

选择题

选 1 如果128种颜色用二进制编码来表示, 择 至少需要()位。

A. 5 B. 6 C. 7 D. 8

答案: C

解析:每一位可以表示0或1,从0000000到

1111111,7位一共可以表示2⁷=128种。

2 若12*25=311成立,则用的是()进制。

A. 11 B. 8 C. 7 D.9

答案: D

解析: $(1*n+2)*(2*n+5) = 3n^2+1*n+1$ 。 $n^2-8n-9=0$,解

得n = 9或n = -1。

3一个袋子里装了100个苹果,100个香蕉,100个橘子,100个梨子。从袋子中取出一个水果需要1分钟,那么需要()分钟就能肯定至少已经拿出1打(12个)相同种类的水果。

A. 12 B. 13 C. 45 D. 101

答案: C

解析: 根据鸽巢原理, 4*11+1=45。

4下列程序段的时间复杂度是()。

```
int cnt=0;
for(int i=1; i<n; i*=2)
for(int j=1; j<n; j++)
cnt++;
```

A. $O(log_2(n))$ B. O(n) C. $O(n*log_2(n))$ D. $O(n^2)$

答案: C

解析:时间复杂度计算。对外层的for循环,设其执行次数为t,则 2^{t} <n,即t<log₂(n)。内层for循环需要执行n次,于是整段程序的时间复杂度是 $O(n*log_2(n))$ 。

5以下是32位机器与64位机器区别的是()。

- A. 硬盘大小不同
- B. 显示器分辨率不同
- C. 操作系统版本号不同
- D. 寻址空间不同

答案: D

解析: 32位机器与64位机器的区别是寻址空间不同。

6 如果一棵二叉树的先序遍历是ACDBEFG,中序遍历是DCAEBFG,那么它的后序遍历是 ()。

- A. DCEGFBA
- B. DCAEFGB
- C. DCEFGBA
- D. DCAEGFB

答案: A

解析: 先序: 根-左-右。中序: 左-根-右。先序中首个结点即为根(在本题中为A),然后通过中序里根A的所在位置,划分左右子树(根分左右);再根据左右子树的结点个数,去划分得到先序的左右子树。递归进行上述步骤。可画出树的形态,最后再求后序遍历。

7如果开始时计算机处于小写输入状态,现在有一只小狗反复按照 CapsLock、字母键 A、字母键 B、字母键 C、字母键 D的顺序来回按键,即 CapsLock、A、B、C、D、CapsLock、A、B、C、D、CapsLock、A、B、C、D、.....,屏幕上输出的第 2020 个字符是字母()。

A. 大写A B. 大写B C. 大写C D. 大写D

答案: D

解析: CapsLock是大小写切换。所以,每组一共八个字母,即ABCDabcd。2020%8=4,第2020个输出的是大写D

8 如果进栈序列e1,e2,e3,e4,则不可能的出栈序列是: ()。

A. e2, e4, e3, e1

B. e4, e3, e2, e1

C. e1, e2, e3, e4

D. e3, e1, e4, e2

答案: D

解析: e3 先出栈,则e1 不可能在e2 之前出栈,D 错误。

```
9有如下程序:
#include <iostream>
using namespace std;
int a[3][3];
int main(){
   for(int i=0; i<3; i++)
        for(int j=0; j<3; j++) a[i][j]=i*3+j+1;
   for(int i=1; i<3; i++)
        for(int j=0; j<2; j++) cout<< a[j][i];
   return 0;
运行后的输出结果是()。
```

A. 2536 B. 2356 C. 4758 D. 4578

答案: A

解析:注意输出的是a[j][i],而不是a[i][j]。所以依次输出的是a[0][1]、a[1][1]、a[0][2]、a[1][2]。

10 8名男生和4名女生围绕圆桌就坐,任意两个女生不相邻的坐法有()种。

A. 67737600

B. 8467200

C. 40320

D. 352800

答案: B

解析: 男生圆排列, 8!/8=7!, 固定一个男生, 把女生

插空到8名男生之间: A(8,4)。

方案数7! * A(8,4)=8467200。

11 关于下面说法,错误的一项是()。

- A. ASCII码是一种字符编码,常用7位码
- B. 程序设计的三种基本结构是顺序、选择、循环
- C. 刷新率、CPI、DPI均可以用于描述鼠标性能
- D. 目前世界上最大的计算机互联网络是IBM网

答案: D

解析: Internet是目前世界上覆盖范围最广、使用者最多、 最成功的计算机互联网络。

12 小红课桌上有5本书,每本书都放有一个写着书名的书签。调皮的小王把小红的书签全部打乱了。那么,每个书签都没有插到对应的书中的可能性总共有()种。

A. 38 B. 40 C. 44 D. 48

答案: C

解析: 经典的错排问题,假设n本书和书签全打乱的可能性一共是f(n)种,那么遵循如下递推公式 f(n) = (n-1)*(f(n-1)+f(n-2)),所以f(5)=44。

13设 G 是有 7 个结点的完全图,要得到一棵生成树,需要从 G 中删去()条边。

A. 10 B. 12 C. 15 D. 18

答案: C

解析:7个点的完全图有7*6/2=21条边,而7个点的树有7-1=6条边,因此需要删去21-6=15条边。

14 46! 的计算结果, 尾数总共有() 个零。

A. 9 B. 10 C. 11 D.12

答案: B

解析:求46!尾数总共有几个零,也就是求46!中2*5的因子的个数。由于因子2的数量远多于5的个数,所以就转换成求5的因子个数,1到46中有5、10、15、20、25(5*5)、30、35、40、45,总计10个。

- A. array是一个指针,它指向一个数组,数组的元素是双精度浮点型。
- B. array是一个数组,数组的每一个元素是指向双精度浮点型数据的指针
- C. C++语言中不允许这样的定义语句
- D. 以上都不对

答案:B

解析:根据数组定义,array是一个数组,包含了8个double型指针。

```
#include <iostream>
                                                          识别变量
   using namespace std;
3 □ int main() {
                                                      常见变量名
       int max, min, sum, cnt = 0;
                                                     翻译循环变量
       int temp;
       cin >> temp;
                                                 根据变量名的英文推断
       if (temp == 0) return 0;
       max = min = sum = temp;
       cnt++:
       while (temp != 0) {
100
                                                  2
                                                        找出关键语句
11
           cin >> temp;
           if (temp != 0) {
12 🖹
                                                    控制结构(for, if)
13
              sum += temp;
                                                  常见算法的基本操作
14
              cnt++;
                                                   函数参数、返回值
15
              if (temp > max) max = temp;
16
              if (temp < min) min = temp;</pre>
17
18
19
       cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
20
       return 0;
                                                  3
                                                       理解代码段作用
21
                                                   翻译解释代码段
```

```
#include <iostream>
                                                  解释变量的作用
   using namespace std:
3 □ int main() {
                                                             最大值
        int max, min, sum, cnt = 0;
                                              max
4
        int temp;
                                              min
                                                             最小值
6
        cin >> temp;
        if (temp == 0) return 0;
                                                             各数总和
                                              sum
8
        max = min = sum = temp;
9
        cnt++;
                                                             数字个数
                                              cnt
        while (temp != 0) {
100
                                                             临时变量
                                             temp
            cin >> temp;
11
            if (temp != 0) {
12
13
                sum += temp;
14
                cnt++;
15
                if (temp > max) max = temp;
16
                if (temp < min) min = temp;</pre>
17
18
        cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
19
        return 0;
20
21
```

