

完全手册 Excel VBA 典型实例大全：通过 368 个例子掌握

目录

第 1 章 宏的应用技巧

宏是一个 VBA 程序，通过宏可以完成枯燥的、频繁的重复性工作。本章的实例分别介绍在 Excel 2003、Excel 2007 中录制宏、使用 Visual Basic 代码创建宏的方法，最后还以实例演示运行宏和编辑宏的方法。

1.1 创建宏 1

例 001 在 Excel 2003 中录制宏 1

例 002 打开 Excel 2007 的录制宏功能 3

例 003 在 Excel 2007 中录制宏 4

例 004 使用 Visual Basic 创建宏 5

1.2 管理宏 6

例 005 运行宏 7

例 006 编辑宏 8

第 2 章 VBE 使用技巧

VBE（Visual Basic Editor）是编写 VBA 代码的工具，在上一章中曾使用 VBE 编辑宏代码。本章的实例介绍了设置 VBE 操作环境、在 VBE 中管理工程代码、使用 VBE 的辅助工具提高代码输入效率等方法。

2.1 设置 VBE 操作环境 10

例 007 停靠 VBE 子窗口 10

例 008 定制 VBE 环境 12

2.2 工程管理 13

例 009 增加模块 13

例 010 删除模块 15

例 011 导出模块 16

例 012 导入模块 17

2.3 管理代码 18

例 013 属性/方法列表 18

例 014 常数列表 19

例 015 参数信息 20

例 016 自动完成关键字 21

第 3 章 程序控制流程技巧

结构化程序设计中使用的基本控制结构有 3 种：顺序结构、选择结构和循环结构。

本章以实例演示了 VBA 中这三种控制结构的控制语句，最后还介绍了在 VBA 中使用数组的方法。

3.1 常用输入/输出语句 23

例 017 九九乘法表（Print 方法的应用） 23

例 018 输入个人信息（Inputbox 函数的应用） 24

例 019 退出确认（Msgbox 函数的应用） 25

3.2 分支结构 27

例 020 突出显示不及格学生 27

例 021 从身份证号码中提取性别 29

例 022 评定成绩等级 30

例 023 计算个人所得税 32

3.3 循环结构 34

例 024 密码验证 34

例 025 求最小公倍数和最大公约数 36

例 026 输出 ASCII 码表 37

例 027 计算选中区域数值之和 39

例 028 换零钱法（多重循环） 40

3.4 使用数组 42

例 029 数据排序 42

例 030 彩票幸运号码 44

例 031 用数组填充单元格区域 46

第 4 章 Range 对象操作技巧

用户在使用 Excel 时，大部分时间都是在操作单元格中的数据，同样地，在 Excel 中使用 VBA 编程时，也需要频繁地引用单元格区域。本章实例介绍用 VBA 引用单元格、获取单元格信息、操作单元格数据、设置单元格格式等内容。

4.1 获取单元格的引用 48

例 032 使用 A1 样式引用单元格 48

例 033 使用索引号引用单元格 49

例 034 引用多个单元格区域 50

例 035 合并单元格区域 51

例 036 引用合并区域的子区域 52

例 037 动态选中单元格区域 53

例 038 引用相对其他单元格的单元格 54

例 039 扩展单元格区域 55

例 040	引用单元格交叉区域	56
例 041	引用当前区域	57
例 042	获取已使用区域	58
例 043	引用区域内的单元格	59
例 044	设置标题行格式	61
例 045	选取条件格式单元格	62
例 046	选择数据列末单元格	63
例 047	获取某列连续数据区域	64
例 048	获取多个不同长度的非连续列	65
例 049	当前单元格的前后单元格	65
例 050	获取三维区域	66
4.2	获取单元格信息	67
例 051	获取标题行和数据行	67
例 052	获取当前区域信息	68
例 053	单元格区域是否有公式	69
例 054	追踪公式单元格	70
例 055	获取单元格地址	71
4.3	操作单元格	72
例 056	合并相同值单元格	72
例 057	删除指定字符后的内容	73
例 058	给单元格设置错误值	75

