实验指导

实验 1

C# 开发环境及程序设计基础

实 验 目 的

- 掌握使用命令行开发简单的 C＃应用程序；
- 掌握使用 Visual Studio 编写控制台应用程序；
- 掌握 Visual Studio 环境下程序的跟踪调试；
- 了解 Visual Studio 在线帮助的使用；
- 掌握应用程序命令行参数的使用。

实 验 内 容

实验 1-1　使用命令行开发简单的 C＃应用程序

实验要求：使用记事本编写“Hello World!” C＃应用程序，使用 C＃命令行编译程序 csc.exe 对应用程序进行编译并运行测试。

操作步骤：

(1) 启动记事本，输入如下代码，并保存为 C:\C＃LAB\LAB01\Hello.cs。

```
//LAB01/Hello.cs //A "Hello World!" program
//compile:csc Hello.cs -> Hello.exe
using System;
namespace CSharpBook.LAB01
{
    class HelloWorld
    {
        static void Main()
        {
            System.Console.WriteLine("Hello World!");
        }
    }
}
```

(2) 启动 Visual Studio 2022 命令提示并进入 C:\C＃LAB\LAB01 文件夹。执行“开始”→“所有应用”→Visual Studio 2022→Developer Command Prompt for VS 2022 命令，进入“VS 2022 开发者命令提示”命令行窗口，并输入命令 cd \C＃LAB\LAB01，然后按 Enter 键，将当前目录切换到实验 1 的实验目录，如图 1-1 所示。

(3) 编译 Hello.cs 程序。在 C:\C＃LAB\LAB01 命令提示状态下输入命令 csc Hello.cs，

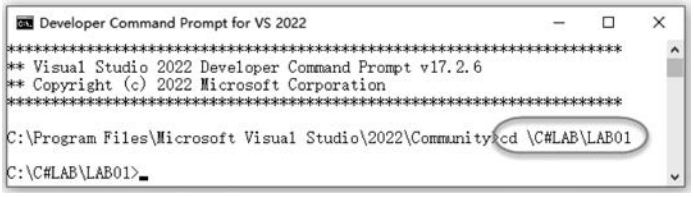


图 1-1　切换到实验 1 的实验目录

并按 Enter 键,如图 1-2 所示,编译 Hello.cs 程序。

图 1-2　编译、运行 Hello.cs 程序

(4) 运行 Hello.exe 程序。在 C:\C#LAB\LAB01 命令提示状态下输入命令 Hello,并按 Enter 键,见图 1-2,运行 Hello.exe 程序,观测运行效果。

实验 1-2　Visual Studio 的基本使用

实验要求:使用 Visual Studio 编写控制台应用程序 HelloConsole。根据命令行参数(如 zhangsan)输出“您好! zhangsan”。

操作步骤:

