一、基本概念

第 4 章 数组与字符串

数组的基本概念：

数组是具有一定顺序关系的若干相同类型数据的集合体。

数组属于复合类型（不是基本数据类型），是不能作为一个整体进行访问和处理的，只能按元素进行个别的访问和处理。

数组元素在内存中是连续存放的，每个元素占用的存储空间相同，根据数组第一个元素的存储单元的起始地址可以计算出任意一个元素的存储单元的起始地址。

数组必须先定义，后使用；

数组名字是数组首元素的内存地址，即数组首地址。

数组下标的范围：从 0 到 n-1。引用数组元素时，其数组下标允许是整型常量或整型表达式。

一维数组定义与初始化：

可以只给一部分元素赋初值（至少一个），当没有为数组中每个元素提供初始值时，C++将未初始化的元素赋值为默认值（0）。例如：int a[5]={0,1,2}; 则 a[3]，a[4]的值为 0

如果没有赋任何一个初值， C++不能自动初始化元素，数组元素将包含垃圾数据。

在对全部数组元素赋初值时，可以不指定数组长度。例如，int a[ ]={4,3,6,7,9}；定义了一个有 5 个元素的数组。

声明数组时，也可在前面加上关键字 static，将数组声明为静态的。例如：static int a[5];若未对静态数组进行初始化，则数组的所有元素将自动获取初始值 0。

二维数组定义与初始化：

可以分行给二维数组赋初值,例如：int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

可以将所有数据写在一个{ }内，按顺序赋值，例如：int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

可以对部分元素赋初值，例如：int a[3][4]={{0,0,1},{0,6},{0,0,11}};此时，其余的元素将获得默认值 0。

如果对全部元素都赋初值，则定义数组时第一维的大小可以省略，但第二维的大小不能省略。例如：int a[][4]={1,2,3,4,5,6,7,8,9,10,11,12};

类似于下面的数组定义语句，会判断正误。

```
int x[ ][3]={{0},{1},{1,2,3}};      // 正确
int x[ ][3]={1,2,3,4};              // 正确
int x[ ][3]={1};                    // 正确
int x[4][ ]={{1,2,3},{4,5,6},{3,2,1},{6,5,4}};    //错误
int x[2][3]={{1,2},{3,4},{5,6}};        //错误
int x[2][3]={1,2,3,4};              // 正确
```

引用二维数组时，需要包含数组名、行下标和列下标。

二维数组的每一行相当于一个一维数组，因此二维数组可以看成是由一维数组构成的数组。

二维数组在内存中存放时是按行存放的。据此，应会计算 a[i][j]是二维数组中的第几个元素。例如：若二维数组 a 有 m 列，a[0][0]是二维数组的第 1 个元素，则 a[i][j]在数组中是第（i*m+j+1）个。若有二维数组 a[3][5]，则第 8 个元素是：a[1][2]。

一维字符数组的基本概念：

可以用一个一维字符数组表示若干个字符，也可以表示一个字符串。例：

```
char c1[ ]={'C', 'h', 'i', 'n', 'a'};
char c2[ ]="China";
```

char c3[ ]={‘C’, ‘h’, ‘i’, ‘n’, ‘a’, ‘\0’};

char s4[8]={'C', 'h', 'i', 'n', 'a'};　　// 初始化部分数据，其余元素被自动赋值为‘ \0’

c1 是若干个字符，只能以元素为单位来参加运算；c2、c3、c4 是字符串，既能以元素为单位参加运算，也可以作为一个整体(字符串)参加运算。

string 类的概念及基本使用：输入，输出，求长度，比较大小等

	字符数组	string 类
定义	char str[20];	string str;
输入(不接收空格)	cin>>str;	
输出	cout<<str;	
取出某个字符	str[i]	
输入(接收空格,遇回车结束)	cin.getline(str, 20);	getline(cin,str);
求字符串的长度	strlen(str)	str.size( ) 或 str.length( )

