预处理命令与习题选讲

Wang Houfeng
EECS,PKU
wanghf@pku.edu.cn

内 容

- 宏定义与宏展开
- 包含与条件预编译
- 习题选讲

预编译简介

- 预处理命令是ANSI C 规定的，用于改进程序设计环境、提高程序设计效率，不属于C 本身的一部分。
- 由于预处理命令不是 C语言的一部分，在编译之前，必须对其预处理，因此也称为预处理命令。
- 有三类预处理命令：
  - 宏定义
  - 文件包含
  - 条件编译

预处理命令的特点

- 形式上，以 # 引导，如：
  - 宏定义， #define
  - 包含， #include
  - 条件编译，#if
- 出现位置：在通常情况下，位于函数外面

不带参数的宏定义

宏体可缺省,表示宏名
定义过或取消宏体

- 一般形式： #define 宏名 [宏体]
- 功能:用指定标识符(宏名)代替字符序列(宏体)

由宏名来表示宏体

如
```
#define   YES   1
#define   NO    0
#define   PI    3.1415926
#define   OUT   printf("Hello,World");
```

宏体： 可以是常数、表达式、格式串等

使用宏与宏展开

宏定义

```
#define   YES   1
#define    NO    0
#define    PI    3.1415926
#define    OUT    printf("Hello,World");
```

宏使用

```
if(x==YES)          printf("correct!\n");
else   if (x==NO)    printf("error!\n");
```

宏展开：用宏体取代宏名（简单的符号串替换）

```
if(x==1)          printf("correct!\n");
else   if (x==0)    printf("error!\n");
```

宏展开过程由预处理程序完成（在编译之前）

宏定义中应使用必要的括号 ( )

例 #define WIDTH 80
#define LENGTH WIDTH+40
var=LENGTH*2;
宏展开： var= 80+40 *2;

简单字符串替换

例 #define WIDTH 80
#define LENGTH (WIDTH+40)
var=LENGTH*2;
宏展开： var= (80+40)* 2;

宏定义的位置与作用域

- 定义位置：一般在函数外面
- 作用域：从定义命令到文件结束
- #undef可终止宏名作用域

格式：　#undef　宏名

```
例  #define   YES     1
    main()
    { ……..
    }
    #undef   YES
    #define   YES   0
    max()
    {……..
    }
```

YES原作用域

YES新作用域

嵌套与递归

宏定义不能递归定义

例 #define MAX MAX+10 (×)

宏定义可嵌套（引用已定义的宏）

```
例 #define   WIDTH   80
    #define   LENGTH   WIDTH+40
    var=LENGTH*2;
宏展开：var= 80+40 *2;
```

宏展开的进一步说明

- 引号中的内容与宏名相同不作宏扩展

```
例  #define  PI   3.14159
    printf("2*PI=%f\n",PI*2);
宏展开：printf("2*PI=%f\n",3.14159*2);
```

- 宏定义是用**宏名**来表示一个**字符串**，在**宏展开**时，以**该字符串取代宏名**，这仅仅是一种简单的代换，预处理程序不对它作任何检查。
- 习惯上宏名用**大写字母**表示，以便于与变量区别。但也允许用小写字母。

宏展开的进一步说明（续）

- 宏定义不是语句，不要用分号结束，否则，会作为宏体的一部分进行转换

例 #define PI 3.14159;
aera= PI*r*r;

宏展开：area = 3.14159;*r*r;

例子：“输出格式”作宏定义

```
#include <stdio.h>
#define P printf
#define D "%d\n"
#define F "%f\n"
main(){
   int a=5, c=8, e=11;
   float b=3.8, d=9.7, f=21.08;
   P(D F,a,b); // => printf("%d\n""%f\n",)
   P(D F,c,d);
   P(D F,e,f);
}
```

带参数宏定义

- 宏名带有参数，定义部分的参数称为形参
- 一般形式：　#define　宏名(参数表)　　宏体

不能加空格

例 #define　S　(r)　PI*r*r

相当于定义了不带参数的宏S，代表字符串 “(r)　PI*r*r”

例　　#define　S(a,b)　a*b

……

area=S(3,2);

宏展开：　　area=3*2;

- 宏调用时的参数称为实参
- 宏展开：形参用实参替换，其它字符保留

参数与括号

例 #define POWER(x) x*x

x=4; y=6;

z=POWER(x+y);

