4.7 函数应用举例

郭延文

ywguo@nju.edu.cn

2019级计算机科学与技术系

例题1：计算常用几何图形的面积

- 编写程序，计算六种常用几何图形:圆形、扇形、矩形、平行四边形、三角形和梯形的面积,输出所求出的面积值。

- 具体要求：
  - 计算这些图形面积所需的参数由用户通过键盘输入；
  - 可反复多次计算图形的面积：
    - 程序不是每计算完一次就结束运行，而是等待用户的下一次选择（不需要每次计算图形面积都重新启动运行程序）；
  - 不考虑输入数据的合法性验证。

算法分析

- 以计算为主的例题。在这个例题中能接触到一些比较简单的算术运算、赋值运算及其相关表达式。

  主函数内的程序结构是什么？

- 函数是C++程序的重要组件，功能相对独立的程序段。计算每一种图形的面积都是功能相对独立的程序段，所以可以考虑将计算不同图形面积的方法设计成独立的函数。所以，**这个程序的总体结构：一个由main函数和6个求面积的函数构成。**

计算面积函数举例

```
float circle(float r) /* 计算圆面积 */
{
    return PI*r*r;
}

float sector(float r, float d) /* 计算扇形面积 */
{
    return PI*r*r*d / 360;
}
```

算法分析

- 这是一个典型的以计算为主的例题。在这个例题中能接触到一些比较简单的算术运算、赋值运算及其相关表达式。

- 函数是C++程序的重要组件，功能相对的独立的程序段。计算圆形、扇形、矩形、平行四边形、三角形和梯形的面积都是功能相对独立的程序段，所以可以考虑将计算这些图形面积的方法设计成独立的函数。所以，**这个程序的总体结构应该是一个由main函数和 6个求面积的函数构成。**

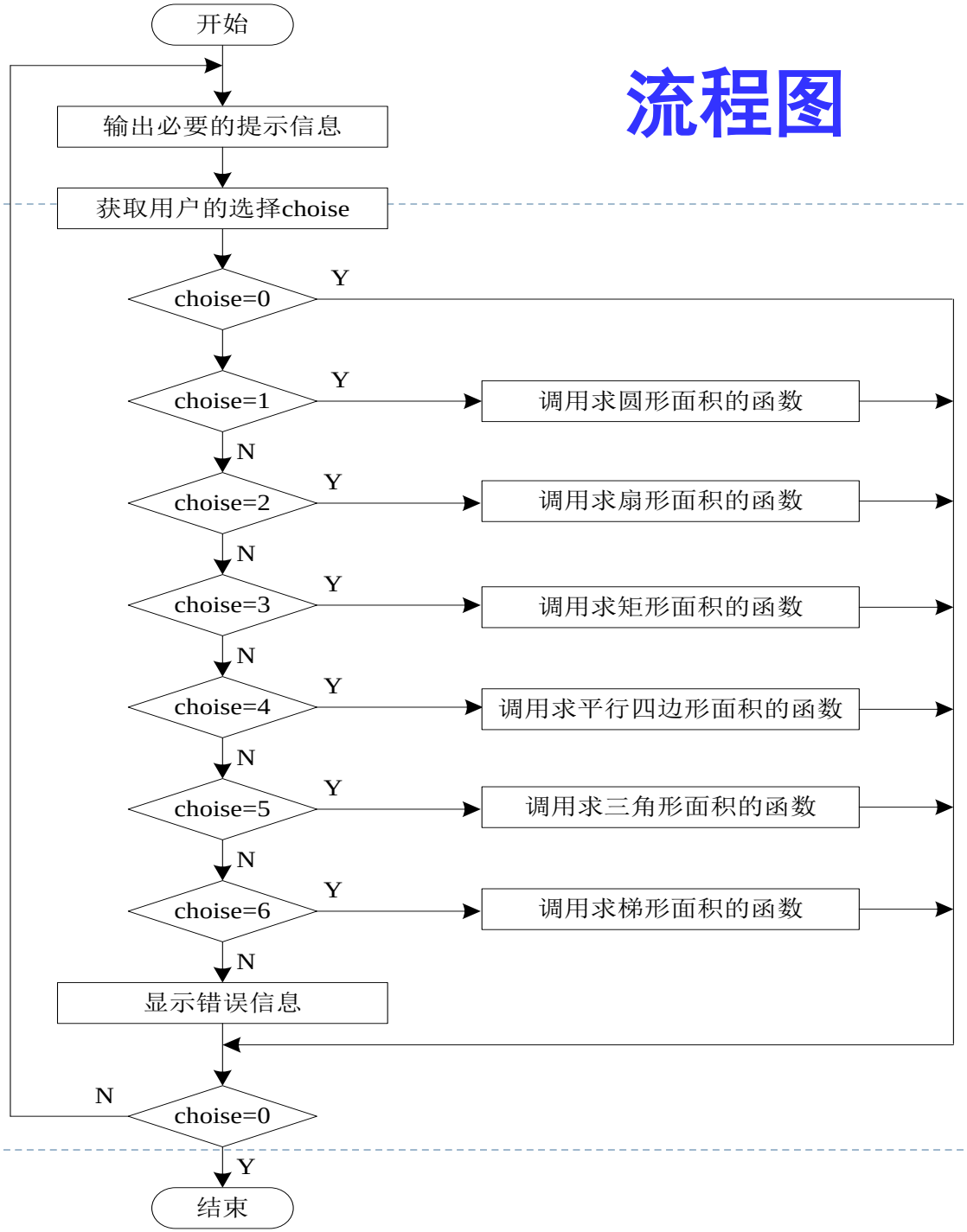
  如何根据用户需求，选择一个面积计算函数?

