第 2 章

顺 序 结 构

本章要点

1. 整型和浮点型；
2. 符号常量；
3. 数据的输入、输出；
4. 数学函数的应用。

2.1 数 据 表 达

计算机程序是一组计算机能识别和执行的指令，程序及其运行过程中所用到的数据都存储在计算机中。因此，了解计算机中数据表达的方式，对编写程序是很有必要的。

【例 2-1】统计某本畅销的 C 语言教材第一季度的平均销售数量。

解题思路

第 1 季度有 1 月、2 月和 3 月，可以假设这 3 个月的销售量及这 3 个月的平均销售量分别为 numJan、numFeb、numMar、numAvg。

需要注意的是：numJan、numFeb 和 numMar 的数值都由用户从键盘输入，可以使用 C 语言提供的 scanf( ) 函数来获得键盘输入的值；计算得到的结果需要展示出来，可以通过 printf( ) 函数将结果输出到屏幕函数将结果输出到屏幕。那此过程所用到的数据在计算机内如何表示，程序的整个计算过程是如何实现的？具体步骤分析如下。

数据分析

例 2-1 涉及的数据主要有：第 1 季度 3 个月的销售量和平均销售量，需要 4 个变量来保存这 4 个数据。销售数量是整数，因此，这 4 个数据需要用 int 类型来表示：

```
int numJan;          //表示 1 月份的销售量
int numFeb;          //表示 2 月份的销售量
```

```
int numMar;          //表示 3 月份的销售量
int numAvg;          //第一季度平均销售量
```

由于这 4 个数据都是 int 类型，因此，上述数据也可以用下列简单形式进行表达：

```
int numJan, numFeb, numMar, numAvg;     //各数据之间用逗号隔开
```

解题流程

输入第 1 季度 3 个月的销售量 numJan、numFeb 和 numMar，根据 numAvg=（numJan+numFeb+numMar）/3 计算第 1 季度的平均销售量，相应流程图如图 2.1 所示。

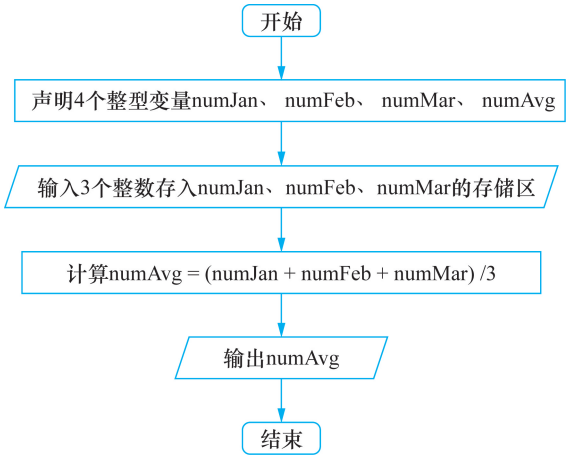

图 2.1　计算第 1 季度的平均销售量流程图

程序代码实现

```
#include<stdio.h>
int main()
{
  int numJan;          //1 月份的销售量
  int numFeb;          //2 月份的销售量
  int numMar;          //3 月份的销售量
  int numAvg;          //第一季度的平均销售量
  printf("请输入第 1 季度 3 个月的销售量:");//输出提示性语句
  scanf("%d %d %d", &numJan, &numFeb, &numMar);//以空格分隔输入 3 个整数
  numAvg=(numJan + numFeb + numMar)/3;
  printf("第 1 季度的平均销售量是:%d\n", numAvg);
  return 0;
}
```

【运行结果】

```
请输入第 1 季度 3 个月的销售量:200 190 205
第一季度的平均销售量是:198
```

思考

第 1 季度销售总额是 595，595 除以 3 的结果按照数学计算应该是 198.333…。为什么程序最后输出的结果只有整数部分的 198 呢？如果想要按照数学计算结果输出包含小数部分，应该怎么做呢？

从例 2-1 中，读者可以发现，编写一个典型的顺序结构程序主要包含 4 个步骤：

（1）变量声明；

（2）获取初始状态的数据；

（3）根据需求进行运算；

（4）输出运算结果。

变量声明，也就是对使用到的数据对象先起名字（标识符）并确定数据的类型。C 语言常用的基本数据类型有整型、浮点型和字符型。本节主要介绍整型和浮点型，字符型将第　章介绍。

2.1.1　标识符

一个 C 语言程序可以看成是一系列字符序列组成的标识符（如变量、函数等）、运算符（如+、-、*、/等）、分隔符（如；和，）等元素组成。

C 语言的标识符（Identifier）只能由字母（A～Z，a～z）、数字（0～9）和下画线（_）组成，并且第一个字符必须是字母或下画线。例如：score_C、_stuName、numAvg 都是合法标识符，而 1num 和 in $ out 都是非法标识符。

另外，C 语言还区分大小写（例如，stu 和 Stu 被认为是两个不同的标识符）。

C 语言的标识符主要分为以下 3 类。

（1）关键字：又称为保留字，是在 C 语言中有固定含义和专门用途的标识符，不能用来给变量、函数等数据对象命名。例如，用来说明数据类型的标识符 int、char，以及控制程序流程的 if、else、for、while 等。

（2）预定义标识符：这类标识符在 C 语言中也有特定含义，例如 C 语言提供的库函数名字（printf、scanf 等）和预处理命令（define、include 等）。虽然 C 语言语法允许用户把这类标识符另作它用，但为了维持这类标识符的原有含义和避免不必要的麻烦，建议不要将预定义标识符改作其他用途。

（3）用户自定义标识符：由用户根据需求定义的标识符，一般用来给变量、函数等数据对象命名。除了要遵循命名规则，还应做到“见名知意”。

2.1.2　整型

整型是不带小数点的数据类型。按照表示范围，整型可分为整型（int）、短整型（short int）和长整型（long int）。按照数值有无正负之分，整型又可分为有符号整型（signed）和无符号整型（unsigned）。在存储空间中，有符号整型的最高比特位用0表示正数，用1表示负数；而无符号整型的最高位也是有效数字位。

不同的编译系统为int类型分配的存储空间可能存在差异，可以使用长度运算符sizeof来判断一个对象（类型、变量等）占用的字节数。

【例2-2】各整型类型占用的字节数及其能够表示最小值和最大值。

解题思路

可以使用sizeof运算符计算各整数类型占用的字节数。在limits.h文件中定义的宏INT_MIN和INT_MAX分别指的是int类型的最小值和最大值，宏SHRT_MIN和SHRT_MAX分别指的是short int类型的最小值和最大值，宏USHRT_MAX是指unsigned short int类型的最大值，宏UINT_MAX指的是unsigned int类型的最大值。

程序代码实现

```
#include <stdio.h>
#include <limits.h>
int main()
{
  printf("int 类型占用字节数:%d\n",sizeof(int));
  printf("short int 类型占用字节数:%d\n",sizeof(short int));
  printf("long int 类型占用字节数:%d\n",sizeof(long int));
  printf("整型最小值: %d, 整型最大值: %d\n", INT_MIN, INT_MAX);
  printf("短整型最小值: %d, 短整型最大值: %d\n", SHRT_MIN, SHRT_MAX);
  printf("无符号短整型最大值: %u, 无符号整型最大值: %u\n", USHRT_MAX,
UINT_MAX);
  return 0;
}
```

【运行结果】

```
int 类型占用字节数:4
short int 类型占用字节数:2
long int 类型占用字节数:4
整型最小值:-2147483648,整型最大值:2147483647
短整型最小值:-32768,短整型最大值:32767
无符号短整型最大值:65535,无符号整型最大值:4294967295
```

思考

比较 int、unsigned short int 和 unsigned int 3 个整数类型的最大值，思考这 3 个类型最大值在计算机中的二进制形式是怎样的？

表 2.1　各类整型数据的相关信息

类型名称	类型标识符	所占字节数	数值范围
有符号基本整型	[signed] int	4	-2 147 483 648～2 147 483 647
有符号短整型	[signed] short [int]	2	-32 768～32 767
有符号长整型	[signed] long [int]	4	-2 147 483 648～2 147 483 647
无符号基本整型	unsigned [int]	4	0～4 294 967 295
无符号短整型	unsigned short [int]	2	0～65 535
无符号长整型	unsigned long [int]	4	0～4 294 967 295

表 2.1 列出各类整型数据所分配的字节数及数值的表示范围，表格中各类型“ [ ] ”中的内容可以省略。例如，short 类型的数值范围是-32768～32767，如果一个 short 类型的数据在运算过程中超过了这个范围，会发生什么情况呢？

【例 2-3】整型数据溢出的情况演示。

解题思路

可以使用两个 short 类型数据 max 和 min，其中 max 的值为 short 类型数值范围允许的最大值 32767，min 的值为 short 类型数值范围允许的最小值-32768。然后 max 做+1 的运算，min 做-1 的运算，看看输出的结果是怎样的。

程序代码实现

```
#include<stdio.h>
int main()
{
  short max=32767, min=-32768;
  printf("初始时 max 和 min 的值分别为:%d, %d\n", max, min);
  max=max + 1;
  min=min - 1;
  printf("计算后 max 和 min 的值分别为:%d, %d\n", max, min);
  return 0;
}
```

【运行结果】

```
初始时 max 和 min 的值分别为:32767, -32768
计算后 max 和 min 的值分别为:-32768, 32767
```

思考

执行语句“max = max +1;”，按数学计算，max 的值应该为 32768；同样地，语句“min = min -1;”的数学计算结果 min 的值应该是-32769。但实际程序的运行结果却不是这样。请读者分析一下产生这样运行结果的原因是什么。

2.1.3　浮点型

浮点型（实型）是指带小数点的数据类型。在 C 语言中，浮点型可分为单精度型（float）和双精度型（double）。系统一般为 float 类型数据分配 4 个字节的存储空间，为 double 类型数据分配 8 个字节的存储空间，因此，double 类型数据占用的存储空间更大，精度更高。

