任务二

计算一名选手的得分

任务描述

◆ 利用 C 语言设计完成如下功能的程序：输入三位评委对一名选手的评分，计算其总分和平均分

学习要点

◆ 各种主要数据类型以及相应的存储格式
◆ 各种运算符的含义和使用方法
◆ 各种表达式的结果和计算过程
◆ 类型转换及其转换规则

学习目标

◆ 学会输入函数 scanf()和输出函数 printf()的使用
◆ 熟悉 C 语言的基本语法单位
◆ 掌握基本数据类型常量的表示、变量的定义、变量的初始化
◆ 掌握各种运算符的功能、优先级和结合性
◆ 掌握各种表达式的正确书写及计算过程

专业词汇

operator 运算符　　expression 表达式　　assignment 赋值
arithmetic 算术　　comma 逗号　　relational 关系　　logic 逻辑

【任务说明】针对一个选手，在屏幕上将输入各评委的打分，要求实现一个简单的计算功能，求出这名选手的总分及平均分。如图 2.1 所示。通过本任务，我们将熟悉输入和输出函数的使用，主要数据类型以及相应的存储格式；掌握各种运算符的功能、优先级和结合性，以及各种表达式的正确书写及计算过程。

【问题引入】输入三名评委对一名选手的打分，要求输出这名选手所得的总分及平均分。

```
f1:99
f2:96
f3:95

total:290       average:96.67
```

图 2.1　计算一名选手得分的程序运行结果

【问题分析】要完成选手的总分及平均分的计算，应完成三个过程：第一步是必须要学会如何输入得分；第二步必须对输入的得分进行总分及平均分的计算；第三步是对所得到的结果进行显示。其中第一步和第三步可合成一个子任务，即选手得分的输入和输出；第二步是第二个子任务，即选手总分及平均分的计算。

任务 2.1　选手得分的输入/输出

问题情景

举办校园歌手大赛，给每一名选手输入三名评委的打分，并按要求进行输出。

实现过程

【例 2.1】（假设只有三名评委打分）

```
#include "stdio.h"          /* 文件预处理 */
void main()                 /* 函数名 */
{
  int f1,f2,f3;             /* 定义3个整型变量存储评委打分*/
  printf("\nf1:");          /* 提示输入打分 */
  scanf("%d",&f1);          /* 输入评分1 */
  printf("f2:");
  scanf("%d",&f2);
  printf("f3:");
  scanf("%d",&f3);
  printf("\nf1=%d\tf2=%d\tf3=%d\t\n",f1,f2,f3);      /* 输出计算结果 */
  getch();                  /*用于读取按键的值。一般放在程序末尾，起到暂停的作用*/
}
```

程序运行结果如图 2.2 所示：

上面的程序可分析出：

本任务中，要掌握的知识点是：

① 要了解 C 语言的结构和运行环境。

② 要掌握如何定义变量。

③ 要掌握如何对变量进行输入和输出。

```
f1:99
f2:96
f3:95

f1=99   f2=96   f3=95
```

图 2.2　例 2.1 的运行结果

相关知识

2.1.1 标识符

标识符是程序设计者为自定义的变量、函数、类型所起的名字，其命名规则如下：

（1）只能由字母、数字、下划线组成。

（2）第一个字符必须是字母或下划线。

（3）不能与关键字同名，尽量“见名知义”。

（4）区分大小写。如 my、My、MY 是 3 个不同的标识符。

【例 2.2】请指出下面哪些是非法的标识符。

a　f-2　f6　m+n　x4b　4af　as_d　a.ss　total　main　month　int

由系统预先定义的标识符称为“关键字”（又称保留字），它们都有特殊的含义，不能用于其他目的。C 语言关键字有 32 个，如表 2.1 所示。

表 2.1　C 语言关键字

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	short	signed	sizeof	static	return
struct	switch	typedef	union	unsigned	void	volatile	while

2.1.2 变量

变量是指在程序运行过程中其值可以改变的量。变量代表着存储器中指定的单元，该单元中的数据就是变量的值。变量名是一个标识符，程序通过变量名访问变量的值。如图 2.3 所示，x 是变量名，方框表示某个存储单元，单元中的数据 8 是变量 x 的值。

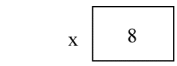

图 2.3　变量 x 的表示

变量所对应的存储单元的大小是由变量的类型决定的。

例如，在程序中有如下说明：

```
int a,b;
```

则该语句有两个作用：

（1）程序运行时，系统根据变量的类型为其分配合适的存储单元。如系统根据 a，b 的类型 int 为 a，b 各分配两个字节，并按照整数的格式存放数据。

（2）编译系统根据变量的类型进行语法检查，即检查对该变量的操作是否合法。如已说明 a,b 为整型变量，则 a%b（a 除以 b 取余数）是合法的。如说明“float a,b；”，则 a%b 是不允许的运算，编译时提示“非法使用浮点数”的错误信息（原因会在运算符中讲解）。

