第 3 章　选择结构程序设计

我们编写程序的目的，是为了通过计算机来解决现实生活中所遇到的种种问题。在现实生活中我们所遇到的问题，往往并不是仅仅按照事情出现的先后次序来依次执行的。对于某个生活中的问题，是否需要我们去解决？如果需要我们解决，那又需要我们如何去解决？这就要求我们先进行一些必要的判断。

例如，①如果室友不在，你要去开门。这需要判断你的室友是否在寝室。②雨天我们要带伞出门。这需要判断今天是否有雨。③饭前洗手。这需要判断是否即将吃饭。诸如此类问题很多，但解决此类问题有一个共同点：即先判断某个条件是否被满足，再去决定应对的策略。

选择结构是结构化程序设计的三种基本结构之一。其作用是根据不同的条件来执行不同的操作。在 C 语言中，判断是否满足某个条件是靠关系表达式和逻辑表达式来实现的。

3.1　关系运算符和关系表达式

在数学运算、字符比较等方面，经常出现一些比较大小的运算。例如，求解一元二次方程 $ax^2 + bx + c = 0$ 的时候，需要根据 $b^2 - 4ac$ 的值是否小于 0 来决定该方程的根是虚根还是实根。判断这种比较大小的运算在 C 语言中可以用关系运算符来实现。

3.1.1　关系运算符及其优先级别

在 C 语言中提供了 6 种关系运算符，如表 3.1 所示。在表中，前 4 种关系运算符（<、<=、>、>=）的优先级别相同，后两种（==、!=）的优先级别也相同，且前 4 种高于后两种。关系运算符的优先级别低于算术运算符，高于赋值运算符。

表 3.1　关系运算符

运算符	含义
<	小于
<=	小于或等于
>	大于
>=	大于或等于
==	等于
!=	不等于

3.1.2　关系表达式

用关系运算符将两个表达式连接起来的式子就称为关系表达式。如：

5 > 3，a < b + 2，a + b != c + d，c == 'x'

都是合法的关系表达式。

关系表达式也有一个值，但与算术表达式不一样，关系表达式所描述的条件只有两种情况，“成立”或者“不成立”。所以在 C 语言中用“真”或者“假”来描述这两种情况。例如，“5<6”的值是“真”而“4>=10”的值是“假”。

在 C 语言中没有逻辑型数据来描述“真”或“假”，而是采用“1”或“0”来反映关系表达式的结果。如果某个关系表达式的值为“真”，则这个关系表达式的值就是“1”，如果为“假”，则值为“0”。

小提示

在使用关系运算符来判断某一个值为实数的表达式是否等于 0 时，由于实数在计算和存储时会出现一些误差，因此不能直接使用“<表达式>== 0”来判断，因为这样可能会出现本来是 0 的值，但是由于存储误差而被判断成不等于 0。所以采取的办法是判断这个表达式的值的绝对值是否小于一个很小的数，如 10^{-6}，如果小于此数，就认为表达式的值等于 0。如“fabs(disc) <= 1e-6”。

3.2　逻辑运算符和逻辑表达式

实际上，我们在决定执行什么样的操作时往往需要几个条件同时成立，而有时候只需要很多条件中满足其中的一个就可以。像这种需要几个条件同时成立或者多个条件中只要有一个成立的情况，单单靠关系运算符还不够，还需要靠逻辑运算符来完成。

3.2.1　逻辑运算符及其优先级别

C 语言中提供了 3 种逻辑运算符，如表 3.2 所示。表中“&&”和“||”是双目运算符，要求有两个运算量，如(a < b)&& (c > d)、(a < b)||(c > d)。“!”是单目运算符，只需要一个运算量，如! (a < b)。

表 3.2　逻辑运算符

运算符	举例	运算规则
&&（逻辑与）	a && b	若 a、b 同时为真，a && b 为真，除此之外都为假
||（逻辑或）	a || b	若 a、b 之一为真，则 a || b 为真
!（逻辑非）	!a	若 a 为真。则!a 为假，反之，若 a 为假，则!a 为真

逻辑运算符的优先级别由高到低依次是：“!”、“&&”、“||”。其中“&&”和“||”低于关系运算符，而“!”高于算术运算符。

3.2.2　逻辑表达式

用逻辑运算符将关系表达式或逻辑量连接起来的式子就是逻辑表达式。跟关系表达式一样，逻辑表达式的值也是一个逻辑量“真”或“假”。在表示逻辑表达式的计算结果的时候，仍然以数值“1”代表逻辑真，以数值“0”代表逻辑假。但在判断某一个量是否为真的时候，则是以“0”值代表假，以“非 0”值代表真。如：

