第5章 函　　数

5.1　引例

例 5.1.1　处理并输出。

先输入整数 n（n<100），然后再输入 n 个整数。请完成以下 3 个任务。

（1）输出这些整数。

（2）把这些整数逆置后输出。

（3）把这些整数升序排列并输出。

输出时，每两个数据之间留一个空格。

输入样例：

```
5
3 2 1 5 4
```

输出样例：

```
3 2 1 5 4
4 5 1 2 3
1 2 3 4 5
```

本题的每个任务都要求输出，若把输出的代码重复使用 3 次，则代码比较冗长。当一段代码需重复多次使用时，通常会考虑把这段代码独立出来作为一个整体，这就需用到自定义函数，具体代码如下。

```
//C++风格代码
#include<bits/stdc++.h>          //万能头文件
using namespace std;
const int N=100;
void prtArray(int a[], int n){ //定义输出数组元素的函数
    for(int i=0; i<n; i++) {
        if(i!=0) cout<<" ";
        cout<<a[i];
    }
    cout<<endl;
}
int main() {
```

```
    int a[N],n,j,k;
    cin>>n;
    for(j=0; j<n; j++) cin>>a[j];
    prtArray(a, n);             //调用自定义函数prtArray
    for(k=0; k<n/2; k++) {
        swap(a[k],a[n-1-k]);
    }
    prtArray(a, n);             //调用自定义函数prtArray
    sort(a, a+n);               //调用系统函数sort,排序区间[&a[0], &a[n-1]+1)
    prtArray(a, n);             //调用自定义函数prtArray
    return 0;
}

//C风格代码
#include<stdio.h>
#define N 100
void prtArray(int a[], int n){ //定义输出数组元素的函数
    int i;
    for(i=0; i<n; i++) {
        if(i!=0) printf(" ");
        printf("%d",a[i]);
    }
    printf("\n");
}
void mySort(int a[], int n) {  //函数定义,冒泡排序
    int i,j,t;
    for(i=0; i<n-1; i++) {
        for(j=0;j<n-1-i;j++) {
            if (a[j]>a[j+1]) {
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
}
int main() {
    int a[N],i,n,t;
    scanf("%d",&n);
    for(i=0; i<n; i++) scanf("%d",&a[i]);
    prtArray(a, n);             //调用自定义函数prtArray
    for(i=0; i<n/2; i++) {
        t=a[i];
        a[i]=a[n-1-i];
        a[n-1-i]=t;
    }
    prtArray(a, n);             //调用自定义函数prtArray
```

```
    mySort(a, n);                    //调用自定义函数 mySort
    prtArray(a, n);                  //调用自定义函数 prtArray
    return 0;
}
```

这里定义了一个自定义函数 prtArray，用于输出一维数组中的 *n* 个元素，数据之间留一个空格，然后 3 次调用该函数（函数必须被调用了才有效果）完成题中 3 个任务的输出。这样代码显然更加简洁。实际上，一些常用功能即使在一个程序中不被调用多次，也经常写成一个个自定义函数。例如，判断一个数是否为素数，求两个整数的最大公约数或最小公倍数、二分查找及排序等。

另外，C++风格代码用了 C++的万能头文件 bits/stdc++.h，包含这个头文件相当于包含了 C++所有的头文件，简化了各种头文件的包含。例如，C++风格代码使用系统函数 sort，本该包含头文件 algorithm，但包含万能头文件后就无须再包含该头文件。使用万能头文件的 C++程序只需要使用以下两句而不用再包含其他头文件。

```
#include<bits/stdc++.h>          //万能头文件
using namespace std;             //引入 std 命名空间
```

需要注意的是，虽然 Dev-C++编译环境和目前很多在线测评系统都支持万能头文件，但也有些在线测评系统和编译系统不支持万能头文件。通用起见，本书代码一般不用万能头文件。读者编写代码时可酌情使用万能头文件。

C 风格代码还定义了一个自定义函数 mySort。在 main 函数中可调用该函数完成排序任务。C 语言中也有实现排序功能的系统函数 qsort。该函数的调用较复杂，本书将在第 7 章对其进行讨论。

5.2 函数基础知识

5.2.1 函数概述

简言之，函数是一组相关语句组织在一起所构成的整体，并以函数名标注。

从用户的角度而言，函数分为库函数（系统函数）和用户自定义函数。库函数有很多，例如，在 math.h 中的 fabs、pow、ceil、floor、round 等，调用示例如下。

```
double x=fabs(-3.5);             //x 等于 3.5,fabs:求实数的绝对值
double x=pow(2.0, 3.0);          //x 等于 8.0,pow:求幂次方
int x=ceil(3.78);                //x 等于 4,ceil:向上取整,即取不小于参数的最小整数
int x=floor(3.78);               //x 等于 3,floor:向下取整,取不大于参数的最大整数
int x=round(3.6),y=round(3.4); //x 等于 4、y 等于 3,round：四舍五入取整
```

