2、利用数组提高算法质量

- 原始信息与处理结果对应存储
- 数组记录状态信息
- 数组使信息有序化

2.1原始信息与处理结果对应存储

- 有时选择用数组存储信息，并把题目中的有关信息作为下标使用，可以使算法的实现过程大大简化。

2.1原始信息与处理结果对应存储

- 例：一次考试共考了语文、数学和外语三科。共有九人，考后各科及格名单如下表，请编写算法找出三科全及格的学生的学号。

科目	及格学生学号
语文	1，9，6，8，4，3，7
数学	5，2，9，1，3，7
外语	8，1，6，7，3，5，4，9

2.1原始信息与处理结果对应存储

- 算法设计1：
  - 从语文及格名单中逐一抽出各及格学生学号，先在数学及格名单核对，若有该学号，再在外语及格名单中继续查找，看该学号是否也在外语及格名单中。
  - 采用枚举尝试的方法，A，B，C三组分别存放语文、数学、外语及格名单，尝试范围为三重循环：
    - I循环，初值0，终值6，步长1
    - J循环，初值0，终值5，步长1
    - K循环，初值0，终值7，步长1
  - 定解条件：A[I]=B[J]=C[K] ，共尝试7*6*8=336次。

2.1原始信息与处理结果对应存储

- 算法实现1：

```
main( )
{
   int a[7]={1,9,6,8,4,3,7},b[6]={5,2,9,1,3,7};
   int c[8]={8,1,6,7,3,5,4,9},i,j,k;
   printf("三科都及格的学生学号为：");
   for(i=0;i<7;i++)
   {
        for(j=0;j<6;j++)
                if(b[j]==a[i])
                {       for(k=0;k<8;k++)
                                if(c[k]==a[i])
                                {       printf("%d, ",a[i]);
                                        break;
                                }
                        break;
                }
   }
}
```

2.1原始信息与处理结果对应存储

- 算法设计2：
  - 用数组A的九个下标分量（a[1]——a[9]）作为各号考生及格科目的计数器。将三科及格名单共存一个数组，凡a[1]——a[9]计数器的值为3的就是全及格的，其余则至少有一科不及格。
  - 累加部分为一重循环，初值1 ，终值为三科分别及格的总人数7+6+8=21，包括重复部分。
  - 定解条件：a[i]=3，共尝试9次。

2.1原始信息与处理结果对应存储

- 算法实现2：

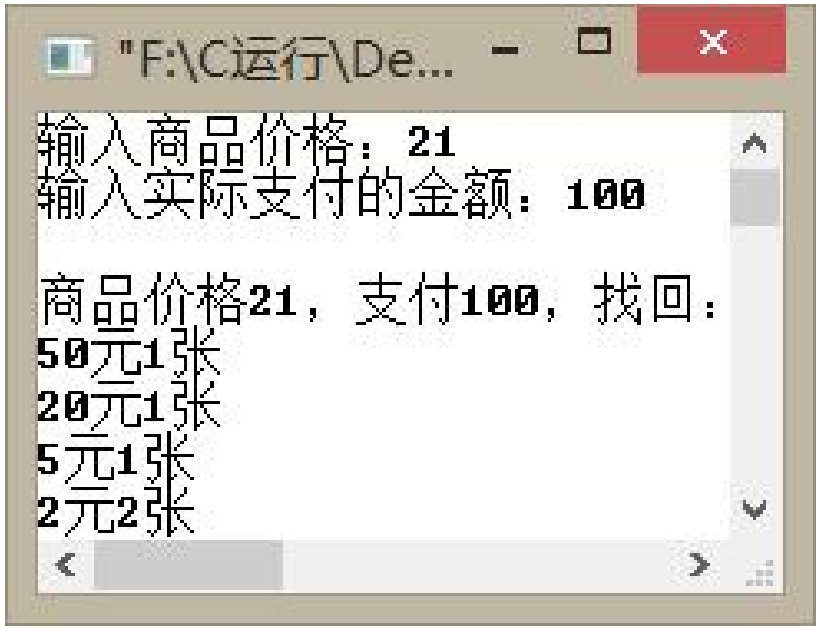
```
main( )
{
   int a[10]={0},i,xh;
   for(i=1;i<=21;i++)
   {
        scanf("%d",&xh); // 1,9,6,8,4,3,7,5,2,9,1,3,7,8,1,6,7,3,5,4,9
        a[xh]++;
   }
   printf("三科都及格的学生学号为：");
   for(xh=1;xh<=9;xh++)
        if(a[xh]==3)
                printf("%d, ",xh);
}
```