关键语句

```
#include <iostream>
   using namespace std;
3 □ int main() {
                                                   定义变量
       int max, min, sum, cnt = 0;
       int temp;
                                              首个数字输入到temp
       cin >> temp;
       if (temp == 0) return 0;
                                              如果数字为0直接结束
       max = min = sum = temp;
       cnt++;
                                                  while循环
       while (temp != 0) {
10 =
                                              逐个读入数字到temp
          cin >> temp;
11
           if (temp != 0) {
12 □
                                             并更新最大值、最小值、
13
              sum += temp;
                                                数字个数、总和
14
              cnt++;
15
              if (temp > max) max = temp;
                                                 注意关键条件
16
              if (temp < min) min = temp;</pre>
17
                                                 遇到0结束while
18
19
       cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
20
       return 0;
                                             依次输出个数、最小值、
21
                                                最大值、平均值
```

```
阅读程序
   #include <iostream>
    using namespace std;
3 □ int main() {
        int max, min, sum, cnt = 0;
        int temp;
        cin >> temp:
        if (temp == 0) return 0;
        max = min = sum = temp;
        cnt++:
        while (temp != 0) {
100
11
            cin >> temp;
            if (temp != 0) {
12
13
                sum += temp;
14
                cnt++;
15
                if (temp > max) max = temp;
                if (temp < min) min = temp;</pre>
16
17
18
        cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
19
20
        return 0;
21
```

1. 本程序统计了输入数字个数、最大值、最小值、平均值,输入遇-1结束。()

```
阅读程序
   #include <iostream>
    using namespace std;
3 □ int main() {
        int max, min, sum, cnt = 0;
        int temp:
        cin >> temp:
        if (temp == 0) return 0;
        max = min = sum = temp;
        cnt++:
        while (temp != 0) {
10 =
            cin >> temp;
11
            if (temp != 0) {
12
13
                sum += temp;
14
                cnt++;
15
                if (temp > max) max = temp;
                if (temp < min) min = temp;</pre>
16
17
18
        cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
19
20
        return 0;
```

2. 对任意的整数值x,只需要将第7行中if条件判断修改为temp==x,就可以将本程序功能变成为输入遇值x结束。(

```
阅读程序
   #include <iostream>
    using namespace std;
3 □ int main() {
        int max, min, sum, cnt = 0;
        int temp:
        cin >> temp:
        if (temp == 0) return 0;
        max = min = sum = temp;
        cnt++:
        while (temp != 0) {
10 =
            cin >> temp;
11
            if (temp != 0) {
12
13
                sum += temp;
14
                cnt++;
15
                if (temp > max) max = temp;
                if (temp < min) min = temp;</pre>
16
17
18
        cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
19
20
        return 0;
```

3. 去掉第8行的连续赋值语句,或者忘记对第4行的整型变量cnt 初始化为0,这两种情况都可能影响程序最终输出结果。()

```
阅读程序
   #include <iostream>
    using namespace std;
3 □ int main() {
        int max, min, sum, cnt = 0;
        int temp;
        cin >> temp:
        if (temp == 0) return 0;
        max = min = sum = temp;
        cnt++:
        while (temp != 0) {
10 =
            cin >> temp;
11
            if (temp != 0) {
12
13
                sum += temp;
14
                cnt++;
15
                if (temp > max) max = temp;
                if (temp < min) min = temp;</pre>
16
17
18
        cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
19
20
        return 0;
```

4. 将15行if条件判断语句改成temp>=max,并且把16行if条件判断语句改成temp<=min,对程序结果没有任何影响。()

```
阅读程序
   #include <iostream>
    using namespace std:
3 □ int main() {
        int max, min, sum, cnt = 0;
        int temp;
6
        cin >> temp;
        if (temp == 0) return 0;
8
        max = min = sum = temp;
9
        cnt++:
        while (temp != 0) {
100
11
            cin >> temp;
12 🖨
            if (temp != 0) {
13
                sum += temp;
14
                cnt++;
15
                if (temp > max) max = temp;
                if (temp < min) min = temp;</pre>
16
17
18
        cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
19
20
        return 0;
21
```

选择 5. 若输入012345,程序运行到第()行结束。

A. 6 B. 7 C. 10 D. 20

```
阅读程序
   #include <iostream>
   using namespace std;
3 □ int main() {
       int max, min, sum, cnt = 0;
       int temp;
       cin >> temp;
       if (temp == 0) return 0;
8
       max = min = sum = temp;
9
       cnt++:
100
       while (temp != 0) {
           cin >> temp;
11
           if (temp != 0) {
12
13
               sum += temp;
14
               cnt++;
15
               if (temp > max) max = temp;
               if (temp < min) min = temp;</pre>
16
17
18
       cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
19
20
       return 0;
21
         6. 若输入12346089, 程序最终输出结果是( )。
   洗择
          A. 5 1 6 3.2 B. 8 1 9 4 C. 6 1 6 3 D. 5 1 6 3
```

```
#include <stdio.h>
                                 阅读程序
7 ☐ int gcd(int a, int b){
        if(b == 0) return a;
        return gcd(b,a%b);
11 □ void work(int a, int b) {
        int s[10009], t[10009], i = 0, d = 1, j;
12
13 🗏
        while(1) {
14
            if(a == 0) break;
            a *= 10; //除法借位
15
16
            t[i] = a;
17
            s[i] = a / b;
18
            a %= b;
19
            for(j = 0; j < i; ++j)
                if(t[j] == t[i]) { d = 0; break; }
20
21
            if(d == 0) break;
22
            printf("%d", s[i]); i++;
23
24
25 ☐ int main() {
        int a, b, g; scanf("%d%d", &a, &b);
26
27
        if(a > b) g = gcd(a, b);
28
        else g = gcd(b, a);
29
        a /= g; b /= g; printf("%d.", a/b);
        a %= b; work(a, b);
30
31
        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

