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 助教：焦涵、桂海田 博士研究生

PPT大纲

- 上次课实验报告总结
- 本节课的实验介绍

上次课实验报告总结

- 第二部分实验原理写的不够好，少部分同学没有写这个部分，基本概念简要写一下，语法格式，如if语句，for语句语法格式要写
- 调试过程有些同学没有写
- Cout输出格式控制，很少有同学会用
- 网上查找相关代码，评价优缺点不够

PPT大纲

- 上次课实验报告总结
- 本节课的实验介绍

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

实验六、循环语句实验

一、实验目的

1.掌握新的选择语句switch和循环结构

2掌握用 do…while、for 语句实现循环结构；

3理解 for、while 、do…while 三者的区别；

4掌握转向语句 break、continue；

5进一步学习VC++单步调试程序的方法。

6.掌握用循环语句实现延时的原理和方法

二、实验原理（用书上或者自己的话或者网上好的讲解，简要介绍理论课知识）

实验内容1

1．重复输入 n(1≤n≤10)，输出 1！ + 2！ +… +n!的结果。

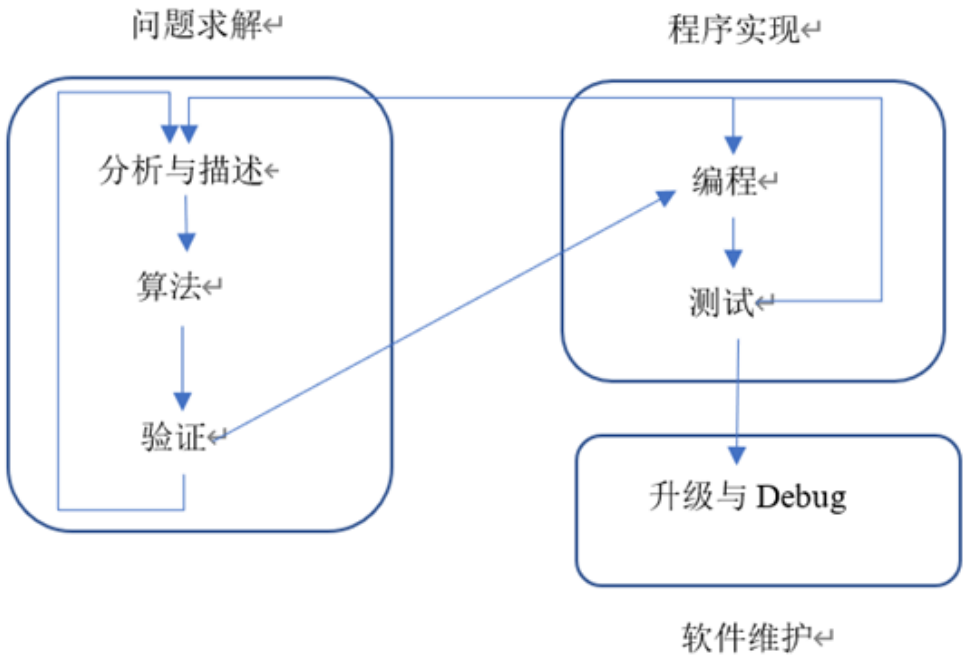
输入 ctrl+z ， 退出程序。

1） 输入输出分析及算法;

2） 编程源代码加注释;

3） 调试与测试;

4） 遇到的问题及解决方法;



2.1 程序设计过程

编程到底学什么

编程学习编程思维（计算机思维即用计算机去解决问题）

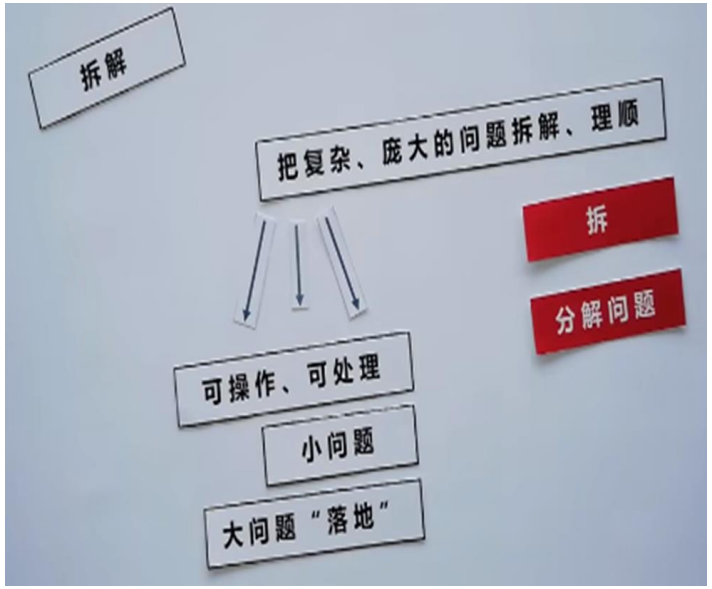
问题求解：分析与描述

程序=数据结构+算法

1.数据结构：输入n(1≤n≤10)，输出1！ + 2！ +… +n!，我们可以知道数据结构选择整数类型就满足要求？

2.算法：

1）求1！ + 2！ +… +n!，我们可以可以分两步骤，把复杂问题分解为多个简单小问题，（1）先求n！（2）在把1！到n!加起来
2）先求n！要用到一层循环（内循环）
3）在把1！到n！加起来，又要用到一层循环（外循环）



