中山大學
SUN YAT-SEN UNIVERSITY

程序设计II实验

电子与信息工程学院（微电子学院）

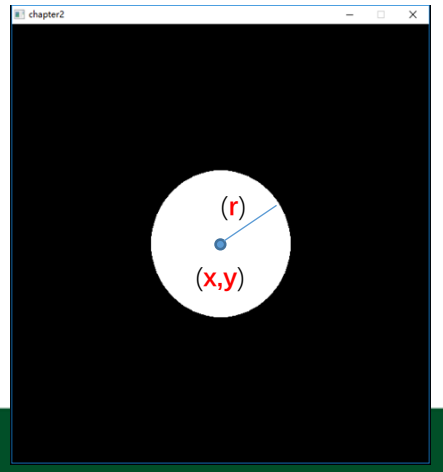
庞志勇 高级实验师 焦涵、桂海田 博士研究生

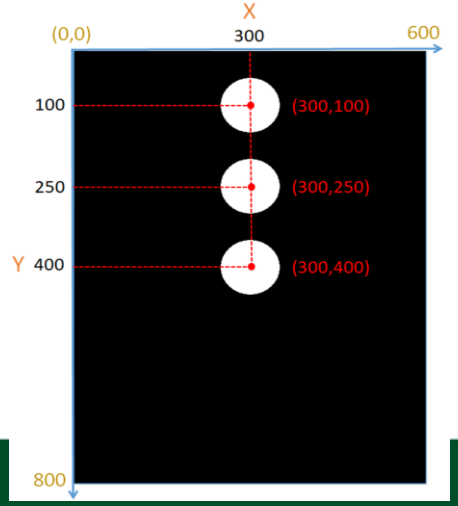
PPT大纲

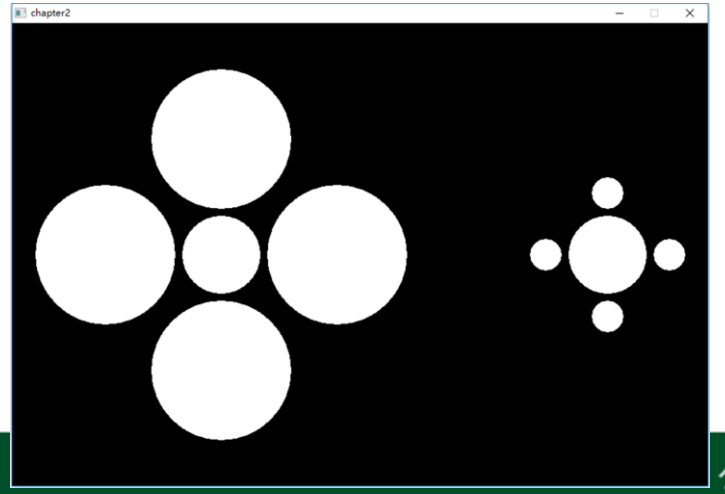
- 上次课实验报告讲解与总结
- 本节课的实验介绍

数学是一切科学的基础

- 这句话是达芬奇的名言。
- 达·芬奇认为，在科学中，凡是用不上数学的地方，凡是和数学没有联系的地方，都是不可靠的，数学是一切科学的基础。他画的各种机械平面图、动植物图，他所从事的每一项工作，没有哪一项不得益于数学的准确性。
- 达·芬奇在哥伦布之前，就算出了地球半径为6000余千米。他还发现了立体几何中正六面体、球体和圆柱之间的面积规律。
- 下面的小球图案动画实现本质上都是数学应用的体现。我们把小球控制最终抽象为基本数学运算。
- void fillcircle(int x, int y, int radius);*//有边框的填充圆 ， 画小球或圆数学本质我们只关系（X,Y）中心坐标和半径R就可以了，也就是说我们把圆抽象为三个数学变量就可以了*







实验四、选择结构程序设计

一、实验目的

1. 了解 C ++语言表示逻辑量的方法（以 0 代表“假”，以非 0 代表“真”）。
2. 掌握关系表达式和逻辑表达式的使用。
3. 熟练使用if语句进行程序设计。
4. 结合程序掌握一些简单的算法。
5. 进一步学习VC++单步调试程序的方法。
6. 学会使用选择结构解决一般的实际问题，能编写简单的应用程序。

实验五 循环结构程序设计

一、实验目的

1．理解结构化程序设计方法的循环结构
2．掌握用 while 语句实现循环结构
3．掌握用嵌套while语句实现循环结构
4． 使用循环来解决问题
5．通过循环语句初步了解算法复杂度
6．掌握VC6跟踪调试循环语句方法

要做哪些实验？

第4章 面向过程编程实验

- 实验一、VC6使用与cout输出程序设计
- 实验二、数据类型、常量、变量、表达式
- 实验三、输入输出流
- 实验四、选择结构程序设计
- 实验五 循环结构程序设计
- 实验六 控制结构综合实验
- 实验七 函数实验
- 实验八 作用域、生存期及函数实验
- 实验九 数组实验
- 实验十 指针实验
- 实验十一 结构体（记录）实验

课程设计I

第5章 面向对象编程实验

- 实验一 类与对象
- 实验二 函数重载与运算符重载
- 实验三 继承与派生
- 实验四 多态性与虚函数
- 实验五 模板与STL
- 实验六 流类库与文件操作
- 实验七 异常处理

课程设计II

实验四 ~ 六

- 实验内容根据理论课内容进行调整
- 每次课实验内容可以根据自己学习情况动态调整
- 课程设计可以自选其他内容

程序

➢ **程序是什么？**

程序是解决某种问题的一组指令的有序集合。

著名计算机科学家沃思（Nikiklaus Wirth）提出一个公式：

程序 = 数据结构 + 算法

结论1：学好C/C++语言首先就必须十分了解数据类型、运算符与表达式。

对数据的描述。在C语言中，体现为数据类型的描述！

<<数据结构>>

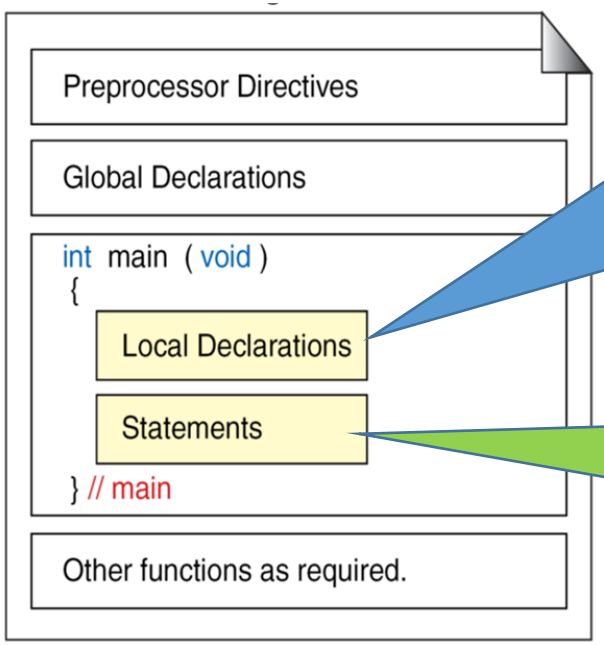
结论2：？

对数据处理的描述。是为解决一个问题而采取的方法和步骤，是程序的灵魂！

<<算法设计与分析>>

C/C++语言程序结构和构成基本要素

C++语言程序框架



- 数据类型
- 标识符
- 常量
- 变量
- 运算符和表达式
- 语句
- 输入输出函数
- 顺序选择循环结构

C++语言程序框架

```
//程序实现屏幕输出Hello World!
#include<iostream>    //头文件
using namespace std;  //使用命名空间
int main()            //main主函数
{


cout<<"Hello,World! "<<endl;
return 0;
}
```

