

循 环

基础 - 循环输出

```
#include <iostream>//1-1881
using namespace std;
int main()
{
    int i=1;
    while(i<=100)
    {
        cout<<i<<endl;
        i++;
    }
    return 0;
}
```

```
#include <iostream>//2-1882
using namespace std;
int main()
{
    int i=100;
    while(i>=1)
    {
        cout<<i<<endl;
        i--;
    }
    return 0;
}
```

```
#include <iostream>//3-1696 推荐学习
using namespace std;
int main()
{
    //i 用来循环, n 用来表示最大数
    int i, n;
    cin>>n;

    i = 1;

    //2、循环条件
    while(i <= n)//3、让循环停止的方式
    {
        cout<<i<<endl;
        i = i + 1;
    }
    return 0;
}
```

```
#include <iostream>//4-1697 推荐学习
using namespace std;
int main()
{
    //i 用来循环, n 用来表示最大数
    int i, n;
    cin>>n;

    //2、循环条件
    while(1 <= n)//3、让循环停止的方式
    {
        cout<<n<<endl;
        n = n - 1;
    }
    return 0;
}
```

```
#include <iostream>//5-1698
using namespace std;

/*
    题目要求打印 1~n 中满足条件的数
    第一步：打印 1~n 的每一个数
    第二步：判断每个数是否满足条件，如果满足条件就打印
*/
int main()
{
    int i, n, g;
    cin>>n;
    i = 1;

    while(i <= n)
    {
        // 拆出每个数的个位
        g = i % 10;
        // 不是每个数都要输出
        // 必须是个位是 1 3 5 7 中任意一个数的数，才能输出
        if(g==1 || g==3 || g==5 || g==7)
        {
            cout<<i<<endl;
        }

        i = i + 1;
    }
    return 0;
}
```

```
#include <iostream>//6-1699
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i;
    // 遍历 1 到 n 的整数
    for (i=1; i<=n; i++)
    {
        // 判断 i 是 2 的倍数，但非 3 的倍数的数
        if (i%2==0&& i%3!=0)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

```
#include <iostream>//7-1700-1 javacn
using namespace std;

/*
    第一步：打印 10~99 的每一个数（两位数）
    第二步：判断每个数是否满足条件，如果满足条件就打印
*/
int main()
{
    int i, g, s;
    //1、初始值
    i = 10;
    //2、循环条件
    while(i <= 99)
    {
        s = i / 10;
        g = i % 10;
        //判断满足条件
        if(s == 2 || g == 2)
        {
            cout<<i<<endl;
        }

        //3、让循环停止的方法
        i = i + 1;
    }
    return 0;
}
```

```
#include <iostream> //7-1700-2 javacn
using namespace std;
int main()
{
    int i, s, g;
    // 遍历所有的 2 位数
    for (i=10; i<=99; i++)
    {
        // 对 i 拆位求出个位和十位
        s=i/10;
        g=i%10;
        // 判断个位和十位是否有等于 2 的并输出
        if (s==2 || g==2)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

```
#include <iostream>//8-1701-1 javacn
using namespace std;

/*
循环 + 筛选
分步骤求解：
1、循环所有的 3 位数；
2、判断每个数是否满足条件，满足条件再输出；
*/
int main()
{
    int i, b, g;
    // 初始值
    i = 100;
    // 循环条件
    while(i <= 999)
    {
        b = i / 100;
        g = i % 10;
        // 如果是对称的，则输出
        if(b == g)
        {
            cout<<i<<endl;
        }

        i = i + 1;// 让循环停止的方法
    }
    return 0;
}
```

```

#include <iostream>//8-1701-2 javacn
using namespace std;
int main()
{
    int i, b, g;
    for (i=100; i<=999; i++)    // 遍历所有的三位整数
    {
        b=i/100;// 拆位求出百位和个位
        g=i%10;
        if (b==g) // 判断百位和个位是否相等
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```

```

#include <iostream>//9-1711
using namespace std;
int main()
{
    int i, s, g, sum;
    for (i=10; i<=99; i++) // 遍历所有的 2 位数
    {
        s=i/10;// 拆位求出十位和个位
        g=i%10;
        sum=s+g;    // 求十位和个位的和
        // 如果十位大于个位，且和是偶数，就输出这个数
        if (s>g&&sum%2==0)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```



```

#include<bits/stdc++.h>//10-1712
using namespace std;
int main()
{
    int i, b, s, g, sum;
    // 遍历所有的三位数
    for (i=100; i<=999; i++)
    {
        // 拆位求出百位，十位，个位，并求和
        b=i/100;
        s=i/10%10;
        g=i%10;
        sum=b+s+g;
        // 各位的和为偶数且百位大于十位，十位大于个位，则输出这个数
        if (sum%2==0&& b>s&& s>g)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```

```

#include<bits/stdc++.h>//11-1713
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i;
    for (i=1; i<=n; i=i+3) // 遍历 1 到 n 的数，每次 +3
    {
        cout<<i<<endl;
    }
    return 0;
}

```

// 遍历 1-n 将个，十，百位（n<1000），分别求出来，判断是否满足要求就行了

```
#include<bits/stdc++.h>//12-1714 输出满足条件的整数 wintereta
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n;
```

```
    cin>>n;
```

```
    for (int i=1; i<=n; i++)
```

```
    {
```

```
        int g=i/1%10;
```

```
        int s=i/10%10;
```

```
        int b=i/100%10;
```

```
        if (g==3 || g==5 || s==3 || s==5 || b==3 || b==5)
```

```
        {
```

```
            if (i%2==0)
```

```
            {
```

```
                cout<<i<<endl;
```

```
            }
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

```

#include<bits/stdc++.h>//13-1715
using namespace std;
int main()
{
    int q, b, s, g, sum1, sum2, i;
    for (i=1000; i<=9999; i++)// 遍历所有的 4 位数 拆位求出千位, 百位, 十位, 个位
    {
        q=i/1000;
        b=i/100%10;
        s=i%100/10;
        g=i%10;
        sum1=q+b;// 千位和百位的和
        sum2=s+g;// 十位和个位的和
        if (sum1%2==0&&sum2%2==1&&sum1>sum2&&i%8==0)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```

// 千位和百位之和为偶数
// 十位和个位之和为奇数
// 且前两位之和大于后两位之和
// 且含有因数 8

```

#include<bits/stdc++.h>//14-1721
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i;
    for (i=1; i<=n; i++)        // 遍历 1 到 n 的所有整数
    {
        if (i%10==5 || i%10==8) // 判断个位为 5 或者个位为 8, 则输出
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```

```

#include<bits/stdc++.h>//15-1722    输出两位的巧数
using namespace std;
int main()
{
    int i, s, g, sum;
    // 遍历所有的 2 位整数
    for (i=10; i<=99; i++)
    {
        s=i/10;        // 拆位求出十位, 个位
        g=i%10;
        sum=s+g+s*g;    //sum 为各个位数字之和加上各个位数字之积
        if (sum==i)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```

```
#include <bits/stdc++.h>//16-1017
using namespace std;
int main()
{
    int i;// 定义整数变量 i 代表零件数量
    for (i=101;i<=199;i++)// 零件数为 100 多个
    {
        // 零件数被 3 除余 2, 被 5 除余 3, 被 7 除余 5
        if (i%3==2&& i%5==3&& i%7==5)
        {
            // 找到第一个符合条件的零件数跳出循环
            cout<<i;
            break;
        }
    }
    return 0;
}
```

```
#include <bits/stdc++.h>//17-1021
using namespace std;
int main() {
    int i;
    for (i=1;i<=500;i++) // 循环 i 的范围 1-500
    {
        // 满足用 3 除余 2, 用 5 除余 3, 用 7 除余 2 则输出 i
        if (i%3==2&& i%5==3&& i%7==2)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

```
#include<iostream>//18-1058-1 推荐学习 javacn
```

```
using namespace std;
```

```
void sxhs(int n)
```

```
{
```

```
    if(n <= 999)
```

```
    {
```

```
        int a = n / 100;
```

```
        int b = n / 10 % 10;
```

```
        int c = n % 10;
```

```
        if(a * a * a + b * b * b + c * c * c == n)
```

```
        {
```

```
            cout << n << endl;
```

```
        }
```

```
        sxhs(n + 1);
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    sxhs(100);
```

```
    return 0;
```

```
}
```

```

#include<bits/stdc++.h>//18-1058-2 仅供参考
using namespace std;
/*
    函数名 lsum(int x)
    参数 int x
    返回值 int r
    说明 求各位数字立方之和
*/
int lsum(int x)
{
    int r=0;//用于存放和,初始值为0
    int a;
    //用短除法循环求各个位的立方和
    while(x/10!=0||x%10!=0)
    {
        a=x%10;
        r=r+a*a*a;
        x=x/10;
    }
    return r;
}
int main()
{
    int i;
    //循环范围为100到999的整数
    for(i=100;i<=999;i++)
    {
        //如果i的各位数字立方之和=i就输出i
        if(lsum(i)==i)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```

```
#include<bits/stdc++.h>//19-1059
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int i;           // 定义整数变量 i 用作循环
```

```
    for (i=1; i<=999; i++) // 从 1 到 999 做循环
```

```
    {
```

```
        if (i%3==0) // 判断 i 是否有因数 3
```

```
        {
```

```
            if (i%10==5 || i/10%10==5 || i/100==5) // 判断 i 中至少有一位数字是 5
```

```
            {
```

```
                cout<<i<<endl;
```

```
            }
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

```
#include<bits/stdc++.h>//20-1264
```

```
using namespace std;
```

```
int main() {
```

```
    int i, q, b, s, g, m;
```

```
    for (i=1000; i<=9999; i++)
```

```
    {
```

```
        q=i/1000;
```

```
        b=i/100%10;
```

```
        s=i%100/10;
```

```
        g=i%10;
```

```
        m=g*1000+s*100+b*10+q;
```

```
        if (9*i==m)
```

```
        {
```

```
            cout<<i<<endl;
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```



```

#include<bits/stdc++.h>//21-1266
using namespace std;
int main()
{
    int i;
    // 遍历所有三位数 ， 从大的数开始
    for (i=999;i>=100;i--)
    {
        if (555555%i==0) // 如果能整除 555555 就输出 i，跳出循环
        {
            cout<<i;
            break;
        }
    }
    return 0;
}

```

```

#include<bits/stdc++.h>//22-1392
using namespace std;
int main()
{
    int i, b, g;
    for (i=100;i<=999;i++) // 遍历所有的三位数
    {
        b=i/100; // 拆位求出百位和个位
        g=i%10;
        // 判断百位和个位相等且这个三位数是个偶数，则输出这个三位数
        if (b==g&& i%2==0)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```

```
#include<bits/stdc++.h> //23-1447
using namespace std;
int main()
{
    // 定义千位，百位，十位，个位
    int i, q, b, s, g;
    // 遍历所有 4 位数
    for (i=1000; i<=9999; i++)
    {
        q=i/1000;
        b=i/100%10;
        s=i%100/10;
        g=i%10;
        // 如果千位 = 个位，百位 = 十位说明是回文数
        if (q==g&& b==s)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

```

#include<bits/stdc++.h>//24-1466
using namespace std;
int main()
{
    //m, n 为输入的两个整数
    //q, b, s, g 为千位，百位，十位，个位
    int i, q, b, s, g, m, n;
    cin>>m>>n;
    // 遍历 m 到 n 的整数
    for (i=m; i<=n; i++)
    {
        // 对 i 拆位
        q=i/1000;
        b=i/100%10;
        s=i%100/10;
        g=i%10;
        // 判断是三位数还是 4 位数
        if (q==0)
        {
            if (b-s==s-g)
            {
                cout<<i<<endl;
            }
        }
        else
        {
            if (q-b==b-s&&b-s==s-g)
            {
                cout<<i<<endl;
            }
        }
    }
    return 0;
}

```

```
#include<bits/stdc++.h>//25-1737
using namespace std;
int main()
{
    int n, q, b, s, g, m;
    cin>>n;
    int i;
    // 遍历所有 4 位整数
    for (i=1000; i<=n; i++)
    {
        // 对 i 进行拆位
        q=i/1000;
        b=i/100%10;
        s=i%100/10;
        g=i%10;
        // 千位和十位对调，百位和个位对调
        m=s*1000+g*100+q*10+b;
        // 对调后仍然等于本身且该数为奇数
        if (i==m&& i%2==1)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

```
#include<bits/stdc++.h>//26-1746
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i;
    // 遍历 1 到 n 之间的整数
    for (i=1;i<=n;i++)
    {
        int x=0;// 记录能够被 2、3、5、7 中几个数整除
        if (i%2==0)
        {
            x++;
        }
        if (i%3==0)
        {
            x++;
        }
        if (i%5==0)
        {
            x++;
        }
        if (i%7==0)
        {
            x++;
        }
        // 输出符合两个及两个以上条件的数
        if (x>1)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

```
#include <bits/stdc++.h>//27-1748
using namespace std;
int main()
{
    int i, b, s, g;
    for (i=100; i<=999; i++) // 遍历 100 到 999 的所有三位整数
    {
        // 对 i 进行拆位
        b=i/100;
        s=i/10%10;
        g=i%10;
        if (s>b&&g>s) // 判断十位比两边的个位和百位都大
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

```
#include<bits/stdc++.h>//28-1861
using namespace std;
int main()
{
    int i, mn, mx, factor;
    cin>>mn>>mx>>factor;
    for (i=mn; i<=mx; i++)
    {
        if (i%factor==0)
        {
            cout<<i<<" ";
        }
    }
    return 0;
}
```

```
#include<bits/stdc++.h>//29-1863
using namespace std;
int main()
{
    int i, q, b, s, g, sum;
    for (i=1000; i<=9999; i++) // 遍历所有的四位数
    {
        q=i/1000;
        b=i/100%10;
        s=i%100/10;
        g=i%10;
        sum=q+b+s+g; // 求各位数字和
        // 如果各位数字和等于 10 就输出这个数
        if (sum==10)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

```

#include<bits/stdc++.h>//30-1085
using namespace std;
int main()
{
    int i, qb, sg, sum;//sum 为前两位 + 后两位的和
    // 对 i 循环范围为 1000 到 9999
    for (i=1000; i<=9999; i++)
    {
        // 对 i 拆位求出前两位和后两位
        qb=i/100;
        sg=i%100;
        sum=qb+sg;
        if (sum*sum==i)
        {
            // 如果 sum 的平方等于 i 就输出 i
            cout<<i<<endl;
        }
    }
    return 0;
}

```

```

#include<bits/stdc++.h>//31-1090
using namespace std;
int main()
{
    int i;
    // 遍历范围为 10 到 1000
    for (i=10; i<=1000; i++)
    {
        // 能同时被 2、3、7 整除的 i 输出
        if (i%2==0&& i%3==0&& i%7==0)
        {
            cout<<i<<endl;
        }
    }
    return 0;
}

```


求和计数