求 n！阶乘

分析问题：求 n！阶乘

- i=1　1
- i=2　1x2
- i=3　1x2x3
- i=4　1x2x3x4
- i=5　1x2x3x4x5
- ……

我们可以发现什么规律？

- ✓ 培养计算机思维，用计算机解题（计算）
- ✓ 用计算机解题（计算）和数学解题相似，但有区别
- ✓ 存储程序概念

程序=数据结构+算法

数据结构：整数
算法：循环，乘法运算
1）循环变量
2）连乘法运算
3）循环变量和连乘(阶乘运算）有什么关系？

//源代码：求n！阶乘（是否正确？）

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

int main()
{
    int n, i;
    cout << “Enter a number（1~10）\n";
    cin >> n;

    for (i = 1; i <= n; i++) //求从n！阶乘
    {
        int  f = 1;
         f = f * i;
     }
    cout << “n! is: " << f;
    return 0;
}
```

//源代码：求n！阶乘

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

int main()
{
    int n, i;
    cout << “Enter a number（1~10）\n";
    cin >> n;
    int  f = 1;
    for (i = 1; i <= n; i++)  //求n！阶乘
    {

        f = f * i;
     }
    cout << “n1 is: " << f;
    return 0;
}
```

//源代码：求n！阶乘（网上找的其他人代码）比较优缺点

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

int main() {
    // 输入待计算阶乘的数
    int num;
    cout << "请输入一个数: ";
    cin >> num;

    // 计算阶乘
    int result = 1;
    for (int i = 2; i <= num; i++) {  //从 2 开始循环，因为 0! 和 1! 都为 1。
        result *= i;
    }

    // 输出结果
    cout << num << "! = " << result << endl;

    return 0;
}
```

第一次课强调如何学好C++，多读多写！

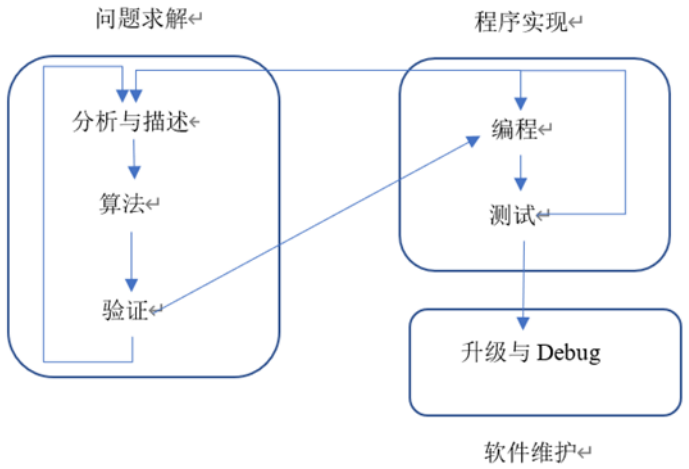
实验报告撰写加分：希望我们做每个程序设计，
都在网上找的其他人代码，
比较优缺点

//源代码：求从1！到n！阶乘的和

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

int main()
{
    int n, sum = 0, i;
    cout << "Enter a number\n";
    cin >> n;

    for (i = 1; i <= n; i++)  //外循环求从1！到n！阶乘的和
    {
        int j, f = 1;
        for (j = 1; j <= i; j++) //内循环求n！累乘累积
            f = f * j;

        sum = sum + f;    //累加：求从1！到n！阶乘的和

    }
    cout << "Sum of the series is: " << sum;
    return 0;
}
```

//源代码：求1! 到n！阶乘和（网上找的其他人代码）比较优缺点

```
#include <iostream>

using namespace std;

int main() {
    // 计算 1 到 10 的阶乘和
    int sum = 0;
    for (int i = 1; i <= 10; i++) {
        int result = 1;
        for (int j = 2; j <= i; j++) {
            result *= j;
        }
        sum += result;
    }

    // 输出结果
    cout << "1 到 10 的阶乘和是: " << sum << endl;

    return 0;
}
```



2.1 程序设计过程

Debug调试

1.从1开始，比较容易口算出程序结果是否正确

2.循环结构程序调试主要看开始第一次循环结果是否正确

3.循环结构程序调试主要看最后一次循环结果是否正确

4.循环结构程序调试主要看中间循环结果随机挑选几个看是否正确

```
Enter a number
1
Sum of the series is: 1
D:\C++1ab\n!\x64\Debug\n!.exe (进程 38908)已退出，代码为 0。
按任意键关闭此窗口. . .
```

Microsoft Visual Studio 调试控制台

```
Enter a number
3
Sum of the series is: 9
D:\C++1ab\n!\x64\Debug\n!.exe (进程 50208)已退出，代码为 0。
按任意键关闭此窗口. . .
```