以上内容按照程序设计语言的基本构成要素安排

➢ 实验原理—C++流控制结构

◆语句(statement)是C++程序的独立指令。

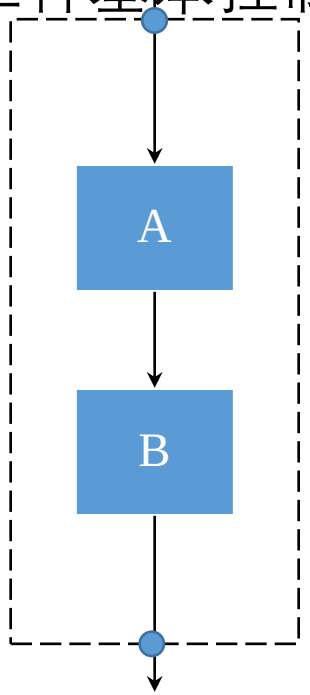
◆程序并不局限于线性（顺序）语句序列。

◆在其过程中，程序可能会重复代码段，或做出决定并分叉。

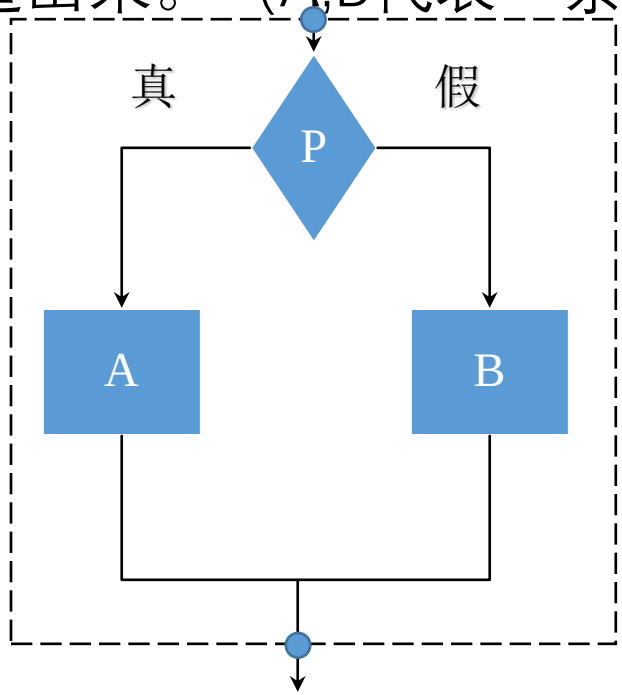
◆为此，C++提供了流控制语句，用于指定程序什么时候什么条件下必须执行什么操作。

程序（算法）三种基本结构

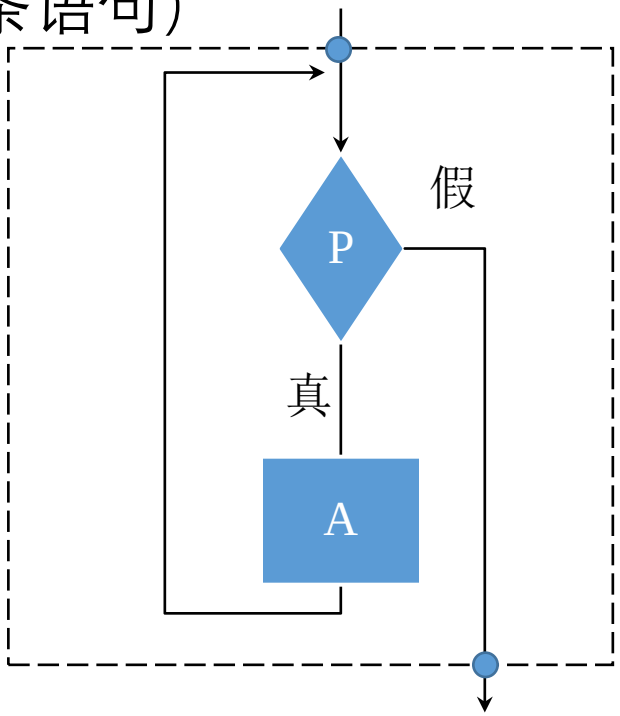
Bohm和Jacopini在1966年证明了：任何单入口单出口，且没有死循环的程序都能由三种基本控制结构构造出来。（A,B代表一条或多条语句）



