递归进阶

用递归改造循环

请输出 1~n 之间所有的整数

```
#include<iostream>//1-1696  jiangyf70
using namespace std;
void print(int n)
{
    if(n == 0) return;
    print(n - 1);
    cout << n << endl;
}
int main()
{
    int n;
    cin >> n;
    print(n);
    return 0;
}
#include<iostream>//2-1697  jiangyf70 请输出 n~1 之间所有的整数
using namespace std;
void print(int n)
{
    if(n == 0) return;
    cout << n << endl;;
    print(n - 1);
}
int main()
{
    int n;
    cin >> n;
    print(n);
    return 0;
}
```

编程求解 1+2+3+...+n

```
#include<iostream>//3-1002  jiangyf70
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;
}
```

递归求解：

```
#include <bits/stdc++.h>//4-1088-1    javacn
using namespace std;
// 求 x 和 y 的最大公约数
long long fun(long x,long long y){
	if(x % y != 0) return fun(y , x % y);
	else return y;
}
long long a,b;
int main(){
	cin>>a>>b;
	cout<<fun(a,b);
	return 0;
}
```

欧几里得算法又称辗转相除法，是指用于计算两个非负整数 a，b 的最大公约数。

比如：假设求 45 和 60 的最大公约数。

45 % 60 = 45

60 % 45 = 15

45 % 15 = 0

因此结果是 15。

循环求解：

```
#include <bits/stdc++.h>//4-1088-2    javacn
using namespace std;
// 辗转相除法，求最大公约数
long long a,b,t;
int main()  {
	cin>>a>>b;
	while(a % b != 0){
		t = a % b;// 存储余数
		a = b;
		b = t;
	}
	cout<<b;
	return 0;
}
```

解法 2：递归法调用

```
#include <bits/stdc++.h>//5-1241-1   javacn
using namespace std;
int n;
int fun(int k){
      int r = 0;
      if(k != 1)
      {
        // 如果 k 是偶数，将 k/2 交还给函数
            if(k % 2 == 0)         r = 1 + fun(k / 2);
            else       r = 1 + fun(k * 3 + 1);
      }
      return r;
}
int main()  {
    cin>>n;
    int x = fun(n);
    cout<<x;
}

#include<iostream>//5-1241-2    jiangyf70
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;
}
```

正整数 N 转换成一个二进制数

```
#include<iostream>//6-1108-1  jiangyf70
using namespace std;
string i2b(int n, string s){
    if(n)
    {
        s = char(n % 2 + '0') + s;
        return i2b(n / 2, s);
    }
    return s;
}
int main(){
    int n;
    cin >> n;
    if(n == 0) cout << 0;
    else cout << i2b(n, "");
    return 0;
}
#include<bits/stdc++.h>//6-1108-2  javacn
using namespace std; 除 2 取余，结果倒过来连成字符串：
int n; //10 进制转 2 进制
string r = "";// 存放转换结果
int main(){
      cin>>n;
      while(n != 0)      // 当 n!=0 循环
      {
            //cout<<n%2;// 取余
            // 将 n%2 的结果转换为字符 + 到 r 前面
            r = char(n%2+'0') + r;
            n=n/2;// 除 2
      }
      if(r == "") cout<<0;     // 特判输入为 0 的情况
      else cout<<r;
    return 0;
}
```

求 100+97+……+4+1 的值

```
#include<iostream>//7-1053  jiagyf70
using namespace std;
int sum(int n)
{
    if(n == 1) return 1;
    return sum(n - 3) + n;
}
int main()
{
    int n = 100;
    cout << sum(n);
    return 0;
}
```

求恰好使 s=1+1/2+1/3+…+1/n 的值大于 X 时 n 的值

```
#include<iostream>//8-1078-1 jiangyf70
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>//8-1078-2    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;
}
```

```cpp
#include<iostream>//9-1261-1    jiangyf70
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

```cpp
#include<bits/stdc++.h>//9-1261-2    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;
            }
            i++;
      }
      return 0;
}
```

求出 100 至 999 范围内的所有水仙花数

