Excel VBA 从入门到精通：实战指南与案例大全

前言

在数据处理与办公自动化的浪潮中，Excel 早已成为职场人必备的工具，但默认功能往往难以满足复杂的业务需求。Excel VBA（Visual Basic for Applications）作为内置的编程语言，能够让你突破 Excel 的功能边界，通过编写代码实现重复操作自动化、复杂数据处理、自定义功能开发等高级应用。

本书专为 Excel 用户打造，无论你是零基础的办公人员，还是希望提升效率的数据分析从业者，都能通过循序渐进的学习掌握 VBA 核心技能。书中摒弃晦涩的理论说教，以“案例驱动”为核心，结合实际办公场景，从基础语法到高级应用，从代码编写到实战优化，全方位讲解 Excel VBA 的实用技巧，帮助你真正将 VBA 融入工作，大幅提升办公效率。

第 1 章：Excel VBA 入门基础

1.1 什么是 Excel VBA

VBA（Visual Basic for Applications）是微软为 Office 套件开发的内置编程语言，它基于 Visual Basic 语言，专门用于扩展 Office 应用程序的功能。在 Excel 中，VBA 可以实现以下核心功能：

- 自动化重复操作（如批量格式设置、数据录入）；
- 处理复杂数据计算与分析（如多表数据汇总、条件筛选）；
- 创建自定义功能（如个性化函数、交互界面）；
- 实现跨文件 / 跨应用的数据交互（如与 Word、Access 联动）。

简单来说，VBA 就像是 Excel 的“万能工具箱”，能让你用代码指挥 Excel 完成各种手动操作，甚至实现手动无法完成的复杂任务。

1.2 启用 VBA 开发环境

默认情况下，Excel 的 VBA 开发环境（VBE）未显示，需手动启用：

1. 打开 Excel，点击【文件】→【选项】→【自定义功能区】；
2. 在右侧“主选项卡”列表中，勾选【开发工具】，点击【确定】；
3. 此时 Excel 顶部菜单栏会出现【开发工具】选项卡，点击【Visual Basic】（或按快捷键 Alt + F11），即可打开 VBA 开发环境。

1.3 VBA 开发环境（VBE）介绍

VBE 是编写、调试 VBA 代码的核心界面，主要由以下部分组成：

- **工程资源管理器**：显示当前 Excel 文件（称为“工程”）的所有组件，包括工作表、工作簿、模块、用户窗体等（按 Ctrl + R 显示 / 隐藏）；
- **代码窗口**：编写代码的区域，双击工程资源管理器中的组件（如“模块 1”）即可打开；
- **属性窗口**：显示选中对象的属性（如工作表名称、窗体大小），按 F4 显示 / 隐藏；
- **立即窗口**：用于调试代码、执行临时命令（如输出变量值），按 Ctrl + G 显示 / 隐藏；
- **工具栏**：包含常用操作按钮（如运行代码、插入模块），可通过【视图】→【工具栏】自定义显示。

1.4 第一个 VBA 程序：Hello World

通过简单的“Hello World”程序，快速体验 VBA 的运行逻辑：

1. 打开 Excel，按 Alt + F11 进入 VBE；
2. 右键点击工程资源管理器中的当前工作簿名称，选择【插入】→【模块】，新建一个标准模块；
3. 在代码窗口中输入以下代码：

```
Sub HelloWorld()
    ' 这是注释：弹出消息框显示“Hello World”
    MsgBox "Hello World! 这是我的第一个 VBA 程序"
End Sub
```

1. 点击代码窗口顶部的【运行子程序 / 用户窗体】按钮（或按 F5），即可执行代码，Excel 会弹出消息框显示对应内容。

代码说明：

- Sub HelloWorld()：定义一个子程序（宏），HelloWorld 是程序名称（可自定义，需以字母开头，不能包含特殊字符）；
- 单引号 ' 后面的内容为注释，不会执行，用于说明代码功能；
- MsgBox 是 VBA 的内置函数，用于弹出消息框；
- End Sub：表示子程序结束。

第 2 章：VBA 核心语法基础

2.1 变量与数据类型

变量是存储数据的容器，使用前需明确其数据类型，以提高代码效率和稳定性。

2.1.1 变量的声明

VBA 中变量声明有两种方式：显式声明和隐式声明：

- **隐式声明**：直接使用变量（如 x = 10），无需提前定义，缺点是容易因拼写错误导致 bug；
- **显式声明**：使用 Dim 语句提前定义变量，推荐使用（可通过 Option Explicit 强制要求显式声明）。

