1. for循环的程序步的计算方法：

中文书24页有介绍for循环程序步的计算方法：

一般情况下一个for循环计一个程序步，注意循环语句块内的程序执行n次时，for语句执行n+1次， 在退出for循环时会多计算一次。

2. 有同学将count++写在for循环内，如

```
for(int i = 0; i < m,count++; i ++)
```

上课的时候说错了，这样是计算不出来count的值的。for的判断语句是***i<m,count++*** ,c++中逗号表达式的值返回最右边的式子的值，所以会返回count++的值，因为count初始值为0，count++返回的值为0，所以执行第一次for时就会跳出for循环，压根不会进入循环体。

或

```
for(int i = 0; i < m; i ++,count++)
```

根据for循环的执行顺序，需要在最后加 last for的count++;

3. for循环执行的顺序。

4. 程序简化：

目的是为了计算步程数，所以简化掉计算语句，留下一些计数语句如i++，并且将count++合并，最外层的count++也可以合并。

5. 是求程序步而不是用大O法表示时间复杂度。

6.计算程序步的表格还是 按书中频度表的格式写。

3. Determine the frequency counts for all statements in the following two program segments:

```
for (i = 1; i <= n; i++)
  for (j = 1; j <= i; j++)
    for (k = 1; k <= j; k++)
      x++;
```

(a)

```
i = 1;
while (i ≤ n)
{
    x++;
    i++;
}
```

(b)

a:

程序	程序步		
for(i=1; i<=n; i++)	n+1	n+1	
for(j=1;j<=i;j++)	$\frac{n^2}{2}+\frac{3n}{2}$	$\frac{n(n+3)}{2}$	$\sum_{i=1}^{n}(i+1)$
for(k=1; k<=j; j++)	$\frac{n^3}{6}+n^2+\frac{5n}{6}$	$\frac{n(n+1)(n+5)}{3}$	$\sum_{i=1}^{n}\sum_{j=1}^{i}(j+1)$
x++;	$\frac{n^3}{6}+\frac{n^2}{2}+\frac{n}{3}$	$\frac{n(n+1)(n+2)}{6}$	$\sum_{i=1}^{n}\sum_{j=1}^{i}j$
	总： $\frac{n^3}{3}+2n^2+\frac{11n}{3}+1$		

行	s/e	频度	行步程数
1	1	n+1	n+1
2	1	$\frac{n^2}{2}+\frac{3n}{2}$	$\frac{n^2}{2}+\frac{3n}{2}$
3	1	$\frac{n^3}{6}+n^2+\frac{5n}{6}$	$\frac{n^3}{6}+n^2+\frac{5n}{6}$
4	1	$\frac{n^3}{6}+\frac{n^2}{2}+\frac{n}{3}$	$\frac{n^3}{6}+\frac{n^2}{2}+\frac{n}{3}$
		总步程数：	$\frac{n^3}{3}+2n^2+\frac{11n}{3}+1$

b:

注意：

行	s/e	频度	行步程数
1	1	1	1
2	1	N+1	N+1
3	0	1	0
4	1	n	n
5	1	n	n
		总步程数：	3n + 2

4. (a) Introduce statements to increment *count* at all appropriate points in Program 1.32.

```
void D(int *x, int n)
{
    int i = 1;
    do {
        x[i] += 2;
        i += 2;
    }
    while (i <= n);
    i = 1;
    while (i <= (n /2))
    {
        x[i] += x[i+1];
        i++;
    }
}
```

Program 1.32: Example program

(b) Simplify the resulting program by eliminating statements. The simplified program should compute the same value for *count* as computed by the program of (a).

(c) What is the exact value of *count* when the program terminates? You may assume that the initial value of *count* is 0.

(d) Obtain the step count for Program 1.32 using the frequency method. Clearly show the step count table.