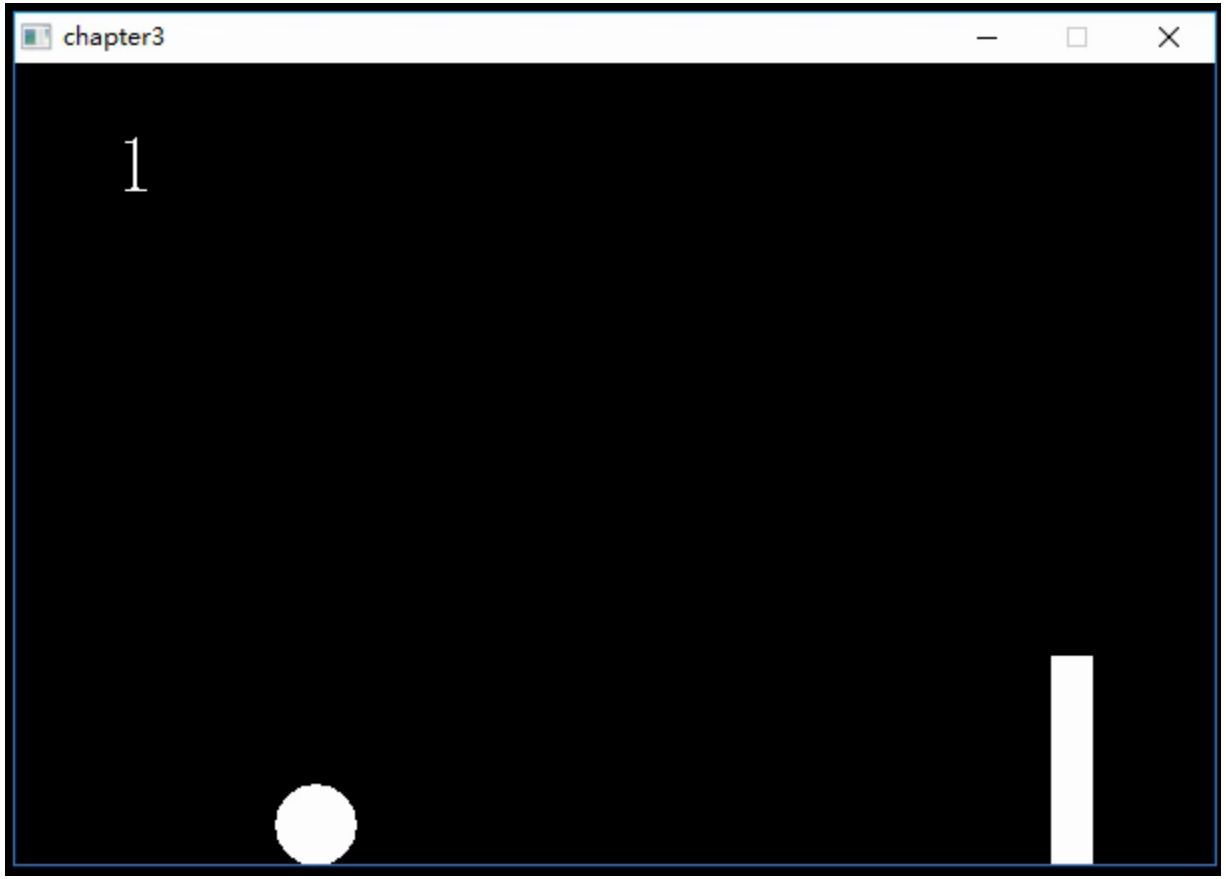
```
Enter a number
10
Sum of the series is: 4037913
D:\C++1ab\n!\x64\Debug\n!.exe (进程 57172)已退出，代码为 0。
按任意键关闭此窗口. . .
```

函数版本源代码

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

// function for finding factorial
int fact(int n)
{
    int i, f = 1;
    for (i = 1; i <= n; i++)
        f = f * i;
    return f;
}
int main()
{
    int n, sum = 0, i;
    cout << "Enter a number\n";
    cin >> n;
    for (i = 1; i <= n; i++)
    {
        sum = sum + fact(i);
    }
    cout << "Sum of the series is: " << sum;
    return 0;
}
```

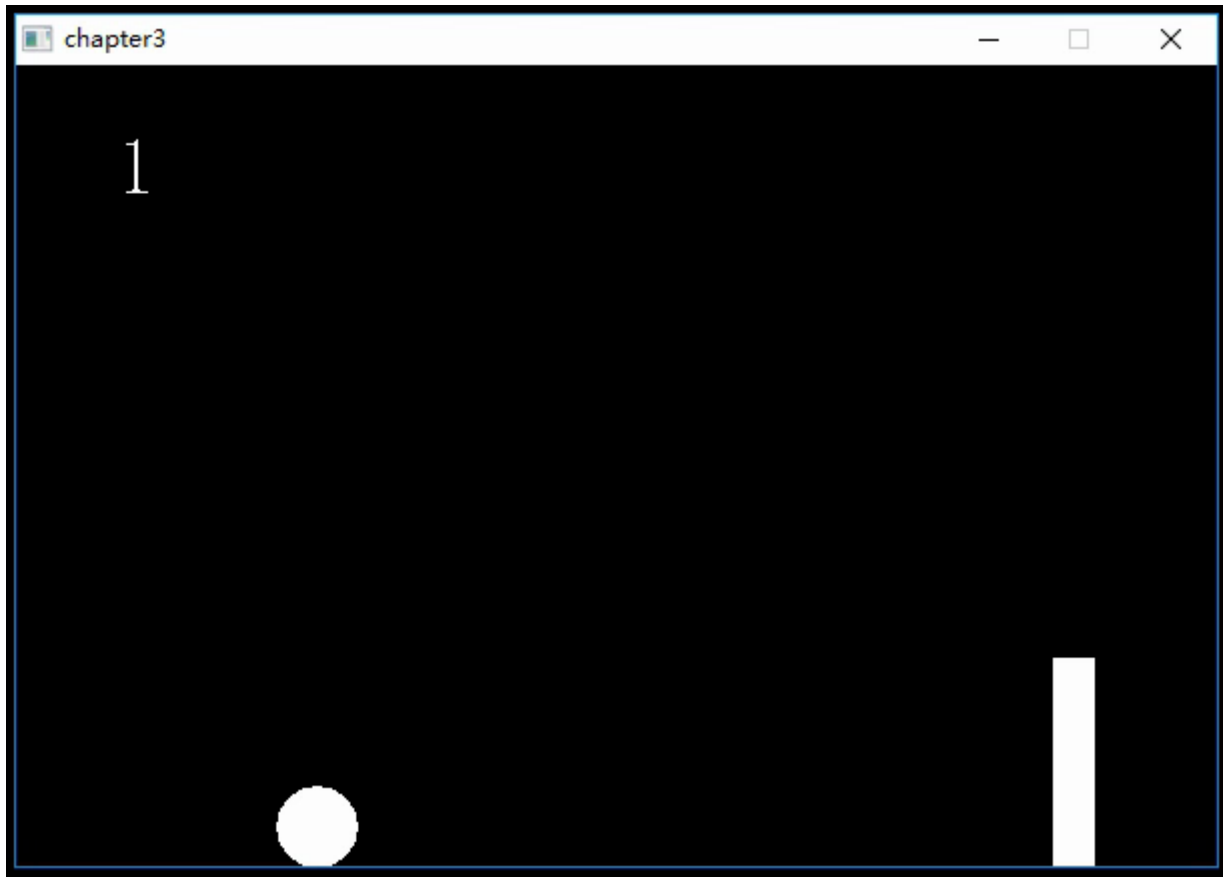
实验内容2：“别碰方块”游戏

- 1 问题分析
- 2 方块的绘制与移动
- 3 按空格键控制小球起跳
- 4 小球和方块的碰撞判断
- 5 随机方块的速度和高度
- 6 得分的计算与显示
- 7 避免空中起跳



问题分析（算法即步骤）

1问题分析：需要哪些变量？用顺序选择循环哪种结构？用到哪些运算？