顺序结构

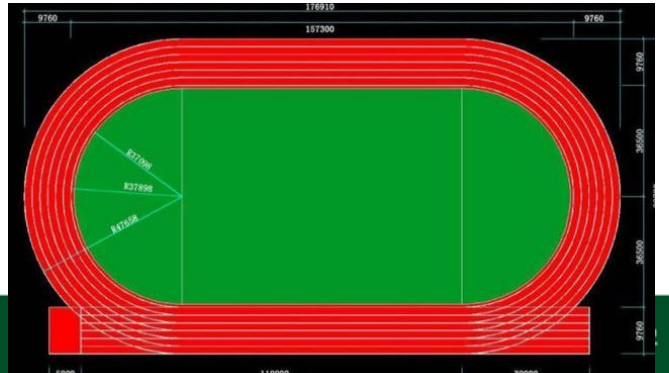


选择结构

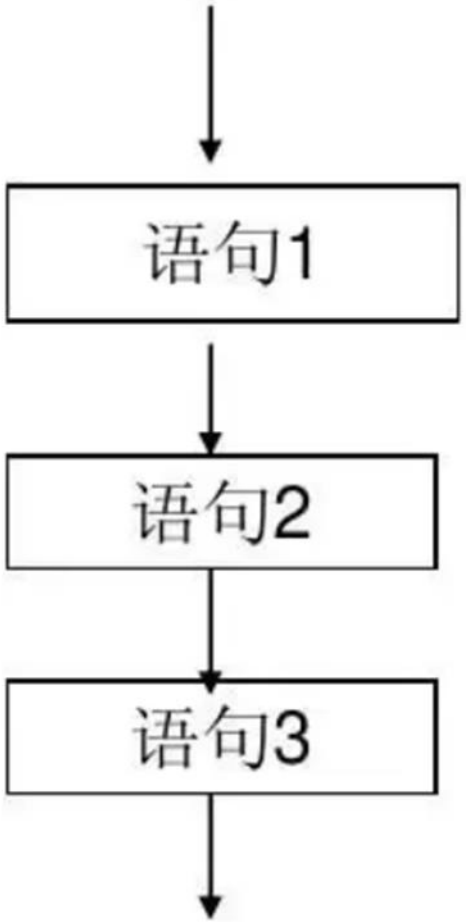


循环结构

C程序多条语句可以根据需要顺序执行、选择执行、循环执行。



顺序结构

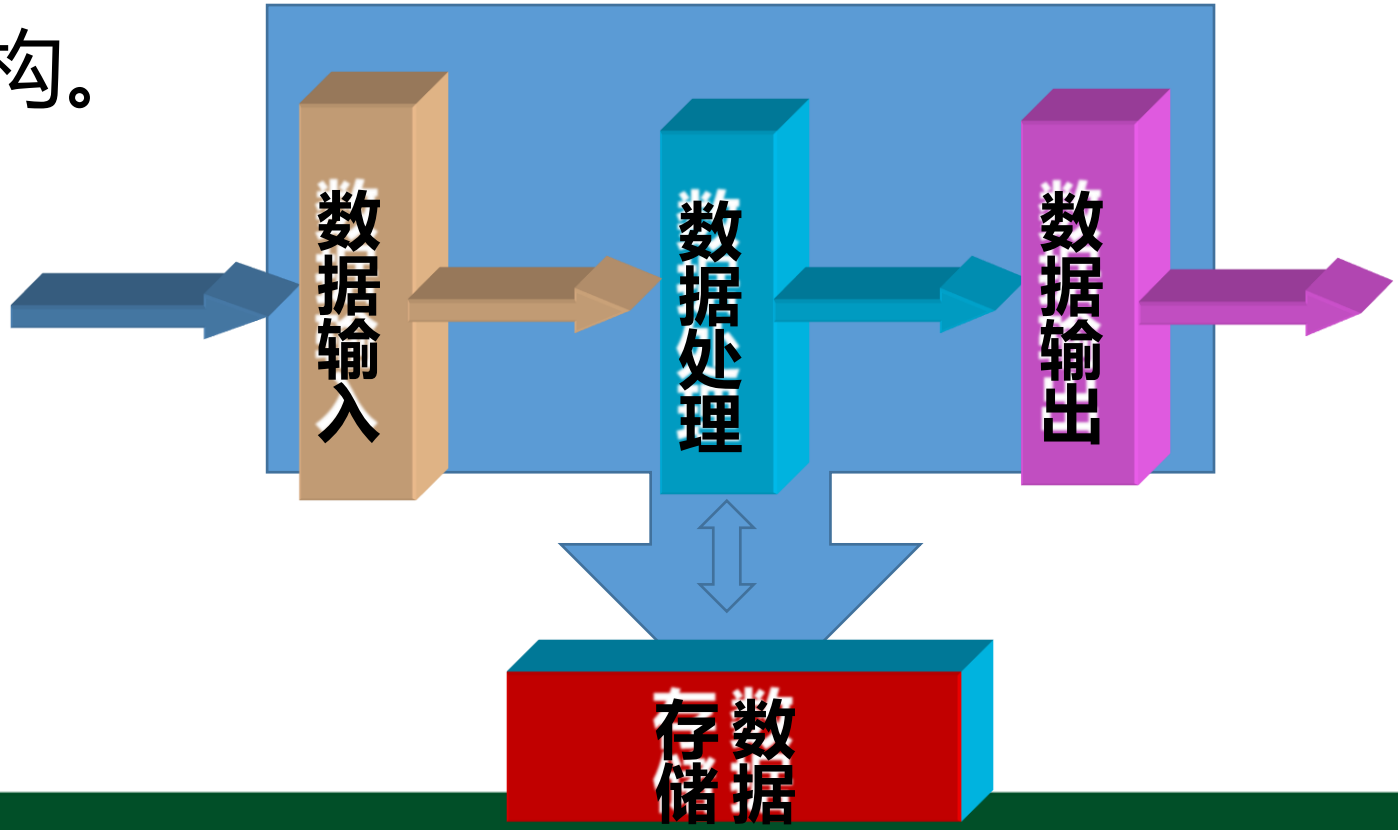


顺序结构

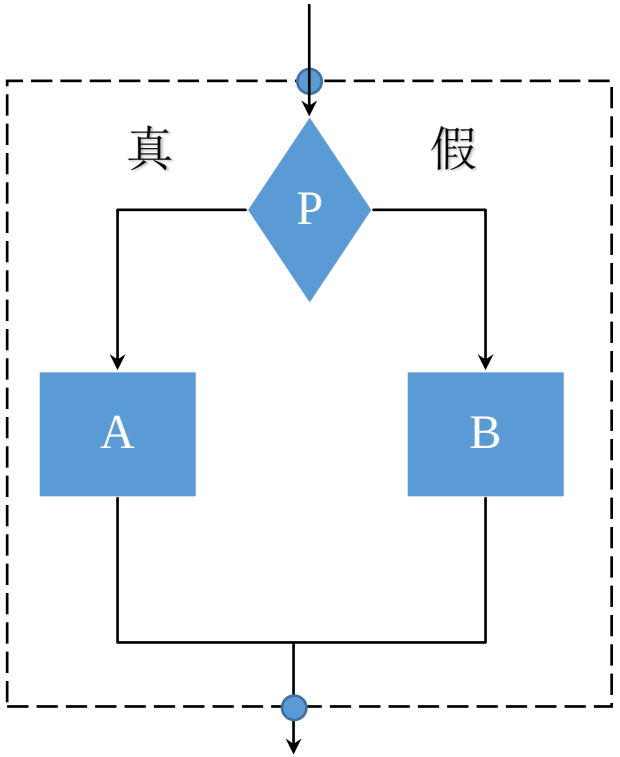
顺序结构的程序设计是最简单的，只要按照解决问题的顺序写出相应的语句就行，它的执行顺序是自上而下，依次执行。顺序结构，就是一条大路走到底，没有岔路口，一步步从上往下执行即可。，

