专题训练

5分钟

阅读程序

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

带入小数据

输入	1
输出	1

输入	5
输出	5

输入	6
输出	-1

阅读程序

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

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

阅读程序

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

n=6

i	r(n-i)
1	5
2	4
3	3
4	2
5	1

返回-1

阅读程序

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

n=7

i	r(n-i)
1	-1

返回1

阅读程序

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

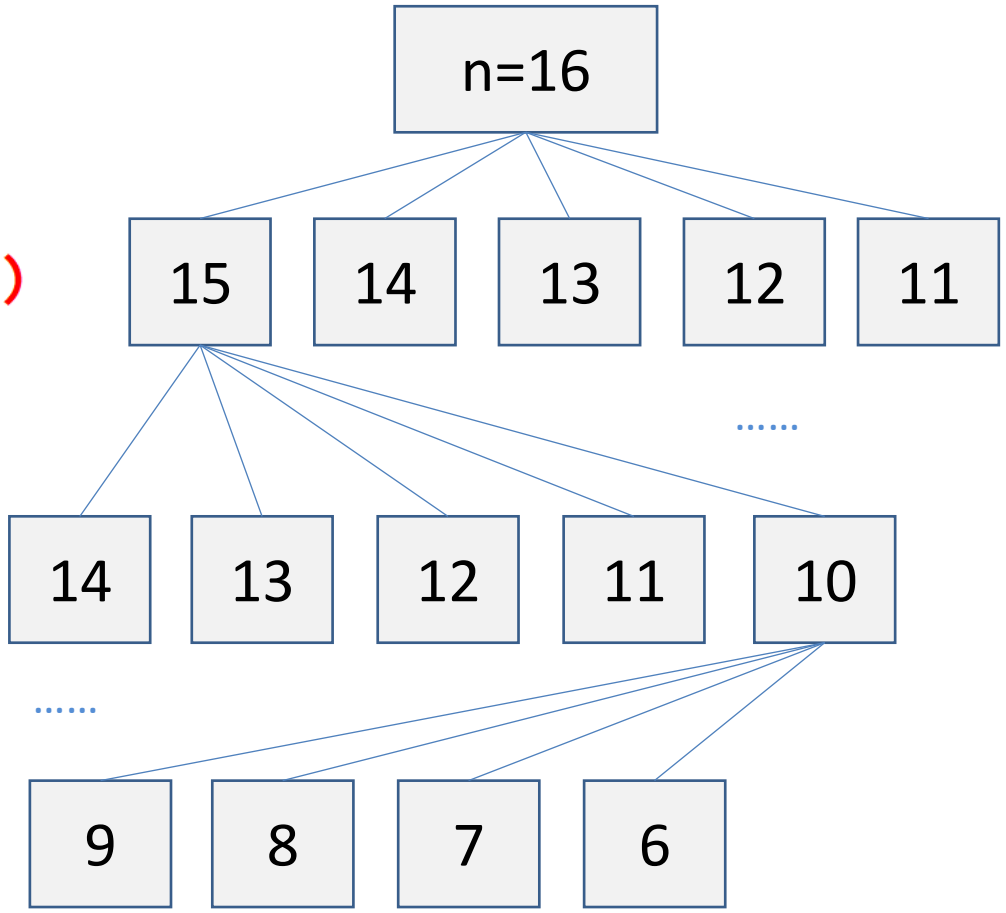
请计算n=16的结果

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

```
4  int r(int n){
5      int i;
6      if(n<=NUM)
7          return n;
8      for(i=1;i<=NUM;i++)
9          if(r(n-i)<0)
10             return i;
11     return -1;
12 }
```



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)
1	1
2	2
3	3
4	4
5	5
6	-1
7	1
8	2

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

阅读程序

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

判断	若输入的数值为负，程序输入和输出一致（　）。

阅读程序

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

判断　程序第3行改写成“const int NUM=11;”，程序运行结果不变（　）。

阅读程序

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

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

阅读程序

```
#include<iostream>
using namespace std;
const int NUM=5;
int r(int n){
    int i;
    if(n<=NUM)
        return n;
    for(i=1;i<=NUM;i++)
        if(r(n-i)<0)
            return i;
    return -1;
}
int main(){
    int n;
    cin>>n;
    cout<<r(n)<<endl;
    return 0;
}
```

选择 输入7，输出是（ ）。

A.7　　B.1　　C. 4　　D. 5

阅读程序

```
#include<iostream>
using namespace std;
const int NUM=5;
int r(int n){
    int i;
    if(n<=NUM)
        return n;
    for(i=1;i<=NUM;i++)
        if(r(n-i)<0)
            return i;
    return -1;
}
int main(){
    int n;
    cin>>n;
    cout<<r(n)<<endl;
    return 0;
}
```

选择 输入16，程序输出为（ ）。

A.2 B.-1 C.4 D.16

阅读程序

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

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

5分钟

阅读程序

```
#include<iostream>
#include<iomanip>
using namespace std;
int n,m;
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);
}
int main() {
    cin>>n>>m;
    for(int i=0;i<=n;i++) {
        for(int j=0;j<=m;j++) {
            cout<<setw(3)<<findans(i,j);
        }
        cout<<endl;
    }
    return 0;
}
```