```
float width,height,gravity; // 游戏画面宽高、重力加速度
float ball_x,ball_y,ball_vy,radius; // 小球圆心坐标、y方向速度、半径

width = 600;  // 游戏画面宽度
height = 400; // 游戏画面高度
gravity = 0.6;  // 重力加速度
initgraph(width, height); // 新建一个画布

radius = 20; // 小球半径
ball_x = width/4; // 小球x位置
ball_y = height-radius;  // 小球y位置
ball_vy = 0;  // 小球初始y速度为0
```

用顺序选择循环结构嵌套，整体是一个大循环结构
用到关系运算、逻辑运算、比较运算、算术运算。。。

2　方块的绘制与移动

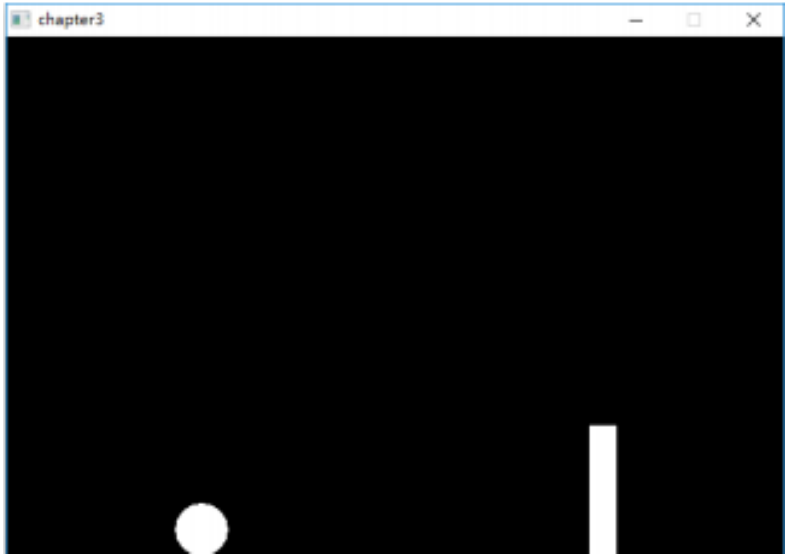
函数fillrectangle(left,top,right,bottom)可以绘制矩形。

其中(left,top)为矩形左上角的(x,y)坐标，(right, bottom)为矩形右下角的(x,y)坐标。

在3-2.cpp的基础上添加右边的代码：

```
8    float rect_left_x,rect_top_y,rect_width,rect_height; // 方块障碍物的相关参数

20   rect_height = 100; // 方块高度
21   rect_width = 20; // 方块宽度
22   rect_left_x = width*3/4; // 方块左边x坐标
23   rect_top_y = height - rect_height; // 方块顶部y坐标

46   // 画方块
47   fillrectangle(rect_left_x, height - rect_height, rect_left_x + rect_
     width,height);
```

可绘制出：



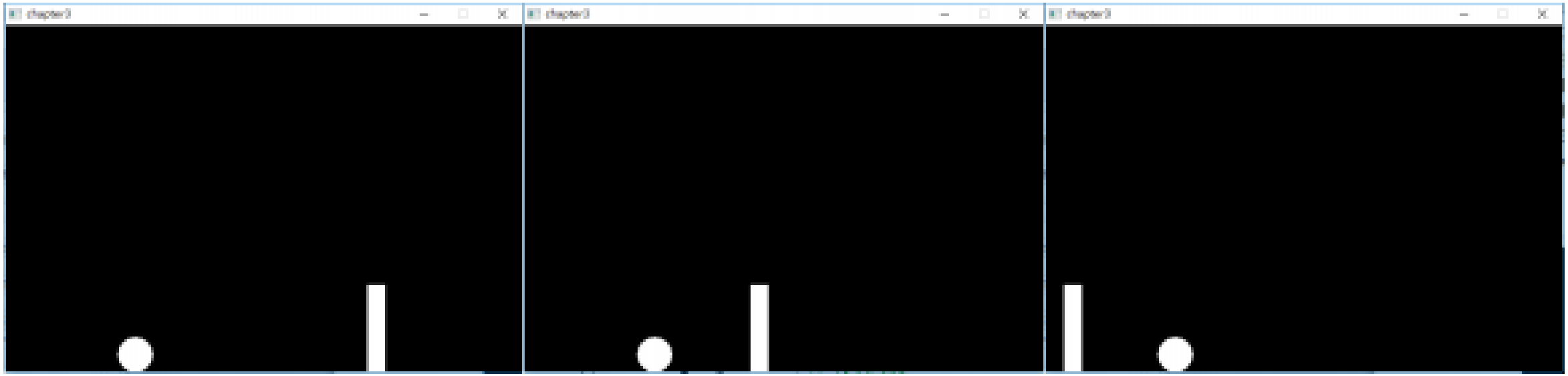
2　方块的绘制与移动

进一步，添加变量rect_vx记录方块在x方向上的速度，并初始化为-3。

在while语句中，让方块从右向左移动。

当方块到达窗口最左边时，再让其从最右边出现。因此需要用循环控制语句。

方块的移动过程下图所示。



3　按空格键控制小球起跳

- 让小球初始在地面上，
- 按下空格键后起跳，
- 落地后小球静止。

