补全程序。

举例来说,4的所有约数是1,2,4。1和2的最大公约数为1;2和4的最大公约数为2;1和4的最大公约数为1。于是答案为1+2+1=4。

要求 getDivisor 函数的复杂度为 $P(\sqrt{n})$,gcd 函数的复杂度为 $P(\log \max(b,c))$ 。

手算样例

输入样例:

6

输出多少?

输出样例:

9

6的约数有1,2,3,6

- 1,2最大公约数1
- 1,3最大公约数1
- 1,6最大公约数1
- 2,3最大公约数1
- 2,6最大公约数2
- 3,6最大公约数3

手算样例

输入样例:

9

输出多少?

输出样例:

5

```
#include <iostream>
     using namespace std;
     const int N = 110000, P = 10007;
     int n;
     int a[N], len;
     int ans:
     void getDivisor() {
          len = 0;
 8
         for (int i = 1; ____(1) ___ <= n; ++i)
if (n % i == 0) {
 9
10 -
                  a[++len] = i;
11
                  if (___(2) ___!= i)
a[++len] = n / i;
12
13
14
15
     int gcd(int a, int b) {
16 -
17 -
          if (b == 0) {
18
19
          return gcd(b, (4) );
20
21
     int main() {
22
23
          cin >> n;
24
          getDivisor();
25
          ans = 0;
26
          for (int i = 1; i <= len; ++i) {
27
              for (int j = i + 1; j \leftarrow len; ++j) {
28
                  ans = ( (5) ) % P;
29
30
31
          cout << ans << endl;
32
          return 0;
33
```

识别变量

常见变量名 翻译循环变量 根据变量名的英文推断

2 找出关键语句

控制结构(for, if) 常见算法的基本操作 函数参数、返回值

3 理解代码段作用

翻译解释代码段

```
#include <iostream>
1
                                                            解释变量的作用
     using namespace std;
     const int N = 110000, P = 10007;
     int n;
     int a[N], len;
     int ans;
                                                                         输入
                                                      n
     void getDivisor() {
        len = 0;
 8
        for (int i = 1; ____(1) ___ <= n; ++i) if (n % i == 0) {
9
                                                     a[]
                                                                     储存n的因子
10 -
                a[++len] = i;
11
                                                                     n的因子数量
               if (____(2) ____!= i)
a[++len] = n / i;
                                                     len
12
13
14
                                                               两两因子的最大公约数之
15
                                                    ans
                                                                          和
     int gcd(int a, int b) {
16 -
17 -
        if (b == 0) {
18
19
                                             getDivisor()
                                                                    求n的全部因子
        return gcd(b, (4) );
20
21
22 -
     int main() {
23
        cin >> n;
                                                 gcd()
                                                                  求a,b的最大公约数
24
        getDivisor();
25
        ans = 0;
26
        for (int i = 1; i <= len; ++i) {
            for (int j = i + 1; j \leftarrow len; ++j) {
27 -
28
                ans = ( (5) ) % P;
29
30
31
        cout << ans << endl;
32
        return 0;
33
```

关键语句

 $P(\sqrt{n})$ 求**n**的因子

```
i范围从1到\sqrt{n}
     void getDivisor() {
 8
          len = 0;
          for (int i = 1;
                                               <= n; ++i)
 9
               if (n % i == 0) {
10
                   a[++len] = i;
11
12
                       a[++len] = n / i;
13
14
15
```

一次求出2个因子,i和n/i 注意i和n/i相等时不要重复

关键语句 辗转相除法求最大公约数 a是被除数,b是除数 int gcd(int a, int b) { 16 if (b == 0) { 17 余数为0返回除数 18 ____(3)____; 19 return gcd(b, ____(4) 20 21 否则继续算法 除数->被除数 余数->除数

关键语句

```
int main() {
                                   双重循环枚举数组任意两
23
        cin >> n;
                                       个不同的元素
        getDivisor();
24
25
        ans = 0;
26
        for (int i = 1; i <= len; ++i) {
            for (int j = i + 1; j <= len; ++j) {
27
28
                ans = ((5)) % P;
29
                                   累加任意两因子的最大公
30
                                          约数
31
        cout << ans << endl;
32
        return 0;
33
```

(全排列)下面程序的功能是利用递归方法生成从 1 到 n(n<10)的 n 个数的全部可能的排列(不一定按升序输出)。例如,输入 3,则应该输出(每行输出 5 个排列): 123 132 213 231 321 312

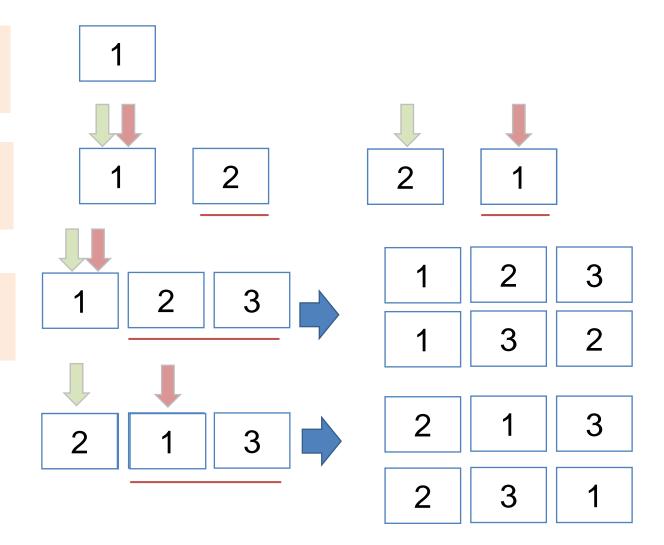
按照字典序的排列 123 132 213 231 312 321 注意结果跟字典序不同