遍历每个字符，判断字符串是否处理完毕	for(i=0; str[i]!='\0'; i++)
	或
	for(i=0; i<strlen(str); i++)
		for(i=0; i<str.size( ); i++)
		注：不能用 str[i]!=‘\0’来判断字符串处理完毕。

比较两个字符串是否相等	strcmp(str1,str2)==0	str1==str2

第 5 章 指针

C++程序中，从内存单元中存取数据的方法有两种：直接访问方式和间接访问方式。

一个变量在内存空间中占用的地址就称为该变量的“指针”。指针变量用于存放内存单元的地址，即存放地址的变量就是指针变量。

指针变量的类型，是指针所指向的变量的类型，而不是自身的类型。指针变量的值是某个变量的内存地址。

取地址运算符&，是一个一元运算符，用来得到一个对象的地址。

用变量地址作为初值时，该变量必须在指针初始化之前定义。例：　　int a=3; int *pa=&a;

不能把常量或表达式的地址赋给指针变量。如：int *p;　p=&67;　p=&(i+5);　是非法的

不能将一个整数赋给指针变量，但可以赋整数值 0，表示该指针是空指针。0 是可以直接赋给指针变量的唯一整数值。　　int *p; p=0; //p 为空指针，不指向任何地址

允许声明指向 void 类型的指针。该指针可以被赋予任何类型对象的地址。

使用指针变量与使用一般变量一样，一定要先定义后使用，使用前，指针变量一定要有明确的指向。

指向一维数组的指针：指针加 1、减 1 运算表示指向下一个或前一个数据。指针 p 加上或减去 n ，其意义是指针当前指向位置的后方或前方第 n 个数据的地址。这种运算的结果值取决于指针指向的数据类型。

两个指针变量指向同一个数组中的元素时，其关系运算的结果表明了这两个指针变量所指向的数组元素的先后关系。

有了指针的概念后，在操作数组时，就可以用如下几种方法：（假设已有： int a[10]; int *p=a;）

- 相对地址的表示方法

  for(i=0; i<10; i++)　cout<< *(a+i) ;　// 或 cout<<*(p+i);

  特别说明：不能写 a++，因为 a 是数组首地址，是常量。
- 移动指针的方法

  for(p=a; p<a+10; p++)　cout<<*p;

  注意：指针 p 进行了移动，下次使用时要重新让 p 指向数组首地址；
- 指针变量的下标方式

  for(p=a,i=0; i<10; i++)　cout<<p[i];　即：指针与数组具有互换性

第 6 章 函数

在 C++中，实参与形参结合方式有 3 种：

- 值传递（值调用）：调用时实参仅将其值赋给了形参，在函数中对形参值的任何修改都不会影响到实参的值，传递时是传递参数值，即“单向传递”。
- 引用传递（引用调用）：

引用(&)是一种特殊类型的变量，某变量的引用可以看成是该变量的一个别名，引用传递的是变量的地址。

引用不是值，不占存储空间，声明引用时，目标的存储状态不会改变，所以引用只有声明。声明一个引用时，必须同时对它进行初始化，使它指向一个已存在的对象。一旦一个引用被初始化后，就不能改为指向其它的对象。

把引用变量作为函数形参，当发生函数调用时，被调用函数的形参变量就会成为对应的主调用函数实参变量的别名，这时实参和形参两个变量对应一个存储单元，也就是说，当希望在执行被调用函数时能够改变主调用函数中的某个变量值时，可在与该变量对应的被调用函数的形参变量名前加上引用符号&。

- 地址传递 (指针传递)（地址调用）

在地址传递中，实参将自己的地址（即内存空间中的位置）传递给形参。

将指针变量或数组名作为函数形参，当发生函数调用时，系统自动将主调函数实参的地址传递给被调用函数的形参，使形参和实参指向相同的内存单元。也即：在被调用函数中对形参的修改相当于在修改对应的实参，所以地址传递是一种“双向传递”。