```
#include <bits/stdc++.h> //1-1002-1
using namespace std;
int main()
{
    //n 代表输入的整数, sum 代表和, 初始值为 0
    int n, i, sum=0;
    // 输入 n
    cin>>n;
    for (i=1; i<=n; i++)
    {
        sum=sum+i;
    }
    // 输出和
    cout<<sum;
    return 0;
}
```

```
#include<iostream> //1-1002-2 仅作参考 javacn
using namespace std;
int sum(int n)
{
    if(n == 0) return 0;
    return sum(n - 1) + n;
}
int main()
{
    int n;
    cin >> n;
    cout << sum(n);
    return 0;
}
```

```
#include<bits/stdc++.h>>//2-1741
using namespace std;
int main()
{
    int n, i, sum=0, m=0;
    cin>>n;
    // 遍历 1 到 n 的所有整数
    for (i=1; i<=n; i++)
    {
        // 判断满足 2 的倍数但不是 3 的倍数的数
        if (i%2==0&& i%3!=0)
        {
            // 求这些数的和
            sum=sum+i;
            // 计数器 +1, 统计符合条件的数的个数
            m++;
        }
    }
    cout<<m<<endl<<sum;
    return 0;
}
```

```
#include <bits/stdc++.h>//3-1003-1
using namespace std;
int main()
{
    int n, s=0 ;
    cin >> n ;
    for ( int i = 1; i <= n ; i += 2)
        s += i ;
    cout << s << endl ;
    return 0;
}
```

```
#include <bits/stdc++.h>//3-1003-2 javacn
using namespace std;
int main()
{
    // 定义整数变量 n 为输入的数，i 用作循环，sum 求和
    int n, i, sum=0;
    // 输入 n
    cin>>n;

    for (i=1; i<=n; i+=2)
    {
        sum=sum+i;
    }
    // 输出 sum
    cout<<sum;
    return 0;
}
```

```
#include <bits/stdc++.h> //4-1004
using namespace std;
int main()
{
    // 定义整数变量 n 作为输入的数，i 用作循环，cj 记录乘积，初始值为 1
    int n, i, cj=1;
    // 输入 n
    cin>>n;

    for (i=1; i<=n; i++)
    {
        cj=cj*i;
    }
    // 输出乘积
    cout<<cj;
    return 0;
}
```

```
#include <bits/stdc++.h> //5-1014
using namespace std;
int main()
{
    int n, i;
    // 定义小数变量 s 用于求和，初始值为 0
    double s=0;
    cin>>n;
    for (i=1; i<=n; i++)
    {
        s=s+1.0/i;
    }
    // 输出和，保留 3 位小数
    cout<<fixed<<setprecision(3)<<s;
    return 0;
}
```

```

#include <bits/stdc++.h>///6-1053 解法 1: for 循环 javacn
using namespace std;
int main()
{
    int i, s=0;
    for (i=100; i>=1; i-=3)    {    s+=i; }
    cout<<s;
    return 0;
}

```

```

#include<bits/stdc++.h>/// 解法 2: while 循环 javacn
using namespace std;
int main() {
    int i=100, sum=0;
    while(i>=1)
    {
        sum=sum+i;
        i=i-3;
    }
    cout<<sum;
    return 0;
}

```

```

#include<iostream>///6-1053
using namespace std;
int sum(int n)
{
    if(n == 1) return 1;
    return sum(n - 3) + n;
}
int main()
{
    int n = 100;
    //cin >> n;
    cout << sum(n);
    return 0;
}

```

```
#include<bits/stdc++.h>//7-1054
using namespace std;
int main()
{
    int n, i; // 定义整数变量 n 作为输入的数, i 用作循环
    cin>>n;
    int sum=0; //sum 作为和, 初始值为 0
    for (i=1; i<=n; i++)
    {
        sum=sum+i*i; // 每次加上 i 的平方
    }
    // 输出 sum
    cout<<sum;
    return 0;
}
```

```
#include<bits/stdc++.h>//8-1055
using namespace std;
int main()
{
    // 定义整数变量 n 代表输入的数, x 用作计数符合条件的数字个数
    int n, x=0;
    cin>>n;
    int i; //i 用作循环
    for (i=1; i<=n; i++) //i 要满足用 3 除余 2, 用 5 除余 3, 用 7 除余 2
    {
        if (i%3==2&& i%5==3&& i%7==2)
        {
            x++;
        }
    }
    cout<<x;
    return 0;
}
```

```
#include<bits/stdc++.h>//9-1056-1 javacn
using namespace std;
int main()
{
    //i 用作循环，q, b, s, g 分别代表千位，百位，十位，个位，x 计数
    int i, x=0, q, b, s, g;
    for (i=1; i<=1000; i++)
    {
        // 对 i 进行拆位
        q=i/1000;
        b=i/100;
        s=i/10%10;
        g=i%10;

        if (q==3 || b==3 || s==3 || g==3) // 符合条件则 x+1
        {
            x++;
        }
    }
    cout<<x;
    return 0;
}
```

```
#include<bits/stdc++.h>>//9-1056-2
using namespace std;
int main()
{
    int i=1,c=0;//i 循环变量初始化, i 从 1 开始; c:count 统计数量
    int g=0,s=0,b=0;// 存放三位数拆位的结果
    // 循环条件
    while(i<1000)
    {
        // 拆出每一位
        g=i%10;
        s=i/10%10;
        b=i/100;
        // 判断每一位是否含有数字 3, 如果有的话计数器加 1
        if(g==3||s==3||b==3)
        {
            c++;
        }
        // 让循环停下来办法
        i++;
    }
    // 循环结束后输出统计的结果
    cout<<i<<endl;
    return 0;
}
```



```
#include <iostream>//10-1057-1 javacn
using namespace std;
int main()
{
    int N, i, w, q, b, s, g, x=0;
    cin>>N;
    for (i=1; i<=N; i++)
    {
        w=i/10000;
        q=i/1000%10;
        b=i/100%10;
        s=i%100/10;
        g=i%10;
        if (i%5==0)
        {
            if (w==5 || q==5 || b==5 || s==5 || g==5)
            {
                x++;
            }
        }
    }
    cout<<x;
    return 0;
}
```

```
#include <bits/stdc++.h>//10-1057-2 仅供参考 javacn
```

```
using namespace std;
```

```
bool five(int x)
```

```
{
```

```
    bool r=false;
```

```
    if(x%5==0)
```

```
    {
```

```
        while(x/10!=0 || x%10==5)
```

```
        {
```

```
            if(x%10==5)
```

```
            {
```

```
                r=true;
```

```
                break;
```

```
            }
```

```
            else
```

```
            {
```

```
                x=x/10;
```

```
            }
```

```
        }
```

```
    }
```

```
    return r;
```

```
}
```

```
int main()
```

```
{
```

```
    int N;
```

```
    cin>>N;
```

```
    int i,num=0;
```

```
    for(i=1;i<=N;i++)
```

```
    {
```

```
        if(five(i)) { num++; }
```

```
    }
```

```
    cout<<num;
```

```
    //cout<<five(5);
```

```
    return 0;
```

```
}
```

```
#include <iostream>//11-1393-1 javacn
using namespace std;
int main()
{
    int n, sum=0, i;
    cin>>n;
    for (i=1; i<=n; i++)
    {
        if (i%7!=0&& i/100!=7&& i%10!=7&& i/10%10!=7)
        {
            sum=sum+i;
        }
    }
    cout<<sum;
    return 0;
}
```

解法 2: 自定义函数

```
#include <bits/stdc++.h> //11-1393-2 推荐学习 javacn
```

```
using namespace std;
```

```
bool noseven(int x)
```

```
{
```

```
    bool r=true;
```

```
    if(x%7==0) { r=false; }
```

```
    while(x/10!=0)
```

```
    {
```

```
        if(x%10==7 || x/10==7)
```

```
        {
```

```
            r=false;
```

```
            break;
```

```
        }
```

```
        else
```

```
        {
```

```
            x=x/10;
```

```
        }
```

```
    }
```

```
    return r;
```

```
}
```

```
int main() {
```

```
    int n;
```

```
    cin>>n;
```

```
    int i, sum=0;
```

```
    for (i=1; i<=n; i++)
```

```
    {
```

```
        if(noseven(i))
```

```
        {
```

```
            sum=sum+i;
```

```
        }
```

```
    }
```

```
    cout<<sum;
```

```
    return 0;
```

```
}
```

```
#include<bits/stdc++.h>//12-1445
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int x, m, n, ci, count=0;
```

```
    cin>>x>>m>>n;
```

```
    for (int i=m; i<=n; i++)
```

```
    {
```

```
        ci=i;
```

```
        while(ci!=0)
```

```
        {
```

```
            if(ci%10==x)
```

```
            {
```

```
                count++;
```

```
                break;
```

```
            }
```

```
            ci=ci/10;
```

```
        }
```

```
    }
```

```
    cout<<count;
```

```
    return 0;
```

```
}
```

```
#include <bits/stdc++.h> //13-1449
using namespace std;
int main()
{
    //r: 求和变量, 要清零
    int i, g, s, b, q, n, c, r = 0;
    cin >> n;
    for (i = 1; i <= n; i++)
    {
        //c 是一个反复使用的计数器, 因此每次都要清零
        // 因为对于每个 i 都要求能被几个数整除
        c = 0;

        if (i % 2 == 0) { c++; }
        if (i % 3 == 0) { c++; }
        if (i % 5 == 0) { c++; }
        if (i % 7 == 0) { c++; }

        // 如果能被 2 个及 2 个以上的数整除
        if (c >= 2) { r = r + i; } // 求和
    }
    // 最后输出总和
    cout << r << endl;
    return 0;
}
```

```

#include <bits/stdc++.h>//14-1742 推荐学习 javacn
using namespace std;
int main() {
    int m, n, i, x=0;
    cin>>m>>n;
    for (i=m; i<=n; i++) // 遍历所有 m 到 n 之间的三位整数
    {
        if (i/100==i%10) // 判断百位和个位是否相等
        {
            cout<<i<<endl;
            x++; // 记录符合条件的数的个数
        }
    }
    cout<<x;
    return 0;
}

```

```

#include <bits/stdc++.h>//15-1743
using namespace std;
int main() {
    int m, n;
    cin>>m>>n;
    int i, sum=0, x=0;
    for (i=m; i<=n; i++) // 遍历 m~n 之间所有的 5 位数
    {
        if (i%2==0&& i/10000==i%10&& i/1000%10==i%100/10)
        {
            // 万位等于个位，千位等于十位，且是个偶数
            x++; // 统计符合条件的数的个数
            sum=sum+i; // 对符合条件的数求和
        }
    }
    cout<<x<<endl<<sum;
    return 0;
}

```

```
#include <bits/stdc++.h> //16-1744
using namespace std;
int main()
{
    int n, i, q, b, s, g, sum=0;
    cin>>n;

    for (i=1000; i<=n; i++) // 遍历 1000 到 n 之间的所有 4 位数
    {
        q=i/1000; // 对 i 进行拆位
        b=i/100%10;
        s=i%100/10;
        g=i%10;

        if (q%2==1&&b%2==1&&s%2==1&&g%2==1) // 判断 i 的各个位是奇数
        {
            sum=sum+i; // 对符合条件的数累加
        }
    }
    cout<<sum;
    return 0;
}
```



```
#include <bits/stdc++.h>//17-1745
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, x1, x2, x3, x4, x5, y=0;
    // 遍历 10000 到 n 之间所有的五位整数
    for (i=10000; i<=n; i++)
    {
        // 对 i 进行拆位
        x1=i/10000;
        x2=i/1000%10;
        x3=i/100%10;
        x4=i%100/10;
        x5=i%10;
        // 判断各个位是否都是偶数
        if (x1%2==0&& x2%2==0&& x3%2==0&& x4%2==0&& x5%2==0)
        {
            y++; // 计数符合条件的数的个数
        }
    }
    cout<<y;
    return 0;
}
```

```
#include <bits/stdc++.h> //18-1747
using namespace std;
int main()
{
    int n, i, sum=0, y=0, b, s, g;
    cin>>n;
    for (i=100; i<=n; i++) // 遍历 100 到 n 的所有三位数
    {
        b=i/100;
        s=i/10%10;
        g=i%10;
        if ((b<s&& s<g) || (b>s&& s>g)) // 连续递增或者连续递减
        {
            sum=sum+i;
            y++;
        }
    }
    cout<<sum<<endl<<y;
    return 0;
}
```

```
#include <iostream>//19-1750-1 推荐学习
```

```
using namespace std;
```

```
int main() {  
    int n, i, x, c=0;  
    cin>>n;  
  
    for (i=1;i<=n;i++)  
    {  
        x=i;  
        // 判断 x 是否含有 0; 如果找到则退出  
        while(x!=0)  
        {  
            if(x%10==0)  
            {  
                c++;  
                break;  
            }  
            x=x/10;  
        }  
    }  
    cout<<c<<<endl;  
    return 0;  
}
```

思路:

第一步: 循环 $1 \sim n$ 的每个数

第二步: 逐个判断每个数是否有 0

判断每个数是否有 0, 有 2 种做法, 可以拆出每个数 i 的个位、十位、百位, 不过这样的话要判断 i 是几位数, 因为如果是 1 位数直接拆十位、百位的话, 也会认为十位百位是 0。

第 2 种方法是采用短除法拆位, 推荐第 2 种做法, 因为短除法并不需要直到 i 是几位数, 不过要注意, 不能直接拆 i , 会将 i 拆到 0, 导致死循环。

以下是 2 种解法的参考程序。

参考程序一：拆出每个数的个位、十位、百位。

解法一：直接拆位，先判断是几位数，再判断是否有 0

```
#include <bits/stdc++.h>//19-1750-2 javacn
using namespace std;
int main() {
    int i, n, c = 0, g, s, b;
    cin >> n;
    for (i = 1; i <= n; i++)
    {
        // 拆位
        g = i % 10;
        s = i / 10 % 10;
        // 判断是否有 0
        // 两位数：个位为 0
        if (i >= 10 && i <= 99 && g == 0)
        {
            //cout << i << endl;
            c++;
        }
        else if (i >= 100 && i <= 999 && (g == 0 || s == 0))
        {
            //cout << i << endl;
            c++;
        }
    }

    cout << c << endl;
    return 0;
}
```

解法二：短除法

```
#include <bits/stdc++.h> //19-1750-3 推荐学习 javacn
using namespace std;
int main()
{
    int i, n, c = 0, t;
    cin >> n;
    for (i = 1; i <= n; i++)
    {
        // 拆位
        // 不能直接拆 i，短除法会将 i 拆到 0
        t = i;
        while (t != 0)
        {
            // 如果这一位为 0
            // 只要有 1 位为 0，其余的位不能看了，算作该数有 0
            // 比如 100，虽然该数有 2 个 0，但只能算作 1 个有 0 的数
            if (t % 10 == 0)
            {
                c++;
                break; // 停止 while 循环
            }
            t = t / 10;
        }
    }

    cout << c << endl;
    return 0;
}
```

```
#include <bits/stdc++.h> //20-1091
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, s1=0, s2=0; //s1 代表奇数的和, s2 代表偶数的和
    for (i=1; i<=n; i++)
    {
        if (i%2==1)
        {
            s1=s1+i;
        }
        else
        {
            s2=s2+i;
        }
    }
    cout<<s1<<" "<<s2;
    return 0;
}
```

思路：遍历循环 1-n 的数

寻找满足各个位的和不能被 2 整除也不能被 5 整除的数，计数个数