又如，在 stdlib.h 中的 malloc、free、srand、rand 等，调用示例如下。

```
//申请可存放 10 个元素的整型动态数组 a
int *a=(int *)malloc(sizeof(int)*10);
free(a);                         //释放动态数组 a 的存储空间
```

```
srand(time(NULL));                  //随机种子初始化,用 time 需包含头文件 time.h
int a=rand();                       //产生 0~RAND_MAX（符号常量）之间的一个随机整数
int b=20+rand()%(80-20+1);          //产生 20~80 之间的一个随机整数
```

注意，库函数使用时需包含相应头文件。另外，在 Dev-C++环境下的 C++风格代码中可直接调用 max、min 等系统函数。调用示例如下。

```
int x=max(123,456);             //x 等于 456,max:求两者中的大者
int x=min(123,456);             //x 等于 123,min:求两者中的小者
```

本章主要介绍用户自定义函数。读者可以自行查阅并测试感兴趣的系统函数。

一个大的程序一般分为若干程序模块。每个模块用来完成一个特定的功能，每个模块一般通过定义并调用自定义函数来实现。

函数是 C/C++程序的基本构成单位，一个 C/C++程序由一个主函数 main 及若干其他函数构成。C/C++程序从 main 函数中开始执行，在 main 函数中结束。

函数的作用是通过函数调用实现的。操作系统调用主函数，主函数调用其他函数，其他函数可以调用主函数之外的其他函数。同一函数可以被一个或几个函数调用任意次。

图 5-1 是一个函数调用示意图。

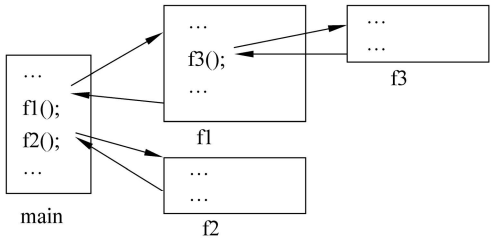

图 5-1　函数调用示意图

由图 5-1 可见，main 函数调用 f1、f2 函数，f1 函数调用 f3 函数，f3 函数调用结束返回到 f1 函数中的调用点，f1 函数调用结束返回到 main 函数中的调用点，f2 函数调用结束返回到 main 函数中的调用点。

5.2.2　函数的定义

函数定义由函数头和函数体两部分组成。函数定义的一般形式如下。

```
函数类型 函数名([形参列表]) {
    函数体
}
```

说明：

（1）“函数类型 函数名([形参列表])”是函数头，{}中的是函数体。

（2）函数类型（也称返回类型）可以是各种基本数据类型、指针类型、结构体类型、void（空类型，明确指定函数不返回值）等。

（3）函数名必须是合法的标识符。

（4）函数定义中的参数为形式参数，简称形参。根据是否有形参，函数可分为带参函

数和无参函数。形参列表的每个参数包括参数类型和参数名。形参列表若有多个参数，则以逗号分开。C++支持参数带默认值，且默认值参数须处于形参列表的最右侧，即若某个参数带有默认值，则其右侧的所有参数都应带默认值。

（5）根据是否有返回值，函数可分为有返回值函数和无返回值函数。通过函数中的return 语句返回函数的返回值。return 语句的一般格式如下。

return [返回值表达式];

其中，返回值表达式的类型一般应与返回类型一致，若不一致则以返回类型为准。语句return;控制程序流程返回到调用点；若 return 语句带返回值表达式，则在控制程序流程返回调用点的同时带回一个值。

下面给出几个函数定义的例子。

```
//C++风格代码
void print() {                                  //无参也无返回值,返回类型用 void
    cout<<"hello"<<endl;
}
int max(int a, int b) {                         //求两个整数的最大值
    return a>=b?a:b;
}
//重载函数：C++中参数不同的若干同名函数
string max(string a, string b) {                //求两个字符串的最大值
    return a>=b?a:b;
}
//C++带默认值参数的函数，带默认值的参数处于形参列表的最右侧
int max3nums(int a, int b=2, int c=3) {         //求 3 个整数的最大值
    int t=max(a, b);                            //嵌套调用 max(int, int)
    return max(t, c);
}

//C 风格代码
void print() {                                  //无参也无返回值,返回类型用 void
    printf("hello\n");
}
int max(int a, int b) {                         //求两个整数的最大值
    return a>=b?a:b;
}
```