带入小数据

输入	0 0
输出	0
输入	1 0
输出	0 1
输入	0 1
输出	0 1

阅读程序

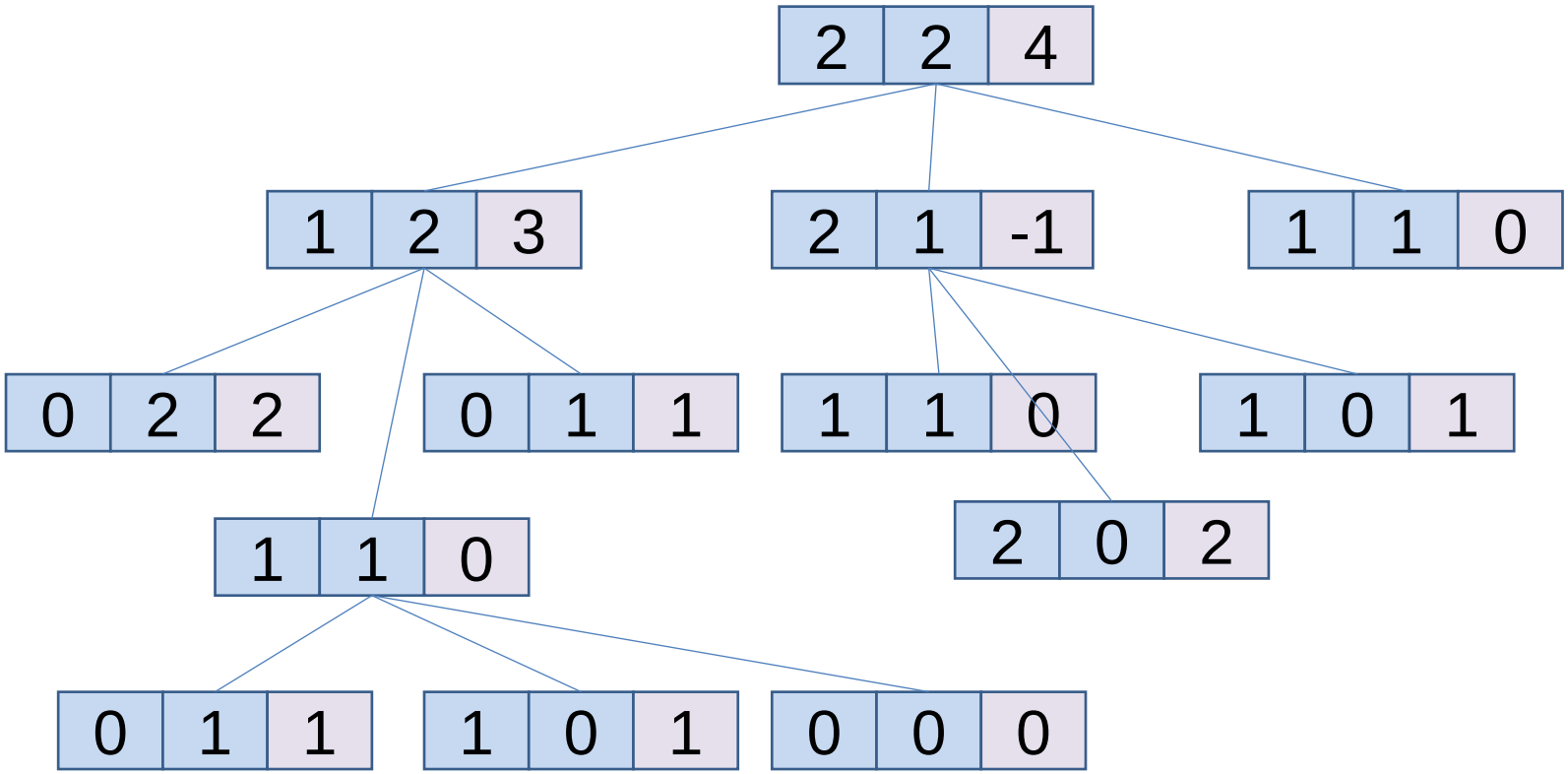
```
#include<iostream>
#include<iomanip>
using namespace std;
int n,m;
int findans(int n, int m) {
    if(n==0) return m;
    if(m==0) return n%3;
    return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
}
int main() {
    cin>>n>>m;
    for(int i=0;i<=n;i++) {
        for(int j=0;j<=m;j++) {
            cout<<setw(3)<<findans(i,j);
        }
        cout<<endl;
    }
    return 0;
}
```

带入小数据

输入	1 1
输出	0 1 1 0

输入	2 2
输出	0 1 2 1 0 3 2 -1 4

递归调用树



n	m	return

```
5  int findans(int n, int m) {
6      if(n==0) return m;
7      if(m==0) return n%3;
8      return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
9  }
```