所以，“变量必须先定义，后使用”。若使用没有定义的变量名，编译时会提示“变量未曾定义”的错误信息。

变量初始化：

若在定义变量时就已知变量在程序开始运行时的值，则可以用变量初始化的方式给变量赋值。没有初始化的变量，其值一般是一个随机数。例如：

```
int a=33；                    /*定义a为整型变量，初值为33*/
float x=12.5；                /*定义x为实型变量，初值为12.5*/
char c='y'；                  /*定义c为字符型变量，初值为'y'*/
int d1=7,d2=9；               /*定义d1、d2为整型变量，初值分别为7、9*/
char s1='2', s2='r'；         /*多个变量初始化 */
char c1='a',c2；              /*定义变量c1、c2，并给变量c1赋初值*/
```

特别提示：若对几个变量赋同一个初值，也必须分别初始化。

例如：下面语句定义 3 个变量 i、j、k 赋初值 0，应写为

```
int i=0,j=0,k=0;
```

而不能写成

```
int i=j=k=0;
```

注意：初始化不是在编译阶段完成的（只有静态存储变量和外部存储变量的初始化是在编译阶段完成的），而是在程序运行时执行本函数时赋以初值的。相当于一个定义语句和一个赋值语句的叠加。

例如：

```
int a=3;
```

相当于

```
int a;                  /*指定a为整型变量*/
 a=3;                   /*赋值语句，将3赋给a */
```

又如：

```
int a,b,c=5;
```

相当于

```
int a,b,c;              /*指定a,b,c为整型变量*/
c=5;                    /*赋值语句，将5赋给c */
```

2.1.3　常量

常量是程序运行期间其值不能被改变的量，即常数。常量也分几种类型。

1. 普通常量

（1）数值型常量.也称常数，分为整型常数、浮点常数、双精度型常数。

如：4、-2.5、.245、5.8E23

（2）字符常量　字符常量是指用单引号括起来的单个字符常量或转义字符。

如：'a'、'$'、'\n'

（3）字符串常量　字符串常量是用双引号括起来的 0 个或多个字符的序列。

如："a"、"Good morning!"

2. 符号常量

用符号代替常量，叫做符号常量，一般用大写字母表示，符号常量一经定义就可以代替常量使用。

如以下程序段：

```
#define  PI  3.14159
main()
{    int r=5, area;
   area=PI*r*r;
   printf("area=%d", area);
}
```

这是一种编译预处理命令，叫做“宏定义”。指定 PI 代替常量 3.14159，在以后的程序中，凡遇到 PI 即用 3.14159 代替，只是简单的符号替换。它不属于 C 语句，所以不必在末尾加上“;”。优点是含义清楚、改动方便。

宏定义一般格式为：

```
# define  符号常量  常量
```

注意：常量名常用大写、变量名常用小写。

2.1.4 数据类型

C 语言中所有变量必须指定数据类型，在学习数据类型时，应主要掌握每种类型的常量表示、变量的定义格式、所占存储空间的大小和取值范围等，从而学会根据要处理的数据确定变量的类型。

C 语言的基本数据类型包括整型、实型、字符型。没有小数部分的数就是整型类型，而加了小数点的数则是实型（也称为浮点型类型），单个字母或符号更广泛地说是字符类型。

1. 整型数据

整型的基本类型符为 int，在 int 之前可以根据需要分别加上修饰符 short（短整型）或 long（长整型），上述类型又分为有符号型（signed）和无符号型（unsigned），即数值是否可以取负值。各种整数类型占用的内存空间大小不同，所提供数值的范围也不同。如表 2.2 所示。

表 2.2　整型数据分类

数 据 类 型	别　　称	解　　释	所 占 位 数	表示数值的范围
int	无	基本类型	16	-32768～+32767
short int	short	短整型	16	-32768～+32767
long int	long	长整型	32	-2147483648～+2147483647
unsigned int	unsigned	无符号整型	16	0～65535
unsigned short	无	无符号短整型	16	0～65535
unsigned long	无	无符号长整型	32	0～4294967295

需要说明的是，数据存储时在内存中所占字节数与具体的机器和系统有关，与具体的编译器也有关系。编程时，可以用运算符 sizeof()求出所使用环境中各种数据类型所占的字节数。