```
#include <iostream>//21-1395-1 推荐学习 javacn
```

```
using namespace std;
```

```
int main() {
```

```
    int n;
```

```
    cin>>n;
```

```
    //x 计数满足条件的数的个数，sum 为各个位的和
```

```
    int i, x=0, sum, q, b, s, g;
```

```
    // 循环遍历 1 到 n 的数
```

```
    for (i=1; i<=n; i++)
```

```
    {
```

```
        // 对 i 进行拆位
```

```
        q=i/1000;
```

```
        b=i/100%10;
```

```
        s=i%100/10;
```

```
        g=i%10;
```

```
        sum=q+b+s+g; // 求各个位的和
```

```
        // 各个位的和不能被 2 整除也不能被 5 整除
```

```
        if (sum%2!=0&&sum%5!=0)
```

```
        {
```

```
            x++;
```

```
        }
```

```
    }
```

```
    cout<<x;
```

```
    return 0;
```

```
}
```

```
#include<iostream>//21-1395-2
using namespace std;
int n, cnt = 0;
void xiaoli(int x)
{
    if(x <= n)
    {
        int a = x, sum = 0;
        while(a)
        {
            sum += a % 10;
            a /= 10;
        }
        if(sum % 2 != 0 && sum % 5 != 0)
        {
            cnt++;
        }
        xiaoli(x + 1);
    }
}

int main()
{
    cin >> n;
    xiaoli(1);
    cout << cnt;
    return 0;
}
```



```
#include<iostream>//21-1395-3
using namespace std;
int xiaoli(int x, int n, int cnt)
{
    if(x > n) return cnt;
    int a = x, sum = 0;
    while(a)
    {
        sum += a % 10;
        a /= 10;
    }

    if(sum % 2 != 0 && sum % 5 != 0)
    {
        cnt++;
    }
    xiaoli(x + 1, n, cnt);
}

int main()
{
    int n;
    cin >> n;
    cout << xiaoli(1, n, 0);

    return 0;
}
```

嵌套循环图形输出

```
#include<bits/stdc++.h>//1-1065   javacn
using namespace std;
int main()
{
    int i, j, n;
    cin>>n;
    for (i=1; i<=n; i++)          // 只确定了每行循环的次数
    {
        // 每行做的事
        // ①每行输出 n 个 *
        for (j=1; j<=n; j++)
        {
            cout<<"*";
        }
        cout<<endl;          // ②换行
    }
    return 0;
}
```

```
#include<bits/stdc++.h>//2-1066 javacn 推荐学习
```

```
using namespace std;
```

```
int main()
```

```
{  
    int n;  
    cin>>n;    // 输入 n  
    int i, j;  
    for (i=1; i<=n; i++)  
    {  
        for (j=1; j<=i; j++) // 每行循环 i 次输出 *  
        {  
            cout<<"*";  
        }  
        cout<<endl;  
    }  
    return 0;  
}
```

```
#include<iostream>//2-1066 jiangyf70 仅作参考
```

```
using namespace std;
```

```
void tx(int a, int n)
```

```
{  
    if(a <= n)  
    {  
        for (int i = 0; i < a; i++) cout << '*';  
        cout << endl;  
        tx(a + 1, n);  
    }  
}
```

```
int main()
```

```
{  
    int n;  
    cin >> n;  
    tx(1, n);  
    return 0;  
}
```

```

#include <bits/stdc++.h>//3-1782 javacn 推荐学习
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++) // 输出 n 行
    {
        for (j=i; j<=n; j++) // 每行输出 n+1-i 个 *
        {
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}

```

```

#include<iostream>//3-1782 递归 jiangyf70
using namespace std;
void tx(int a, int n)
{
    if(a >= 1)
    {
        for(int i = 0; i < a; i++) cout << '*';
        cout << endl;
        tx(a - 1, n);
    }
}
int main()
{
    int n;
    cin >> n;
    tx(n, n);
    return 0;
}

```

```

#include <bits/stdc++.h>//4-1783 javacn 推荐学习
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 输出 n 行
    {
        for (j=1; j<=i; j++)    // 每行输出 i 列
        {
            cout<<i;
        }
        cout<<endl;
    }
    return 0;
}

```

```

#include <bits/stdc++.h>//4-1783 fangweijiaoyu
using namespace std;
void print(int n)
{
    if(n==0) return;// 递归出口
    print(n-1); // 先调用下一层，保证先输出小的数字
    for (int i=1; i<=n; i++)// 再输出
        cout<<n;
    cout<<endl;
}
int main()
{
    int n;
    cin>>n;
    print(n);
    return 0;
}

```

```
#include <bits/stdc++.h>//5-1784 javacn 推荐学习
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n;
```

```
    cin>>n;
```

```
    int i, j;
```

```
    for (i=1; i<=n; i++)    // 输出 n 行
```

```
    {
```

```
        for (j=1; j<=i; j++)    // 每行输出 i 列
```

```
        {
```

```
            cout<<j;
```

```
        }
```

```
        cout<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

```
#include<iostream>//5-1784 jiangyf70
```

```
using namespace std;
```

```
void tx(int a, int n)
```

```
{
```

```
    if(a <= n)
```

```
    {
```

```
        for (int i = 1; i <= a; i++) cout << i;
```

```
        cout << endl;
```

```
        tx(a + 1, n);
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    int n;
```

```
    cin >> n;
```

```
    tx(1, n);
```

```
    return 0;
```

```
}
```

```

#include <bits/stdc++.h>//6-1785  javacn
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 输出 n 行
    {
        for (j=1; j<=n-i+1; j++)    // 每行输出 n-i+1 列
        {
            cout<<j;
        }
        cout<<endl;
    }
    return 0;
}

```

```

#include <bits/stdc++.h>//7-1786  javacn
using namespace std;
int main()
{
    int i, j, n;
    cin>>n;
    for (i=1; i<=n; i++)    // 输出 n 行
    {
        for (j=1; j<=n-i+1; j++)    // 每行输出 n-i+1 列
        {
            cout<<n-i+1;
        }
        cout<<endl;
    }
    return 0;
}

```

```
#include<bits/stdc++.h>//8-1067
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=i-1; j++) // 每行循环 i-1 次输出空格
        {
            cout<<" ";
        }

        for (j=1; j<=n; j++) // 每行循环 n 次输出 *
        {
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}
```



```

#include<bits/stdc++.h>//9-1068 javacn
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n-i; j++) // 每行输出 n-i 个空格
        {
            cout<<" ";
        }
        for (j=1; j<=2*i-1; j++) // 每行输出 2*i-1 个 *
        {
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}

```

/*

n 行

| 第几行 | 几列空格 | 几列星 |
|-------|-------|-------|
| 第 1 行 | 2 个 | 1 个星 |
| 第 2 行 | 1 个 | 3 个星 |
| 第 3 行 | 0 个 | 5 个星 |
| 第 i 行 | n-i 个 | 2*i-1 |

*/

```
#include<bits/stdc++.h>//10-1069
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n-i; j++) // 每行输出 n-i 个空格
        {
            cout<<" ";
        }

        for (j=1; j<=2*i+1; j++) // 每行输出 2*i+1 个*
        {
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}
```

```
#include<bits/stdc++.h>//11-1070
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=i-1; j++) // 每行输出 i-1 个空格
        {
            cout<<" ";
        }

        for (j=1; j<=2*(n-i)+1; j++) // 每行输出 2*(n-i)+1 个 *
        {
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}
```

```

#include<bits/stdc++.h>//12-1071
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n+1; i++)    // 循环 n+1 行
    {
        for (j=1; j<=n-i+1; j++)    // 每行输出 n-i+1 个空格
        {
            cout<<" ";
        }
        for (j=1; j<=2*i-1; j++)    // 每行输出 2*i-1 个 *
        {
            cout<<"*";
        }
        cout<<endl;
    }

    for (i=1; i<=n; i++)    // 循环 n 行
    {
        for (j=1; j<=i; j++)    // 每行输出 i 个空格
        {
            cout<<" ";
        }
        for (j=1; j<=2*(n-i)+1; j++)    // 每行输出 2*(n-i)+1 个 *
        {
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}

```

```

#include<bits/stdc++.h>//13-1072
using namespace std;
int main() {
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 循环 n 行
    {
        for (j=1; j<=n-i; j++) // 每行输出 n-i 个空格
        {
            cout<<" ";
        }
        for (j=1; j<=2*i-1; j++)    // 每行输出 2*i-1 个数，数字为 j
        {
            cout<<j;
        }
        cout<<endl;
    }
    return 0;
}

```

```

#include<bits/stdc++.h>//14-1422
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 循环 n 行
    {
        for (j=1; j<=n; j++) // 每行重复输出 n 次行号
            cout<<i;
        cout<<endl; // 每行结束换行
    }
    return 0;
}

```

```
#include<bits/stdc++.h>//15-1363
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 循环 n 行
    {
        for (j=1; j<=n; j++) // 每行循环 n 次
            cout<<j;
        cout<<endl;
    }
    return 0;
}
```

```
#include<bits/stdc++.h>//16-1491
using namespace std;
int main() {
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 循环 n 行
    {
        for (j=1; j<=i-1; j++)    // 先循环输出每行前面的空格
        {
            cout<<" ";
        }
        for (j=1; j<=n-i+1; j++) // 再循环输出每行的数字
        {
            cout<<j;
        }
        cout<<endl;
    }
    return 0;
}
```

```
#include <bits/stdc++.h>//17-1787
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 输出 n 行
    {
        for (j=1; j<=n-i; j++)    // 每行先输出 n-i 个空格
        {
            cout<<" ";
        }

        for (j=1; j<=2*i-1; j++)    // 每行再输出 2*i-1 个数
        {
            cout<<j;
        }
        cout<<endl;
    }
    return 0;
}
```

```
#include <bits/stdc++.h>//18-1788
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 输出 n 行
    {
        for (j=1; j<=i-1; j++)    // 每行先输出 i-1 个空格
        {
            cout<<" ";
        }

        for (j=1; j<=2*(n-i)+1; j++)    // 每行再输出 2*(n-i)+1 个数
        {
            cout<<n-i+1;
        }
        cout<<endl;
    }
    return 0;
}
```



```
#include<bits/stdc++.h>//19-1492
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 循环 n 行
    {
        for (j=1; j<=n; j++)    // 循环 n 列
        {
            // 当循环到列号为 1 或列号为 n 或行号为 1 或行号为 n, 就输出 *, 其他输出空格
            if (j==1 || j==n || i==1 || i==n)
            {
                cout<<"*";
            }
            else
            {
                cout<<" ";
            }
        }
        cout<<endl;
    }
    return 0;
}
```

```
#include <iostream>//20-1493
using namespace std;
int main()
{
    int i, j, n;
    cin>>n;
    for(i = 1; i <= n; i++)        // 控制输出 n 行
    {
        if(i == 1 || i == n)    // 第 1 行和第 n 行有 n 个星
        {
            for(j = 1; j <= n; j++)
            {
                cout<<"*";
            }
        }
        else
        {
            for(j = 1; j <= n- i; j++)// 其他行，第 i 行有 n-i 个空格，1 个星
            {
                cout<<" ";
            }
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}
```

```
#include <bits/stdc++.h>//21-1494
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 循环 n 行
    {
        for (j=1; j<=n-i; j++)    // 每行输出 n-i 个 *
        {
            cout<<"*";
        }

        for (j=1; j<=2*i-1; j++) // 每行输出 2*i-1 个 @
        {
            cout<<"@";
        }

        for (j=1; j<=n-i; j++)    // 每行输出 n-i 个 *
        {
            cout<<"*";
        }

        cout<<endl;    // 输出换行
    }
    return 0;
}
```

```

#include <bits/stdc++.h> //22-1073   javacn
using namespace std;
/*n 行的沙漏：由 n/2+1 行倒三角 + n/2+1 行正三角组成
中间有一个 * 是公用的 */
int main()
{
    int n;
    cin >> n;
    n = n / 2 + 1; // 得到一半的三角形的行数
    for (int i = n; i >= 1; i--)
    {
        for (int j = 1; j <= n - i; j++)
        {
            cout << " ";
        }
        for (int j = 1; j <= 2 * i - 1; j++)
        {
            cout << "*";
        }
        cout << endl;
    }
    for (int i = 2; i <= n; i++)
    {
        for (int j = 1; j <= n - i; j++)
        {
            cout << " ";
        }
        for (int j = 1; j <= 2 * i - 1; j++)
        {
            cout << "*";
        }
        cout << endl;
    }
    return 0;
}

```

行的沙漏：由 $n/2+1$ 行倒三角 + $n/2+1$ 行正三角组成
中间有一个 * 是公用的
读入 n 之后，立刻计算出有一半的三角需要的行数，再分别输出上一半和下一半的三角。

```

// 正三角
/*
n = 3
i=1 2 个空格 1 个星
i=2 1 个空格 3 个星
i=3 0 个空格 5 个星
第 i 行 n-i 个空格
2 * i - 1 个星
*/

```

```

#include<bits/stdc++.h>//23-1219
using namespace std;
int main() {
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++)    // 循环 n 行
    {
        if(i<=n/2+1)    // 循环上半部的 n/2+1 行
        {
            for (j=1; j<=i-1; j++)// 先输出空格，每行 i-1 个
            {
                cout<<" ";
            }

            for (j=1; j<=n; j++) // 输出 * 每行 n 个
            {
                cout<<"*";
            }
        }
        else    // 循环下半部分
        {
            for (j=1; j<=n-i; j++)// 先输出空格，每行 n-i 个
            {
                cout<<" ";
            }
            for (j=1; j<=n; j++)    // 输出 * 每行 n 个
            {
                cout<<"*";
            }
        }
        cout<<endl;
    }
    return 0;
}

```

```
#include <bits/stdc++.h>//24-1008 javacn 推荐学习
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n;
```

```
    cin>>n;
```

```
    int i, j;
```

```
    for (i=1; i<=n; i++)// 循环展示 n 行
```

```
    {
```

```
        for (j=1; j<=n-i; j++) { cout<<" "; }// 循环数字前空格的展示
```

```
        for (j=1; j<=2*i-1; j++) { cout<<i; }// 数字的循环展示
```

```
        cout<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

```
#include<iostream>//24-1008 jiangyf70
```

```
using namespace std;
```

```
void tx(int a, int n)
```

```
{
```

```
    if(a <= n)
```

```
    {
```

```
        for(int i = a; i < n; i++ ) cout <<' ';
```

```
        for(int i = 1; i <= 2 * a - 1; i++) cout << a;
```

```
        cout << endl;
```

```
        tx(a + 1, n);
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    int n;
```