使用表格递推

n=3 m=3

```
5  int findans(int n, int m) {
6      if(n==0) return m;
7      if(m==0) return n%3;
8      return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
9  }
```

n\m	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	-1	4	1
3	0	1	2	3

阅读程序

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int n,m;
5  int findans(int n, int m) {
6      if(n==0) return m;
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() {
11     cin>>n>>m;
12     for(int i=0;i<=n;i++) {
13         for(int j=0;j<=m;j++) {
14             cout<<setw(3)<<findans(i,j);
15         }
16         cout<<endl;
17     }
18     return 0;
19 }
```

判断　输入的n和m如果是负数，输出结果是0（　）。

阅读程序

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int n,m;
5  int findans(int n, int m) {
6      if(n==0) return m;
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() {
11     cin>>n>>m;
12     for(int i=0;i<=n;i++) {
13         for(int j=0;j<=m;j++) {
14             cout<<setw(3)<<findans(i,j);
15         }
16         cout<<endl;
17     }
18     return 0;
19 }
```

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（　）。

阅读程序

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int n,m;
5  int findans(int n, int m) {
6      if(n==0) return m;
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() {
11     cin>>n>>m;
12     for(int i=0;i<=n;i++) {
13         for(int j=0;j<=m;j++) {
14             cout<<setw(3)<<findans(i,j);
15         }
16         cout<<endl;
17     }
18     return 0;
19 }
```

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（　）。

阅读程序

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int n,m;
5  int findans(int n, int m) {
6      if(n==0) return m;
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() {
11     cin>>n>>m;
12     for(int i=0;i<=n;i++) {
13         for(int j=0;j<=m;j++) {
14             cout<<setw(3)<<findans(i,j);
15         }
16         cout<<endl;
17     }
18     return 0;
19 }
```

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（　）。

阅读程序

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int n,m;
5  int findans(int n, int m) {
6      if(n==0) return m;
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() {
11     cin>>n>>m;
12     for(int i=0;i<=n;i++) {
13         for(int j=0;j<=m;j++) {
14             cout<<setw(3)<<findans(i,j);
15         }
16         cout<<endl;
17     }
18     return 0;
19 }
```

选择 输入n=3，m=5时，输出共有(　　)个数字。

A. 15　　B. 20　　C. 25　　D. 24

阅读程序

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int n,m;
5  int findans(int n, int m) {
6      if(n==0) return m;
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() {
11     cin>>n>>m;
12     for(int i=0;i<=n;i++) {
13         for(int j=0;j<=m;j++) {
14             cout<<setw(3)<<findans(i,j);
15         }
16         cout<<endl;
17     }
18     return 0;
19 }
```

选择 输入n=3，m=5时，输出的数字中最大值是(　)。

A. -1　　B. 3　　C. 5　　D. 6

5分钟

阅读程序

```
1  #include<iostream>
2  using namespace std;
3  int n,i,j,a[100][100];
4  int solve(int x,int y){
5      int u,v;
6      if(x==n) return a[x][y];
7      u=solve(x+1,y);
8      v=solve(x+1,y+1);
9      if(u>v) return a[x][y]+u;
10     else return a[x][y]+v;
11 }
12 int main(){
13     cin>>n;
14     for(i=1;i<=n;i++)
15         for(j=1;j<=i;j++) cin>>a[i][j];
16     cout<<solve(1,1)<<endl;
17     return 0;
18 }
```

带入小数据

输入	1 1
输出	1
输入	2 1 1 2
输出	3

解释变量作用

阅读程序

```
1  #include<iostream>
2  using namespace std;
3  int n,i,j,a[100][100];
4  int solve(int x,int y){
5      int u,v;
6      if(x==n) return a[x][y];
7      u=solve(x+1,y);
8      v=solve(x+1,y+1);
9      if(u>v) return a[x][y]+u;
10     else return a[x][y]+v;
11 }
12 int main(){
13     cin>>n;
14     for(i=1;i<=n;i++)
15         for(j=1;j<=i;j++) cin>>a[i][j];
16     cout<<solve(1,1)<<endl;
17     return 0;
18 }
```

i代表二维数组行号

j代表二维数组列号

n是输入的总行数

a是输入的二维数组

u是x+1行y列solve值

v是x+1行y+1列solve值

关键语句

阅读程序

```
1  #include<iostream>
2  using namespace std;
3  int n,i,j,a[100][100];
4  int solve(int x,int y){
5      int u,v;
6      if(x==n) return a[x][y];
7      u=solve(x+1,y);
8      v=solve(x+1,y+1);
9      if(u>v) return a[x][y]+u;
10     else return a[x][y]+v;
11 }
12 int main(){
13     cin>>n;
14     for(i=1;i<=n;i++)
15         for(j=1;j<=i;j++) cin>>a[i][j];
16     cout<<solve(1,1)<<endl;
17     return 0;
18 }
```