递归函数即自调用函数，在函数体内部直接或间接地自己调用自己。

递归调用有直接递归调用和间接递归调用两种形式。直接递归即在函数中出现调用函数本身。间接递归调用是指函数中调用了其他函数，而在其他函数却又调用了本函数。

递归条件：

1)必须具有递归结束条件及结束时的值，即必须具有递归的“**出口**”;

2)求解的问题要能用递归形式表示，即有“**递归公式**”。

二、经典程序

第 4 章 数组与字符串

（一）一维数组的基本操作

求和、求平均值、求最大值最小值、统计奇数（或偶数）个数

【案例 4.1】某门课程 n 个学生成绩统计（求平均值）。 第 4 章 PPT18-20

（二）**排序、查找、插入、删除**问题

【案例 4.12】 排序的问题（1）- **选择排序**。以下两种方法都可以。

方法一：先找最小值 a[min]，然后再和 a[i]交换。在一轮排序中只有一次交换。	方法二：发现一个比 a[i]稍小的就交换，在一轮排序中会有多次交换

for(i=0;i<n-1;i++) // n-1 轮排序
{ min = i; //记录最小值的下标
for(j=i+1; j<n; j++)
if(a[j] < a[min]) min = j;
if (min != i)
{ t=a[i]; a[i]=a[min]; a[min]=t; }
}
	for(i=0;i<n-1;i++) // n-1 轮排序
	for(j=i+1; j<n; j++)
	if(a[i]>a[j])
	{
	t=a[i]; a[i]=a[j]; a[j]=t;
	}

【案例 4.13】 排序的问题（2）- **起泡排序(冒泡排序)**。

```
for(i=0; i<n-1; i++)       // n-1 轮排序
    for(j=0; j<n-i-1; j++)
        if(a[j]>a[j+1]) //  两两比较，不合顺序就交换
            {   t=a[j];     a[j]=a[j+1];     a[j+1]=t;   }
```

【案例 4.14】 在升序排列的数组中进行**折半查找**。

```
int a[N],i, top=0, bot=N-1, mid, x;
while(top<=bot)
{
        mid=(top+bot)/2;   //确定中间位置
        if(x==a[mid])       break;
        else if(x>a[mid])      top=mid+1;     //调整 top 的值，到后半部分查找
        else          bot=mid-1;     //调整 bot 的值，到前半部分查找
}
    if(top<=bot)
            cout<<x<<"在第"<<mid+1<<"个位置。"<<endl;
    else
            cout<<"该数不存在。"<<endl;
```

【案例 4.7】 **插入数据**问题。插入一个数并要求仍然保持由小到大的顺序。

```
const int n=10;
int a[n+1], p=0, x, i;
while(x>a[p]&&p<n)
        p++;                          //找到 x 应插入的正确位置
 for(i=n-1; i>=p; i--)
            a[i+1]=a[i];          //将 a[p]~a[n-1]后移
   a[p]=x;       //x 插入正确位置
   for(i=0; i<=n; i++)
       cout<<setw(4)<<a[i];
```

拓展：**在有序数组中插入多个元素**，即：在上述程序的外面再加一层循环，每循环一次，插入一个元素，注意，每一次循环开始前，都要给 p 赋值为 0。

【案例 4.8】 **删除指定数据**问题。思想：将后面的元素往前移，覆盖掉想要删除的元素。

```
const int N = 10;
int ss[N], n = N, i, a, pos, flag = 0;
cin>>a;            // a 是想要删除的数据
for (i = 0; i <N; i++)
        if (ss[i] == a)
                { pos = i; flag = 1; break;   }     //  定位
if (flag == 0)
    cout << "无此数！" << endl;
else
{       for (i = pos; i < N-1; i++)     ss[i] = ss[i + 1];     //  后面的数向前递补
        n = n - 1;     //  有效长度减 1
        for (i = 0; i < n; i++)
                cout << "     " << ss[i];
         cout << endl;
 }
```