强制显式声明：在模块的最顶部输入 Option Explicit，此时未声明的变量会报错，避免拼写错误。

2.1.2 常用数据类型

VBA 支持多种数据类型，常用类型如下：

数据类型	说明	占用空间	示例
Integer	整数（-32768 到 32767）	2 字节	Dim age As Integer
Long	长整数（-2147483648 到 2147483647）	4 字节	Dim id As Long
Double	双精度浮点数（用于小数）	8 字节	Dim score As Double
String	字符串（文本）	1 字节 / 字符	Dim name As String
Boolean	布尔值（True/False）	2 字节	Dim isOK As Boolean
Date	日期时间	8 字节	Dim today As Date

Object	对象（如工作表、单元格）	4 字节	Dim ws As Worksheet

声明示例：

```
Option Explicit ' 强制显式声明
Sub DeclareVariable()
    Dim userName As String ' 字符串变量：存储用户名
    Dim userAge As Integer ' 整数变量：存储年龄
    Dim userScore As Double ' 浮点数变量：存储分数
    Dim isPass As Boolean ' 布尔变量：存储是否通过

    ' 给变量赋值
    userName = "张三"
    userAge = 25
    userScore = 89.5
    isPass = userScore >= 60

    ' 弹出消息框显示变量值
    MsgBox userName & "，" & userAge & "岁，分数：" & userScore & "，是否通过：" & isPass
End Sub
```

2.2 运算符

VBA 中的运算符用于执行计算或比较操作，主要分为以下几类：

2.2.1 算术运算符

用于数值计算，常用运算符：+（加）、-（减）、*（乘）、/（除）、^（幂）、Mod（取余）。

示例：

```
Sub ArithmeticOperator()
    Dim a As Integer, b As Integer
    a = 10: b = 3 ' 一行声明多个变量，用冒号分隔
    MsgBox "a + b = " & a + b ' 结果：13
    MsgBox "a * b = " & a * b ' 结果：30
    MsgBox "a Mod b = " & a Mod b ' 结果：1（10 除以 3 的余数）
End Sub
```

2.2.2 比较运算符

用于比较两个值，返回布尔值（True/False），常用运算符：=（等于）、、>（大于）、、>=（大于等于）、`（小于等于）。

示例：

```
Sub CompareOperator()
    Dim score As Double
    score = 75
    If score >= 60 Then
        MsgBox "成绩合格" ' 条件成立，执行此语句
    Else
        MsgBox "成绩不合格"
    End If
End Sub
```

2.2.3 字符串运算符

用于字符串拼接，运算符：&（推荐）、+（可能因类型混淆出错）。

示例：

```
Sub StringOperator()
    Dim firstName As String, lastName As String
    firstName = "李"
```

```
    lastName = "四"
    MsgBox "姓名：" & lastName & firstName ' 结果：姓名：李四
End Sub
```

2.3 流程控制语句

流程控制语句用于控制代码的执行顺序，主要包括条件语句和循环语句。

2.3.1 条件语句（If...Then...Else）

根据条件执行不同的代码块，语法如下：

```
' 单条件
If 条件 Then
    条件成立时执行的代码
Else
    条件不成立时执行的代码
End If
' 多条件
If 条件 1 Then
    条件 1 成立时执行的代码
ElseIf 条件 2 Then
    条件 2 成立时执行的代码
Else
    所有条件都不成立时执行的代码
End If
```

示例：根据分数评级

```
Sub ScoreGrade()
    Dim score As Double
    score = InputBox("请输入分数：") ' 弹出输入框获取分数
```

```
    If score >= 90 Then
        MsgBox "评级：优秀"
    ElseIf score >= 80 Then
        MsgBox "评级：良好"
    ElseIf score >= 60 Then
        MsgBox "评级：合格"
    Else
        MsgBox "评级：不合格"
    End If
End Sub
```

2.3.2 循环语句（For...Next、Do...Loop）

用于重复执行某段代码，常用两种循环：

（1）For...Next 循环（固定次数循环）

语法：

```
For 变量 = 起始值 To 结束值 Step 步长 ' 步长默认 1，可省略
    循环执行的代码
Next 变量
```

示例：计算 1 到 10 的和

```
Sub ForLoopSum()
    Dim sum As Integer, i As Integer
    sum = 0
    For i = 1 To 10 ' i 从 1 到 10，步长 1
        sum = sum + i ' 累加 i 的值
    Next i
    MsgBox "1 到 10 的和为：" & sum ' 结果：55
```

```
End Sub
```