若 a = 5，b = 0，c = 2，则!a 的值为 0，因为 a 的值为 5，是一个非 0 值，被认作“真”。同理!b 的值为 1，a && c 的值为 1。

小提示：逻辑表达式的求解

在求解逻辑表达式的过程中，并非所有的逻辑运算符都被执行，只是在必须执行下一个逻辑运算符才能求出表达式的解时，才执行该运算符。如:

（1）a && b && c 只有在 a 为“真”的时候，才需要判别 b 的值，同时，也只有在 a 和 b 都为“真”的情况下才需要判别 c 的值。因为，只要 a 为“假”，就不必判别 b 和 c 就可以确定整个表达式的值为“假”。同理，若 a 为“真”，b 为“假”，也不必判别 c。

（2）a || b || c 只要 a 为“真”，就不必判别 b 和 c。只有 a 为“假”才需要判别 b，a 和 b 都为“假”，才需要判别 c。

例如 a = 1，b = 2，c = 3，d = 4，m = 1，n = 1，在执行表达式(m = a > b)&&(n = c > d)后，由于“a > b”的值为“0”，因此 m = 0，而“n = c > d”并不被执行，此时 n 仍保持原值“1”。

C 语言中的语句（一）

（1）表达式语句：表达式语句由一个表达式加上一个分号“;”构成。如 a = 3 是一个赋值表达式，而 a = 3; 是一个赋值语句。分号“;”是语句中不可缺少的组成部分，任何表达式加上“;”都成为相应的语句。

（2）函数调用语句：由一个函数调用加上一个“;”构成，如“printf("abc\n");”。

（3）空语句：仅有一个分号的语句，它什么都不做。

（4）复合语句：可以使用{}把一些语句括起来就成为复合语句，如:

```
{
    z = x + y; t = z / 100; printf("%f", t);
}
```

3.3 if 语句

在 C 语言程序中，解决问题时往往需要由对条件的判断结果，来决定程序的执行流程。这就要依靠选择结构来实现。选择结构也被称为分支结构。

多数情况下，我们通过 if 语句来实现选择结构。if 语句有如下几种格式。

3.3.1 简单的 if 语句

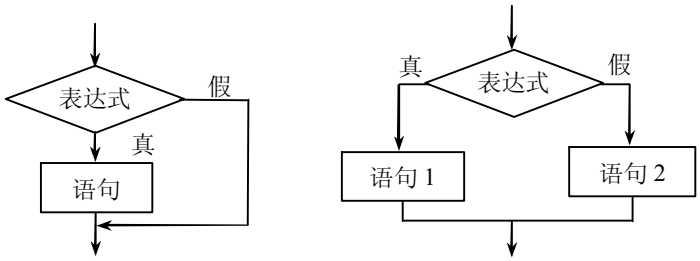
最为简单的 if 语句，是用来决定是否执行某个语句或者语句组。格式如下：

```
if(表达式)
{
    语句组;
}
```

注意：这里没有“;”

如果表达式的值为“真”（或者非零），就执行语句组，否则程序就会跳过该语句组，直接执行 if 语句之后的其他语句。其程序流程如图 3.1（a）所示。通常，表达式是一个关系表达式或者逻辑表达式，根据表达式值的真假来判断是否执行语句组。更一般的情况，表达式可以是任意的表达式，表达式的值为 0 就被视为假，非 0 就被视为真。

当语句组只有一个语句的时候，if 语句的一对花括号可以省略。



（a）　　（b）

图 3.1　if 语句结构流程图

例 3.1　在键盘上输入两个不同的整数，并输出较大的那个整数。

程序清单如下：（文件名 3_1.c）

```
#include <stdio.h>
int main(){
    int a,b;
    printf("请输入两个不同的整数：");
    scanf("%d,%d",&a,&b);
    if (a > b) {                          //如果 a 的值大于 b 的值
        printf("较大的数是：%d\n",a);     //输出 a 的值
        return 0;                         //结束程序
    }
    printf("较大的数是：%d\n",b);         //当 a 大于 b 的情况不成立，输出 b 的值
    return 0;
}
```

