# 综合实验 4：函数、数组及结构体的综合应用

## ——在二维封闭房间中的弹球模拟程序

（实验设计 裴继红）

## 实验任务：

1. 进一步掌握数组的定义与使用；进一步掌握函数的定义和函数调用方法；
2. 学习和掌握结构体（结构体数组）的定义和使用方法。
3. 进一步掌握 C 语言的编程方法；学习动画程序的基本设计思想和方法。
4. 编译并运行你的程序。调试正确后，将必要的文件如项目 project 目录压缩文件上载。
5. **提交正式的实验报告。**实验报告的文件名为“学号姓名 综 4”，如“202x280168 王敬华 综 4.doc”

## 实验说明：

本实验编写在控制台(console)窗口中，模拟若干个弹球在一个二维封闭房间中运动的动画(animation)显示程序。为了简化，假设弹球之间、弹球与四壁间的碰撞都是不损失速度的弹性碰撞，也不考虑弹球重力。本实验涉及到最简单的计算机动画的生成。

### 一、 计算机是如何生成动画的？

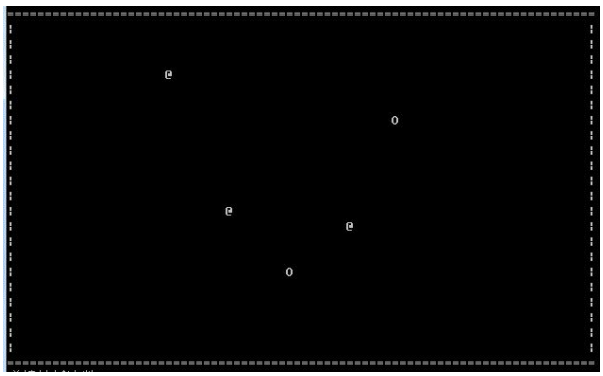
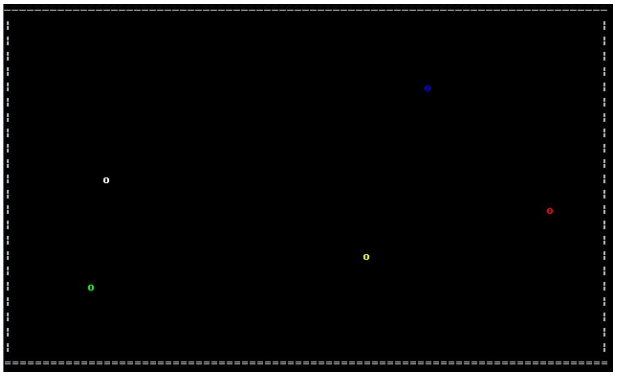
计算机动画显示的原理与电影、电视一样，都是视觉暂留现象（人眼延续约 0.1~0.4 秒时间）。因此将逐帧的图像连续播放的话，就会在视觉上造成一种流畅的活动影像。

在 Windows OS 下控制台(console)默认窗口大小可以显示 25 行 80 列字符，因此可以定义一个 24*80 的二维字符数组 `char cWin[24][80]`来保存每一帧图像的数据。

其中动画中图像的真实大小规定为 24 行 79 列字符。这是为了加快字符数组的显示，使用该二维数组的最后一列 cWin[][79]写入字符串结束标志'\0'（也可写入'\n'），以便可以使用字符串输出函数显示二维字符数组 cWin 的每一行字符。

### 如何利用二维字符数组生成并显示一帧图像？

① 先将动画的背景图像（即二维封闭房间）写入到二维字符数组 cWin 中，初始化图像。②然后，将要绘制的弹球以字符（如字符'o'或字符'@"x'等）的形式写入 cWin 对应位置的元素中，二维数组 cWin 中的每个元素对应于图像的一个像素点。③因此用字符串输出函数，将这个以字符阵列的形式存储在二维数组 cWin 中的图像，显示在屏幕上即可。如下方左图。

注：另外一种显示方法就是先输出二维封闭房间的图像，再将表示弹球的字符（如单字节字符'o'或双字节字符串"●"）直接显示在对应位置的屏幕上。如上方右图。

## 将逐帧的图像连续、延时播放就变成了动画

要想让一个图像序列连续显示时看起来像动画，每一帧图像在屏幕上的停留时间要基本与人眼的视觉暂留时间相适应。因此在显示每一帧图像后，要继续适当延时。在 Windows OS 中可以用头文件<Windows.h>中的函数 Sleep(n)（延时 n 毫秒）来实现。