（2）Do...Loop 循环（条件满足时循环）

语法 1：先判断条件，再执行循环

```
Do While 条件
    循环执行的代码
Loop
```

语法 2：先执行一次循环，再判断条件

```
Do
    循环执行的代码
Loop While 条件
```

示例：计算 1 到 100 中偶数的和

```
Sub DoLoopSum()
    Dim sum As Integer, i As Integer
    sum = 0
    i = 1
    Do While i 100
        If i Mod 2 = 0 Then ' 判断是否为偶数
            sum = sum + i
        End If
        i = i + 1 ' 变量自增
    Loop
    MsgBox "1 到 100 中偶数的和为：" & sum ' 结果：2550
End Sub
```

2.4 数组

数组是存储多个同类型数据的集合，可分为一维数组和多维数组，适用于批量处理数据。

2.4.1 一维数组

语法：Dim 数组名(起始索引 To 结束索引) As 数据类型（默认起始索引为 0，可通过 Option Base 1 设置为 1）。

示例：存储并遍历学生姓名

```
Option Base 1 ' 设置数组起始索引为 1
Sub OneDimensionalArray()
    Dim students(5) As String ' 一维数组，索引 1-5
    Dim i As Integer

    ' 给数组赋值
    students(1) = "张三"
    students(2) = "李四"
    students(3) = "王五"
    students(4) = "赵六"
    students(5) = "孙七"

    ' 遍历数组并显示
    For i = 1 To 5
        MsgBox "第" & i & "个学生：" & students(i)
    Next i
End Sub
```

2.4.2 多维数组

常用二维数组（类似表格的行和列），语法：Dim 数组名(行索引 To 行结束, 列索引 To 列结束) As 数据类型。

示例：存储学生姓名和分数

```
Option Base 1
Sub TwoDimensionalArray()
    Dim studentInfo(3, 2) As Variant ' 二维数组：3 行 2 列（3 个学生，姓名+分数）
    Dim i As Integer

    ' 给数组赋值（行：学生，列 1：姓名，列 2：分数）
    studentInfo(1, 1) = "张三": studentInfo(1, 2) = 85
    studentInfo(2, 1) = "李四": studentInfo(2, 2) = 92
    studentInfo(3, 1) = "王五": studentInfo(3, 2) = 78

    ' 遍历数组并显示
    For i = 1 To 3
        MsgBox studentInfo(i, 1) & "的分数：" & studentInfo(i, 2)
    Next i
End Sub
```

第 3 章：Excel 对象模型与核心操作

3.1 Excel 对象模型概述

Excel VBA 的核心是通过操作 “对象” 来控制 Excel，Excel 的对象模型层次分明，主要核心对象包括：

- **Application**：代表整个 Excel 应用程序；
- **Workbook**：代表一个 Excel 工作簿（.xlsx/.xls 文件）；
- **Worksheet**：代表一个工作表（如 “Sheet1”）；
- **Range**：代表单元格或单元格区域（如 “A1”“A1:B10”）；
- **Chart**：代表图表；
- **Shape**：代表形状（如图片、文本框）。

对象之间的层级关系：Application → Workbook → Worksheet → Range，通过 “点语法” 访问子对象，例如：Application.Workbooks("工作簿1.xlsx").Worksheets("Sheet1").Range("A1")。

3.2 工作簿（Workbook）操作

3.2.1 打开工作簿

语法：Workbooks.Open(文件名及路径)

示例：打开指定路径的工作簿

```
Sub OpenWorkbook()
    Dim wb As Workbook
    ' 打开 D 盘下的“数据.xlsx”，并赋值给变量 wb
    Set wb = Workbooks.Open("D:\数据.xlsx")
    MsgBox "已打开工作簿：" & wb.Name
End Sub
```

3.2.2 新建工作簿

语法：Workbooks.Add

示例：新建工作簿并保存

```
Sub NewWorkbook()
    Dim wb As Workbook
    Set wb = Workbooks.Add ' 新建空白工作簿
    wb.SaveAs "D:\新建工作簿.xlsx" ' 保存工作簿到指定路径
    MsgBox "已新建并保存工作簿：" & wb.Name
End Sub
```

