```cpp
//输出字符串“hello world”
void helloworld()
{
    cout<<"Hello World"<<endl;
}
//输出一维数组的元素
void array1(int arr[],int length)
{
    for(int i=0;i<length;i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}
//输出二维数组的元素
void array2(int arr[][3],int length_row,int length_col)
{
    for(int i=0;i<length_row;i++)
    {
        for(int j=0;j<length_col;j++)
        {
            cout<<arr[i][j]<<" ";
        }
    }
    cout<<endl;
}
//两个整数求和
int sum(int a,int b)
{
    return a+b;
}
//数组元素求和
int array1_sum(int arr[],int length)
{
    int s=0;
    for(int i=0;i<length;i++)
    {
        s=s+arr[i];
    }
    return s;
}
```

```
//二维数组元素求和
int array2_sum(int arr[][3],int length_row,int length_col)
{
    int s=0;
    for(int i=0;i<length_row;i++)
    {
        for(int j=0;j<length_col;j++)
        {
            s=s+arr[i][j];
        }
    }
    return s;
}
//判断年份是否为闰年
bool leap_year(int year)
{
    if((year%4==0 && year%100!=0)||(year%400==0))
    {
        return true;
    }
    return false;
}
//判断整数是否为素数
bool prime_num(int num)
{
    if(num==0||num==1)
    {
        return false;
    }
    if(num==2)
    {
        return true;
    }
    for(int i=2;i<=num/2;i++)
    {
        if(num%i==0)
        {
            return false;
        }
    }
    return true;
}
```

```
//判断字符串中是否包含某个字符
bool find_char(char str[],char s)
{
    for(int i=0;str[i]!='\0';i++)
    {
        if(str[i]==s)
        {
            return true;
        }
    }
    return false;
}

//===============整型数组============
//交换int元素位置
void swapInt(int& x,int& y)
{
    int temp;
    temp = x;
     x = y;
    y=temp;
}
void SortInt(int arr[], int n)
{
    for(int i = 0;i <n;i++){
        //比较两个相邻的元素
        for(int j = 0;j < n-i-1;j++){
             if(arr[j] > arr[j+1])
               swapInt(arr[j],arr[j+1]);
         }
     }
    for(int i=0;i<n;i++)
        cout<<arr[i]<<" ";
    cout<<endl;
}
//=============字符型数组==========================
//交换char元素位置
void swapChar(char& x,char& y)
{
    char temp;
    temp = x;
    x=y;
    y=temp;
}
```

```
void SortChar(char arr[], int n)
{
    for(int i = 0;i <n;i++){
        //比较两个相邻的元素
        for(int j = 0;j < n-i-1;j++){
              if(arr[j] > arr[j+1])
             {
                swapChar(arr[j],arr[j+1]);
             }
        }
     }
    for(int i=0;i<n;i++)
        cout<<arr[i]<<" ";
    cout<<endl;
}

//=============字符串数组========================
//对字符串数组排序，使用字符串处理
void SortChar2(char str[][20], int n) //数组行数由n确定
{
     char a[20];
     int i, j;
     for (i = 0; i < n-1; i++) {
        for(j=0;j<n-i-1;j++ ) //冒泡排序
              if (strcmp(str[j], str[j + 1]) > 0) {
                   strcpy(a, str[j]);
                   strcpy(str[j], str[j + 1]);
                   strcpy(str[j+1], a);
             }
    }
    for(int i=0;i<n;i++)
        cout<<str[i]<<" ";
    cout<<endl;
}
```

```
//对字符串数组排序，不使用字符串处理函数
int compStr(char a1[],char a2[])
{
    int n1 = sizeof(a1);
    int n2 = sizeof(a2);
    int n = 0;
    if(n1<n2)
        n=n1;
    else n= n2;
    int i;
    for(i=0;i<n;i++){
        if(a1[i]>a2[i])
            return 1;
        else if(a1[i]<a2[i])
            return 2;
    }
    if(n1==n2&&i==n)
        return 0;
}
//交换第x行和第y行
void swap2DChar(char str[][20],int x,int y){
    char temp;
    for(int j=0;j<20;j++){
       temp=str[x][j];
       str[x][j]=str[y][j];
       str[y][j]=temp;
