1.1-N的整数中k的个数，k=(0,1,2,3,4,5,6,7,8,9)

1.1.1 剑指offer上的方法

按照剑指offer上的分治法，写出递推公式函数，然后统计所有的数字个数，减去1-9的数字，余下的就是0的个数

代码如下：

```cpp
#include<iostream>
#include <math.h>
#include <sstream>
#include <string>
#include <vector>
using namespace std;
int numberofk(int k,string & str,int bg,int len){
    if(len<=bg){
        return 0;
    }
    int first=str[bg]-'0';
    if(first==0){
        return numberofk(k,str,bg+1,len);
    }
    if(len-bg==1 ){
        if(first==0){
            return 0;
        }
        else if(first>=k)
            return 1;
        else{
            return 0;
        }
    }
    //在最高位k出现的次数
    int numfd=0;
    if(first>k){
        numfd=pow(10.0,len-bg-1);
    }
    else if(first<k){
        numfd=0;
    }
    else if(first==k){
        stringstream ss;
        ss<<str.substr(bg+1,len);
        ss>>numfd;
        numfd++;
    }
    //  if(first==k)
    //      numfd++;
    //余下k出现的次数
    int otherfd=0;
    otherfd=first*(len-bg-1)*pow(10.0,len-bg-2);
    int subfd=numberofk(k,str,bg+1,len);

    return numfd+otherfd+subfd;
}
double sumnum(long long int num,int len){
    double sumv=0;
    for(int i=len-1;i>=1;i--){
        sumv+=num-pow(10.0,i)+1;
    }

    sumv+=num;
```

```
    return sumv;

}
int main(){
    long long int num=0;
    while(cin>>num){
        string str;
        stringstream ss;
        ss<<num;
        ss>>str;
        int len=str.size();
        vector<int> v(10);
        double sumv=sumnum(num,len);
        double sumu=0;
        for(int k=1;k<10;k++){
            v[k]=numberofk(k,str,0,len);
            sumu+=v[k];
        }
        v[0]=sumv-sumu;
        for(int k=0;k<=8;k++){
            cout<<v[k]<<" ";
        }
        cout<<v[9]<<endl;
    }
    return 0;
}
```

1.1.2 编程之美的方法

推导

1-9的递推公式

假设有十进制数abcde，假设取百位的数字c，需要求百位上的k的个数，c有三种情况，大于k，等于k，和小于k。

$$\begin{cases} c<k: numofk: k00-k99, 1k00-1k99, 2k00-2k99, \dots, (ab-1)k00-(ab-1)k99 \\ c=k: numofk: k00-k99, 1k00-1k99, 2k00-2k99, \dots, (ab-1)k00-(ab-1)k99, abk00-abkde \\ c>k: numofk: k00-k99, 1k00-1k99, 2k00-2k99, \dots, (ab-1)k00-(ab-1)k99, abk00-abk99 \end{cases}$$

上面的公式表示在百位上k出现的次数的各种情况，横杠为多少到多少。

将各个段加起来，有：

$$\begin{cases} c<k: numofk=ab*10^2 \\ c=k: numofk=ab*10^2+de+1 \\ c>k: numofk=ab*10^2+10^2 \end{cases}$$

其中ab代表高位，de代表低位，c代表当前位，将上面的公式带入到最高位和最低位的情况，依然成立，10^2代表此时c的位数的数量级，如果是个位就是10^3，如果是个位就是10^0.

现在考虑k=0的情况，0不能出现在首位：

$$\begin{cases} c=0: numof0: 1000-1099, 2000-2099, \dots, (ab-1)000-(ab-1)099, ab000-ab0de \\ c>0: numof0: 1000-1099, 2000-2099, \dots, (ab-1)000-(ab-1)099, ab000-ab099 \end{cases}$$

各段加起来，有：

$$\begin{cases} c=0:numof0=(ab-1)*10^2+de+1 \\ c>0:numof0=(ab-1)*10^2+10^2 \end{cases}$$

从某一位开始，遍历每一位，就可以得到每一位上出现k的次数，加起来则是这个数中出现k的次数。当k==0的时候，不要遍历最高位。

```
#include<iostream>
#include <sstream>
using namespace std;
int numofk(unsigned int &num,unsigned int k){
    if(k>9)
        return 0;
    int factor=1;
    int hfactor=10;
    int curnum=0;
    int high=0,low=0;
    int sum=0;

    while(num/factor!=0){
        high=num/hfactor;
        curnum=num/factor-(high)*10;
        low=num-high*hfactor-curnum*factor;
        if(k==0){
            high--;
            if(num/hfactor<=0){
                break;
            }
        }
        sum+=high*factor;
        if(curnum>k)
            sum+=factor;
        else if(curnum==k)
            sum+=low+1;
        factor=hfactor;
        hfactor*=10;
    }
    return sum;
}
int main(){
    unsigned int num=0;
    while(cin>>num){
        for(int i=0;i<=8;i++){
            cout<<numofk(num,i)<<" ";
        }
        cout<<numofk(num,9)<<endl;
    }
    return 0;
}
```

2. 二进制数中1的个数

设二进制数为num=11011，num-1=11010,num&(num-1)=11011&11010=11010,减少了最左边的1.

二进制数为num=11010，num-1=11001,num&(num-1)=11010&11001=11000,减少了最左边的1.

二进制数为num=11000，num-1=10111,num&(num-1)=11000&10111=10000,减少了最左边的1.

二进制数为num=10000，num-1=01111,num&(num-1)=10000&01111=00000,减少了最左边的1，数字减为0

```
int binarynumof1(unsigned int num){
    int sum=0;
    while(num){
        num=num&(num-1);
        sum++;
    }
    return sum;
}
```

可以用来判断数是否是2的幂次，只需要判断num>0且num&(num-1)==0,即右边只有一个1.

3. 阶乘中尾部零的个数

由于$N!$可以是整数相乘，可以分解为质因数相乘的形式即$N! = (2^x) * (3^y) * (5^z) * (7^a)...$,除了2和5的质因子乘起来会产生零，其余都不可能产生零，而2的个数远多于5，故尾部零的个数和5的幂相等。

在1-N之间，5的倍数的每隔5个数就会产生一个，第一轮[N/5]表示可以隔多少隔5，就有多少隔5的倍数，那么每个5的倍数贡献一个5，一共就有[N/5]个5。第二轮[N/25]，在贡献完5之后，如果是25的倍数，则每隔25会产生一个，之前贡献了一个5，第二轮除以25就会再贡献一个5。

代码如下：

```
    int getFactorSuffixZero(int n) {
        // write code here
        int znum=0;
        while(n){
            n/=5;
            znum+=n;
        }
        return znum;

    }
```