【例 2-4】浮点型数据的应用：输入一个圆柱体的高度和底面圆半径，计算圆柱体底面圆面积、侧面积和体积。

解题思路

假设圆柱体底面圆半径为 radius，高度为 height，底面圆面积为 areaBottom，侧面积为 areaSide，体积为 volume，然后根据相应的公式进行计算。

数据分析

本例涉及的数据主要有圆柱体的底面半径、底面面积、高度、侧面面积和体积，需要 5 个变量来保存这 5 个数据。这些数据可以是浮点数，因此，这 5 个数据需要用 float 类型来表示：

```
float radius;           //表示圆柱体的底面半径
float height;           //表示圆柱体的高
float areaBottom;       //表示圆柱体的底面面积
float areaSide;         //表示圆柱体的侧面面积
float volume;           //表示圆柱体的体积
```

由于这 5 个数据都是 float 类型，因此，上述数据也可以用下列简单形式进行表达：

```
float radius, height, areaBottom, areaSide, volume;//各数据之间用逗号隔开
```

本例中还有一个圆周率 π 需要表示，可以先用 3.14 这个数值来简单表示，本书将在 2.1.4 节中介绍如何使用宏定义的方式来表示 π。

解题流程

输入半径 radius 和高度 height，根据相应公式进行计算：areaBottom = π * radius×radius，areaSide = 2 * π * radius * height，volume = areaBottom * height。

程序代码实现

```
#include <stdio.h>
int main()
{
  float radius, height, areaBottom, areaSide, volume;
  printf("请输入圆柱体的底面半径和高度:");  //输出提示性语句
  scanf("%f, %f", &radius, &height);      //以逗号分隔,输入两个浮点数
  areaBottom=3.14*radius*radius;          //计算圆柱体的底面积
  areaSide=2*3.14* radius*height;         //计算圆柱体的侧面积
  volume=areaBottom*height;               //计算圆柱体的体积
  printf("圆柱体底面积为:%f\n", areaBottom);//输出圆柱体的底面积
  printf("圆柱体侧面积为:%f\n", areaSide);//输出圆柱体的侧面积
  printf("圆柱体体积为:%f\n", volume);     //输出圆柱体的体积
  return 0;
}
```

【运行结果】

```
请输入圆柱体的底面半径和高度:3, 10
圆柱体的底面积为:28.260000
圆柱体的侧面积为:188.399994
圆柱体的体积为:282.600006
```

思考

底面半径为3，高度为10，按照公式计算圆柱体的侧面积应该是188.4，为什么输出的结果却是188.399994？读者可以试试看将5个变量的类型改为double，将scanf( )函数中的两个格式字符%f改为%lf，再次输入3和10，比较一下运行结果的区别。

2.1.4 变量和常量

1. 变量

变量是指运行过程中值可以变化的量。在使用变量前，需要给其指定类型和名字（变量声明），其一般形式为：

```
类型名 变量名列表;
```

例如，在例2-1中，1月份销售量的声明：

```
int numJan;
```

如果声明多个同一类型的变量，可以用“,”隔开各变量名。例如，在例2-3中，声明圆柱体的底面半径、高度、底面面积、侧面面积和体积这5个变量：

```
float radius, height, areaBottom, areaSide, volume;
```

2. 常量

在程序运行过程中，其值不能被改变的量称为常量。根据类型划分，常量可分为整型常量、浮点型常量、字符常量和字符串常量。另外，常量还可以分为直接常量和符号常量。例如，100、-1 都是整型常量，.23、3.14 和 1.5e-2 都是浮点型常量。符号常量需要用#define 的方式进行定义，其定义形式如下：

```
#define  宏名  常量
```

上面这个#define 命令中，“宏名”与“常量”之间要有一个空格，程序将使用这个“宏名”来代表“常量”的值。例如：

```
#define PI 3.14
```

上述定义中，定义了一个宏名 PI，表示的值是 3.14。

在例 2-4 中，程序代码有两处使用了 3.14 这个常量值来表示圆周率，也可以使用宏定义的方式来表示，具体如下。

【例 2-5】使用宏定义表示圆周率，改写例 2-4 的程序。

程序代码实现

```
#include <stdio.h>
#define PI 3.14//定义了一个宏名 PI,表示的值为 3.14
int main()
{
  float radius, height, areaBottom, areaSide, volume;
  printf("请输入圆柱体的底面半径和高度:");
  scanf("%f, %f", &radius, &height);
  areaBottom=PI*radius*radius;          //此处出现了宏名 PI
  areaSide=2*PI*radius*height;          //此处出现了宏名 PI
  volume=areaBottom*height;
  printf("圆柱体的底面积为:%f\n", areaBottom);
  printf("圆柱体的侧面积为:%f\n", areaSide);
  printf("圆柱体的体积为:%f\n", volume);
  return 0;
}
```

在例 2-4 和例 2-5 的程序中，使用 3.14 来表示圆周率。如果需要增加圆周率的精度，使用 3.1415926 这个数值，程序可以怎么写呢？

【例 2-6】使用 3.1415926 表示圆周率的值，改写例 2-4 的程序。

方法一：修改例 2-4 程序中各处圆周率的值。

程序代码实现

```
#include <stdio.h>
int main()
{
  float radius, height, areaBottom, areaSide, volume;
  printf("请输入圆柱体的底面半径和高度:");
  scanf("%f, %f", &radius, &height);
  areaBottom=3.1415926*radius*radius;      //此处出现了圆周率
  areaSide=2*3.1415926*radius*height;      //此处出现了圆周率
  volume=areaBottom*height;
  printf("圆柱体的底面积为:%f\n", areaBottom);
  printf("圆柱体的侧面积为:%f\n", areaSide);
  printf("圆柱体的体积为:%f\n", volume);
  return 0;
}
```

方法二：修改例 2-5 程序中#define 命令中宏名表示的常量值。

程序代码实现

```
#include <stdio.h>
#define PI 3.1415926//宏名 PI,表示的值为 3.1415926
int main()
{
  float radius, height, areaBottom, areaSide, volume;
  printf("请输入圆柱体的底面半径和高度:");
  scanf("%f, %f", &radius, &height);
  areaBottom=PI*radius*radius;        //此处出现了宏名 PI
  areaSide=2*PI*radius*height;        //此处出现了宏名 PI
  volume=areaBottom*height;
  printf("圆柱体的底面积为:%f\n", areaBottom);
  printf("圆柱体的侧面积为:%f\n", areaSide);
  printf("圆柱体的体积为:%f\n", volume);
  return 0;
}
```

思考

从例 2-5 和例 2-6 中可以看出，编写程序的时候可以直接使用宏名来表示其相应的常

量值，而且当程序要修改这个常量数值时，只需修改#define 中宏名代表的数值即可。代码中的有特定意义的常量用宏名表示，代码的可读性也更好。

2.1.5　算术运算符和赋值运算符

C 程序对数据的操作，需要用相应的运算符将数据连接成符合语法规则的表达式。C 语言具有功能丰富的运算符，这些运算符按照运算形式可以分成 13 类。本节主要介绍常用的算术运算符和赋值运算符。

1. 算术运算符

C 语言的基本算术运算包含 5 个双目运算符：+（加）、-（减）、*（乘）、/（除）和%（求余）。双目运算是指要求有 2 个操作数参与运算。使用算术运算符和一对圆括号将运算量连接起来的符合 C 语言语法规则的表达式称为算术表达式，其结果一定是整型或浮点型。

在使用这几个算术运算符的过程中，需要注意以下几点。

（1）这 5 个算术运算符的结合性都是自左向右，其中 *、/和%的优先级高于+、-运算符；有圆括号时，圆括号内的表达式先计算。

例如：表达式 1+2/2.0 的结果为 2.0，而表达式（1 + 2）/ 2.0 的结果为 1.5。

（2）做/运算时，整数除以整数的结果还是整数。

例如：表达式 1/2 的结果为 0。

回顾例 2-1 中的思考问题，现在是否明白了为什么 595 除以 3 的输出结果是 198，而不是 198.333…呢?

（3）%运算要求参与的两个运算量均为整型。

（4）+、-、*和/这 4 个运算符，当两个运算量的数据类型相同时，运算结果的类型和运算量类型相同；当两个运算量类型不同时，系统会先自动进行类型转换，使得两个运算量类型相同后再进行运算。自动转换的规则如图 2.2 所示，横向箭头表示必定的转换，例如字符型 char 和短整型 short，必须先转换成 int 类型，而单精度浮点型 float 也要先转换成双精度型 double，以保证运算精度。纵向箭头表示不同类型数据进行混合运算时的转换方向。

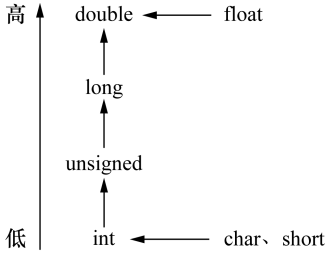

图 2.2　数据类型自动转换规则

例如：表达式 1/2.0，系统先自动将其转换为 1.0/2.0，然后计算得到结果 0.5。

（5）运算的过程中，除了系统会自动进行类型转换，C 语言还提供了强制类型转换符“（类型）”，其一般形式为：（类型）表达式，含义是将表达式转换成圆括号内标明的类

型。例如，表达式 1%2.5 是非法的，因为求余运算%要求运算量均为整型，而 2.5 是浮点数。而表达式 1%（int）2.5 是正确的，结果为 1。在表达式 1%（int）2.5 中，先用强制类型转换运算将 2.5 的类型转换为 int 类型。