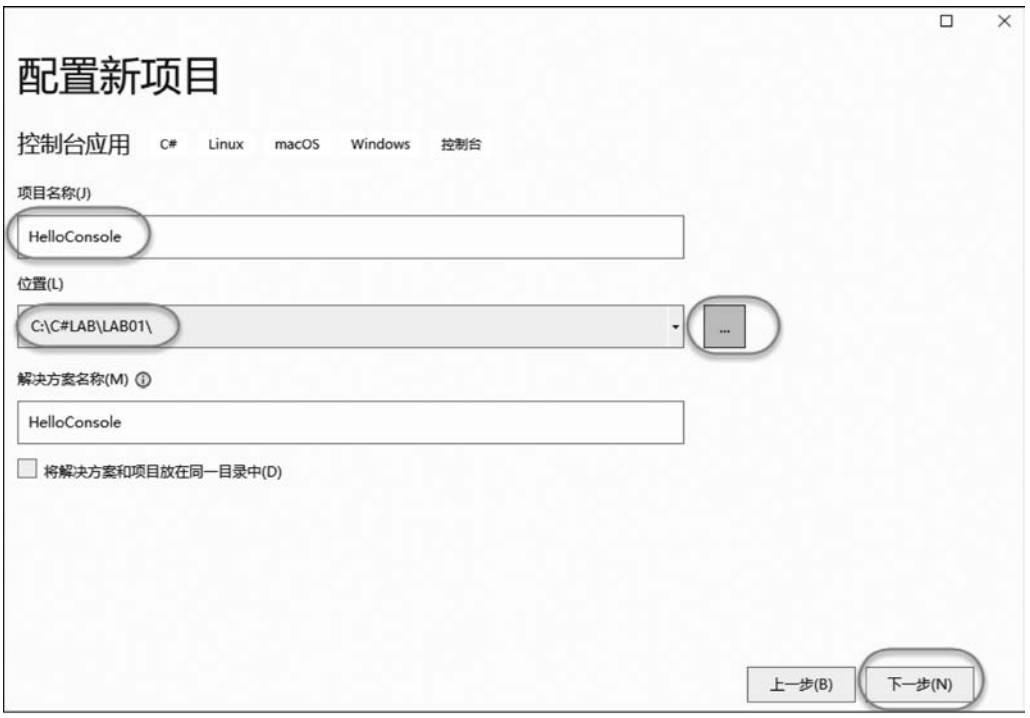
(1) 打开 Visual Studio 2022。

(2) 创建控制台应用程序 HelloConsole。通过选择“文件”→“新建”→“项目”命令,打开“创建新项目”对话框,在“语言”下拉列表框中选择 C#,然后选择“控制台应用”模板,单击“下一步”按钮,如图 1-3(a)所示。在随后出现的“配置新项目”对话框中,在“项目名称”文本框中输入控制台应用程序名称 HelloConsole,利用“浏览”按钮选择文件夹 C:\C#LAB\LAB01,单击“下一步”按钮,如图 1-3(b)所示。在随后出现的“其他信息”对话框中,勾选“不使用顶级语句”复选框,如图 1-3(c)所示,单击“创建”按钮,创建控制台应用程序解决方案和项目。

(3) 输入处理程序。在 Program.cs 的 Main()方法中添加如下粗体代码:

```
static void Main(string[] args)
{
    if (args.Length == 0)
    {
        Console.WriteLine("请输入您的姓名作为参数!");
    }
    else
    {
        Console.WriteLine("您好! " + args[0]);
    }
    Console.ReadKey();
}
```

(a)“创建新项目”对话框



(b)“配置新项目”对话框

图 1-3　创建新项目(控制台应用)

(c)“其他信息”对话框

图 1-3 (续)

(4) 编译运行。单击工具栏上的“启动”调试按钮,或者按 F5 键运行并测试该应用程序。运行效果如图 1-4 所示,提示用户输入姓名作为参数。

图 1-4 提示用户输入姓名作为参数

(5) 在 Visual Studio 环境输入命令行参数并调试运行。右击“解决方案资源管理器”中的项目 HelloConsole,从弹出的快捷菜单中选择“属性”命令,在弹出的对话框中选择“调试”选项,单击“打开调试启动配置文件 UI”链接,在随后出现的“启动配置文件”中的“命令行参数”列表框中输入 zhangsan,如图 1-5 所示。单击该对话框右上角的“关闭”按钮,关闭属性设置。按 F5 键再次运行并测试应用程序,运行效果如图 1-6 所示。

图 1-5 输入命令行参数

```
您好！ zhangsan
```

图 1-6　提供命令行参数的运行效果

实验 1-3　Visual Studio 环境下程序的跟踪调试

实验要求：熟悉 Visual Studio 环境下程序的跟踪调试功能，包括设置断点并进行跟踪调试等。

操作步骤：

(1) 打开 Visual Studio 2022。

(2) 在 C:\C#LAB\LAB01 中创建控制台应用程序 TraceDebug。

(3) 输入处理程序。在 Program.cs 的 Main()方法中添加如下粗体代码(注意：每条语句末故意未输入“;”，制造 5 个编译错误)：

```
static void Main(string[] args)
{
    int a = 20
    int b = 5
    int c = 100 / a + b
    Console.WriteLine(c)
    Console.ReadKey()
}
```

(4) 编译程序。执行“生成”→“生成解决方案”命令编译程序，“错误列表”窗口中将列出如图 1-7 所示的 5 个错误信息，表明源代码中第 13～17 行缺少“;”。

图 1-7　错误列表信息(1)

(5) 修正编译错误。依次双击“错误列表”窗口中所列出的错误信息，光标将自动定位到出错的行列位置，依次输入“;”，改正 5 个编译错误。

(6) 重新编译程序。执行“生成”→“生成解决方案”命令编译程序，如果程序完全正确，Visual Studio 窗口底部的状态栏左侧将显示“生成成功”的提示信息。

(7) 修正第二类编译错误。故意删除 Program.cs 第一行的“using System;”语句，执行“生成”→“生成解决方案”命令编译程序。双击图 1-8 所示“错误列表”窗口中第一行的错误信息，光标将自动定位到出错的行列位置，右击 Console，从弹出的快捷菜单中选择“快速操作和重构”→“using System;”语句，如图 1-9 所示，系统将自动在 Program.cs 中添加导入名称空间的语句“using System;”，改正第二类编译错误。