（1）整型常量

整型描述我们日常使用的整数，整数在计算机中是准确表示的。

整型常量又称整常数，即 C 语言可以识别的十进制、八进制和十六进制三种进制的整数。

十进制整数：由正负号（+或-）后跟数字串组成，正号可以省略不写，且开头的数字不

能为 0。例如：1234，-23，+187，32767，5600 等。

八进制整数：C 语言中，整数不仅可以用十进制表示，还可以用八进制或十六进制表示，并能自动进行相互的转换，不需用户干预。八进制整数的书写方式是以数字 0 打头，后跟 0～7 组成的数字串。例如，0123 表示八进制常数 123，相当于十进制数 83。如果是负数，则在打头的数字 0 前面冠以负号，如-057，-026 等。

十六进制整数：以数字 0 和小写字母 x（或大写字母 X）打头，后跟 0～9 及 A～F（或 a～f）组成的数字字母串。其中，A～F（或 a～f）分别表示十进制的 10～15。例如，0x2f 是一个十六进制数，相当于十进制的 47。如果在 0X 前冠以负号，则构成十六进制负数，如-0xB43F、-0X3a4f 等。

（2）整型变量（其内容阐述参见 2.1.2 变量）

例如，在程序中有如下说明：

```
int age=20, num=1;
long sum = 2345465;
```

【说明】

（1）最常用的整数类型是 int。默认情况下，整数字面值是 int 类型。

（2）当整数范围超过 int 型范围时，就要使用 long 型。如果要指定 long 型的整数字面值，必须在数值的后面加上大写 L 或小写 l。例如：10L、-100l。

2. 实型数据

（1）实型常量：实型常量在 C 语言中又称为浮点数。实数有两种表示形式：

① 十进制数形式。它由数字和小数点组成（注意必须有小数点）。如 0.123、.12、123.。

② 指数形式。它由尾数、e（或 E）和整数指数（阶码）组成，E（或 e）的左边为尾数，可以是整数或实数，右边是整数，指数必须为整数，表示尾数乘以 10 的多少次方。如 123e3 或 123E3 都代表 123×10^3。

（2）实型变量：实型变量分为单精度（float 型）和双精度（double 型）两类。在一般系统中，一个 float 型数据在内存中占 4 个字节，一个 double 型数据占 8 个字节。

例如，在程序中有如下说明：

```
float score = 45.6;
double dscore = 67.7;
```

【说明】

① 实型常量不分 float 和 double 型。一个实型常量可以赋给一个 float 或 double 型变量。

② 单精度实数提供 7 位有效数字，双精度实数提供 15～16 位有效数字，数值的范围随机器系统而异。例如：

```
float a = 112121.123;
```

由于 float 变量只能接收 7 位有效数字，因此最后两位小数不起作用。

【例 2.3】浮点数的有效位实例：

```
#include "stdio.h"
void main()
{
    float x;
```

```
    x = 0.1234567890;
    printf("%20.18f\n", x);
}
```

运行结果为 0.123456791043281560。

【说明】

① x 被赋值了一个有效位数为 10 位的数字，但由于 x 为 float 类型，所以 x 只能接收 7 位有效数字。

② printf 语句中，使用格式符号%20.18f，表示 printf 语句在输出 x 时总长度为 20 位，小数点位数占 18 位，输出的结果显示了 20 位数，但只有 0.123456 共 7 位有效数字被正确显示出来，后面的数字是一些无效的数字。这表明 float 型的数据只接收 7 位有效数字。

3. 字符数据

（1）字符常量

字符型常量包括普通字符常量和转义字符常量。

普通字符常量：代表 ASCII 码字符集里的某一个字符，在程序中用单引号括起来构成。如'a'、'A'、'p'等。注意'a'和'A'是两个不同的字符常量。

转义字符：又叫控制字符常量，指除了上述的字符常量外，C 语言还有一些特殊的字符常量，例如转义字符"\n"，其中"\"是转义的意思。表 2.3 列出了 C 语言中常用的特殊字符。

表 2.3　特殊字符常量及含义

转义字符序列	描　　述
\b	退格
\f	换页
\n	换行
\r	回车
\t	横向制表
\v	纵向制表
\'	单引号
\"	双引号
\\	反斜杠
\ooo	八进制数
\xhh	十六进制数

【例 2.4】试输出特殊符号常量。

```
#include "stdio.h"
void main()
{
   printf("  ab c\t de\rf\tg\n");
}
```

【说明】

① 第一个 printf 先在第一行左端开始输出“ab c”，然后遇到“\t”，它的作用是“跳格”，

即跳到下一个“输出位置”，输出“de”。下面遇到“\r”，它代表“回车”（不换行），返回到本行最左端（第一列），输出字符“f”，然后“\t”再使当前输出位置移到下一个“输出位置”，输出“g”。下面是“\n”，作用是“回车换行”。