程序执行结果如图 3.2 所示。

图 3.2　例 3.1 程序运行结果

3.3.2　在 if 语句中添加 else 子句

简单形式的 if 语句可以让我们根据条件的成立与否来判断是执行还是忽略某种操作。更多的时候，我们需要根据条件的成立与否在两个操作中选择一个来执行，如果只是使用 if 语句，程序就会显得笨拙，我们可以使用 if else 的形式来解决这个问题。

if else 语句的一般格式如下：

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

注意：这里没有“;”

注意：这里也没有“;”

if else 语句的执行过程如图 3.1（b）所示，表达式的值为“真”，则执行语句组 1，否则，执行语句组 2。同样，当语句组 1 只有一个语句时，if 后面的一对花括号可以省略，语句组 2 只有一个语句时，else 后面的一对花括号也可以省略。

例 3.2 在键盘上输入三个实数，表示一个三角形三条边的边长，判断这三条边能否构成一个三角形。

解题思路：三角形要求任意两边的边长之和大于第三边的边长。转化在 C 语言中，即输入的任意两个实数之和大于第三个实数。

程序清单如下：（文件名 3_2.c）

```
#include <stdio.h>
int main(){
    float a,b,c;
    printf("请输入三角形三边的边长：");
    scanf("%f,%f,%f",&a,&b,&c);
    if (a+b>c && a+c>b && b+c>a){        //判断任意两个数是否大于第三个数
        printf("此三条边能构成一个三角形。\n");
    }
    else{
        printf("此三条边不能构成一个三角形。\n");
    }
}
```

程序运行结果如图 3.3 所示。

图 3.3 例 3.2 程序运行结果

例 3.3 在键盘上输入一个正整数，判断该数是否能同时被 3 和 7 整除。

解题思路：该数若能同时被 3 和 7 整除，则说明该数除以 3 余数为 0，该数除以 7 余数也为 0，这两个条件之间是“与运算”关系。

程序清单如下：（文件名 3_3.c）

```
#include <stdio.h>
int main(){
    int n;
    printf("请输入一个正整数：");
    scanf("%d",&n);
    if (n % 3 ==0 && n % 7 == 0){
        printf("该数可以被 3 和 7 整除。\n");
    }
    else{
        printf("该数不可以被 3 和 7 整除。\n");
    }
    return 0;
}
```

程序运行结果如图 3.4 所示。

图 3.4　例 3.3 程序运行结果

在 C 语言中有一个条件运算符“？：”，其作用相当于一条 if 语句。如下例：

```
max=(a>b)?a:b;
```

其中“max=(a>b)?a:b”是一个“条件表达式”。它的执行过程是：如果（a>b）为真，则条件表达式取值 a；否则取值 b。无论表达式“a > b”成立与否，都执行一个赋值语句，并且向同一个变量 max 赋值。

条件运算符要求有 3 个操作对象，称三目运算符，它是 C 语言中唯一的一个三目运算符。其一般形式为：

表达式 1？表达式 2：表达式 3

其执行顺序为：先求解表达式 1，若为真（非 0），则求解表达式 2，并把表达式 2 的值作为整个条件表达式的值；若表达式 1 为假（0），则求解表达式 3，并把表达式 3 的值作为整个条件表达式的值。

对于条件运算符的使用，需要注意以下几点：

（1）条件运算符的优先级别高于赋值运算符，但是比关系运算符和算术运算符要低。如 max=(a>b)?a:b 这个表达式是先求解条件表达式，再将条件表达式的值赋值给变量 max。其中（a>b）的括号也可以取消。

（2）条件运算符的结合方向为“自右向左”，如有：

```
a>b? a:c>d?c:d
```

等价于：a>b? a:(c>d?c:d)。为了程序的可读性好，建议加上括号。

（3）条件表达式中的表达式 2 和表达式 3 可以是任意的表达式。表达式 1 与表达式 2 和表达式 3 的类型也可以不同。

3.3.3　多重选择 else if

在解决实际问题的时候，我们经常会遇到多重选择的情况，这个时候就可以使用 else if 来扩展 if else 结构以适应这种情况。其一般格式如下：

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

```
        语句组 3;
    }
    …
    else if(表达式 n-1)
    {
        语句组 n-1;
    }
    else
    {
        语句组 n;
    }
```