注意内层for循环
列数j <= 行数i

关键语句

阅读程序

```
1  #include<iostream>
2  using namespace std;
3  int n,i,j,a[100][100];
4  int solve(int x,int y){
5      int u,v;
6      if(x==n) return a[x][y];
7      u=solve(x+1,y);
8      v=solve(x+1,y+1);
9      if(u>v) return a[x][y]+u;
10     else return a[x][y]+v;
11 }
12 int main(){
13     cin>>n;
14     for(i=1;i<=n;i++)
15         for(j=1;j<=i;j++) cin>>a[i][j];
16     cout<<solve(1,1)<<endl;
17     return 0;
18 }
```

solve(x,y)是递归函数

当行数=n 返回a[x][y]

否则递归调用
solve(x+1,y)
solve(x+1,y+1)

并将两次递归返回值中的较大者与a[x][y]之和返回

关键语句

阅读程序

```
1  #include<iostream>
2  using namespace std;
3  int n,i,j,a[100][100];
4  int solve(int x,int y){
5      int u,v;
6      if(x==n) return a[x][y];
7      u=solve(x+1,y);
8      v=solve(x+1,y+1);
9      if(u>v) return a[x][y]+u;
10     else return a[x][y]+v;
11 }
12 int main(){
13     cin>>n;
14     for(i=1;i<=n;i++)
15         for(j=1;j<=i;j++) cin>>a[i][j];
16     cout<<solve(1,1)<<endl;
17     return 0;
18 }
```

找到入口

solve(1,1)依赖哪几个函数

solve(2,1) solve(2,2)

solve(2,1)依赖哪几个函数

solve(3,1) solve(3,2)

solve(3,1)依赖哪几个函数

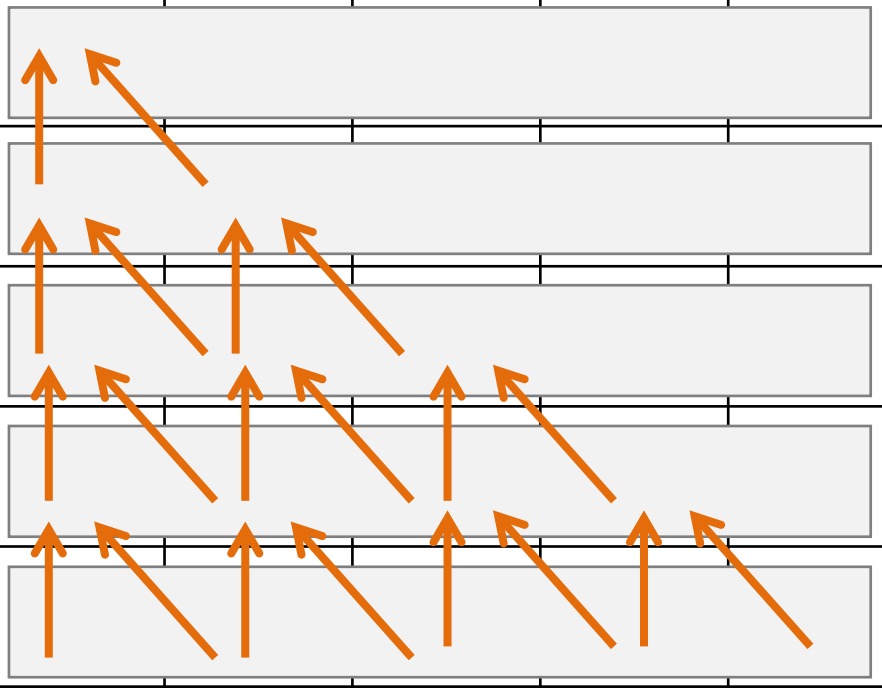
solve(4,1) solve(4,2)

关键语句

二维递推

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

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



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)

代码段作用

阅读程序

```
1  #include<iostream>
2  using namespace std;
3  int n,i,j,a[100][100];
4  int solve(int x,int y){
5      int u,v;
6      if(x==n) return a[x][y];
7      u=solve(x+1,y);
8      v=solve(x+1,y+1);
9      if(u>v) return a[x][y]+u;
10     else return a[x][y]+v;
11 }
12 int main(){
13     cin>>n;
14     for(i=1;i<=n;i++)
15         for(j=1;j<=i;j++) cin>>a[i][j];
16     cout<<solve(1,1)<<endl;
17     return 0;
18 }
```

递归函数solve
① 在各层 solve(x,y)
是solve(x+1,y)和solve(x+1,y+1)
中较大者与a[x][y]的和
② 当x==n 返回a[x][y]本身

读入三角形数字阵列

从solve(1,1)开始递归调用

阅读程序

```
1  #include<iostream>
2  using namespace std;
3  int n,i,j,a[100][100];
4  int solve(int x,int y){
5      int u,v;
6      if(x==n) return a[x][y];
7      u=solve(x+1,y);
8      v=solve(x+1,y+1);
9      if(u>v) return a[x][y]+u;
10     else return a[x][y]+v;
11 }
12 int main(){
13     cin>>n;
14     for(i=1;i<=n;i++)
15         for(j=1;j<=i;j++) cin>>a[i][j];
16     cout<<solve(1,1)<<endl;
17     return 0;
18 }
```