② 显示屏显示的可能不同，这是因为在输出前面的字符后很快又输出后面的字符，在人们还没看清楚之前，新的已取代了旧的，所以误以为没有输出应该输出的字符。

（2）字符串常量

字符串常量是用双引号括起来的一串字符序列。例如："as"，"a"，""（空串）。

双引号是字符串的标记，每个字符串占用内存的字节数等于字符串长度加 1，多出的 1 个字节用于存放字符串的结束标志"\0"。

不要将字符常量和字符串常量混淆。'a'是字符常量，"a"是字符串常量，二者不同。假设 c 被指定为字符常量：

```
char c;
c = 'a';*/是正确的*/
c = "a";*/是错误的*/
c="CHINA"；也是错误的。不能把字符串赋给一个字符常量*/
```

C 规定：在每一个字符串的结尾加一个“字符串结束标志”，以便系统据此判断字符串是否结束。C 语言以字符'\0'作为字符串结束标志。如果有一个字符串"CHINA"，实际上在内存中是

C	H	I	N	A	\0

它的长度不是 5 个字符，而是 6 个字符，最后一个字符为'\0'。但在输出时不输出'\0'。例如在 printf("How do you do.")中，输出时字符一个一个输出，直到遇到最后的'\0'字符，就知道字符串结束，停止输出。注意，在写字符串时不必加'\0'，它是系统自动加上的。

在 C 语言中没有专门处理字符串的变量，字符串如果存放在变量中，需要用字符数组来存放，即用一个字符型数组来存放一个字符串。

（3）字符变量

字符数据类型以 char 表示，字符型变量用来存放字符，注意只能存放一个字符。

字符变量的定义形式如下：

```
char c1, c2;
```

它表示 c1 和 c2 为字符型变量，可以存放一个字符，因此可以用下面语句对其赋值：

```
c1='a';     c2='b';
```

一般以一个字节来存放一个字符，或一个字符变量在内存中占一个字节。

【例 2.5】定义字符变量，并赋值字符和整型数据，然后将其输出。

```
#include "stdio.h"
void main()
{
    char c1 , c2 ;
    c1='a' ; c2='b' ;
    printf("%c  %c  %d  %d ", c1, c2, c1, c2) ;
    c1=97 ; c2=98;
```

```
        printf("%c  %c  %d  %d ", c1, c2, c1, c2) ;
    }
```

【说明】

① 字符常量存放到一个字符变量中，实际上并不是把该字符本身存放到内存单元中去，而是将该字符的相应 ASCII 码值存放到存储单元中。如字符'a'的 ASCII 码值为 97，'b'为 98。

② 字符的存储形式与整型的相类似，使字符数据和整型数据之间可以通用。

③ 字符数据可以以字符形式输出，也可以以整数形式输出。以字符形式输出时，先将存储单元中的 ASCII 码转换成相应字符，然后输出。以整数形式输出时，直接将 ASCII 码值作为整数输出。

④ 可以对字符数据进行算术运算，此时相对于对它们的 ASCII 码进行算术运算。

【例 2.6】实现将小写字母转换成大写字母并求出下一个字母。

```
main()
{char c1,c2；              /*定义字符变量c1,c2*/
 c1='a'；                  /*将字符'a'赋值给变量c1*/
 c1=c1-32；                /*小写字母的ASCII码比大写字母的ASCII码大32*/
 c2=c1+1；                 /*相邻字符的ASCII码差1*/
 printf("\n%c  %c",c1,c2)；
 printf("\n%d  %d",c1,c2)；
}
```

【说明】

字符型数据与整型数据在 ASCII 码范围（0～255）内是通用的。

2.1.5 格式输出函数——printf()

printf()函数的作用是向终端输出若干个任意类型的数据（putchar()只能输出字符，而且只能输出一个字符，而 printf()可以输出多个字符，且为任意字符）。printf()的一般格式为：

```
printf(格式控制，输出列表)；
```

其中，“格式控制”是用双引号括起来的字符串，也称“转换控制字符串”，它包含信息格式字符（如%d，%f 等，如表 2.4 所示）和普通字符（需要原样输出的字符）。“输出列表”是一些与“格式字符”中的格式字符一一对应的需要输出的数据，可以是变量或表达式。

表 2.4 输出数据的格式字符表

格 式 字 符	描　　述
%d	按整型数据的实际长度输出
%md	m 为指定的输出字段的宽度(右对齐)，%-md 为左对齐
%ld	输出长整型数据
%o	以八进制形式输出整数
%x	以十六进制数形式输出整数
%u	输出 unsigned 型数据，即无符号数，以十进制形式输出
%c	输出一个字符
%s	输出一个字符串

续表

格 式 字 符	描　　述
%f	输出实数（包括单双精度），以小数形式输出（默认六位小数）
%m.nf	指定数据占 m 列，其中有 n 位小数。如果数值长度小于 m，左端补空格（右对齐）
%-m.nf	指定数据占 m 列，其中有 n 位小数。如果数值长度小于 m，右端补空格（左对齐）
%e	以指数形式输出实数
%g	输出实数，它根据数值的大小，自动选 f 格式或 e 格式（选择输出是占宽度较小的一种），且不输出无意义的零