顺序结构程序三步曲

顺序结构可以独立使用构成一个简单的完整程序，常见的数据输入（数据存储变量）、数据计算、数据输出三步曲的程序就是顺序结构。



选择结构程序设计

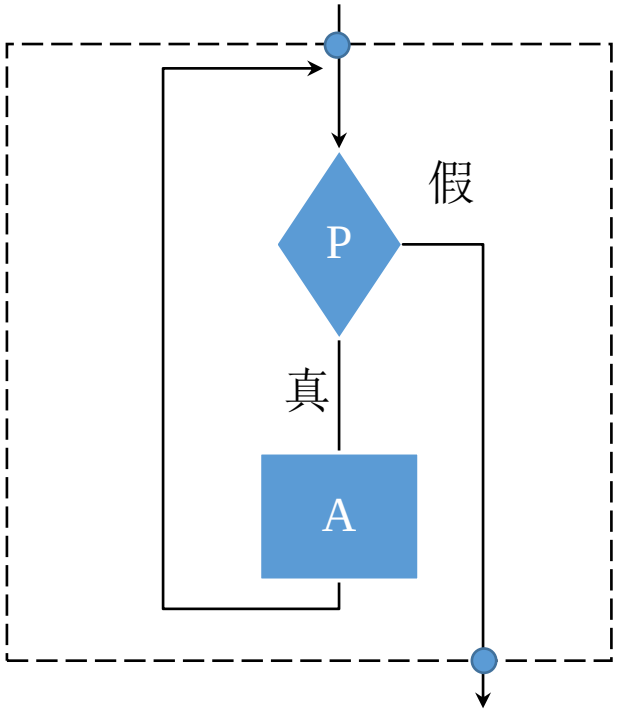


判断选择结构

上图中A、B代表语句(块)

- IF选择(判断）语句（结构)
- 关系运算符和关系表达式
- 逻辑运算符和逻辑表达式
- 条件运算符和条件表达式
- If选择语句嵌套
- Switch多分支选择语句
- 选择结构设计实例

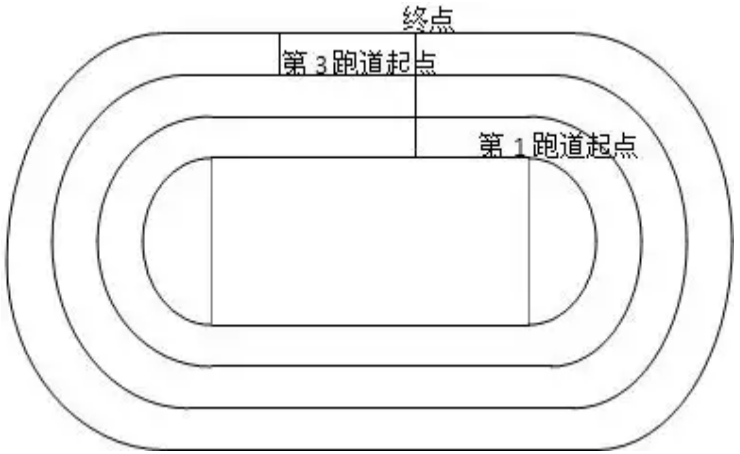
循 环 结 构 程 序 设 计



循环结构

上图中A代表语句(块)

- 循环控制语句（结构）
- While语句
- Do while语句
- For语句
- 循环嵌套
- Break和continue循环跳出语句
- 循环语句编程实例



结构化程序设计(C语言）三种基本结构的关系（独立或嵌套）

It has been demonstrated that any algorithm can be structured, using the three basic control structure, namely, sequence structure, selection structure, and looping structure.

顺序结构、分支结构和循环结构并不彼此孤立的，在循环中可以有分支、顺序结构，分支中也可以有循环、顺序结构，其实不管哪种结构，均可广义的把它们看成一个语句。在实际编程过程中常将这三种结构相互结合以实现各种算法，设计出相应程序。

if语句的一般形式

if (表达式) 语句块1
[else 语句块2]

“表达式”可以是关系表达式、逻辑表达式，甚至是数值表达式

方括号内的部分(即else子句)为可选的，既可以有，也可以没有

语句块1和语句块2可以是一条简单的语句，可以省略花括号，也可以是一个复合语句，复合语句必须加花括号，还可以是另一个if语句

形式1　没有else子句部分

```
if(表达式) {语句块1}
```

形式2　有else子句部分

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

形式3　在else部分又嵌套了多层的if语句

```
if(表达式1)          {语句块1}
else if(表达式2)     {语句块2}
else if(表达式3)     {语句块3}
      ⋮                ⋮
else if(表达式m)     {语句块m}
else                 {语句块m+1}
```

用switch语句实现多分支选择结构

```
switch(表达式)
{
    case  常量1 : {语句块1}
    break;
    case  常量2 : {语句块2}
    ⋮  ⋮  ⋮
    case  常量n : {语句块n}
    break;
    default :{语句块n+1}
    break;
}
```

(1) 括号内的“表达式”，其值的类型应为整数类型(包括字符型)。

(2) 花括号内是一个复合语句，内包含多个以关键字case开头的语句行和最多一个以default开头的行。case后面跟一个常量(或常量表达式)，它们和default都是起标号作用，用来标志一个位置。执行switch语句时，先计算switch后面的“表达式”的值，然后将它与各case标号比较，如果与某一个case标号中的常量相同，流程就转到此case标号后面的语句。如果没有与switch表达式相匹配的case常量，流程转去执行default标号后面的语句。

(3) 可以没有default标号，此时如果没有与switch表达式相匹配的case常量，则不执行任何语句。

(4) 各个case标号出现次序不影响执行结果。

(5) 每一个case常量必须互不相同；否则就会出现互相矛盾的现象。

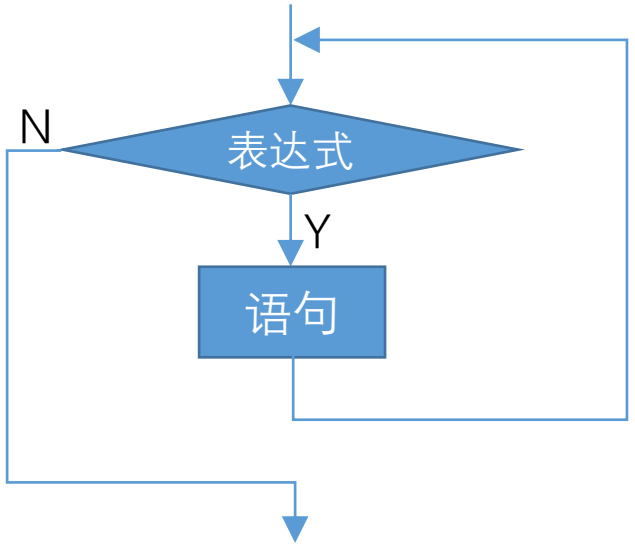
(6) case标号只起标记的作用。在执行switch语句时，根据switch表达式的值找到匹配的入口标号，在执行完一个case标号后面的语句后，就从此标号开始执行下去，不再进行判断。因此，一般情况下，在执行一个case子句后，应当用break语句使流程跳出switch结构。最后一个case子句(今为default子句)中可不加break语句。