判断	二维数组a通过cin输入了n^2次（　） 。

阅读程序

```
#include<iostream>
using namespace std;
int n,i,j,a[100][100];
int solve(int x,int y){
    int u,v;
    if(x==n) return a[x][y];
    u=solve(x+1,y);
    v=solve(x+1,y+1);
    if(u>v) return a[x][y]+u;
    else return a[x][y]+v;
}
int main(){
    cin>>n;
    for(i=1;i<=n;i++)
        for(j=1;j<=i;j++) cin>>a[i][j];
    cout<<solve(1,1)<<endl;
    return 0;
}
```

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（　）。

阅读程序

```
1  #include<iostream>
2  using namespace std;
3  int n,i,j,a[100][100];
4  int solve(int x,int y){
5      int u,v;
6      if(x==n) return a[x][y];
7      u=solve(x+1,y);
8      v=solve(x+1,y+1);
9      if(u>v) return a[x][y]+u;
10     else return a[x][y]+v;
11 }
12 int main(){
13     cin>>n;
14     for(i=1;i<=n;i++)
15         for(j=1;j<=i;j++) cin>>a[i][j];
16     cout<<solve(1,1)<<endl;
17     return 0;
18 }
```

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（　）。

题目

阅读程序

```
#include<iostream>
using namespace std;
int n,i,j,a[100][100];
int solve(int x,int y){
    int u,v;
    if(x==n) return a[x][y];
    u=solve(x+1,y);
    v=solve(x+1,y+1);
    if(u>v) return a[x][y]+u;
    else return a[x][y]+v;
}
int main(){
    cin>>n;
    for(i=1;i<=n;i++)
        for(j=1;j<=i;j++) cin>>a[i][j];
    cout<<solve(1,1)<<endl;
    return 0;
}
```

选择	n可以输入的最大值（　）。
	A. 100　B. 99　C. 10000　D. 9801

题目

阅读程序

```
#include<iostream>
using namespace std;
int n,i,j,a[100][100];
int solve(int x,int y){
    int u,v;
    if(x==n) return a[x][y];
    u=solve(x+1,y);
    v=solve(x+1,y+1);
    if(u>v) return a[x][y]+u;
    else return a[x][y]+v;
}
int main(){
    cin>>n;
    for(i=1;i<=n;i++)
        for(j=1;j<=i;j++) cin>>a[i][j];
    cout<<solve(1,1)<<endl;
    return 0;
}
```

输入：

2

5

1 3

输出结果是（　）。

选择

A. 6　B. 5

C. 8　D. 4

题目

阅读程序

```
#include<iostream>
using namespace std;
int n,i,j,a[100][100];
int solve(int x,int y){
    int u,v;
    if(x==n) return a[x][y];
    u=solve(x+1,y);
    v=solve(x+1,y+1);
    if(u>v) return a[x][y]+u;
    else return a[x][y]+v;
}
int main(){
    cin>>n;
    for(i=1;i<=n;i++)
        for(j=1;j<=i;j++) cin>>a[i][j];
    cout<<solve(1,1)<<endl;
    return 0;
}
```

输入：

```
5
2
-1 4
2 -1 -2
-1 6 4 0
3 2 -1 5 8
```

输出结果是（　）。

选择

A. 14　B. 2
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

5分钟

完善程序

```cpp
#include <iostream>
using namespace std;
const int N = 110000, P = 10007;
int n;
int a[N], len;
int ans;
void getDivisor() {
    len = 0;
    for (int i = 1; _____(1) _____ <= n; ++i)
        if (n % i == 0) {
            a[++len] = i;
            if (_____(2) _____ != i)
                a[++len] = n / i;
        }
}
int gcd(int a, int b) {
    if (b == 0) {
        _____(3)_____;
    }
    return gcd(b, _____(4) _____);
}
int main() {
    cin >> n;
    getDivisor();
    ans = 0;
    for (int i = 1; i <= len; ++i) {
        for (int j = i + 1; j <= len; ++j) {
            ans = (_____(5) _____) % P;
        }
    }
    cout << ans << endl;
    return 0;
}
```

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	

完善程序

```
1   #include <iostream>
2   using namespace std;
3   const int N = 110000, P = 10007;
4   int n;
5   int a[N], len;
6   int ans;
7   void getDivisor() {
8       len = 0;
9       for (int i = 1; _____(1) _____ <= n; ++i)
10          if (n % i == 0) {
11              a[++len] = i;
12              if (_____(2) _____ != i)
13                  a[++len] = n / i;
14          }
15  }
16  int gcd(int a, int b) {
17      if (b == 0) {
18          _____(3)_____;
19      }
20      return gcd(b, _____(4) _____);
21  }
22  int main() {
23      cin >> n;
24      getDivisor();
25      ans = 0;
26      for (int i = 1; i <= len; ++i) {
27          for (int j = i + 1; j <= len; ++j) {
28              ans = (_____(5) _____) % P;
29          }
30      }
31      cout << ans << endl;
32      return 0;
33  }
```

解释变量的作用

n	输入
a[]	储存n的因子
len	n的因子数量
ans	两两因子的最大公约数之和
getDivisor()	求n的全部因子
gcd()	求a,b的最大公约数