```
float width,height,gravity; // 游戏画面宽高、重力加速度
float ball_x,ball_y,ball_vy,radius; // 小球圆心坐标、y方向速度、半径

width = 600;  // 游戏画面宽度
height = 400; // 游戏画面高度
gravity = 0.6;  // 重力加速度
initgraph(width, height); // 新建一个画布

radius = 20; // 小球半径
ball_x = width/4; // 小球x位置
ball_y = height-radius;  // 小球y位置
ball_vy = 0;  // 小球初始y速度为0
```

```
while(1) // 一直循环
{
    if (kbhit()) // 当按键时
    {
        char input = _getch(); // 获得输入字符
        if (input==' ') // 当按下空格键时
        {
            ball_vy = -16; // 给小球一个向上的初速度
        }
    }

    ball_vy = ball_vy + gravity;   // 根据重力加速度更新小球y方向速度
    ball_y = ball_y + ball_vy;   // 根据小球y方向速度更新其y坐标
    if (ball_y >= height-radius) // 如果小球落到地面上
    {
        ball_vy = 0;  // y速度为0
        ball_y = height-radius; // 规范其y坐标，避免落到地面下
    }

    cleardevice(); // 清空画面
    fillcircle(ball_x, ball_y, radius); // 绘制小球
    Sleep(10); // 暂停10毫秒

}
closegraph();
return 0;
```

4　小球和方块的碰撞判断（判断选择语句if）

小球和方块发生碰撞有3种边界情况，如下图所示。

（a）rect_left_x <= ball_x + radius（方块最左边在小球最右边的左侧或二者x坐标相同）。关系运算

（b）rect_left_x + rect_width >= ball_x - radius（方块最右边在小球最左边的右侧或二者x坐标相同）。

（c）height - rect_height <= ball_y + radius（方块最上边在小球最下边的上侧或二者y坐标相同）

C语言提供了3种逻辑运算符：!（非）、&&（与）、||（或），用于实现多个逻辑条件的组合。

4　小球和方块的碰撞判断（判断选择语句if）

利用关系运算符，逻辑与运算符，在3-3-2.cpp基础上添加如下代码，即可实现小球和方块的碰撞判断。当发生碰撞时，利用Sleep( )函数实现类似慢动作的效果：

```
50    // 如果小球碰到方块
51    if ((rect_left_x <= ball_x + radius)
52        && (rect_left_x + rect_width >= ball_x - radius)
53        && (height - rect_height <= ball_y + radius) )
54    {
55        Sleep(100); // 慢动作效果
56    }
```

5　随机方块的速度和高度

rand()函数可以生成随机整数，为了得到设定范围内的随机数，需要利用浮点数除法、整数除法、取余运算符。输入并运行代码：

- 当除号“/”两边有一个数字或变量为浮点数时，

实行浮点数除法，即5.0/2=2.5。

- 当除号“/”两边数字或变量都为整数时，

实行整数除法，得到两个整数相除的商，即5/2=2。

- “10%3”中的百分号“%”为取余运算符，

得到两个整数相除的余数，即10%3=1。

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main()
{
    float a = 5.0/2;
    printf("%f\n",a);
    int b = 5/2;
    printf("%d\n",b);
    int c = 10%3;
    printf("%d\n",c);
    _getch();
    return 0;
}
```

5　随机方块的速度和高度

类型转换

float()、int()形式称为强制类型转换。

float(5)把5转换为浮点型，进行浮点数除法后，float(5)/2等于2.5;

int(6.3)把6.3转换为整数6，进行整数除法后，int(6.3)/2等于3;

int(9.0/2)把浮点数除法结果4.5转换为整数，结果为4。

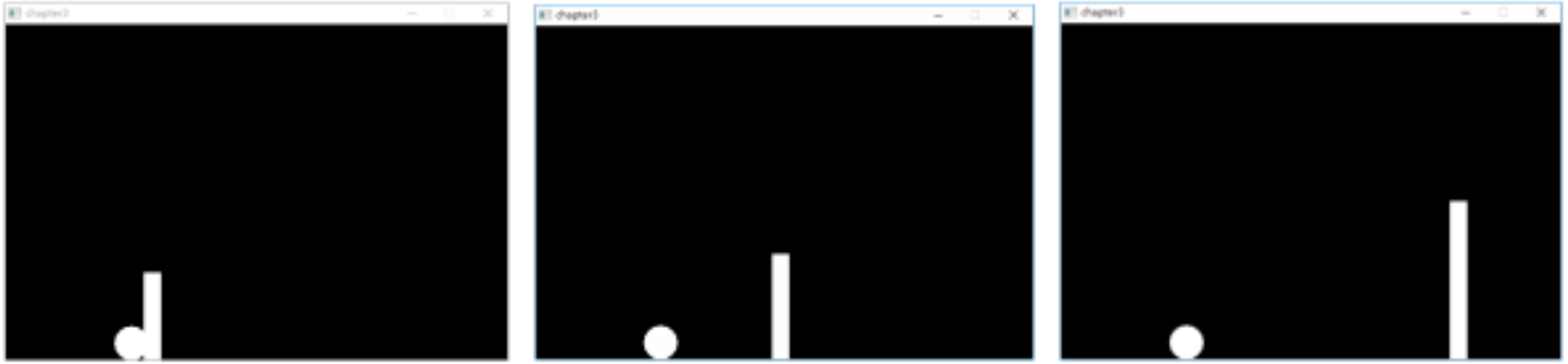
RAND_MAX存储了rand( )函数所能生成的最大整数，rand( )/float(RAND_MAX)即可生成0 ～ 1的随机小数。