图 1-8　错误列表信息(2)

图 1-9　解析错误

(8) 编译运行。按 F5 键运行测试该应用程序,运行结果为 10。

(9) 运行时错误。故意将 Program.cs 的 Main()方法中的"int a=20;"改为"int a=0;",按 F5 键运行测试该应用程序。编译通过,但是将出现如图 1-10 所示的运行时错误:"尝试除以零"。关闭当前运行窗口(也可以单击工具栏上的"停止调试"按钮■,否则运行调试状态将无法修改源代码),再将"int a = 0;"改回"int a = 20;"。按 F5 键重新运行测试该应用程序,再次得到正确的运行结果:10。

图 1-10　运行时错误

(10) 设置断点。假设本程序其实是计算"100/(a+b)"而不是程序设计时的"100/a+b",这将是一个逻辑错误。可以借助设置断点查看变量的中间结果加以调试解决。在"int a=20;"所在行的左侧断点区域单击鼠标,在设置断点的位置会出现一个红点,如图 1-11 所示。

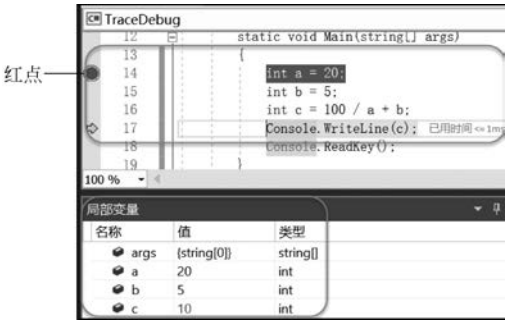


图 1-11　单步调试

(11) 启动调试。按 F5 键启动调试,调试器将在断点处停止并获得当前窗口焦点。

(12) 单步执行。从断点处可以通过调试工具栏的逐语句执行或逐过程执行,单步

运行调试程序。通过选择“调试”→“窗口”→“局部变量”命令，打开“局部变量”调试窗口，查看局部变量 a、b、c 在运行过程中值的变化，如图 1-11 所示。

(13) 删除断点并修正逻辑错误。单击调试工具栏中的“停止调试”按钮，停止调试，将“int c=100/a+b;”改为“int c=100/(a+b);”。在“int a=20;”所在行的左侧断点区域再次单击鼠标，删除断点。按 F5 键运行测试该应用程序，得到正确的运行结果：4。

实验 1-4　Visual Studio 在线帮助的使用

实验要求：使用 Visual Studio 在线帮助功能。

操作步骤：

(1) 打开实验 1-2 的解决方案。执行“文件”→“打开”→“项目/解决方案”命令，在随后的“打开项目”对话框中选择 C:\C#LAB\LAB01\HelloConsole\HelloConsole.sln 解决方案文件，打开实验 1-2 的解决方案。

(2) 使用上下文帮助。鼠标定位到要获得帮助的对象，例如关键字 WriteLine 中，按 F1 键打开帮助窗口，显示 WriteLine 的帮助信息，如图 1-12 所示。

图 1-12　WriteLine 的帮助信息

实验 1-5　创建控制台应用程序 ConsoleClass

实验要求：使用 Visual Studio 编写控制台应用程序 ConsoleClass，实现主教材例 1.4 类和对象示例程序，输出平面上点的坐标值。运行效果如图 1-13 所示。

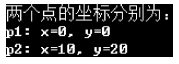


图 1-13　实验 1-5 运行效果

操作步骤：程序代码如图 1-14 所示。

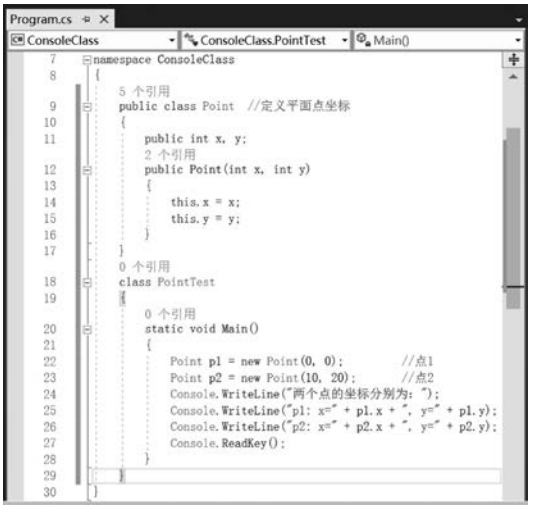
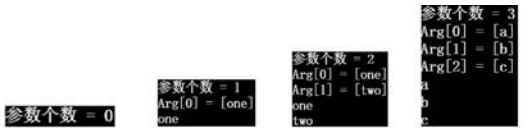