完善程序

关键语句

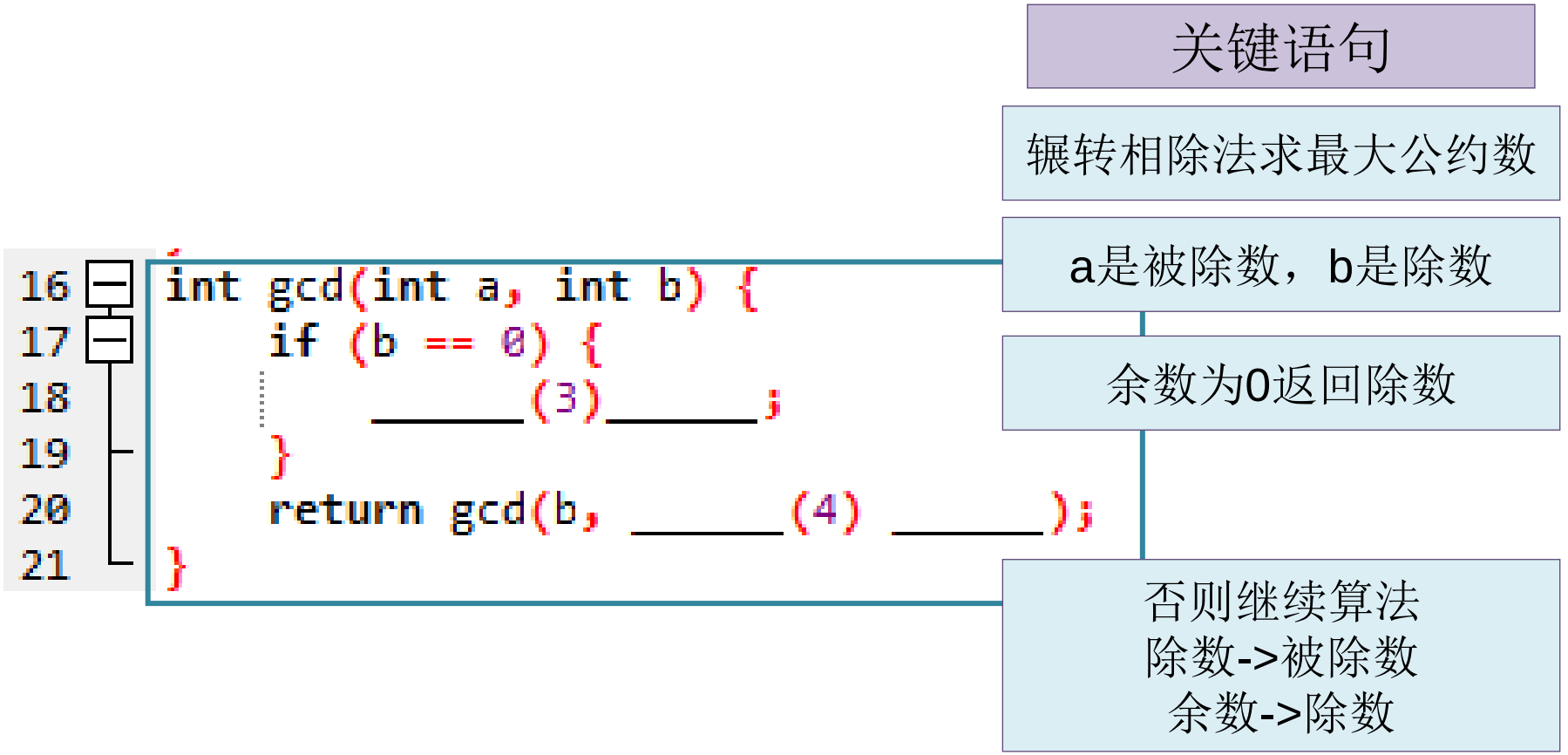
$P(\sqrt{n})$求n的因子

i范围从1到$\sqrt{n}$

```
7  void getDivisor() {
8      len = 0;
9      for (int i = 1; _____(1)_____ <= n; ++i)
10         if (n % i == 0) {
11             a[++len] = i;
12             if (_____(2)_____ != i)
13                 a[++len] = n / i;
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

完善程序


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

完善程序

关键语句