拓展：**删除数组中小于（或大于）平均值的元素**：实验 9-2

删除无序数组中的最大值（或最小值）（先确定最大值的位置，再将后面的元素往前移，覆盖掉最大值）。

【案例 4.9】 删除重复数据问题（使用两个数组）

```
int cf[N], wcf[N], i,j, ib=0;
for(i=0; i<N; i++)      cin>>cf[i];
wcf[0]=cf[0];
for(i=1;i<N;i++)
{    // 下面两段代码任选一段
```

for(j=0; j<i; j++)
if(cf[i]==cf[j])
break;
if (j>=i)
	for(j=0; j<=ib; j++)
	if(cf[i]==wcf[j])
	break;
	if (j>ib)

```
        wcf[++ib]=cf[i];
}
cout<<"删除重复数据后："<<endl;
for(i=0; i<=ib; i++)
        cout<<setw(4)<<wcf[i];
```

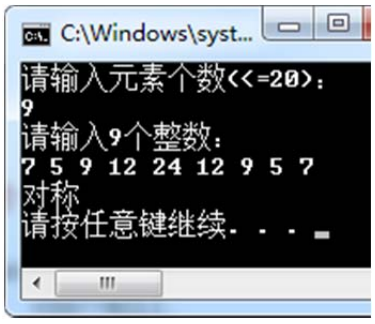
拓展：**只使用一个数组来实现删除重复数据**。

```
const int N = 10;
 int a[N], i, j, pos = 1;
 for (i = 0; i < N; i++)          cin >> a[i];
 for (i = 1; i < N; i++)
 {       for (j = 0; j < pos; j++)
                 if (a[i] == a[j])        break;
          if (j >= pos)
                  a[pos++] = a[i];
  }
  for (i = 0; i < pos; i++)       cout << a[i] << "    ";
```

（三）数组中元素换位、平移问题

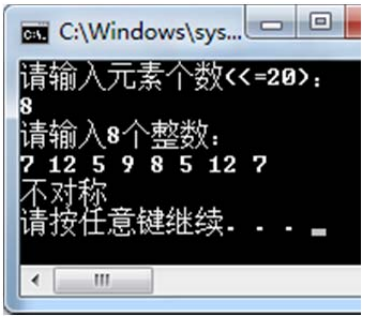
【案例 4.10】 将一个数据序列**逆序存放**。

```
for(i=0; i<N/2; i++)
{   t=db[i];   db[i]=db[N-1-i];   db[N-1-i]=t;   }
```



拓展：判断**一个数组中的数据是否对称**。即：第 1 个和最后 1 个元素相等，第 2 个和倒数第 2 个元素相等……

与前面的“逆序存放”的程序有一定的相似之处。




```
void main( )
{       int a[20],i,j,n, flag=1;   //flag ：数组是否对称的标记
        cout<<"请输入元素个数(<=20)："<<endl;
        cin>>n;
        cout<<"请输入"<<n<<"个整数："<<endl;
        for(i=0;i<n;i++)   cin>>a[i];
        for(i=0,j=n-1; i<j; i++,j--)
              if(a[i]!=a[j]) { flag=0;break; }
```

```
        if(flag==0) //   也可以不要 flag 变量，直接将本行改成  if(i<j)
            cout<<"不对称"<<endl;
        else
            cout<<"对称"<<endl;
}
```

将数组中的各元素**循环左移**（或右移）m 位：实验 12 第 4 题。

一组数据中，将所有大于平均值的放在数组前部，所有小于平均值的放在后部。见实验 8 第 1 题
一组数据中，将所有偶数放前部，所有奇数放在后部。见“第 4 章课后作业”

（四）自动生成数组元素

【案例 4.15】　兔子繁殖问题。**斐波那契数列**。

```
        int i,fb[13]={1,1};
        for(i=2;i<13;i++)
            fb[i]=fb[i-2]+fb[i-1];
```