图 1-14　实验 1-5 程序代码

实验 1-6　创建控制台应用程序 ConsoleArgs

实验要求：使用 Visual Studio 编写控制台应用程序 ConsoleArgs，实现例 1.11 命令行参数示例程序，输出命令行参数个数以及各参数内容。运行效果如图 1-15 所示。



(a) 无参数　(b) 1个参数　(c) 2个参数　(d) 3个参数

图 1-15　实验 1-6 运行效果

操作步骤：请读者参考例 1.11 和实验 1-2 完成。程序代码如图 1-16 所示。

图 1-16　实验 1-6 程序代码

实验 2

程序流程控制（一）

实 验 目 的

- 掌握常量和变量的使用；
- 掌握运算符和表达式的使用；
- 掌握赋值语句的使用；
- 掌握顺序结构的程序流程；
- 掌握单分支语句的使用；
- 掌握双分支语句的使用；
- 掌握多分支语句的使用。

实 验 内 容

实验 2-1　常量、变量、运算符、表达式和赋值语句的使用

实验要求：输入半径，求对应圆的周长、面积；对应球体的体积。运行效果如图 2-1 所示。

图 2-1　实验 2-1 运行效果

操作提示：

（1）利用如下语句输入半径 r：

```
String s = Console.ReadLine();          //读入字符串
r = double.Parse(s);                    //将数字字符串转换为等效的双精度浮点数
```

（2）程序代码如图 2-2 所示。

```
static void Main(string[] args)
{
    const double PI = 3.14159;
    double r, perimeter, area, volume;
    Console.Write("请输入半径： ");
    String s = Console.ReadLine();
    r = double.Parse(s);        //将数字字符串转换为等效的双精度浮点数
    Console.WriteLine("圆的半径为 = {0}", r);
    perimeter = 2 * PI * r;
    area = PI * r * r;
    volume = 4 / 3 * PI * Math.Pow(r, 3);
    Console.WriteLine("圆的周长为 = {0}，圆的面积 = {1}", perimeter, area);
    Console.WriteLine("球体的体积 = {0}", volume);
    Console.ReadLine();
}
```

图 2-2　实验 2-1 程序代码

实验 2-2　求三角形周长和面积

实验要求：输入三角形三条边，先判断是否可以构成三角形，如果可以，则求三角形的周长和面积，否则报错。运行效果如图 2-3 所示。

(a) 三角形的周长和面积　　(b) 不能构成三角形

图 2-3　实验 2-2 运行效果

操作提示：

(1) 三个数可以构成三角形必须满足如下条件：每条边长均大于 0，并且任意两边之和大于第三边。

(2) 已知三角形的三条边，则三角形的面积 $=\sqrt{h(h-a)(h-b)(h-c)}$，其中 h 为三角形周长 p 的一半。

(3) 程序代码如图 2-4 所示。

```
static void Main(string[] args)
{
    double a, b, c, p, h, area;
    Console.Write("请输入三角形的边A:    ");
    String s = Console.ReadLine();
    a = double.Parse(s);        //边A
    Console.Write("请输入三角形的边B:    ");
    s = Console.ReadLine();
    b = double.Parse(s);        //边B
    Console.Write("请输入三角形的边C:    ");
    s = Console.ReadLine();
    c = double.Parse(s);        //边C
    if (a > 0 && b > 0 && c > 0 && a + b > c && a + c > b && b + c > a)
    {
        Console.WriteLine("三角形三边分别为:  a={0}, b={1}, c={2}", a, b, c);
        p = a + b + c;
        h = p / 2;
        area = Math.Sqrt(h * (h - a) * (h - b) * (h - c));
        Console.WriteLine("三角形的周长 = {0}, 面积 = {1}", p, area);
    }
    else Console.WriteLine("无法构成三角形!");
    Console.ReadKey();
}
```

图 2-4　实验 2-2 程序代码

实验 2-3　分段函数的实现

实验要求：输入 x，根据如下公式计算分段函数 y 的值。要求参照例 4.3，分别利用“一句单分支语句”“两句单分支语句”“双分支结构”以及“条件运算符”4 种方法实现。运行效果如图 2-5 所示。

$$y=\begin{cases}\dfrac{x^2-3x}{x^2+1}+2\pi+\sin x & x\geqslant 0\\ \ln(-5x)+6\sqrt{|x|+e^4}-(x+1)^3 & x<0\end{cases}$$