```
    cin >> n;
```

```
    tx(1, n);
```

```
    return 0;
```

```
}
```

```

#include <bits/stdc++.h>//25-1006
using namespace std;
int main()
{
    int n;    //n 代表输入的行数
    cin>>n;  // 输入 n
    int i, j, k;
    for (i=1; i<=n; i++)    // 循环行数
    {
        for (j=1; j<=3; j++)    // 循环 3 个三角形
        {
            for (k=1; k<=n-i; k++)    // 循环展示空格
            {
                cout<<" ";
            }

            for (k=1; k<=2*i-1; k++)    // 循环展示 *
            {
                cout<<"*";
            }

            for (k=1; k<=n-i; k++)    // 循环展示空格
            {
                cout<<" ";
            }
        }

        cout<<endl;    // 第 i 行结束，换行
    }
    return 0;
}

```

```

#include <bits/stdc++.h> //26-1011-1 空心六边形 javacn
using namespace std;
int main() {
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++) // 循环展示 n 行 , 边长即为行数
    {
        for (j=1; j<=n-i; j++) { cout<<" "; } // 循环前面的空格
        if (i==1) // 除第一行外只要展示头尾的星号
        { for (j=1; j<=n; j++) cout<<"*"; }
        else
        {
            cout<<"*";
            for (j=1; j<=n+2*(i-2); j++) { cout<<" "; }
            cout<<"*";
        }
        cout<<endl;
    }

    for (i=n-1; i>=1; i--) // 倒序重复上面的循环 减去一行
    {
        for (j=1; j<=n-i; j++) { cout<<" "; }
        if (i==1)
        { for (j=1; j<=n; j++) cout<<"*"; }
        else
        {
            cout<<"*";
            for (j=1; j<=n+2*(i-2); j++) { cout<<" "; }
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}

```



```

#include<iostream>//26-1011-2 空心六边形 leirui
using namespace std;
int main()
{
    int n;
    cin>>n;
    for (int i=1;i<=n;i++) // 前 n 行
    {
        for (int j=1;j<=n-i;j++)// 空格
            cout<<" ";
        for (int j=1;j<=(n+(i-1)*2);j++)// 图形
        {
            if (i==1||j==1||j==n+(i-1)*2)// 第一行, 第一列, 最后一列
                cout<<"*";
            else
                cout<<" ";
        }
        cout<<endl;
    }

    for (int i=n-1;i>=1;i--) // 后 n-1 行
    {
        for (int j=1;j<=n-i;j++)// 空格
            cout<<" ";
        for (int j=1;j<=(n+(i-1)*2);j++)// 图形
        {
            if (i==1||j==1||j==n+(i-1)*2)// 第一行, 第一列, 最后一列
                cout<<"*";
            else
                cout<<" ";
        }
        cout<<endl;
    }
    return 0;
}

```

| 行 | 空格 | 图形 |
|-----|----|----|
| i=1 | 5 | 6 |
| i=2 | 4 | 8 |
| i=3 | 3 | 10 |

空格 = n-i
 图形 = n+(i-1)*2
 */

```
#include <bits/stdc++.h> //27-1225   javacn
```

```
using namespace std;
```

```
/*
```

```
1. 先打实心
```

```
2. 将实心中间的 * 替换为空格
```

```
(1) 第 1 行和最后 1 行都属于边界上的 *
```

```
(2) 其他行，只有第 1 个和最后一个是边界上的 *，其余的都在内部
```

```
*/
```

```
int main()
```

```
{
```

```
    int n;
```

```
    cin>>n;
```

```
    for(int i = 1; i <= n; i++)
```

```
    {
```

```
        for(int j = 1; j <= n - i; j++)      {    cout<<" "; }
```

```
        for(int j = 1; j <= 2 * i - 1; j++)
```

```
        {
```

```
            if(i==1 || i == n) // 如果是第 1 行或者最后一行，都直接输出
```

```
            {
```

```
                cout<<"*";
```

```
            }
```

```
            else // 如果是本行第 1 个或者最后一个，输出 *，其余的替换为空格
```

```
            {
```

```
                if(j == 1 || j == 2 * i - 1) {    cout<<"*"; }
```

```
                else {    cout<<" "; }
```

```
            }
```

```
        }
```

```
        cout<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

`#include<bits/stdc++.h> //28-1230-1` 推荐学习

```
using namespace std;
int n;
int main()
{
    cin>>n;
    for (int i=1;i<=n/2+1;i++)
    {
        for (int j=1;j<=n/2+2-i;j++)    cout<<"*";
        cout<<endl;
    }
    for (int i=n/2;i>=1;i--)
    {
        for (int j=1;j<=n/2+2-i;j++)    cout<<"*";
        cout<<endl;
    }
    return 0;
}
```

`#include<bits/stdc++.h> //28-1230-2`

```
using namespace std;
int n;
int main() {
    cin>>n;
    for (int i=1, j=n/2+1; i<=n/2+1; i++, j--)
    {
        for (int k=1; k<=j; k++)    cout<<"*";
        cout<<endl;
    }
    for (int i=2, j=2; i<=n/2+1; i++, j++)
    {
        for (int k=1; k<=j; k++)    cout<<"*";
        cout<<endl;
    }
    return 0;
}
```

```
#include <bits/stdc++.h> //29-1239
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n;
```

```
    cin>>n;
```

```
    n = n / 2 + 1; // 得到一半的三角的行数
```

```
    for (int i = n; i >= 1; i--)
```

```
    {
```

```
        for (int j = 1; j <= n - i; j++) { cout<<" "; }
```

```
        for (int j = 1; j <= 2 * i - 1; j ++)
```

```
        {
```

```
            if (j == 1 || j == 2 * i - 1) { cout<<"X"; }
```

```
            else { cout<<" "; }
```

```
        }
```

```
        cout<<endl;
```

```
    }
```

```
    for (int i = 2; i <= n; i++)
```

```
    {
```

```
        for (int j = 1; j <= n - i; j++) { cout<<" "; }
```

```
        for (int j = 1; j <= 2 * i - 1; j ++)
```

```
        {
```

```
            if (j == 1 || j == 2 * i - 1) { cout<<"X"; }
```

```
            else { cout<<" "; }
```

```
        }
```

```
        cout<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

```
/*
```

```
n 行的沙漏： 由 n/2+1 行倒三角 +  
n/2+1 行正三角组成
```

```
中间有一个 * 是公用的
```

```
*/
```

```
先打印实心的沙漏 X
```

```
判断每一行，只有第 1 个和最后一个字  
符是 X，其余的替换为空格
```

```

#include<bits/stdc++.h>//30-1247
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n/2+1; i++) // 上半部分循环 n/2+1 行
    {
        for (j=1; j<=i; j++) // 先遍历输出 *
        {
            cout<<"*";
        }
        for (j=1; j<=n-i*2+1; j++) // 遍历输出空格
        {
            cout<<" ";
        }
        for (j=1; j<=i; j++) // 遍历输出 *
        {
            cout<<"*";
        }
        cout<<endl;
    }

    for (i=n/2; i>=1; i--) // 上半部分的逆序
    {
        for (j=1; j<=i; j++) { cout<<"*"; }
        for (j=1; j<=n-i*2+1; j++) { cout<<" "; }
        for (j=1; j<=i; j++) { cout<<"*"; }
        cout<<endl;
    }
    return 0;
}

```

```

#include<bits/stdc++.h>//31-1353   javacn
using namespace std;
int main() {
    int n;
    char z;
    cin>>n>>z;
    int i, j;
    if(z=='x') // 沿 x 轴对称
    {
        for (i=1; i<=n; i++)
        {
            for (j = 1; j <= n - i; j++)    {cout<<" ";} //n-i 个空格
            for (j=1; j<=2*i-1; j++)        {cout<<"*";} //2*i-1 个星
            cout<<endl;
        }
        for (i=n; i>=1; i--) // 将上面的三角倒过来
        {
            for (j = 1; j <= n - i; j++)    {cout<<" ";} //n-i 个空格
            for (j=1; j<=2*i-1; j++)        {cout<<"*";} //2*i-1 个星
            cout<<endl;
        }
    }
    else // 沿 y 轴对称
    {
        for (i=1; i<=n; i++)
        {
            for (j = 1; j <= n - i; j++)    {cout<<" ";} //n-i 个空格
            for (j=1; j<=2*i-1; j++)        {cout<<"*";} //2*i-1 个星
            for (j = 1; j <= 2*(n - i); j++) {cout<<" ";} //n-i 个空格
            for (j=1; j<=2*i-1; j++)        {cout<<"*";} //2*i-1 个星
            cout<<endl;
        }
    }
    return 0;
}

```

嵌套循环应用

```
#include <bits/stdc++.h>//1-1246
using namespace std;
int main() {
    int n;
    cin>>n;
    int i, j;
    for (i=1; i<=n; i++) // 循环 n 行
    {
        for (j=1; j<=i; j++) // 遍历 输出式子 第几行 * j
        {
            cout<<i<<"*"<<j<<"="<<i*j<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

```
#include <bits/stdc++.h>//2-1019-1 解法 1: 嵌套循环 推荐学习 javacn
using namespace std;
int main() {
    int n, i, j, s=0, c;
    cin>>n;
    for (i=1; i<=n; i++)
    {
        c=1;
        for (j=1; j<=i; j++)
        {
            c = c*j;
        }
        s+=c;
    }
    cout<<s;
    return 0;
}
```

```

#include<bits/stdc++.h> //2-1019-2
using namespace std;
int s, n, i, t=1; // 计算过 (i-1)! 后, 只要将结果再乘 i 即可
int main()
{
    cin>>n;
    for (i=1; i<=n; i++)
    {
        t=t*i;
        s=s+t;
    }
    cout<<s<<endl;
    return 0;
}

```

```

#include <bits/stdc++.h> //2-1019-3 解法 2: 自定义函数 javacn
using namespace std;
int jc(int x) // 阶乘
{
    int r=1;
    int i;
    for (i=1; i<=x; i++) {        r=r*i;    }
    return r;
}
int main() {
    int n;
    cin>>n;
    int i, sum=0;
    for (i=1; i<=n; i++)
    {
        sum=sum+jc(i);
    }
    cout<<sum;
    return 0;
}

```



```
#include <bits/stdc++.h> //3-1495-1 javacn
```

```
using namespace std;
```

/* 解法一：嵌套循环求出每个数因子的数量，注意因子个数计数器需要在 i 循环进入之后才能清零。思路：第一步：循环 $1 \sim n$ 的每个数 第二步：嵌套循环求出每个数有几个因子 */

```
int main() {  
    int i, j, n, c;  
    cin >> n;  
    for (i = 1; i <= n; i++)  
    {  
        c = 0; // 计数器清零，求每个数 i 有几个因子，都要从 0 开始求  
        // 循环 i 可能的因子范围  
        // 写法一：循环  $2 \sim i-1$ ，逐个判断是否是因子 建议采用第一种写法，简洁易懂  
        for (j = 2; j <= i - 1; j++)  
        {  
            if (i % j == 0) { c++; }  
        }  
        for (j = 2; j <= sqrt(i); j++) // 写法二：循环  $2 \sim \sqrt{i}$ ，求解 i 的因子  
        {  
            if (i % j == 0)  
            {  
                // 两个因子相等  
                if (j == i / j)  
                {  
                    r = r + 1; // 只能算 1 个  
                }  
                else  
                {  
                    r = r + 2; // 两个因子不等，算 2 个  
                }  
            }  
        }  
        cout << c << endl;  
    }  
    return 0;  
}
```

```
#include <bits/stdc++.h> //3-1495-2 javacn
```

```
using namespace std;
```

/* 解法二：定义函数求因子数量，在 main 中调用函数求每个数的因子数量

思路：第一步：定义函数，求一个整数 n 有几个因子

第二步：在 main 中循环 1~n 求出每个数有几个因子 */

```
int fun(int n) {
    int r = 0;
    int i;
    /* 不成对求，循环到 n-1 */ 建议采用第一种写法，简洁易懂
    for(i = 2; i <= n - 1; i++)
    {
        if(n % i == 0)
        {
            r++;
        }
    }
    /*
    for(i = 2; i <= sqrt(n); i++) // 成对求，循环到 sqrt(n)
    {
        if(n % i == 0)
        {
            if(i != n / i) r = r + 2;
            else r = r + 1; // 判断 2 个因子是否相等，不等算 2 个，相等算 1 个
        }
    }
    return r;
}

int main() {
    int i, n;
    cin >> n;
    for(i = 1; i <= n; i++)
    {
        cout << fun(i) << endl;
    }
    return 0;
}
```

思路：循环遍历从 1 到 n 的整数

计算 i 的阶乘 $i! = 1 * 2 * 3 * 4 * \dots * i$ 用 for 循环遍历 1 到 i 相乘
当 i 为偶数从 sum 中减去, i 为奇数加入 sum

```
#include <bits/stdc++.h>//4-1518 推荐学习 javacn
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j, sum=0;
    // 循环遍历 1 到 n 的数
    for (i=1; i<=n; i++)
    {
        int cj=1;
        // 计算 i 的阶乘
        for (j=1; j<=i; j++)
        {
            cj=cj*j;
        }
        //i 为偶数从 sum 中减去, i 为奇数加入 sum
        if (i%2==0)
        {
            sum=sum-cj;
        }
        else
        {
            sum=sum+cj;
        }
    }
    cout<<sum;
}
```

```

#include <bits/stdc++.h>//5-1519
using namespace std;
int main() {
    int n, i, j;
    cin>>n;
    for (i=1; i<=n; i++)// 遍历从 1 到 n
    {
        cout<<i<<" ";
        for (j=1; j<=i; j++)// 遍历从 1 到 i, 找出 i 的所有因子并输出
        {
            if (i%j==0)
                cout<<j<<" ";
        }
        cout<<endl;
    }
    return 0;
}

```

```

#include<bits/stdc++.h>//6-1086
using namespace std;
int main()
{
    int n;
    cin>>n;
    int i, j, sum, num=0;//num 代表姐妹数对的个数
    for (i=1; i<n; i++)
    {
        for (j=i+1; j<=n; j++)
        {
            sum=i+j;//sum 为两个不同整数的和
            if (sum%3==0 || sum%7==0) {num++;} //sum 被 3 除尽或被 7 除尽姐妹数对
        }
    }
    cout<<num;
    return 0;
}

```

while 循环

```
#include <bits/stdc++.h> //1-1244-1 推荐学习 javacn
using namespace std;
int main()
{
    int n;
    cin >> n;
    int i=0; //i 作为计数器, 计算能被 2 整除的数的个数
    while (n%2==0)
    {
        n=n/2;
        i++;
    }
    cout << i;
    return 0;
}
```

```
#include <iostream> //1-1244-2 递归
using namespace std;
int zc(int n)
{
    if (n % 2 == 0)
        return 1 + zc(n/2);
    return 0;
}
int main()
{
    int n;
    cin >> n;
    cout << zc(n);
    return 0;
}
```

```

#include<bits/stdc++.h>//2-1062
using namespace std;
int main() {
    double h = 100;    // 初始高度，注意使用小数类型
    int c = 0;        // 落地次数
    while (h>0.5)
    {
        c++;
        h = h/2;
    }
    cout<<c;
    return 0;
}