- main函数中必须接受用户的选择，所以在main函数中应该设计相关的语句来接受用户的输入。然后应该是一个**6个方向的分支**。根据选择的不同，决定调用6个求面积函数中的某一个。

算法分析

- **程序要求：可反复多次计算图形的面积。**
  - 可考虑把main函数的总流程设计成一个循环，每次执行完一个函数后，再回到起点，等待用户再次选择。这样用户就可以反复计算多个图形的面积了，而不需要每计算一次都重新运行一次程序。
- **程序如何结束？**
  - 解决的方法很多，一种很简便的方法就是专门设置一个退出的选项。

流程图


开始
输出必要的提示信息
获取用户的选择choise
choise=0
Y
choise=1
Y
调用求圆形面积的函数
N
choise=2
Y
调用求扇形面积的函数
N
choise=3
Y
调用求矩形面积的函数
N
choise=4
Y
调用求平行四边形面积的函数
N
choise=5
Y
调用求三角形面积的函数
N
choise=6
Y
调用求梯形面积的函数
N
显示错误信息
choise=0
N
Y
结束

具体实现

```
int main()
{/* 选择0，则程序结束 */
    int choice;
    float r, d, a, b, h;
    do {
        cout << "请选择计算面积的图形种类:";
        cin >> choice;
        switch (choice)
        {    /* 根据用户的选择，决定执行相应的函数

        case 0: break;
        case 1:
                cout << "请输入半径：";
                cin >> r;
                cout<<circle(r);
                break;          /* 圆形 */
        case 2:
                cout << "请输入半径和角度：";
                cin >> r >> d;
                cout<<sector(r, d);
                break;          /* 扇形*/
        case 3:
                cout << "请输入长和宽：";
                cin >> a >> b;
                cout<<rectangle(a,b); break;      /* 矩形*/
        case 4:
                cout << "请输入底边长和高：";
                cin >> a >> h;
                cout << parallelogram(a,h);
                break; /* 平行四边形 */
        case 5:
                cout << "请输入底边长和高：";
                cin >> a >> h;
                cout << triangle(a,h);
                break;      /* 三角形 */
        case 6:
                cout << "请输入上边长，下边长和高：";
                cin >> a >> b >> h;
                cout<<trapezia(a,b,h);
                break;      /* 梯形 */
        default: cout << "选择错误，请重新选择！\n";
        }
    } while (choice != 0);
    cout << "程序运行结束";
    return 0;
}
```

具体实现

```
float circle(float r) /* 计算圆面积 */
{
    return PI*r*r;
}

float sector(float r, float d) /* 计算扇形面积 */
{
    return PI*r*r*d / 360;
}
```

具体实现

```
float rectangle(float a, float b) /* 计算矩形面积 */
{
    return a*b;
}

float parallelogram(float a, float h) /* 计算平行四边形面积 */
{
    return a*h;
}
```