递归生成全排列

1个数字 全排列

2个数字 全排列

3个数字 全排列



递归生成全排列

3个数字 全排列

手算样例

输入

4

输出

1234 1243 1324 1342 1432 1423 2134 2143 2314 2341 2431 2413 3214 3241 3124 3142 3412 3421 4231 4213 4321 4312 4132 4123

```
#include <iostream>
 2 #include <iomanip>
   using namespace std;
   int n,a[10];//a[1],a[2],...,a[n]构成 n 个数的一个排列
    long count=0;//变量 count 记录不同排列的个数,这里用于控制换行
    void perm(int k)
7 □ {
 8
        int j,p,t;
10 🖨
11
            count++;
12
            for(p=1;p<=n;p++)
13
                cout <<setw(1)<<a[p];</pre>
            cout <<" ";
14
           if( 2 )
15
                          cout <<endl;
16
            return;
17
        for(j=k;j<=n;j++)
18
19 🗀
20
           t=a[k];a[k]=a[j];a[j]=t;
21
           ____3___;
t=a[k]; 4
22
23
24
25
    int main()
26 □ {
27
        int i;
28
        cin >>n:
        for(i=1;i<=n;i++) a[i]=i;
29
30
        return 0;
31
32 L }
```

1 识别变量

常见变量名 翻译循环变量 根据变量名的英文推断

2 找出关键语句

控制结构(for, if) 常见算法的基本操作 函数参数、返回值

3 理解代码段作用

翻译解释代码段

```
#include <iostream>
2 #include <iomanip>
 3 using namespace std;
4 int n,a[10];//a[1],a[2],...,a[n]构成 n 个数的一个排列
    long count=0;//变量 count 记录不同排列的个数,这里用于控制换行
    void perm(int k)
7 □ {
8
        int j,p,t;
 9
10 □
            count++;
11
12
            for(p=1;p<=n;p++)
13
                cout <<setw(1)<<a[p];</pre>
            cout <<" ";
14
            if( 2 ) cout <<endl;</pre>
15
16
            return;
17
18
        for(j=k;j<=n;j++)
19 🖨
            t=a[k];a[k]=a[j];a[j]=t;
20
21
            ____3___;
t=a[k]; ____4___;
22
23
24
25
    int main()
26 ₽ {
27
        int i;
28
        cin >>n:
        for(i=1;i<=n;i++) a[i]=i;
29
30
        return 0;
31
32 L }
```

解释变量的作用

k	递归进行全排列的开始位置
j	将开头与第j个元素交换
р	循环变量
t	用于交换两个变量时暂存数 据

```
#include <iostream>
2 #include <iomanip>
 3 using namespace std;
   int n,a[10];//a[1],a[2],...,a[n]构成 n 个数的一个排列
    long count=0;//变量 count 记录不同排列的个数,这里用于控制换行
    void perm(int k)
7 □ {
8
        int j,p,t;
 9
        if(
10 □
11
            count++;
12
            for(p=1;p<=n;p++)
13
                cout <<setw(1)<<a[p];</pre>
            cout <<" ":
14
            if( 2
                           cout <<endl;
15
16
            return:
17
        for(j=k;j<=n;j++)
18
19 🗀
20
            t=a[k];a[k]=a[j];a[j]=t;
21
            ____3___;
t=a[k]; ____4___;
22
23
24
25
    int main()
26 □ {
27
        int i;
28
        cin >>n:
        for(i=1;i<=n;i++) a[i]=i;
29
30
            5 ;
        return 0;
31
32 L }
```

关键语句

递归边界,方案数+1, 打印当前方案

每输出5个方案换行

#include <iostream>

```
关键语句
2 #include <iomanip>
   using namespace std;
   int n,a[10];//a[1],a[2],...,a[n]构成 n 个数的一个排列
   long count=0;//变量 count 记录不同排列的个数,这里用于控制换行
   void perm(int k)
7 □ {
      int j,p,t;
8
10 🖨
          count++;
11
12
          for(p=1;p<=n;p++)
                                               递归生成全排列算法
13
             cout <<setw(1)<<a[p];</pre>
          cout <<" ";
14
                                          将当前位k与(k到n)位交换,
          if( 2 )
15
                      cout <<endl;
16
          return;
                                              归生成k+1到n的全排列
17
      for(j=k;j<=n;j++)
18
19 🗀
                                                交换a[k]和a[j]
20
          t=a[k];a[k]=a[j];a[j]=t;
21
22
                                         递归生成k+1到n的全排列
23
24
25
   int main()
                                             回溯,换回a[k]和a[j]
26 □ {
27
      int i;
28
      cin >>n;
      for(i=1:i<=n:i++) a[i]=i:
29
30
                                                调用递归函数
      return 0;
31
32 L }
```

作业: 完善程序

1999 最大公约数之和

1993 全排列