注意:

1. 取整

向下取整与向上取整

对任意实数 x，我们用 $\lfloor x \rfloor$ 表示小于或等于 x 的最大整数(读作“x 的向下取整”)，并用 $\lceil x \rceil$ 表示大于或等于 x 的最小整数(读作“x 的向上取整”)。对所有实数 x，

$$x-1<\lfloor x\rfloor \leqslant x \leqslant \lceil x\rceil < x+1 \tag{3.3}$$

2. do while循环的程序步:

do内程序块和while语句执行的次数一样

a: 添加程序步骤计数语句

```
void D(int * x, int n) {
 int i = 1;
 count++; //i赋值
 do {
 x[i] += 2;
 i += 2;
 count += 2; //循环体内语句程序步
 count++; //while条件程序步
 } while (i <= n);
  //count++; 没有这一步
  i = 1;
  count++; //i赋值
  while (i <= (n / 2)) {
  count++; //while条件

  x[i] += x[i + 1];
  i++;

  count += 2;//循环体内语句程序步
  }
  count++; //出循环的最后一个while条件
}
```

b: 简化计数程序

```
void D(int * x, int n) {
 int i = 1;
 do {
 i += 2;
 count += 3;
 } while (i <= n);
 i = 1;
 while (i <= (n / 2)) {
 i++;
```

```
10     count += 3;
11     }
12     count += 3;
13   }
```

c: 程序的准确程序步

3n + 3

d: 给程序的行编号如下：

```
1   void D(int * x, int n) {
2     int i = 1;
3     do {
4     x[i] += 2;
5     i += 2;
6     } while (i <= n);
7     i = 1;
8     while (i <= (n / 2)) {
9     x[i] += x[i + 1];
10     i++;
11     }
12  }
```

	n为奇	n为偶	任意情况
int i=1;	1	1	1
do{	0	0	0
x[i] += 2;	$\frac{n+1}{2}$	$\frac{n}{2}$	$\lceil \frac{n}{2} \rceil$
i += 2;	$\frac{n+1}{2}$	$\frac{n}{2}$	$\lceil \frac{n}{2} \rceil$
}while(i<=n);	$\frac{n+1}{2}$	$\frac{n}{2}$	$\lceil \frac{n}{2} \rceil$
i=1;	1	1	1
while(i<=(n/2))	$\frac{n-1}{2}+1$	$\frac{n}{2}+1$	$\lfloor \frac{n}{2} \rfloor+1$
{	0	0	0
x[i] += x[i+1];	$\frac{n-1}{2}$	$\frac{n}{2}$	$\lfloor \frac{n}{2} \rfloor$
i++;	$\frac{n-1}{2}$	$\frac{n}{2}$	$\lfloor \frac{n}{2} \rfloor$
}	0	0	0
	$3n+3$	$3n+3$	$3\lceil \frac{n}{2} \rceil+3\lfloor \frac{n}{2} \rfloor+3$

5. Do Exercise 4 for function *Transpose* (Program 1.33).

```
void Transpose(int **a; int n)
{
    for (int i = 0; i < n-1; i++)
        for (int j = i+1; j < n; j++)
            swap (a[i][j], a[j][i]);
}
```

Program 1.33: Matrix transpose

a: 添加计算步程数语句

```
void Transpose(int ** a, int n) {
 for (int i = 0; i < n - 1; i++) {
 count++;
 for (int j = i + 1; j < n; j++) {
 count++;
 swap(a[i][j], a[j][i]);
 count++;
 }
 count++;
  }
  count++;
}
```

b:简化的计算步程数程序：

```
void Transpose(int ** a, int n) {
 for (int i = 0; i < n - 1; i++) {
 count += 2;
 for (int j = i + 1; j < n; j++) {
 count += 2;
 }
 }
 count++;
}
```

c：总步程数

$$n^2 + n - 1$$

d：给程序行编号

```
void Transpose(int ** a, int n) {
 for (int i = 0; i < n - 1; i++) {
 for (int j = i + 1; j < n; j++) {
 swap(a[i][j], a[j][i]);
```

```
    }
  }
}
```

程序	单层循环步数	程序步
for(int i=0;i<n-1;i++)	i=0~n-2; (n-2-0+1)+1 = n	n
for(int j=i+1;j<n;j++)	j=i+1~n-1; [n-1-(i+1)+1]+1 = n-i	$\sum_{i=0}^{n-2}(n-i) = \frac{n^2}{2}+\frac{n}{2}-1$
swap(a[i][j],a[j][i]);	n-i-1	$\sum_{i=0}^{n-2}(n-i-1) = \frac{n^2}{2}-\frac{n}{2}$
		总： n^2+n-1