具体实现

```
float triangle(float a, float h) /* 计算三角形面积 */
{
    return a*h / 2;
}

float trapezia(float a, float b, float h) /* 计算梯形面积 */
{
    return (a + b)*h / 2;
}
```

再回顾：主函数

注意do while用法

switch用法

```
int main()
{/* 选择0，则程序结束 */
    int choice;
    float r, d, a, b, h;
    do {
        cout << "请选择计算面积的图形种类:";
        cin >> choice;
        switch (choice)
        {    /* 根据用户的选择，决定执行相应的函数 */

        case 0: break;
        case 1:
                cout << "请输入半径：";
                cin >> r;
                cout<<circle(r);
                break;          /* 圆形 */
        case 2:
                cout << "请输入半径和角度：";
                cin >> r >> d;
                cout<<sector(r, d);
                break;          /* 扇形*/
        case 3:
                cout << "请输入长和宽：";
                cin >> a >> b;
                cout<<rectangle(a,b); break;      /* 矩形*/
        case 4:
                cout << "请输入底边长和高：";
                cin >> a >> h;
                cout << parallelogram(a,h);
                break; /* 平行四边形 */
        case 5:
                cout << "请输入底边长和高：";
                cin >> a >> h;
                cout << triangle(a,h);
                break;       /* 三角形 */
        case 6:
                cout << "请输入上边长，下边长和高：";
                cin >> a >> b >> h;
                cout<<trapezia(a,b,h);
                break;       /* 梯形 */
        default: cout << "选择错误，请重新选择！\n";
        }
    } while (choice != 0);
    cout << "程序运行结束";
    return 0;
}
```