3.2.3 关闭工作簿

语法：Workbook.Close(保存标识)，True 表示保存，False 表示不保存，省略则弹出保存提示。

示例：关闭工作簿

```
Sub CloseWorkbook()
    Dim wb As Workbook
    Set wb = Workbooks("数据.xlsx") ' 选中已打开的工作簿
    wb.Close SaveChanges:=True ' 保存并关闭
    MsgBox "已关闭工作簿：" & wb.Name
End Sub
```

3.3 工作表（Worksheet）操作

3.3.1 新建工作表

语法：Worksheets.Add

示例：新建工作表并命名

```
Sub NewWorksheet()
    Dim ws As Worksheet
    Set ws = Worksheets.Add ' 新建工作表
    ws.Name = "销售数据 2024" ' 给工作表重命名
    MsgBox "已新建工作表：" & ws.Name
End Sub
```

3.3.2 激活 / 选中工作表

语法：Worksheet.Activate

示例：激活指定工作表

```
Sub ActivateWorksheet()
    Worksheets("销售数据 2024").Activate ' 激活“销售数据 2024”工作表
    MsgBox "当前激活的工作表：" & ActiveSheet.Name
End Sub
```

3.3.3 删除工作表

语法：Worksheet.Delete，删除前会弹出确认提示，可通过 Application.DisplayAlerts = False 取消提示。

示例：删除指定工作表

```
Sub DeleteWorksheet()
    Dim ws As Worksheet
    Set ws = Worksheets("临时表")
    Application.DisplayAlerts = False ' 取消删除确认提示
    ws.Delete ' 删除工作表
    Application.DisplayAlerts = True ' 恢复提示（避免影响其他操作）
    MsgBox "已删除工作表：" & ws.Name
End Sub
```

3.3.4 复制工作表

语法：Worksheet.Copy(Before/After)，指定复制到哪个工作表之前或之后。

示例：复制工作表

```
Sub CopyWorksheet()
    ' 将“销售数据 2024”复制到“Sheet1”之后
    Worksheets("销售数据 2024").Copy After:=Worksheets("Sheet1")
    ' 给复制后的工作表重命名
    ActiveSheet.Name = "销售数据 2024-副本"
    MsgBox "已复制工作表"
End Sub
```

3.4 单元格区域（Range）操作

Range 是 Excel VBA 中最常用的对象，用于操作单元格或单元格区域，支持赋值、格式设置、数据读取等功能。

3.4.1 引用单元格区域

常用引用方式：

- 单个单元格：`Range("A1")` 或 `Cells(1, 1)`（行号，列号）；
- 连续区域：`Range("A1:B10")` 或 `Range(Cells(1,1), Cells(10,2))`；
- 整行 / 整列：`Range("A:A")`（A 列）、`Range("1:1")`（第 1 行）；
- 多个不连续区域：`Range("A1:A5, C1:C5")`。

示例：引用不同区域

```
Sub ReferenceRange()
    ' 给 A1 单元格赋值
    Range("A1") = "姓名"
    ' 给 B1 单元格赋值（用 Cells 引用，行 1，列 2）
    Cells(1, 2) = "分数"
    ' 给 A2:A5 区域赋值为“张三”“李四”等
    Range("A2:A5") = Array("张三", "李四", "王五", "赵六")
    ' 给 B2:B5 区域赋值为 85、92、78、90
    Range(Cells(2, 2), Cells(5, 2)) = Array(85, 92, 78, 90)
End Sub
```

3.4.2 读取 / 修改单元格值

通过 `Range.Value` 属性读取或修改单元格值（`Value` 可省略，直接 `Range("A1") = 10`）。

示例：读取和修改单元格值

```
Sub ReadWriteRange()
    Dim name As String, score As Double
    ' 读取 A2 和 B2 单元格的值
    name = Range("A2").Value
    score = Range("B2").Value
    MsgBox name & "的原始分数：" & score
```

```
    ' 修改 B2 单元格的值（加 5 分）
    Range("B2") = score + 5
    MsgBox name & "的新分数：" & Range("B2").Value
End Sub
```

