1.求一个数字的反序数

```
int Reverse(int x){
    int revx=0;
    while(x!=0){
        revx *= 10;
        revx =rev+(x%10);
        x /= 10;
    }
    return revx;
}
```

2.十进制数字转化为二进制

```
int main(){
int n=0;
while(scanf("%d",&n)!=EOF){
    int a[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};    //需要几位的二进制数就创建长度为几的数组
    int i=0;                        //作为数组的下标
    while(n>0){                    //进行分解
        a[i]=n%2;                   //取余
        n=n/2;                     //除二
        ++i;                       //下标自增
    }
    for(int i=15;i>=0;--i){         //这里要特别注意反序输出
        cout<<a[i];
    }
    cout<<endl;
```

```
}
}
```

3.求一个数字的每一位数

```
int main(){
        int i=5689
        int a=i%10;
        int b1=i%100;
        int b=b1/10;
        int c1=i%1000;
        int c=c1/100;
        int d=i/1000;
        cout<<a<<endl;
        cout<<b<<endl;
        cout<<c<<endl;
        cout<<d<<endl;
    return 0;
}
```

4.使用二维数组处理日期问题

```
//日期问题想到二维数组

#include<iostream>
#include<cstdio>

using namespace std;

int dayt[2][13]={   //最重要的就是这个二维数组的应用
    {0,31,28,31,30,31,30,31,31,30,31,30,31},
    {0,31,29,31,30,31,30,31,31,30,31,30,31}
};

bool isRunYear(int year){
    return(year%4==0&&year&100!=0)||(year%400==0);
}

int main(){
    int year,month,day;
    while(scanf("%d%d%d",&year,&month,&day)!=EOF){
        int number=0;
        int row=isRunYear(year);
        for(int i=0;i<month;++i){
            number=number+dayt[row][i];
        }
        number=number+day;
        printf("%d\n",number);
    }
}
```

5.函数表达式的计算

1.数学公式对应的头文件的语句是：#include<math.h>
//C++用的头文件#include<cmath>

2.n开根号函数是：sqrt(n)

3.e的n次方的函数是：exp(n)

4.求n的绝对值的函数是：fabs(n)。使用绝对值的时候要看一下自变量的取值范围，如果是特殊的取值范围就可以不使用函数

//C++用到的三角函数：Sin，cos，asin，acos
5.求sin30°的函数是三步曲：定义，转化为弧度，计算

```
#define PI acos(-1.0)
double r=30*PI/180;
sin(r);      //sin（x）就是直接求函数的sin值
```

//这里的acos（x）函数，实际就是求x的arccos。
//arccos(1)=0; arccos(0)=pi/2; arccos(-1.0)=pi;
//结合余弦函数的图像来理解

6.x的y次方函数是：pow(x,y)

6.输入输出方法合集

1.循环输入一个字符数组：（可以用来输入字符串）
将字符串看成字符数组，先声明一个字符数组，使用gets()函数将字符串读入字符数组中。此处的gets函数在头文件stdio.h中，不用额外扩入头文件

```
char str[1000];  //声明字符数组，长度应该尽量大
while(gets(str)){}  //每次读取一个字符数组
int len=strlen(str);      //得到字符数组的长度，此处的strlen()函数是属于#include <string.h>头文件的
```

2.循环输入一个小数运行程序的框架是：在主题部分外围加上语句：

```
while(scanf("%lf",&x)!=EOF){}
```

如果是double类型的小数这里是%lf，如果是float类型的小数这里是%f，如果是整数这里是%d，如果是字符串这里是%s

以上所有输入方式都可以采用标准输入流完成，使用标准输入流符号cin>>

3.要求保留两位小数输出：

```
printf("%.2f\n", z);
```

4.使用gets()函数和getline()函数都能从控制台输入一个包含空格的字符串
gets()函数的用法：

```
char s[1000];
gets(s);          //赋值给一个字符数组
```

getline()函数的用法：

```
string str;
getline(cin,str);    //赋值给一个字符串
```

5.字符串里面如果有数字，要将字符串中的数字看做字符，而非直接看成整

6.使用while循环进行读入一组数字进行判断

```
    int m=0;
    while(1){       //循环输入
                scanf("%d",&m); //赋值
                if(m==n){
                    ++i;
```