(7) 在case子句中虽然包含了一个以上执行语句，但可以不必用花括号括起来，会自动顺序执行本case标号后面所有的语句。当然加上花括号也可以。

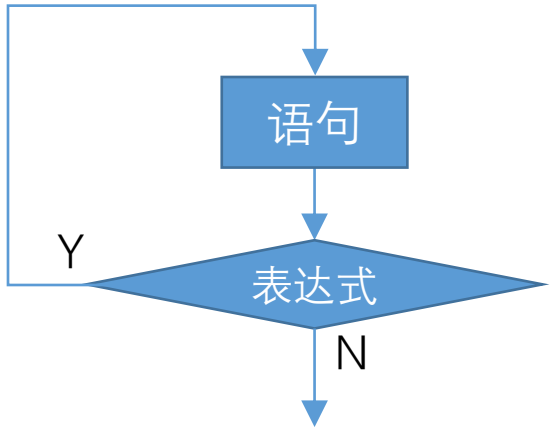
(8) 多个case标号可以共用一组执行语句。

用while、do…while、for语句实现循环

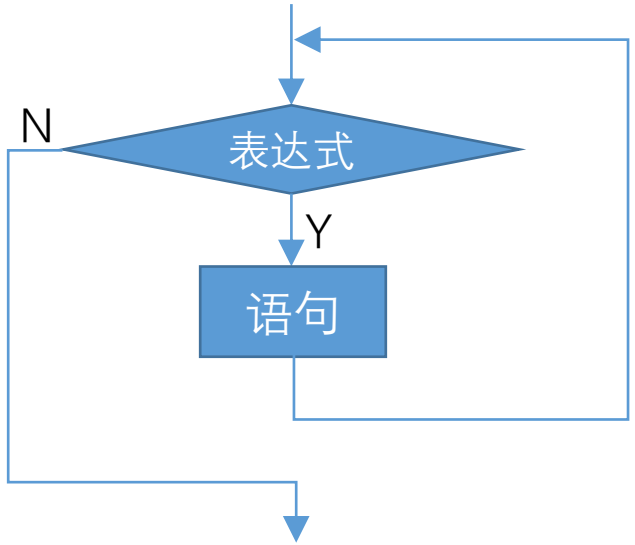
```
while(表达式) 语句
```




```
do
    语句
while(表达式);
```