宏展开： z=**x+y*x+y**;

一般写成： #define POWER(x) ((x)*(x))

宏展开： z=(**(x+y)*(x+y)**);

一个例子

```
#define MAX(a,b) (a>b)?a:b
main()
{
  int x,y,max;
  printf("input two numbers:    ");
  scanf("%d%d",&x,&y);
  max=MAX(x,y);
  printf("max=%d\n",max);
}
```

参数说明

- 在带参宏定义中，形式参数不分配内存单元，因此不必作类型定义；
- 宏调用中的实参有具体的值。要用它们去代换形参，因此必须作类型说明；
- 参数传递只是符号代换，不存在值传递的问题；
- 宏定义中的形参是标识符，而宏调用中的实参可以是表达式。如，前面的POWER(x)

带参数宏与函数

	带参宏	函数
处理时间	预编译时	程序运行时
参数类型	无类型问题	需给出形参类型
处理过程	不分配内存 简单的字符置换	分配内存 先求实参值,再代入形参
程序长度	变长	不变
运行速度	不占额外运行时间	调用和返回另占时间

内 容

- 宏定义与宏展开
- 包含与条件预编译
- 习题选讲

文件包含

- 功能：一个源文件可将另一个源文件的内容全部包含进来
- 一般形式：　#include "文件名"