按照算术表达式的计算规则，如果希望例 2-1 中 595 除以 3 的输出结果是 198.333…，程序应该怎么写呢？

【例 2-7】改写例 2-1 的程序，使得输出的结果为浮点数。

方法一：将例 2-1 程序中各变量的类型改为浮点型。

程序代码实现

```
#include<stdio.h>
int main()
{
  float numJan, numFeb, numMar, numAvg;//变量均为浮点型
  printf("请输入第 1 季度 3 个月的销售量:");
  scanf("%f %f %f", &numJan, &numFeb, &numMar);
  numAvg=(numJan + numFeb + numMar)/3;
  printf("第 1 季度的平均销售量是:%f\n", numAvg);
  return 0;
}
```

【运行结果】

```
请输入第 1 季度 3 个月的销售量:200 190 205
第 1 季度的平均销售量是:198.333328
```

思考

为什么 scanf( ) 函数和第二个 printf( ) 函数里的格式控制字符要改成%f 呢？另外，为什么输出的结果是 198.333328？如果输入的 3 个月销售量仍然是 200、190 和 205，希望得到输出结果为 198.333333，应该怎么修改程序呢？

方法二：计算平均销售量时用 3.0 作除数。

程序代码实现

```
#include<stdio.h>
int main()
{
  int numJan;          //1 月份的销售量
  int numFeb;          //2 月份的销售量
  int numMar;          //3 月份的销售量
  float numAvg;        //第 1 季度的平均销售量,变量 numAvg 的类型为浮点型
```

```
    printf("请输入第 1 季度 3 个月的销售量:");
    scanf("%d %d %d", &numJan, &numFeb, &numMar);//以空格分隔输入 3 个整数
    numAvg=(numJan + numFeb + numMar)/3.0;
    printf("第 1 季度的平均销售量是:%f \n", numAvg); //注意输出格式字符为%f
    return 0;
}
```

【运行结果】

```
请输入第 1 季度 3 个月的销售量:200 190 205
第 1 季度的平均销售量是:198.333328
```

思考

在这个程序中，numJan、numFeb 和 numMar 仍然是整型，但为什么表达式（numJan + numFeb + numMar）/3.0 的结果为浮点型？要注意方法二的程序中，numAvg 的类型为 float。

方法三：使用强制类型转换。

程序代码实现

```
#include<stdio.h>
int main()
  {
  int numJan;          //1 月份的销售量
  int numFeb;          //2 月份的销售量
  int numMar;          //3 月份的销售量
  float numAvg;        //第 1 季度的平均销售量
  printf("请输入第 1 季度 3 个月的销售量:");//输出提示性语句
  scanf("%d %d %d", &numJan, &numFeb, &numMar);//以空格分隔输入 3 个整数
  numAvg=(float)(numJan + numFeb + numMar)/3;
  printf("第 1 季度的平均销售量是:%f \n", numAvg);
  return 0;
}
```

【运行结果】

```
请输入第 1 季度 3 个月的销售量:200 190 205
第 1 季度的平均销售量是:198.333328
```

思考

在方法三的程序中，numJan、numFeb 和 numMar 仍然是整型，但用了强制类型转换运算（float）（numJan+numFeb+numMar）/3，将 numJan+numFeb+numMar 的结果先转换为

float 类型后再除以 3。这个语句还可以写成：numAvg = (numJan + numFeb + numMar)/(float) 3；但不能写成 numAvg = (float)((numJan+numFeb+numMar)/3)。请读者分析一下，为什么不能这样写？如果这样写，程序运行结果是怎样的？

2. 赋值运算符

C 语言中，符号“=”称为赋值运算符。赋值表达式是由赋值运算符将一个变量和一个表达式连接起来的式子，其一般形式如下：

```
变量 = 表达式
```

赋值表达式的作用是将右侧表达式的值赋给左侧的变量。需注意的是，C 语言中的“=”与数学中的“=”是不一样的。例如，x-y=1 在数学中是正确的，表示 x-y 的结果为 1；但是在 C 语言中，x-y=1 不符合语法规则，是一个非法表达式，因为“=”的左侧不是一个变量。

赋值运算符“=”还可以与算术运算符（+、-、*、/、%）、位运算符（<<、>>、|、^、&）构成复合赋值运算符。例如，复合赋值表达式 x*=y-1，等价于表达式 x=x*(y-1)。

在使用赋值运算符时，需要注意以下几点。

（1）赋值运算符的优先级只高于逗号运算符，低于其他任何运算符，其结合性为自右向左。

（2）赋值表达式的计算过程是先求赋值运算符右侧表达式的值，然后将求出的值赋给左侧的变量。

（3）当赋值运算符左右两侧的数据类型不相同时，根据左侧变量的类型，系统会自动将右侧的类型转换成左侧的类型后再赋值。

例如，有变量 i 声明为整型：“int i;”则赋值表达式“i = 1.5”中，虽然右侧的值是浮点型，但左侧变量 i 是整型，因此，将右侧 1.5 的数值转为整型，即取数值中的整数部分赋给变量 i，因此 i 的值为 1。

【例 2-8】复合赋值表达式 num+=num-=num*=num 的计算过程。

解题思路

赋值运算符的计算顺序是从右到左，因此表达式“num+=num-=num*=num”中，首先进行的计算是 num*=num，该表达式包含复合赋值运算符 *=，因此等价于 num=num*num；然后进行的计算是 num-=num；最后计算 num+=num。

程序代码实现

```
#include <stdio.h>
int main()
{
  int num;
  printf("请输入 num 的值:");
```

```
    scanf("%d", &num);
    num +=num -=num *=num;
    printf("运算后 num 的值为:%d", num);
    return 0;
}
```

【运行结果】

```
请输入 num 的值:3
运算后 num 的值为:0
```

2.2　数据的输入

C 语言标准 I/O 函数库中提供了相关的函数来实现数据的输入，例如例 2-1 和例 2-4的程序中使用的 scanf( ) 函数。scanf( ) 函数在 stdio. h 文件中声明，要使用 scanf( ) 函数需要在源程序开头使用编译预处理命令#include <stdio. h>。

从前面的例子可以看出，scanf( ) 函数按照指定的格式从键盘上接收输入的数据，并存入指定的存储单元，其使用的一般形式如下：

```
scanf("格式控制字符串", 地址表列);
```

格式控制字符串用%符号开头，其形式：

```
% [*][输入数据宽度][长度]类型
```

（1）格式控制字符串中，类型是必不可少的部分，[ ] 内的 * 输入数据宽度和长度是可选项。

常见的类型说明见表 2. 2 所列：

表 2. 2　scanf（）输入数据类型

类型说明符	含义
d、i	输入有符号的十进制整数
u	输入无符号的十进制整数
o	输入无符号的八进制数
x、X	输入无符号的十六进制整数
c	输入单个字符
s	输入字符串
f、e、E、g、G	以小数形式或指数形式输入浮点数

[ ] 内的可选项格式控制字符的说明见表 2.3 所列。

表 2.3　可选项格式控制字符

附加说明符	含义
*	表示对应的输入项在读入后不保存，不需要为其指定地址参数
m（正整数）	指定输入数据所占的域宽（列数）
l、L	加在 d、i、u、o、x 前，表示输入长整型数据； 加在 f、e 前，表示输入双精度类型数据
h、H	加在 d、i、u、o、x 前，表示输入短整型数据

例如：

```
scanf("%d %*d %d", &a, &b);
```

当输入 1 2 3，则把 1 赋值给 a，2 输入后跳过，把 3 赋值给 b。

```
scanf("%3d", &x);
```

当输入 123456，只把 123 赋值给 x，剩余数据 456 会在键盘中缓存，若后续还有接收输入，将会从缓存中获取数据。因此，如果有

```
scanf("%3d%3d", &x, &y);
```

当输入 123456，那么系统会把 123 赋值给 x，把 456 赋值给 y。

（2）使用 scanf（）函数时需要注意以下几点。

① 地址表列中，需要给出的是变量的地址，而不是变量名。如有“int x”，则“scanf("%d", x);”是错误的，正确用法为“scanf("%d", &x);”

② 在输入多个数值型数据时，如果格式控制字符串中没有非格式字符作为输入数据之间的间隔，则在输入数据时可以用空格键、回车键或者 Tab 键作为间隔。例如：

```
int a,b,c;
scanf("%d%d%d",&a,&b,&c);
```

可输入的方式有：3 4 5

或者：3 4↙

5

③ 若格式控制字符串中有非格式字符，则输入时也要输入该非格式字符。例如：

```
scanf("%d,%d,%d", &a,&b,&c);
```

格式控制字符串中包含了“,”，则在输入数据时应输入：3，4，5

④ scanf( )函数没有精度控制，如“float x; scanf("%5.2f", &x);”是错误的，不能通过此语句输入一个小数部分只有 2 位的浮点数。

【例 2-9】以 YYYY-MM-DD HH：MM：SS 形式输入日期和时间。

解题思路

日期和时间都是整数值，数据输入可以用%d 的格式控制字符串。输入数据时，要注

意日期数值之间有“-”分隔符，时间数值之间有“:”，日期和时间之间还有一个空格。“-”“:”和空格都作为 scanf( ) 函数的非格式符输入。

程序代码实现

```
#include <stdio.h>
int main()
{
  int year, mon, day, hour, min, sec;
  printf("请以 YYYY-MM-DD HH:MM:SS 形式输入日期和时间\n");
  scanf("%d-%d-%d %d:%d:%d", &year, &mon, &day, &hour, &min, &sec);
  printf("输入的日期为:%d-%d-%d\n", year, mon, day);
  printf("输入的时间为:%d:%d:%d\n", hour, min, sec);return 0;
}
```

【运行结果】

```
请以 YYYY-MM-DD HH:MM:SS 形式输入日期和时间
2022-1-1 10:20:30
输入的日期为:2022-1-1
输入的时间为:10:20:30
```

2.3　数据的输出

stdio.h 文件还包含格式化输出函数 printf（）的声明。printf( ) 函数按照指定的格式将数据输出到显示器，其使用的一般形式如下：

```
printf("格式控制字符串", 输出表列);
```

