拳的 *computer* = rand() %3; 部分变成 *computer* = rand() %10;，*computer* 变量就可以得到 0～9 的随机数了。然后，再将 if… else if… else 这一部分改造为，变量 *computer* 的值是 0～4 时显示“石头”、为 5～7 时显示“剪刀”、为 8～9 时显示“布”。通过这些变化，石头剪刀布出现的几率就分别成 50%、30%、20% 了（代码清单 12-2）。

代码清单 12-2　具有习惯的猜拳游戏程序示例

```
#include <stdio.h>
#include <stdlib.h>

void main() {
    // 用来保存计算机出拳信息的变量
    int computer;

    // 等待用户键盘输入
    printf("石头剪刀……");
    getchar();
    printf("布！ \n");

    // 计算机决定出拳
    srand(time(NULL));
    computer = rand() % 10;

    // 输出计算机的出拳信息
    if(computer >= 0 && computer <= 4) {
       printf ("计算机的出拳是：石头 \n");
    } else if (computer >= 5 && computer <= 7) {
       printf ("计算机的出拳是：剪刀 \n");
    } else {
       printf ("计算机的出拳是：布 \n");
    }
}
```

这样，具有某种习惯的猜拳游戏就完成了。让我们把程序运行一下看看（表 12－2）。相比前面的程序，该程序的出拳方式更类似于人类的习惯。多次猜拳后，就会发现“这个计算机有出石头的习惯”。不过，真正的计算机并不具有习惯。这里只是运行了具有的习惯的程序而已。

表 12-2 代码清单 12-2 的运行结果和计算机的出拳信息

次数	1	2	3	4	5	6	7	8	9	10
出拳信息	石头	剪刀	剪刀	石头	石头	石头	剪刀	石头	布	石头

12.4 程序生成随机数的方法

接下来，让我们看一下随机数在程序中扮演的角色。在编写游戏程序时，以及在计算机模拟[①]等情况下，经常使用随机数。随机数也是用程序来表示人类的直觉及念头的一种方法。从代码清单 12-2 的运行结果中大家可以发现，“一直在出石头的时候突然出了一个剪刀”，这确实很像人类的行为方式。

随机数色子[②]是用来产生随机数的一种工具，每个色子有 20 面。晃动随机数色子后，出现在正面的数字就是随机数。由于计算机没法晃动随机数色子，因此程序一般会通过生成类似于随机数的数值公式来得到随机数。在 C 语言中，虽然该公式的实体是隐藏的，但只要调用 rand() 函数，就可以得到结果（随机数）。不过，由于借助公式产生的随机数具有一定的规律性，因此并不是真正的随机数，通常称为**伪随机数**。不过，虽然是伪随机数，仍然十分有用。

作为参考，这里向大家介绍一个获取伪随机数的公式。该公式称为**线性同余法**[③]。如果把 *Ri* 作为当前随机数的话，那么下一个出现的随

① 计算机模拟指的是利用计算机模拟实际试验的方式。经常被用于建筑物的耐震实验等实际难以进行的实验中。使用随机数的计算机模拟有时也称为“蒙特卡洛法”，来源于因赌博而闻名的城市——蒙特卡洛。