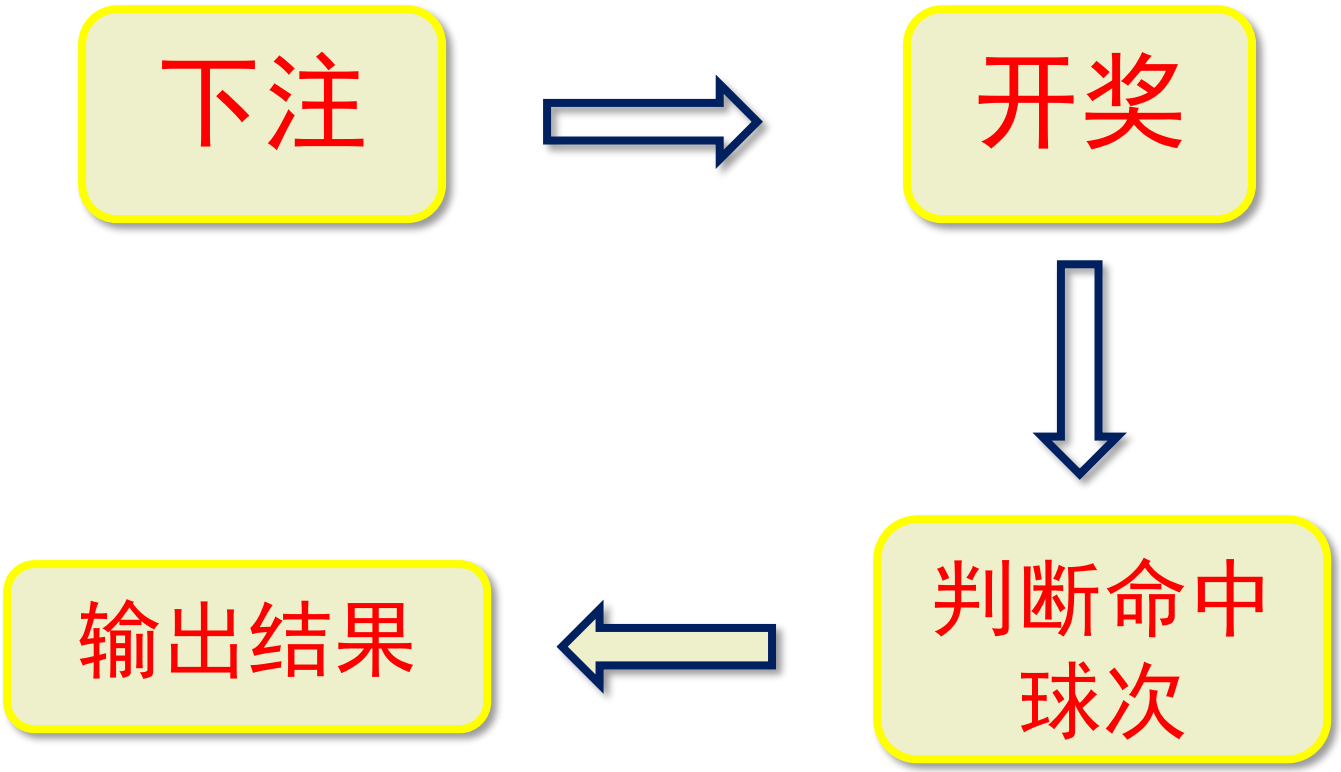
例题2：双色球彩票

- **投注方式**
  - 双色球彩票投注区分为**红色球号码区**和**蓝色球号码区**；
  - 每注投注号码由6个红色球号码和1个蓝色球号码组成；
  - 红色球号码从1-33中选择；蓝色球号码从1-16中选择。
- **开奖方式**
  - 随机摇号，号码各不相同。
- **获奖方式**
  - 一等奖：7个号码相符（红色球号码顺序不限，下同）；
  - 二等奖：6个号码相符（6个红色球号码相符，或5个红色球号码和1个蓝色球号码相符）；
  - 三等奖：5个号码相符（5个红色球号码或4个红色球号码和1个蓝色球号码相符）；
  - 四等奖：1个蓝色球号码相符（有无红色球号码相符均可）。

要 求

- **编写一个程序，模拟投注和开奖的过程：**
  - 由用户输入7个号码投注，要区分红色和蓝色号球；
  - 由计算机产生7个随机数，也要区分红色和蓝色号球；
  - 根据规则，判断获奖情况；
  - 假设投注的红球没有重复数字。

买彩票、摇奖、中奖的过程



主函数基本结构

- main函数中主要包含（调用）下列函数
  - 下注函数
  - 开奖函数
  - 判断投注号码命中号球的次数
  - 判断中奖情况并输出结果

算法分析

如何模拟投注过程，选号？

- 投注的过程可以设计为由用户从键盘输入7个整数，判断所输入数据的合法性；

下注函数

```
/* 下注 */
void chipin()
{    cout<<"红球1：";
     cin>>ured1;
     cout<< "红球2：";
     cin>>ured2;
     cout<<"红球3：";
     cin>>ured3;
     cout<<"红球4：";
     cin>>ured4;
     cout<<"红球5：";
     cin>>ured5;
     cout<<"红球6：";
     cin>>ured6;
     cout<<"蓝球：";
     cin>>ublue;
     return;
}
```

1-33的
如何加上判断数据的合法性？

程序如何自动开奖？

- 开奖的过程可以使用随机函数，随机生成生成7个在一定范围内的正整数；

- 需要以下函数支持：
  - 生成要求范围内的随机整数的函数
  - 判断生成的整数是否已经存在的函数
  - 返回一个不重复的随机正整数

开奖函数

```
/* 摇奖 */
void ernie()
{
    /* 将六个红球初始化成不在MINRED – MAXRED之间的某个值 */
    red1=red2=red3=red4=red5=red6=INIT;      //如-10000
    /* 生成不重复的红球中奖号 */
    red1=getDig(red1,red2,red3,red4,red5,red6);
    red2=getDig(red1,red2,red3,red4,red5,red6);
    red3=getDig(red1,red2,red3,red4,red5,red6);
    red4=getDig(red1,red2,red3,red4,red5,red6);
    red5=getDig(red1,red2,red3,red4,red5,red6);
    red6=getDig(red1,red2,red3,red4,red5,red6);
    blue=genDigital(1,16);
    return;
}
```

```
/* 返回一个与目前红球不重复的随机正整数 */
int  getDig(int r1,int r2,int r3,int r4,int r5,int r6)
{
        int   tmp;
        do{
         /* 生成min – max间的随机正整数 */
            tmp = genDigital(1,33);
        }while (isExists(tmp,r1,r2,r3,r4,r5,r6)==1);
        return tmp;
}

/* 生成min – max间的随机正整数 */
int genDigital(int min,int max)
{
        return   rand()%(max-min)+min;
}
```

```
int    isExists(int x,int rd1,int rd2,int rd3,
                int rd4,int rd5,int rd6)
//判断某一元素在已经生成的红球序列中是否已经存在
{
        if(x==rd1||x==rd2||x==rd3||x==rd4||x==rd5||x==rd6)
                return   1;
        else
                return   0;
}
```