格式控制字符串用%符号开头，其形式：

```
% [标志][输出最小宽度][. 精度][长度]类型
```

格式控制字符串中，类型是必不可少的部分，[ ] 内的标志字符、输出最小宽度、精度字符和长度是可选项。

（1）常见的类型说明见表 2.4 所列。

表 2.4　printf( ) 输出数据类型

格式说明符	含义
d，i	输出有符号的十进制整数
u	输出无符号的十进制整数

续表

格式说明符	含义
o	输出无符号的八进制数
x、X	输出无符号的十六进制整数
c	输出单个字符
s	输出字符串
f	以小数形式输出单、双精度浮点数
e、E	以指数形式输出单、双精度浮点数
g、G	以%f%e 中较短的输出宽度输出单、双精度实数，不输出小数点后无意义的 0

读者可以将例 2-4 中的 printf( ) 函数的格式控制字符串的类型由%f 改为%e 和%g，对比输出结果的差异。

（2）输出最小宽度，用十进制整数来表示输出的最少位数。若实际位数多于定义的宽度，则按实际位数输出，若实际位数少于定义的宽度则左补空格或 0。

（3）标志字符有 4 种，其说明见表 2.5 所列。

表 2.5　printf( ) 标志字符

标志字符	含义
-	输出结果左对齐，右边填空格
+	输出符号（正号或负号）
空格	输出值为正时冠以空格，为负时冠以负号
#	对 c、s、d、u 类无影响；对 o 类，在输出时加前缀 0；对 x 类，在输出时加前缀 0x；对 e、g、f 类，当结果有小数时才给出小数点

（4）精度控制符以“.”开头，后跟一个十进制整数。如果输出的是浮点数，则表示输出小数的位数；如果输出的是字符串，则表示输出的字符的个数。若实际位数大于给定的精度位数，则截去超出的部分，浮点数的小数部分还需要四舍五入。

（5）长度有“l”和“h”两个说明符，“l”表示输出长整型数据，“h”表示输出短整型数据。

【例 2-10】整型数据 12345 的格式化输出。

解题思路

整型数据的输出主要用到%d 的格式控制，可以配合最小输出宽度、标志字符等进行格式化输出。

程序代码实现

```
#include <stdio.h>
int main()
{
  int x=12345;
  printf("x=%d;\n", x);
  printf("x=%+d;\n", x);
  printf("x=% d;\n", x); //注意%和d之间有一个空格
  printf("x=%10d;\n", x);
  printf("x=%+10d;\n", x);
  printf("x=%-10d;\n", x);
  printf("x=%-+10d;\n", x);
  printf("x=%x;\n", x);
  printf("x=%#10x;\n", x);
  return 0;
}
```

【运行结果】

```
x=12345;
x=+12345;
x= 12345;
x=     12345;
x=    +12345;
x=12345     ;
x=+12345    ;
x=3039;
x=    0x3039;
```

思考

对照输出格式的说明，理解程序运行结果。读者可以参照样例编写程序，完成浮点型数据和字符串数据的格式化输出。

2.4　数学函数

C语言编译系统提供了功能丰富的库函数，编程人员只需要通过include命令将这些函数的说明文件包含到源程序中就可以调用相关函数。例如，math.h文件中包含了许多数

学函数的声明，包括平方根函数（sqrt( )）、三角函数（sin( )，cos( )，tan( )）、反三角函数（asin( )，acos( )，atan( )）、指数与对数函数（exp( )，log( )，log10( )）、绝对值函数（abs( )，fabs( )）等。常用数学函数功能和使用说明见附录 G。

【例 2-11】任意输入一个 x 的值，根据公式 $y=\dfrac{x^3-\sqrt{|x|}}{2x}$，计算 y 的值。

解题思路

本题需要两个变量 x 和 y，两个变量可以用 float 类型来表示，其中 x 的值由用户从键盘输入。根据公式，可以用 fabs 函数求 x 的绝对值，用 sqrt 函数求平方根，用pow函数求 x 的三次方。使用这些数学函数，需要包含 math. h 文件。

程序代码实现

```
#include <stdio. h>
#include <math. h>
int main( )
{
  float x, y;
  printf("请输入 x 的值:");
  scanf("%f", &x);
  y=(pow(x, 3) -sqrt(fabs(x))) / (2*x);
  printf("y 的值为:%f", y);
  return 0;
}
```

【运行结果】

```
请输入 x 的值:4
y 的值为:7. 750000
```

思考

求绝对值的函数有两个：abs( )和 fabs( )，其中 abs( )函数是对整数求绝对值，而 fabs( )函数是对浮点数求绝对值。本例中，x 的类型为 float，所以应使用 fabs 函数；C 语言不存在上下标的写法，因此数学上的 x^3 可以使用数学函数 pow（x，3）或者 x * x * x 来表示，但是如果幂次较大时用 pow( )函数较合适；2x 在数学上是正确的，表示的是 2 * x，在写程序时 2x 是错误的表达式，必须用表达式 2 * x，标明每个数据之间的运算符。

在例 2-11 中，由于分母是 2 * x，如果输入的 x 值为 0，计算将产生错误。那么，程序该如何改进才能避免出现除以 0 这样的非法操作呢？请在第　章学习分支结构后，来解决这个问题。

习题 2

一、选择题

1. 以下合法的标识符是（　　）

A. int　　B. addNum　　C. 3rd　　D. _add. Num

2. 算数运算符中，要求运算量必须是整数的是（　　）

A. %　　B. /　　C. *　　D. +

3. scanf( ) 函数输入 double 类型数据时，要使用的格式字符串是 ( )

A. %ld　　B. %lf　　C. %f　　D. %s

4. 以下哪个式子的结果不是浮点型（　　）

A. 3/2　　B. 3. 0/2　　C. （float）3/2　　D. 3/2. 0

5. 语句“scanf（"x=%f，y=%f"，&x，&y);”，正确的输入是（　　）

A. 1. 5，2. 5　　B. 1. 5 2. 5

C. x=1. 5，y=2. 5　　D. x=1. 5 y=2. 5

二、程序填空题

1. 下列程序的功能是按照转换规则为：华氏度=32+摄氏度×1. 8，将输入的摄氏温度转换为华氏温度。请在横线处填入代码，将程序补充完整。

```
#include <stdio.h>
int main()
{
   ___(1)___ cTemp, fTemp;
   scanf("%f", ___(2)___);
   fTemp=___(3)___ ;
   printf("%f\n", fTemp);
   return 0;
}
```

(1)_____ ;

(2)_____ ;

(3)_____ 。

2. 假设三角形的三边分别为 a、b、c，周长的一半为 $s=(a+b+c)/2$，则三角形面积 $area=\sqrt{s(s-a)(s-b)(s-c)}$ 。下列程序的功能是根据输入的三边长，计算相应三角形的面积。请在横线处填入代码，将程序补充完整。

```
#include <stdio.h>
int main()
{
```

```
    float a, b, c, s, area;
    printf("请输入三边长:\n");
    scanf("%f %f %f", &a, &b, &c);
    s=(1)_____ ;
    area= (2)_____ ;
    printf("area=%f\n", area);
    return 0;
}
(1)_____ ;
(2)_____ ;
```

三、请写出下列程序的运行结果

```
#include <stdio.h>
int main()
{
    int a, b;
    float x;
    double y;
    a=123, b=12345;
    x=123.0456789;
    y=123456.0456789;
    printf("a=%4d, b=%4d\n", a, b);
    printf("a=%-4d, b=%-4d\n", a, b);
    printf("a=%-+4d, b=%4d\n", a, b);
    printf("x=%12f, y=%12f\n", x, y);
    printf("x=%5.2f, y=%5.2f\n", x, y);
    printf("%s\n", "How are you");
    printf("%7.2s\n", "How are you");
    printf("%-.5s\n", "How are you");
    return 0;
}
```

第 3 章

分 支 结 构

本章要点

1. 分支结构语法及语句书写规范；
2. 字符型数据如何表示？ASCII 编码原理；
3. 多分支结构；
4. 关系运算符，逻辑运算符的运算规则；
5. 嵌套分支结构。

3.1 二分支结构

3.1.1 计算男女的标准体重

【例 3-1】体重是反应和衡量一个人健康状况的重要标志之一，过胖和过瘦都不利于健康，身高体重不协调也不会给人以美感。请编写一个程序，根据用户输入的性别、身高，计算其标准体重。

女性标准体重=身高-105；男性标准体重=身高-100

解题思路

编写程序，不要急着写代码，而是要先分析程序中有哪些数据需要表达、程序功能流程要如何控制。本例具体分析步骤如下：

数据分析

题面涉及的数据主要有性别、身高、体重，则需要 3 个变量来保存这 3 个数据，分别为

```
char sex;     //表示性别,一般用字符 F、M 表示,故用字符型变量表示
int height;   //表示身高
int weight;   //表示体重
```

解题流程

先输入性别、身高，再根据性别计算体重。如果性别为女（这里用'F'表示），体重=身高-105。否则，体重=身高-100。流程图如图 3.1 所示，程序需要用到二分支结构来实现性别判断和相应体重计算。

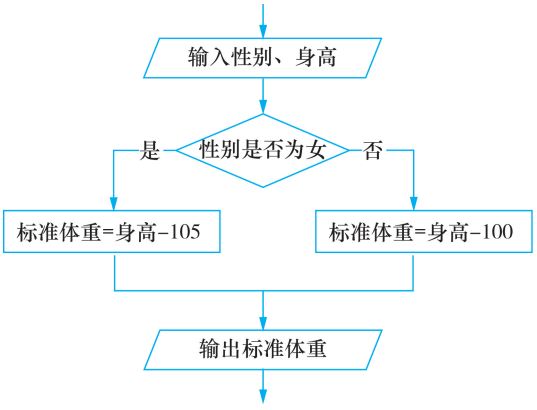

图 3.1　男女标准体重流程图

程序代码实现

```
#include <stdio.h>
int main()
{
  char sex;
  int height;
  int weight;
  printf("请输入您的性别、身高:");
  scanf("%c %d", &sex, &height);
  if (sex=='F')      //女性标准体重计算
  {
      weight=height-105;
  }
  else               //男性标准体重计算
  {
      weight=height-100;
```

```
    }
    printf("您的标准体重是:%d", weight);
    return 0;
}
```