② 随机数色子的各面上都标有 1～20（或 1～10 每两个面为同一个数值）的数值。晃动随机数色子后，就可以得到 1～20（或 1～10）的一个随机数。

③ 除了线性同余法以外，还有其他获取伪随机数的方法。如可以获得更接近“真实随机数”的“乘同余法”、“M 系法”以及能够快速生成随机数的“Knuth 减算法”等。

机数 R_{i+1} 就可以用下面的公式来获取。

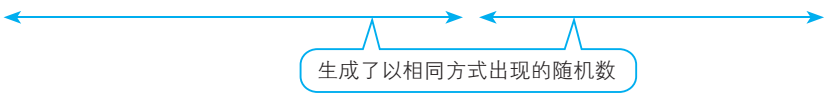
$$R_{i+1} = (a \times Ri + b) \bmod c$$

公式中出现的 mod，是整除后取余的意思。同 C 语言的 % 运算符的功能是一样的。对 *a*、*b*、*c* 各参数设定合适的整数后，可以从该公式获得的随机数的范围就是 0 到 *c*（不包含）。因为是用 *c* 来进行取余，所以得到这个范围也是理所当然的。我们不妨做一下尝试，把 *a* 设定为 5，*b* 设定为 3，*c* 设定为 8，获得的随机数就如表 12-3 所示。这里把 *Ri* 的初始值定为了 1。可以看出，这些随机数确实很像是无规则随机出现的数值。不过，产生 8 次随机数后，下 8 次产生的随机数就和前面的数值相同了。这种周期性是伪随机数的特征，也是为什么不是真随机数的原因。

C 语言的 rund() 函数中，也肯定通过某些公式生成了伪随机数。假如使用的是线性同余法的话，就需要提前设定 *Ri*、*a*、*b*、*c* 的数值，为此就要用到代码清单 12-1 及代码清单 12-2 中的 srand(time(NULL));。srand() 函数中的参数 time(NULL)，是用来获取当前时间的参数。以 time(NULL) 的值为基础，来设定 *Ri*、*a*、*b*、*c* 的数值。由于每次启动程序时的当前时间都是变化的，因此 *Ri*、*a*、*b*、*c* 的数值也会随之发生变化。*Ri*、*a*、*b*、*c* 的数值就称为**随机数的种子**，这一点大家要有个印象。而假如在不运行 srand(time(NULL)); 的情况下重复调用 rand() 函数的话，会出现什么情况呢？因为 *Ri*、*a*、*b*、*c* 的数值都有默认值，因此每次都会生成以相同方式出现的随机数。这样一来，游戏以及计算机模拟就都无法成立了。当然也就无法表示人类的思考了。

表 12-3 用线性同余法获得的随机数具有周期性

次数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
随机数	0	3	2	5	4	7	6	1	0	3	2	5	4	7	6	1



12.5 活用记忆功能以达到更接近人类的判断

人类的日常判断通常是根据直觉和经验做出的。直觉并不仅仅是简单的任意思考，通常还带有一些个人的思维习惯。在前面的介绍中我们已经提到，通过借助随机数，思考习惯等也是可以表示的。而如果在此基础上再加上经验（记忆）元素的话，想必就可以作成更接近人类思考的程序了。

请大家考虑一下猜拳游戏中是如何用到经验的。经过多次猜拳后，我们可能就会得到类似于“小 B 同学在出石头后出剪刀的概率比较高”这样的经验。基于这一经验，我们就可以应用以下策略，即“刚才小 B 同学出了一个石头，接下来应该会出剪刀，因此计算机出石头的话就赢了”。代码清单 12-3 是实现该策略的程序示例。在该程序中，通过键盘输入 0、1、2 来决定出拳。当键盘输入 0、1、2 以外的数值时，结束游戏。

代码清单 12-3 利用经验来决定出拳的猜拳游戏程序示例

```
#include <stdio.h>
#include <stdlib.h>

void main() {
    // 对手的出拳
    int human;

    // 假设对手刚才出了石头
    int prev = 0;
```

```
// 记忆对手出拳信息的 2 维数组
int memory[3][3] = { { 0, 0, 0 }, { 0, 0, 0 }, { 0, 0, 0 } };

// 预测的对手出拳信息
int max;

// 猜拳的回合数
int counter = 0;

// 计算机的出拳
int computer;

// 设定随机数的种子
srand(time(NULL));

// 重复猜拳
While (-1) {
    // 对手决定出拳信息
    printf("石头剪刀 （0= 石头，1= 剪刀，2= 布，其他 = 退出游戏）…");
    scanf("%d", &human);
    printf("布 \n");

    // 输入 0、1、2 以外的数值时游戏结束
    if (human < 0 || human > 2) break;

    // 记录猜拳的回合数
    counter++;

    // 计算机决定出拳信息
    if (counter < 10 ) {
        // 低于 10 次时，随机出拳
        computer = rand() % 3;
    } else {
        // 高于 10 次时，根据记忆来出拳
        max = 0;
        if (memory[prev][max] < memory[prev][1]) max = 1;
        if (memory[prev][max] < memory[prev][2]) max = 2;
        computer = (max + 2) % 3;
    }

    // 输出计算机的出拳信息
    if (computer == 0) {
        printf ("计算机的出拳是：石头 \n");
    } else if (computer == 1) {
        printf ("计算机的出拳是：剪刀 \n");
    } else {
```

```
                printf ("计算机的出拳是：布\n");
            }
            printf("\n");

            // 记录对手的出拳信息
            memory[prev][human]++;
            prev = human;
        }
    }
```