```
#include<iostream>//10-1058-1    jiangyf70
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;
}
```

```cpp
#include<bits/stdc++.h>//10-1058-2  javacn          本题解题思路不是递归
using namespace std;

/*
      函数名 lfsum(int x)
      参数 int x
      返回值 int r
      说明  求各位数字立方之和
*/
int lfsum(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(lfsum(i)==i){
            cout<<i<<endl;
        }
    }
    return 0;
}
```

猴子吃桃子

```
#include<iostream>//11-1082-1   jiangyf70
using namespace std;
int hz(int n, int s)
{
    if(n > 1)
    {
        hz(n - 1, (s + 1) * 2);
    }
    else return s;
}
int main()
{
    cout << hz(10, 1);
    return 0;
}
```

写法二：

```
#include<iostream>//11-1082-2    jiangyf70
using namespace std;
int hz(int n, int s)
{
    if(n == 1) return s;
    hz(n - 1, (s + 1) * 2);
}
int main()
{
    cout << hz(10, 1);
    return 0;
}
```

爱因斯坦的数学题

```
#include<iostream>//12-1265-1  jiangyf70
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;
}
```

思路：用穷举法，从一开始循环，直到找到符合条件的数，就跳出循环

```
#include <bits/stdc++.h>//12-1265-2  javacn
using namespace std;
int main()
{
      int i=1;
      // 从一开始穷举，直到找到复合条件的数，跳出循环
      while(1)
      {
            if(i%2==1&&i%3==2&&i%5==4&&i%6==5&&i%7==0)
            {
                  cout<<i<<endl;
                  break;
            }
            i++;
      }
      return 0;
}
```

```
/*
	1265 - 【入门】爱因斯坦的数学题
*/
#include <iostream>//12-1265-3   kevinh
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;
}
```

解法 2

```
#include<iostream>//13-1395-1    jiangyf70
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;
}
```

小丽找数？

解法 1

```
#include<iostream>//13-1395-2  jiangyf70
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;
}
```

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>//13-1395-3    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;
}
```

```cpp
#include<iostream>//14-1083-1  递归 jiangyf70
using namespace std;
int cnt = 0;
int hw(int n)
{
	int m = 0, a = n;
	while(a)
	{
		m = m * 10 + a % 10;
		a /= 10;
	}
	if(m == n) return 0;
	return 1 + hw(n + m);// 增加次数

}
int main()
{
	int n;
	cin >> n;
	cout << hw(n);
	return 0;
}
```

回文数

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>//14-1083-2  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;
}
```

字符图形 2- 星号直角　　递归

```
#include<iostream>//15-1066-1  jiangyf70
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>//15-1066-2   javacn
using namespace std;
int main()
{
      int n;
      cin>>n;      // 输入 n
      int i,j;
      for(i=1;i<=n;i++)
      {
            // 每行循环 i 次输出 *
            for(j=1;j<=i;j++) {   cout<<"*";    }
            cout<<endl;
      }
      return 0;
}
```

字符图形 2- 星号倒直角　　递归

```
#include<iostream>//16-1782-1  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>//16-1782-2  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=i;j<=n;j++){    cout<<"*";       }  //每行输出 n+1-i 个 *
            cout<<endl;
      }
      return 0;
}
```

```cpp
#include <bits/stdc++.h>//17-1783-1    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;
}
```

数字直角（1）

```cpp
#include <bits/stdc++.h>//17-1783-2  javacn
using namespace std;
int main()
{
	int n;
	cin>>n;
	int i,j;
	for(i=1;i<=n;i++)		// 输出 n 行
	{
		// 每行输出 i 列
		for(j=1;j<=i;j++)		cout<<i;
		cout<<endl;
	}
	return 0;
}
```

数字直角（2）

```
#include<iostream>//18-1784-1    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>//18-1784-2  javacn
using namespace std;
int main(){
      int n;
      cin>>n;
      int i,j;
      //输出n行
      for(i=1;i<=n;i++)
      {
            //每行输出i列
            for(j=1;j<=i;j++)  cout<<j;
            cout<<endl;
      }
      return 0;
}
```

字符图形 9- 数字正三角