```
22  int main() {
23      cin >> n;
24      getDivisor();
25      ans = 0;
26      for (int i = 1; i <= len; ++i) {
27          for (int j = i + 1; j <= len; ++j) {
28              ans = (_____(5) _____) % P;
29          }
30      }
31      cout << ans << endl;
32      return 0;
33  }
```

双重循环枚举数组任意两个不同的元素

累加任意两因子的最大公约数

完善程序

（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

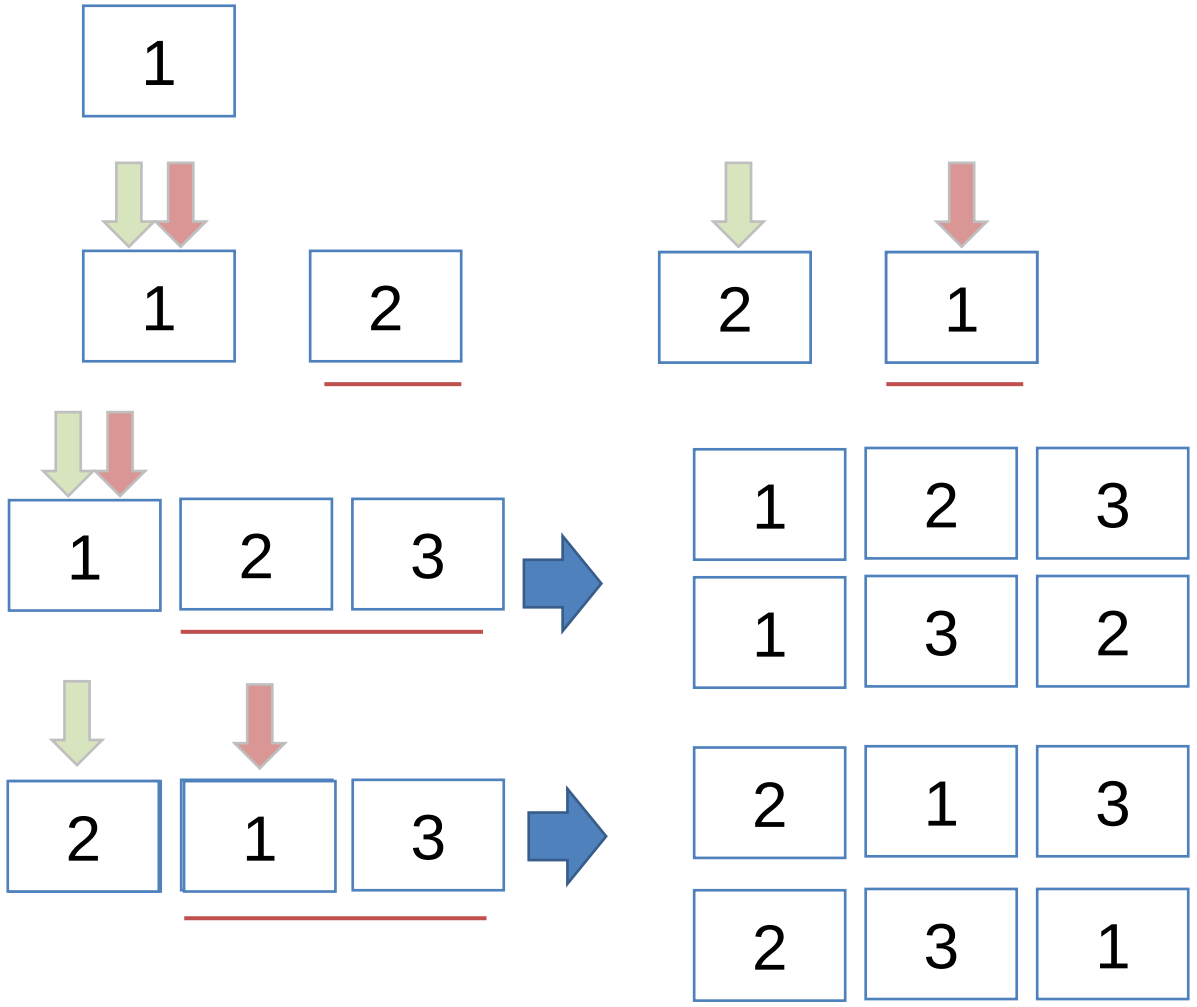
注意结果跟字典序不同

递归生成全排列

1个数字
全排列

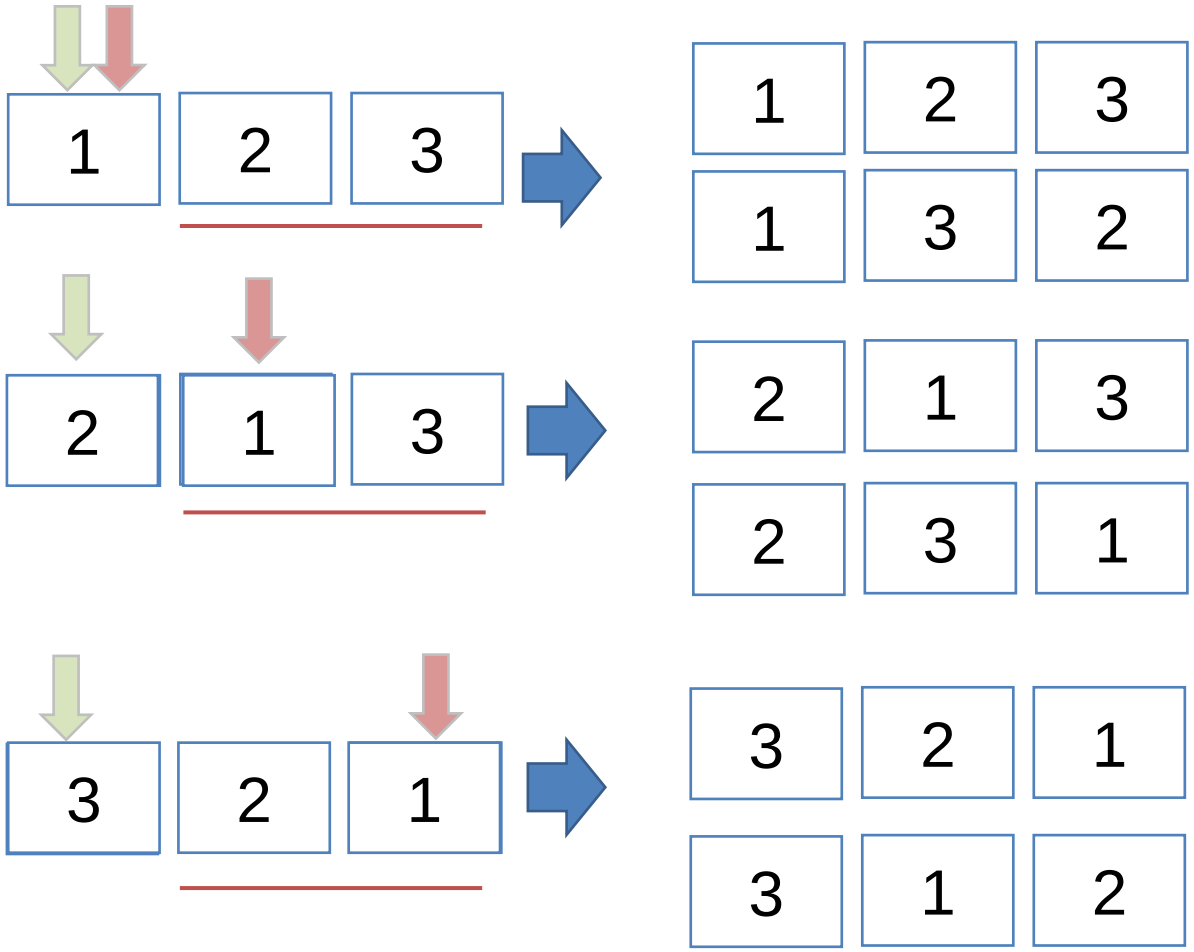
2个数字
全排列

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
```