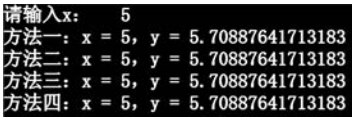


图 2-5　实验 2-3 运行效果

操作提示：程序代码如图 2-6 所示。

```
static void Main(string[] args)
{
    double x, y;
    Console.Write("请输入x：  ");
    String s = Console.ReadLine();
    x = double.Parse(s);
    // 方法一：利用“一句单分支语句”实现
    y = (x * x - 3 * x) / (x * x + 1) + 2 * Math.PI + Math.Sin(x);
    if (x < 0)
        y = Math.Log(-5 * x) + 6 * Math.Sqrt(Math.Abs(x) + Math.Pow(Math.E, 4)) - Math.Pow(x + 1, 3);
    Console.WriteLine("方法一：x = {0}, y = {1}", x, y);
    // 方法二：利用“两句单分支语句”实现
    if (x >= 0)
        y = (x * x - 3 * x) / (x * x + 1) + 2 * Math.PI + Math.Sin(x);
    if (x < 0)
        y = Math.Log(-5 * x) + 6 * Math.Sqrt(Math.Abs(x) + Math.Pow(Math.E, 4)) - Math.Pow(x + 1, 3);
    Console.WriteLine("方法二：x = {0}, y = {1}", x, y);
    // 方法三：利用“双分支结构”实现
    if (x >= 0)
        y = (x * x - 3 * x) / (x * x + 1) + 2 * Math.PI + Math.Sin(x);
    else
        y = Math.Log(-5 * x) + 6 * Math.Sqrt(Math.Abs(x) + Math.Pow(Math.E, 4)) - Math.Pow(x + 1, 3);
    Console.WriteLine("方法三：x = {0}, y = {1}", x, y);
    // 方法四：利用“条件运算符”实现
    y = (x >= 0) ? (x * x - 3 * x) / (x * x + 1) + 2 * Math.PI + Math.Sin(x) : Math.Log(-5 * x) + 6 * Math.Sqrt(Math.Abs(x) + Math.Pow(Math.E, 4)) - Math.Pow(x + 1, 3);
    Console.WriteLine("方法四：x = {0}, y = {1}", x, y);
    Console.ReadKey();
}
```

图 2-6　实验 2-3 程序代码

实验 2-4　三个数比较大小

实验要求：产生三个 0～100(包含 0 和 100)的随机数 a、b 和 c，按从小到大的顺序排序。运行效果如图 2-7 所示(其中 a、b 和 c 的值随机生成)。

```
原始值：   a=80,  b=44,  c=50
<方法一>升序值：   a=44,  b=50,  c=80
<方法二>升序值：   a=44,  b=50,  c=80
```

图 2-7　实验 2-4 运行效果

操作提示：

(1) 方法一：先 a 和 b 比较，使得 a＜b；然后 a 和 c 比较，使得 a＜c，此时 a 最小；最后 b 和 c 比较，使得 b＜c。

(2) 方法二：利用 Max 函数和 Min 函数求 a、b、c 三个数中的最大数、最小数，而三个数之和减去最大数和最小数就是中间数。

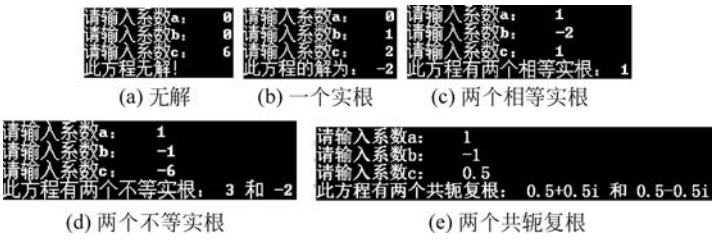
(3) 程序代码如图 2-8 所示。

```
static void Main(string[] args)
{
    int a, b, c, a1, b1, c1, t, Nmax, Nmin, Nmid;
    Random rNum = new Random();
    a = rNum.Next(101);  //产生0~100的随机数a
    b = rNum.Next(101);  //产生0~100的随机数b
    c = rNum.Next(101);  //产生0~100的随机数c
    Console.WriteLine("原始值：  a={0},  b={1},  c={2}", a, b, c);
    a1 = a; b1 = b; c1 = c;  // 保留a,b,c的值，以方便两种方法的比较
    //方法一：先a和b比较，使得a<b；然后a和c比较，使得a<c，此时a最小；最后b和c比较，使得b<c
    if (a > b)
    {
        t = a; a = b; b = t;
    }
    if (a > c)
    {
        t = a; a = c; c = t;
    }
    if (b > c)
    {
        t = b; b = c; c = t;
    }
    Console.WriteLine("(方法一)升序值：  a={0},  b={1},  c={2}", a, b, c);
    a = a1; b = b1; c = c1;  // 恢复a,b,c的值，使用第二种方法
    //方法二：利用Max函数和Min函数求a、b、c三个数中最大数、最小数，而三个数之和减去最大数和最小数就是中间数
    Nmax = Math.Max(Math.Max(a, b), c);
    Nmin = Math.Min(Math.Min(a, b), c);
    Nmid = a + b + c - Nmax - Nmin;
    a = Nmin;
    b = Nmid;
    c = Nmax;
    Console.WriteLine("(方法二)升序值：  a={0},  b={1},  c={2}", a, b, c);
    Console.ReadKey();
}
```