【运行结果 1】

```
请输入您的性别、身高:F 160
您的标准体重是:55
```

【运行结果 2】

```
请输入您的性别、身高:M 175
您的标准体重是:75
```

程序用到的语法点：（1）二分支结构 if-else 语句实现男、女性别的判断。（2）性别使用字符型数据'F'、'M'表示，故需要用 char 类型声明字符型变量。字符型数据的输入、输出语法见 3. 1. 3 节介绍。

思考

如果输入的数据是：A 165，表示性别的字符是'A'，既不是'F'也不是'M'，程序的运行结果是怎样的？如何改进，使程序能有更好的健壮性，能够应对这样意外的输入情况呢？

3. 1. 2　二分支语句

1. 单分支结构

单分支结构语句的形式：

（1）第一种：

```
if (表达式) 语句 1;
```

（2）第二种：

```
if (表达式)
{
    语句 1;
    语句 2;
    …
    语句 n;
}
```

单分支结构语句执行流程为：先求解表达式的值，如果值为真（非 0 值），则执行语句 1；表达式的值为假则不执行语句 1，直接执行分支结构后的程序段。这和顺序结构不一样，在顺序结构中，各语句是否执行是没有条件的，且按照顺序依次执行各语句。单分支语句流程图如图 3.2 所示。

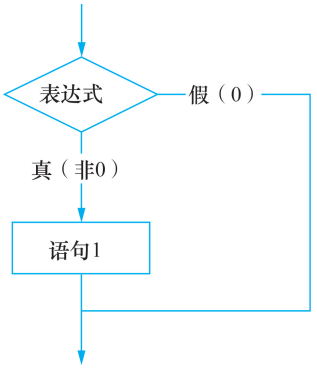

图 3.2　单分支语句流程

【注意】关于单分支的这两种语法，第一种语法语句 1 没有大括号，适用 if 条件只有一条语句。第二种情况适用 if 条件内有多条语句，这些语句组成一条复合语句，故必须用大括号括起来。如果没有大括号，那么与表达式有关的只有语句 1，语句 2 不属于 if 的条件限定。为了避免复合语句少了大括号的情况，不管是一条语句还是多条语句，都建议用第二种格式书写，即 if 表达式后都有一对大括号，然后将表达式为真时要执行的语句放在大括号内。语句要注意缩进，这也是一个好的编程习惯。

【例 3-2】输入一个学生成绩，如果小于 60 分，则输出 E。

程序代码实现

```
#include <stdio.h>
int main()
{
  int grade;
  printf("请输入成绩:");
  scanf("%d", &grade);
  if (grade < 60)
  {                    //注意一个大括号一行
      printf("E");  //注意 if 内语句要缩进,程序结构层次分明
  }
  return 0;
}
```

【运行结果 1】

```
请输入成绩:59
E
```

【运行结果 2】

```
请输入成绩:90
```

思考

为什么输入 90 时，程序没有相应的输出结果呢？

2. 二分支结构

二分支结构的语句形式如下：

```
if (表达式)
{
    语句 1;
}
else
{
    语句 2;
}
```

二分支结构的执行流程如图 3.3 所示：先求解表达式的值，如果值为真（非 0），则执行语句 1，否则执行语句 2。语句书写的时候，需要注意：

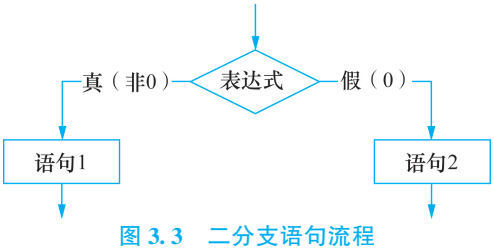

图 3.3　二分支语句流程

（1）if 条件语句一行，一个大括号一行，else 语句一行。

（2）else 后面不能加表达式，即不能加条件。else 就是对 if 的否定，含义就是 if 表达式为假。

（3）if 条件语句，务必用大括号括起来。尤其是 if 条件有多条语句时，如果没有大括号，else 将无法匹配 if 条件，程序编译会出现语法错误。因此，if、else 内的语句都要用大括号括起来，语句按层次缩进，这样的代码层次分明，可读性更好。

【例 3-3】输入一个数，判断其是否是 3 的倍数，是则输出“Yes”，否则输出“No”。

本题难点在是否为 3 的倍数，这个 C 语言的语句要如何书写？3 的倍数即能被 3 整除的整数，故可以选择算术运算符%去表示整除关系，故 if 内条件可以设计为 number%3==0。

程序代码实现

```
#include <stdio.h>
int main()
{
  int number;
  printf("请输入一个整数:");
  scanf("%d", &number);
  if (number %3==0) //判断整数是否为 3 的倍数
  {
    printf("Yes");
  }
  else
  {
    printf("No");
  }
  return 0;
}
```

【运行结果 1】

```
请输入一个整数:10
No
```

【运行结果 2】

```
请输入一个整数:27
Yes
```

【注意】判断两个数是否相等，要用两个等号==，不能用一个等号。一个等号是赋值符，计算机会将 number%3=0 理解成将 0 赋值给左边的变量，而这个表达式左边不是变量，语法会出错。

3.1.3 字符型数据

例 3.1 中，出现性别类型的数据，其值只能是‘F’‘M’。这是字符型数据，只能用字符型变量存储。那字符型数据在计算机中如何表示？

1. ASCII 编码

字符型数据计算机无法识别，所以需要对字符型数据进行二进制编码。目前所有的拉丁文字字母都采用美国信息交换标准代码即 ASCII 码。ASCII 码（见附录 F）使用指定的 7 位或 8 位二进制数组合来表示 128 或 256 种可能的字符。如'A'的 ASCII 码是 01000001，转成十进制数就是 65。ASCII 码有如下特点。

（1）ASCII 码是连续编码的，而且其对应的十进制是递增的。如'A'的 ASCII 码值是 65（十进制值），则不用查 ASCII 码表，即可知道字符'C'的 ASCII 码值是 67。同理，字符'a'～'z'，'0'～'9'也是递增编码。

（2）字符'1'和数值 1 是不同的。'1'在计算机内存中用其 ASCII 码（对应十进制是 49）表示，即字符'1'在内存中的二进制形式是 00110001。数值 1 在计算机内存是其数值对应的二进制数，如果采用 8 位二进制数表示，则数值 1 在内存中的形式是 00000001。计算机认为'1'和 1 是完全不同的两个数据。

（3）由于 ASCII 码最多 8 个二进制位（即 1 字节），因此字符型数据只占用 1 字节内存空间。

2. 字符常量

C 语言中字符常量是用单引号括起来的字符表示，如字符'a'，'（'，'+'都属于字符常量。

还有一类不能显示的字符，C 语言以转义字符表示。如换行符，其对应字符常量为'\n'。转义字符使用反斜杠+一个字符或者八进制数或者十六进制数表示。常见的转义字符见表 3.1 所列。

表 3.1　常用的转义字符

转义字符	含义	转义字符	含义
\n	换行符	\t	横向制表符
\\	一个反斜杠	\\\	两个反斜杠
\’	单引号	\ddd	1～3 位八进制数代表的字符
\’’	双引号	\xhh	1～2 位十六进制数代表的字符

注意：在 C 语言中表示单引号、双引号、反斜杠字符，也需要用转义字符。

3. 字符变量

字符型变量主要存储字符常量。字符变量声明语法如下：

```
char 变量名;
```

例如：

```
char ch;     //声明一个字符型变量 ch
ch='a';      //字符常量'a'赋值给变量 ch
```

```
ch='a'+1;   //字符型数据可以运算,ch 值更改为'b'
```

3.1.4 字符型数据的输入、输出

1. 字符型数据的输入函数

输入函数有 scanf( )、getchar( )。具体格式如下：

```
char ch;                //声明一个字符型变量 ch
scanf("%c", &ch);       //接收键盘输入的字符赋给 ch,类型格式字符为%c
```

或者

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

如果键盘输入多个字符，getchar( )函数只接收第一个字符。

2. 字符型数据的输出函数

输出函数有 printf( )、putchar( )。具体格式如下：

```
printf("%c", ch);//类型格式字符为%c
putchar(ch);       //ch 作为 putchar 函数的参数,输出一个字符(ch 的值)
```

3. 字符型数据、数值型数据的混合输入

假如要输入如下一组数据，输入语句要怎么写？

数据为：90 m 5.6

分析：该组数据有整型数据、字符型数据、浮点型数据，这样多种混合类型的数据输入建议用 scanf 函数，可采用如下输入语句：

```
scanf("%d %c %lf", &num1, &ch, &num2);
```

这里 num1，num2 存储 2 个数值型数据 90，5.6，ch 存储字符型数据。如果把上述语句改成这样写法：

```
scanf("%d%c%lf", &num1,&ch,&num2);
```

键盘上输入数据格式为：90 m 5.6。输入后，如果用语句"printf（"%c"，m);"输出 m 的值，会发现是空格。这是因为输入时 90 后有一个空格，而输入格式中%d%c 之间并没有空格，scanf 函数会以%d 的类型接收 90，以%c 的类型接收其后的空格，于是 m 的值是空格。输入数据该空格后是字符 m，并不符合 scanf 函数第三个数据%lf 的类型格式要求，于是 num2 值是乱码。因此，要严格按照 scanf 函数的格式控制字符串输入数据。