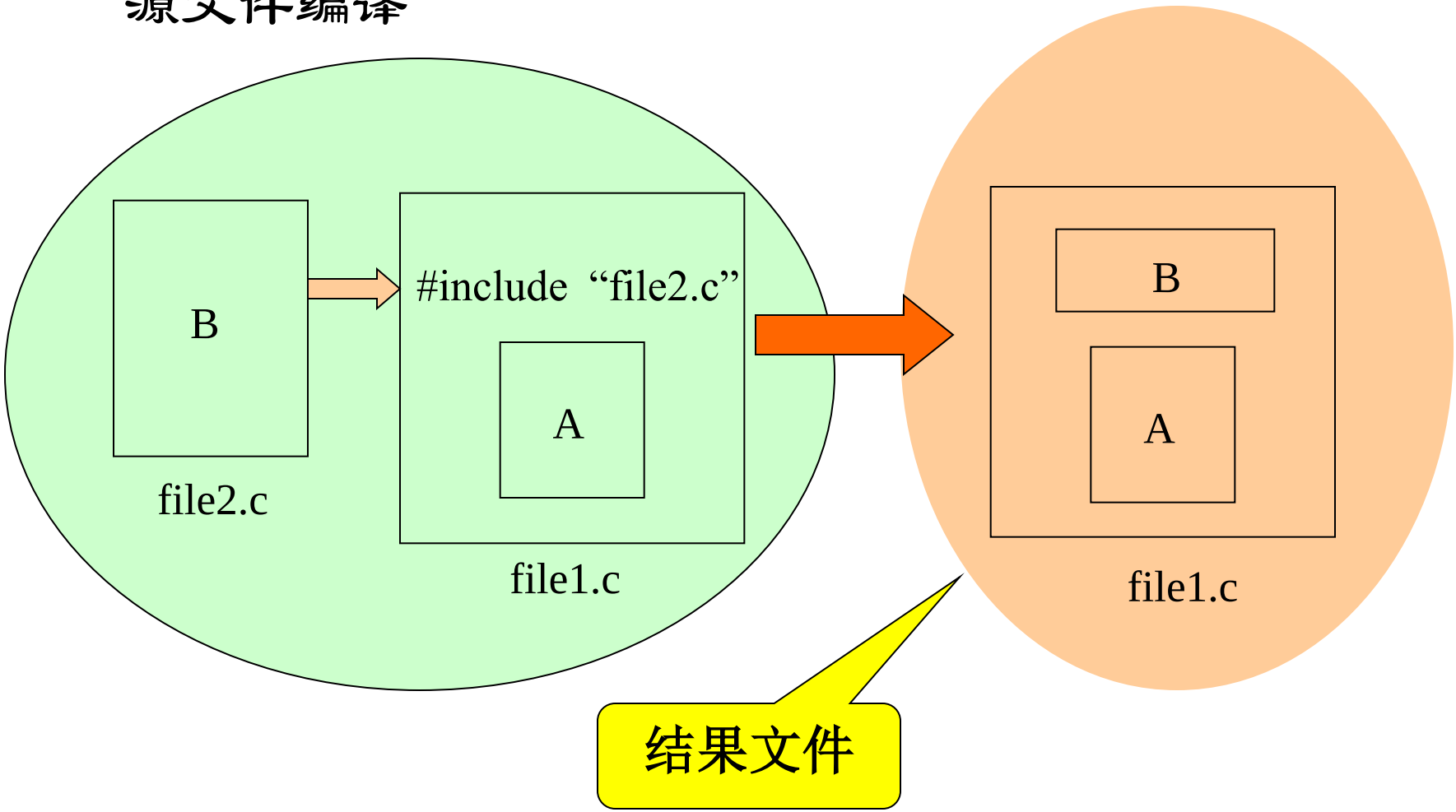
　　或　#include 〈文件名〉

〈〉　直接按标准目录搜索

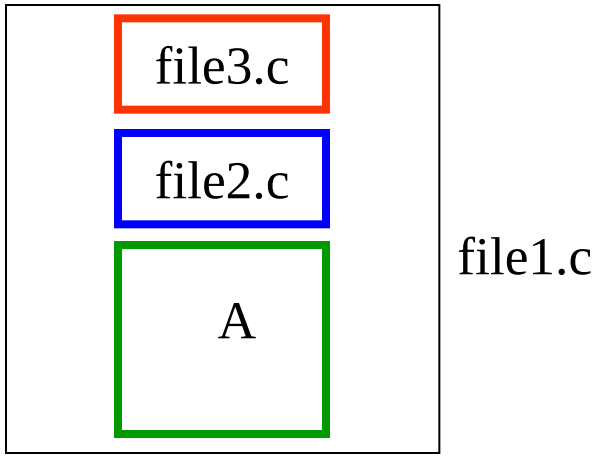
""先在当前目录搜索，再搜索标准目录

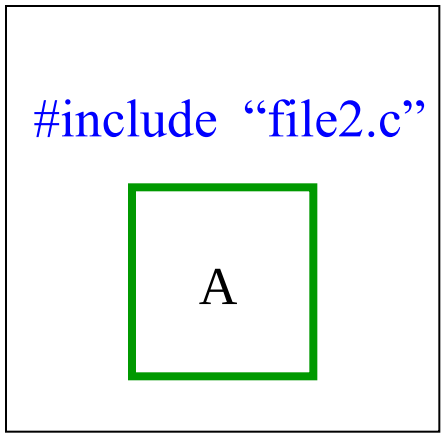
可指定路径

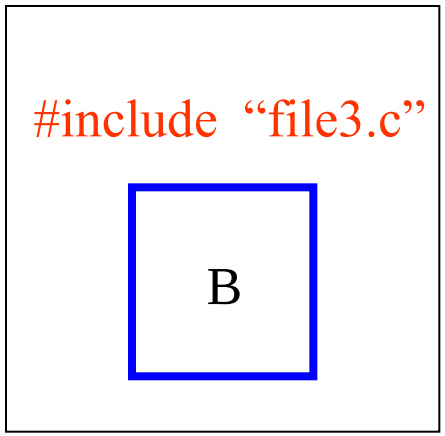
文件包含的处理过程：预编译时，用被包含文件的内容取代该预处理命令，再对“包含”后的文件作一个源文件编译

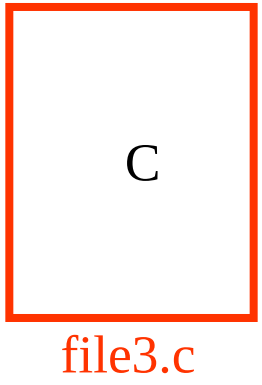


– 被包含文件内容
  - 源文件(*.c)
  - 头文件(*.h)









文件包含举例

```
/*  powers.h  */
#define  sqr(x)     ((x)*(x))
#define  cube(x)   ((x)*(x)*(x))
#define  quad(x)   ((x)*(x)*(x)*(x))
```

```
#include <stdio.h>
#include "d:\macro\powers.h"
#define  MAX_POWER 10
void main()
{   int n;
   printf("number\t exp2\t exp3\t exp4\n");
   printf("----\t----\t-----\t------\n");
   for(n=1;n<=MAX_POWER;n++)
     printf("%2d\t %3d\t %4d\t %5d\n",n,sqr(n),cube(n),quad(n));
}
```