多重选择 else if 的程序流程如图 3.5 所示，当表达式 1 为真的时候则执行语句组 1，表达式 1 后面若干种情况不再判断。否则的话，将判断表达式 2 的真假，表达式 2 为真，则执行语句组 2，同理也不再判断表达式 2 后面的各种情况。否则将继续判断表达式 3，……，若表达式 n-1 为真，则执行语句组 n-1，否则执行语句组 n。同样，如果某一个语句组只有一个语句，相应的一对花括号也可以省略。

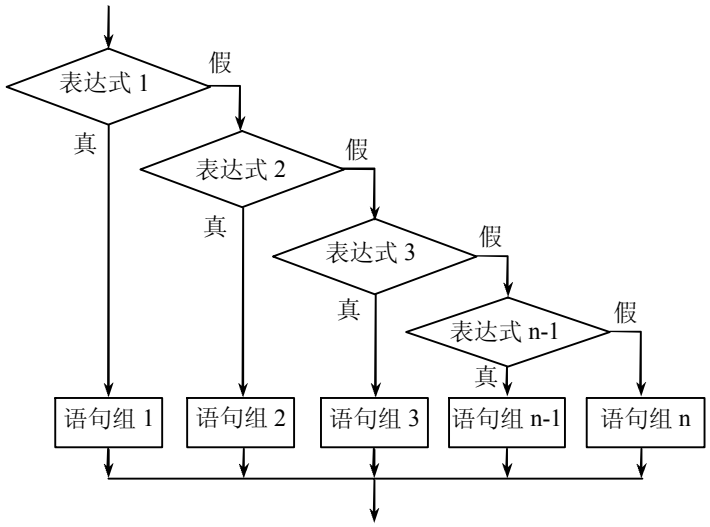


图 3.5　多重选择 else if 结构流程图

例 3.4　学生成绩测评等级规则如下：90 分以上为“优秀”，80～89 分为“良好”，70～79 分为“中等”，60～69 分为“及格”，低于 60 分为“不及格”。编写程序：通过输入学生的分数，评价学生的成绩等级。

程序清单如下：（文件名 3_4.c）

```
#include <stdio.h>
int main(){
    int s;
    printf("请输入学生的成绩(0-100):");
    scanf("%d",&s);
    if (s >= 90 && s <= 100){
        printf("优秀!\n");
    }
    else if (s >= 80){
```

```
            printf("良好!\n");
        }
        else if (s >= 70){
            printf("中等!\n");
        }
        else if (s >= 60){
            printf("及格!\n");
        }
        else{
            printf("不及格!\n");
        }
        return 0;
    }
```

程序运行结果如图 3.6 所示。

图 3.6　例 3.4 程序运行结果

3.3.4　if 语句的嵌套

在 if 语句的语句组中又包含一个或多个 if 语句称为 if 语句的嵌套。其一般形式如下：

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

内嵌 if 语句

同理，在每一个语句组里还可以继续嵌套 if 语句

内嵌 if 语句

当有众多的 if 和 else 的时候，由于某些语句组只有一个语句省略了花括号，计算机是怎样判断哪个 if 对应哪个 else 的？例如：

```
if(score >= 60)
if(score <= 100)
    printf("PASS!\n");
else
    printf("FAILURE!\n");
```

什么时候输出“FAILURE!”？是在 score 小于 60 的时候，还是在 score 大于 100 的时候？即 else 是对应第一个 if 语句还是第二个？在 C 语言中，如果没有花括号指明，else 将和它最接近的一个 if 相匹配。即在 score 大于 100 的时候输出“FAILURE!”。程序的缩进使得看起来 else 好像是与第一个 if 匹配的，但是，编译器会忽略缩进。事实上，我们本来的意愿也是需要与第一个 if 语句匹配，这个时候，就需要加上花括号。如下所示：

```
if（score >= 60）
{
    if（score <= 100）
        printf（"PASS!\n"）;
}
else
    printf（"FAILURE!\n"）;
```

加上一对花括号

例 3.5 将例 3.4 用 if 语句的嵌套形式编写。

程序清单如下：（文件名 3_5.c）

```
#include <stdio.h>
int main(){
    int s;
    printf("请输入学生的成绩（0-100）：");
    scanf("%d",&s);
    if (s >= 60 && s <= 100){
        if (s >= 90){
            printf("优秀!\n");
        }
        else{
            if (s >= 80){
                printf("良好!\n");
            }
            else{
                if (s >= 70){
                    printf("中等!\n");
                }
                else{
                    printf("及格!\n");
                }
            }
        }
    }
    else{
        printf("不及格!\n");
```

```
    }
    return 0;
}
```