3.1.5 条件运算符

简单的二分支语句，也可以用条件表达式表示。条件表达式是通过条件运算符将 3 个表达式连接在一起。条件表达式一般格式如下：

```
表达式 1? 表达式 2:表达式 3
```

条件表达式计算过程是：先判断表达式 1 是否成立，成立则将表达式 2 的值作为整个条件表达式的值，不成立则将表达式 3 的值作为整个表达式的值。

如例 3-1 的二分支的语句如下：

```
if (sex=='F')   //女性标准体重计算
{
    weight=height-105;
}
else            //男性标准体重计算
{
    weight=height-100;
}
```

改成条件表达式，可以写为

```
weight=(sex=='F')? height-105: height-100;
```

上述表达式包含赋值运算符、算术运算符、条件运算符。由于优先级算术运算符>条件运算符>赋值符。故计算过程是：先计算括号内的关系表达式 sex=='F'，再计算条件运算符。若该关系表达式值为非 0，则 weight = height - 105；若该关系表达式值为 0，则 weight=height-100。

可见，灵活使用条件表达式，可使程序更简单明了，也能提高程序的运算效率。

3.2　多分支结构：if-else if 语句

3.2.1　检查你的 BMI 指数是否合格

【例 3-4】身体质量指数 BMI，是用体重的公斤数除以身高米数的平方得到的数值。成年人 BMI 的正常值在 18.5～23.9 之间，如果 BMI 低于 18.5，考虑体重过轻，BMI 达到 24～27 是体重过重，BMI 在 28～32 之间属于肥胖。如果 BMI 超过 32，就是非常肥胖的情况。请编写一个程序，根据用户输入的体重、身高，计算其 BMI 值，并给出其 BMI 情况。

数据分析

题面涉及的数据主要有：身高、体重和 BMI 值，需要 3 个变量来分别保存这些数据，数据类型可以是浮点型，变量声明如下：

```
double height;        //表示身高
double weight;        //表示体重
double BMI;           //表示 BMI
```

解题流程

本题出现 BMI 值的多个判断，可借助流程图辅助梳理流程。如图 3.4 所示的流程图有多个分支关系，需要借助多分支的 if…else 语句实现多个条件的判断。

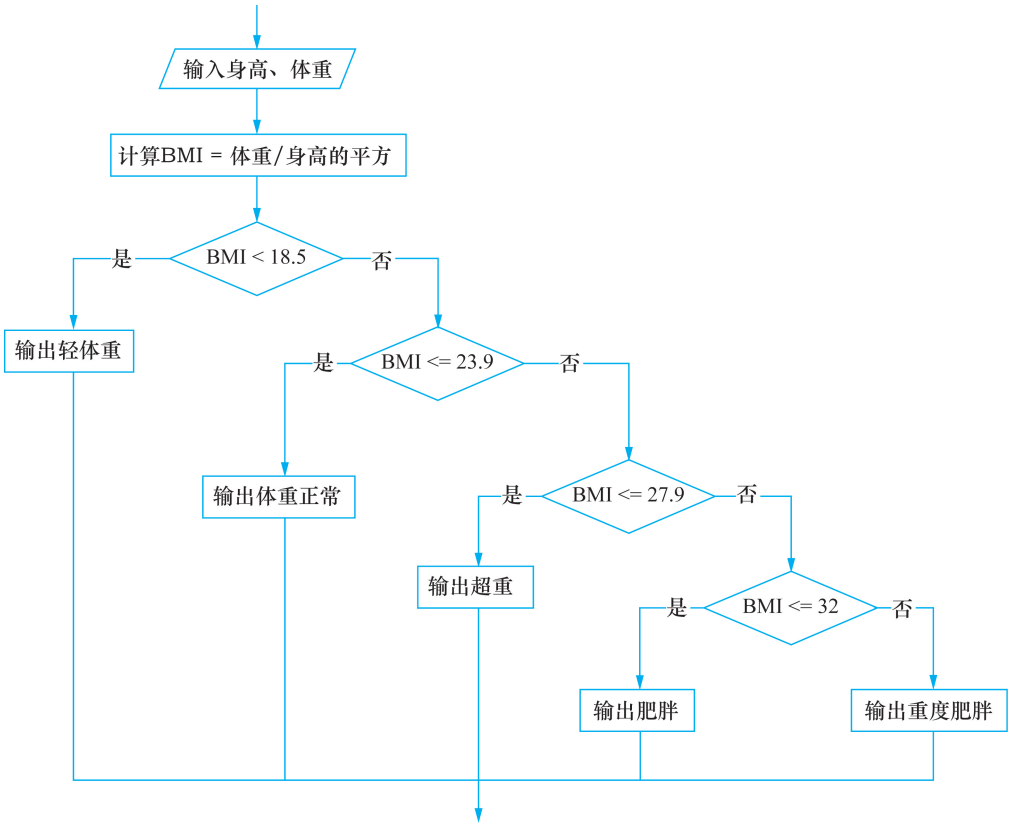

图 3.4　多分支语句流程

程序代码实现

```
#include <stdio.h>
int main()
{
  double height;                          //表示身高
  double weight;                          //表示体重
  double BMI;                             //表示 BMI
  printf("请输入您的身高(m)、体重(kg):");
  scanf("%lf,%lf", &height, &weight);
  BMI=weight/(height*height);                    //计算 BMI 值
  printf("您的 BMI 值为%.2f\n", BMI);
  if (BMI <18.5)
  {
    printf("轻体重");
```

```
    }
    else if (BMI <=23.9)
    {
      printf("体重正常");
    }
    else if (BMI <=27.9)
    {
      printf("超重");
    }
    else if (BMI <=32)
    {
      printf("肥胖");
    }
    else
    {
      printf("重度肥胖");
    }
    return 0;
}
```

【运行结果】

```
请输入您的身高(m)、体重(kg):1.59,60
您的 BMI 值为 23.73
体重正常
```

3.2.2　多分支语句

多分支结构：表示程序有多个判断流程。其语法主要通过 if-else if 语句实现，一般形式为：

```
if(表达式 1)
{
   语句块 1;
}
else if(表达式 2)
{
   语句块 2;
}
……
else if(表达式 n-1)
```

```
{
   语句块 n-1;
}
else
{
   语句块 n;
}
```

多分支语句的执行流程为：若表达式 1 为真，则执行语句块 1；若表达式 1 为假但表达式 2 为真，则执行语句块 2；…；若表达式 $n-2$ 为假但表达式 $n-1$ 为真，执行语句块 $n-1$；若表达式 1，表达式 2，…，表达式 $n-1$ 均为假，则执行语句块 n。最后一个 else 是对前面所有分支条件的否定。如例 3-4，最后一个 else 就是否定前面所有分支的 if 条件，也就是 BMI>32 的情况。

```
if (BMI <18.5)
{
    printf("轻体重");
}
if (BMI >= 18.5 && BMI < 23.9)  //逻辑表达式表示 BMI 在[18.5,23.9)区间的值
{
    printf("体重正常");
}
if (BMI >= 23.9 && BMI <= 27.9)
{
    printf("超重");
}
if (BMI >27.9 && BMI <= 32)
{
    printf("肥胖");
}
if(BMI > 32)
{
    printf("重度肥胖");
}
```

单分支语句的执行流程如图 3.5 所示。BMI 值的判断流程改成单分支写法，代码虽然好阅读，但是从程序的执行过程看，每个分支条件都需要判断，程序需要执行到最后一个分支才结束。因此，如果例 3-4 采用单分支结构，程序的执行效率低。例 3-4 的多分支结构程序（图 3.4 所示），只要满足某个分支条件，程序就输出 BMI 判断结果，不需要再进行其他分支条件的判断，程序的执行效率更高。

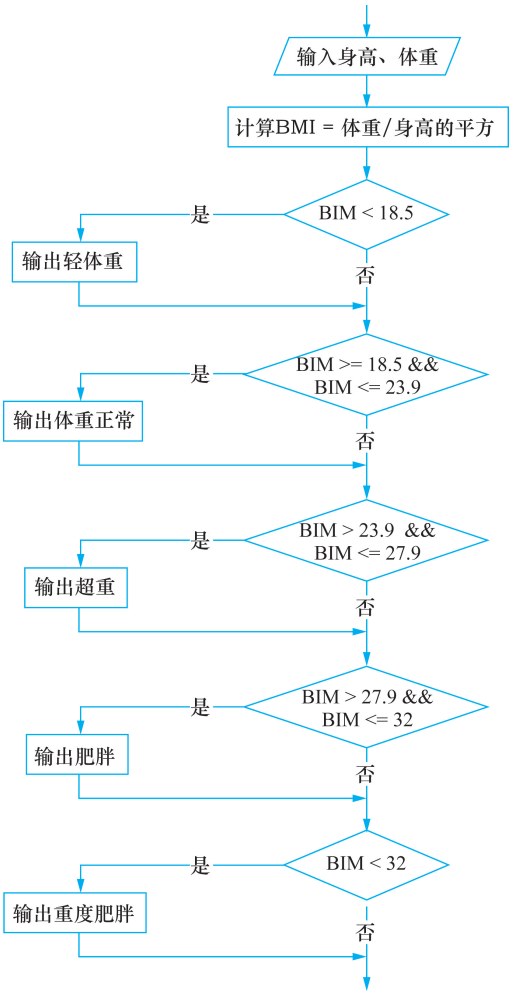

图 3.5　单分支语句流程

3.2.3　关系运算符

判断某个变量是否大于、等于或小于某个值，可以使用关系运算符。如 BMI>32，就是由关系运算符'>'连接两个运算量构成的关系运算表达式。C 语言中，关系运算就是比较运算，结果为“真”或“假”，即成立或不成立。