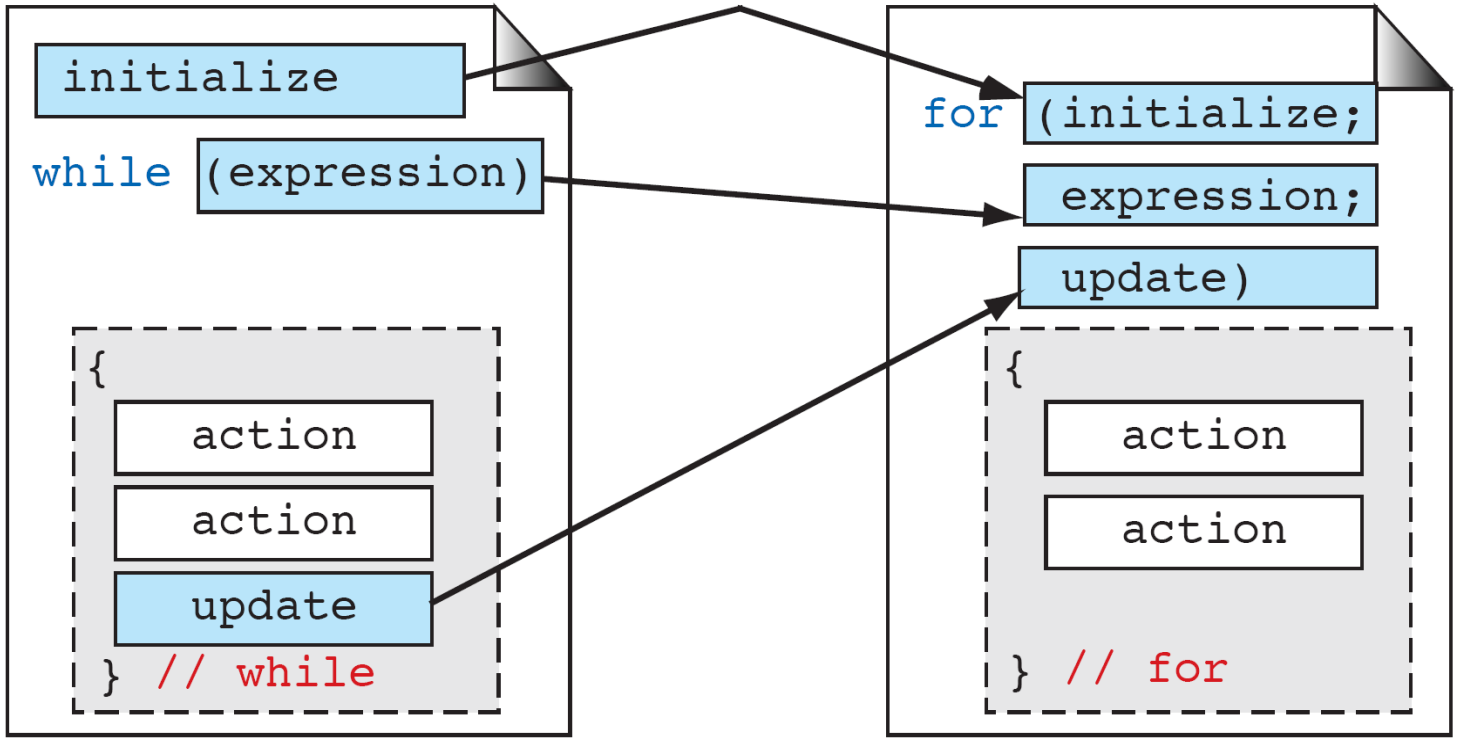



```
for(表达式1；表达式2；表达式3)
    语句
```



Y表示表达式的值为“真”既是非零，N表示表达式的值为“假”既是零。

几种循环的比较



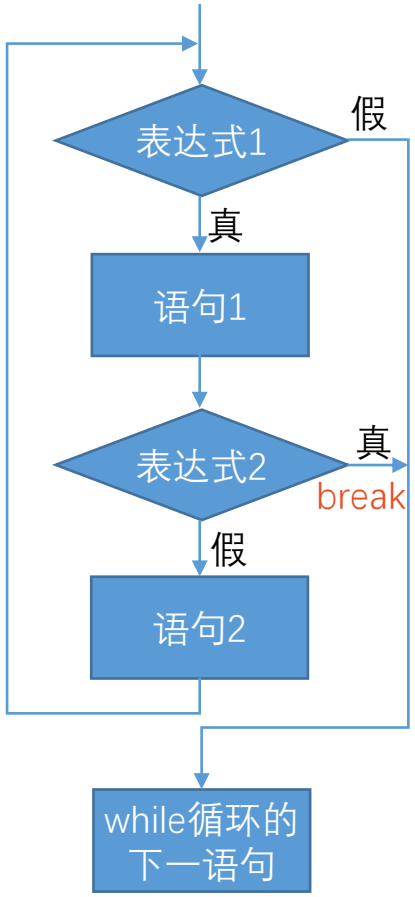
（循环变量初始条件、表达式、循环变量更新）

FIGURE **Comparing *for* and *while* Loops**

break语句和continue语句的区别

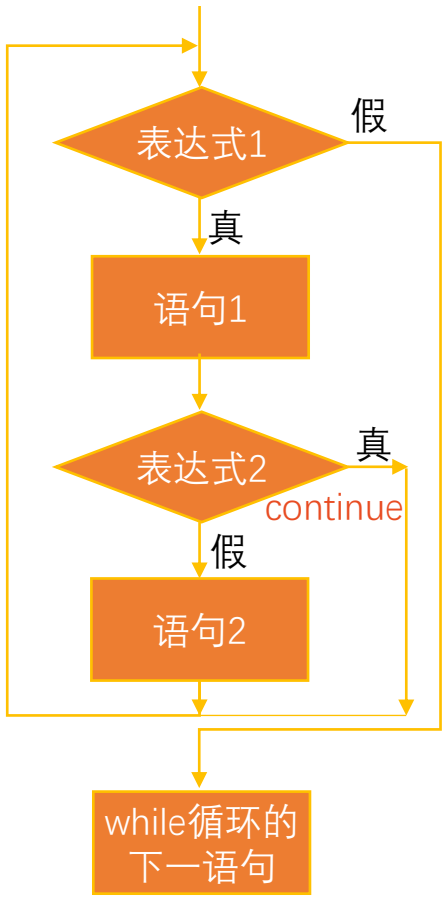
break;

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



continue;

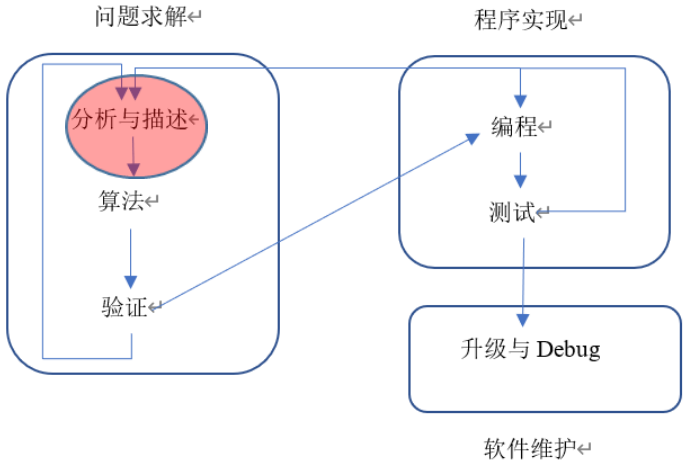
```
while(表达式1)
{
        语句1
        if(表达式2) continue;
        语句2
}
```



continue语句只结束本次循环，而非终止整个循环。break语句结束整个循环，不再判断执行循环的条件是否成立。

C++循环结构程序举例---乘法表

In this C++ program, we are generating the multiplication table of a user-entered number.



2.1 程序设计过程

问题求解：分析与描述What is a multiplication table?

A multiplication table is a table of values used to define a multiplication operation in a mathematical system. Multiplication is defined as one of the basic mathematical operation. For 'a' and 'b' are positive integers, 'a*b' means 'a' is to be added to itself as many times as there are units in 'b'.

For example, 2*3 means 2 is added 3 times itself or 3 is added 2 times itself. 2*3 = 6; 2+2+2 = 6.

How can we generate a multiplication table using C++?

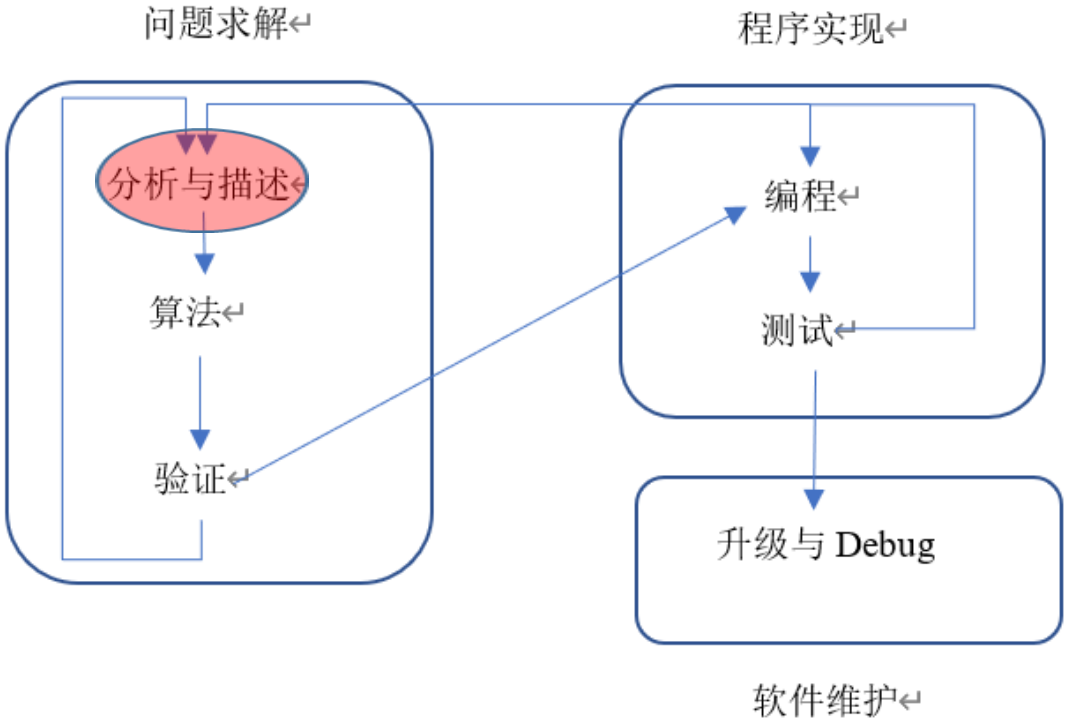
Here we are generating a C++ program to generate the multiplication table of a user-entered positive integer up to 10 only. So, first read a number n from the user and store the value into an integer type variable n. Here we are using a for loop for generating the multiplication table.

Here, we start our multiplication from 1 so initialize i =1; The table computes up to 10 so our condition is i<=10; The value of 'i' is incremented by one in each iteration and updated after each execution of the body of the loop.

C++循环结构程序举例---乘法表

In this C++ program, we are generating the multiplication table of a user-entered number.