【案例 4.19】 十进制与二、八、十六**进制转换**的问题。

```
        int m, j, number, trans, jzzh[32];
        char ch;
        cout<<"请输入待转换的十进制数：";
        cin>>number;
        for(int i=0;i<=2;i++)
        {       m=number;
                if(i==0) trans=2;
                if(i==1) trans=8;
                if(i==2) trans=16;
            j=0;   // 数组下标，每次进入内循环前都要初始化
            while (m!=0)          //   除基数取余数
            {   jzzh[j++]=m%trans;      m=m/trans;   }
            cout<<"转换为"<<trans<<"进制,"<<number<<"是：";
            j--;
            for(   ; j>=0; j--)     // 数组逆序输出
            {
                if( jzzh[j]>=0&&jzzh[j]<=9)   ch=jzzh[j]+'0';
                else     ch=jzzh[j]-10+'A';
                cout<<ch;
            }
```

【案例 4.16】　自动生成一个分数序列。如自动生成一个分数序列：2/1，3/2，5/3，8/5，13/8...　教材 153 页

（五）求两个集合的**交集、并集、差集**：　第 5 章 PPT

```
// 求交集：
    int cLen=0;
    for(i=0; i<aLen; i++)
        for(j=0; j<bLen; j++)
            if(a[i]==b[j])
            {       c[cLen++]=a[i];   //将 a、b 中相同元素写入 c,同时 c 的长度加 1
```

```
                    break;   //找到 a、b 中相同元素即退出本轮比较
                }
    cout<<"交集 c 的各个元素依次为:"<<endl;
    for (i=0; i<cLen; i++)
            cout<<setw(3)<<c[i];

// 求并集：
    for(i=0; i<n; i++)     c[i]=a[i];      // 先将数组 A 的所有元素都复制到并集 C 中
    k=n;
    for(i=0;i<m;i++)        // 遍历数组 B
    {    for(j=0;j<n;j++)        // j 是数组 A 的下标
            if(b[i]==a[j])     break;
        if(j>=n)      // 或 if (j==n)
            c[k++]=b[i];           // 等价于   { c[k]=b[i]; k++; }
     }
    【蓝色部分的代码，其实是在求差集 B-A。其实，只要掌握了求差集的方法，求并集也不成问题。】
      cout<<"并集是:"<<endl;
      for (i=0;i<k; i++)       cout<<c[i]<<"   ";

// 求差集 A-B
    for(i=0;i<n;i++)        // 遍历数组 A
    {       for(j=0;j<m;j++)
                if(a[i]==b[j]) break;
            if(j>=m)     // 或 if (j==m)
                c[k++]=a[i];
    }
    cout<<"A-B 的差集是："<<endl;
    for(i=0;i<k;i++)     cout<<c[i]<<"   ";
```

有序数组的有序合并：第 4 章 PPT（编程思路），第 4 章课后作业，实验 13 第 4 题。

```
    int ia = 0, ib = 0, ic = 0;
    while (ia<m && ib<n)
    {    if (a[ia]<b[ib])   c[ic++] = a[ia++];
        else          c[ic++] = b[ib++];
    }
    while (ib<n)
        c[ic++] = b[ib++];
    while (ia<m)
        c[ic++] = a[ia++];
```

（六）二维数组

【案例 4.2】m 门课程 n 个学生成绩统计（求二维数组每行的平均值）。第 4 章 PTT84-86

【案例 4.11】 求二维数组的最大值及其行、列下标。第 4 章 PTT87-90

```
    int xs[4][3],   max_row=0;   max_col=0;
    max_value=xs[0][0];
    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
```

```
            if(xs[i][j]>max_value)
                { max_value=xs[i][j]; max_row=i; max_col=j; }
```