行	s/e	频度	行步程数
2	1	n	n
3	1	$\frac{n^2}{2}+\frac{n}{2}-1$	$\frac{n^2}{2}+\frac{n}{2}-1$
4	1	$\frac{n^2}{2}-\frac{n}{2}$	$\frac{n^2}{2}-\frac{n}{2}$
		总步程数：	n^2+n-1

6. Do Exercise 4 for Program 1.34. This program multiplies two $n \times n$ matrices a and b.

```
void Multiply( int **a, int **b, int **c, int n)
{
    for (int i = 0; i < n ; i++)
        for (int j = 0; j < n ; j++)
        {
            c[i][j] = 0;
            for (int k = 0; k < n ; k++)
                c[i][j] += a[i][k] * b[k][j];
        }
}
```

Program 1.34: Square matrix multiplication

a: 添加计算步程数语句

```
void Multiply(int ** a, int **b, int **c, int n) {
 for (int i = 0; i < n; i++) {
 count++; //for循环
 for (int j = 0; j < n; j++) {
 count++; //for循环
```

```
c[i][j] = 0;
count++; //赋值
for (int k = 0; k < n; k++) {
count++; //for循环
 c[i][j] += a[i][k] * b[k][j];
 count++; //计算
 }
 count++; //for循环最后一步
 }
 count++; //for循环最后一步
 }
 count++; //for循环最后一步
}
```

b:简化的计算步程数程序:

```
void Multiply(int ** a, int **b, int **c, int n) {
 for (int i = 0; i < n; i++) {
 count += 2;
 for (int j = 0; j < n; j++) {
 count += 3;
 for (int k = 0; k < n; k++) {
 count += 2;
 }
 }
  }
  count++;
}
```

c：总步程数

$$2n^3 + 3n^2 + 2n + 1$$

d：给程序行编号

```
void Multiply(int ** a, int **b, int **c, int n) {
 for (int i = 0; i < n; i++) {
 for (int j = 0; j < n; j++) {
 c[i][j] = 0;
 for (int k = 0; k < n; k++) {
 c[i][j] += a[i][k] * b[k][j];
 }
 }
 }
}
```

行	s/e	频度	行步程数
2	1	n + 1	n + 1
3	1	n(n + 1)	n(n + 1)
4	1	n^2	n^2
5	1	$n^2(\boldsymbol{n}+\mathbf{1})$	$n^2(\boldsymbol{n}+\mathbf{1})$
6	1	$\boldsymbol{n^3}$	$\boldsymbol{n^3}$
		总步程数：	$\mathbf{2}\boldsymbol{n}^{\mathbf{3}}+\ \mathbf{3}\boldsymbol{n}^{\mathbf{2}}+\mathbf{2}\boldsymbol{n}+\mathbf{1}$

7. (a) Do Exercise 4 for Program 1.35. This program multiplies two matrices *a* and *b* where *a* is an $m \times n$ matrix and *b* is an $n \times p$ matrix.

(b) Under what conditions will it be profitable to interchange the two outermost **for** loops?

```
void Multiply(int **a, int **b, int **c, int m, int n, int p)
{
    for (int i = 0; i < m ; i++)
        for (int j = 0; j < p ; j++)
        {
            c [i ][j ] = 0;
            for (int k = 0; k < n ; k++)
                c [i ][j ] += a [i ][k ] * b [k ][j ];
        }
}
```

Program 1.35: Matrix multiplication

问题一：

a: 添加计算步程数语句

```
void Multiply(int **a, int **b, int **c, int m, int n, int p) {
 for (int i = 0; i < m; i++) {
 count++;
 for (int j = 0; j < p; j++) {
 count++;
 c[i][j] = 0;
 count++;
 for (int k = 0; k < n; k++) {
```

```
count++;
 c[i][j] += a[i][k] * b[k][j];
 count++;
 }
 count++;
 }
 count++;
 }
 count++;
}
```

b:简化的计算步程数程序:

```
void Multiply(int **a, int **b, int **c, int m, int n, int p) {
 for (int i = 0; i < m; i++) {
 count += 2;
 for (int j = 0; j < p; j++) {
 count += 3;
 for (int k = 0; k < n; k++) {
 count += 2;
 }
 }
  }
  count++;
}
```

c：总步程数

2mpn+3mp+2m+1

d：给程序行编号

```
void Multiply(int **a, int **b, int **c, int m, int n, int p) {
 for (int i = 0; i < m; i++) {
 for (int j = 0; j < p; j++) {
 c[i][j] = 0;
 for (int k = 0; k < n; k++) {
 c[i][j] += a[i][k] * b[k][j];
 }
 }
 }
}
```