```
#include<iostream>//19-1088-1  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>//19-1088-2   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
      // 循环数字前空格的展示
        for(j=1;j<=n-i;j++){     cout<<" ";      }
        // 数字的循环展示
        for(j=1;j<=2*i-1;j++){      cout<<i;     }
        cout<<endl;
    }
      return 0;
}
```

请问一个正整数能够整除几次 2

```
#include<iostream>//20-1244-1  jiangyf70
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>//20-1244-2    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;
}
```

递归的应用

```
#include <iostream>//1-1088-1   anselxu
using namespace std;
long long gcd(long long x,long long y)// 辗转相除法求最大公约数
{
	if(y==0)	return x;	// 除数为 0，输出 x

	return	gcd(y,x%y);	// 否则，除数为被除数，余数为除数继续调用 gcd
}
int main()
{
	long long n,m;
	cin>>n>>m;
	cout<<gcd(n,m);
	return 0;
}
```

递归求解：

```
#include <bits/stdc++.h>//1-1088-2  javacn
using namespace std;
long long fun(long x,long long y)// 求 x 和 y 的最大公约数
{
	if(x % y != 0)	return fun(y , x % y);
	else		return y;
}
long long a,b;
int main()
{
	cin>>a>>b;
	cout<<fun(a,b);
	return 0;
}
```

欧几里得算法又称辗转相除法，是指用于计算两个非负整数 a，b 的最大公约数。

比如：假设求 45 和 60 的最大公约数。

45 % 60 = 45　　60 % 45 = 15　　45 % 15 = 0

因此结果是 15。　　循环求解：

```
#include <bits/stdc++.h>//1-1088-3    javacn
using namespace std;
long long a,b,t;
int main() {
	cin>>a>>b;
	while(a % b != 0){// 辗转相除法，求最大公约数
		t = a % b;// 存储余数
		a = b;
		b = t;
	}
	cout<<b;
	return 0;
}
```

求两个数 M 和 N 的最大公约数

```
#include<bits/stdc++.h>//1-1088-4  hasome
using namespace std;
/*
更相减损法：
a=b，则最大公约数为 a
如果 a>b 则 a=a-b 否则 b=b-a
*/
int main(){
	long long a,b;
	cin>>a>>b;
	while(a!=b){
		if(a>b) a=a-b;
		else b=b-a;
	}
	cout<<a<<endl;
	return 0;
}
```

两个数 M 和 N 的最小公倍数

```
#include<iostream>//2-1087-1  jiangyf70
using namespace std;
long long gcd(long long n, long long m)
{
    if(n % m == 0) return m;
    return gcd(m, n % m);
}
int main()
{
    long long n, m;
    cin >> n >> m;
    cout << n / gcd(n, m) * m; // 不要先做 n * m 可能会爆 long long 范围
    return 0;
}
```

用乘积 / 最大公约数：

```
#include<bits/stdc++.h>//2-1087-2  javacn
using namespace std;
typedef long long LL;
int main() {
      LL m,n,t,a,b;
      cin>>m>>n;
      a = m;
      b = n;// 临时存储 mn 的值，因为接下来要修改 mn 的值
      while(m % n != 0){
            t = m % n;
            m = n;
            n = t;
      }
      // 最小公倍数 = a * b / 最大公约数
      cout<<a * b / n;
}
```

阿克曼（Ackmann）函数

```
#include <bits/stdc++.h>//3-1695  hyb
using namespace std;
int ack(int m,int n)
{
	if(m == 0) return n + 1;
	else if(m != 0 && n == 0) return ack(m-1,1);
	else if(m != 0 && n != 0) return ack(m-1,ack(m,n-1));
}

int main()
{
	int m,n;
	cin>>m>>n;
	cout<<ack(m,n);
	return 0;
}
```

数根

```
#include<iostream>//4-1514-1  jiangyf70
using namespace std;
int sg(int n)
{
    if(n < 10) return n;
    int s = 0;
    while(n)
    {
        s += n % 10;
        n /= 10;
    }
    sg(s);
}
int main()
{
    int  n;
    cin >> n;
    cout << sg(n);
    return 0;
}
```