7.第一个数字表示实际上要输入几个数字，后面的数字是实际上输入的数字，这些数字之间使用空格隔开

```
    int len=0;
    cin>>len;           //cin把第一个赋值给len作为长度
    int a[len];
    for(int i=0;i<len;++i){ //cin能够直接判断空格，只把数组赋值给数组
        cin>>a[i];
    }
                }
                c=getchar();        //getchar()表示从标准输入流中读取一个字符，当该字符是回车时退出
                if(c=='\n'){
                    break;
                }
            }
```

7.将一个合数分解为多个质数相乘

```
//代码的整体思想是先从2开始分解，如果2能将合数整除直接输出，判断得到的结果是否为合数，如果是重新调用分解函数，如果不是合数就直接输出
//如果不能被2整除，就++i

void toResult(int x){       //分解一个合数，使用递归的方法
    int t=0,i=2;
    while(i<x){
        if(x%i==0){
            t=x/i;          //能够整除就其中的一个因子
            cout<<i<<"*";
            ++i;
            if(isNotZhiShu(t)){     //如果t不是质数就继续分解
                toResult(t);
                break;      //当进入递归之后，就结束当前循环，不要让i继续++了
            }else{
                cout<<t<<endl;
                return ;        //如果t是质数（则说明已经分解完毕）就跳出递归
            }
        }else{
            ++i;        //如果不能整除则++i
        }
    }
}
```

8.冒泡排序函数

```
//从小到大排序，冒泡排序使用双循环，外层循环是0-len，内层循环是0-len-i-1

        for(int i=0;i<len;++i){
            for(int j=0;j<len-i-1;++j){
                if(str[j]>str[j+1]){
                    char temp=str[j];
                    str[j]=str[j+1];
                    str[j+1]=temp;
                }
            }
        }
```

9.将数组中的元素逆置

```
for(int j=0;j<i;j++,i--){    //将得到的字符数组进行反转，i是数组的最后一个元素的下标，j是数组的第一个元素的下标
        temp=str[i];
        str[i]=str[j];
        str[j]=temp;
    }


//这里最需要注意的就是：一个下标是0，一个小标是最后一个位置（最后一个位置是长度减一）
```

10.字符型数字转化为整形数字

```
1.字符型数字转化为整型数字的方法

            int len=strlen(str);              //得到数组的实际长度
            int s=0,k=0;              //辅助变量，s用于存储实际的二进制数字，可用于将字符型数字转换为整型数字
            for(int i=0;i<len;++i){           //循环操作字符数组中的字符
                if(str[i]>='0'&&str[i]<='9'){    //遇到字符型数字
                    k=str[i]-'0';             //字符型数字通过减零操作转化为整型数字
                    s=s*10+k;         //乘加得到整型数字
                }else{                //遇到小数点就说明得到了一个输入的数字，即可以进行转化了
                    change(s);
                    s=0;                    //转化为二进制，则将s重置为0
                    cout<<".";
                }
            }
            change(s);
            cout<<endl;


2.注意字符转化为数字一定是减零操作，加零操作是不可以的
```

11.关于ASCII码

1.比较字符的ASCII大小，可以直接使用><运算符

2.A的ASCII码是65，a的ASC码是97，它们之间相差32

3.ASC||码的加减法直接加数字即可，实际上表示的是字符

12.关于杨辉三角

1.先确定两头的元素是1，再将中间的元素计算出来

```
2.int main(){
    int n=0;
    cout<<"请输入杨辉三角的维数："<<endl;
    while(scanf("%d",&n)!=EOF){
        int a[n][n];    //申请二维数组空间
        a[0][0]=1;  //初始化第一个元素为1
        for(int i=1;i<n;++i){
            for(int j=0;j<=i;++j){
                if(j==0||j==i){ //每一行的第一个元素和最后一个元素都是1
                    a[i][j]=1;
                }else{
                    a[i][j]=a[i-1][j-1]+a[i-1][j];  //如果不是每一行的首尾元素，就计算
出来
                }
            }
        }
        for(int i=n-1;i>=0;--i){    //二维数组的循环输出，注意这里是倒着输出，正序还是倒
序输出只需要控制输出就行了，往二维数组中写入的时候统一正序
            for(int j=i;j>=0;--j){
                cout<<a[i][j]<<" ";
            }
            cout<<endl;
        }
    }
}
```