问题求解：分析与描述

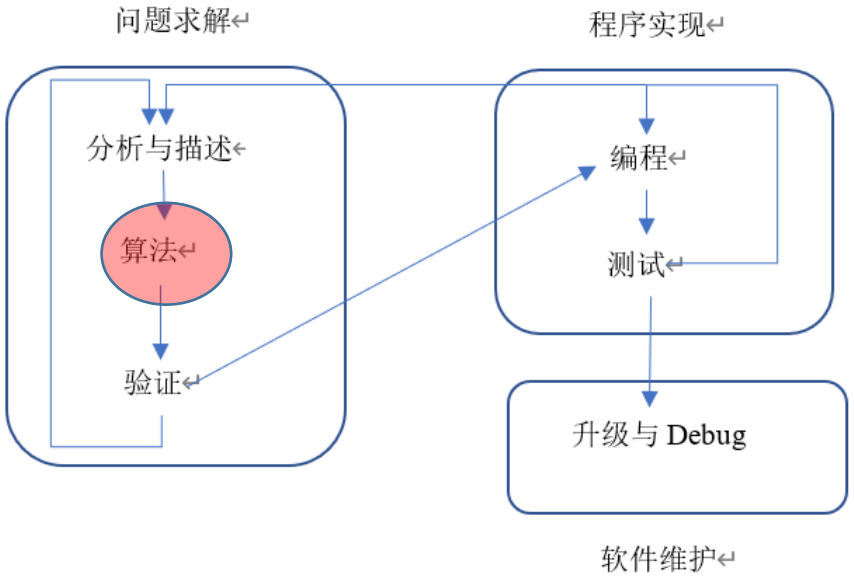


2.1 程序设计过程

举例分析更直观容易：

```
Enter a positive integer: 6
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
```

问题求解：算法



2.1 程序设计过程

选择哪种结构，顺序、选择、循环？

Algorithm

Step 1: Call the header file iostream.

Step 2: Use the namespace std.

Step 3: Open the main()

Step 4: Declare integer variable n;

Step 5: Print a message to enter a positive integer.

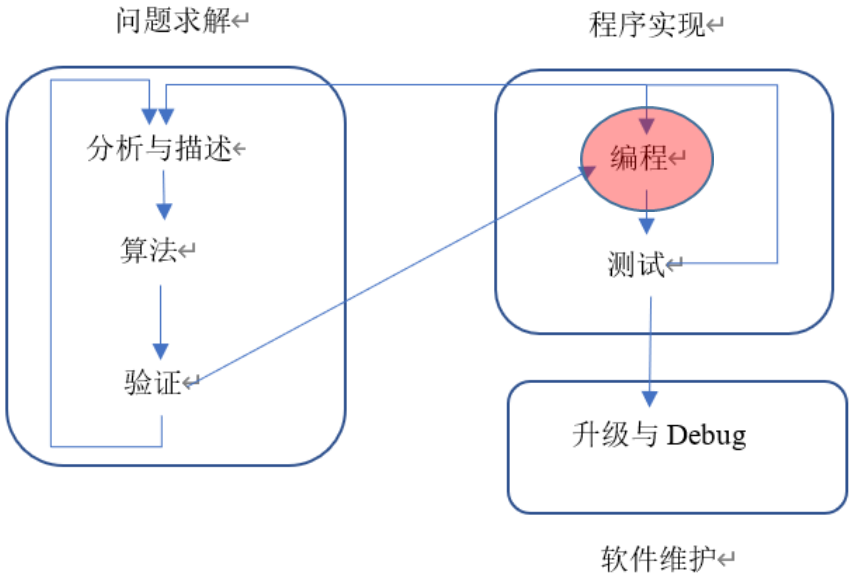
Step 6: Read the number into the variable n.

Step 7: using a for loop find the product of the number n up to 10

Step 8: print the result.

Step 9: Exit.

程序实现：编程



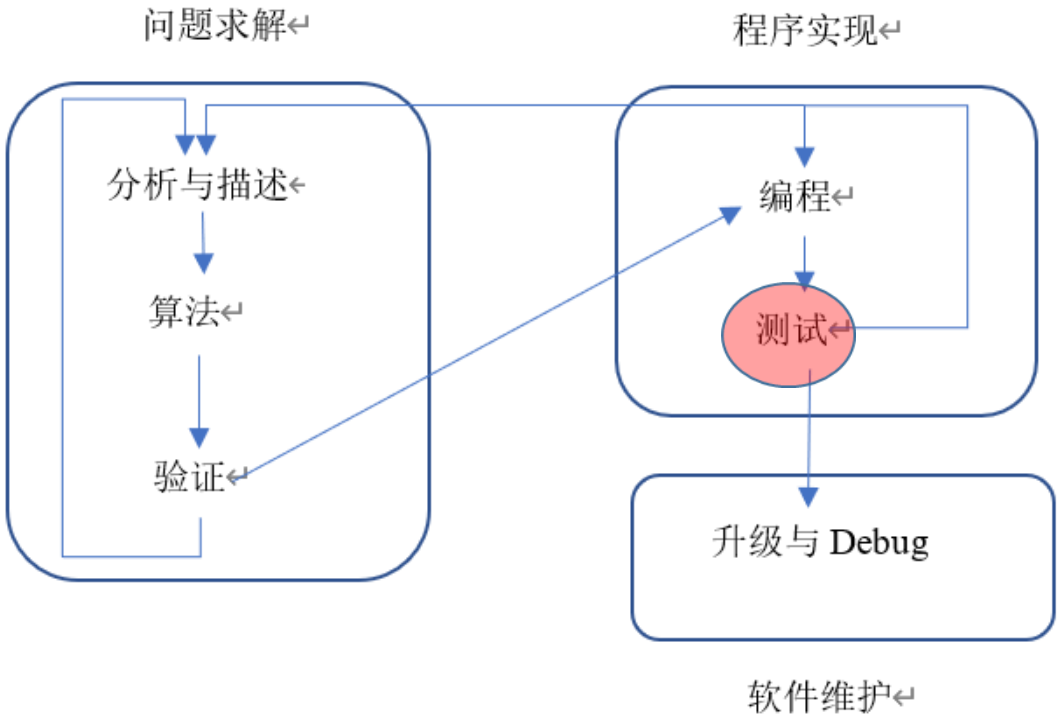
2.1 程序设计过程

编码：C++ Source Code

```
#include <iostream>
using namespace std;

int main()
{
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    for (int i = 1; i <= 10; ++i) {
        cout << n << " * " << i << " = " << n * i << endl;
    }

    return 0;
}
```

语法：for (initialization; condition; increase) statement;

程序实现：测试



2.1 程序设计过程

测试：

```
Enter a positive integer: 6
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
```