条件编译（一般了解）

- 指定编译的条件，对满足条件的进行编译。
- 三种格式：
  - #Ifdef ~ [#else ~] #endif
  - #ifndef ~ [#else ~] #endif
  - #if ~ [#else ~] endif

内 容

- 宏定义与宏展开
- 包含与条件预编译
- **习题选讲**

题目：最小正整数因子

- 题目：任给两个正整数N、M，求一个最小的正整数a，使得a和(M-a)都是N的因子。如果没有满足条件的正整数a，则在对应行输出-1
-

问题分析与求解

- **满足条件**: (N%a)==0 && (N%(M-a))==0
- a从1 到(M/2) 逐个测试.

```
a=1;
p=M/2;
while((a<=p) && !((N%a)==0 && (N%(M-a))))
    a++;
if(a>p) printf("-1\n");
else printf("%d\n",a);
```

两倍

- 题目：给定2到15个不同的正整数，你的任务是计算这些数里面有多少个数对满足：数对中一个数是另一个数的两倍。

  比如给定1 4 3 2 9 7 18 22，得到的答案是3，因为2是1的两倍，4是2个两倍，18是9的两倍。

问题分析

- **问题（仅考虑1组数的情况）：**
  - 输入（以0结尾，判断相对简单，不分析）
  - 检查：关键问题
    - 假设有一个数组 a[]，共有n个元素；
    - 有一个计数器count,初始化为0
    - 计算每个元素的2倍为m，判断m是否在a中出现
      - 如果有，则count++；

程序实现

- 请大家自己思考

约瑟夫问题

总数：N=12, 每次报到 p=3 时出去

关键点：循环，如何实现循环？

问题分析与数据设置

- 假设总数为 (N<300),
- 引入数组:josephus[300]
- 数组的下标 j 对应编号为(j+1)的猴子，如果编号为(j+1)的猴子没有出队列，则josephus[j]=1;否则，josephus[j]=0;
- 初始化:

  for(j=0;j<N;++j) josephus[j]=1;// 使josephus[0~N-1]=1;
- 如何维持圆圈循环报数？

  下标 j=(j+1)%N
- 使 (N-1) 只猴子出队列

程序实现

```
int joseph(int, N, int p)
{
   char josephus[300];
   char count,j,k;
   for(j=0;j<N;j++)
         josephus[j]=1;
   j=-1;
   for(k=1;k<N;++k)
   {    count=0; //准备报数
        do
        {j=(j+1)%N;
         count +=josephus[j]
         }while(count<p)
        count=0;
        josephus[j]=0;//出队列
   }// for 结束，(N-1)个猴子出队
   j=0;
   while(josephus[j]!=1) j++;
   return(j+1);
}
```

删除单词后缀

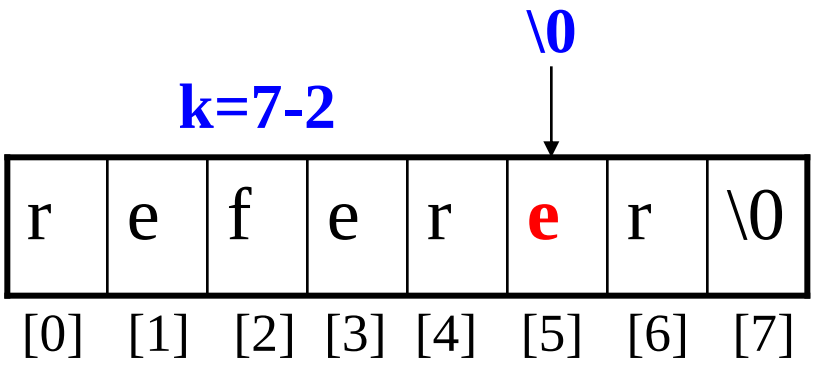
- 题目：
  - 给一组各分别以er、ly和ing结尾的单词， 请删除每个单词的结尾的er、ly或ing， 然后按原顺序输出删除后缀后的单词。
- 要求：
  - 输入的第一行是一个整数n（n≤50），表示后面有n个单词。其后每行一个单词（单词中间没有空格）。
  - 按原顺序输出删除后缀后的单词。