```cpp
#include<bits/stdc++.h>//4-1514-2    javacn
using namespace std;
// 定义函数  求一个数的各个位上的和
int qiuhe(int n) {
	int s=0;//s 变量存放数的每位的和
	// 通过短除法求
	while(n!=0) {
		s=s+n%10;
		n=n/10;
	}
	return s;
}
int main() {
	int n,m;
	cin>>n;
	//n 变量如果是一位数，树根就是自己  否则要不停的求和直到是个位数
	while(n>=10){
		m = qiuhe(n);
		n = m;
	}
	cout<<n<<endl;
	return 0;
}
```

汉诺塔的移动次数

```
#include <bits/stdc++.h>//5-1223  javacn
using namespace std;
/*
通过递推发现规律：
n 个金片要移动次数 = （n-1）个金片要移动的次数 * 2 + 1
用 A(n) 表示 n 个金片移动的步数：
A(n) = A(n-1) * 2 + 1
因为：
需要先将（n-1）个顶部的金片从 A，借助于 C，移动到 B（A(n-1) 步）
将最底层的最大金片移动到 C（1 步）
需要先将（n-1）个顶部的金片从 B，借助于 A，移动到 C（A(n-1) 步）

因此得出结论：
A(n) = A(n-1) * 2 + 1
*/

// 求 n 个金片的移动次数
int fun(int n){
	if(n == 1) return 1;//1 个金片不需要递归
	else return fun(n-1)*2+1;
}

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

经典递归问题——汉诺塔

A 为存放盘子的塔，C 为目标塔，B 为辅助塔

```
#include <bits/stdc++.h>//6-1222  javacn
using namespace std;
/*
    移动的过程：
    1、先将 n-1 个金片（n 个金片除了最下面一个以外的金片）
       从 A 位置，借助 C 位置，移动到 B 位置，需要 fun(n-1) 步
    2、将 A 位置的最下方的一个金片，直接移动到 C 位置，需要 1 步
    3、将 B 位置的 n-1 个金片，从 B 位置借助 A 位置移动到 C 位置
       需要 fun(n-1) 步
    因此得到结论如下：fun(n) = fun(n-1) * 2 + 1
  函数的作用：将 n 个金片从 p1 位置，借助 p2 位置，移动到 p3 位置
  p1：源位置（金片所在的源位置）
  p2：辅助位置（移动的过程中要借助的位置）
  p3：目标位置（移动到的位置）
*/
void move(int n,char p1,char p2,char p3){
      // 递归的出口：只要有金片就要递归，没有金片递归停止
      if(n > 0){
            // 第 1 步：将 n-1 个金片，从 p1 位置，借助 p3 位置，移动到 p2 位置
            move(n-1,p1,p3,p2);
            // 第 2 步：将 p1 位置的第 n 个金片直接移动到 p3 位置
            cout<<p1<<" To "<<p3<<endl;
            // 第 3 步：将 p2 位置的 n-1 个金片，借助于 p1 位置，移动到 p3 位置
            move(n-1,p2,p1,p3);
      }
}
int main(){
      int n;
      cin>>n;
      // 递归调用，打印移动的步骤
      move(n,'A','B','C');
      return 0;
}
```