3.4.3 设置单元格格式

通过 Range.Font（字体）、Range.Interior（填充色）、Range.Borders（边框）等属性设置格式。

示例：设置单元格格式

```
Sub FormatRange()
    Dim titleRange As Range
    Set titleRange = Range("A1:B1") ' 选中标题区域

    ' 设置字体：黑体、12 号、加粗、红色
    With titleRange.Font
        .Name = "黑体"
        .Size = 12
        .Bold = True
        .Color = RGB(255, 0, 0) ' 红色
    End With

    ' 设置填充色：浅黄色
    titleRange.Interior.Color = RGB(255, 255, 204)

    ' 设置边框：所有边框为细黑线
    titleRange.Borders.LineStyle = xlContinuous
    titleRange.Borders.Weight = xlThin
    titleRange.Borders.Color = RGB(0, 0, 0)
End Sub
```

3.4.4 批量处理单元格区域

结合循环语句批量处理区域内的单元格。

示例：批量设置不及格分数为红色

```
Sub BatchFormatRange()
    Dim scoreRange As Range, cell As Range
    Set scoreRange = Range("B2:B10") ' 分数区域

    ' 遍历区域内的每个单元格
    For Each cell In scoreRange
        If cell.Value 0 Then ' 分数不及格
            cell.Font.Color = RGB(255, 0, 0) ' 字体红色
            cell.Interior.Color = RGB(255, 204, 204) ' 填充浅红色
        End If
    Next cell
    MsgBox "已完成不及格分数标记"
End Sub
```

第 4 章：VBA 实战案例（办公自动化）

4.1 案例 1：批量生成员工工资条

需求：现有员工工资表（包含姓名、部门、基本工资、绩效工资、扣除项、实发工资），需为每位员工生成独立的工资条，包含表头和员工数据。

实现步骤：

1. 复制工资表表头到新位置；
2. 插入空白行分隔不同员工的工资条；
3. 复制每位员工的数据到对应位置。

代码：

```
Sub GenerateSalarySlip()
    Dim wsSource As Worksheet, wsTarget As Worksheet
    Dim lastRow As Integer, i As Integer, targetRow As Integer

    ' 设置源工作表（工资表）和目标工作表（工资条）
    Set wsSource = Worksheets("工资表")
    On Error Resume Next ' 忽略工作表已存在的错误
    Set wsTarget = Worksheets("工资条")
    On Error GoTo 0
    If wsTarget Is Nothing Then ' 如果目标工作表不存在，新建
        Set wsTarget = Worksheets.Add
        wsTarget.Name = "工资条"
    End If

    ' 获取源表最后一行数据
    lastRow = wsSource.Cells(wsSource.Rows.Count, "A").End(xlUp).Row
    targetRow = 1 ' 目标表起始行

    ' 复制表头到目标表
    wsSource.Range("A1:F1").Copy wsTarget.Cells(targetRow, 1)
    targetRow = targetRow + 2 ' 表头后空一行

    ' 批量生成工资条
    For i = 2 To lastRow ' 从第 2 行开始（跳过表头）
        ' 复制员工数据
        wsSource.Range("A" & i & ":F" & i).Copy wsTarget.Cells(targetRow, 1)
        ' 复制表头到下一个工资条
        targetRow = targetRow + 1
        wsSource.Range("A1:F1").Copy wsTarget.Cells(targetRow, 1)
        targetRow = targetRow + 2 ' 空一行，准备下一个员工
```

```
    Next i

    ' 调整目标表列宽
    wsTarget.Columns("A:F").AutoFit
    MsgBox "工资条生成完成！共生成" & lastRow - 1 & "份工资条"
End Sub
```

4.2 案例 2：多工作表数据汇总

需求：现有多个月份的销售工作表（如 “1 月”“2 月”“3 月”），每个表结构相同（A 列：产品名称，B 列：销售额），需将所有工作表的销售数据汇总到 “汇总表” 中，相同产品的销售额累加。