行	s/e	频度	行步程数
2	1	m + 1	m + 1
3	1	m(p + 1)	m(p + 1)
4	1	mp	mp
5	1	mp(n + 1)	mp(n + 1)
6	1	mpn	mpn
		总步程数：	2mpn+3mp+2m+1

问题二：

计算得出程序的总步程数是**2mpn+3mp+2m+1**，外两层循环涉及到的变量为m和p，交换外两次循环不会影响2mpn+3mp的大小，但是会影响2m的大小，因此当m > p时，交换外两层循环的步程数为**2mpn+3mp+2p+1** < **2mpn+3mp+2m+1**,因此此时交换能提高效率。

程序题：

2. Given n Boolean variables $x_1, \cdots, x_n$ we wish to print all possible combinations of truth values they can assume. For instance, if $n = 2$, there are four possibilities: true, true; true, false; false, true; false, false. Write a C++ program to accomplish this and do a frequency count.

方法：

1：递归

2：位运算

以下的程序没有用上面的任何一种方法，自己随便写的，同学们之后写程序的时候，可以多用STL库中包装好的东西，比如<vector>,<stack>,<queue>,<string>, <unordered_map>,<list>等等代替使用char*，int[],int*等东西。

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

vector<vector<string>> solution(int n) {
```

```
if (n <= 0) {
cout << "n必须大于等于1。\n";
exit(0);
}

 //当n等于1时的结果数组
 vector<vector<string>> res;
 res.push_back({ "false" });
 res.push_back({ "true" });

 if (n == 1) {
 return res;
 }
 //将数组中的每一个值分别追加true和false，没循环一次，数组扩大一倍
 //例如{ {"false"} ,{ "true"}}会变为
 //{ {"false", "true"} ,{ "true", "true"},{ "true", "false"},{ "true", "false"}}
 for (int i = 2; i <= n; i++) {
 int size = res.size();
 for (int j = 0; j < size; j++) {
 vector<string> a = res[j];
 a.push_back("true");
 res.push_back(a);
 res[j].push_back("false");
 }
 }
 return res;
}

//int main() {
// vector<vector<string>> temp = solution(9);
// int raw = temp.size();
// int col = temp[0].size();
// for (int i = 0; i < raw; i++) {
// cout << "( " << temp[i][0].c_str();
// for (int j = 1; j < col; j++) {
// cout << " , " << temp[i][j].c_str();
// }
// cout << " ) " << endl;
// }
```

```
// cout << "有" << temp.size() << "中组合。\n";
//}
```

14. If S is a set of n elements, the *powerset* of S is the set of all possible subsets of S. For example, if $S = (a,b,c)$, then *powerset* $(S) = \{(\,), (a), (b), (c), (a,b), (a,c), (b,c), (a,b,c)\}$. Write a recursive function to compute

powerset (S).

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

vector<vector<string>> util(vector<string> S, int start, int end) {
 vector<vector<string>> res;
 if (start > end) {
 res.push_back({});
 return res;
  }
  res = util(S, start, end - 1);
  int size = res.size();
  for (int i = 0; i < size; i++) {
  vector<string> temp = res[i];
  temp.push_back(S[end]);
  res.push_back(temp);
  }
  return res;
}


vector<vector<string>> solution(vector<string> S) {
 return util(S, 0, S.size() - 1);
}

//
//int main() {
// vector<string> S{ "a","b","c" ,"d"};
// vector<vector<string>> res = solution(S);
// int size = res.size();
// cout << "{ ";
```

```
//
// for (int i = 0; i < size; i++) {
//
// int size1 = res[i].size();
// if (size1 == 0)
// cout << "( )";
// else {
// cout << "( " << res[i][0].c_str();
// for (int j = 1; j < res[i].size(); j++) {
// cout << " , " << res[i][j].c_str() ;
// }
// cout << " )";
// }
// if( i != size - 1)
// cout << " , ";
// }
// cout << " }";
// cout << endl;
// cout << "超集中共有" << size << "个元素" << endl;
//}
```