土地分割

把一块 M*N 米的土地分割成同样大的正方形，要求 1：没有土地剩余，2：分割出的正方形土地最大。 没有剩余就是说 M、N 都能被分割出来的正方形边长整除； 分割出来的土地最大，也就是说在可以整除 M、N 的边长中找最大的那个数 ； 结论：求 M、N 的最大公约数。 利用欧几里得算法轻松实现：（注意数据范围）

```
#include <iostream>//7-1335   anselxu
using namespace std;
// 辗转相除法求最大公约数
long long gcd(long long x,long long y){
	// 除数为 0，输出 x
	if(y==0)	return x;
	// 否则，除数为被除数，余数为除数继续调用 gcd
	return	gcd(y,x%y);
}
int main()
{
	long long n,m;
	cin>>n>>m;
	cout<<gcd(n,m);
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//8-1208-1    螺旋方阵 深搜 jiangyf70      推荐学习
using namespace std;
int q[20][20];
int dx[] ={0, 1, 0, -1};
int dy[] ={1, 0, -1, 0};
int n, i = 0;
void fun(int x, int y, int k)
{
	if(x>0 && x<=n && y > 0 && y <= n && q[x][y] == 0)// 不越界，并且数值为 0
	{
		q[x][y] = k;
		int tx, ty;
		tx = x + dx[i], ty = y + dy[i];
		if(tx<1 || tx>n || ty<1 || ty>n || q[tx][ty])// 越界，或者数值不为 0
		{
			i = (i + 1) % 4;// 拐弯
			tx = x + dx[i], ty = y + dy[i];
		}
		fun(tx, ty, k+1);
	}
}
int main()
{
	cin >> n;
	fun(1, 1, 1);
	for(int i = 1; i <= n; i++)
	{
		for(int j = 1; j <= n; j++)
			cout << setw(3) << q[i][j];
		cout << endl;
	}
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//8-1208-2    螺旋方阵 递归 jiangyf70    仅供参考
using namespace std;
int q[20][20];
int dx[] ={0, 1, 0, -1};
int dy[] ={1, 0, -1, 0};
int n;
void lxfz(int a, int b, int c)
{
	if(b > 0)
	{
		int x = a , y = a, k = 0;
		q[x][y] = c++;
		for(int i = 0; i < 4; i++)
		{
			for(int j = 0; j < b - 1; j++ )
			{
				x = x + dx[i], y = y + dy[i];
				if(q[x][y] == 0)	q[x][y] = c++;
			}
		}
		lxfz(a + 1, b - 2, c);
	}
}
int main()
{
	cin >> n;
	lxfz(1, n, 1);
	for(int i = 1; i <= n; i++)
	{
		for(int j = 1; j <= n; j++)
			cout << setw(3) << q[i][j];
		cout << endl;
	}
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//8-1208-3  螺旋方阵 偏移数组 jiangyf70   仅供参考
using namespace std;
int q[20][20];
int dx[] ={0, 1, 0, -1};
int dy[] ={1, 0, -1, 0};
int main()
{
	int n;
	cin >> n;
	int x = 0, y = 0, k = 0;
	for(int i = 1; i <= n * n; i++)
	{
		q[x][y] = i;
		int a = x + dx[k], b = y + dy[k];
		if(a < 0 || a >= n || b < 0 || b >= n || q[a][b])
		{		// 如果越界，或者数值不为 0，那么拐弯
			k = (k + 1) % 4;
			a = x + dx[k], b = y + dy[k];
		}
		x = a, y = b;
	}
	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j < n; j++)
		{
			cout << setw(3) << q[i][j];
		}
		cout << endl;
	}

	return 0;
}
```