如何判断中奖？

- 设置两个变量bcount=0和rcount=0，分别记录蓝色和红色号球相同号码的个数。
- 判断彩民投注的蓝色号球是否命中，如命中，置bcount=1；
- 逐个判断用户投注的6个红色号球是否命中；
  - 红色号球不分次序，所以不能进行对应位置的判断，必须把彩民投注的6个号码依次和每个号球比较；
  - 如果存在，记录命中一次rcount++，并且要把这个号球的号码删除，防止用户投注的号码中有重号；
  - 如果不存在，不进行任何处理。
- 最后根据bcount和rcount的值，判断中奖等级。

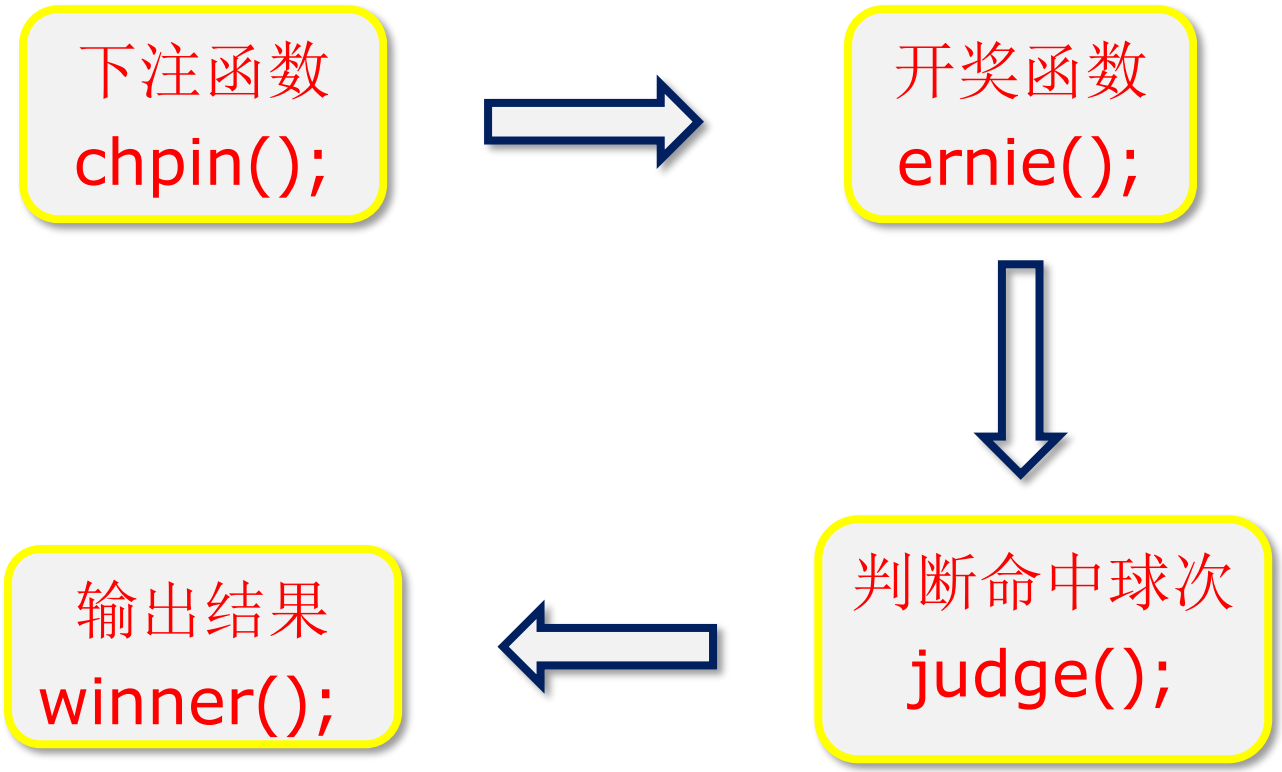
统计命中球次的函数

```
/* 统计匹配的数字数量 */
void judge()
{
        bcount=rcount=0;
        if (ublue==blue)
                bcount++;
        if (isExists(ured1,red1,red2,red3,red4,red5,red6)==1)
                rcount++;
        if (isExists(ured2,red1,red2,red3,red4,red5,red6)==1)
                rcount++;
        if (isExists(ured3,red1,red2,red3,red4,red5,red6)==1)
                rcount++;
        if (isExists(ured4,red1,red2,red3,red4,red5,red6)==1)
                rcount++;
        if (isExists(ured5,red1,red2,red3,red4,red5,red6)==1)
                rcount++;
        if (isExists(ured6,red1,red2,red3,red4,red5,red6)==1)
                rcount++;
}
```