5　随机方块的速度和高度

当方块重新出现时，添加代码设置其随机高度范围为height/4到height/2，随机速度为-3到7，添加代码如下：

```
46      if (rect_left_x <= 0) // 如果方块跑到最左边
47      {
48          rect_left_x = width; // 在最右边重新出现
49          rect_height = rand() % int(height/4) + height/4; // 设置随机高度
50          rect_vx = rand()/float(RAND_MAX) *4 - 7; // 设置方块随机速度
51      }
```

程序运行后输出如图：



6　得分的计算与显示

定义整型变量记录游戏的得分，并初始化为0：

```
int score = 0; // 得分
```

当方块跑到画面最左边时，得分增加1：

```
if (rect_left_x <= 0) // 如果方块跑到最左边
{
    score = score + 1; // 得分+1
}
```

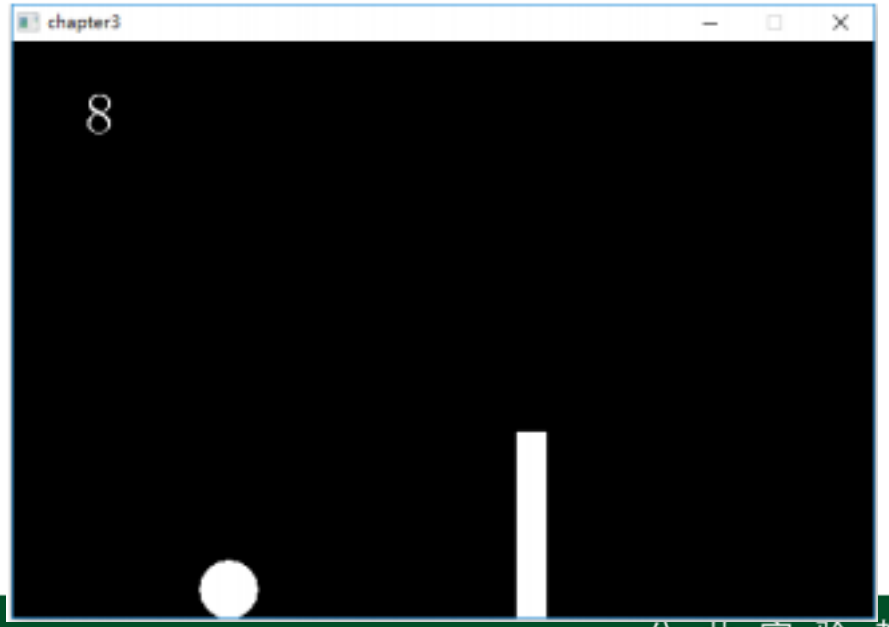
当方块碰到小球时，得分清零：

```
// 如果小球碰到方块
if ((rect_left_x <= ball_x + radius)
    && (rect_left_x + rect_width >= ball_x - radius)
    && (height - rect_height <= ball_y + radius) )
{
    score = 0; // 得分清零
}
```

另外，利用EasyX的文字输出功能，可输出score：

```
TCHAR s[20]; // 定义字符串数组
_stprintf(s, _T("%d"), score); // 将score转换为字符串
settextstyle(40, 0, _T("宋体")); // 设置文字大小、字体
outtextxy(50, 30, s); // 输出得分文字
```

实现效果如下图所示：



7　避免空中起跳

首先，定义变量isBallOnFloor记录小球是否在地面上，并初始化为1，表示开始小球在地面上：

```
int isBallOnFloor = 1; // 小球是否在地面上，避免重复起跳
```

当用户按下空格键时，必须同时满足isBallOnFloor为1，才让小球起跳。起跳后，设定isBallOnFloor＝0，表示目前小球不在地面上了：

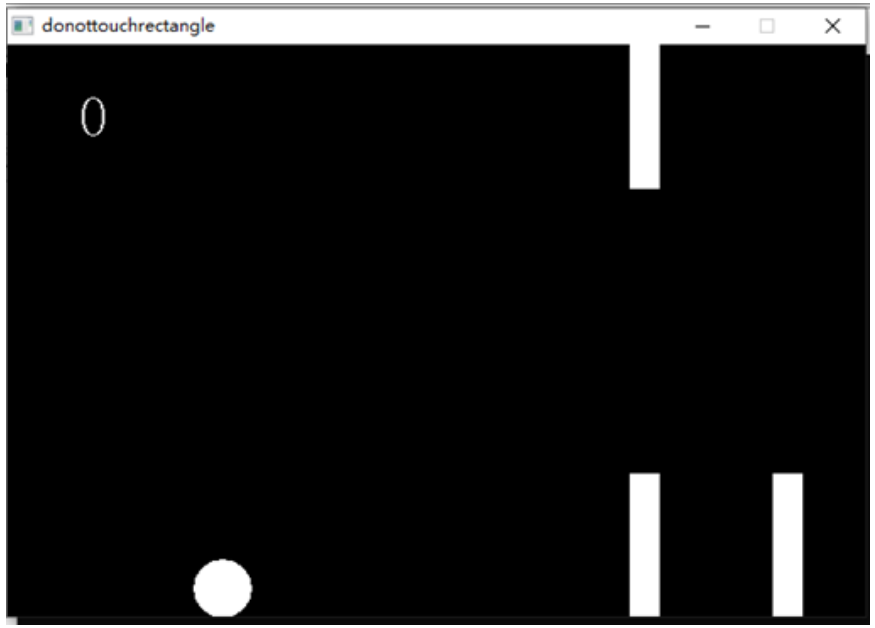
```
if (input==' ' && isBallOnFloor==1) // 当按下空格键时，并且小球在地面上时
{
    ball_vy = -17; // 给小球一个向上的速度
    isBallOnFloor = 0; // 表示小球不在地面了，不能重复起跳
}
```