```
阅读程序
    #include <stdio.h>
7 ☐ int gcd(int a, int b){
       if(b == 0) return a;
                                                 解释变量的作用
       return gcd(b,a%b);
10
                                                               分子
                                                a
11 □ void work(int a, int b) {
12
       int s[10009], t[10009], i = 0, d = 1, j;
                                                b
                                                               分母
13 🗏
       while(1) {
                                                             计数器变量
14
           if(a == 0) break;
           a *= 10; //除法借位
15
                                                t[i]
                                                          第i位上10倍a的值
16
           t[i] = a;
17
           s[i] = a / b;
                                                        求第i位上10*a除以b的值
                                                s[i]
18
           a %= b;
                                                        得到第i小数位上真实值
19
           for(j = 0; j < i; ++j)
              if(t[j] == t[i]) { d = 0; break; }
20
21
           if(d == 0) break;
                                                        标记是否已经到了第一个
22
           printf("%d", s[i]); i++;
                                                d
                                                           重复值所在位置
23
24
25 ☐ int main() {
       int a, b, g; scanf("%d%d", &a, &b);
26
27
       if(a > b) g = gcd(a, b);
28
       else g = gcd(b, a);
                                                    本程序输入一个分数的
       a /= g; b /= g; printf("%d.", a/b);
29
                                                  正整数分子a和正整数分母b
30
       a %= b; work(a, b);
                                                   将其转换为小数形式输出
```

22

return 0;

```
阅读程序
    #include <stdio.h>
    int gcd(int a, int b){
8
        if(b == 0) return a;
9
        return gcd(b,a%b);
10
11 🖃
    void work(int a, int b) {
12
        int s[10009], t[10009], i = 0, d = 1, j;
13 E
        while(1) {
14
            if(a == 0) break;
            a *= 10; //除法借位
15
16
            t[i] = a;
17
            s[i] = a / b;
18
            a %= b;
19
            for(j = 0; j < i; ++j)
20
                if(t[j] == t[i]) { d = 0; break; }
            if(d == 0) break;
21
            printf("%d", s[i]); i++;
22
23
24
25 □ int main() {
        int a, b, g; scanf("%d%d", &a, &b);
26
        if(a > b) g = gcd(a, b);
27
28
        else g = gcd(b, a);
        a /= g; b /= g; printf("%d.", a/b);
29
        a %= b; work(a, b);
30
31
        return 0;
```

关键语句

递归返回a和b的最大公约数

定义并输入分子a和分母b 调用gcd函数求a和b最大公约数 将a和b约分到互质

先输出a/b整数部分和小数点 真分数部分调用work求小数表示

```
阅读程序
    #include <stdio.h>
                                                          关键语句
7 □ int gcd(int a, int b){
        if(b == 0) return a;
        return gcd(b,a%b);
10
11 □ void work(int a, int b) {
        int s[10009], t[10009], i = 0, d = 1, j;
12
13 =
        while(1)
14
            if(a == 0) break;
            a *= 10; //除法借位
15
16
            t[i] = a;
17
            s[i] = a / b;
                                                     计算对应位小数值,存到s[i]中
18
            a %= b;
19
            for(j = 0; j < i; ++j)
20
                if(t[j] == t[i]) { d = 0; break;}
21
            if(d == 0) break;
22
            printf("%d", s[i]); i++;
23
25 □ int main() {
26
        int a, b, g; scanf("%d%d", &a, &b);
27
        if(a > b) g = gcd(a, b);
        else g = gcd(b, a);
28
29
        a /= g; b /= g; printf("%d.", a/b);
30
        a %= b; work(a, b);
31
        return 0;
```

递归返回a和b的最大公约数

定义数组s,t和变量i,d,j

while循环逐位计算小数表示

a==0表示已经除尽,结束循环

除法借位乘10,存到t[i]中

检查i位a值是否已在前序位出现 若出现则到**重复值**,标记并结束

否则继续输出第i位小数值s[i]

定义并输入分子a和分母b

调用gcd函数求a和b最大公约数

将a和b约分到互质 先输出a/b整数部分和小数点

真分数部分调用work求小数表示

```
#include <stdio.h>
7 ☐ int gcd(int a, int b){
                                     阅读程序
        if(b == 0) return a;
        return gcd(b,a%b);
10
11 □ void work(int a, int b) {
12
        int s[10009], t[10009], i = 0, d = 1, j;
13 🗏
        while(1) {
14
            if(a == 0) break;
            a *= 10; //除法借位
15
            t[i] = a;
16
17
            s[i] = a / b;
18
            a %= b;
19
            for(j = 0; j < i; ++j)
                if(t[j] == t[i]) { d = 0; break; }
20
            if(d == 0) break;
21
22
            printf("%d", s[i]); i++;
23
24
25 □ int main() {
26
        int a, b, g; scanf("%d%d", &a, &b);
        if(a > b) g = gcd(a, b);
27
28
        else g = gcd(b, a);
        a /= g; b /= g; printf("%d.", a/b);
29
30
        a %= b; work(a, b);
31
        return 0;
32
```

1. 函数gcd使用了递归的方法,返回参数a和b的最小公倍数。()

```
#include <stdio.h>
7 ☐ int gcd(int a, int b){
                                    阅读程序
        if(b == 0) return a;
        return gcd(b,a%b);
10 L
11 □ void work(int a, int b) {
12
        int s[10009], t[10009], i = 0, d = 1, j;
13 🗏
        while(1) {
14
            if(a == 0) break;
15
            a *= 10; //除法借位
            t[i] = a;
16
17
            s[i] = a / b;
18
            a %= b;
19
            for(j = 0; j < i; ++j)
                if(t[j] == t[i]) { d = 0; break; }
20
            if(d == 0) break;
21
22
            printf("%d", s[i]); i++;
23
24
25 □ int main() {
26
        int a, b, g; scanf("%d%d", &a, &b);
        if(a > b) g = gcd(a, b);
27
28
        else g = gcd(b, a);
29
        a /= g; b /= g; printf("%d.", a/b);
30
        a %= b; work(a, b);
31
        return 0;
32
```

2.30行作为work函数的参数传入a和b,一定满足a<b且a≠0。()

```
#include <stdio.h>
7 ☐ int gcd(int a, int b){
                                    阅读程序
        if(b == 0) return a;
        return gcd(b,a%b);
10
11 □ void work(int a, int b) {
12
        int s[10009], t[10009], i = 0, d = 1, j;
13 🗏
        while(1) {
            if(a == 0) break;
14
15
            a *= 10; //除法借位
            t[i] = a;
16
17
            s[i] = a / b;
18
            a %= b;
19
            for(j = 0; j < i; ++j)
                if(t[j] == t[i]) { d = 0; break; }
20
            if(d == 0) break;
21
22
            printf("%d", s[i]); i++;
23
24
25 □ int main() {
26
        int a, b, g; scanf("%d%d", &a, &b);
27
        if(a > b) g = gcd(a, b);
28
        else g = gcd(b, a);
29
        a /= g; b /= g; printf("%d.", a/b);
30
        a %= b; work(a, b);
31
        return 0;
32
```

3. 去掉19-21行的代码,将增加程序发生运行时错误的风险。()