例 059 活动单元格错误类型 76

例 060 自动设置打印区域 77

例 061 按设置长度换行 77

例 062 选择不含公式的单元格 79

例 063 生成不重复随机数 80

例 064 拆分单元格 82

例 065 添加超链接 83

例 066 删除超链接 84

例 067 限制单元格移动范围 85

例 068 插入批注 86

例 069 隐藏/显示批注 87

例 070 删除批注 87

例 071 复制单元格区域 88

例 072 给单元格设置公式 90

例 073 复制公式 90

例 074 查找并填充空白单元格 91

例 075 清除单元格 92

例 076 删除单元格区域 93

4.4 设置单元格格式 94

例 077 按颜色统计单元格数量 94

例 078 获取单元格底纹和图案 95

例 079 设置页眉为单元格值	96
例 080 设置日期格式	97
例 081 生成大写金额	98
例 082 格式化当前区域的数据	100
例 083 设置自动套用格式	101
例 084 突出显示当前位置	101
例 085 设置边框线	103
例 086 设置文本对齐格式	104
例 087 单元格文本缩排	105
例 088 设置文本方向	107
例 089 设置自动换行格式	108
例 090 设置缩小字体填充	108
例 091 设置条件格式	109
例 092 设置单元格图案	111
例 093 合并单元格	112

第 5 章 Worksheet 对象操作技巧

Worksheet 对象表示 Excel 工作簿中的工作表，Worksheet 对象是 Worksheets 集合的成员。在 VBA 中，通过操作 Worksheet 对象和 Worksheets 集合对象，即可控制 Excel 的工作表。本章实例介绍了用 VBA 代码操作工作表、操作工作表行和列、通过工作表事件控制工作表等内容。

5.1 控制工作表集合 114

例 094 增加工作表	114
-------------	-----

例 095 窗体方式新增工作表	115
-----------------	-----

例 096	窗体方式删除工作表	118
例 097	批量新建工作表	119
例 098	获取工作表数	120
例 099	循环激活工作表	121
例 100	选择工作表	121
例 101	选取前一个工作表/后一个工作表	122
例 102	选中工作表的名称	123
例 103	保护工作表	124
例 104	撤销工作表的保护	126
例 105	判断工作表是否存在	127
例 106	工作表排序	129
例 107	复制工作表	130
例 108	移动工作表	131
例 109	删除工作表	132
例 110	删除空工作表	133
例 111	密码控制删除工作表	134
例 112	隐藏/显示工作表	135
例 113	工作表移至最前/最后	136
例 114	工作表打印页数	137
例 115	重命名工作表	138
例 116	设置工作表标签颜色	140

例 117 导出工作表 141

5.2 操作工作表的行和列 144

例 118 删除空行 144

例 119 插入行 145

例 120 插入多行 146

例 121 插入列 147

例 122 隐藏/显示行 148

例 123 隐藏/显示列 149

例 124 设置行高 149

例 125 设置列宽 151

5.3 操作工作表 152

例 126 合并工作表数据 152

例 127 工作表是否被保护 153

例 128 制作工作表目录 154

例 129 删除图片 155

例 130 修改工作表的代码名 156

5.4 控制工作表事件 158

例 131 为输入数据的单元格添加批注 158

例 132 自动填充相同值 159

例 133 记录同一单元格多次输入值 160

例 134 禁止选中某个区域 161

- 例 135 禁止输入相同数据 162
- 例 136 设置滚动区域 163
- 例 137 自动添加边框线 164
- 例 138 限制在数据区域下一行输入数据 165
- 例 139 增加快捷菜单 166
- 例 140 限制选择其他工作表 168
- 例 141 自动隐藏工作表 169
- 例 142 将原数据作批注 170
- 例 143 输入编码 171

第 6 章 Workbook 对象操作技巧

Workbook 对象表示 Excel 工作簿，Workbooks 集合对象表示 Excel 中所有打开的工作簿。本章实例介绍 VBA 控制工作簿的方法，包括对工作簿集合和工作簿的操作、通过工作簿事件控制工作簿的操作。