【例 2.7】输出学生的姓名、年龄、学号、成绩、性别等信息。

```
#include "stdio.h"
void main()
{
    int age=19, num=23;
    float score=87.5;
    char sex='m';                /*f:女, m:男*/
    printf("Name is Rose\n");
    printf("ID is %d", num);
    printf("Age:%d\tSex:%c\tscore:%f\n", age, sex, score);
}
```

【说明】

① 不输出变量或表达式的值，直接输出一个字符串。例如 printf("Name is Rose\n")，其中 \n 是转义字符，表示回车换行。

② 格式化输出。语句 printf("ID is %d", num)输出 “ID is 23”。语句中"ID is %d"是格式控制部分，num 是输出列表。格式控制"ID is %d"中的%d 以十进制整数形式输出变量 num 中的值。

③ 将多个输出项放在一条输出语句中格式输出。语句 printf("Age:%d\tSex: %c\ tscore:%f\n", age, sex, score)将年龄、性别和成绩一起输出。"Age:%d\tSex: %c\tscore:%f\n"是格式控制部分，age、sex、score 是输出列表。其中%d、%c 和%f 是格式控制符，它们指明输出列表中变量 age 以十进制整数形式输出，变量 sex 以字符形式输出，变量 score 以浮点数形式输出，\t 是转义字符，表示将光标移到下一个位置的水平制表符，\n 表示回车换行，其余的字符按原样输出。

④ “格式控制”部分中的格式控制符与输出列表中变量或表达式要一一对应。

2.1.6 格式输入函数——scanf()

scanf()函数的作用是从键盘上输入若干个任意类型的数据（getchar()只能输入字符，而且只能输入一个字符，而 scanf()可以输出多个字符或其他类型的任意数据）。scanf()的一般格式为：

```
scanf(格式控制, 输出列表);
```

其中，“格式控制”的含义同 printf 函数，格式字符含义如表 2.5 所示。“地址列表”是由若干个地址组成的列表，可以是变量的地址。

表 2.5　输入数据的格式字符表

格 式 字 符	描　　述
%d	输入十进制整数
%o	输入八进制整数
%x	输入十六进制整数
%u	输入无符号十进制整数
%c	输入一个字符
%s	输入一个字符串
%f	以小数形式输入实型数
%e	以指数形式输入实型数

【例 2.8】输入学生的年龄、学号、成绩、性别等信息。

```
#include "stdio.h"
void main()
{
    int age, num;
    float score;
    char sex;                    /*f:女, m:男*/
    printf("input the information\n");
    scanf("%d%d%f%c", &age, &num, &score, &sex);
    printf("Age:%d\tID:%d\tSex:%c\tscore:%f\n", age, num, sex, score);
}
```

【说明】

① scanf("%d%d%f%c", &age, &num, &score, &sex)语句表示用户从键盘输入两个整数，一个浮点型数据，一个字符。其中"%d%d%f%c"是格式控制部分，&age, &num, &score, &sex 是地址列表部分，表示从键盘接收的两个整数第一个给变量 age，第二个给变量 num，接收的第三个浮点型数据给变量 score，接收的第四个字符数据给变量 sex。

② "%d%d%f%c"为格式字符，以%开始，以一个格式字符结束。

③ 变量前面加"&"符号，表示取变量的地址。例如&age 表示取变量 age 的地址。

④ 运行时输入数据，数据之间可以用空格或回车键分隔。

⑤ 如果在“格式控制”字符串中除了格式说明外还有其他字符，则在输入数据时应输入与这些字符相同的字符。

例如：scanf("%d, %d", &a, &b);　则输入时应用形式：3, 4（中间必须是逗号）

如果是：scanf("%d%d", &a, &b);　则输入时应用形式：3　4（中间用空格、回车符或 Tab 键）

⑥ scanf 函数中没有精度控制，如：scanf("%5.2f",&a); 是非法的。不能企图用此语句输入小数为 2 位的实数。

2.1.7　字符输出函数——putchar()

putchar 函数的作用是向终端输出一个字符，例如：

```
    putchar(c);
```

输出字符变量 c 的值。c 可以是字符型变量或整型变量。在使用标准 I/O 库函数时，要用预编译命令"#include"将“ stdio.h ”文件包含到用户源文件中。即

```
    #include "stdio.h"
```

stdio.h 是 standard input & output 的缩写, 它包含了与标准 I/O 库有关的变量定义和宏定义。在需要使用标准 I/O 库中的函数时，应在程序前使用上述预编译命令，但在使用 printf 和 scanf 函数时，则可以不要（只有 printf 和 scanf 例外）。