程序运行结果如图 3.7 所示。

图 3.7　例 3.5 程序运行结果

将例 3.4 的程序与例 3.5 的程序相对比，我们可以看出：如果对于多种情况进行讨论，使用嵌套的 if 形式，嵌套层数过多，程序过长；采用 else if 结构可以使程序表达的思路更加清晰，可读性更强。

小提示：if 语句嵌套中的花括号

从技术的角度讲，if 和 if else 语句可以作为一个整体看成单个语句，所以在嵌套的时候可以不用加上花括号，然而，当语句很长的时候，花括号使人更容易读懂程序，更不容易出错。所以建议初学者将花括号都添加上去，不管语句组是一个语句还是多个。

3.4　switch 语句

在 C 语言中还可以使用 switch 语句来直接处理多分支选择的情况。它的一般形式如下：

```
switch（表达式）
{
    case  常量表达式 1：语句组 1；
    case  常量表达式 2：语句组 2；
    …
    case  常量表达式 n：语句组 n；
    default：              语句组 n+1；
}
```

例如：

```
switch(c)
{
    case 0: d = 0; break;
    case 1: d = 0.02; break;
    case 2:
    case 3: d = 0.05; break;
    case 4:
    case 5:
    case 6:
    case 7: d = 0.08; break;
    case 9:
    case 10:
```

```
        case 11: d = 0.1; break;
        case 12: d = 0.15;break;
    }
```

对 switch 语句做以下几点说明：

（1）switch 后面括号内的表达式可以是任意类型的表达式，当表达式的值与某一个 case 后面的常量表达式的值相等时，就执行此 case 后面的语句，如果所有 case 后面的常量表达式的值都没有与表达式匹配的，则执行 default 后面的语句。default 可以省略。

（2）每一个 case 的常量表达式的值必须互不相同，否则就会出现互相矛盾的现象。而且其值必须是一个常量（在程序运行过程中，值不会发生改变的量）。各个 case 和 default 的出现次序不影响执行结果。

（3）执行完一个 case 后面的语句后，程序流程转移到下一个 case 继续执行，直到 switch 语句执行结束或者遇见 break 语句的时候才结束 switch 语句的执行。“case 常量表达式”只是起语句标号的作用，并不是在该处进行条件判断。在执行 switch 语句时，根据 switch 后面表达式的值找到匹配的入口标号，就从此标号开始执行，不再进行判断。因此应该在需要跳出 switch 结构的 case 分支处使用 break 语句来终止 switch 语句的执行。

（4）多个 case 可以共用一组执行语句。如上例的

```
case 4:
case 5:
case 6:
case 7: d = 0.08; break;
```

当 c 的值为 4、5、6 或 7 时，都执行“d=0.08;break;”这组语句。

例 3.6 有一个函数：

$$y=\begin{cases} x+1 & (1<x<10) \\ x & (10<x<20) \\ x+2 & (20<x<30) \end{cases}$$

编写一程序，从键盘上输入 x 的值，根据上面的函数，求出 y 的值，并输出。

程序清单如下：（文件名 3_6.c）

```
#include <stdio.h>
int main()
{
    float x,y;
    int c;
    printf("请输入 x 值：");
    scanf("%f",&x);
    c = (int) x / 10;
    switch (c){
        case 0:     y = x + 1; break;
        case 1:     y = x; break;
        case 2:     y = x + 2; break;
    }
    printf("y 的值为：%f\n",y);
    return 0;
}
```

程序运行结果如图 3.8 所示。

图 3.8　例 3.6 程序运行结果

例 3.7　为运输公司对客户计算运费。路程 s（单位：km）越远，每吨每千米运费越低。标准如下：

s<250	没有折扣
250≤s<500	2%的折扣
500≤s<1000	5%的折扣
1000≤s<2000	8%的折扣
3000≤s<3000	10%的折扣
3000≤s	15%的折扣

设每吨每千米货物的基本运费为 p，货物重为 w，距离为 s，折扣为 d，则总运费 f 的计算公式为：$f = p \times w \times s \times (1 - d)$ 据此写出的程序如下所示。