实现步骤：

1. 新建 “汇总表” 并设置表头；
2. 遍历所有销售工作表；
3. 读取每个工作表的销售数据，汇总到 “汇总表”（相同产品累加）。

代码：

```
Sub SummarizeSalesData()
    Dim ws As Worksheet, wsSummary As Worksheet
    Dim lastRow As Integer, i As Integer, j As Integer
    Dim productName As String, salesAmount As Double
    Dim isExist As Boolean

    ' 新建或选中汇总表
    On Error Resume Next
    Set wsSummary = Worksheets("汇总表")
    On Error GoTo 0
    If wsSummary Is Nothing Then
        Set wsSummary = Worksheets.Add
        wsSummary.Name = "汇总表"
```

```
        ' 设置汇总表表头
        wsSummary.Range("A1") = "产品名称"
        wsSummary.Range("B1") = "总销售额"
        wsSummary.Range("A1:B1").Font.Bold = True
    Else
        ' 清空汇总表已有数据（保留表头）
        wsSummary.Range("A2:B" & wsSummary.Rows.Count).ClearContents
    End If

    ' 遍历所有工作表
    For Each ws In Worksheets
        ' 跳过汇总表本身
        If ws.Name 汇总表" Then
            lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
            ' 读取当前工作表的销售数据（从第 2 行开始）
            For i = 2 To lastRow
                productName = ws.Range("A" & i).Value
                salesAmount = ws.Range("B" & i).Value
                isExist = False ' 标记产品是否已在汇总表中

                ' 检查汇总表中是否已有该产品
                For j = 2 To wsSummary.Cells(wsSummary.Rows.Count, "A").End(xlUp).Row
                    If wsSummary.Range("A" & j).Value = productName Then
                        ' 产品已存在，累加销售额
                        wsSummary.Range("B" & j).Value = wsSummary.Range("B" & j).Value +
salesAmount
                        isExist = True
                        Exit For
                    End If
                Next j
```

```
            ' 产品不存在，新增一行
            If Not isExist Then
                j = wsSummary.Cells(wsSummary.Rows.Count, "A").End(xlUp).Row + 1
                wsSummary.Range("A" & j).Value = productName
                wsSummary.Range("B" & j).Value = salesAmount
            End If
        Next i
    End If
Next ws

' 调整列宽并激活汇总表
wsSummary.Columns("A:B").AutoFit
wsSummary.Activate
MsgBox "数据汇总完成！"
End Sub
```

4.3 案例 3：自动生成数据报表与图表

需求：根据 “销售数据” 工作表中的数据（A 列：月份，B 列：销售额），自动生成柱状图报表，包含图表标题、坐标轴标签，并设置图表格式。

实现步骤：

1. 读取 “销售数据” 中的有效数据；
2. 插入柱状图；
3. 设置图表标题、坐标轴标签、颜色等格式。

代码：

```
Sub GenerateSalesChart()
    Dim ws As Worksheet, cht As Chart
    Dim lastRow As Integer, dataRange As Range

    ' 选中销售数据工作表
```

```
Set ws = Worksheets("销售数据")
lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
Set dataRange = ws.Range("A1:B" & lastRow) ' 数据区域（包含表头）

' 删除已存在的同名图表
On Error Resume Next
ActiveSheet.ChartObjects("销售趋势图").Delete
On Error GoTo 0

' 插入柱状图（在当前工作表）
Set cht = ws.Shapes.AddChart2(251, xlColumnClustered).Chart ' 251 对应柱状图类型
cht.Parent.Name = "销售趋势图" ' 给图表命名

' 设置图表数据来源
cht.SetSourceData Source:=dataRange

' 设置图表标题
cht.HasTitle = True
cht.ChartTitle.Text = "2024 年销售趋势图"
cht.ChartTitle.Font.Name = "微软雅黑"
cht.ChartTitle.Font.Size = 14
cht.ChartTitle.Font.Bold = True

' 设置 X 轴（月份）标签
cht.Axes(xlCategory, xlPrimary).HasTitle = True
cht.Axes(xlCategory, xlPrimary).AxisTitle.Text = "月份"
cht.Axes(xlCategory, xlPrimary).AxisTitle.Font.Size = 12

' 设置 Y 轴（销售额）标签
cht.Axes(xlValue, xlPrimary).HasTitle = True
```

```
    cht.Axes(xlValue, xlPrimary).AxisTitle.Text = "销售额（元）"
    cht.Axes(xlValue, xlPrimary).AxisTitle.Font.Size = 12

    ' 设置数据系列颜色（蓝色）
    cht.FullSeriesCollection(1).Format.Fill.ForeColor.RGB = RGB(54, 162, 235)

    ' 调整图表位置和大小
    cht.Parent.Left = ws.Range("D1").Left
    cht.Parent.Top = ws.Range("D1").Top
    cht.Parent.Width = 500
    cht.Parent.Height = 300

    MsgBox "销售趋势图生成完成！"
End Sub
```