【例 2.9】观察下面程序输出的结果。

```
#include "stdio.h"
void main()
{
    char a, b, c, d;
    a='B';
    b='O';
    c='Y';
    d='a';
    putchar(a);
    putchar(b);
    putchar(c);
    putchar(d);             /*输出小写字母a*/
    d=d-32;
    putchar(d);             /*输出大写字母A*/
}
```

【说明】

① 程序运行结果：BOYaA。

② 可以输出控制字符，如 putchar('\n')输出一个换行符，也可以输出其他转义字符。

③ d=d-32 将 d 中的字符的 ASCII 码值取出减去 32 后再存放到 d 中。这是因为，大写字母和其相应的小写字母的 ASCII 码值相差 32，比如大写字母'A'的 ASCII 码值是 65，则小写字母'a'的 ASCII 的码值是 97。

2.1.8　字符输入函数——getchar()

getchar 函数的作用是从终端输入一个字符。getchar 函数没有参数，其一般格式为：

```
    ch=getchar();
```

ch 为一个字符型或整型变量，ch 的值就是从输入设备得到的字符。

【例 2.10】从键盘上输入一个字符，然后将其输出。

```
#include "stdio.h"
void main()
{
    char c;
    c=getchar();
```

```
        putchar(c);
    }
```

【说明】

① getchar()只能接收一个字符。

② getchar 函数得到的字符可以赋给一个字符变量或整型变量，也可以不赋给任何变量，作为表达式的一部分。例如上面代码中的第 5、6 行可以用下面一行代替:

```
    putchar(getchar());
```

任务 2.2　选手总分和平均分的计算

问题情景

举办校园歌手大赛，三名评委分别对每一名选手打分，计算选手所获得的总分和平均分。

实现过程

【例 2.11】(更新例 2.1)

```
#include "stdio.h"
void main()                 /* 函数名 */
{
    int f1,f2,f3,sum;       /* 定义3个整型变量存储评委打分，1个存储总分*/
    float ave;              /* 定义一个单精度浮点型变量存储平均分 */
    printf("\nf1:");        /* 提示输入打分 */
    scanf("%d",&f1);        /* 输入评分1 */
    printf("f2:");
    scanf("%d",&f2);
    printf("f3:");
    scanf("%d",&f3);
    sum=f1+f2+f3;           /* 计算总分 */
    ave=sum/3.0;            /* 计算平均分 */
    printf("\ntotal:%d\taverage:%.2f\n",sum,ave);      /* 输出计算结果 */
    getch();                /*用于读取按键的值。一般放在程序末尾，起到暂停的作用*/
}
```

程序运行结果如图 2.1 所示。

上面的程序可分析出：比例 2.1 多定义了两个实型变量 sum 和 avg，因为要将 3 个学生和总分放在 sum 中, 而 3 个学生的平均分放在 avg 中, 同时出现了 sum=f1+f2+f3;和 ave=sum/3.0; 语句，即出现了运算符和表达式。所以在本任务中，要掌握的知识点是：

① 算术运算符和算术表达式。

② 赋值运算符和赋值表达式。

③ C 语言其他的运算符和表达式以及优先级和结合性。

④ 复杂表达式中数据类型转换及转换规则。

相关知识

2.2.1　算术运算符和算术表达式

1. 基本算术运算符

基本算术运算符有：+（加）、-（减）、×（乘）、/（除）、%（求余）共 5 种。

各运算符的功能：前 3 种运算符我们已很熟悉，这里对另外两个需要特别提出：

（1）关于求除运算符： /

当两个整数相除时，结果为整数，小数部分舍去。如：5/2=2。

当两个实型数相除时，结果为实型，如：5.0/2.0=2.5。

如果商为负数，则取整的方向随系统而异。但大多数的系统采取“向零取整”原则，换句话说，取其整数部分。如：-5/3=-1。

（2）关于求余运算符： %

要求两个操作数必须都是整型数，否则出错。如：5%3=2；(-5) %3=-2；5%（-3）=2；(-5) %（-3）=-2；3%5=3 等等。但是 5.2%3 是语法错。

2. 表达式及算术表达式

（1）表达式

用运算符和括号将运算对象（变量、常量和函数）连接起来的符合 C 语言语法规则的式子。

单个变量、常量可以看作是表达式的一种特例。将单个变量、常量构成的表达式称作简单表达式，其他表达式称作复杂表达式。

（2）算术表达式

用算术运算符和括号将操作对象（即操作数）连接起来的式子，称为算术表达式。

例如：

a+b/c-2、sin(x)+1.5、x*y+z、1/2 都是合法的算术表达式。

特别提示：书写算术表达式时，一定要注意各种运算符的优先级和结合性，适当使用括号来保证原表达式的运算顺序。而且表达式中的各种符号均书写在同一行中，不能写在上角或下角，如 x2 或 x2 都是错误的 C 表达式。