6.1 操作工作簿集合 173

- 例 144 批量新建工作簿 173
- 例 145 设置背景音乐 174
- 例 146 打开工作簿 176
- 例 147 保存工作簿 177
- 例 148 更名保存工作簿 178
- 例 149 将工作簿保存为 Web 页 180
- 例 150 打开文本文件 181
- 例 151 设置工作簿密码 182

例 152	保护工作簿	184
例 153	查看文档属性	185
例 154	处理命名单元格区域	187
例 155	判断工作簿是否存在	190
例 156	判断工作簿是否打开	191
例 157	备份工作簿	192
例 158	获取关闭工作簿中的值（方法 1）	194
例 159	获取关闭工作簿中的值（方法 2）	196
例 160	多工作簿数据合并	197
6.2	控制工作簿事件	199
例 161	自动打开关联工作簿	199
例 162	禁止拖动单元格	200
例 163	设置新增工作表为固定名称	201
例 164	退出前强制保存工作簿	202
例 165	限制打印	203
例 166	限制保存工作簿	204
例 167	限制工作簿使用次数	205
例 168	限制工作簿使用时间	207
例 169	设置应用程序标题	207
例 170	根据密码打开工作簿	209
例 171	打开工作簿禁用宏	210

例 172 用 VBA 删除宏代码 212

第 7 章 Application 对象操作技巧

Application 对象代表整个 Excel 应用程序，使用 Application 对象可控制应用程序范围的设置和选项。本章实例介绍使用 VBA，通过 Application 对象自定义 Excel 外观、设置 Excel 操作选项、控制 Excel 应用程序，以及通过 Application 对象的 OnTime 方法和 OnKey 方法响应用户操作的内容。

7.1 自定义 Excel 外观 214

例 173 显示/关闭编辑栏 214

例 174 设置状态栏 215

例 175 控制鼠标指针形状 217

例 176 全屏幕显示 218

例 177 最大化 Excel 窗口 219

例 178 查询计算机信息 219

7.2 设置 Excel 操作选项 220

例 179 关闭屏幕刷新 220

例 180 禁止弹出警告信息 222

例 181 复制/剪切模式 223

例 182 获取系统路径 224

7.3 控制应用程序 225

例 183 激活 Microsoft 应用程序 225

例 184 控制最近使用文档 226

例 185 文件选择器 228

例 186 快速跳转 230

例 187 激活 Excel 2007 的功能区选项卡 232

7.4 Application 对象事件处理 234

例 188 工作表上显示时钟 234

例 189 整点报时 235

例 190 自定义功能键 236

第 8 章 Window 对象操作技巧

Window 对象代表一个窗口，许多工作表特征（如滚动条和标尺）实际上是窗口的属性。本章实例介绍用 VBA 控制窗口的方法，包括通过 Window 对象的属性和方法创建、拆分窗口、设置窗口大小、显示比例、控制窗口显示状态等。

8.1 控制窗口 238

例 191 创建窗口 238

例 192 调整窗口大小 239

例 193 获取窗口状态 242

例 194 拆分窗格 243

例 195 并排比较窗口 244

例 196 排列窗口 245

例 197 窗口显示比例 246

8.2 控制工作表的显示选项 248

例 198 工作簿显示选项 248

例 199 工作表显示选项 249

例 200 工作表网格线 250

例 201 获取指定窗口选中的信息 252

第 9 章 Chart 对象操作技巧

在 Excel 中可以快速简便地创建图表。在程序中，通过 VBA 代码也可方便地创建图表。本章实例介绍用 VBA 创建图表（包括嵌入式图表）、控制图表中的对象、通过图表事件响应用户操作等内容。