```

```

#include <bits/stdc++.h>//3-1254
using namespace std;
int main()
{
    int i = 95859; // 求下一个对称数
    while (1) // 一直循环，直到 break 程序主动跳出
    {
        i++;
        int t = i, s = 0; // 求 i 倒过来的数，判断对称，用短除法
        while (t != 0)
        {
            s = s * 10 + t % 10;
            t = t / 10;
        }
        if (s == i)
        { // 如果是对称的，找到了
            cout<<(i-95859)/2<<endl<<i;
            break;
        }
    }
    return 0;
}

```

```
#include<iostream>//4-1261-1 仅供参考
```

```
using namespace std;
```

```
int x;
```

```
int hxdb(int n)
```

```
{
```

```
    if(n % 5 == 1 && n % 6 == 5 && n % 7 == 4 && n % 11 == 10)
```

```
        return n;
```

```
    return hxdb(n + 1);
```

```
}
```

```
int main()
```

```
{
```

```
    cout << hxdb(1);
```

```
    return 0;
```

```
}
```

思路：利用 while 无限循环遍历寻找满足的值，一旦找到就输出值并跳出循环

```
#include<bits/stdc++.h>//4-1261-2 推荐学习 javacn
```

```
using namespace std;
```

```
int main() {
```

```
    int i=1;
```

```
    // 无限循环
```

```
    while(true)
```

```
    {
```

```
        if(i%5==1&&i%6==5&&i%7==4&&i%11==10)
```

```
        {
```

```
            cout<<i<<endl;
```

```
            // 要求最少有多少人，所以一找到满足条件的就输出并跳出循环
```

```
            break;
```

```
        }
```

```
    else
```

```
    {
```

```
        i++;
```

```
    }
```

```
}
```

```
return 0;
```

```
}
```

```

#include<bits/stdc++.h>//5-1263 穷举法 javacn
using namespace std;
int main()
{
    int n=0;
    while ((n%8==1&&n/8%8==1&&n/8/8%8==7&&n%17==4&&n/17%17==15&&
        n/8/8/8*2==n/17/17)==false)
    {
        n=n+1;
    }
    cout<<n;
    return 0;
}

```

```

#include<iostream>//6-1265-1 仅供参考

```

```

using namespace std;
int ayst(int n)
{
    //if(n % 2 == 1 && n % 3 == 2 && n % 5 == 4 && n % 6 == 5 && n % 7 == 0)
return n; // 两种方法都可以
    if((n + 1) % 30 == 0 && n % 7 == 0)
        return n;
    // 观察可知，余数加 1 刚好可以被 2、3、5、6 整除，他们的最小公倍数是 30；

    ayst(n + 1);
}
int main()
{
    cout << ayst(1);
    return 0;
}

```



```
#include <iostream>//6-1265-2 推荐学习
```

```
using namespace std;
```

```
int main()
```

```
{  
    int i=1;  
    while (!(i%2==1 && i%3==2 && i%5==4 && i%6==5 && i%7==0))  
    {  
        i++;  
    }  
    cout<<i<<endl;  
    return 0;  
}
```

```
#include <bits/stdc++.h>//6-1265-3 推荐学习    javacn
```

```
using namespace std;
```

```
int main()
```

```
{  
    int i=1;  
    while (1) // 从一开始穷举，直到找到复合条件的数，跳出循环  
    {  
        if (i%2==1&& i%3==2&& i%5==4&& i%6==5&& i%7==0)  
        {  
            cout<<i<<endl;  
            break;  
        }  
        i++;  
    }  
    return 0;  
}
```

```
#include<iostream>//7-1078-1
```

```
using namespace std;
```

```
int x;
```

```
int sum(int n, double s)
```

```
{
```

```
    s += 1.0 / n;
```

```
    if(s >= x)
```

```
        return n;
```

```
    return sum(n + 1, s);
```

```
}
```

```
int main()
```

```
{
```

```
    cin >> x;
```

```
    cout << sum(1, 0);
```

```
    return 0;
```

```
}
```

```
#include <bits/stdc++.h>//7-1078-2 推荐学习 javacn
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    double s = 0;
```

```
    int i, x;
```

```
    cin>>x;
```

```
// 分母是 1 2 3... 的循环
```

```
    i = 1;
```

```
    while(s <= x)
```

```
    {
```

```
        s = s + 1.0 / i;
```

```
        i++;
```

```
    }
```

```
// 由于每次计算完 s, 都执行 i++ // 因此当最后一次求和结束, i 会多加一次
```

```
    cout<<i-1;
```

```
    return 0;
```

```
}
```

```
#include<iostream>//8-1241-1 递归 仅供参考
```

```
using namespace std;
int jgcx(int x, int k)
{
    if(x == 1) return k;
    if(x % 2 == 0) return jgcx(x / 2, k + 1);
    return jgcx(x * 3 + 1, k + 1);
}
int main()
{
    int n;
    cin >> n;
    cout << jgcx(n, 0);
    return 0;
}
```

```
#include<iostream>//8-1241-2 推荐学习 javacn
```

```
using namespace std;
int main() { // 解法 1: 利用 while 循环当 n 不等于 1 时就继续循环处理
    int n, c=0; //c: 计数器
    cin>>n;
    while(n!=1)
    {
        if(n%2!=0)
        {
            n=n*3+1;
        }
        else if(n%2==0)
        {
            n=n/2;
        }
        c++;
    }
    cout<<c<<<endl;
    return 0;
}
```

解法 2: 递归法调用

```
#include <bits/stdc++.h> //8-1241-3 javacn
using namespace std;
int n;

// 递归计算, 直到 k 为 1
/*
    k 要经过多少次计算得到 1
    如果是偶数:  $1 + k/2$  要经过多少次
    如果是奇数:  $1 + (k*3+1)$  要经过多少次
*/
int fun(int k)
{
    int r = 0;
    if(k != 1)
    {
        if(k % 2 == 0) // 如果 k 是偶数, 将 k/2 交还给函数
        {
            r = 1 + fun(k / 2);
        }
        else
        {
            r = 1 + fun(k * 3 + 1);
        }
    }
    return r;
}

int main()
{
    cin >> n;
    int x = fun(n);
    cout << x;
    return 0;
}
```

```
#include<bits/stdc++.h>//9-1460-1 javacn
using namespace std;
int main()
{
    //m 为每次游的距离 //s 为游的总距离 //c 为换气的次数 //x 为游的目标距离
    double x, s = 0, m = 2, c = 0;
    cin>>x;
    while(s < x) // 当游的总距离小于目标距离时继续循环
    {
        s = s + m;
        m = m * 0.98;
        c++;
    }
    // 输出换气次数
    cout<<c;
    return 0;
}
```

短除法

```
#include<bits/stdc++.h>//1-1389
using namespace std;
int main()
{
    int x, n, i=0, sum=0;
    cin>>n;
    while(n>0)
    {
        i++; // 计数一共有几位数
        x=n%10;
        if(x%2==0) // 判断每位数是否是偶数，是就加到总和中
        {
            sum=sum+x;
        }
        n=n/10;
    }
    cout<<i<<" "<<sum;
    return 0;
}
```

思路：第一步：循环 $1 \sim n$ 的每个数，第二步：逐个判断每个数是否有 0

判断每个数是否有 0，有 2 种做法，可以拆出每个数 i 的个位、十位、百位，不过这样的话要判断 i 是几位数，因为如果是 1 位数直接拆十位、百位的话，也会认为十位百位是 0。

第 2 种方法是采用短除法拆位，推荐第 2 种做法，因为短除法并不需要直到 i 是几位数，不过要注意，不能直接拆 i ，会将 i 拆到 0，导致死循环。

以下是 2 种解法的参考程序。

参考程序一：拆出每个数的个位、十位、百位。

解法一：直接拆位，先判断是几位数，再判断是否有 0

`#include <bits/stdc++.h> //2-1750-1` 解法二：短除法 推荐学习 `javacn`

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int i, n, c = 0, t;
```

```
    cin >> n;
```

```
    for (i = 1; i <= n; i++)
```

```
    {
```

```
        t = i; // 拆位 // 不能直接拆 i，短除法会将 i 拆到 0
```

```
        while (t != 0)
```

```
        {
```

```
            // 如果这一位为 0
```

```
            // 只要有 1 位为 0，其余的位不能看了，算作该数有 0
```

```
            // 比如 100，虽然该数有 2 个 0，但只能算作 1 个有 0 的数
```

```
            if (t % 10 == 0)
```

```
            {
```

```
                c++;
```

```
                break; // 停止 while 循环
```

```
            }
```

```
            t = t / 10;
```

```
        }
```

```
    }
```

```
    cout << c << endl;
```

```
    return 0;
```

```
}
```

```
#include <bits/stdc++.h>//2-1750-2 仅供参考 javacn
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int i, n, c = 0, g, s, b;
```

```
    cin >> n;
```

```
    for (i = 1; i <= n; i++)
```

```
    {
```

```
        // 拆位
```

```
        g = i % 10;
```

```
        s = i / 10 % 10;
```

```
        // 判断是否有 0
```

```
        // 两位数：个位为 0
```

```
        if (i >= 10 && i <= 99 && g == 0)
```

```
        {
```

```
            // cout << i << endl;
```

```
            c++;
```

```
        }
```

```
        else if (i >= 100 && i <= 999 && (g == 0 || s == 0))
```

```
        {
```

```
            // cout << i << endl;
```

```
            c++;
```

```
        }
```

```
    }
```

```
    cout << c << endl;
```

```
    return 0;
```

```
}
```



```
#include <iostream>//3-1962-1 推荐学习 javacn
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, s = 0, c = 0;
```

```
    cin>>n;
```

```
    // 求 n 倒过来的数，求 n 的位数
```

```
    while(n != 0)
```

```
    {
```

```
        s = s * 10 + n % 10;
```

```
        n /= 10;
```

```
        c++;
```

```
    }
```

```
    cout<<c<<endl;
```

```
    // 求 s 倒过来的数
```

```
    int t = s;
```

```
    while(t != 0)
```

```
    {
```

```
        cout<<t % 10<<" ";
```

```
        t /= 10;
```

```
    }
```

```
    cout<<endl;
```

```
    cout<<s<<endl;
```

```
    return 0;
```

```
}
```

解法 2: 字符串

```
#include<bits/stdc++.h> //3-1962-2 javacn
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    string s;
```

```
    cin>>s;
```

```
    cout<<s.size()<<endl;
```

```
    int i;
```

```
    // 逐一输出每一位
```

```
    for (i=0; i<s.size(); i++)
```

```
    {
```

```
        cout<<s[i]<<" ";
```

```
    }
```

```
    cout<<endl;
```

```
    for (i=s.size()-1; i>=0; i--)
```

```
    {
```

```
        cout<<s[i];
```

```
    }
```

```
    return 0;
```

```
}
```

```

#include <bits/stdc++.h>//4-1121
using namespace std;
int main()
{
    int n, t=0;//t 记录倒序后的值
    cin>>n;
    while(n!=0)// 当 n 不为 0 时继续循环
    {
        t = n%10+t*10;//n 尾数变成 t 的头部 //n 去掉尾数
        n = n/10;
    }
    cout<<t;
    return 0;
}

```

```

#include <bits/stdc++.h>//5-1469
using namespace std;
int main()
{
    int i, n, x, t, c = 0;
    cin>>n>>x;
    for (i = 1; i <= n; i++) // 遍历 1 到 n 之间的整数
    {
        t = i;
        while(t != 0)// 对 t 循环拆位，判断等于 x 的数位，计算总次数
        {
            if(t % 10 == x)
            {
                c++;
            }
            t = t / 10;
        }
    }
    cout<<c<<endl;
    return 0;
}

```

```
#include <bits/stdc++.h>//6-1511
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, s=0, c=0;
```

```
    cin>>n;
```

```
    for (int i=1; i<=n; i++)
```

```
    {
```

```
        s=0;
```

```
        int t = i;
```

```
        while (t!=0)
```

```
        {
```

```
            s = s + t%10;
```

```
            t = t/10;
```

```
        }
```

```
        if (s==13)
```

```
        {
```

```
            c++;
```

```
        }
```

```
    }
```

```
    cout<<c;
```

```
    return 0;
```

```
}
```

思路一：遍历 1 到 n 的数，逐个调用函数判断回文

```
#include <bits/stdc++.h>//7-1149-1 推荐学习 javacn
```

```
using namespace std;
```

```
// 用于反读数字，返回倒序后的 x
```

```
int fun(int x)
```

```
{
```

```
    int r=0;
```

```
    while(x!=0)
```

```
    {
```

```
        r = x%10+r*10;
```

```
        x = x/10;
```

```
    }
```

```
    return r;
```

```
}
```

```
int main()
```

```
{
```

```
    int n, i, num=0;//num 计数回文数的个数
```

```
    cin>>n;
```

```
    for (i=1; i<=n; i++)
```

```
    {
```

```
        // 如果正读和反读一样就是回文数
```

```
        if(i==fun(i))
```

```
        {
```

```
            num++;
```

```
        }
```

```
    }
```

```
    cout<<num;
```

```
    return 0;
```

```
}
```

思路二：嵌套循环，循环每个数，逐个求每个数倒过来的数，判断是否是回文。

```
#include <bits/stdc++.h>//7-1149-2 推荐学习 javacn
```

```
using namespace std;
```

```
int n, c = 0, s; //s 代表倒过来的数
```

```
int t; // 用来过渡 i 的值，不能直接拆 i
```

```
//1~n 中的回文数的个数
```

```
int main()
```

```
{
```

```
    cin>>n;
```

```
    for (int i = 1; i <= n; i++)
```

```
    {
```

```
        t = i;
```

```
        // 反复使用的求和变量，每次清零
```

```
        s = 0;
```

```
        // 求 i 倒过来的数
```

```
        while (t != 0)
```

```
        {
```

```
            s = s * 10 + t % 10;
```

```
            t = t / 10;
```

```
        }
```

```
        // 如果是回文
```

```
        if (i == s)
```

```
        {
```

```
            //cout<<i<<endl;
```

```
            c++;
```

```
        }
```

```
    }
```

```
    cout<<c;
```

```
    return 0;
```

```
}
```

```
#include<bits/stdc++.h>//7-1149-3  字符数组解法
```

```
using namespace std;
```

```
char a[10010];
```

```
bool pd(int i)
```

```
{  
    int l=0;  
    while(i!=0)  
    {  
        a[l]=char(i%10+'0');  
        i=i/10;  
        l++;  
    }  
    string st="",ts="";  
    bool flag=false;  
    for(int j=0;j<l;j++)  
    {  
        st=st+a[j];  
        ts=a[j]+ts;  
    }  
    if(st==ts) { flag=true; }  
    return flag;  
}
```