13.关于货币表示

1.货币表示，每三位要求一个逗号，将输入的数字表示为字符数组

```
for(int i=0;i<=k;++i){  //k是整数部分的长度
            if((k-i)%3==0&&(k!=i)&&(i!=0)){ //在特定的位置加上逗号
                cout<<",";
            }
            cout<<str[i];
        }
```

14.二维数组初始化

1.二维数组的初始化：（应该是使用两重循环）

```
int b[10][10];
    for(int i=0;i<10;++i){  //二维数组全部初始化为10
        for(int j=0;j<10;++j){
            b[i][j]=10;
            cout<<b[i][j]<<" ";
        }
        cout<<endl;
    }
```

15.给定一个数字，乘数之差最小的那一组

```
//给定一个数字，找到乘数之差最小的那一组

    int len=0,N=0,M=0,min=0;
    int t=INT_MAX;
    cin>>len;            //输入指定数字
    for(int i=1;i<=len;++i){     //找到符合题意的矩阵形式
        if(len%i==0){    //当可以整除的时候进行判断
            int n=i;         //n列
            int m=len/i;     //m行
            if((m>=n)&&(m-n<=t)){    //找到差最小的那一组
                min=t;
                M=m;     //差距最小的那一组的行数
                N=n;     //差距最小的那一组的列数
            }
        }
    }

    int ans[M][N];       //声明符合题意的二维数组
```

16.将数字转化为字符数组

一、调用标准库实现

```
    string s=to_string(int)；函数
```

二、代码实现

```
char* my_itoa(int num){ //将数字转化为字符串
        char *str;       //声明字符数组
    int i=0;         //数组下标
    while(num!=0){  //对数字进行除10
        str[i]=num%10+'0';  //余数作为字符数组中的一项
        num=num/10;
        i++;
    }
    str[i]='\0';         //最后一位的下一位设置为NULL
    i--;         //将数组下标调整到最后一位的位置
```

```
    char temp;
    for(int j=0;j<i;j++,i--){    //将得到的字符数组进行反转
        temp=str[i];
        str[i]=str[j];
        str[j]=temp;
    }
    return str;      //返回字符数组
}
```

17.求斐波那契数列的两种方法

```
1.递归的方法

int feibo(int n){        //求斐波那契数列的第n项，使用递归
    if(n==1||n==2){
        return 1;
    }else{
        return feibo(n-1)+feibo(n-2);
    }
}


2.非递归

int a=1,b=1,cou=0;
    int sum[30];    //声明一个数组，这里求的是斐波那契数列的前三十项
    sum[0]=a;
    sum[1]=b;
    for(int i=2;i<=29;++i){ //使用for循环求
        int c=a+b;
        a=b;
        b=c;
        sum[i]=c;
    }
```

18.判断回文数

```
一、判断回文数
bool isHuiwen(int n){    //判断是否为回文数
    int a[10]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};  //初始化辅助数组
    int i=0;
    int m=n;     //保存传入参数的初值
    while(n!=0){     //分解数字
        a[i]=n%10;
        n=n/10;
        ++i;
    }
    int j=0,s=0;
    while(a[j]!=-1){
        s=s*10+a[j];
        ++j;
    }
```

```
    if(s==m){
        return true;
    }else{
        return false;
    }
}
```