拓展：**求二维数组每行（或每列）的平均值、最大值最小值**

【案例 4.17】 自动生成**杨辉三角形**。第 4 章 PTT91-94

```
    const int N=10;
    int yh[10][10], i, j;
    for(i=0;i<N;i++)    //初始化两条斜边上元素的值
    {       yh[i][0]=1;     yh[i][i]=1;          }
    for(i=2;i<N;i++)
        for(j=1;j<=i-1;j++)
            yh[i][j]=yh[i-1][j-1]+yh[i-1][j];
    cout<<"杨辉三角形:"<<endl;
    for(i=0;i<N;i++)
    {       for(j=0; j<=i; j++)
                cout<<setw(4)<<yh[i][j];
            cout<<endl; //换行输出下一行
    }
```

拓展：以右对齐格式输出杨辉三角形。

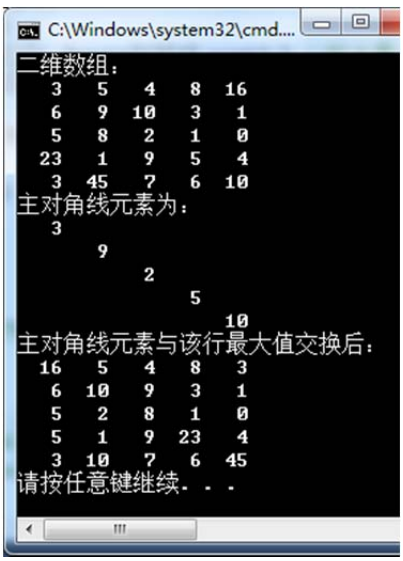
拓展：以等边三角形式输出杨辉三角形。见“第 4 章课后作业”

二维数组的**主对角线、次对角线元素**。 第 4 章 PTT95

假设定义了二维数组 int a[M][M]，则其主对角线元素可表示为 **a[i][i]**，次对角线元素可表示为 **a[i][M-1-i]**。

输出二维数组的主对角线，然后将主对角线元素与该行的最大值交换。

```
void main( )
{   const int M=5;
    int a[M][M]={{3,5,4,8,16},{6,9,10,3,1},{5,8,2,1,0},{23,1,9,5,4},{3,45,7,6,10}};
    int i,j,max,pos,t;
    cout<<"二维数组："<<endl;
    for(i=0;i<M;i++)
    {   for(j=0;j<M;j++) cout<<setw(4)<<a[i][j];
        cout<<endl;
    }
    cout<<"主对角线元素为："<<endl;
    for(i=0;i<M;i++)
    {   for(j=0;j<i;j++) cout<<setw(4)<<" ";      // 输出每行前面的空格
        cout<<setw(4)<<a[i][i]<<endl;             // 主对角线元素为 a[i][i]
    }
    for(i=0;i<M;i++)
    {   max=a[i][0];  pos=0;
        for(j=1;j<M;j++)
            if(max<a[i][j])  {  max=a[i][j];  pos=j;  }
                //找出每行的最大值及其列下标
        t=a[i][i];  a[i][i]=a[i][pos];  a[i][pos]=t;
                // 最大值与主对角线元素交换
    }
    cout<<"主对角线元素与该行最大值交换后："<<endl;
```



```
    for(i=0;i<M;i++)
    {    for(j=0;j<M;j++)    cout<<setw(4)<<a[i][j];
         cout<<endl;
    }
}
```

次对角线元素与该行的最小值交换。见第 4 章课后作业

按规律自动产生二维 N*N 数据（倒序填充、蛇形填充），获取二维数组的指定区域（下三角或上三角）的数据，并按指定图形或格式（直角三角形、等腰三角形）输出：参考第 4 章 PPT96-104

倒序填充	蛇形填充
36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1	1 2 3 4 5 6 12 11 10 9 8 7 13 14 15 16 17 18 24 23 22 21 20 19 25 26 27 28 29 30 36 35 34 33 32 31