图 2-8　实验 2-4 程序代码

实验 2-5　求解一元二次方程

实验要求：输入一元二次方程的三个系数 a、b 和 c，求方程的解。运行效果如图 2-9 所示。



(a) 无解　(b) 一个实根　(c) 两个相等实根

(d) 两个不等实根　(e) 两个共轭复根

图 2-9　实验 2-5 运行效果

操作提示：

(1) 方程 $ax^2+bx+c=0$ 的解有以下几种情况：

① a=0 和 b=0，无解。

② a=0 和 b!=0，有一个实根：$x=-\frac{c}{b}$。

③ $b^2-4ac=0$，有两个相等实根：$x_1=x_2=-\frac{b}{2a}$。

④ $b^2-4ac>0$，有两个不等实根：$x_1=\frac{-b+\sqrt{b^2-4ac}}{2a}$，$x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}$。

⑤ $b^2-4ac<0$，有两个共轭复根：$x_1=-\frac{b}{2a}+\frac{\sqrt{4ac-b^2}}{2a}i$，$x_2=-\frac{b}{2a}-\frac{\sqrt{4ac-b^2}}{2a}i$。

(2) 程序代码如图 2-10 所示。

```
static void Main(string[] args)
{
    double a, b, c, delta, x1, x2, realPart, imagPart;
    Console.Write("请输入系数a:    ");
    String s = Console.ReadLine();
    a = double.Parse(s);          //系数a
    Console.Write("请输入系数b:    ");
    s = Console.ReadLine();
    b = double.Parse(s);          //系数b
    Console.Write("请输入系数c:    ");
    s = Console.ReadLine();
    c = double.Parse(s);          //系数c
    if (a == 0)
    {
        if (b == 0) Console.WriteLine("此方程无解! ");
        else Console.WriteLine("此方程的解为:  {0}", -c / b);
    }
    else
    {
        delta = b * b - 4 * a * c;
        if (delta > 0)
        {
            x1 = (-b + Math.Sqrt(delta)) / (2 * a);
            x2 = (-b - Math.Sqrt(delta)) / (2 * a);
            Console.WriteLine("此方程有两个不等实根:  {0} 和 {1} ", x1, x2);
        }
        else
        {
            if (delta == 0) Console.WriteLine("此方程有两个相等实根:  {0}", -b / (2 * a));
            else
            {
                realPart = -b / (2 * a);
                imagPart = Math.Sqrt(-delta) / (2 * a);
                Console.WriteLine("此方程有两个共轭复根:  {0}+{1}i 和 {0}-{1}i  ", realPart, imagPart);
            }
        }
    }
    Console.ReadKey();
}
```

图 2-10　实验 2-5 程序代码

实验 2-6　使用 switch 语句实现多重分支结构

实验要求：输入一个数字(1～7)，用中文显示对应的星期(星期一至星期日)。运行效果如图 2-11 所示。

(a) 显示星期　　(b) 错误提示

图 2-11　实验 2-6 运行效果

操作提示：程序代码如图 2-12 所示。

```
static void Main(string[] args)
{
    int i;
    Console.Write("输入一个数字(1~7)：");
    String s = Console.ReadLine();
    i = int.Parse(s);
    switch (i)
    {
        case 1:
            Console.WriteLine("对应的星期为：星期一");
            break;
        case 2:
            Console.WriteLine("对应的星期为：星期二");
            break;
        case 3:
            Console.WriteLine("对应的星期为：星期三");
            break;
        case 4:
            Console.WriteLine("对应的星期为：星期四");
            break;
        case 5:
            Console.WriteLine("对应的星期为：星期五");
            break;
        case 6:
            Console.WriteLine("对应的星期为：星期六");
            break;
        case 7:
            Console.WriteLine("对应的星期为：星期日");
            break;
        default:
            Console.WriteLine("输入错误！");
            break;
    }
    Console.ReadKey();
}
```