9.1 创建图表 254

例 202 创建图表工作表 254

例 203 创建嵌入图表 255

例 204 转换图表类型 257

例 205 删除图表 258

9.2 控制图表对象 260

例 206 获取嵌入图表的名称 260

例 207 获取图表标题信息 260

例 208 获取图例信息 262

例 209 获取图表坐标轴信息 263

例 210 获取图表的系列信息 264

例 211 判断工作表的类型 265

例 212 重排嵌入图表 266

例 213 调整图表的数据源 268

例 214 为图表添加阴影 269

例 215 显示数据标签 271

例 216 将图表保存为图片 273

例 217 设置图表颜色 274

例 218 按值显示颜色 276

例 219 修改嵌入图表外形尺寸 277

例 220 修改图表标题 279

例 221 修改坐标轴 280

例 222 图表插入到 Word 文档 282

9.3 图表事件 283

例 223 激活图表工作表 283

例 224 显示图表各子对象名称 284

例 225 捕获嵌入图表事件 285

第 10 章 用户界面设计技巧

在 Excel 中，用户大部分时间是在工作表中进行操作。在 Excel 中，也可以设计用户窗体，用户直接在窗体上进行操作，而将工作表作为保存数据的地方。本章实例介绍在 VBA 中调用 Excel 内置对话框、在 VBE 中创建自定义窗体等内容。

10.1 使用内置对话框 288

例 226 显示打开对话框（使用 GetOpenFilename 方法） 288

例 227 显示保存文件对话框（使用 GetSaveAsFilename 方法） 290

例 228 显示内置对话框 291

例 229 用 VBA 调用 Excel 2007 功能区功能 293

10.2 创建自定义窗体 294

例 230 制作 Splash 窗口 294

例 231 控制窗体显示 295

例 232 列表框间移动数据 297

例 233 通过窗体向工作表添加数据 302

例 234 制作多页窗体——报名登记 305

例 235 通过窗体设置单元格格式 307

例 236 用窗体控制工作表显示比例 308

例 237 调色板窗体 311

例 238 在窗体中显示图表 312

例 239 制作向导窗体 314

例 240 拖动窗体上的控件 317

例 241 制作交通信号灯 318

例 242 制作进度条 320

第 11 章 命令栏和功能区操作技巧

在 Excel 2007 中，以新的功能区取代了以前版本的命令栏（包括菜单栏和工具栏）。本章实例分别介绍了用 VBA 控制 Excel 2003 以前版本的命令栏、用 XML 自定义 Excel 2007 功能区等内容。

11.1 控制命令栏 322

例 243 显示内置菜单和工具栏的 ID 322

例 244 创建自定义菜单 323

例 245 删除自定义菜单 325

例 246 创建快捷菜单 326

例 247 禁止工作表标签快捷菜单 328

例 248 屏蔽工作表标签部分快捷菜单 329

11.2 Excel 2007 的功能区 330

例 249 创建功能区选项卡 330

例 250 禁用 Office 按钮的菜单 332

例 251 在“Office 按钮”中新建菜单 333

例 252 重定义“Office 按钮”菜单项功能 335

例 253 为内置选项卡增加功能 336

第 12 章 Excel 处理工作表数据技巧

通过 Excel 相关对象可对工作表中的数据进行操作，如处理单元格区域的公式、对数据进行查询、排序、筛选等操作。本章实例介绍了用 VBA 处理公式，对数据进行查询、排序、筛选等内容。