第 5 章：VBA 进阶应用

5.1 自定义函数（User Defined Function）

Excel 内置函数无法满足需求时，可通过 VBA 编写自定义函数，在工作表中直接调用。

5.1.1 自定义函数的创建

语法：Function 函数名(参数 1 As 类型, 参数 2 As 类型) As 返回值类型

示例：自定义“多条件求和”函数（计算指定部门、指定岗位的员工工资总和）

```
Function SumSalary(dept As String, position As String) As Double
    Dim ws As Worksheet
    Dim lastRow As Integer, i As Integer
    Dim salary As Double

    Set ws = Worksheets("员工工资表")
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
```

```
    salary = 0

    ' 遍历员工数据，满足条件则累加工资
    For i = 2 To lastRow
        If ws.Range("B" & i).Value = dept And ws.Range("C" & i).Value = position Then
            salary = salary + ws.Range("F" & i).Value ' F 列为实发工资
        End If
    Next i

    SumSalary = salary ' 返回计算结果
End Function
```

5.1.2 调用自定义函数

在 Excel 工作表单元格中输入：`=SumSalary("销售部", "销售员")`，即可计算销售部所有销售员的实发工资总和。

5.2 用户窗体（UserForm）交互设计

用户窗体是可视化的交互界面，可通过文本框、按钮、下拉框等控件收集用户输入，提升 VBA 程序的易用性。

5.2.1 创建用户窗体

1. 打开 VBE，右键点击工程资源管理器，选择【插入】→【用户窗体】；
2. 在工具箱中选择控件（如文本框、命令按钮、标签），拖动到窗体上；
3. 选中控件，在属性窗口修改属性（如名称、标题、大小）。

5.2.2 案例：用户登录窗体

需求：创建登录窗体，输入用户名和密码，验证通过后执行数据汇总程序。

步骤：

1. 创建用户窗体，添加 2 个标签（“用户名”“密码”）、2 个文本框（`txtUserName`、`txtPassword`，密码框设置 `PasswordChar = *`）、1 个命令按钮（`cmdLogin`，标题 “登录”）；
2. 编写命令按钮的点击事件代码。

代码:

```
' 登录按钮点击事件
Private Sub cmdLogin_Click()
    Dim userName As String, password As String
    userName = Me.txtUserName.Value ' 获取用户名输入
    password = Me.txtPassword.Value ' 获取密码输入

    ' 验证用户名和密码（此处为示例，实际可从工作表或数据库读取）
    If userName = "admin" And password = "123456" Then
        MsgBox "登录成功！即将开始数据汇总", vbInformation
        Me.Hide ' 隐藏登录窗体
        SummarizeSalesData ' 调用第 4 章的多表数据汇总函数
        Unload Me ' 关闭登录窗体
    Else
        MsgBox "用户名或密码错误，请重新输入！", vbExclamation
        Me.txtUserName.Value = "" ' 清空用户名
        Me.txtPassword.Value = "" ' 清空密码
        Me.txtUserName.SetFocus ' 光标定位到用户名文本框
    End If
End Sub
' 窗体加载事件（初始化）
Private Sub UserForm_Initialize()
    Me.Caption = "数据汇总系统登录" ' 设置窗体标题
    Me.txtPassword.PasswordChar = "*" ' 密码框显示为星号
End Sub
```

调用登录窗体:

在模块中编写代码，点击按钮时显示登录窗体:

```
Sub ShowLoginForm()
    LoginForm.Show ' 显示登录窗体（LoginForm 为用户窗体名称）
End Sub
```

5.3 错误处理与代码优化

5.3.1 错误处理（On Error 语句）

VBA 程序运行时可能因各种原因报错（如文件不存在、数据格式错误），通过错误处理语句可捕获错误，避免程序崩溃。

常用错误处理语法：

```
Sub 程序名()
    On Error GoTo ErrorHandler ' 遇到错误时跳转到 ErrorHandler 标签

    ' 正常执行的代码
    ...

    Exit Sub ' 正常结束，跳过错误处理代码

ErrorHandler: ' 错误处理代码
    MsgBox "程序执行出错！错误编号：" & Err.Number & "，错误描述：" & Err.Description
End Sub
```

示例：带错误处理的 “打开工作簿” 程序

```
Sub OpenWorkbookWithErrorHandle()
    Dim wb As Workbook
    Dim filePath As String

    On Error GoTo ErrorHandler
```

```
    filePath = "D:\数据.xlsx"
    Set wb = Workbooks.Open(filePath)
    MsgBox "已成功打开工作簿：" & wb.Name

    Exit Sub

ErrorHandler:
    Select Case Err.Number
        Case 1004 ' 文件不存在或路径错误
            MsgBox "错误：找不到文件“" & filePath & "”，请检查文件路径是否正确", vbCritical
        Case Else ' 其他错误
            MsgBox "打开工作簿失败！错误编号：" & Err.Number & "，错误描述：" & Err.Description, vbCritical
    End Select
End Sub
```