5.2.3 函数的声明

若函数定义在函数调用之前，则定义时的函数头可以充当函数声明（或称函数原型）；否则函数在调用之前应先进行函数声明，形式如下。

函数类型 函数名([形参列表]);

即，以函数定义时的函数头加分号表示函数声明。其中，形参列表中的形参名可以省略。

例如 5.2.1 节定义 max 函数声明如下。

```
//C++风格代码
```

```
string max(string a, string b);          //函数头加分号作为函数声明
int max(int, int);                       //函数声明中的形参名可省略

//C 风格代码
int max(int, int);                       //函数声明中的形参名可省略
```

5.2.4　函数的调用

函数的调用的形式一般如下。

[变量=]函数名([实际参数表])[;]

void 返回类型的函数只能以语句形式调用，其他返回类型的函数一般以表达式形式调用，否则其返回值没有意义。

调用时的参数称为实际参数，简称实参，参数的类型、顺序、个数必须与函数定义中的一致，但带默认值参数的函数调用时实参个数可以与形参个数不一致（有默认值的形参可使用其默认值而无须提供实参）。

函数调用时，把实参依序传递给形参，然后执行函数定义体中的语句，执行到 return 语句或函数结束时，程序流程返回到调用点。

调用 5.2.1 节定义的函数的方法如下。

```
//C++风格代码
print();                                 //void 返回类型的函数以语句形式调用
int t=max(123,99);                       //有返回值的函数一般以表达式形式调用
cout<<max(1,2)<<endl;                    //调用 max(int, int)
cout<<max("abc", "cdfg")<<endl;          //调用 max(string, string)
cout<<max3nums(1)                        //实参为 1,2,3,后两个参数使用默认值
    <<" "<<max3nums(4,5)                 //实参为 4,5,3,后一个参数使用默认值
    <<" "<<max3nums(5,3,4)<<endl;        //实参为 5,3,4,不使用默认值

//C 风格代码
print();                                 //void 返回类型的函数以语句形式调用
int t=max(123,99);                       //有返回值的函数一般以表达式形式调用
printf("%d\n",max(1,2));                 //调用 max(int, int)
```

函数调用也可用变量作为实参，实参和形参可以同名，但它们实际上是局限于各自所在函数的不同变量。

在线做题时，题目通常有多组测试数据，可把一组测试的代码单独作为一个函数处理，然后在循环中调用该函数完成多组测试。

5.3　函数举例

例 5.3.1　逆序数的逆序和。

输入两个正整数，先将它们分别倒过来，然后再相加，最后再将结果倒过来输出。注意：前置的零将被忽略。例如，输入 305 和 794，倒过来相加得到 1000，输出时只输出 1

即可。

因为求逆序数的方法是一样的，可以编写一个求逆序数的函数，调用 3 次即可完成两个输入的整数及 1 个结果整数的逆置。

思考：当 x=1234，如何得到 x 的逆序数？

设 r 为 x 的逆序数，可以这样考虑：r=((4×10+3)×10+2)×10+1=4321，即让 r 一开始为 0，再不断把 x 的个位取出来加上 r×10 重新赋值给 r，直到 x 为 0（通过 x=x/10 不断去掉个位）。具体代码如下。

```
//C++风格代码
#include<iostream>
using namespace std;
int reverseNum(int x) {                    //构成逆序数的函数,x 是正整数
    int r=0;
    while(x>0) {
        r=r*10+x%10;                       //移位后右边加上 x 的个位数
        x=x/10;                            //去掉 x 的个位
    }
    return r;
}
int main() {
    int a,b;
    cin>>a>>b;
    int c=reverseNum(a)+reverseNum(b);
    cout<<reverseNum(c)<<endl;
    return 0;
}

//C 风格代码
#include<stdio.h>
int reverseNum(int x) {                    //构成逆序数的函数,x 是正整数
    int r=0;
    while(x>0) {
        r=r*10+x%10;                       //移位后右边加上 x 的个位数
        x/=10;                             //去掉 x 的个位
    }
    return r;
}
int main() {
    int T,a,b,c;
    scanf("%d", &T);
    while(T--) {
        scanf("%d %d",&a,&b);
        c=reverseNum(a)+reverseNum(b);
        printf("%d\n",reverseNum(c));
    }
    return 0;
```

```
}
```