```
#include <stdio.h>
7 ☐ int gcd(int a, int b){
                                    阅读程序
        if(b == 0) return a;
        return gcd(b,a%b);
10 L
11 □ void work(int a, int b) {
12
        int s[10009], t[10009], i = 0, d = 1, j;
13 🗏
        while(1) {
14
            if(a == 0) break;
15
            a *= 10; //除法借位
            t[i] = a;
16
17
            s[i] = a / b;
18
            a %= b;
19
            for(j = 0; j < i; ++j)
                if(t[j] == t[i]) { d = 0; break; }
20
            if(d == 0) break;
21
22
            printf("%d", s[i]); i++;
23
24
25 ☐ int main() {
26
        int a, b, g; scanf("%d%d", &a, &b);
27
        if(a > b) g = gcd(a, b);
28
        else g = gcd(b, a);
        a /= g; b /= g; printf("%d.", a/b);
29
30
        a %= b; work(a, b);
31
        return 0;
32
```

4. 去掉27行、28行以及29行前两个语句,程序输出结果一定发生 改变。()

```
#include <stdio.h>
7 ☐ int gcd(int a, int b){
                                   阅读程序
        if(b == 0) return a;
8
        return gcd(b,a%b);
10 L }
11 □ void work(int a, int b) {
12
        int s[10009], t[10009], i = 0, d = 1, j;
13 🗏
        while(1) {
14
            if(a == 0) break;
15
            a *= 10; //除法借位
16
            t[i] = a;
17
            s[i] = a / b;
18
            a %= b;
19
            for(j = 0; j < i; ++j)
20
               if(t[j] == t[i]) { d = 0; break; }
            if(d == 0) break;
21
22
            printf("%d", s[i]); i++;
23
24
25 □ int main() {
26
        int a, b, g; scanf("%d%d", &a, &b);
27
        if(a > b) g = gcd(a, b);
28
        else g = gcd(b, a);
        a /= g; b /= g; printf("%d.", a/b);
29
30
        a %= b; work(a, b);
31
        return 0;
32
    选择
            5.若输入91 13,输出结果是(
```

A. 7 B. 7. C. 7.0 D. 7.00

```
#include <stdio.h>
7 ☐ int gcd(int a, int b){
                                   阅读程序
        if(b == 0) return a;
        return gcd(b,a%b);
10 L }
11 □ void work(int a, int b) {
12
        int s[10009], t[10009], i = 0, d = 1, j;
13 🗏
        while(1) {
14
            if(a == 0) break;
15
           a *= 10; //除法借位
           t[i] = a;
16
17
            s[i] = a / b;
18
            a %= b;
19
            for(j = 0; j < i; ++j)
20
               if(t[j] == t[i]) \{ d = 0; break; \}
            if(d == 0) break;
21
22
            printf("%d", s[i]); i++;
23
24
25 □ int main() {
26
        int a, b, g; scanf("%d%d", &a, &b);
27
        if(a > b) g = gcd(a, b);
28
        else g = gcd(b, a);
        a /= g; b /= g; printf("%d.", a/b);
29
30
        a %= b; work(a, b);
31
        return 0;
32
            6. 以下哪个结果是程序可能输出的()。
    选择
```

A. 10.0 B. 1.571428571428 C. 1.5714285 D. 1.571428

```
#include <iostream>
    using namespace std;
 3 p int solve(int n, int m){
        int i, sum;
 4
 5
        if (m == 1) return 1;
 6
        sum = 0;
        for (i = 1; i < n; i++)
 8
            sum += solve(i, m - 1);
 9
        return sum;
10
11 pint main(){
12
        int n, m;
13
        cin>>n>>m;
14
        cout<<solve(n, m)<<endl;</pre>
15
        return 0;
16
```

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

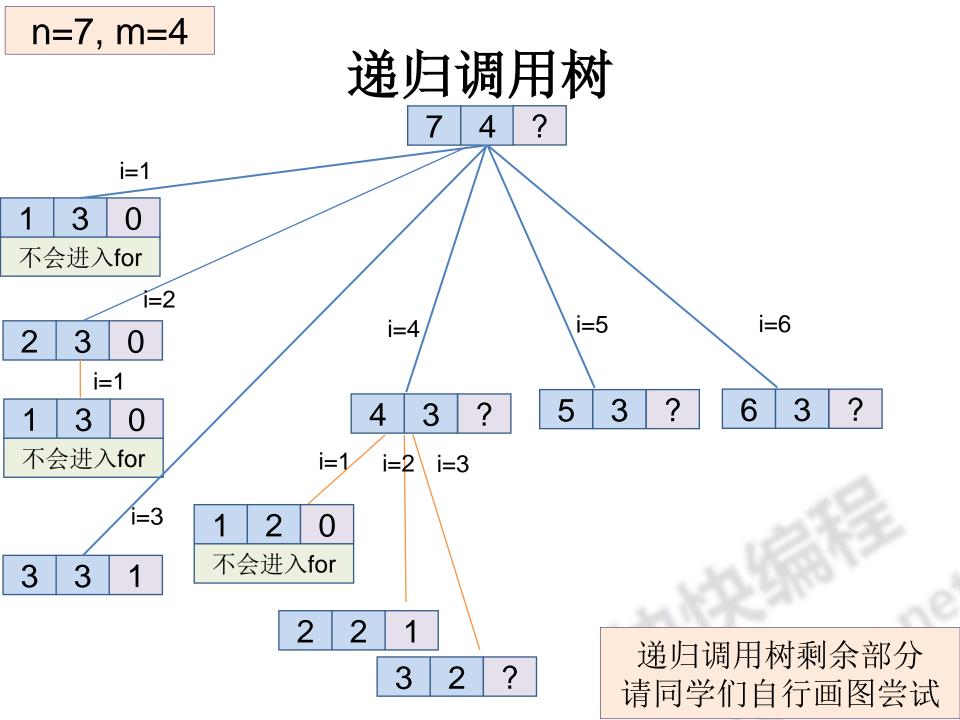
翻译解释代码段

```
#include <iostream>
                                             带入小数据
   using namespace std;
                                          输入
                                                    1 1
 3 p int solve(int n, int m){
        int i, sum;
 4
                                          输出
 5
        if (m == 1) return 1;
 6
        sum = 0;
        for (i = 1; i < n; i++)
                                           输入
                                                    12
 8
            sum += solve(i, m - 1);
 9
        return sum;
                                          输出
                                                    0
11 pint main(){
12
        int n, m;
                                          输入
                                                    22
13
        cin>>n>>m;
                                          输出
14
        cout<<solve(n, m)<<endl;</pre>
15
        return 0;
16
```