5.3.2 代码优化技巧

1. **禁用屏幕刷新**：批量操作时禁用屏幕刷新，提升运行速度（结束后恢复）；

```
Application.ScreenUpdating = False ' 禁用屏幕刷新
' 批量操作代码
Application.ScreenUpdating = True ' 恢复屏幕刷新
```

1. **禁用自动计算**：复杂计算时禁用自动计算，结束后恢复；

```
Application.Calculation = xlCalculationManual ' 禁用自动计算
' 代码执行
Application.Calculation = xlCalculationAutomatic ' 恢复自动计算
```

1. **使用变量引用对象**：避免重复书写长对象路径，提升代码可读性和效率；

```
' 不推荐
Workbooks("数据.xlsx").Worksheets("Sheet1").Range("A1") = 10
' 推荐
Dim ws As Worksheet
Set ws = Workbooks("数据.xlsx").Worksheets("Sheet1")
ws.Range("A1") = 10
```

1. **避免使用 Select/Activate**：直接引用对象，而非选中对象后操作；

```
' 不推荐
Worksheets("Sheet1").Activate
Range("A1").Select
Selection.Value = 10
' 推荐
Worksheets("Sheet1").Range("A1") = 10
```

第 6 章：VBA 实用工具与资源

6.1 常用 VBA 代码片段

6.1.1 批量删除空行

```
Sub DeleteBlankRows()
    Dim ws As Worksheet
    Set ws = ActiveSheet
    ' 从最后一行向上删除空行（避免漏删）
    ws.Range("A1:A" & ws.Rows.Count).SpecialCells(xlCellTypeBlanks).EntireRow.Delete
    MsgBox "空行删除完成！"
End Sub
```

6.1.2 批量替换文本

```
Sub BatchReplaceText()
    Dim ws As Worksheet
    Set ws = ActiveSheet
    ' 将工作表中所有“旧文本”替换为“新文本”
    ws.Cells.Replace What:="旧文本", Replacement:="新文本", LookAt:=xlPart
    MsgBox "文本替换完成！"
End Sub
```

6.1.3 导出工作表为 PDF

```
Sub ExportToPDF()
    Dim ws As Worksheet, pdfPath As String
    Set ws = Worksheets("报表")
    pdfPath = "D:\报表.pdf" ' PDF 保存路径
    ' 导出工作表为 PDF（包含所有打印区域）
    ws.ExportAsFixedFormat Type:=xlTypePDF, Filename:=pdfPath,
Quality:=xlQualityStandard
    MsgBox "已导出 PDF 到：" & pdfPath
End Sub
```

6.2 VBA 调试技巧

1. **设置断点**：在代码窗口的行号左侧点击，出现红色圆点，运行到该行时暂停，可逐行查看变量值；
2. **逐行执行**：设置断点后，按 F8 逐行执行代码，观察代码执行流程；
3. **立即窗口输出**：在代码中使用 Debug.Print 变量名，在立即窗口查看变量值；
4. **监视窗口**：右键点击变量，选择【添加监视】，在监视窗口实时查看变量值变化。

6.3 学习资源推荐

- **官方文档**：Microsoft Excel VBA 官方帮助文档（可通过 Office 帮助中心访问）；
- **书籍**：《Excel VBA 实战大全》《Excel 2021 VBA 编程入门到精通》；

- **在线教程**：Excel Home 论坛、B 站 Excel VBA 系列教程；
- **工具**：Rubberduck（VBA 代码分析工具）、VBA Code Cleaner（代码格式化工具）。

后记

Excel VBA 的学习核心在于“实践”，只有将所学知识应用到实际工作中，才能真正掌握其精髓。本书从基础语法到实战案例，再到进阶应用，为你搭建了完整的 VBA 知识体系，但实际工作中的需求千变万化，需要你不断探索和积累。

建议你在学习过程中，结合自身工作场景，尝试编写简单的 VBA 程序解决实际问题，从错误中总结经验，逐步提升代码编写能力。随着学习的深入，你会发现 VBA 不仅能帮你节省大量时间，还能让你在职场中具备更强的竞争力。

祝你在 Excel VBA 的学习之路上不断进步，成为办公自动化高手！

> （注：文档部分内容可能由 AI 生成）