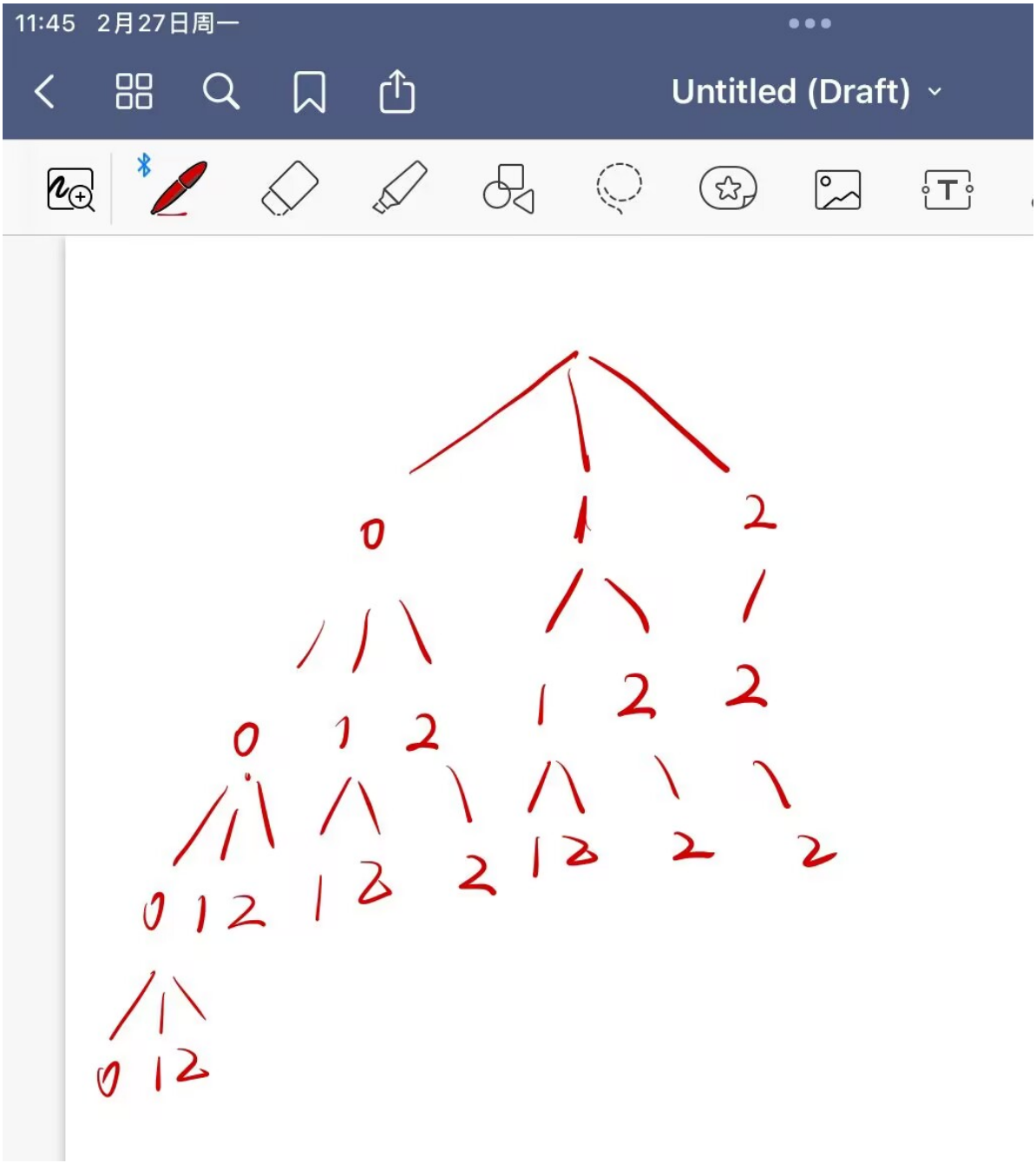
19.在不使用辅助数组的情况下，循环右移数组元素

```
1.代码：
void move(int a[],int n){
    int t=0;
    t=a[n-1];
    for(int i=n-1;i>=1;--i){
        a[i]=a[i-1];         //循环右移一位
    }
    a[0]=t;
}

2.一定要注意数组下标的越界，n个元素的数组，最后一个元素是a[n-1]
```

20.递归实现1元

思路：优化与剪枝。



用递归实现，显示用1分、2分和5分的硬币凑成1元，一共有多少种方法。

```
int ans = 0;
int a[3] = {5, 2, 1};
void func (int m, int n) {
    if (m == 0) {
        ans++;
        return;
    }
    for (int i = n; i < 3; i++) {
        if (m >= a[i]) {
            func(m - a[i], i);
        }
    }
}

int main() {
    func(100, 0);
    printf("%d", ans);
    return 0;
```

```
}
```