```
n=3, m=3
```

递归调用树

```
3
                           3
                                     i=2
          i=1
                                       2
      不会进入for
                                        i=1
                                       m==1返回1
 3 p int solve(int n, int m){
        int i, sum;
 4
 5
        if (m == 1) return 1;
                                              return
 6
                                          m
        sum = 0;
        for (i = 1; i < n; i++)
 8
             sum += solve(i, m - 1);
 9
        return sum;
10
```



简化: 使用表格递推

n\m 3 0 0 0 2 0 3 0 4 5 4 6 6 sum = 0;8

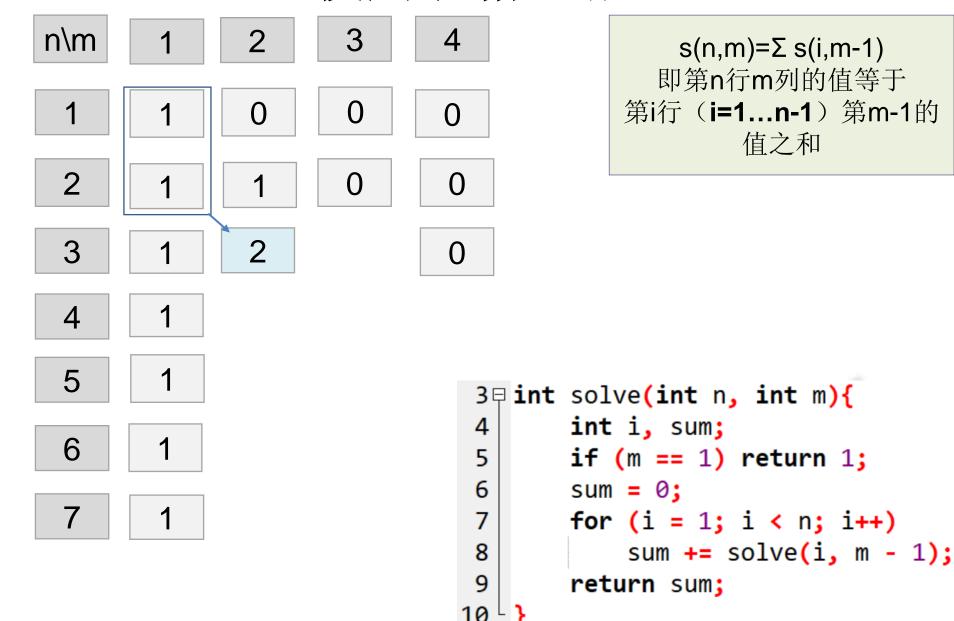
递归调用solve(n,m) 每深入一层m减1 n至少减1(for从1到n-1) 最终n减少到1无法进入for时 m必然还大于1,没有返回1 因此当n<m必然返回sum=0

```
int solve(int n, int m){
    int i, sum;
    if (m == 1) return 1;
    sum = 0;
    for (i = 1; i < n; i++)
        sum += solve(i, m - 1);
    return sum;
}</pre>
```

10

```
n\m
                      3
                                               当m==1时返回1
                      0
               0
                             0
 2
                              0
 3
                              0
 5
                                3 p int solve(int n, int m){
                                      int i, sum;
                               4
 6
                                      if (m == 1) return 1;
                               6
                                      sum = 0;
                                      for (i = 1; i < n; i++)
                               8
                                           sum += solve(i, m - 1);
                               9
                                      return sum;
```

```
n\m
                      3
                                              s(n,m)=\Sigma s(i,m-1)
                                             即第n行m列的值等于
                                          第i行(i=1...n-1)第m-1的
                      0
               0
                             0
                                                   值之和
                             0
 3
                             0
 5
                               3 p int solve(int n, int m){
                                      int i, sum;
                               4
 6
                                      if (m == 1) return 1;
                               6
                                      sum = 0;
                                      for (i = 1; i < n; i++)
                                          sum += solve(i, m - 1);
                               8
                               9
                                      return sum;
                              10
```



```
n\m
                      3
                                              s(n,m)=\Sigma s(i,m-1)
                                             即第n行m列的值等于
                                          第i行(i=1...n-1)第m-1的
                      0
               0
                             0
                                                   值之和
 2
                             0
               2
 3
                             0
               3
 5
                               3 p int solve(int n, int m){
                                      int i, sum;
                               4
 6
                                      if (m == 1) return 1;
                               6
                                      sum = 0;
                                      for (i = 1; i < n; i++)
                                          sum += solve(i, m - 1);
                               8
                               9
                                      return sum;
                              10
```

```
n\m
                                                s(n,m)=\Sigma s(i,m-1)
                                               即第n行m列的值等于
                                            第i行(i=1...n-1)第m-1的
                              0
                                                     值之和
                               0
 3
                               0
                       3
               3
 5
                       6
                               4
                                     3 pint solve(int n, int m){
                                           int i, sum;
 6
               5
                      10
                              10
                                           if (m == 1) return 1;
                                           sum = 0;
               6
                      15
                              20
                                           for (i = 1; i < n; i++)
                                               sum += solve(i, m - 1);
```

return sum;

n\m	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	2	1	0
4	1	3	3	1
5	1	4	6	4
6	1	5	10	10
7	1	6	15	20

总结递推规律!

当n>=2&&m>=2时 s(n,m)=s(n-1,m-1)+s(n-1,m)

```
#include <iostream>
                                              判断
   using namespace std;
                                1. 程序第三行改写成solve(int m, int n),
 3 p int solve(int n, int m){
                                程序运行结果不变。( )
 4
        int i, sum;
 5
        if (m == 1) return 1;
 6
        sum = 0;
        for (i = 1; i < n; i++)
 8
            sum += solve(i, m - 1);
 9
        return sum;
10
11 pint main(){
12
        int n, m;
13
        cin>>n>>m;
14
        cout<<solve(n, m)<<endl;</pre>
15
        return 0;
16
```

```
#include <iostream>
                                             判断
   using namespace std;
                               2. 去掉程序第6行后,程序运行结果不
 3 p int solve(int n, int m){
                               变。( )
        int i, sum;
 4
 5
        if (m == 1) return 1;
 6
        sum = 0;
        for (i = 1; i < n; i++)
 8
            sum += solve(i, m - 1);
 9
        return sum;
10
11 pint main(){
12
        int n, m;
13
        cin>>n>>m;
14
        cout<<solve(n, m)<<endl;</pre>
15
        return 0;
16
```

```
#include <iostream>
                                              选择
   using namespace std;
 3 p int solve(int n, int m){
                                3.若输入的n<m,则输出结果为(
 4
        int i, sum;
 5
        if (m == 1) return 1;
                                A. 0
                                           B. n
 6
                                           D. m+(m+1)+...+(n-1)
        sum = 0;
                                C. m
        for (i = 1; i < n; i++)
 8
            sum += solve(i, m - 1);
 9
        return sum;
10
11 pint main(){
12
        int n, m;
13
        cin>>n>>m;
14
        cout<<solve(n, m)<<endl;</pre>
15
        return 0;
16
```