分析问题

- 两个子问题需要解决：
  - 删除后缀
    - 用一个函数实现删除长度为 len 的后缀：
    - delsuffix(char source[], int len) //其中，len 为后缀长度。
  - 判断后缀
    - 用一个函数检查串 source 中是否含有后缀 suffix：
    - int chksuffix(char source[], int len1, char suffix[], int len2)
      - 功能：先比较看是否是后缀，
      - 若不是后缀，则返回0；
      - 若是，返回1；

如何删除后缀

```
delsuffix(char source[], int len )
{  int k;
   k=strlen(source)-len;
   source[k]='\0';
}
```



判断

```
int chksuffix(char source[], int len1, char suffix[], int len2)
{   int m=len1, n=len2;
    if(m<n) return 0;
    else
    {
        while((n>=0)&&(suffix[n]==source[m]))
        { n--; m--;}   // 从后向前逐个比较。
        if(n>=0) return 0;
        else return 1;
    }
}
```

主程序

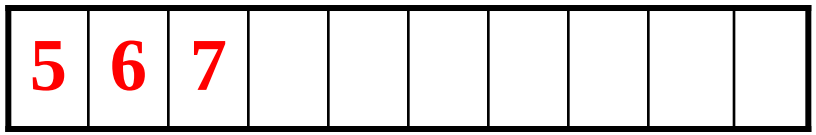
```
#include <stdio.h>
#include <string.h>
main()
{   int count, k, m, len, tag, suflen[3]={2,2,3};
    char s[30],suffix[3][4]={“er”, “ly”, “ing”};
    scanf(“%d”, &count);
    for(k=0; k<count; ++k)
    {       gets(s);
            len=strlen(s);
            m=0; tag=0;
            while((m<3) &&(tag=chksuffix(s,len,suffix[m],suflen[m])==0)) m++;
            if(m<3) delsuffix(s, suflen[m]);
            printf(“%s\n”,s);
    }
}
```

删除重复元素保留1个

- 题目：
  - 输入n 个整型数，把其中重复出现的元素删去（如出现三个5，则只保留第一次出现的5，删去后两个5），将剩余的元素顺次输出。整数个数 n 不大于300；
- 两种方法：
  - 边输入，边比较，边删除；
  - 一次性输入，然后再作删除处理；

第1种方法：边输入边删除

- 输入：
  5 6 7 5 8 5 7 9 8 6
- 读入与存放：
  1. 读入值 d,
  2. 与数组的前面n个元素 array[0..n-1] 逐一比较，如果没有相同元素，则放入array[n]中，且n++; 否则，不做任何操作；
  3. 转步骤1