动画显示的原理用伪代码可简单表示如下：

```
while(没结束){
    清除屏幕，为显示下一帧做准备；
    计算出一帧图像；
    显示这一帧图像，停留若干毫秒；
}
```

## 用什么样的数据结构来描述弹球？

定义描述一个弹球的结构体 BALL，一种可能的形式如下：

```
struct BALL {
  int body[2];//两个不同的字符，分别代表两个不同颜色的球
  int sel;    //当前球的颜色，0表示第一种颜色，1表示第二种颜色
  int wX;     //在二维数组中，球在x方向的实际显示位置（整数）
  int wY;     //在二维数组中，球在y方向的实际显示位置（整数）
  double X;  //球在x方向的精确位置（实数）
  double Y;  //球在y方向的精确位置（实数）
  double dX; //球在x方向的速度（实数）
  double dY; //球在y方向的速度（实数）
};
```

其中屏幕的最左上角为坐标原点 *O*，向下为 x 方向，向右为 y 方向。

1) 对弹球 BALL 结构体的每一个元素进行初始化。为了使模拟程序看起来更自然，我们可以用随机数对其进行初始化：

随机生成 0、1 作为当前弹球的颜色值 sel;

随机生成 1-22 之间的随机数，作为当前弹球的行坐标的位置 wX，X;

随机生成 1-77 之间的随机数，作为当前弹球的列坐标的位置 wY，Y;

每个弹球的速度大小一直都保持为1，但速度的方向θ是一个0-359之间的随机整数，表示角度。这样弹球X、Y方向的速度分量分别为：

```
dX = cos(θ*π/180);
dY = sin(θ*π/180);
```

**如何让一个弹球运动起来？**

2) 弹球根据自己的速度，移动一步

弹球运动的实质是改变弹球当前的位置。由于弹球在X、Y方向的速度分量dX、dY都为 ＜1 的值，因此弹球移动一步后的精确位置X、Y是两个实数分量：

```
X = X + dX;
Y = Y + dY;
```

但是，弹球在二维数组图像中的显示位置是二维数组的行、列两个下标，只能是整数值。因此，需要对弹球当前的精确实数位置进行四舍五入取整，得到实际显示的数组行、列位置wX、wY。可以用下面的方法实现四舍五入取整：

```
wX = (int)(X + 0.5);
wY = (int)(Y + 0.5);
```

**如何检测弹球是否撞到了墙壁？该如何弹回来？**

假设弹球当前的位置是（X, Y），弹球运动一步以后的位置是：

```
X = X + dX;
Y = Y + dY;
```

假设表示图像的二维字符数组有24行，则若 X＜0，则说明弹球撞到了上面的墙壁；若X＞23，则说明弹球撞到了下面的墙壁。

检测到弹球撞墙壁后，弹球应该被弹回。也就是说弹球在X方向的速度分量需要改变方向，Y方向的速度分量不改变。具体可用下面数学模型实现：

```
dX = - dX;
X = X + dX;
```

对弹球在左、右方向（即Y方向）的撞墙检测，以及被弹回的原理同上。

**如何检测两个弹球之间是否相撞？**

首先，根据两个弹球的当前位置（X1, Y1）、（X2, Y2），计算它们之间的距离：

```
dist = sqrt((X1 - X2) * (X1 - X2) + (Y1 – Y2) * (Y1 – Y2));
```

若 dist < 1，则可判定两个弹球相撞。

**说明：**为计算简单起见，dist 也可以用下面的公式计算后判断。（为什么？）

```
dist = (X1 - X2) * (X1 - X2) + (Y1 – Y2) * (Y1 – Y2);
```

**弹球相撞之后如何处理？如何让弹球的速度方向逆时针旋转 90 度？**

为了简化处理，我们假设弹球相撞之后每个弹球都按原来的速度方向逆时针旋转90度后弹开。因此若弹球当前的速度矢量为（dX，dY），则逆时针方向旋转90度后的速度矢量（dX'，dY'）为：

```
dX' = - dY

dY' = dX
```