本次课实验报告实验内容1：给大家10分钟，课堂完成部分如下

1. 打印如下九九乘法表。 要求至少用两种循环语句实现。改进乘法口诀表的显示模式（完整，左上，左下，右上，右下）。

完整型九九乘法口诀表:

```
1*1= 1  1*2= 2  1*3= 3  1*4= 4  1*5= 5  1*6= 6  1*7= 7  1*8= 8  1*9= 9
2*1= 2  2*2= 4  2*3= 6  2*4= 8  2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1= 3  3*2= 6  3*3= 9  3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*1= 4  4*2= 8  4*3=12  4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*1= 5  5*2=10  5*3=15  5*4=20  5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*1= 6  6*2=12  6*3=18  6*4=24  6*5=30  6*6=36  6*7=42  6*8=48  6*9=54
7*1= 7  7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49  7*8=56  7*9=63
8*1= 8  8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64  8*9=72
9*1= 9  9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

左上三角形九九乘法口诀表:

```
1*1= 1  1*2= 2  1*3= 3  1*4= 4  1*5= 5  1*6= 6  1*7= 7  1*8= 8  1*9= 9
2*2= 4  2*3= 6  2*4= 8  2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*3= 9  3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*6=36  6*7=42  6*8=48  6*9=54
7*7=49  7*8=56  7*9=63
8*8=64  8*9=72
9*9=81
```

EasyX：利用三种控制语句实现运动小球

- 利用 while 循环语句实现小球下落动画
- **利用 if 选择判断语句和while循环语句实现小球重复下落**
- **实现小球落地反弹**

EasyX动画

- 所谓动画，其实是连续显示一系列图形而已。
- 结合到程序上，我们需要以下几个步骤：
- 1.绘制图像　如：　void fillcircle(int x, int y, int radius);//有边框的填充圆
- 2.延时　Sleep(n);
- 3.擦掉图像　　void cleardevice();//这个函数使用当前背景色清空绘图设备。
- 循环以上即可实现动画。

动画和视频的区别只有一个：动画是一种表现形式，视频是一种播放方式。

动画是一种综合艺术，它是集合了绘画、漫画、电影、数字媒体、摄影、音乐、文学等众多艺术门类于一身的艺术表现形式。

视频（Video）泛指将一系列静态影像以电信号的方式加以捕捉、纪录、处理、储存、传送与重现的各种技术。连续的图像变化每秒超过24帧（frame）画面以上时，根据视觉暂留原理，人眼无法辨别单幅的静态画面；看上去是平滑连续的视觉效果，这样连续的画面叫做视频。

利用 while 实现小球下落动画

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main()
{
    int y = 50;
    initgraph(600, 600);
    while (1)
    {
        y = y + 1;
        cleardevice();
        fillcircle(300, y, 20);
        Sleep(10);
    }
    _getch();
    closegraph();
    return 0;
}
```

1.绘制图像

2.延时

3.擦掉图像

利用while语句，我们可以实现小球连续下落的动画效果。

小球的初始y坐标为50，while循环语句中y坐标增加1、清空屏幕、绘制新位置的圆、暂停10毫秒。如此重复执行，即实现了小球连续下落的动画效果。

- 绘制图像fillcircle函数
- 延时Sleep()函数
- 擦掉图像cleardevice

利用 if 选择判断语句实现小球重复下落

增加两行代码，得到小球重复下落的效果。

```
while (1)
{
    y = y + 1;
    if (y>620)
        y = -20;
    cleardevice();
    fillcircle(300, y, 20);
    Sleep(10);
}
```

添加的代码叫if语句，也叫选择判断语句。

if (y>620)表示当y的值大于620时，执行后面的语句y = -20，即将小球的y坐标设为-20。

小球落地反弹

在小球逐渐下落代码的基础上，

设定变量vy记录小球在y轴方向上的速度，

vy初始化为3。

在while语句中，小球的y坐标每次增加vy。

initgraph(640, 480, EW_SHOWCONSOLE);
我们可以用这个函数打开控制台，然后用printf输出函数打印输出需要观察的变量的值，如y和vy，从而更好的理解程序执行过程。

```c
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main()
{
    int y = 50;
    int vy = 3;
    initgraph(600, 600);
    while (1)
    {
        y = y + vy;
        if (y>=620)
            y = -20;
        cleardevice();
        fillcircle(300, y, 20);
        Sleep(10);
    }
    _getch();
    closegraph();
    return 0;
}
```

小球落地反弹

当小球落地时，即小球刚和窗口最底部接触时，

小球中心y坐标恰好等于600 -20 = 580，

即窗口高度减去小球半径。

将小球在y轴上的速度反向（vy = -vy），

执行y = y+ vy就相当于将y逐渐变小，

即实现了小球向上反弹。

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main()
{
    int y = 50;
    int vy = 3;
    initgraph(600, 600);
    while (1)
    {
        y = y + vy;
        if (y>=580)
            vy = -vy;
        cleardevice();
        fillcircle(300, y, 20);
        Sleep(10);
    }
    _getch();
    closegraph();
    return 0;
}
```

小球落地反弹

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main()
{
    float y = 100; // 小球的y坐标
    float vy = 0;  // 小球y方向速度
    float g = 0.5; // 小球加速度，y方向
    initgraph(600, 600); // 初始化游戏窗口画面，宽600，高600
    while(1) // 一直循环运行
    {
        cleardevice(); // 清除掉之前绘制的内容
        vy = vy+g; // 利用加速度g更新vy速度
        y = y+vy; // 利用y方向速度vy更新y坐标
        if (y>=580) // 当碰到地面时
            vy = -0.95*vy; // y方向速度改变方向，并受阻力影响，绝对值变小
        if (y>580) // 防止小球穿过地面
            y = 580;
        fillcircle(300, y, 20); // 在坐标(300,y)处画一个半径为20的圆
        Sleep(10); // 暂停10毫秒
    }
    _getch(); // 等待按键
    closegraph(); // 关闭窗口
    return 0;
}
```

1、在之前代码的基础上

- 将y、vy改成浮点型变量
- 初始化x、y坐标
- 增加浮点型变量g描述重力加速度，设为0.5

2、在while循环语句中

- 根据重力加速度g增加速度vy
- 用vy更新y坐标，碰到地面后vy反向
- 重复运行

3、优化一下

- 根据现实情况设置阻力
- 加入对小球触及地面的判断
- 为代码添加注释

本次课实验报告实验内容如下：

1. 打印如下九九乘法表。 要求至少用两种循环语句实现。改进乘法口诀表的显示模式（完整，左上，左下，右上，右下）。

完整型九九乘法口诀表:
左上三角形九九乘法口诀表:

```
1*1= 1  1*2= 2  1*3= 3  1*4= 4  1*5= 5  1*6= 6  1*7= 7  1*8= 8  1*9= 9
2*2= 4  2*3= 6  2*4= 8  2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*3= 9  3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*6=36  6*7=42  6*8=48  6*9=54
7*7=49  7*8=56  7*9=63
8*8=64  8*9=72
9*9=81
```

2.采用EasyX实现小球自由落体运动，弹跳运动以及模拟多个小球碰撞运动。

祝大家实验顺利！

知学➡好学➡乐学！