例如，以下数学式子可以用“=>”号右边的 C 表达式表示。

$\dfrac{a+b}{a-b}$ => (a+b)/(a-b)

$\dfrac{a+b}{xy}$ => (a+b)/(x*y)

$3a+5\sin 2x$ => 3*a+5*sin(2*x)

3. 自增、自减运算符

自增（++）、自减（--）运算符是 C 语言最具有特色的两个单目运算符，操作对象只有一个，且必须是整型变量，其功能分别是使变量的值增 1、减 1。例如：

i++（先使用 i，然后使 i 的值增 1）

++i（先使 i 的值增 1，然后使用 i）

k--（先使用 k，然后使 k 的值减 1）

--k（先使 k 的值减 1，然后使用 k）

可见，自增、自减运算符既可以放在变量的左边（称为前缀用法），又可以放在变量的右边（称为后缀用法），但两者效果不同，使用时要特别小心。

【说明】

① ++i 和 i++异同点:

++i 和 i++的相同之处是单独使用加分号作为一个独立的句子时：即++i; 和 i++; 因为++i 和 i++的作用相当于 i=i+1。

但++i 和 i++不同之处在于++i 是先执行 i=i+1，再使用 i 的值；而 i++是先使用 i 的值，再执行 i=i+1，所以参与其他式子运算时，前后缀用法的使用效果就不同了。

② 自增运算符（++），自减运算符（--），只能用于变量，而不能用于常量或表达式，如 5++或(a+b)++都是不合法的。因为 5 是常量，常量的值不能改变。(a+b)++也不可实现，若 a+b 的值为 3，那么自增后得到的 4 无变量可存放。

【例 2.12】自增、自减运算符的使用

```
#include "stdio.h"
main()
{  int i=2,j=2,k=2,h=2,m,n,x,y;
   m=i++;  n=++j;   x=k--;   y=--h;
   printf("\ni=%d,m=%d,j=%d,n=%d",i,m,j,n);
   printf("\nk=%d,x=%d,h=%d,y=%d",k,x,h,y);getch();
}
```

运行结果如下：

```
i=3,m=2,j=3,n=3
k=1,x=2,h=1,y=1
```

【说明】

通过结果，可以看出自增、自减运算符的功能和前缀、后缀用法的不同。如“m= i++;”等价于“m=i; i++;”，即先取出变量 i 的值“2”，赋给变量 m，然后 i 的值增 1。其他的类似。

【例 2.13】思考如下程序段的输出结果是什么？

```
#include "stdio.h"
main()
   {   int a=100;
       printf("%d\t",a);
       printf("%d\n",++a);
```

```
        printf("%d\t",a++);
        printf("%d\n",a);
    }
```

【说明】

① 此程序注意四个 printf()语句之间的连贯性。

② 可以自行修改四个 printf()语句中的输出对象，然后再思考运行结果，从而充分理解++、--运算符的应用技巧。

4. 算术运算符的优先级和结合性

在计算表达式时，先按运算符的优先级由高到低运算，优先级相同时按运算符的结合性运算。结合性就是指级别相同的运算符的执行顺序。如表达式

```
2+6*3-8/2*7%3
```

含有多种运算符，如何计算呢？

先来看一下算术运算符的优先级（由高到低排列：①②③，同一组中级别相同）：

① ++、--、-（负号）

② *、/、%

③ +、-

结合性：双目运算符的结合方向为“自左向右”，单目运算符的结合方向为“自右向左”，如“-i++”等价于“-(i++)”。建议读者尽量不要连续使用++、--、-（负号）等单目运算符，以免出现歧义，必要时加括号以示直观。

按以上规则，计算上述表达式：①计算 6*3，结果 18。②计算 8/2，结果 4。③计算 4*7，结果 28。④计算 28%3，结果 1。⑤计算 2+18，结果 20。⑥计算 20-1，结果 19。

在 C 语言中，运算符的优先级有 15 级，1 级最高，15 级最低。在表达式中，优先级别较高的运算符先于优先级别较低的运算符。而在同一个运算量两侧的运算符优先级相同时，则按照结合性所规定的结合方向处理。C 语言中的各运算符的优先级和结合性，参见附录。

5. 数据类型转换

表达式运算中数据类型的转换有两种：一种是在运算时不必用户指定，系统自动进行类型转换，即隐式转换；另一种是强制类型转换，当自动类型转换不能实现目的时，可以用强制类型转换，即显式转换。

（1）隐式转换

C 语言允许整型、实型和字符型的数据之间进行混合运算，即在一个表达式中可以同时出现不同类型的数据。如表达式

```
4*`b`+2.5-123456.789/2
```