5分钟

完善程序

```
#include <iostream>
#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)
{
    int j,p,t;
    if(____1____)
    {
        count++;
        for(p=1;p<=n;p++)
            cout <<setw(1)<<a[p];
        cout <<"  ";
        if(____2____)  cout <<endl;
        return;
    }
    for(j=k;j<=n;j++)
    {
        t=a[k];a[k]=a[j];a[j]=t;
        ____3____;
        t=a[k]; ____4____;
    }
}
int main()
{
    int i;
    cin >>n;
    for(i=1;i<=n;i++)  a[i]=i;
    ____5____;
    return 0;
}
```

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	

完善程序

```
#include <iostream>
#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)
{
    int j,p,t;
    if(____1____)
    {
        count++;
        for(p=1;p<=n;p++)
            cout <<setw(1)<<a[p];
        cout <<"  ";
        if(____2____)  cout <<endl;
        return;
    }
    for(j=k;j<=n;j++)
    {
        t=a[k];a[k]=a[j];a[j]=t;
        ____3____;
        t=a[k]; ____4____;
    }
}
int main()
{
    int i;
    cin >>n;
    for(i=1;i<=n;i++)  a[i]=i;
    ____5____;
    return 0;
}
```

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
p	循环变量
t	用于交换两个变量时暂存数据

完善程序

```
#include <iostream>
#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)
{
    int j,p,t;
    if(____1____)
    {
        count++;
        for(p=1;p<=n;p++)
            cout <<setw(1)<<a[p];
        cout <<"  ";
        if(____2____)  cout <<endl;
        return;
    }
    for(j=k;j<=n;j++)
    {
        t=a[k];a[k]=a[j];a[j]=t;
        ____3____;
        t=a[k]; ____4____;
    }
}
int main()
{
    int i;
    cin >>n;
    for(i=1;i<=n;i++)  a[i]=i;
    ____5____;
    return 0;
}
```

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

完善程序

```
#include <iostream>
#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)
{
    int j,p,t;
    if(____1____)
    {
        count++;
        for(p=1;p<=n;p++)
            cout <<setw(1)<<a[p];
        cout <<"  ";
        if(____2____)  cout <<endl;
        return;
    }
    for(j=k;j<=n;j++)
    {
        t=a[k];a[k]=a[j];a[j]=t;
        ____3____;
        t=a[k]; ____4____;
    }
}
int main()
{
    int i;
    cin >>n;
    for(i=1;i<=n;i++)  a[i]=i;
    ____5____;
    return 0;
}
```

关键语句

递归生成全排列算法
将当前位k与(k到n)位交换，递归生成k+1到n的全排列

交换a[k]和a[j]

递归生成k+1到n的全排列

回溯，换回a[k]和a[j]

调用递归函数

作业：完善程序

1999 最大公约数之和

1993 全排列