```
#include <iostream>
                                             选择
   using namespace std;
 3 p int solve(int n, int m){
                                4. 输入43,程序第8行共循环()次。
        int i, sum;
 4
 5
        if (m == 1) return 1;
                                A. 3
                                      B. 6
                                              C. 9
                                                       D. 12
 6
        sum = 0;
        for (i = 1; i < n; i++)
 8
            sum += solve(i, m - 1);
 9
        return sum;
10
11 pint main(){
12
        int n, m;
13
        cin>>n>>m;
14
        cout<<solve(n, m)<<endl;</pre>
15
        return 0;
16
```

```
#include <iostream>
                                              选择
   using namespace std;
 3 p int solve(int n, int m){
                                5. 输入74, 输出是()
 4
        int i, sum;
 5
        if (m == 1) return 1;
                                        B. 10
                                A. 35
                                               C. 20
                                                        D. 15
 6
        sum = 0;
        for (i = 1; i < n; i++)
 8
            sum += solve(i, m - 1);
 9
        return sum;
10
11 pint main(){
12
        int n, m;
13
        cin>>n>>m;
14
        cout<<solve(n, m)<<endl;</pre>
15
        return 0;
16
```

```
#include <iostream>
                                           选择
   using namespace std;
                              6. (4分)以下四组输出数据中,输出
 3 pint solve(int n, int m){
                              最大的是()。
       int i, sum;
 4
 5
       if (m == 1) return 1;
                                     B. 53 C. 75
                              A. 63
                                                    D. 87
 6
       sum = 0;
       for (i = 1; i < n; i++)
 8
           sum += solve(i, m - 1);
 9
       return sum;
10
11 pint main(){
12
       int n, m;
13
       cin>>n>>m;
14
       cout<<solve(n, m)<<endl;</pre>
15
       return 0;
16
```

完善程序

完善程序

(活动选择)学校在最近几天有n个活动,这些活动都需要使用学校的大礼堂,在同一时间,礼堂只能被一个活动使用。由于有些活动时间上有冲突,学校办公人员只好让一些活动放弃使用礼堂。现给出n个活动使用礼堂的起始时间begini和结束时间endi(begini<endi)。

程序在第一行输入一个整数n(n<=1000),接下来n行,每行两个整数,第1个是begini,第2个是endi

(begini<endi<=32767),程序输出最多能安排的活动个数。

手算样例

输入样例:

3

1 4

3 5

4 5

输出多少?

输出样例:

2

手算样例

输入样例:

6

1 4

3 5

4 6

0 8

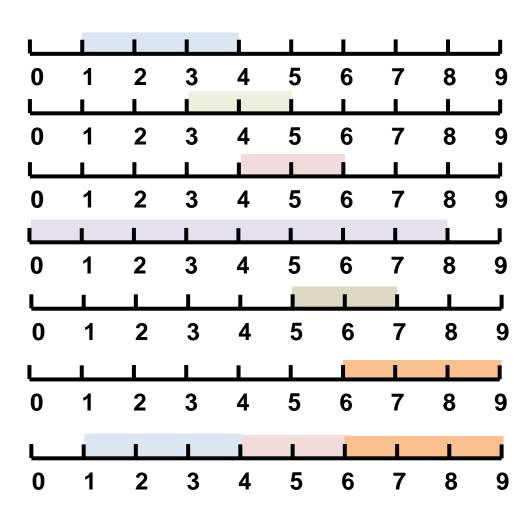
5 7

6 9

输出多少?

输出样例:

3

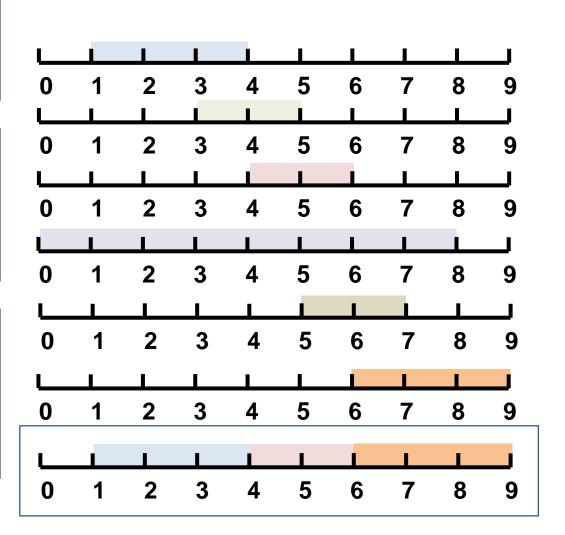


完善程序

给出n个开区间(begini, endi) 需选择**尽量多**的区间 保证两两不重叠

将活动按照endi的顺序排序, 依次考虑各活动,若未与已选 活动冲突就选,否则不选

反证贪心:假设首选最优选项不是end1而是endi。若两者时段不交叉,则可多选一个活动。若交叉了则将endi换成end1,不会使后续选择结果更差



```
#include<iostream>
     using namespace std;
     int n, begin[1009], end[1009]; 完善程序
void init() {
         cin >> n;
         for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
     void qsort(int x, int y) {
         int i, j, mid, t;
         i = ___(1)___; j = y;
10
11
         mid = (2);
         while(i <= j) {
12 -
             while(end[i] < mid) i++;
13
14
             while(___(3)___) j--;
15 -
             if(i <= j) {
16
                 t = end[j]; end[j] = end[i]; end[i] = t;
                 t = begin[j]; ____(4)___; begin[i] = t;
17
18
                 i++; j--;
19
20
21
         if(x < j) qsort(x, j);
22
         if(i < y) qsort(i, y);
23
     void solve() {
25
         int ans = 0;
26
         for(int i = 1, t = -1; i <= n; i++)
             if(begin[i]>=___(5)___) {ans++; t=end[i];}
27
28
         cout << ans << endl;
29
30 ☐ int main() {
31
         init(); qsort(1, n); solve();
32
         return 0;
33
```