## 二、 对编程的具体要求：

1) 每个弹球用一个结构体 BALL 变量描述（可参考前面的程序说明），一组弹球用结构体数组定义；
2) 除 main()主函数外，你的程序至少还要编写 4 个以上的自定义子函数。
3) 主函数在一个单独的源文件中；其余子函数放在另外一个源文件中；必要的声明与定义（包括结构体的定义、以及子程序的声明等）放在自定义的一个*.h 头文件中。需要在新建的项目 project 里面这样添加多个文件实现。
4) 本程序应可以模拟 1 到 NUM 个弹球的运动，NUM 是由预处理语句定义的一个整型常量，表示模拟中可能的最多的弹球数。
5) 模拟程序在每次运行时，弹球的实际个数由用户输入。若用户输入的弹球个数小于 1，则设置弹球数量为 1；若用户输入的弹球个数大于 NUM 时，则设置弹球个数为 NUM；若用户输入的弹球个数在 1 到 NUM 之间，则弹球个数为用户输入的个数。
6) 弹球在画面中的初始位置、运动方向（运动速度均为 1.0）、显示的颜色（不同颜色的弹球可以用不同的字符表示）等，在程序运行开始时随机设置。
7) 弹球运动时，如果碰到墙壁则直接弹回；如果碰到地面（下墙壁），则在弹回的同时，要求发出碰撞声，同时统计弹球撞到地面的次数。发出碰撞声可以使用下面语句实现：putchar('\a');
8) 弹球运动过程中，如果两个弹球相撞，两个弹球的速度大小不变，但运动速度的方

向都逆时针旋转 90 度，同时两个弹球的颜色都发生改变。

9) 附件程序 bounceNBalls.exe 是一个可运行的模拟参考程序。

## 三、 拓展编程（选作）：

1) 考虑物理中的动量守恒定律，假如两个球之间的碰撞都是弹性碰撞。碰撞后，如果一个球旋转了 90 度，试根据动量守恒定律，计算出另一个球的运动速度矢量。把你的计算结果应用在该实验中，修改你的程序，观察实验效果。

2) 如果考虑到重力的作用，其他条件不变，如何设计你的程序？谈谈你的设计思想。

3) 假设你模拟的是在封闭盒子中游走的一组贪食蛇，如何设计你的程序？谈谈你的设计思想。

4) 假设你模拟的是在封闭盒子中的贪食蛇吃老鼠，如何设计你的程序？谈谈你的设计思想。

## 四、 本实验编程的一些提示：

主程序 main 功能：可包括随机数发生器初始化，确定接受用户输入的弹球数，初始化弹球参数，控制模拟程序的循环运行，输出弹球碰撞地面的总次数，等。

你的程序中可能用到的一些子函数：

```
void initBall(struct BALL ball[], int num)
```

功能：对num个弹球的数据进行初始化。

```
int moveBall(struct BALL ball[], int num)
```

功能：将num个弹球根据各自的速度移动一次。返回本次运动中弹球碰撞地面的次数。
注意：若弹球撞边界墙，则直接被弹回，速度分量需要改变方向（变相反数）；
特别的若弹球撞到地面（下面的墙），则被弹回的同时，发出弹回的声音。
若两个弹球相撞，则两个弹球都改变自身的颜色，以逆时针旋转90度改变速度的方向（直接设定而非物理规律）。

```
void initCharPicture(char cWin[24][80])
```

功能：将二维图像数组初始化为房间背景图模式。上下边界可以用-=*等，左右边界可以用|][等。最后一列全部赋值为'\0'或者'\n'，最后一个元素赋值为'\0'以方便输出。

```
void redrawCWin(struct BALL ball[], int num)
```

功能：根据num个弹球的最新位置，在屏幕上绘制出当前弹球位置的图像。

```
double distBetweenBalls(struct BALL ball[], int i, int j)
```
功能：计算弹球 i 和弹球 j 之间的距离的平方，用来判断是否相互碰撞。不同的弹球之间的距离小于1.0时判定为相互碰撞

```
void delay(int nTime)
```
功能：延时一段时间。其中，参数 nTime 调节延时的长度。

```
int randX(void)
```
功能：生成一个随机数，作为弹球的初始行位置。

```
int randY(void)
```
功能：生成一个随机数，作为弹球的初始列位置。

```
int randA(void)
```
功能：生成0-359之间的一个随机数，作为弹球的运动方向角度。