`int a[10][10], n, i, j, data;`
`cin >> n;`
`data = n*n;`
`for (i = 0; i<n; i++)`
`  for (j = 0; j<n; j++)`
`  a[i][j] = data--;`
	`int a[10][10], n, i, j, data = 1;`
	`cin >> n;`
	`for (i = 0; i<n; i++)`
	`  if (i % 2 == 0)`
	`  for (j = 0; j<n; j++)  a[i][j] = data++;`
	`  else`
	`  for (j = n - 1; j >= 0; j--)  a[i][j] = data++;`

【案例 4.20】 矩阵的转置。参考第 4 章 PTT105-107
【案例 4.21】 矩阵的和运算。参考第 4 章 PTT108-109

（七）字符数组
掌握本文档第 2 页的表格中涉及到的内容，并会使用这些内容完成编程。

【案例 4.3】两个字符串比较问题。第 4 章 PTT122-123，会使用 strcmp( )函数
拓展：不用 strcat 函数，完成将两个字符串连接起来。见第 4 章课后作业

【案例 4.25】恺撒加密、解密：第 4 章 PTT126-131

会对字符数组进行简单的处理。例如：大小写字母的互相转换（实验 11 第 1 题）、统计某个字符的个数（实验 11 第 2 题）。
1. 统计 ASCII 码值为偶数的字符个数：

```
void main( )
{    int i=0,cnt=0;    char s[30];
     cout<<"请输入字符串：";
     cin.getline(s,30);
     for(i=0;s[i]!='\0';i++)
          if(s[i]%2==0) cnt++;
     cout<<"ASCII 值为偶数的字符共有："<<cnt<<"个"<<endl;
}
```

2. 字符串逆序存放：编程思路类似于数值型数组的逆序存放

（八）string 定义的字符串

掌握本文档第 2 页的表格中涉及到的内容，并会使用这些内容完成编程。

【案例 4.5】两个字符串比较问题。第 4 章 PTT142

1. 实验 11 中的相关程序

例如，（1）将小写字母替换成大写字母

（2）将字符串中的数字 0 到 9 替换成小写字母 a 到 j。

```
if(str[i]>='0'&&str[i]<='9')     str[i]=str[i]-'0'+'a';
```

2. **用 string 字符串类编程**，其编程思路与字符数组是一致的，不同之处在于：定义、输入、判断字符串是否结束。（请参考本文档中第 2 页的表格）

以上面的第 1 题（统计 ASCII 码值为偶数的字符个数）为例，红色部分是需要改动的。

```
#include <iostream>
#include <string>
using namespace std;
void main( )
{    int i=0,cnt=0;     string s;
     cout<<"请输入字符串：";
     getline(cin,s);
     for(i=0;i<s.size( );i++)
          if(s[i]%2==0) cnt++;
     cout<<"ASCII 值为偶数的字符共有："<<cnt<<"个"<<endl;
}
```

第 5 章 指针

利用指针在数组中完成各种操作，算法思想在第 4 章都有，只是将 a[i]改成指针的形式。有 3 种形式都可以用，详见本文档第 2 页。

特别注意：

（1）指针必须有明确的指向（即：必须初始化）；

（2）用指针操作数组时，若执行了 p++的操作，下一轮循环用 p 之前，记得先执行 p=a，即让指针 p 再次指向数组首元素。

【案例 5.1】多种方式访问一维数组。完成输入、输出、求和、平移等基本操作。第 5 章 PPT36-37

【案例 5.7】数组降序存放，查找数据，用指针方式操作。第 5 章 PPT38-39

教材 P207 拓展⑥：使用指针法编程实现移动一维数组中的内容，键盘输入 n 个数据存放数组中，要求把下标从 0~pi（含 pi，pi 小于等于 n-1）的数组元素顺序平移到数组的最后（即数组的右部）。 其实就是实验 12 的第 4 题。