中奖函数

```
/* 根据传入的红球、蓝球匹配情况，输出相应的获奖信息 */
void winner(int rcnt,int bcnt)
{
        int cnts=rcnt+bcnt;
        if (cnts==7)
              cout<<"恭喜，您获得了一等奖!\n";
        else if (cnts==6)
              cout<<"恭喜，您获得了二等奖!\n";
        else if (cnts==5)
             cout<<"恭喜，您获得了三等奖!\n";
        else if (bcnt==1)
             cout<<"恭喜，您获得了四等奖!\n";
        else
             cout<<"抱歉，此次您未能获奖。祝下次中奖！\n";
}
```

算法分析



具体程序实现

```
int red1, red2, red3, red4, red5, red6, blue;
int ured1, ured2, ured3, ured4, ured5, ured6, ublue;
int rcount, bcount;
void main()
 {
        /* 设定时间种子，保证后续生成的随机数在重复执行时能有所区别 */
       srand((unsigned)time(NULL));
       chipin();   /* 下注 */
       cout<<"您投注的号码是：";
       cout<<ured1<<ured2<<ured3<<ured4<<ured5<<ured6;
       cout<<"蓝球："<<ublue;
       ernie();    /* 摇奖 */
       cout<<"\n本期中奖号码是：\n";
       cout<<"红球："<<red1<<red2<<red3<<red4<<red5<<red6;
       cout<<"蓝球："<<blue;
       judge();    /* 判断获奖类型 */
       winner(rcount,bcount);
 }
```

具体程序实现

```
/* 下注 */
void chipin()
{       cout<<"红球1：";
        cin>>ured1;
        cout<< "红球2：";
        cin>>ured2;
        cout<<"红球3：";
        cin>>ured3;
        cout<<"红球4：";
        cin>>ured4;
        cout<<"红球5：";
        cin>>ured5;
        cout<<"红球6：";
        cin>>ured6;
        cout<<"蓝球：";
        cin>>ublue;
        return;
}
```

上机修改程序，

添加判断数据合法性的语句

具体程序实现

```
/* 摇奖 */
void ernie()
{
    /* 将六个红球初始化成不在MINRED – MAXRED之间的某个值 */
        red1=red2=red3=red4=red5=red6=INIT;          //如-10000
        /* 生成不重复的红球中奖号 */
        red1=getDig(red1,red2,red3,red4,red5,red6);
        red2=getDig(red1,red2,red3,red4,red5,red6);
        red3=getDig(red1,red2,red3,red4,red5,red6);
        red4=getDig(red1,red2,red3,red4,red5,red6);
        red5=getDig(red1,red2,red3,red4,red5,red6);
        red6=getDig(red1,red2,red3,red4,red5,red6);
        blue=genDigital(1,16);
        return;
}
```

具体程序实现

```
/* 返回一个与目前红球不重复的随机正整数 */
int getDig(int r1,int r2,int r3,int r4,int r5,int r6)
{
        int tmp;
        do{
         /* 生成min – max间的随机正整数 */
                tmp=genDigital(1,33);
   }while (isExists(tmp,r1,r2,r3,r4,r5,r6)==1);
        return tmp;
}
```

具体程序实现

```
int isExists(int x,int rd1,int rd2,int rd3,
             int rd4,int rd5,int rd6)
//判断某一元素在已经生成的红球序列中是否已经存在
{
    if(x==rd1||x==rd2||x==rd3||x==rd4||x==rd5||x==rd6)
    return 1;
    else
    return 0;
}

/* 生成min – max间的随机正整数 */
int genDigital(int min,int max)
{
    return rand()%(max-min+1)+min;
}
```