例 5.3.2　素数判定函数。

输入一个正整数 n，判断 n 是否为素数，是则输出 yes，否则输出 no。要求写一个函数判断一个正整数是否为素数。

关于 n 是否为素数，已知可以从 2 至 $\sqrt{n}$ 判断是否有 n 的因子，若有则不是素数。因为结果只有两种可能（是或否），所以返回类型设为 bool（返回 true、false 分别表示是、否）或 int（返回 1、0 分别表示是、否）；而 n 是要被判断的数，因此需要一个整型参数。具体代码如下。

```
//C++风格代码
#include<iostream>
#include<cmath>                  //系统函数 sqrt 求开方数须包含此头文件
using namespace std;
bool isPrime(int n) {            //判断 n 是否为素数,若是则返回 true,否则返回 false
    bool flag=true;              //一开始假设是素数,标记变量初值设为 true
    double limit=sqrt(1.0*n);    //参数最好为 double 类型,以免在线提交编译错误
    for(int i=2; i<=limit; i++) {
        if(n%i==0) {             //若有因子,则可以判断 n 不是素数
            flag=false;
            break;
        }
    }
    if (n==1) flag=false;        //对 1 特判
    return flag;
}
int main() {
    int n;
    cin>>n;
    if(isPrime(n)==true)
        cout<<"yes"<<endl;
    else
        cout<<"no"<<endl;
    return 0;
}

//C 风格代码
#include<stdio.h>
#include<math.h>                 //使用系统函数 sqrt 求开方数须包含此头文件
int isPrime(int n) {             //判断 n 是否为素数,是则返回 1,否则返回 0
    int flag=1;                  //一开始假设是素数,标记变量初值设为 1
    int limit=sqrt(1.0*n);       //参数最好为 double 类型,以免在线提交编译错误
    for(int i=2; i<=limit; i++) {
        if(n%i==0) {             //若有因子,则可以判断为不是素数
            flag=0;
            break;
```

```
        }
    }
    if (n==1) flag=0;           //对1特判
    return flag;
}
int main() {
    int n;
    scanf("%d",&n);
    if(isPrime(n)==1)
        puts("yes");
    else
        puts("no");
    return 0;
}
```

例 5.3.3 最小回文数。

输入整数 *n*，输出比该数大的最小回文数。回文数指的是正读、反读一样的数，例如131、1221 等。要求写一个函数判断一个整数是否为回文数。

判断一个整数是否为回文数可以调用例 5.3.1 中的求逆序数的函数 reverseNum，判断该数与逆序数是否相等。因为要找比 *n* 大的最小回文数，可以从 *n*+1 开始逐个尝试是否满足逆序数等于本身的条件，第 1 个满足条件的数即为结果。

```
//C++风格代码
#include<iostream>
using namespace std;
int main() {
    bool check(int);            //定义在调用之后,则须在调用前先声明
    int n;
    cin>>n;
    while(true) {
        n++;
        if(check(n)==true) break;
    }
    cout<<n<<endl;
    return 0;
}
int reverseNum(int x) {         //构成逆序数的函数,x是正整数
    int r=0;
    while(x>0) {
        r = r*10 + x%10;        //移位后右边加上x的个位数
        x = x/10;
    }
    return r;
}
bool check(int n) {
    return n==reverseNum(n);
```

```
}

//C 风格代码
#include<stdio.h>
int main() {
    int check(int n);            //定义在调用之后,则须在调用前先声明
    int i,n;
    scanf("%d",&n);
    while(1) {
        n++;
        if(check(n)==1) break;
    }
    printf("%d\n",n);
    return 0;
}
int reverseNum(int x) {          //构成逆序数的函数,x 是正整数
    int r=0;
    while(x>0) {
        r = r*10 + x%10;         //移位后右边加上 x 的个位数
        x = x/10;
    }
    return r;
}
int check(int n) {
    return n==reverseNum(n);
}
```

例 5.3.4　大整数加法。

输入 2 个大正整数（长度可能达到 1000 位），求两者之和。

基本思路：两个大正整数作为字符串处理，加法根据“右对齐、逐位相加”的方法，关键在于右对齐相加及进位处理。右对齐相加可以在把两个字符串逆置后从第 1 个字符开始相加；进位处理方面，可以用一个整型变量表示，其初值一开始设为 0，在加法计算过程中把其加到和中，并不断更新为最新的进位。字符串的逆置可以写一个以 string 类型变量（C++）或 char 数组（C 语言）为形参的函数。

需要注意的是，string 类型形参的变化不会影响到实参，因此通过返回值返回变化后的结果。C++风格的代码中，用第 1 个串存放最终结果，因此需要保证其长度不小于第 2 个串，方法是判断两个串的长度，若前一个串短，则调用系统函数 swap 交换两个串。C 风格的代码中，额外使用一个结果字符数组存放结果，具体代码如下。

```
//C++风格代码
#include<iostream>
#include<string>
using namespace std;
string reverse(string s) {       //逆置字符串
    int n=s.size(), mid=n/2;
```