【案例 5.8】 用指针方式将数组逆序存放。第 5 章 PPT40-43

【案例 5.11】用指针方式求两个集合的交集。第 5 章 PPT44-53

第 6 章 函数

1. 关于函数参数的传递：**数值传递、引用传递、地址传递（指针传递）**

关于函数声明、函数定义（含函数头、函数体两部分）、函数调用：

函数声明、函数头、函数调用语句，这 3 者中已知 1 个，填另外 2 个。方法：

函数声明、函数头：两者的**返回值类型**、**函数名称**、**参数类型**、**参数个数**都是一样的，参数名称可以不一

样，也可以一样；函数声明中可以省略参数名称（建议：所有参数名称都省略、或者全部都不省略，尽量不要有些参数名称省略了，有些又没省略）。

填写函数调用语句：函数名称与函数头中的名称一致，参数个数、顺序与形参一致；有返回值函数的返回值可以直接输出、或赋值给其它变量；无返回值函数：直接调用即可，不能赋值给其它变量。

例 1：已知直角三角形的两直角边，求斜边长（有返回值函数，参数是数值传递）

```
void main( )
{    double xiebian(double,double);
     double a,b,c;
     cout<<"请输入两个直角边长度:";
     cin>>a>>b;
     _____;        // 填  c=xiebian(a,b)

     cout<<"斜边是:"<<c<<endl;
}
_____ // 此处缺少函数头，应填  double xiebian(double x, double y)
{    double z;
     z=sqrt(x*x+y*y);
     return z;
}
```

例 2：已知直角三角形的两直角边，求斜边长（无返回值函数，第 3 个参数是引用传递，通过引用将子函数中修改过的值带回到主函数）

```
void main( )
{
    _____// 填函数声明 void xiebian(double, double, double &);
                              // 或 void xiebian(double x, double y, double &z);
                              // 注意：第 3 个参数是引用，在函数声明中可以省略变量名，但&不能省略
    double a,b,c;
    cout<<"请输入两个直角边长度:";
    cin>>a>>b;
    _____// 填 xiebian(a,b,c);
                             // 注意第 3 个参数是 c，不是&c，而且不能将函数调用结果赋值给其它变量
    cout<<"斜边是:"<<c<<endl;
}
void xiebian(double x,double y, double &z)
{    z=sqrt(x*x+y*y);    }
```

例 3：求一维数组的最小值（第 1 个参数是地址传递，第 2 个参数是数值传递，第 3 个参数是引用传递）

```
void main( )
{
    _____  // 填 void min(int a[ ], int n, int &m);  或  void min(int [ ], int , int &);
// 说明：此处需要函数声明，可以把所有参数名称都写上，也可以省略所有的参数名称。
// 特别注意黄色部分：数组名可以省略，但[ ]不能省；表示引用的符号&也不能省。
// 同样的道理，若形参中有 int *p 这种指针变量，在函数声明时，p 可以省略，但*不能省。
    int a[15],i,m;
    for(i=0;i<15;i++) cin>>a[i];
    _____  // 填 min(a,15,m); 注意：第一个参数是数组首地址，第 2 个参数是元素个数
```

```
    cout<<"最小值为："<<m<<endl;
}
void min(int a[ ],int n,int &m)
{   m=a[0];
    for(int i=0;i<n;i++)
        if(a[i]<m) m=a[i];
}
```

2. 相关例题及实验 13

【案例 6.1】变量的作用域和存储类型。

【案例 6.2】利用函数调用、三种参数传递方法，完成两个数据的交换。第 6 章 PPT29,34

【案例 6.3】利用函数调用完成连乘测试。参数传递方式：数组参数传递+值传递。第 6 章 PPT46-48

第 6 章 PPT35-44 页的 3 个例题（排序，逆序存放，求数组最小值最大值），主要关注子函数。

3. 子函数中涉及到的算法：第 3 章、第 4 章中的程序或与之类似的程序。

4. 递归函数：求 **n!, a^b，裴波那契数列，年龄推算问题**，十进制转二进制：见第 6 章 PPT 5-17