C 语言提供 6 种关系运算符，如表 3.2 所示。

表 3.2　关系运算符

运算符	名称	运算符	名称
<	小于	>	大于
<=	小于等于	>=	大于等于
==	是否等于	! =	是否不等于

关系表达式是指由关系运算符连接操作数的表达式。若关系成立，则表达式值为1（真）；若关系不成立，则表达式为0（假）。如5>3，值为1；5<3，值为0。

注意：判断2个数是否相等，写法为x==5，这里是2个等号。如果写成x=5，则C语言会理解为“把5赋值给x”。如果if语句中这样写：if（x=5），虽然没有语法错误，但是这样的写法并不是判断x是否与5相等，而是将5赋值给x，if条件等价为if（5），由于5是非0值，则if条件为真。因此，即使x原先的值为4，写成if（x=5），那么if条件就为真，并且x的值被修改为5。

常见关系表达式：

```
x %3==0          //判断 x 是否被 3 整除
x/10==0          //判断 x 是否是个位数
x!='\n'          //x 不是换行符
```

注意：算术运算符优先级高于关系运算符，故先算算术运算符，再算关系运算符。

3.2.4 逻辑运算符

在程序设计中，判断条件可能不止一个。

这些条件有些需要同时成立，有些只要满足其中一个即可。多个条件的连接需要用到逻辑运算符。逻辑运算符连接的表达式也称为逻辑表达式。逻辑表达式结果为“真”（值为1）或“假”（值为0），如：

```
x <= 10 && x >= 3     //表示 x 小于等于 10 同时 x 还要大于等于 3,即区间[3,10]的表示方法
```

在上述表达式中，&&是逻辑与运算符，用来连接两个关系表达式。表示两个关系要同时成立才为真，有一个关系不成立，则为假。逻辑运算符有3种，见表3.3所列。

表3.3 逻辑运算符

运算符	名称	运算规则
&&	逻辑与	a&&b：a和b同时为真，则结果为真；有一个为假，则结果为假
||	逻辑或	a||b：a和b有一个为真，结果为真；a和b都为假，结果为假
!	逻辑非	!a：如果a为真，则结果为假；如果a为假，则结果为真

【例3-5】判断字符变量是否是字母。

解题思路

字母都是用ASCII码表示，因此只要判断字符变量ch是否在'a'-'z'或者'A'-'Z'之间即可。其中：

'a'--'z'条件写成C语句为：ch >='a'&& ch <= 'z'　　（条件1）

'A'--'Z'条件写成C语句为：ch >= 'A'&& ch <= 'Z'　　（条件2）

变量 ch 只要满足上面其中一个条件，就是字母。故判断 ch 是否为字母的条件为：

```
(ch >= 'a'&& ch <= 'z') || ( ch >= 'A'&& ch <= 'Z')
```

程序代码实现

```
#include <stdio.h>
int main()
{
  char ch;
  ch = getchar();
  if ((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z'))
    printf("Yes");
  else
    printf("No");
  return 0;
}
```

【运行结果 1】

```
c
Yes
```

【运行结果 2】

```
3
No
```

逻辑运算符优先级小于关系运算符，但逻辑与和逻辑或的优先级一样，结合方向都是从左到右。因此，如果一个表达式里面有多个逻辑运算符，要适当使用圆括号进行表达式分隔。例如在例 3-5 的 if 条件中，判断变量 ch 是否是字符，用了两对圆括号将条件 1 和条件 2 分别括起来，然后再进行逻辑或运算。少了圆括号，计算结果将会不正确。

【注意】x 在区间［3，10］中，数学的写法是 3<=x<=10，但在 C 语言中，这样写，会被这样理解为是包含 2 个关系运算符'<='的表达式，<=的结合方向从左到右，表达式“3<=x<=10”先计算 3<=x 是否成立，成立为真，值为 1。不成立为假，值为 0。不管关系成立与否，3<=x 的值都会小于等于 10，因此无论 x 的值是什么，表达式“3<=x<=10”的值都是 1，无法判断 x 是否在［3，10］内。表示 x 在区间［3，10］中的 C 语句，正确写法应该是：

```
x <= 10 && x >= 3
```

思考

判断年份 year 是否为闰年的条件表达式该怎么写？闰年的条件是 year 能被 4 整除但不

能被 100 整除，或 year 能被 400 整除。

【解答】（year % 4==0 && year % 100!=0） ‖（year % 400==0）

3.3 多分支结构：switch 语句

3.3.1 制作党建学习菜单

【例 3-6】做一个党建学习的菜单。用户输入 1，选择“我要读文章”；输入 2，选择“视听学习”；输入 3，选择“每日答题”；输入 4，选择“双人对战”。输入 0，退出程序。输入其他选项，提示“输入错误”。

菜单界面显示如下：

```
* * * * * * * * * * * * * 欢迎进入党建学习系统 * * * * * * * * * * * *
*                          1：我要读文章                                *
*                          2：视听学习                                  *
*                          3：每日答题                                  *
*                          4：双人对战                                  *
*                          0：退出程序                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

数据分析

本题涉及的数据：用户输入的选项，均为整型数据，可以声明一个整型变量 option 来表示。输出的数据为常量，即根据用户输入的选项，输出相应的内容。

解题流程

本题为一个菜单界面，解题步骤如下：

（1）先展示一个菜单选项界面。菜单有 5 个内容，需要用顺序结构输出这 5 个内容。

（2）提示用户输入选项，然后根据选项，输出相应的内容。菜单可选项有 5 个，需用多分支结构实现。

程序代码实现

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int option;
  printf("************ 欢迎进入党建学习系统*********** \n");
  printf("*              1:我要读文章                    * \n");
  printf("*              2:视听学习                      * \n");
  printf("*              3:每日答题                      * \n");
```

```
    printf("*                4:双人对战                          * \n");
    printf("*                0:退出程序                          * \n");
    printf("*****************************************\n");

    printf("请输入您的选择:");
    scanf("%d", &option);

    switch (option)
    {
        case 1:printf("您的选择是:我要读文章\n"); break;
        case 2:printf("您的选择是:视听学习\n"); break;
        case 3:printf("您的选择是:每日答题\n"); break;
        case 4:printf("您的选择是:双人对战\n"); break;
        case 0:
            printf("您的选择是:退出游戏");
            exit(0);  //退出游戏,需要包含头文件 stdlib.h
        default:printf("输入有误,输入数字只能是 0, 1, 2, 3, 4!\n");
    }  return 0;
}
```

【运行结果 1】

```
请输入您的选择:3
您的选择是:每日答题
```

【运行结果 2】

```
请输入您的选择:6
输入有误,输入数字只能是 0, 1, 2, 3, 4!
```

思考

将程序中的 break 语句去掉，程序的执行结果是什么？为什么？

3.3.2　switch 语句

例 3-6 中使用了 switch 语句来实现多分支结构。switch 语句形式如下：

```
switch(变量或表达式)
{
    case 常量表达式 1: 语句段 1; break;
    case 常量表达式 2: 语句段 2; break;
    ……
```

```
        case 常量表达式 n: 语句段 n; break;
        default: 语句段 n+1;      //不在上述分支内的情况
    }
```

switch 语句的执行流程如图 3.6 所示：首先计算 switch 后圆括号内表达式的值，如果表达式的值与 case 后某个常量表达式的值相同，则执行该 case 后的语句段；如果表达式的值与任何一个常量表达式的值都不相同，则执行 default 后的语句段。

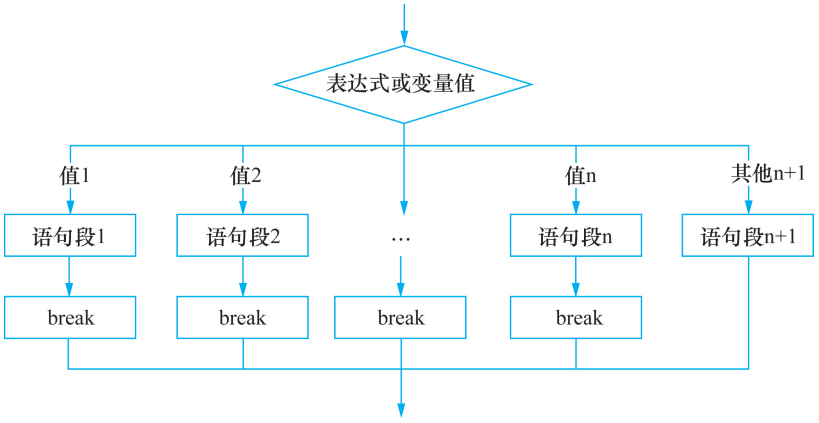

图 3.6　switch 语句流程图

这里要注意：

（1）case 后的常量表达式必须是整型或字符型常量，不能是浮点小数或表达式。如 case x>3 是错误的。

（2）case 和后面的常量表达式之间要空格隔开，并且所有常量表达式的值不能相等。

（3）每个语句段可以包含 1 条或多条语句，也可以是空语句。一般建议每个语句段后必须有 break 语句，这样程序才能终止 switch 语句的继续执行。如果语句段后没有 break 语句，当表达式的值与某个常量表达式的值相等时，即执行其后的语句，然后不再进行判断，继续执行后面所有 case 后的语句。如在例 3-6 程序代码中去掉 break 语句，输入 1 时的执行结果为：

```
请输入您的选择:1
您的选择是:我要读文章
您的选择是:视听学习
您的选择是:每日答题
您的选择是:双人对战
您的选择是:退出游戏
```

输入 1，case 1 语句段后没有 break 语句，程序不会终止 switch 语句的执行，而是会继续执行后续所有的 case 语句段，直到碰到 break 语句或者 switch 语句结束。

（4）default 是否定所有 case 的情况，也可以不写 default 语句。

思考

参考图 3.6 和图 3.4，思考 switch 语句和 if-else if 的多分支语句区别？

【解答】

共同点：二者都能表示多分支结构。区别主要有：

1. switch 表达式的值与某个 case 后的常量表达式的值相同，则执行该 case 语句，不再判断后续 case 的常量表达式值。if-else if 语句需要一直判断到满足条件的分支，判断的分支比 switch 语句多，故 switch 语法执行效率更高。

2. if-else if 适用所有多分支条件。如例 3-4 中，判断条件是关系表达式，switch 语句中，case 后常量表达式只能是整型或字符型常量，不能是变量或表达式，如果这样写：case x<32 是错误的。因此，例 3-4 不能使用 switch 来改写，if-else if 适用更多情况。

3. switch 语句判断的值必须是整数，if 判断整型数和浮点数都可以。

3.4　嵌套的分支结构

3.4.1　用户登录验证程序

【例 3-7】设计一个用户登录验证程序。假如某个系统目前只有一个用户，其登录账号为 123，密码为 369。现在要做一个用户登录验证程序，可以根据用户输入的指令，判断其输入的账号和密码是否正确。如果用户输入指令‘N’，表示用户为新用户，需要注册新账号，那么就要先判断系统是否存在该账号。若账号已经存在，则不能创建（假设系统目前只有一个账号 123），提示“用户已存在!”；若不存在，可新建账号，并提示“用户注册成功!”。如果用户输入指令‘L’，表示用户要登录系统。若账号、密码正确，则提示“登录成功!”；若用户名错误，提示“用户名错误!”；若密码错误，则提示“密码错误!”。下列两组输入输出的情况是程序运行的示例：

输入数据 1：L 123 345

输出数据 1：密码错误!