是合法的。运算时，按运算符的优先级顺序和结合性逐步计算，每一步计算之前，若参加运算的两个数类型不同，则先自动进行类型转换，然后再计算结果。

转换的原则是低类型数据转换成高类型数据，结果的类型与转换后的类型相同。各种数据类型的级别如图 2.4 所示。

【说明】

① 转换按数据长度增加的方向进行，以保证不降低精度。如 int 型和 long 型运算时，先把 int 型转换成 long 型后再计算。

② 所有的浮点运算都是以双精度进行的，即使仅含有 float(单精度)型运算的表达式，也要先转换成 double 型，再作运算。

③ char 型和 short 型参与运算时，将其先转换成 int 型。

④ 所需的转换均为系统自动进行类型转换，无需人为干预。

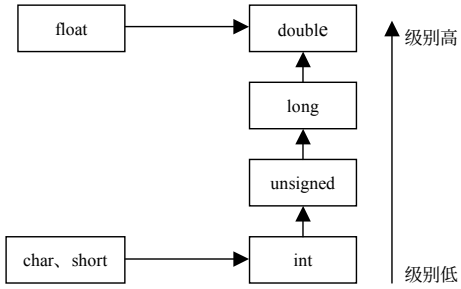

图 2.4　不同数据类型的自动转换规则

（2）强制类型转换

利用强制类型转换可以将一个表达式的值强制转换成指定的类型。这种转换是通过使用强制类型转换运算符实现的，因此又称为显式转换。

其一般形式为：

```
（类型名）（表达式）
```

其中，类型名是指定转换后的类型，必须用括号括起来。表达式是要转换的对象，一般也用括号括起来，只有当表达式是一个变量或常量时才可以省略括号。例如：

```
(int)(a+b)    表示将a+b的值转换成整型（但a+b本身的类型不变）
(double)8     表示想得到整型常量8转换成double型值（但8本身还是整型常量）
(folat)x      表示读取x的值并将其转换成float型（但x本身的类型不变）
```

【例 2.14】含有强制类型转换的表达式的计算。

```
     #include "stdio.h"
main()
{   int a=2,b=3;
          float x=3.5,y=2.5,z;
          z=(float)(a+b)/2+(int)x%(int)y;
          printf("\n%f",z);
          getch();
}
运行结果为：3.500000
```

【说明】

程序中表达式的执行过程：先将表达式 a+b 的值 5 强制转换成 double 型数 5.0；再计算表达式 5.0/2，结果为 double 型数 2.5；然后将 x、y 的值分别强制转换成 int 型数据 3、2；计算 3/2，结果为整型数 1；最后计算 2.5+1，结果为 double 型数 3.5。

强制类型转换符是单目运算符，它的优先级高于一般的算术运算符。

2.2.2　赋值运算符和赋值表达式

1. 赋值运算符

赋值符号“ = ”就是赋值运算符，它的作用是将一个数据赋给一个变量。其格式是：

```
<变量名>=<表达式>;
```

它的作用就是将右边表达式的值赋给左边的变量。如“ a=3 ”的作用是执行一次赋值操作。把常量 3 赋值给变量 a，也可以将一个表达式的值赋给一个变量。

2. 复合赋值运算符

在赋值符“ = ”之前加上其他运算符，可以构成复合的运算符。复合算术运算符的格式是：

```
<变量名> <基本算术运算符>=<表达式>；
```

它等价于：

```
<变量名>=<变量名> <基本算术运算符> <（表达式）>
```

C 语言采用这种复合运算符，一是为了简化程序，使程序精炼；二是为了提高编译效率，产生质量较高的目标代码。常见的符号运算符见表 2.6。

表 2.6　复合运算符

运　算　符	例　　子	等　价　于	运　算　符	例　　子	等 价 于
+=	x+=5	x=x+5	/=	x/=5	x=x/5
-=	x-=5	x=x-5	%=	x%=y+5*z	x=x%(y+5*z)
*=	x*=y+3	x=x*(y+3)			

【例 2.15】运行下面的程序，观察并分析用法。

```
#include "stdio.h"
void main()
{
    int a, b, c, x, y ;
    a=2;
    c=3;
    b=2*a+6;
    c*=a+b;
    x=a*a + b + c;
    y=2*a*a*a+3*b*b*b+4*c*c*c;
    printf("%d %d %d %d %d", a, b, c, x, y);
}
```

【说明】

① 此处的表达式 y=2*a*a*a+3*b*b*b+4*c*c*c 不可以写成 y=2aaa+3bbb+4ccc。

② 可以用赋值表达式同时对多个变量赋同样的值，如 a=b=c= 3，表示同时将 3 赋给变量 a、b 和 c，相当于 a=3，b=3，c=3。

③ 赋值运算符的结合方向是“自右向左”，即从右向左计算。如 a=b=c=3*2，先计算 c=3*2，再计算 b=c，最后计算 a=b。