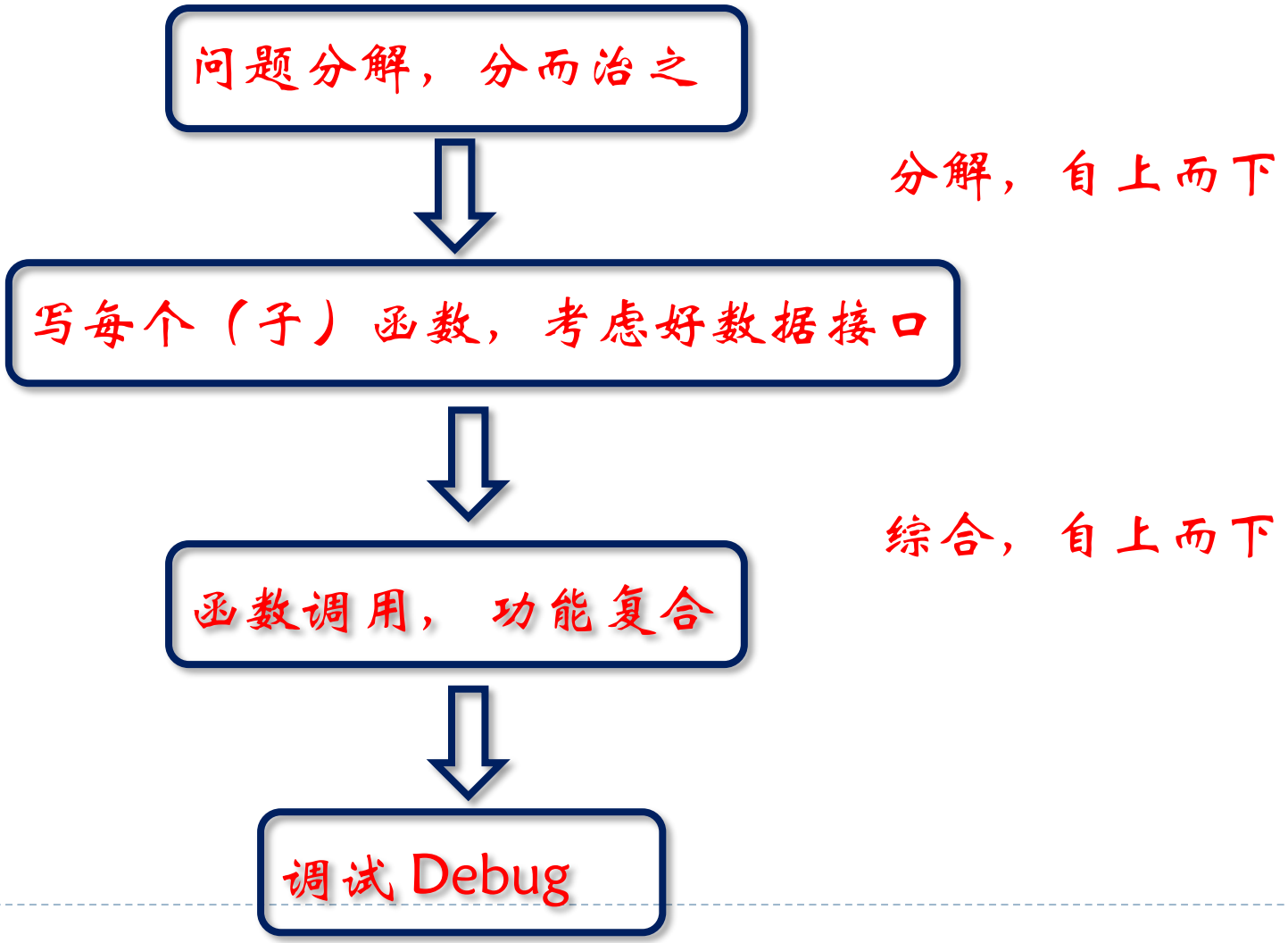
具体程序实现

```
/* 统计匹配的数字数量 */
void judge()
{
        bcount=rcount=0;
        if (ublue==blue)
            bcount++;
        if (isExists(ured1,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured2,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured3,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured4,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured5,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured6,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
}
```