在该程序中，猜拳结果被保存在了计算机内部的内存中。而对手的出拳信息也通过 2 维数组[①]记录了下来。例如 player[0][0] 这个数组元素记录的就是对手出石头后再出石头的次数。数组的索引 0、1、2 分别表示石头、剪刀、布。由于猜拳游戏刚开始时，数据记录还不够充足，因此这里使用了变量 *counter* 来记录猜拳的次数，当不满 10 次时，由随机数来决定出拳。变量 *prev* 记录的是对手先前的出拳信息。

运行代码清单 12-3 的程序后，就会发现计算机变强了（图 12-4）。表 12-4 表示的是对手连续出了 15 次石头时计算机的出拳信息。借助记忆功能，在猜拳游戏进行了 10 次以后，计算机出的都是布，全胜。这是因为计算机基于“对手出石头后还会出石头”这一记忆，做出了出布的判断。

① 有两个索引的数组称为 2 维数组。2 维数组在处理表格形式的数据时很便利。由于 int player[3][3] 数组前后的索引数值分别是 0、1、2，因此就可以用类似于下面这种 3 行 ×3 列的表格形式来进行数据的处理。

再次出拳的次数	出石头的次数	出剪刀的次数	出布的次数
前一回出石头后	player[0][0]	player[0][1]	player[0][2]
前一回出剪刀后	player[1][0]	player[1][1]	player[1][2]
前一回出布后	player[2][0]	player[2][1]	player[2][2]

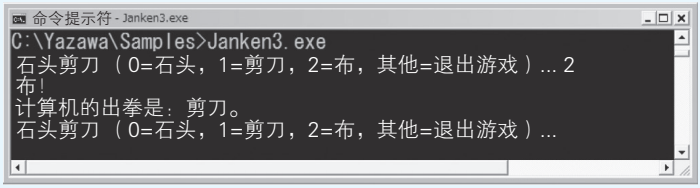


图 12-4　代码清单 12-3 的运行结果

表 12-4　代码清单 12-3 的运行结果和计算机的出拳信息

次数	1	…	10	11	12	13	14	15
出拳信息	剪刀	…	布	布	布	布	布	布

单纯就记忆能力来说，计算机要比人类强大得多。因此，只要对程序进行一些改造，使计算机记住“对手出石头获胜后接下来会出剪刀，出石头输了后接下来会出布”这些细节信息的话，计算机就会更加擅长猜拳游戏了。不过，如果太过于强大的话，可能又会不像人类的思考方式了。

12.6　用程序来表示人类的思考方式

到目前为止，我们已经用程序表示了直觉、想法、习惯以及经验等。不过，除此之外，人类还有一个思考方式。思考方式是思考方法的节奏。人类大脑中有类似于“石头、石头、布、剪刀”或“剪刀、石头、石头、布”这种具有节奏感的短语，人类会在此基础上做出判断，这就是思考方式。

代码清单 12-4 是用程序来实现思考方式的示例。这里用 2 维数组 pttern[2][4] 来表示“石头、石头、布、剪刀”及“剪刀、石头、石头、布”这两种思考方式。人类会在不知不觉中按照自己的思考方式出拳，但连续输掉多次后也会变换一些方式。在该程序中，我们将其设定为