```
int main() {  
    int n, sum=0;  
    cin>>n;  
    for(int i=1;i<=n;i++)  
    {  
        if(pd(i)==true)  
        {  
            sum++;  
            //cout<<i<<endl;  
        }  
    }  
    cout<<sum;  
    return 0;  
}
```

```
#include<bits/stdc++.h>//8-1846
using namespace std;
int main()
{
    int x;
    cin>>x;
    // 当这个数各个位的乘积不是一位数时，对它的乘积继续求解
    while(x>=10)
    {
        int t=1;
        // 计算整数非0位的乘积
        while(x!=0)
        {
            // 判断尾数不等于0就累乘
            if(x%10!=0)
            {
                t = t*(x%10);
            }
            // 去掉尾数
            x = x/10;
        }
        x = t;
    }
    cout<<x;
    return 0;
}
```


循环综合应用

求 n 的质因子：使用变量 i 从 2 开始找 n 的因子，如果 i 是 n 的因子，输出 i，且 $n=n/i$
如果 i 不是 n 的因子，看 $i+1$ ，直到 n 为 1

`#include<bits/stdc++.h> //1-1234 这道题很重要，认真学习`

```
using namespace std;
```

```
int main( )
```

```
{
```

```
    int n, i=2, c=0; // 设置因子的初始值为 2
```

```
    cin>>n;
```

```
    while(n!=1)
```

```
    {
```

```
        if(n%i==0) // 如果 n 可以被 i 整除，说明 i 是此时 n 的因子
```

```
        {
```

```
            cout<<i<<" ";
```

```
            n=n/i;
```

```
            c++;
```

```
        }
```

```
        else
```

```
        {
```

```
            i++; // i 无法整除此时的 i 则 +1
```

```
        }
```

```
    }
```

```
    cout<<endl<<c;
```

```
    return 0;
```

```
}
```

```
#include <bits/stdc++.h>//2-1446
using namespace std;
int main()
{
    int x,n;
    cin>>x>>n;
    double r = x;
    // 循环 n 次
    for (int i = 1;i <= n;i++)
    {
        r = r * 1.001;
    }
    cout<<fixed<<setprecision(4)<<r<<endl;
    return 0;
}
```

```
#include <bits/stdc++.h>//3-1448
using namespace std;
int main()
{
    int n, s=0;//s 用作记录总共参加测试的次数
    cin>>n;

    for (int i=1; i<=n; i++)// 遍历 1-n 号同学
    {
        int num=0;//num 用作记录 i 号同学要参加测试的次数
        int t=i;
        while (t>0)// 循环记录出 i 中有几个 1
        {
            if (t%10==1)
            {
                num++;
            }
            t=t/10;
        }
        // 添加到总次数中
        s+=num;
    }

    cout<<s;
    return 0;
}
```

- 思路 :1. 遍历 10000 到 30000 之间的五位数
2. 用拆位求出对应的 sub1, sub2, sub3
3. 需要计数满足条件的数的个数, 没有就输出 "No"

```
#include<bits/stdc++.h> //4-1457
using namespace std;
int main()
{
    //num 计数满足条件的数的个数
    int k, sub1, sub2, sub3, num=0;
    cin>>k;
    // 遍历 10000 到 30000 之间的五位数
    for (int i=10000; i<=30000; i++)
    {
        sub1=i/100;
        sub2=i/10%1000;
        sub3=i%1000;
        if (sub1%k==0&&sub2%k==0&&sub3%k==0)
        {
            cout<<i<<endl;
            num++;
        }
    }
    // 如果无解, 则输出 "No"
    if (num==0)
    {
        cout<<"No";
    }

    return 0;
}
```

思路：遍历 a 值范围从小到大，因为不能有重复的，所以范围为从 0 到 $n/2$

```
#include<bits/stdc++.h>//5-1515
using namespace std;
int main()
{
    int n;
    cin>>n;
    // 遍历 a 值范围从小到大
    for (int i=0;i<=n/2;i++)
    {
        // 输出式子
        cout<<n<<"="<<i<<"+"<<n-i<<endl;
    }
    return 0;
}
```

```
#include<bits/stdc++.h>//6-1517
using namespace std;
int main()
{
    int n;
    cin >> n;
    // 因子成对出现，遍历范围 1 到 n 的平方根
    for (int i = 1; i <= sqrt(n); i++)
    {
        // 判断 n 能否被 i 整除，能整除则 i 就是因子
        if (n%i == 0)
        {
            cout << n << "=" << i << "*" << n/i << endl;
        }
    }
    return 0;
}
```

思路：遍历从 1 到 n 的整数，如果 i 是奇数用加法，i 是偶数用减法

注意：结果会是小数，所以接收和的值要定义成小数变量

```
#include<bits/stdc++.h> //7-1521
```

```
using namespace std;
```

```
int main() {  
    int n;  
    cin>>n;  
    double s =0; // 定义小数变量代表 n 项的和  
    for (int i=1; i<=n; i++) // 循环遍历 1 到 n  
    {  
        if (i%2==0) // 如果 i 是奇数用加法, i 是偶数用减法  
        {  
            s-=1.0/i;  
        }  
        else  
        {  
            s+=1.0/i;  
        }  
    }  
    cout<<fixed<<setprecision(4)<<s; // 结果保留 4 位小数  
    return 0;  
}
```

思路：循环遍历 i，范围为 1 到 n，每次加上 i 的平方

```
#include<bits/stdc++.h> //8-1553
```

```
using namespace std;
```

```
int main() {  
    int n, s=0; //s 代表 n 项的和  
    cin>>n;  
    for (int i=1; i<=n; i++) // 遍历 1 到 n 的整数  
    {  
        s+=i*i; // 每次加上 i 的平方  
    }  
    cout<<s; // 输出和  
    return 0;  
}
```

要判断 $i*i$ 的右侧是否有 i

假设 i 是 x 位数，要取得 i^2 的右侧的 x 位数： $i^2 \% 10^x$
求：

1. i 是几位数，也就是 x
2. 10 的 x 次方

```
#include <bits/stdc++.h> //9-1729
using namespace std;
// 求 10 的 x 次方 (x 就是 n 的位数)
int fun(int n)
{
    int r = 1;
    while(n != 0)
    {
        r = r * 10; // 求 10 的次方
        n = n / 10;
    }
    return r;
}
int n, c = 0;
int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        if (i*i%fun(i)==i)
        {
            //cout<<i<<endl;
            c++;
        }
    }
    cout<<c;
    return 0;
}
```

思路：遍历 1900 到 n 的所有年份

能够被 4 整除但不能被 100 整除的为闰年或者能够被 400 整除的为闰年

```
#include <bits/stdc++.h> //10-1749
using namespace std;
int main()
{
    int n, s=0;
    cin>>n;
    // 遍历 1900 到 n 的所有年份
    for (int i=1900; i<=n; i++)
    {
        // 能够被 4 整除但不能被 100 整除的为闰年;
        // 或者能够被 400 整除的为闰年
        if ((i%4==0&& i%100!=0) || i%400==0)
        {
            s++; // 计数器 +1
        }
    }
    cout<<s;
    return 0;
}
```



```
#include <bits/stdc++.h>//11-1963
using namespace std;

int main()
{
    int n, m, k=0;
    cin>>n;
    for (int i=1; i<=n; i++)
    {
        m=i;
        // 短除法判断各个位是否有等于 6 的
        while (m>0)
        {
            if (m%10==6)
            {
                k++; // 计数器
                break;
            }
            m=m/10;
        }
    }
    // 判断含有 6 的数字的个数是奇数还是偶数
    // 是奇数，就去图书馆、否则就去游乐场
    if (k%2==0)
    {
        cout<<"playground";
    }
    else
    {
        cout<<"library";
    }
    return 0;
}
```

求 n 的质因子:

使用变量 i 从 2 开始找 n 的因子, 如果 i 是 n 的因子, 输出 i , 且 $n=n/i$

如果 i 不是 n 的因子, 看 $i+1$

直到 n 为 1

```
#include<bits/stdc++.h>//12-1080
using namespace std;
int main()
{
    // 从 2 开始找 n 的因子
    int n, i = 2;
    cin>>n;
    while(n != 1)
    {
        // 如果 i 是 n 的因子
        if(n % i == 0)
        {
            cout<<i<<endl;
            n = n / i;
        }
        else i=i+1;// 看下一个数
    }
    return 0;
}
```

$p+1/q+1/r+1/s=1$ 且 $p \leq q \leq r \leq s$ ，能得到 $p=q=r=s$ 时 p 最大， $4/p \geq 1$ ，则 $p \leq 4$ ，那么循环中的最大值就确定了，for 循环的最大值就是 4，比 4 大的数值就都不用考虑。同理就能推出 $q \leq 6$ 、 $r \leq 12$ 、 $s \leq 42$ ，能缩小 for 循环的范围。

```
#include<bits/stdc++.h>//13-1255
```

```
using namespace std;
```

```
int main()
```

```
{  
    for (int p = 2; p <= 4; p++)  
    {  
        for (int q = p; q <= 6; q++)  
        {  
            for (int r = q; r <= 12; r++)  
            {  
                // 分母不能为 0  
                if (p*q*r-q*r-p*r-p*q == 0)  
                    continue;  
                // 前三项确定，计算出第 4 项  
                int s = p*q*r/(p*q*r-q*r-p*r-p*q);  
                if (p*q*r%(p*q*r-q*r-p*r-p*q)==0&&s>=r)  
                {  
                    cout<<p<<" "<<q<<" "<<r<<" "<<s<<endl;  
                }  
            }  
        }  
    }  
    return 0;  
}
```

小明有五本新书，要借给 A, B, C 三位小朋友，若每人每次只能借一本。比如借出 3 本书，就有 123, 132, 312, 321, 213, 231 六种方案，可见借出的书与顺序有关，与顺序有关的组合就是排列，可以借助排列组合公式直接求出答案。排列：从 n 个数中选择 m 个不重复的数有序的排列。 $p(n, m) = n! / (n-m)!$ 此题已经告诉了 $n=5, m=3$ ；计算可得 $p(5, 3) = 60$ 。

```
#include <iostream>//14-1256-1
using namespace std;
// 求排列问题 p(n, m)
int main()
{
    int n, m, ans=1;
    n=5;
    m=3;
    //n!/(n-m)! 可以转化为 (n-m+1)*(n-m+2)*...*n;
    for (int i=n-m+1; i<=n; i++)
        ans*=i;
    cout<<ans;
    return 0;
}
```

思路：利用嵌套循环遍历所有的情况，计数器计数借法

定义 i, j, k 分别代表借给 A, B, C 的书

先借给 A 一本书，有 5 种情况，进行遍历

后借给 B 一本书，还剩 4 本书，所以有 4 种情况，进行遍历

最后借给 C 一本书，还剩 3 本书，所以有 3 种情况，进行遍历

```
#include <bits/stdc++.h> //14-1256-2 javacn
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    //i, j, k 分别代表借给 A, B, C 的书
```

```
    //num 用作计数有多少种借法，初始值为 0
```

```
    int i, j, k, num=0;
```

```
    // 先借给 A 一本书，有 5 种情况，进行遍历
```

```
    for (i=1; i<=5; i++)
```

```
    {
```

```
        // 然后借给 B 一本书，还剩 4 本书，所以有 4 种情况，进行遍历
```

```
        for (j=1; j<=4; j++)
```

```
        {
```

```
            // 最后借给 C 一本书，还剩 3 本书，所以有 3 种情况，进行遍历
```

```
            for (k=1; k<=3; k++)
```

```
            {
```

```
                // 每循环一次，借法加 1
```

```
                num++;
```

```
            }
```

```
        }
```

```
    }
```

```
    // 输出总借法
```

```
    cout<<num;
```

```
    return 0;
```

```
}
```

思路：利用嵌套循环，对 a, b, c, d, e 的情况进行遍历，需要满足 5 个数字互不相等 且 $ab * cde == adb * ce$ ，满足则计数器 +1

```
#include<bits/stdc++.h> //15-1257
```

```
using namespace std;
```

```
int main() {
```

```
    int count = 0; // 计数器记录 算式的种类
```

```
    for (int a=1; a<=9; a++) // 对 a, b, c, d, e 进行嵌套循环
```

```
    {
```

```
        for (int b=1; b<=9; b++)
```

```
        {
```

```
            for (int c=1; c<=9; c++)
```

```
            {
```

```
                for (int d=1; d<=9; d++)
```

```
                {
```

```
                    for (int e=1; e<=9; e++)
```

```
                    {
```

```
                        if (a!=b && a!=c && a!=d && a!=e && b!=c &&
```

```
b!=d && b!=e && c!=d && c!=e && d!=e) // 5 个数字互不相等
```

```
                        { // 计算 ab * cde 的结果, // 计算 adb * ce 结果
```

```
                            int k1 = (a*10+b)*(c*100+d*10+e);
```

```
                            int k2 = (a*100+d*10+b)*(c*10+e);
```

```
                            if (k1==k2) // 如果两个结果相同计数器 +1
```

```
                            {
```

```
                                count++;
```

```
                            }
```

```
                        }
```

```
                    }
```

```
                }
```

```
            }
```

```
        }
```

```
    }
```

```
    cout<<count; // 输出种类数
```

```
    return 0;
```

```
}
```

思路：用嵌套循环遍历取出的 8 个球中三种颜色球的可能的个数
三种球的个数之和要满足等于 8

```
#include <bits/stdc++.h> //16-1260
using namespace std;
int main()
{
    //i, j, k 分别代表红, 白, 黑球
    //s 用作计数器
    int i, j, k, s=0;

    // 遍历取出的 8 个球中, 红色球可能的个数
    for (i=0; i<=3; i++)
    {
        // 遍历取出的 8 个球中, 白色球可能的个数
        for (j=0; j<=3; j++)
        {
            // 遍历取出的 8 个球中, 黑色球可能的个数
            for (k=0; k<=6; k++)
            {
                // 三种颜色的球个数总和要等于 8
                if (i+j+k==8)
                {
                    s++;
                }
            }
        }
    }

    cout<<s;
    return 0;
}
```

嵌套循环，注意：不能有重复方案，要注意循环范围

```
#include<bits/stdc++.h>//17-1516
using namespace std;
int main()
{
    int n, i, j, k;
    cin>>n;
    for (i=1; i<=n/3; i++)    // 循环 i 的范围 1 到 n/3
    {
        for (j=i; j<=(n-i)/2; j++) // 不能有重复的方案，所以 j 的范围从 i 开始
        {
            k=n-i-j;
            cout<<n<<"="<<i<<"+"<<j<<"+"<<k<<endl;
        }
    }
    return 0;
}
```

```
#include<bits/stdc++.h>//18-1074
using namespace std;
int n, m, h, d, s;    //d 代表爬行天数
int main()
{
    cin>>n>>m>>h;
    while (s<h)    // 当还没出井口时，爬行高度 < 井口高度
    {
        d++; // 天数 +1
        s=s+n; // 向上爬
        if (s<h)    s=s-m;    // 如果没爬上来，往下掉
    }
    cout<<d;
    return 0;
}
```


思路：找到第 1 个，比 n 大的 2 的幂的值 x，比较 n 是离 x 更近，还是离 x/2 更近