2.2数组使信息有序化

- 当题目中的数据缺乏规律时，很难把重复的工作抽象成循环不变式来完成，但先用数组结构存储这些信息后，问题就迎刃而解了。

2.2数组使信息有序化

- 例：一个顾客买了价值x元的商品，并将y元的钱交给售货员。售货员希望用张数最少的钱币找给顾客。（共有六种币值：50，20，10，5，2，1 ）



2.2数组使信息有序化

- 算法设计：
  - 为了能达到找给顾客钱的张数最少，先尽量多地取大面额的币种，由大面额到小面额币种逐渐进行。
  - 六种面额不是等差数列，为了能构造出循环不变式，将六种币值存储在数组A。为了能达到先尽量多地找大面额的币种，六种币值应该由大到小存储。
  - 为统计六种面额的数量，设置一个有六个元素的累加器数组S。

2.2数组使信息有序化

- 算法实现：

```
main( )
{  int i,j,x,y,z,m,a[6]={50,20,10,5,2,1},s[6]={0};
   printf("输入商品价格：");
   scanf("%d",&x);
   printf("输入实际支付的金额：");
   scanf("%d",&y);
   z=y-x;
   for(i=0;i<6;i++)
   {    m=z/a[i];
        s[i]=s[i]+m;
        z=z-m*a[i];
   }
   printf("\n商品价格%d，支付%d，找回：\n",x,y);
   for(i=0;i<6;i++)
        if(s[i]!=0)
              printf("%d元%d张\n",a[i],s[i]);
}
```

2.2数组使信息有序化

- 例： 将编号“翻译”成英文。例如35706“翻译”成three-five-seven-zero-six 。

- 算法设计：
  - 将英文的“zero——nine”存储在数组中，对应下标为0——9。这样无数值规律可循的单词，通过下标就可以方便存取、访问了。
  - 通过求余、取整运算，可以取到编号的各个位数字。用这个数字作下标，正好能找到对应的英文数字。

2.2数组使信息有序化

- 算法实现：

```
char english[10][6] =
   {"zero","one","two","three","four","five","six","seven","eight
   ","nine"};
void f1(int n)
{  if(n>0)
   {    f1(n/10);
        printf("%s-",english[n%10]);
   }
}
int main( )
{  int num1;
   printf("输入编号：");
   scanf("%d",&num1);
   printf("%d翻译为：",num1);
   f1(num1);
   return 0;
}
```

2.3数组记录状态信息

- 有的问题会限定在现有数据中，每个数据只能被使用一次，怎么样表示一个数据“使用过”还是没有“使用过”？

2.3数组记录状态信息

- 例：12个小朋友手拉手站成一个圆圈，从某一个小朋友开始报数，报到7的那个小朋友退到圈外，然后他的下一位重新报“1”。这样继续下去，直到最后只剩下一个小朋友，求解这个小朋友原来站在什么位置上呢？

2.3数组记录状态信息

- 算法设计：
  - 使用12个元素的数组，记录12个小朋友的状态，开始时将12个元素的数组值均赋为1，表示大家都在圈内。累加到7时，该元素所代表的小朋友退到圈外，将相应元素的值改赋为0，模拟已出圈外的状态。
  - 为了算法的通用性，对游戏的总人数、报数的起点、退出圈外的报数点任意输入。

2.3数组记录状态信息

- 算法实现：

```
main()
{
   int a[100],i,k,p,m,n,x;
   printf("输入游戏总人数：");
   scanf("%d",&n);
   printf("输入游戏开始小朋友的编号：");
   scanf("%d",&k);
   printf("输入退出圈外的报数号：");
   scanf("%d",&m);
   for(i=1;i<=n;i++)  //所有圈内小朋友标识为1
        a[i]=1;
   p=0;
   k--;
```

2.3数组记录状态信息

```
while(p<n-1)    //只要圈内不止一个小朋友则循环
{
     x=0;
     while(x<m)    //还没数到报数号则循环
     {
            k=k+1;  //从k编号开始报数
            if(k>n)     //如果报数到最后一个小朋友
                  k=1;  //又从1号小朋友开始
            x=x+a[k];    //报数人数叠加
     }
     printf("退出的小朋友编号为：%d\n",k);
     a[k]=0;    //退出圈的小朋友标识为0
     p=p+1;  //已退出圈外的人数叠加
}
for( i=1;i<=n;i++)
     if (a[i]==1)
            printf("最后剩下的小朋友编号为：%d\n",i);
}
```