7　避免空中起跳

当小球再次落到地面上后，设定 isBallOnFloor = 1，表示目前小球又重新到了地面上，可以起跳了：

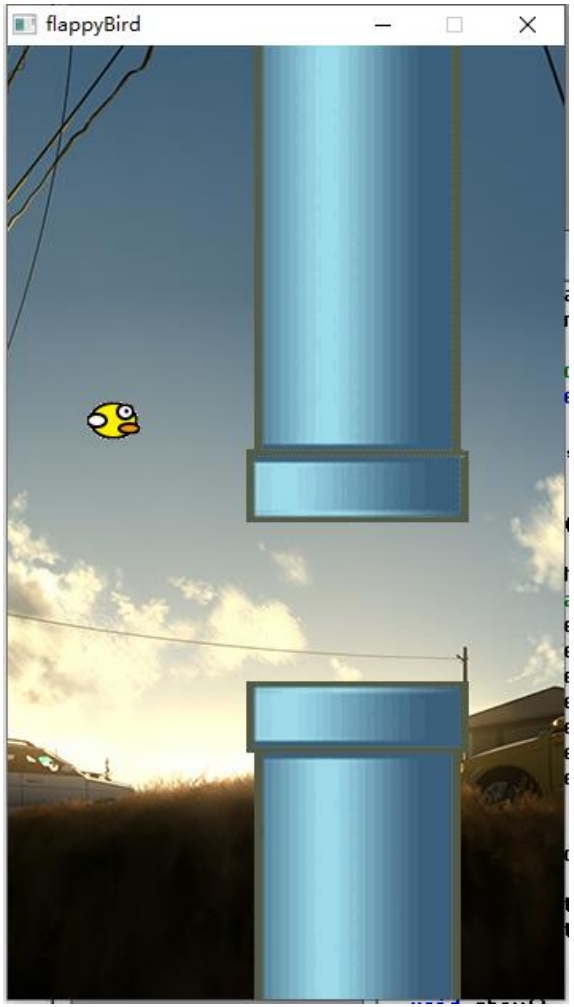
```
if (ball_y >= height-radius)  // 如果小球落到地面上
{
    ball_vy = 0;  // y速度为0
    ball_y = height-radius;  // 规范其y坐标，避免落到地面下
    isBallOnFloor = 1; // 表示小球在地面上
}
```

程序的改进（创新技能培养）



复现别人的程序之后，要进行思考程序优缺点，
尝试着进行改进，这样才能真正学会学好

flappyBird



Any question?

学习的三重境界

知学 ➡ 好学 ➡ 乐学

祝大家实验顺利！

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。

请同学们思考下面程序有何缺陷？

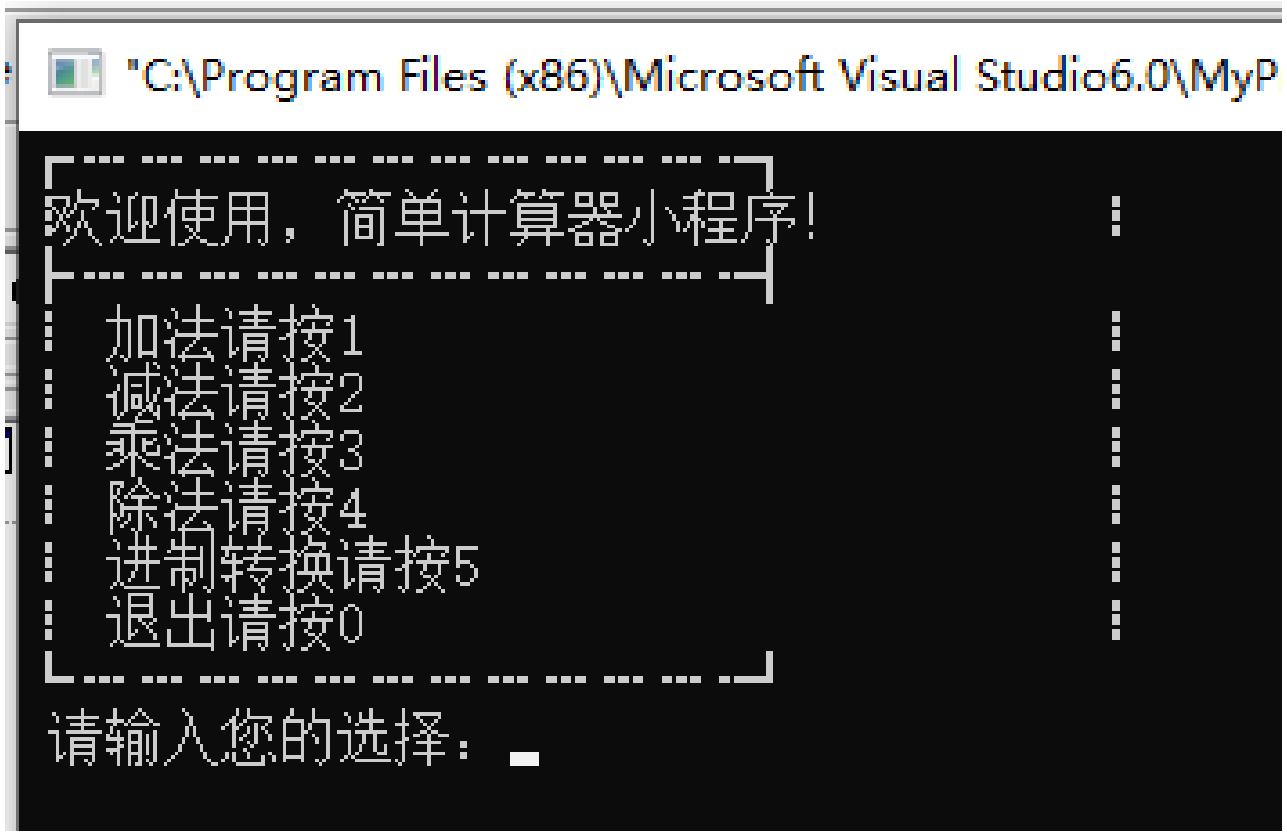
```
//Write a program which accept two numbers and print their sum.
#include<iostream>
using namespace std;

int main()
{
        int a,b,sum;//变量定义
        cout<< "\nEnter first number : ";
        cin>>a;
        cout<<"\nEnter second number : ";
        cin>>b;

         sum=a+b;

         cout<<"\nThe Sum is : "<<sum;

        return 0;
}
```