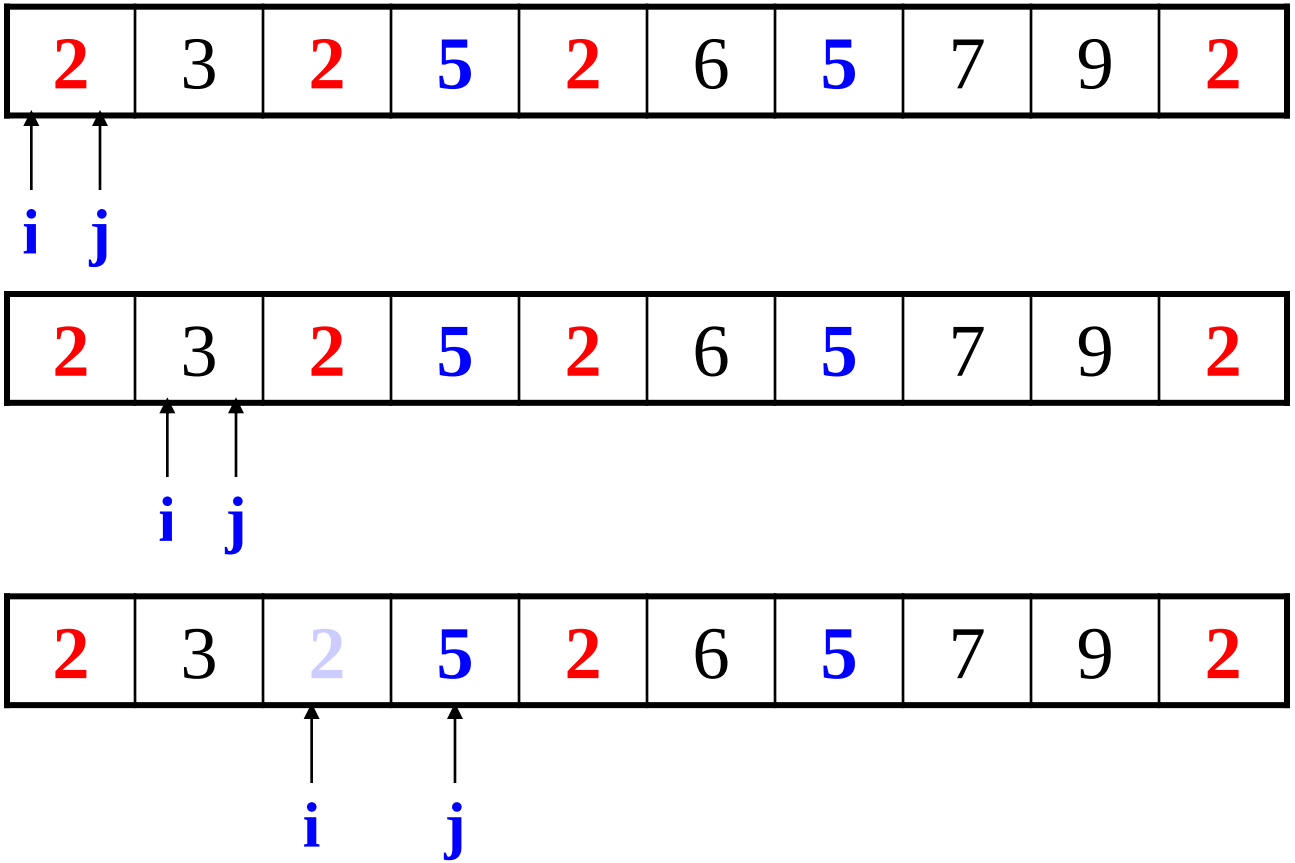


程序

```
#include <iostream>
using  namespace std;
main()
{
    int  k, j,  n=0, m, d, array[301];
    cin>>m;
    for(k=0; k<m;++k)
    {
        cin>>d;
        j=0;
        while((j<n)&&(array[j]!=d) ++j;
        if(j==n) arrar[n++]=d;
    }
// 输出部分
    cout<<array[0]);
    k=1;
    while(k<n)
        cout<< ',' <<array[k++];
}
```

第2种方法：输入完后再删除

初始：
i=j=0



j 所指的元素与 i 的前面每个元素逐一比较：

(1) 发现相同元素，j++：继续后移

(2) 没有发现相同元素， j 指的元素移至 i 处, 且i++， j++：同时后移

程序实现

```
int del(int a[], int num)
{   int  i=0, j=0, k, temp;
    while(j<num)   //num 表示当前元素个数;
    {       k=0, temp=a[j]   //k用于扫描 i 前面的元素
            while((k<i)&&(a[k]!= temp))  k++;  // 在 i 前面比较是否有相同
    元素。
            if (k==i)   // 不相等，考虑前移
            {
                if(i!=j)   // 如果没有指向相同位置，则元素前移;
                   a[i] = temp;
                i++;     //  i 向后移动一个位置
            }
            j++;
    }
    return i;
}
```

两个大正整数相减

- 输入描述：char a[101],b[101];
- 输出结果：char c[101];
- 处理过程：设计函数：
  - **大小比较（可以放在main中）**
    - **如何比较大小（自己思考）？**
  - 函数-1：字符串右对齐（或反向处理）
  - 函数-2：两个数相减（**大数减小数**）

两个大整数相减（续）

- 函数-1：字符串右对齐
  - align(char a[],int len)
  - a[len]='\0'; jump=len-strlen(a)
  - for(m=len-jump, m>0,--m)
    - a[m+jump]=a[m]
  - for(m=0, m<jump,++m)
    - A[m]='0';

也可以逆序后再相减

两个大整数相减（续）

- 函数-2：两个数相减
  - subtract(char a[],char b[],char c[])
  - a-b 的结果放在 c 中。
  - 引入借位变量 carry, 初始值为0；
  - 从低位向高位逐位进行，实现两种操作：
    - 若（carry+b[m]>a[m]）表示要借位(c[m]=a[m]–(b[m] +carry) +10 +’0’; carry=1;)；
    - 否则，不必借位（c[m]=a[m]-(b[m]+carry)+’0’; carry=0; ）

另一种思路

- 不对齐，先反向
- 减法（从左向右减）
- 结果反向

思考题

- 带符号的两个数相减
  - 减法？
  - 加法？