```
#include<bits/stdc++.h>//19-1075
using namespace std;

int main()
{
    long long n, x;
    cin>>n;
    x = 2;// 起始值
    // 当 x<=n 的时候循环，循环停止 x 就一定是比 n 大的 2 的次方
    while(x <= n)
    {
        x = x * 2;
    }

    // 如果较大数 x 离 n 更近
    if(x - n < n - x / 2)
    {
        cout<<x;
    }
    else
    {
        cout<<x / 2;
    }

    return 0;
}
```

```

#include<bits/stdc++.h>//20-1079
using namespace std;

int main()
{
    int l, a, b, c, s=0;//s 作为三角形的个数
    cin>>l;
    //a 最长为 L/3
    //b>=a
    for ( a=1 ; a <= l / 3;a++ )
    {
        for ( b=a; b <= (l - a) / 2;b++)
        {
            // 第三条边的长度
            c = l - a - b;
            // 如果两边之和大于第三边，且不是等边三角形，个数加 1
            if (a+b>c&&(a==b&&b==c)==false)
            {
                s++;
                //cout<<a<<" "<<b<<" "<<c<<endl;
            }
        }
    }
    cout<<s;
    return 0;
}

```

思路：已知第十天剩下 1 个桃子，从第十天往前推，得出上一天剩下的桃子数量，以此逆循环得出第一天的数量。本题是递推，用数组更合适。

```
#include<bits/stdc++.h>//21-1082 推荐学习 javacn
```

```
using namespace std;
```

```
int main()
{
    int i;//代表第几天吃桃子
    int n=1;//代表剩下桃子的数量
    for (i=9;i>=1;i--)
    {
        //昨天剩下的桃子数量；
        n = (n+1)*2;
    }
    //输出 n
    cout<<n;
    return 0;
}
```

思路:

- 1、判断回文的方式: 如果数 $n == n$ 倒过来的数, 因此需要一个函数, 求 n 倒过来的数 (短除法)
- 2、使用 while 循环重复描述的过程, 直到 n 为回文数

```
#include<bits/stdc++.h> //22-1083 javacn
```

```
using namespace std;
```

```
// 求 n 倒过来的数
```

```
int fun(int n)
{
    int r = 0;
    // 短除法求 n 倒过来的数
    while(n != 0)
    {
        r = r * 10 + n % 10;
        n = n / 10;
    }
    return r;
}
```

```
int main()
{
    //cout<<fun(123); // 测试函数
    int n, c = 0;
    cin>>n;
    // 如果 n 不是回文
    while(n != fun(n))
    {
        c++;
        n = n + fun(n);
    }

    cout<<c;
    return 0;
}
```

`#include<iostream>`//23-1248-1 推荐学习 参考国王的金币，第 10 天发几个金币？

`using namespace std;`

```
int main() {
    int n, day=0, sum=0;//day 天数
    cin>>n;
    for (int i=1;i<=n;i++)
    {
        for (int j=1;j<=i;j++)
        {
            day++;
            if(day==n)
            {
                cout<<i;
                return 0;
            }
        }
    }
}
```

`#include <bits/stdc++. h>`//23-1248-2 javacn

`using namespace std;`

```
int main() {
    int n;
    cin>>n;
    int i, s=0;
    for (i=1;i<=n;i++)
    {
        s+=i;
        if(s>=n)
        {
            cout<<i;
            break;
        }
    }
    return 0;
}
```

思路：定义 4 人年龄为 i, j, k, l

因为 4 个人年龄是等差数列，所以知道前两人年龄，就可知后两人年龄

等差数列是指从第二项起，每一项与它的前一项的差等于同一个常数的一种数列

所以只需要遍历前两人的年龄就可以了，因为要从小到大展示，所以是个递增的等差数列

第二人的年龄要比第一人大，所以遍历范围要写成 $i+1$ 到 $26-i-2$

```
#include <bits/stdc++.h> //24-1251
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int i, j, k, l; // 分别代表 4 个人的年龄
```

```
    // 遍历第一个人年龄可能的范围
```

```
    for (i=1; i<=23; i++)
```

```
    {
```

```
        // 遍历第二个人年龄可能的范围
```

```
        for (j=i+1; j<=26-i-2; j++)
```

```
        {
```

```
            k = j + j - i; // 第三人年龄
```

```
            l = k + j - i; // 第四人年龄
```

```
            // 总年龄 = 26 且 乘积 = 880 则输出
```

```
            if (i+j+k+l==26 && i*j*k*l==880)
```

```
            {
```

```
                cout<<i<<" "<<j<<" "<<k<<" "<<l<<endl;
```

```
            }
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

```

#include <bits/stdc++.h>//25-1318
using namespace std;
int main()
{
    long long m, n, num=0;//num 是方案数量
    cin>>m>>n;
    // 总人数 s
    long long s = m*n;
    for (int i=2;i<=sqrt(s);i++)
    {
        if (s%i==0)
        {
            num++;
        }
    }
    // 可变换方案需要总数 -1
    cout<<num-1;
    return 0;
}

```

```

#include <bits/stdc++.h>//26-1332
using namespace std;
int main()
{
    int n, s=0;
    cin>>n;
    // 两两比一次，每个队要打的队伍数累加，去掉重复的比赛
    for (int i=n-1;i>=1;i--)
    {
        s+=i;
    }
    cout<<s;
    return 0;
}

```

思路：遍历裁掉的小正方形的边长可能的范围
计算出体积，在循环中保留最大的体积

```
#include <bits/stdc++.h> //27-1350
using namespace std;
int main()
{
    int n, v=0, v1;
    cin>>n;

    // 遍历裁掉的小正方形的边长可能的范围
    for (int i=1; i<=n/2; i++)
    {
        // 裁掉的小正方形边长为 i 时的体积为 v1
        v1 = (n-i*2)*(n-i*2)*i;
        // 存放较大的体积
        if (v1>v)
        {
            v = v1;
        }
    }
    cout<<v;
    return 0;
}
```


`#include <bits/stdc++.h> //28-1352` 思路：穷举找出结果不为质数的数，输出并跳出循环

`using namespace std;`

`bool zs(long long n) // 判断是否是质数，是质数返回 true，不是返回 false`

```
{
    bool r = true;
    for (int i=2; i<=sqrt(n); i++)
    {
        if (n%i==0)
        {
            r = false;
            break;
        }
    }
    if (n<=1)
    {
        r = false;
    }
    return r;
}
```

`long long fun(long long n) // 计算 2 的 n 次阶乘`

```
{
    long long s=1;
    if (n==0)
    {
        s=1;
    }
    else
    {
        for (long long i = 1; i<=n; i++)
        {
            s *= 2;
        }
    }
    return s;
}
```

```
int main()
{
    int i=1;
    while(true)// 穷举 i, 找出结果不为质数的 i, 输出并跳出循环
    {
        long long s = fun(fun(i))+1;
        if(!zs(s))
        {
            cout<<i;
            break;
        }
        i++;
    }
    return 0;
}
```

遍历门票增加的价格可能范围

原本售出张数 1200 / 每增加一元减少的张数 = 门票最多可增加的价格

```
#include <bits/stdc++.h> //29-1355
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    //n 为门票减少的张数
```

```
    //p 为票价增加的价格
```

```
    //z 为总门票收入
```

```
    //p0 存放总门票收入最高时票价增加的价格
```

```
    //s 存放最高的总门票收入
```

```
    int n, p, z, p0, s=0;
```

```
    cin>>n;
```

```
    // 遍历门票增加的价格可能范围
```

```
    // 本题中 n 为 20, 1200/20=60, 也就是如果你增加价格超 60 元, 门票一张也卖不出
```

```
    for (p=0;p<=1200/n;p++)
```

```
    {
```

```
        // 总门票收入
```

```
        z = (30+p)*(1200-p*n);
```

```
        if (z>s)
```

```
        {
```

```
            s=z;
```

```
            p0=p;
```

```
        }
```

```
    }
```

```
    cout<<p0+30<<" "<<s;
```

```
    return 0;
```

```
}
```

思路：已知周长，遍历一条边可能的范围 循环记录最大的面积

```
#include <bits/stdc++.h>//30-1356
using namespace std;
int main()
{
    int n, s=0; // 周长
    cin>>n;
    for (int i=1; i<=n/2-1; i++) // 遍历一条边可能的范围
    {
        if (i*(n/2-i)>s) // 记录最大的面积
        {
            s = i*(n/2-i);
        }
    }
    cout<<s;
    return 0;
}
```

```
#include <bits/stdc++.h>//31-1468
using namespace std;
int main()
{
    int n, x, s=0;
    cin>>x>>n;
    // 从周 x 开始，周 x 也要算一天，所以范围到 x+n-1 号
    for (int i=x; i<=x+n-1; i++) //7 天是一周，周六周日休息
    {
        if (i%7!=6&& i%7!=0)
        {
            s+=250;
        }
    }
    cout<<s;
    return 0;
}
```

思路：循环 n 天，当发放 k 金币的天数等于 k 时，金币数要增加

```
#include<bits/stdc++.h> //32-1520
using namespace std;
int main()
{
    // 定义整数 n 代表天数，sum 代表总金币，k 代表每天发放的金币数，c 代表发放 k 金
    币的天数
    int n, sum=0, c=0, k=1;
    cin>>n;
    // 循环 n 天
    for (int i=1; i<=n; i++)
    {
        sum+=k; // 总金币
        c++; // 发放 k 金币数的天数
        // 当发放 k 金币的天数等于 k 时
        if (c==k)
        {
            c=0; //c 要清 0, 重新开始计数
            k++; // 发放金币数 +1
        }
    }
    cout<<sum<<endl;
    return 0;
}
```

简单穷举

```
#include <bits/stdc++.h> //1-1015
using namespace std;
int main()
{
    // 定义变量 j, t 分别代表鸡的只数和兔子的只数
    int j, t;
    for (j=1; j<=49; j++)
    {
        t = 50 - j; // 兔子 + 鸡 =50
        if ((2*j+4*t)==160) // 当腿的数量 =160, 总只数 =50, 就输出 j, t
        {
            cout<<j<<" "<<t<<endl;
        }
    }
    return 0;
}
```

思路：遍历成人人数可能的范围 $40 - \text{成人票总价} = \text{儿童票总价}$ ，儿童票总价要满足是儿童票单价的倍数 满足条件则输出结果

注意：要从少到多输出成人，从多到少输出儿童，所以要注意循环遍历的顺序

```
#include <bits/stdc++.h> //2-1351
using namespace std;
int main() {
    int c, e; // 分别表示成人和儿童的人数
    for (c=1; c<=(40-3)/8; c++) // 遍历成人人数可能的范围
    {
        if ((40-c*8)%3==0)
        { //40- 成人票总价 = 儿童票总价, 儿童票总价是儿童票单价的倍数
            e = (40-c*8)/3;
            cout<<c<<" "<<e<<endl;
        }
    }
    return 0;
}
```

```

#include <iostream>//3-1016-1 javacn 推荐学习
using namespace std;
int main()
{
    // 共 x 元，小狗 a 元 / 只，小猫 b 元 / 只，至少猫狗各一，正好用完
    int x, a, b, y, i; // y 代表买了 i 只小狗剩余的钱
    int c=0; // 计数的变量 (count)
    cin>>x>>a>>b; // 循环小狗可能的只数范围
    for (i=1; i<=(x-b)/ a; i++)
    {
        y =x-i * a; // 计算买了 i 只小狗剩余的金额
        if(y % b == 0) // 如果剩余的金额是小猫单价的倍数，则方案成立
        {
            //cout<<i<<" " <<y / b<<endl;
            c=c+1;
        }
    }
    cout<<c<<endl;
    return 0;
}

```

```

#include <iostream>//3-1016-2 买小猫小狗 xuyuling 推荐学习
using namespace std;
int main()
{
    int x, a, b, cnt =0 ;
    cin >> x >> a >> b ;
    for ( int i = 1 ; i <= (x-b)/a ; i ++)// i 是狗，j 是猫，至少有一直狗
    {
        int j = (x - a * i )/b;
        if (a*i + b*j == x )
            cnt ++ ;
    }
    cout << cnt ;
    return 0;
}

```

```

#include <iostream>//3-1016-3  liangls
using namespace std;
int main()
{
    int x, a, b, cnt=0;
    cin>>x>>a>>b;
    int max = x/(a<b?a:b);
    for (int m=1; m<=max; ++m)
    {
        for (int g=1; g<=max; ++g)
        {
            if (m*a+g*b==x) { cnt++; }
        }
    }
    cout<<cnt;
    return 0;
}

```

```

#include <iostream>//3-1016-4  xiaogangsi  推荐学习
using namespace std;
int main()
{
    int x = 0;
    int a = 0;
    int b = 0;
    int count = 0;
    cin>>x>>a>>b;
    for (int i = 1; i <= (x-b)/a; i++)
    {
        int j = x-i*a;
        if (j%b == 0) { count++; }
    }
    cout<<count<<endl;
    return 0;
}

```


题目需要面包最多的方案，那么我们从大到小输出面包可能的购买件数范围，输出第 1 个满足条件的解，就是面包最多的方案。

```
#include <bits/stdc++.h> //4-1220   javacn
using namespace std;
int main()
{
    int n, x, y, t, i;
    cin >> n >> x >> y;
    for (i = (n - y) / x ; i >= 1; i--)    // 循环面包可能的件数范围
    {
        t = n - i * x;    // 计算剩余的钱
        if (t % y == 0)    // 如果剩余的钱是蛋挞单价的倍数
        {
            cout << i << " " << t / y << endl;
            break; // 找到第 1 个方案就不要继续找了
        }
    }
    return 0;
}
```

注意：掌握 break 和 continue 的用法！

1、break 表示：停止循环

```
for (i = 1; i <= 10; i++)
{
    cout << i << " ";
    if (i >= 3) { break; }
}
```

输出：1 2 3

2、continue：忽略本次循环下面的语句，直接开始下一次循环

```
for (i = 1; i <= 10; i++) {
    if (i % 3 == 0) {
        continue;
    }
    cout << i << " ";
}
```

输出：1~10 之间，除了 3 的倍数以外的数，1 2 4 5 7 8 10

```

#include <bits/stdc++.h>//5-1227  javacn
using namespace std;
int main()
{
    int q, b, s; //q 表示带的钱, b 表示大碗价格, s 代表小碗价格
    cin >> q >> b >> s;
    for (int i = 2; i <= (q - 2 * s) / b; i += 2) // 遍历大碗可能的只数
    {
        if ((q - b * i) % s == 0) // 减去大碗的钱剩下的是否正好买小碗没有剩余
        {
            int x = (q - b * i) / s; // 小碗的只数
            if (x % 2 == 0 && (b * i + x * s) == q) // 小碗只数是偶数且带的钱正好花完
            {
                cout << i << " " << x << endl;
            }
        }
    }
    return 0;
}