具体程序实现

```
/* 根据传入的红球、蓝球匹配情况，输出相应的获奖信息 */
void winner(int rcnt,int bcnt)
{
        int cnts=rcnt+bcnt;
        if (cnts==7)
            cout<<"恭喜，您获得了一等奖!\n";
        else if (cnts==6)
            cout<<"恭喜，您获得了二等奖!\n";
        else if (cnts==5)
            cout<<"恭喜，您获得了三等奖!\n";
        else if (bcnt==1)
            cout<<"恭喜，您获得了四等奖!\n";
        else
            cout<<"抱歉，此次您未能获奖。祝下次中奖！\n";
}
```

分析问题、编写程序



再回顾 摇奖过程

```
/* 下注 */
void chipin()
{       cout<<"红球1：";
        cin>>ured1;
        cout<<"红球2：";
        cin>>ured2;
        cout<<"红球3：";
        cin>>ured3;
        cout<<"红球4：";
        cin>>ured4;
        cout<<"红球5：";
        cin>>ured5;
        cout<<"红球6：";
        cin>>ured6;
        cout<<"蓝球：";
        cin>>ublue;
        return;
}
```

上机修改程序，
添加判断数据合法性的语句

再回顾 摇奖过程

```
/* 摇奖 */
void ernie()
{
    /* 将六个红球初始化成不在MINRED – MAXRED之间的某个值 */
        red1=red2=red3=red4=red5=red6=INIT;          //如-10000
        /* 生成不重复的红球中奖号 */
        red1=getDig(red1,red2,red3,red4,red5,red6);
        red2=getDig(red1,red2,red3,red4,red5,red6);
        red3=getDig(red1,red2,red3,red4,red5,red6);
        red4=getDig(red1,red2,red3,red4,red5,red6);
        red5=getDig(red1,red2,red3,red4,red5,red6);
        red6=getDig(red1,red2,red3,red4,red5,red6);
        blue=genDigital(1,16);
        return;
}       // 后续学了数组实现会简单
```

再回顾 摇奖过程

```
/* 统计匹配的数字数量 */
void judge()
{
        bcount=rcount=0;
        if (ublue==blue)
            bcount++;
        if (isExists(ured1,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured2,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured3,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured4,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured5,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
        if (isExists(ured6,red1,red2,red3,red4,red5,red6)==1)
            rcount++;
}       // 这种实现 如果用户重复买一个数字会怎么样？
```

上机习题1： 改写该程序

1. 程序能验证用户输入的数据合法性（不能超出有效范围1~33，但可以重复投）

2. 当判断用户投注的一个号码命中后，避免用户重复下注导致的“重复命中”

例题3：计算上楼的走法

楼梯共有n阶台阶，上楼可以一步上1个台阶，也可以一步上2个台阶。

请编一程序计算共有多少种不同的走法？

算法分析

- 由题可得，假如楼梯只有一台阶那么很简单有一种走法，如果有两个台阶只有简单的两种走法。随着阶数的增大，直接计算出多少走法是很难的事。

- 如果有n台阶时，要么是从n-1直接上到n阶，要么是n-2走两个台阶上到n阶；有n-1台阶时，要么是从n-2直接上到n-1阶，要么是n-3走两个台阶上到n阶……

从简化情况入手

WalkMethods（1）=1;

WalkMethods（2）=2;

WalkMethods（3）=3;

WalkMethods（3）= WalkMethods(1)+ WalkMethods(2) ;

算法分析

✱由上面分析可得：

WalkMethods（n） =WalkMethods(n-1)+ WalkMethods(n-2) ;
WalkMethods（n-1）=WalkMethods(n-2)+ WalkMethods(n-3) ;
WalkMethods（n-2）=WalkMethods(n-3)+ WalkMethods(n-4) ;

.
.
.

WalkMethods（3）=WalkMethods(2)+ WalkMethods(1) ;
WalkMethods（2）=2;
WalkMethods（1）=1;

可以看出是典型的**递归算法**，
递归的结束条件是n=2和n=1

具体实现

```
int WalkMethods (int n)
{
    if(n==1)        //终止条件
            return 1;
    else if(n==2)
            return 2;
    else            //递归条件
            return (WalkMethods(n-1) +
                WalkMethods(n-2));
 }
```

上机习题2：计算上楼梯的方法

楼梯共有n阶台阶,上楼可以一步上1个台阶,也可以一步上2个台阶，也可以一步上3个台阶。

请编一程序计算共有多少种不同的走法？

Q & A