图 2-12　实验 2-6 程序代码

实验 2-7　分别使用 if 语句和 switch 语句实现多分支结构

实验要求：计算有固定工资收入的党员，每月所交纳的党费。计算有固定工资收入的党员，每月所交纳的党费。工资基数 3000 元及以下者，交纳工资基数的 0.5%；工资基数 3000 元到 5000 元者，交纳工资基数的 1%；工资基数在 5000 元到 10000 元者，交纳工资基数的 1.5%；工资基数超过 10000 元者，交纳工资基数的 2%。即

$$\text{党费 f}=\begin{cases}0.5\% * \text{salary} & \text{salary} \leqslant 3000\\ 1\% * \text{salary} & 3000 < \text{salary} \leqslant 5000\\ 1.5\% * \text{salary} & 5000 < \text{salary} \leqslant 10000\\ 2\% * \text{salary} & \text{salary} > 10000\end{cases}$$

运行效果如图 2-13 所示。

```
请输入有固定工资收入的党员的月工资基数：12000
月工资基数 = 12000，交纳党费 = 240
```

图 2-13　实验 2-7 运行效果

操作提示：

(1) 为了使用 switch 语句,首先使用如下语句将党费 f 的大范围区间数值转换为小范围区间数值(即 switch 语句中的控制表达式 c)：

```
if (salary > 10000) c = 11;
else c = (int)Math.Ceiling(salary/1000.0);
```

则党费 f 的计算公式变换为：

$$党费\ f=\begin{cases}0.5\% * salary & c=0\sim 3\\ 1\% * salary & c=4\sim 5\\ 1.5\% * salary & c=6\sim 7\\ 2\% * salary & c=11\end{cases}$$

(2) 使用 switch 语句实现的程序代码如图 2-14 所示。

```
static void Main(string[] args)
{
    int c; double f = 0;
    Console.Write("请输入有固定工资收入的党员的月工资基数：");
    int salary = int.Parse(Console.ReadLine());  //月工资基数
    if (salary > 10000) c = 11;
    else c = (int)Math.Ceiling(salary/1000.0);
    switch (c)
    {
        case 0:
        case 1:
        case 2:
        case 3:
            f = 0.5 / 100 * salary;  break;
        case 4:
        case 5:
            f = 1.0 / 100 * salary;  break;
        case 6:
        case 7:
        case 8:
        case 9:
        case 10:
            f =1.5 / 100 * salary;  break;
        case 11:
            f = 2.0 / 100 * salary; break;
    }
    Console.WriteLine("月工资基数 = {0}，交纳党费 = {1}", salary, f);
    Console.ReadKey();
}
```

图 2-14　实验 2-7 程序代码(switch 语句)

（3）使用 if 语句实现的程序代码如图 2-15 所示。

```
static void Main(string[] args)
{
    int c; double f = 0;
    Console.Write("请输入有固定工资收入的党员的月工资基数：");
    int salary = int.Parse(Console.ReadLine());  //月工资基数
    if (salary > 0 && salary <= 3000) f = 0.5 / 100 * salary;
    else if (salary > 3000 && salary <= 5000) f = 1.0 / 100 * salary;
    else if (salary > 5000 && salary <= 10000) f = 1.5 / 100 * salary;
    else if (salary > 10000) f = 2.0 / 100 * salary;
    else Console.WriteLine("月工资基数输入有误！");
    Console.WriteLine("月工资基数 = {0}, 交纳党费 = {1}", salary, f);
    Console.ReadKey();
}
```

图 2-15　实验 2-7 程序代码(if 语句)

（4）请改进 switch 语句的实现代码，使其能处理月工资基数小于 0 的情况。

（5）请使用 if 语句的其他语法形式实现本实验的功能。

实验3

程序流程控制(二)

实验目的

- 掌握 for 循环语句的使用；
- 掌握 while 循环语句的使用；
- 掌握 do…while 循环语句的使用；
- 掌握多重循环结构程序流程；
- 了解跳转语句的使用；
- 了解程序异常处理机制。

实验内容

实验 3-1　求 n!