12.1 处理公式 339

例 254 判断单元格是否包含公式 339

例 255 自动填充公式 340

例 256 锁定和隐藏公式 341

例 257 将单元格公式转换为数值 342

例 258 删除所有公式 343

例 259 用 VBA 表示数组公式 345

12.2 数据查询 346

例 260 查找指定的值 346

例 261 带格式查找 349

例 262 查找上一个/下一个数据 349

例 263 代码转换 351

例 264 模糊查询 353

例 265 网上查询快件信息 354

例 266 查询基金信息 357

例 267 查询手机所在地 358

例 268 使用字典查询 360

12.3 数据排序 361

例 269 用 VBA 代码排序 362

例 270 乱序排序 363

例 271 自定义序列排序 364

例 272 多关键字排序 366

例 273 输入数据自动排序 367

例 274 数组排序 369

例 275 使用 Small 和 Large 函数排序 370

例 276 使用 RANK 函数排序 372

例 277 姓名按笔画排序 374

12.4 数据筛选 376

例 278 用 VBA 进行简单筛选 377

例 279 用 VBA 进行高级筛选 378

例 280 筛选非重复值 380

例 281 取消筛选 381

第 13 章 Excel 处理数据库技巧

通过 VBA 代码，可在 Excel 中访问数据库。本章实例介绍通过 ADO 访问 Excel 工作簿中的数据、在 Excel 中处理 Access 数据库（包括获取、添加、修改、删除记录，创建 Acce

ss 数据) 等内容。

13.1 用 ADO 访问 Excel 工作表 383

例 282 使用 ADO 连接数据库 383

例 283 从工作表中查询数据 384

例 284 使用 ADO 导出数据 386

例 285 汇总数据 387

例 286 不打开工作簿获取工作表名称 388

13.2 处理 Access 数据库 390

例 287 从 Access 中获取数据 390

例 288 添加数据到 Access 391

例 289 创建 Access 数据库 393

例 290 是否存在指定表 395

例 291 列出数据库的表名 396

例 292 列出数据表的字段信息 398

例 293 修改记录 399

例 294 删除记录 401

第 14 章 创建加载宏技巧

在 Excel 中, 通过加载宏可以扩展功能, 加载宏是为 Excel 提供自定义命令或自定义功能的补充程序。本章实例介绍了在 Excel 中通过 VBA 代码创建 Excel 加载宏和 COM 加载宏的方法。

14.1 创建加载宏的方法 404

例 295 创建 Excel 加载宏 404

例 296 创建 COM 加载宏 408

例 297 系统加载宏列表 413

14.2 常用加载宏示例 414

例 298 时间提示 414

例 299 大写金额转换 416

例 300 计算个人所得税 418

例 301 加盖公章 419

第 15 章 文件和文件夹操作技巧

通过 VBA 代码可在 Excel 中操作文件。有两种方式访问操作文件和文件夹：一

是使用过程形式的 VB 访问和操作方法，另一种是将文件系统作为对象的文件对象模型方式。本章实例介绍了用 VB 语句操作文件和文件夹、用 FSO 操作文件和文件夹的内容。

15.1 用 VB 语句操作文件和文件夹 422

例 302 显示指定文件夹的文件 422

例 303 判断文件（文件夹）是否存在 424

例 304 新建文件夹 425

例 305 复制文件 426

例 306 重命名文件或文件夹 427

例 307 删除文件 429

例 308 查看文件属性 430

15.2 用 FSO 操作文件和文件夹 432

例 309 判断文件是否存在（FSO） 432

例 310 分离文件名和扩展名 434

例 311 新建和删除文件夹 (FSO) 435

例 312 复制文件 (FSO) 437

例 313 复制文件夹 (FSO) 439

例 314 列出文件夹名称 440

例 315 显示文件属性 441

例 316 删除所有空文件夹 442

例 317 显示驱动器信息 444

第 16 章 文本文件操作技巧

上一章介绍了对文件进行复制、删除之类的操作，更多的时候用户还需要从文本文件中读取数据，或向文本文件中写入数据。本章实例介绍用 VBA 操作文本文件的内容，包括用 VB 语句操作文本文件和用 FSO 操作文本文件。

16.1 用 VB 语句操作文本文件 446

例 318 创建文本文件 446

例 319 读取文本文件数据 448

例 320 工作表保存为文本文件 450

例 321 导出批注到文本文件 451

例 322 从文本文件导入批注 453

16.2 用 FSO 操作文本文件 455

例 323 创建文本文件 (FSO) 455

例 324 添加数据到文本文件 (FSO) 456

例 325 读取文本文件数据 (FSO) 458

第 17 章 用 Excel 控制其他程序技巧

在 Excel 中，通过 VBA 代码可调用其他应用程序的功能。包括创建和打开 Word 文档、创建和打开幻灯片、发送和导入 Outlook 邮件等。另外，使用 VBA 提供的 Shell 函数还可打开 Windows 中的其他程序、打开控制面板对应的选项等。