- 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（对象）
- 顺序**选择循环结构程序**

实验内容3：简单计算器设计

- 设计一个简单计算器，实现两个数加减乘除取余运算。
- 给出菜单选择：

1.加法
2.减法
3.乘法
4.除法
5.取余
0.退出。



简单计算器框图

- 分析问题：设计一个简单计算器，实现加减乘除取余运算。
- 前面我们实现的简单加法乘法运算器程序功能很有限，每运行一次只能实现一种运算，怎么改进？
- 用选择结构（算法），循环结构（算法）

分析问题

1.变量定义：首先我们定义了两个double型的变量x,y作为计算的两个值，然后我们定义了两个int型的变量chose(接收用户输入选择的数字)和isFlag(判断是否要退出计算循环)

2.程序整体使用do--while循环结构，将各种计算提示菜单，及计算功能放进去，使用户可以反复使用计算器

3.在do--while循环里我们整体使用顺序结构，先打印用户可见的菜单选择提示包含加法计算器，减法计算器，乘法计算器，除法计算器，，计算器(程序)的退出功能按键菜单

简单计算器菜单menu：

```
printf("*****************计算器*****************\n");
printf("***************1.加法计算***************\n");
printf("***************2.减法计算***************\n");
printf("***************3.乘法计算***************\n");
printf("***************4.除法计算***************\n");
printf("***************5.退出*******************\n");
printf("*************请选择  (1~~5)*************\n");
```

4.在do--while循环内部嵌套使用if--else语句，对用户从键盘输入的值进行操作，首先使用if--else语句判断用户从键盘输入的数字是否合理，若不合理，则提示用户重新输入数字，若输入的数字合理则进入switch--case语句进行对于输入数字相应的操作，大概框架如下

```
if(chose > 5 || chose < 1)
{
        printf("非法输入、请重新输入(1~~5)\n");
}
else
{
switch(chose)
{
                                case 1:
                                case 2:
                                case 3:
                                case 4:
                                case 5:
}
 }
```

5.设置do--while循环的循环条件为(isFlag),并对每个case中的内容功能进行添加

这里我们case 1：为例，分别使用printf和scanf提示并获取用户输入需要进行计算的两个值x，y　并将两个数相加的值输出出来，并使用break退出switch--case语句

```
case 1:
        printf("请输入x:");
        scanf("%lf",&x);
        printf("请输入y:");
        scanf("%lf",&y);
        printf("x + y = %f\n",x+y);
        break;
```

6.对用户退出计算器的功能进行添加，将退出功能的case 5：进行添加，若进入case 5:则将0赋值给isFlag，再使用break退出switch--case语句，并设置do--while循环的循环条件为isFlag。所以若程序进入了case 5：，isFlag的值将从1变成0。即可退出循环，即完成程序退出功能的实现。

```
case 5:
        isFlag = 0;
        break;
```

简单计算器算法

1)用循环语句do while进行重复计算

2)循环体内部先显示菜单，提示用户进行各种输入选择

3)循环体内部先用if语句判断用户输入是否合法，合法就进行选择的运算

4)循环体内部菜单选择用switch选择语句（多种选择）

5)Switch选择语句内部加减乘除每一项计算功能的实现

注意上述循环结构嵌套选择结构，选择结构嵌套顺序结构

中山大學 SUN YAT-SEN UNIVERSITY

源代码：

```c
#include<stdio.h>
int main()
{
    double x,y;
    int chose,isFlag=1;
    do
    {
        printf("****************计算器****************\n");
        printf("**************1.加法计算***************\n");
        printf("**************2.减法计算***************\n");
        printf("**************3.乘法计算***************\n");
        printf("**************4.除法计算***************\n");
        printf("***************5.退出*******************\n");
        printf("*************请选择(1~~5)**************\n");
        scanf("%d",&chose);
        if(chose > 5 || chose < 1)
        {
            printf("非法输入、请重新输入(1~~5)\n");
        }
        else
        {

            switch(chose)
            {
                case 1:
                    printf("请输入x:");
                    scanf("%lf",&x);
                    printf("请输入y:");
                    scanf("%lf",&y);
                    printf("x + y = %f\n",x+y);
                    break;
                case 2:
                    printf("请输入x:");
                    scanf("%lf",&x);
                    printf("请输入y:");
                    scanf("%lf",&y);
                    printf("x - y = %f\n",x-y);
                    break;
                case 3:
                    printf("请输入x:");
                    scanf("%lf",&x);
                    printf("请输入y:");
                    scanf("%lf",&y);
                    printf("x * y = %f\n",x*y);
                    break;
                case 4:
                    printf("请输入x:");
                    scanf("%lf",&x);
                    printf("请输入y:");
                    scanf("%lf",&y);
                    printf("x / y = %f\n",x/y);
                    break;
                case 5:
                    isFlag = 0;
                    break;
            }
        }
    }while(isFlag);
        printf("程序已退出哦~~~");
        return 0;
}
```