实验要求：输入整数 n(n≥0)，分别利用 for 循环、while 循环、do…while 循环求 n!。运行效果如图 3-1 所示。

图 3-1　实验 3-1 运行效果

操作提示：

(1) n!=n×(n−1)×(n−2)×…×2×1。例如，5!=5×4×3×2×1=120。特别地，0!=1。

(2) 一般地，累乘的初值为 1，而累加的初值为 0。

(3) 如果输入的是负整数，则继续提示输入非负整数，直至 n≥0。

(4) 程序代码如图 3-2 所示。

实验 3-2　显示 Fibonacci 数列

实验要求：显示 Fibonacci 数列：1,1,2,3,5,8,…。当 Fibonacci 的值大于 10000 时停止显示。要求每行显示 5 项，运行效果如图 3-3 所示。

```
static void Main(string[] args)
{
    int i, n, fac = 1;
    string s;
    n = -1;
    while (n < 0)
    {
        Console.Write("请输入非负整数n: ");
        s = Console.ReadLine();
        n = int.Parse(s);
    }
    //方法一: for循环
    for (i = 1; i <= n; i++) fac *= i;
    Console.WriteLine("       for循环: {0}! = {1}", n, fac);
    //方法二: while循环
    i = 1; fac = 1;
    while (i <= n) fac *= i++;
    Console.WriteLine("     while循环: {0}! = {1}", n, fac);
    //方法三: do...while循环
    i = 1; fac = 1;
    do
    {
        fac *= i; i++;
    } while (i <= n);
    Console.WriteLine("do...while循环: {0}! = {1}", n, fac);
    Console.ReadKey();
}
```

图 3-2　实验 3-1 程序代码

```
  1   1   2   3   5
  8  13  21  34  55
 89 144 233 377 610
987 1597 2584 4181 6765
```

图 3-3　实验 3-2 运行效果

操作提示：Fibonacci 数列的生成规律为：

$$\begin{cases} F_1 = 1 & n = 1 \\ F_2 = 1 & n = 2 \\ F_n = F_{n-1} + F_{n-2} & n \geqslant 3 \end{cases}$$

程序代码如图 3-4 所示。

```
static void Main(string[] args)
{
    int f1 = 1, f2 = 1, f3, num = 2;
    Console.Write("{0,5}\t{0,5}\t", f1, f2);
    f3 = f1 + f2;
    while (f3 <= 10000)
    {
        Console.Write("{0,5}\t", f3);
        num++;
        if (num % 5 == 0) Console.WriteLine();
        f1 = f2;
        f2 = f3;
        f3 = f1 + f2;
    }
    Console.ReadKey();
}
```

图 3-4　实验 3-2 程序代码

实验 3-3　鸡兔同笼问题

实验要求：已知在同一个笼子里总共有 h 只鸡和兔，鸡和兔的总脚数为 f 只，其中 h 和 f 由用户输入，求鸡和兔各有多少只？要求使用两种方法：一是求解方程；二是利用循环进行枚举测试。运行效果如图 3-5 所示。

(a) 合理解　　(b) 无解

图 3-5　实验 3-3 运行效果

操作提示：

(1) 已知鸡和兔的总数为 h,它们的总脚数为 f。假设鸡有 c 只,兔有 r 只。

(2) 方法一：求解方程法。由公式：

$$\begin{cases} c + r = h \\ 2c + 4r = f \end{cases}$$

解得：

$$\begin{cases} r = \dfrac{f}{2} - h \\ c = h - r \end{cases}$$

由公式推得,鸡和兔的总脚数 f 必须是偶数,并且鸡和兔的只数必须是非负整数。

(3) 方法二：利用循环进行枚举测试。鸡的只数 c 取值范围为 0～h,兔的只数为 r,如果满足条件,则求得解。

(4) 程序代码如图 3-6 所示。

```
static void Main(string[] args)
{
    int c, r;                       //number of chicken & rabbit
    Console.Write("请输入总头数： ");
    String s = Console.ReadLine();
    int h = int.Parse(s);           //total heads of chicken & rabbit
    int f = 1;
    while (f % 2 != 0)
    {
        Console.Write("请输入总脚数(必须是偶数)： ");
        s = Console.ReadLine();
        f = int.Parse(s);           //total feet of chicken & rabbit
    }
    //方法一：利用循环
    bool solution = false;          //判断是否有解
    for (c = 0; c <= h; c++)
    {
        r = h - c;
        if (2 * c + 4 * r == f)
        {
            Console.WriteLine("方法一：鸡：{0} 只， 兔：{1} 只", c, r);
            solution = true;
        }
    }
    if (!solution) Console.WriteLine("方法一：无解，请重新运行测试! ");
    //方法二：解方程
    r = f / 2 - h;
    c = h - r;
    solution = false;               //判断是否有解
    if (r >= 0 && c >= 0)
    {
        Console.WriteLine("方法二：鸡：{0} 只， 兔：{1} 只", c, r);
        solution = true;
    }
    if (!solution) Console.WriteLine("方法二：无解，请重新运行测试! ");
    Console.ReadKey();
}
```

图 3-6　实验 3-3 程序代码