17.1 控制 Office 应用程序 460

例 326 打开 Word 文档 460

例 327 从 Word 文档中获取数据 462

例 328 生成成绩通知书 464

例 329 在 Excel 中打开 PPT 469

例 330 在 Excel 中创建 PPT 470

例 331 使用 SendMail 发送邮件 473

例 332 用 Outlook 发送邮件 474

例 333 导入 Outlook 中的邮件 476

例 334 保存 Outlook 中的附件 478

17.2 调用其他程序 479

例 335 运行系统自带程序 479

例 336 在 Excel 中打开控制面板 481

第 18 章 VBE 工程实用操作技巧

在第 2 章中介绍了 VBE 的使用方法。其实，VBE 也包含一个对象模型，通过该对象模型可控制 VBA 工程的主要元素。本章实例介绍通过 VBE 对象模型，用 VBA 代码添加或删除模块、创建用户窗体、生成 VBA 代码等内容。

18.1 显示工程相关信息 483

例 337 显示工程信息 483

例 338 列出工程所有组件 485

例 339 显示工作簿中 VBA 的过程名 486

例 340 导出 VBA 过程代码 488

例 341 列出工程引用的外部库 489

18.2 动态创建组件 490

例 342 重命名组件 490

例 343 导出/导入模块代码 492

例 344 删除指定子过程代码 494

例 345 查找代码 496

例 346 增加模块 498

例 347 增加类模块 500

例 348 控制 VBE 的子窗口 501

例 349 工作表中动态增加按钮 502

例 350 创建动态用户窗体 504

第 19 章 VBA 程序调试优化技巧

Excel 应用程序的顺利完成，调试的过程是非常重要的。本章实例介绍 Excel VBA 调试程序的基本方法和 VBA 程序的优化技巧。

19.1 VBA 程序调试技巧 507

例 351 设置断点 507

例 352 使用本地窗口 508

例 353 使用立即窗口 510

例 354 单步执行 510

例 355 运行选定部分代码 512

例 356 调用堆栈 513

例 357 使用监视窗口 514

例 358 使用条件编译 515

例 359 错误处理语句 517

19.2 VBA 程序优化技巧 518

例 360 使用 VBA 已有功能 518

例 361 避免使用变体 520

例 362 使用对象变量 521

例 363 使用数组处理单元格 522

例 364 检查字符串是否为空 522

例 365 优化循环体 523

例 366 使用 For Each 循环 525

例 367 关闭屏幕刷新 526

例 368 使用内置函数 527

VBA 应用程序由一系列的 VBA 代码组成，这些代码将按照一定的顺序执行。有时程序根据一定的条件只能执行某一部分代码，有时需要重复执行某一段代码。通过程序结构控制代码来完成这些功能，本章介绍这些程序控制流程方面的技巧。

3.1 常用输入/输出语句

结构化程序设计中使用的基本控制结构有 3 种：顺序结构、选择结构和循环结构。顺序结构就是按照语句的书写顺序从上到下、逐条语句地执行。执行时，编写在前面的代码先执行，编写在后面的代码后执行。这是最普遍的结构形式，也是后面两种结构的基础。

顺序结构不需要使用结构控制语句，本节介绍常用的输入输出语句的技巧。

例 017 九九乘法表（Print 方法的应用）

1. 案例说明

在早期的 Basic 版本中，程序运行结果主要依靠 Print 语句输出到终端。在 VB 中，Print 作为窗体的一个方法，用来在窗体中显示信息。但是在 VBA 中，用户窗体已经不支持 Print 方法了。

在 VBA 中，Print 方法只能向“立即窗口”中输出程序的运行中间结果，供开发人员调试程序时使用。

本例使用 Print 方法在立即窗口中输入九九乘法表。

2. 关键技术

在 VBA 中，Print 方法只能应用于 Debug 对象，其语法格式如下：

Debug.Print [outputlist]