    }
}
void Sort2DChar(char str[][20], int n)
{
    char a[20];
     int i, j;
     for (i = 0; i < n-1; i++) {
        for(j=0;j<n-i-1;j++ )
            if (compStr(str[j], str[j + 1]) == 1) {
                swap2DChar(str,j,j + 1);
    }
    }
    for(int i=0;i<n;i++)
        cout<<str[i]<<" ";
    cout<<endl;
}
```

```
//=====================对自定义类型Date进行排序===================
struct date{
    int year,month,day;
}d[10];
//比较两个date类型的大小
int compDate(date d1,date d2)
{
    int n =
(d1.year-d2.year)*365+(d1.month-d2.month)*30+(d1.day-d2.day);
    if(n>0)
        return 1;
    else if(n<0)
        return 2;
    else return 0;
}
//交换位置
void swapDate(date& d1,date& d2)
{
    date temp={0,0,0};
    temp = d1;
    d1 = d2;
    d2 = temp;
}
void SortDate(date d[],int n)
{
    for(int i = 0;i < n;i++){
        //比较两个相邻的元素
        for(int j = 0;j < n-i-1;j++){
            if(compDate(d[j],d[j+1])==1)
            {
                swapDate(d[j],d[j+1]);
            }
         }
    }
    for(int i=0;i<n;i++)
        cout<<d[i].year<<"-"<<d[i].month<<"-"<<d[i].day<<" ";
    cout<<endl;
}
```

```
//字符串求长度
int string_length(char c[]) {
    int length = 0;
    int i = 0;
    while (c[i]!='\0') {
        length++;
        c++;
    }
    return length;
}

//字符串拷贝
void string_copy(char c1[], char c2[]) {
    int len1 = string_length(c1);
    int len2 = string_length(c2);
    int len = 0;
    if (len1 < len2)
        len = len2;
    else
        len = len1;
    for (int i = 0; i < len; i++) {
        c1[i] = c2[i];
    }
}
//字符串连接
void string_cat(char c1[], char c2[]) {
    int len1 = string_length(c1);
    int len2 = string_length(c2);
    for (int i = len1; i < len1 + len2; i++) {
        c1[i] = c2[i-len1];
    }
}
//字符串比较
int string_compare(char c1[], char c2[]) {
    int i = 0;
    while (c1[i]==c2[i])
    {
        if (c1[i++] == '\0')
        {
            return 0;
        }
    }
    return (c1[i] - c2[i]);
}
```

```c
//字符串逆序
void string_reverse(char c[]) {
    int len = string_length(c);
    int i, n = len / 2;
    char tem;
    for (i = 0; i < n; i++)
    {
        tem = c[i];
        c[i] = c[len - 1 - i];
        c[len - 1 - i] = tem;
    }
}


//字符串统计
struct Char {
    char ch;
    int num;
};
void statistics(char source[100], Char chars[100]) {
    int i_chars = 0;
    for (int i = 0; i < strlen(source); i++) {
        // 判断是否已经出现过
        int j = 0;
        while (j < i_chars) {
            if (chars[j].ch == source[i]) {
                chars[j].num++;
                break;
            }
            j++;
        }
        // if 没出现过
        if (j == i_chars) {
            chars[i_chars].ch = source[i];
            chars[i_chars++].num = 1;
        }
    }
    // 加入结束标识
    chars[i_chars].ch = '\0';
    chars[i_chars].num = -1;
}
```

```
//查找单个字符，返回全部下标
void find(char source[100], char x, int index[100]) {
    int i_index = 0;
    for (int i = 0; i < strlen(source); i++) {
        if (source[i] == x)
            index[i_index++] = i;
    }
}

//查找子串，返回全部下标
void find(char source[100], char xstr[20], int index[100]) {
    int i_index = 0;
    for (int i = 0; i < strlen(source); i++) {
        int j = 0;
        while (j < strlen(xstr)) {
            if (source[i + j] != xstr[j])
                break;
            j++;
        }
        if (j == strlen(xstr))
            index[i_index++] = i;
    }
}
```