输入数据 2：N 126 345

输出数据 2：用户注册成功!

数据分析

本题涉及的数据有用户输入的指令，账号和密码，需要 3 个变量来表示。变量声明如下：

```
char option;        //指令
int  user;          //账号
int  password;      //密码
```

解题流程

本题条件判断分支较多，流程稍微有点复杂，可以借助流程图来梳理。按照题目要求，首先是根据用户输入指令‘N’，‘L’，来判断是登录还是注册，再进入登录和注册2个流程判断，分支流程结构如图3.7所示。如果用户输入登录指令‘L’，判断用户名和密码是否正确的分支又分为两个部分：判断用户名是否正确和判断密码是否正确。从图3.7可以看出，这几个判断条件是一种嵌套关系，需要用if-else的嵌套结构来实现。

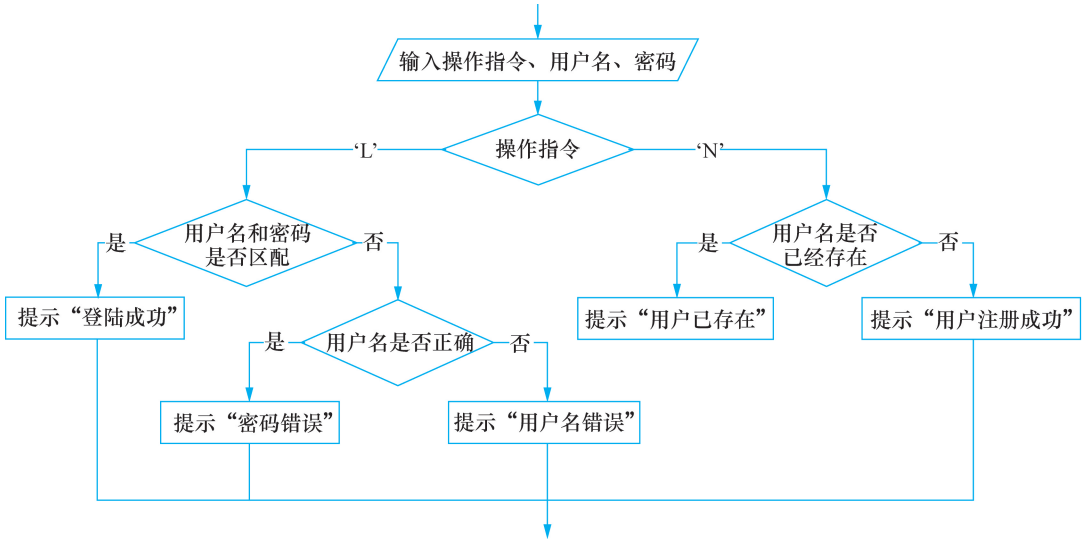

图3.7　登录验证流程图

程序代码实现

```
#include <stdio.h>
int main()
{
  char option;                          //指令
  int user;                             //账号
  int password;                         //密码
  printf("请输入操作指令及用户账号、密码(空格隔开):\n");
  scanf("%c %d %d", &option, &user, &password);

  if (option=='L')
  {
    if (user==123 && password==369)   //账号密码正确
    {
      printf("登录成功!\n");
```

```
        }
        else
        {
          if (user!=123)  //用户名不正确
          {
            printf("用户名错误!\n");
          }
          else            //密码错误
          {
            printf("密码错误!\n");
          }
        }
      }
      if (option=='N')
      {
        if (user==123)
        {
          printf("用户已存在!\n");
        }
        else
        {!\n");
          printf("用户注册成功
        }
      }
      return 0;
    }
```

从上面代码可以看到，每个条件分支内又有条件的判断，即 if 语句里面还有 if 语句，这就是嵌套的 if-else 语句。

3.4.2　嵌套的 if…else 语句

嵌套的 if-else 语句，因为多层嵌套，所以每个分支对应的语句务必用大括号括起来，大括号里面的内容要有缩进，语法形式为：

```
if(表达式)
{
    if(表达式)
        {
```

```
            语句段 1;
        }
    else
        {
            语句段 2;
        }
}
else
{
    if(表达式)
        {
            语句段 3;
        }
    else
        {
            语句段 4;
        }
}
```

初学者刚写嵌套 if-else 语句时，常常忘记书写大括号，如下面语句：

```
if (表达式 1)
if (表达式 2) 语句 1
else  语句 2     //满足表达式 1,否定表达式 2
else             //否定表达式 1
if (表达式 3) 语句 3
else  语句 4     //否定表达式 3
```

编译器解析上面代码时，由于没有大括号，else 会和最近的未配对的 if 配对。这种代码写法不仅阅读性很差，而且容易造成程序错误，故嵌套 if…else 语句不建议这样写。

思考

如下的 if-else 嵌套语句，请改写 else 语句，使得 else 否定第一个 if 语句的条件。

```
if(x<5)
    if(x%2==0) y=x/10;
    else y=x*2;
```

【解答】

```
if(x<5)
{
```

```
    if(x%2==0) y=x/10;
}
else y=x*2;
```

习题 3

一、选择题

1. 若已定义：char x；正确判断 x 中字符是大写英文字母的逻辑表达式为（　　）

A. X>=A && x<=Z　　B. "A"<=x && x<="Z"

C. 'A'<=x<'Z'　　D. 'A'<=x && x<='Z'

2. 若已定义：int t，a，b；语句 t=（a=5，b=a++）；执行后，变量 t、a、b 的值依次为（　　）

A. 6、6、5　　B. 6、6、6

C. 5、6、5　　D. 5、6、6

3. 以下程序段执行后变量 m 的值为（　　）

```
int x=6, y=4, z=8, m;
m=(x<y)?(x<z? x:z):(y<z? y:z)
```

A. 6　　B. 8　　C. 4　　D. 1

4. 以下程序段运行结果是（　　）

```
#include <stdio.h>
int main()
{ int x=1, y=2, z=3, a=4;
  if (!x)
    a--;
  else if (y)
    ;
  if (z=0)
    a=3;
  else
    a=-3;
  printf("%d\n", a);
  return 0;
}
```

A. 4　　B. 3　　C. 4　　D. -3

5. 以下程序段的运行结果是（　　）

```
int score=87;
char grade='A';
```

```
    switch (score/10)
   { case 7:
     grade ++;
     case 8:
     grade ++;
     case 9:
     grade ++;
   }
   printf("%c\n", grade);
```

A. A　　B. B　　C. C　　D. D

二、填空题

1. 写出以下程序段的运行结果。

```
    int  a;
    scanf("%d", &a);
    if(a > 50)  printf("%d", a);
    if(a > 40)  printf("%d", a);
    if(a > 30)  printf("%d", a);
```

输入 32,输出_____(1)_____

输入 46,输出_____(2)_____

输入 58,输出_____(3)_____

2. 以下程序段的运行结果是：_____

```
char c='b';
int k=4;
switch(c)
{
    case 'a': k=k+1; break;
    case 'b': k=k+2;
    case 'c': k=k+3;
}
    printf("%d", k);
```

三、编程题

1. 输入三个整数 x，y，z，请把这三个数由小到大输出。

输入样例：

2 1 3

输出样例：

```
x=1, y=2, z=3
```

2. 假设90号汽油6.95元/升、93号汽油7.44元/升、97号汽油7.93元/升。为吸引顾客，某自动加油站推出了“自助服务”和“协助服务”两个服务等级，分别可得到5%和3%的折扣。本题要求编写程序，根据输入顾客的加油量a，汽油品种b（90号、93号或97号）和服务类型c（m表示自助，e表示协助），计算并顾客输出应付款。

   输入格式：

   在一行中输入两个整数和一个字符，分别表示顾客的加油量a，汽油品种b（90、93或97）和服务类型c（m-自助，e-协助）。

   输出格式：

   在一行中输出应付款额，保留小数点后2位。

   输入样例：

   ```
   40 97 m
   ```

   输出样例：

   ```
   301.34
   ```

3. 本题要求编写程序计算某年某月某日是该年中的第几天。

   输入格式：在一行中按照格式“yyyy/mm/dd”（即“年/月/日”）输入日期。注意：闰年的判别条件是该年年份能被4整除但不能被100整除或者能被400整除；闰年的2月有29天。

   输出格式：

   在一行中输出日期是该年中的第几天。

   输入样例1：

   ```
   2009/03/02
   ```

   输出样例1：

   ```
   61
   ```

   输入样例2：

   ```
   2000/03/02
   ```

   输出样例2：

   ```
   62
   ```