非官方题解

```
#include<bits/stdc++.h>//8-1208-4    hasome
using namespace std;
int a[20][20],n;
int main()
{
	int i,j,x=1;
	cin>>n;
	for(i=1;i<=(n+1)/2;i++)
	{
		for(j=i;j<=n-i+1;j++)    a[i][j]=x++;
		for(j=i+1;j<=n-i+1;j++) a[j][n-i+1]=x++;
		for(j=n-i;j>=i;j--)      a[n-i+1][j]=x++;
		for(j=n-i;j>i;j--)       a[j][i]=x++;
	}
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
			cout<<setw(3)<<a[i][j];
		cout<<endl;
	}
	return 0;
}
```

```
#include <bits/stdc++.h>//9-1209-1 javacn 递归实现：    推荐学习
using namespace std;
int a[20][20], n;
// 为二维数组的第 x 圈赋值 //start：起始点的坐标 //len：赋值的宽度 //x：起始值
void fun(int start, int len, int x)
{
	if(len > 0)		// 递归出口
	{
		int i, j;

		for(j = start; j <= start + len - 1; j++) // 循环第 1 行的列（向右）
		{
			a[start][j] = x;
		}
		// 循环向下，最后一列，循环行
		for(i = start + 1; i <= start + len - 1; i++)
		{
			a[i][start + len - 1] = x;
		}
		// 循环向左，最后一行，循环列
		for(j = start + len - 2; j >= start; j--)
		{
			a[start + len - 1][j] = x;
		}
		// 循环向上，第 1 列，循环行
		for(i = start + len - 2; i >= start + 1; i--)
		{
			a[i][start] = x;
		}
		// 递归为 start+1 这一圈赋值
		fun(start + 1, len - 2, x - 1);
	}
}
```

```
int main()
{
	cin >> n;
	// 为从 1,1 点开始的一圈赋值，边长为 n * 2 + 1
	fun(1, n * 2 + 1, n);
	n = n * 2 + 1;	// 输出
	int i, j;
	for(i = 1; i <= n; i++)
	{
		for(j = 1; j <= n; j++)
		{
			cout << setw(2) << a[i][j];
		}
		cout << endl;
	}
}
```

```
#include<bits/stdc++.h>//9-1209-2	tc_ljp6	仅供参考
using namespace std;
int main()
{
	int n;
	cin>>n;
	int s=2*n+1;
	int a[s][s];
	for(int i=0;i<s;i++)
	{
		for(int j=0;j<s;j++)
		{
			cout<<setw(2)<<max(abs(n-j),abs(n-i));
		}
		cout<<endl;
	}
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//9-1209-3  回形方阵        递归    jiangyf70
using namespace std;
int q[30][30];
int dx[] ={0, 1, 0, -1};
int dy[] ={1, 0, -1, 0};
void hxfz(int a, int n) //在 a 位置填充 n
{
    if(n >= 0)
    {
        int x = a , y = a;
        q[x][y] = n;
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 2 * n; j++ )
            {
                x = x + dx[i], y = y + dy[i];
                q[x][y] = n;
            }
        }
        hxfz(a + 1, n - 1);
    }
}
int main()
{
    int n;
    cin >> n;
    hxfz(1, n);
    for(int i = 1; i <= 2 * n + 1; i++)
    {
        for(int j = 1; j <= 2 * n + 1; j++){   cout << setw(2) << q[i][j]; }
        cout << endl;
    }
    return 0;
}
```

递归

```
#include<iostream>//10-1562-1  jiangyf70
using namespace std;
int sum = 0;
int cnt(int n) //求位数
{
    int res = 0;
    while(n)
    {
        n /= 10;
        res++;
    }
    return res;
}

int fun(int n)
{
    if(n == 0) return sum;
    sum += cnt(n);
    fun( n / 2);
}


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

加数

边界返回 0，递归返回当前数长度 + 除 2 后的数长度

```
#include<iostream>//10-1562-2  anselxu
#include<iomanip>
using namespace std;
int cd(int x){
	int len=0;
	while(x){
		len++;
		x/=10;
	}
	return len;
}
int fc(int x){
	if(x==0) return 0;
	return cd(x)+fc(x/2);
}
int main(){
	int n,tot=0;
	cin>>n;
	cout<<fc(n);
	return 0;
}
```

计算出当前数的位数 + 当前数不断除 2 的位数的和，直到计算到 0。

```
#include <bits/stdc++.h>//10-1562-3    javacn
using namespace std;

int n;

// 计算一个整数的位数
int fun(int x){
	int r = 0;
	while(x != 0){
		r++;
		x /= 10;
	}
	return r;
}


int main(){
	int ans = 0;
	cin>>n;
	while(n != 0){
		ans += fun(n);
		n /= 2;// 加一半的位数
	}
	cout<<ans;
	return 0;
}
```