程序清单如下：（文件名 3_7.c）

```
#include <stdio.h>
int main()
{
    int s;
    float p, w, d, f;
    scanf("%f%f%d", &p, &w, &s);
    if(s < 250)
    {
        d = 0;
    }
    else if(s >= 250 && s < 500)
    {
        d = 0.02;
    }
    else if(s >= 500 && s < 1000)
    {
        d = 0.05;
    }
    else if(s >= 100 && s < 2000)
    {
        d = 0.08;
    }
    else if(s >= 2000 && s < 3000)
    {
        d = 0.1;
    }
    else if(s >= 3000)
    {
        d = 0.15;
```

```
    }
    f = p * w * s * (1 - d);
    printf("freight = %15.4f\n", f);
    return 0;
}
```

例 3.7 的运行情况如图 3.9 所示。

图 3.9　例 3.7 程序运行结果

小提示：几个条件同时成立的表示方法

在前面已经提到过，要表示几个条件同时成立，需要用到逻辑与“&&”运算符。但是很多初学者受数学上的表示方法的影响，往往直接使用“250 <= s < 500”这样的表达式来表示 s 大于或等于 250，同时 s 小于 500 这样的条件。恰好，这样书写，是一个合法的关系表达式，C 编译器不会报告错误信息。导致很多人误以为这个表达式能正确描述 s 在[250，500）这个区间内。但实际上，根据关系运算符的结合性，表达式“250 <= s < 500”先计算“250 <= s”，其值是一个逻辑值，要么是值“1”，要么是值“0”，而无论是 1 还是 0，小于 500 肯定是成立的，所以表达式“250 <= s < 500”无论 s 取何值，该表达式的值都是“1”。无法正确表示 s 在区间[250，500）内。

我们再来分析例 3.7，发现折扣的变化是有规律的：折扣的变化点都是 250 的倍数。利用这一点，可以定义变量 c，c 的值为 s/250。c 代表 250 的倍数。当 c < 1 时，表示 s < 250，无折扣；1≤c < 2 时，表示 250≤s < 500，折扣 d = 2%；2≤c < 4 时，d = 5%；4≤c < 8 时，d = 8%；8≤c < 12 时，d = 10%；c≥12 时，d = 15%。据此，可使用 switch 语句来实现侧 3.7。

程序清单如下：（文件名 3_8.c）

```
#include <stdio.h>
int main()
{
    int c, s;
    float p, w, d, f;
    scanf("%f%f%d", &p, &w, &s);
    if(s >= 3000)
    {
        c = 12;
    }
    else
    {
        c = s / 250;
    }
    switch(c)
    {
        case 0: d = 0; break;
```

```
            case 1: d = 0.02; break;
            case 2:
            case 3: d = 0.05; break;
            case 4:
            case 5:
            case 6:
            case 7: d = 0.08; break;
            case 8:
            case 9:
            case 10:
            case 11: d = 0.1; break;
            case 12: d = 0.15; break;
        }
        f = p * w * s * (1 - d);
        printf("freight = %15.4f\n", f);
        return 0;
    }
```

注意：c、s 是整型变量，因此 c = s/250 为整数。当 s≥3000 时，令 c = 12，而不使 c 随 s 增大，这是为了在 switch 语句中便于处理，用一个 case 就可以处理所有 s≥3000 的情况。

3.5　小结

本章主要介绍了以下几方面内容：

1．关系运算符、逻辑运算符及其相应表达式；

2．实现选择结构的两种语句：if…else 语句和 switch…case 语句。

3.6　习题

1．输入一个字符，判断其是否为小写字母，如果是小写字母就将其转换为大写字母输出；如果不是小写字母，则原样输出。

2．设函数如下，编写程序从键盘输入不同的 x 值，求 y 值。

$$y=\begin{cases}|x^3| & (x<-1)\\ \dfrac{x}{2} & (-1\leqslant x<0)\\ \sqrt{x+4} & (x\geqslant 0)\end{cases}$$

3．某商场进行打折促销活动，按照购买商品所付的货款数的不同有相应的折扣率，共有 4 个级别，如下表所示。编写一个程序，要求输入购买商品的钱款数，输出相应的折扣率。

货款数额范围（元）	0～499.99	500～999.99	1000～1999.99	大于 2000
折扣率	9.9 折	9.6 折	9.3 折	8.8 折

4．输入某年某月某日，判断这一天是这一年的第几天。

5．输入一个字符，判断它是大写字母、小写字母、数字还是其他字符。