```

思路：对从甲组抽出的可能人数范围进行遍历
当抽出的人数能使乙人数是甲的两倍时，输出 i

```

#include <bits/stdc++.h>//6-1349  javacn
using namespace std;
int main()
{
    int j = 17, y = 25; // 甲组有 17 人, 乙组有 25 人
    for (int i = 1; i <= j; i++) // 对从甲组抽出的可能人数范围进行遍历
    {
        if ((y + i) == 2 * (j - i)) // 当抽出的人数能使乙人数是甲的两倍时, 输出 i
        {
            cout << i;
        }
    }
    return 0;
}

```

思路：遍历篮球和排球可能的个数范围，要满足篮球和排球的总数要超过 50 且 n 元经费全部用完。注意：至少都有一个

```
#include <iostream>//7-1396   javacn
using namespace std;
int main() {
    int n, x, y, t;
    cin>>n>>x>>y;
    for (int i = 1; i <= (n - y) / x; i++) // 枚举篮球的购买范围
    {
        t = n - i * x;           // 计算剩余的钱
        if (t % y == 0 && i + t / y > 50)
        {
            cout<<i<<" "<<t / y<<endl;
        }
    }
}
```

```
#include<bits/stdc++.h>//8-1792   javacn
using namespace std;
int main() {
    // 使用穷举法猜 可以兑换的张数
    int i, j; //i:10 元张数 j:20 元张数
    for (i=1; i<=10; i++) // 因为 10 元最多可以换 10 张
    {
        for (j=5; j>=1; j--)//5 元最多可以换 5 张，从大到小猜
        {
            if ( 10*i + 20*j == 100 )
            {
                cout<<i<<" "<<j<<endl;
            }
        }
    }
    // 注意：每一种面额都要有，都应该从 1 开始
    return 0;
}
```

思路 :for 循环遍历苹果重量可能的范围 ,
通过苹果的重量可得梨子的重量
梨子的重量要满足 n 元正好花完且大于等于 10
符合条件则计数器 +1

```
#include<bits/stdc++.h>//9-1793   javacn
using namespace std;
int main()
{
    // 分别代表 总金额、苹果的单价和梨的单价
    //num 计数方案的数量
    int n, x, y, num=0;
    cin>>n>>x>>y;
    int i, j; // 代表苹果的重量, 梨子的重量
    // 遍历苹果重量可能的范围
    for (i=10; i<=(n-10*y)/x; i++)
    {
        j=(n-i*x)/y; // 梨子的重量
        //n 元正好花完且梨子的重量大于等于 10
        if ((n-i*x)%y==0&&j>=10)
        {
            num++; // 符合条件则方案数 +1
        }
    }
    cout<<num; // 输出总的方案数
    return 0;
}
```

本题的已知条件有：

- 1、霸王龙玩偶一只需要 x 元，三角龙玩偶一只需要 y
- 2、有 n 元，两种能购买，不能有钱剩下
- 3、霸王龙的数量 \geq 三角龙的数量
- 4、总数要在 5 个或者 5 个以上

穷举思路如下：循环霸王龙能够买到的只数范围，用买完霸王龙剩余的钱买三角龙，只要满足上述条件就可以。

```
#include <bits/stdc++.h>//10-1394   javacn
using namespace std;
int main()
{
    //t 代表剩余的钱
    int n, x, y, t, i;
    cin >> n >> x >> y;

    // 循环霸王龙可能买到的只数范围
    for (i = 1; i <= (n - y) / x; i++)
    {

        t = n - i * x; // 计算买完霸王龙剩余的钱

        //t%y==0 判断不能有钱剩下，也就是买完霸王龙的钱剩余的，正好可以买三角龙
        if (t%y==0&&i>=t/y&&i+t/y>=5)
        {
            cout << i << " " << t / y << endl;
        }
    }
    return 0;
}
```

嵌套穷举

穷举公鸡能买到的只数范围，用买公鸡剩余的钱买母鸡，再用买母鸡剩余的钱买小鸡，如果总只数是 100 只，则符合题意。

```
#include <iostream>//1-1022-1 javacn 推荐学习
using namespace std;
int main()
{
    //x 代表买完公鸡剩余的钱
    //y 代表买完母鸡剩余的钱
    int x, y, i, j;
    // 枚举公鸡的范围
    for (i = 1; i <= (100-3-1)/5; i++)
    {
        // 买完公鸡计算剩余的钱
        x = 100 - i * 5;
        // 枚举母鸡的范围
        for (j = 1; j <= (x - 1) / 3; j++)
        {
            // 买完公鸡和母鸡剩余的钱
            y = x - j * 3;
            // 剩余 y 元必然买小鸡
            if (i+j+y*3==100)
            {
                cout<<i<<" "<<j<<" "<<3*y<<endl;
            }
        }
    }
    return 0;
}
```

```
#include <iostream>//1-1022-2 最佳实践
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int i, j, k;
```

```
    for (int i=1; i<=(100-1-3)/5; i++)
```

```
    {
```

```
        for (int j=1; j<=(100-1-3)/3; j++)
```

```
        {
```

```
            k = 100 - i - j;
```

```
            if (i*5+j*3+k/3.0 == 100)
```

```
            {
```

```
                cout << i << " " << j << " " << 100 - i - j << endl;
```

```
            }
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

```

#include <iostream>//2-1024-1 javacn
using namespace std;
int main()
{
    /*
    N元钱 每样至少买一支 总数要超过 30 支 而且钱要全部花完
    圆珠笔 8 角钱一支、铅笔 2 角钱一支、铅笔芯 1 角钱一支
    */
    int n, i, j, x, y;
    int c = 0; // 方案总数
    cin >> n;
    n = n * 10; // 将单位元换算为角

    for (i = 1; i <= (n - 2 - 1) / 8; i++) // 循环圆珠笔可能的购买数量范围
    {
        x = n - i * 8; // 计算买完圆珠笔剩余的钱
        for (j = 1; j <= (x - 1) / 2; j++) // 用 x 角去购买铅笔
        {
            y = x - j * 2; // 计算买完圆珠笔和铅笔剩余的钱
            if (i + j + y > 30) // 如果圆珠笔 + 铅笔 + 铅笔芯总数超过 30 支、
            {
                c++;
            }
        }
    }
    cout << c << endl;
    return 0;
}

```



```
#include<iostream>//2-1024-2
using namespace std;

int main()
{
    int n, cnt = 0;
    cin >> n;
    for (int i = 1; i <= 10 * n / 8; i ++) // 注意要写 <=
    {
        for (int j = 1; j <= 10 * n / 2; j ++) // 注意要写 <=
        {
            int h = (10 * n - 8 * i - 2 * j) / 1;
            if ( i + j + h > 30 && h > 0 )
            {
                // cout << i << " " << j <<" " << h << endl;
                cnt ++;
            }
        }
    }
    cout << cnt;
    return 0;
}
```

穷举 1 分能换到的数量，用换完 1 分剩余的钱换 2 分，用换完 2 分剩余的钱换 5 分。

```
#include <bits/stdc++.h> //3-1025   javacn
using namespace std;

/*
一元票换 1 分、2 分和 5 分
每种至少一枚
*/
int c = 0;
//x 表示换完 1 分剩余的钱
//y 表示换完 1 分和 2 分剩余的钱
int x, y;

int main()
{

    for (int i = 1; i <= (100-2-5)/1; i++) // 穷举 1 分可能的换法
    {
        x = 100 - i * 1;

        for (int j = 1; j <= (x - 5) / 2; j++) // 换 2 分
        {
            y = x - j * 2;

            if (y % 5 == 0) // 换 5 分
            {
                c++;
            }
        }
    }

    cout<<c;
    return 0;
}
```

```

#include<bits/stdc++.h>//4-1076    javacn
using namespace std;

int main()
{
    int i, j, x, y, c=0;

    for (i=1; i<=(100-2-4)/2; i++)    // 循环九头鸟的只数
    {
        x=100-i*2;    // 除去九头鸟剩下的脚数量
        for (j=1; j<=(x-4)/2; j++)    // 循环鸡的只数
        {
            y=x-j*2;    // 除去九头鸟和鸡的剩下的脚的数量
            // 总头数 =100 且剩下脚的数量是 4 的倍数，就输出对应的只数
            if (y%4==0&& i*9+j*1+y/4==100)
            {
                cout<<i<<" "<<j<<" "<<y/4<<endl;
                c++;    // 题目解的总数 +1
            }
        }
    }

    cout<<c;
    return 0;
}

```

```

#include<bits/stdc++.h>//5-1077   javacn
using namespace std;
int main()
{
    // 定义整数变量 n,m 分别代表 n 元钱 ,m 只鸡
    int n,m;
    // 输入 n,m
    cin>>n>>m;
    for (int g=0;g<=n/5;g++) // 循环公鸡的范围
    {
        for (int h=0;h<=(n-5*g)/3;h++) // 循环母鸡的范围
        {
            int x = (n-5*g-3*h)*3; // 小鸡的只数
            if (g+h+x==m)
            {
                cout<<g<<" "<<h<<" "<<x<<endl;
            }
        }
    }
    return 0;
}

```

思路：循环遍历男人人数可能的范围

循环遍历女人人数可能的范围

根据循环中男人，女人的人数，可以得到小孩的人数

人数需要满足共 36 人

```
#include <bits/stdc++.h> //6-1249-1 javacn
using namespace std;
int main()
{
    int i, j, k; //i, j, k 分别表示男人，女人，小孩
    // 循环男人人数的可能范围，男，女搬得都是整数，所以小孩搬得也是整数
    for (i=1; i<=(36-3-1)/4; i++)
    {
        for (j=1; j<=(36-4*i-1)/3; j++) // 循环遍历女人人数的范围
        {
            k=36-4*i-3*j; // 剩余的砖数

            if (i+j+k*2==36) // 检验总人数为 36
            {
                cout<<i<<" "<<j<<" "<<k*2<<endl;
            }
        }
    }

    return 0;
}
```

```

#include <iostream>//6-1249-2 kevinh
using namespace std;

int main()
{
    /*
        36 块砖， 36 人搬。
        男搬 4 ， 女搬 3 ， 两个小儿抬一砖。
        要求一次全搬完
        男、女、小孩都有

        所有可能的人数分配方案，按照由小到大输出

        循环男的人数
        再循环女的人数
        再判断满足条件则输出
    */

    int i, j, x, y;
    for (i=1; i<=(36-3-i)/4; i++) // 循环男的人数，至少 1 人，至多 (36-3-1) /4
    {
        x = 36-i*4; // 剩余的砖数
        for (j=1; j<=(x-1)/3; j++) // 循环女的人数，至少 1 人，至多 (x-1) /3
        {
            y = x-j*3; // 剩余的砖数
            if (i+j+y*2==36) // 满足总人数要 36 人
            {
                cout<<i<<" "<<j<<" "<<y*2<<endl;
            }
        }
    }

    return 0;
}

```

```

#include <iostream>//7-1250-1 kevinh
using namespace std;
int main()
{
    /*
        有 30 个人，其中可能有男人、女人和小孩
        共花了 50 先令
        每个男人各花 3 先令，每个女人各花 2 先令，每个小孩各花 1 先令

        问男人、女人和小孩各有几人
        按照男人、女人、小孩的顺序，由小到大依次输出所有可能的人数方案

        循环男人人数
        再循环女人人数
        再判断是否满足条件

    */

    int i, j, x, y;
    for (i=0; i<=50/3; i++) // 循环男人人数，至少 0 人，至多 (50-0-0) /3
    {
        x = 50-i*3; // 男人花完剩余的钱
        for (j=0; j<=x/2; j++) // 循环女人人数，至少 0 人，至多 x/2
        {
            y = x - j*2; // 男人和女人花完剩余的钱，小孩花的钱
            if (i+j+y==30) // 满足 30 人
            {
                cout<<i<<" "<<j<<" "<<y<<endl;
            }
        }
    }
    return 0;
}

```

思路：循环遍历男人的人数范围

循环遍历女人的人数范围

可得小孩的人数

需要满足总人数为 30 人

```
#include <bits/stdc++.h> //7-1250-2 javacn
using namespace std;
int main()
{
    int i, j, x, y;

    for (i = 0; i <= 50 / 3; i++) // 循环男人可能的人数
    {
        x = 50 - i * 3; // 剩余的钱
        for (j = 0; j <= x / 2; j++) // 循环女人可能的人数
        {
            y = x - j * 2;

            if (i+j+y==30) // 剩余的 y 元，分配给小孩，要求总人数为 30 人
            {
                cout<<i<<" "<<j<<" "<<y<<endl;
            }
        }
    }
    return 0;
}
```


已知：

- 1、准备种 n 棵树
- 2、三种树都得有（告知我们：最少种 1 棵）
- 3、每种树的数量都得是偶数
- 4、桃树的数量不能比梨树的数量多，梨树的数量不能比苹果树的数量多

思路：

1、用变量 i ，循环桃树可能的数量范围： $2 \sim n-4$ （不可能都是桃树，至少有 2 棵梨树、2 棵苹果树）

当然，也可以循环 $2 \sim n/3$ ，因为桃树数量 \leq 梨树数量 \leq 苹果树数量，意味着桃树数量最多占 $1/3$

2、使用 j 循环梨树的数量范围： $i \sim (n-i-2)$ ：梨树最少有 i 个，最多有 $n-i-2$ 个

当然，范围还可以缩小到 $(n-i)/2$ ，因为除了桃树，剩余 $n-i$ 棵树，梨树 \leq 苹果树的数量，所以最多占 1 半，也就是最多有 $(n-i)/2$ 棵

本题类似：1516：【入门】将 n 拆成 3 个数的和

解法一：假设桃树最多循环的 $n-4$ ，梨树循环到 $n-i-2$

```
#include <bits/stdc++.h> //8-1342-1 javacn
using namespace std;
int main()
{
    int i, j, n;
    cin >> n;
    for (i = 2; i <= n - 4; i = i + 2) // 循环桃树的数量
    {
        for (j = i; j <= n - i - 2; j = j + 2) // 循环梨树的数量
        {
            // 剩余一定是苹果树，要保证梨树的数量 <= 苹果树的数量
            if (j <= n - i - j)
            {
                cout << i << " " << j << " " << (n - i - j) << endl;
            }
        }
    }
    return 0;
}
```

解法二：假设桃树最多循环的 $n/3$ ，梨树循环到 $(n-i)/2$

```
#include <bits/stdc++.h> //8-1342-2 javacn
using namespace std;

int main()
{
    int i, j, n;
    cin >> n;

    for (i = 2; i <= n / 3; i=i+2) // 循环桃树的数量
    {
        for (j = i; j <= (n - i) / 2; j=j+2) // 循环梨树的数量
        {
            // 剩余一定是苹果树，如果梨树的数量不到 n-i 的一半，梨树数量 <= 苹果树的数量
            cout << i << " " << j << " " << (n-i-j) << endl;
        }
    }
    return 0;
}
```