连续输两次就改变思考方式。在时赢时输的情况下，则按照节奏以同一种方式出拳。

代码清单 12-4 根据思考方式来决定出拳的猜拳游戏程序示例

```
#include <stdio.h>
#include <stdlib.h>

void main() {
    // 表示思考方式的 2 维数组
    int pattern[2][4] = { { 0, 0, 2, 1 }, { 1, 0, 0, 2 } };

    // 连续输的次数
    int lose = 0;

    // 用来切换思考方式的变量（0 和 1 之间切换）
    int p = 0;

    // 根据思考方式决定出拳信息
    int n = 0;

    // 对手的出拳
    int human;

    // 计算机的出拳
    int computer;

    // 设定随机数的种子
    srand(time(NULL));

    // 重复猜拳
    while( -1 ) {
        // 对手决定出拳信息
        printf(" 石头剪刀 （0= 石头，1= 剪刀，2= 布，其他 = 退出游戏）...");
        scanf("%d", &human);
        printf(" 布 \n");

        // 输入 0、1、2 以外的数值时游戏结束
        if(human < 0 || human > 2) break;

        // 计算机决定出拳信息
        computer = pattern[p][n];
        n = (n + 1) % 4;

        // 输出计算机的出拳信息
        if (computer == 0) {
            printf (" 计算机的出拳是：石头 \n");
        } else if( computer == 1 ) {
```

```
            printf ("计算机的出拳是：剪刀\n");
        } else {
            printf ("计算机的出拳是：布\n");
        }
        printf("\n");

        // 记录计算机连续输拳的次数
        if ((human == 0 && computer == 1)  ||
            (human == 1 && computer == 2) ||
            (human == 2 && computer == 3)) {
            lose ++ ;
        } else {
            lose = 0;
        }

        // 连续输拳时变换思考方式
        if (lose >= 2) {
            p = (p + 1) % 2;
            n = 0;
        }
    }
}
```

运行该程序后，大家可能就会察觉到“该计算机有自己的出拳方式”。在至今为止我们所介绍的程序中，该示例程序可能最接近人类的思考方式。

大家应该都听过人工智能（AI，Artificial Intelligence）这个术语。**人工智能**是用计算机来实现人类智能的尝试。从计算机诞生之初的 1950 年代开始，关于人工智能的研究就层出不穷，到现在已经有了大量成果。本章介绍的《猜拳游戏》，虽然只是涉及了一点皮毛，但也可以说是人工智能。

不过，计算机本身并不智能，它只是运行了表现人类思考方式的程序而已。也就是说，开发程序的程序员，赋予了计算机这些智能。程序只是将人类的想法在计算机上进行了重现。想到这些，是不是感觉很愉悦呢？

如果是你，你会怎样介绍？

向常光临的酒馆老板讲解计算机的思考机制

小老板：噢，欢迎光临！怎么看起来这么疲惫呢？

笔者：唉！还不是那个策划折腾的！向完全不了解计算机的女高中生和老奶奶说明计算机的机制，真是太折磨人了。

小老板：还真是挺折腾的呢。那么，最后结果咋样啊？

笔者：差不多明白了吧。差不多。

小老板：厉害啊！不过实在不好意思，您这已经累得够呛了，可我也有个问题想请教一下。

笔者：啊，你可饶了我吧。

小老板：可别这么说。来来，先请你喝一杯。

笔者：这样啊，那好吧，你问吧。

小老板：计算机和机器人看起来差不多吧。你说要是在我这个店里面也放一台计算机的话，是不是能帮我做点啥呢？

笔者：虽然计算机也可以和机器人一样智能地使用，不过就算放到你店里也不能立马就帮到你。

小老板：啊！搞不懂，这是为什么呢？用简单的方式给我解释解释吧。

笔者：将来的计算机是怎么样的谁也不知道啊，不过现在的计算机是无法自己思考的。假如要让计算机进行思考的话，就必须要用程序来实现思考步骤。

小老板：程序这个东西我还真不懂。打个比方说，它像什么呢？

笔者：这个程序和运动会及音乐会等的程序是一样的。就是把每一步做什么都按照顺序写下来的文件。把这个文件用和英语相似的程序语法记述下来，就是程序。

小老板：那么，具有思考顺序的程序，能用来做什么呢？