```
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

理解代码段作用 翻译解释代码段

```
using namespace std;
     int n, begin[1009], end[1009]; 完善程序
void init() {
                                                              解释变量的作用
     void init() {
        cin >> n;
5
        for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
                                                                    活动总数输入
                                                         n
    void qsort(int x, int y) {
        int i, j, mid, t;
9
                                                      begin[i]
                                                                  第i项活动起始时间
10
        i = _{(1)}; j = y;
        mid = ___(2)___;
11
        while(i <= j) {
                                                       end[i]
                                                                 第i项活动的结束时间
12 -
            while(end[i] < mid) i++;
13
14
            while(___(3)___) j--;
                                                                本层快排的左右边界编号
                                                        X, y
15 -
            if(i <= j) {
16
               t = end[j]; end[j] = end[i]; end[i] = t;
                                                        i, j
                                                                  快速排序的双游标
                t = begin[j]; ____(4) ___; begin[i] = t;
17
18
                i++; j--;
                                                        mid
                                                                本层快排的基准值pivot
19
20
                                                                上一个被选活动结束时间
21
        if(x < j) qsort(x, j);
22
        if(i < y) qsort(i, y);
23
                                                                 可以安排的最多活动数
                                                        ans
24 🖳
     void solve() {
25
        int ans = 0;
26
        for(int i = 1, t = -1; i <= n; i++)
            if(begin[i]>=___(5)___) {ans++; t=end[i];}
27
28
        cout << ans << endl;
29
30 ☐ int main() {
31
        init(); qsort(1, n); solve();
32
        return 0;
33
```

#include<iostream>

```
代码段作用
     using namespace std;
    cin >> n;
                                                        1: 输入各活动起始结束时间
        for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
    void qsort(int x, int y) {
        int i, j, mid, t;
                                                       2: 按endi顺序对活动快速排序
        i = _{(1)}; j = y;
        mid = (2)___;
        while(i <= j) {
12
13
14
15
16
17
18
19
20
21
22
23
            while(end[i] < mid) i++;</pre>
            while(___(3)___) j--;
            if(i <= j) {
               t = end[j]; end[j] = end[i]; end[i] = t;
               t = begin[j]; ____(4) ___; begin[i] = t;
                i++; j--;
        if(x < j) qsort(x, j);
        if(i < y) qsort(i, y);
24
25
26
27
     void solve() {
                                                        3: 排好序后贪心地安排活动
        int ans = 0;
        for(int i = 1, t = -1; i <= n; i++)
            if(begin[i]>=___(5)___) {ans++; t=end[i];}
28
        cout << ans << endl;
   int main() {
31
        init(); qsort(1, n); solve();
                                                         1.初始化 2.排序 3.贪心安排
32
        return 0;
```

#include<iostream>

```
#include<iostream>
                                                           关键语句
    using namespace std;
                             完善程序
    int n, begin[1009], end[1009];
    void init() {
        cin >> n;
                                                  1: 输入各活动起始结束时间
        for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
    void qsort(int x, int y) {
       int i, j, mid, t;
                                                 2: 按endi顺序对活动快速排序
10
11
        mid =
                (2)
                                                  左右游标初始化为本层边界
       while(i <= j)
12
           while(end[i] < mid) i++;
13
                                                  选取一个endi设为本层基准
14
           while(__(3)___) j--;
15 -
           if(i <= j) {
                                                 游标定位到首对左右互换位
15
              t = end[j]; end[j] = end[i]; end[i] = t;
              t = begin[j]; ____(4)___; begin[i] = t;
17
                                                 左右坑位互换值并移动游标
18
              i++; j--;
19
23
                                                  按游标位置分别对左侧小区
        if(x < j) qsort(x, j);
21
                                                   和右侧大区递归进行分区
        if(i < y) qsort(i, y);
23
24 -
    void solve() {
                                                  3: 排好序后贪心地安排活动
25
        int ans = 0;
25
        for(int i = 1, t = -1; i \le n; i \leftrightarrow n)
                                                   排序后顺序遍历每个活动i,
27
           if(begin[i]>=___(5)___) {ans++; t=end[i];}
        cout << ans << endl;
28
                                                  只要起始时间>=上轮结束时间
29
                                                 则安排它,并更新最新结束时间
    int main() {
31
        init(); qsort(1, n); solve();
                                                   1.初始化 2.排序 3.贪心安排
32
        return 0;
```

```
2
     using namespace std;
     int n, begin[1009], end[1009];
     void init() {
 5
         cin >> n;
         for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
 6
     void qsort(int x, int y) {
         int i, j, mid, t;
9
10
         i = X ; j = y;
         mid = __(2)_{j} end[(x+y)/2]
11
         while(i <= j) {
12 -
13
             while(end[i] < mid) i++;</pre>
                                          end[i]>mid
14
             while( (3) ) j--;
             if(i <= j) {
15 -
16
                 t = end[j]; end[j] = end[i]; end[i] = t;
                 t = begin[j]; ___(4) __; begin[i] = t;
17
18
                 i++; j--;
                             begin[i]=begin[j]
19
20
21
         if(x < j) qsort(x, j);
22
         if(i < y) qsort(i, y);
23
24 \rightarrow void solve() {
25
         int ans = 0;
26
         for(int i = 1, t = -1; i \le n; i \leftrightarrow n
27
             if(begin[i]>= (5) ) {ans++; t=end[i];}
28
         cout << ans << endl;
29
30 ☐ int main() {
31
         init(); qsort(1, n); solve();
32
         return 0;
33
```

#include<iostream>

完善程序

完善程序

(细胞问题)一个矩形阵列,由数字0到9组成。数字1到9是代表细胞。细胞的定义是,沿细胞数字上下左右四个方向,只要还是非零的细胞数字,则被视为同一个细胞,求给出的矩形阵列的细胞总个数。

程序在第一行输入两个数字,分别表示矩阵的行数m和列数n,接下来m行,每行n个个位数字,表示整个m*n矩阵的情况。

手算样例

输入样例:

4 10 0234500067 1034560500

2045600671

000000089

输出多少?

输出样例:

4

```
int direction_x[4] = ____(1)____;
    int direction_y[4] = {0, 1, 0, -1};
    int flag[100][100], m, n, num_of_cells = 0;
    void find(int start x, int start y) {
8
        int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
        num of cells++; t = 0; w = 1;
                                                                       识别变量
10
        queue[1][1]=start_x; queue[1][2]=start_y;
11 🖹
        do {
                                                                    常见变量名
12
            t++:
                                                                   翻译循环变量
13 🖹
            for (int i = 0; i <= 3; i++) {
14
                int x=queue[t][1]+direction_x[i];
                                                               根据变量名的英文推断
15
                int y=___(2)___;
                if((x>=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {
16 🖃
17
                    w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18
                                                                     找出关键语句
19
        }while( (3) );
20
                                                                  控制结构(for, if)
21
                                                                常见算法的基本操作
22 ☐ int main(){
                                                                 函数参数、返回值
23
        int i, j; string s; cin >> m >> n;
24
        for(i = 0; i <= m - 1; i++)
                                                               3
                                                                    理解代码段作用
25
            for(j = 0; j \leftarrow n - 1; j++) flag[i][j] = 1;
26 🖨
        for(i = 0; i <= m - 1; i++) {
                                                                  翻译解释代码段
27
            cin >> s:
28
            for(j=0;j<=n-1;j++) if(s[j]=='0') ___(4)___;
29
30
        for(i = 0; i <= m - 1; i++)
31
            for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)___;</pre>
        cout<<"NUMBER of cells is "<< num_of_cells << endl;</pre>
32
33
        return 0;
```

```
int direction_x[4] = ____(1)____;
                                                                解释变量作用
    int direction_y[4] = {0, 1, 0, -1};
    int flag[100][100], m, n, num_of_cells = 0;
    void find(int start x, int start y) {
        int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
8
        num of cells++; t = 0; w = 1;
10
        queue[1][1]=start_x; queue[1][2]=start_y;
                                                                  行数: 第0行到第m-1行
                                                           m
11 🖹
        do {
                                                                  列数:第0列到第n-1列
12
                                                           n
            t++;
13 🖹
            for (int i = 0; i <= 3; i++) {
                                                        flag[i][j]
                                                                  标记(i,j)位是否是细胞
14
                int x=queue[t][1]+direction_x[i];
15
                int y = ___(2)___;
                if((x>=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {</pre>
16 🖃
17
                    w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18
19
20
        }while(___(3)___);
                                                         游标t
                                                                  标识队列中当前访问位
21
22 ☐ int main(){
                                                         游标W
                                                                  队尾,新点入队则加1
23
        int i, j; string s; cin >> m >> n;
24
        for(i = 0; i <= m - 1; i++)
                                                                    存放队列中各点行号
                                                        queue[][1]
25
            for(j = 0; j <= n - 1; j++) flag[i][j] = 1;
26 🖨
        for(i = 0; i <= m - 1; i++) {
                                                                    存放队列中各点列号
                                                        queue[][2]
27
            cin >> s:
28
            for(j=0;j<=n-1;j++) if(s[j]=='0') ____(4)___
29
                                                                         连通块数量
                                                        num_of_cells
30
        for(i = 0; i <= m - 1; i++)
31
            for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)___;</pre>
32
        cout<<"NUMBER of cells is "<< num_of_cells << endl;</pre>
33
        return 0:
```

```
int direction x[4] = (1)
                                                  设置连通四方向数组d_x和d_y
    int direction_y[4] = {0, 1, 0, -1}
    int flag[100][100], m, n, num_of_cells = 0;
    void find(int start_x, int start_y) {
        int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
        num of cells++; t = 0; w = 1;
10
        queue[1][1]=start_x; queue[1][2]=start_y;
                                                      从(start_x, start_y)开始
11 -
        do {
                                                  通过一个二维数组模拟的队列
12
           t++;
13 🖹
                                                  用BFS发洪水算法搜索并标记
            for (int i = 0; i <= 3; i++) {
14
               int x=queue[t][1]+direction_x[i];
15
               int y = (2);
               if((x)=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y]))  {
16
                   w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
17
18
19
        }while(___(3)___);
20
22 | int main(){
23
        int i, j; string s; cin >> m >> n;
24
        for(i = 0; i <= m - 1; i++)
25
           for(j = 0; j <= n - 1; j++) flag[i][j] = 1;
26 🖃
        for(i = 0; i <= m - 1; i++) {
                                                     输入矩阵并标记flag数组
27
            cin >> s;
28
            for(j=0;j<=n-1;j++) if(s[j]=='0') ____(4)__
29
                                                  统计矩阵中四方向连通块数量
        for(i = 0; i <= m - 1; i++)
30
31
            for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)_</pre>
32
        cout<<"NUMBER of cells is "<< num of cells << endl;
                                                                关键语句
33
        return 0:
```

```
int direction_x[4] = ____(1)____;
                                                从(start_x, start_y)开始搜索
    int direction_y[4] = {0, 1, 0, -1};
                                               通过一个二维数组模拟的队列
    int flag[100][100], m, n, num of cells = 0;
                                               用BFS发洪水算法访问并标记
    void find(int start_x, int start_y) {
       int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
       num of cells++; t = 0; w = 1;
                                                1:定义二维数组queue模拟队列
10
       queue[1][1]=start_x; queue[1][2]=start_y;
                                              2:初始化当前访问位游标t和队尾w
11 -
       do {
                                              3:将起始点flag标记为0并加入队列
12
           t++;
                                              4: 连通块统计量num_of_cells加1
13 <u></u>
           for (int i = 0; i <= 3; i++) {
                                                游标t++移动到应访问的队首点
14
              int x=queue[t][1]+direction_x[i];
              int y=___(2)___;
15
                                              利用d_x和d_y访问目前点四方向点
16 -
              if((x>=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {
17
                 w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18
                                              如果该邻居点还在矩阵范围内且是
19
20
       }while( (3) );
                                              未访问过的细胞点(x,y) 则加入队列
21
22 ☐ int main(){
                                                 只要队列中还有元素(t<w)
23
       int i, j; string s; cin >> m >> n;
                                               整个BFS搜索通过while持续进行
24
       for(i = 0; i <= m - 1; i++)
                                              遍历完整个连通块并取消它们标记
25
           for(j = 0; j <= n - 1; j++) flag[i][j]</pre>
26 🗀
       for(i = 0; i <= m - 1; i++) {
27
           cin >> s;
           for(j=0;j<=n-1;j++) if(s[j]=='0') ___(4)___;
28
29
                                               统计矩阵中四方向连通块数量
       for(i = 0; i <= m - 1; i++)
30
           for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)___;</pre>
31
       cout<<"NUMBER of cells is "<< num_of_cells << endl;
32
                                                            关键语句
33
       return 0:
```

```
int direction x[4] = (1);
                                             \{-1, 0, 1, 0\}
    int direction_y[4] = {0, 1, 0, -1}:
5
     int flag[100][100], m, n, num_of_cells = 0;
    void find(int start x, int start y) {
8
        int t=0, w=1, queue[1000][2], flag[start x][start y]=0;
9
        num of cells++: t = 0: w = 1:
10
        queue[1][1]=start x: queue[1][2]=start y:
11 -
        do {
12
            t++;
13 🖹
            for (int i = 0; i <= 3; i++) {
14
                int x=queue[t][1]+direction_x[i];
                int y=___(2)___; queue[t][2]+direction_y[i]
15
16 =
                if((x)=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {
17
                    w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18
19
                                t<w
        }while( (3) );
20
22 ☐ int main(){
23
        int i, j: string s: cin >> m >> n;
24
        for(i = 0; i <= m - 1; i++)
25
            for(j = 0; j <= n - 1; j++) flag[i][j] = 1;
26 =
        for(i = 0; i <= m - 1; i++) {
27
            cin >> s;
28
            for(j=0;j<=n-1;j++) if(s[j]=='0')
29
30
        for(i = 0; i <= m - 1; i++)
                                                            find(i, j)
31
            for(j=0;j<=n-1;j++) if(flag[i][j]) (5)
32
        cout<<"NUMBER of cells is "<< num of cells << endl;
33
        return 0:
```