参数 outputlist 是要打印的表达式或表达式的列表。如果省略，则打印一个空白行。

- Print 首先计算表达式的值，然后输出计算的结果。在 outputlist 参数中还可以使用分隔符，以格式化输出的数据。格式化分隔符有以下几种：
- Spc(n)：插入 n 个空格到输出数据之间；
- Tab(n)：移动光标到适当位置，n 为移动的列数；

- 分号：表示前后两个数据项连在一起输出；
- 逗号：以 14 个字符为一个输出区，每个数据输出到对应的输出区。

3. 编写代码

(1) 在 VBE 中，单击菜单“插入/模块”命令插入一个模块。

(2) 在模块中输入以下代码：

```
Sub multi()  
  
    For i = 1 To 9  
  
        For j = 1 To i  
  
            Debug.Print i; "x"; j; "="; i * j; " ";  
  
        Next  
  
        Debug.Print                                '换行  
  
    Next  
  
End Sub
```

(3) 按功能键“F5”运行子过程，在“立即窗口”输出九九乘法表，如图 3-1 所示。



图 3-1 立即窗口

例 018 输入个人信息（Inputbox函数的应用）

1. 案例说明

本例演示 Inputbox 函数的使用方法。执行程序，将弹出“输入个人信息”对话框，要

求用户输入“姓名、年龄、地址”信息，然后在“立即窗口”中将这些信息打印输出。

2. 关键技术

为了实现数据输入，VBA 提供了 `InputBox` 函数。该函数将打开一个对话框作为输入数据的界面，等待用户输入数据，并返回所输入的内容。其语法格式如下：

`InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])`

各参数的含义如下：

- **Prompt:** 为对话框消息出现的字符串表达式。其最大长度为 1024 个字符。如果需要在对话框中显示多行数据，则可在各行之间用回车符换行符来分隔，一般使用 VBA 的常数 `vbCrLf` 代表回车换行符。
- **Title:** 为对话框标题栏中的字符串。如果省略该参数，则把应用程序名放入标题栏中。
- **Default:** 为显示在文本框中的字符串。如果省略该参数，则文本框为空。
- **Xpos:** 应和 **Ypos** 成对出现，指定对话框的左边与屏幕左边的水平距离。如果省略该参数，则对话框会在水平方向居中。
- **Ypos:** 应和 **Xpos** 成对出现，指定对话框的上边与屏幕上边的距离。如果省略该参数，则对话框被放置在屏幕垂直方向距下边大约三分之一的位置。
- **Helpfile:** 设置对话框的帮助文件，可省略。
- **Context:** 设置对话框的帮助主题编号，可省略。

3. 编写代码

(1) 在 VBE 中，单击菜单“插入/模块”命令插入一个模块。

(2) 在模块中输入以下代码：

```
Sub inputinfo()
```

```
    Title = "输入个人信息"
```

```
    name1 = "请输入姓名："
```

```

age1 = "请输入年龄： "

address1 = "请输入地址： "

strName = InputBox(name1, Title)

age = InputBox(age1, Title)

Address = InputBox(address1, Title)

Debug.Print "姓名： "; strName

Debug.Print "年龄： "; age

Debug.Print "地址： "; Address

End Sub

```

（3）按功能键“F5”运行子过程，将弹出“输入个人信息”窗口，如图 3-2 所示。在对话框中输入内容后按“回车”，或单击“确定”按钮。

（4）接着输入“年龄”和“地址”信息，在“立即窗口”中将输出这些内容，如图 3-3 所示。



图 3-2 输入个人信息



图 3-3 输出结果

例 019 退出确认（Msgbox函数的应用）

1. 案例说明

在应用程序中，有时用户会由于误操作关闭 Excel，为了防止这种情况，可在退出 Excel 之前弹出对话框，让用户确认是否真的要关闭 Excel。

本例使用 MsgBox 函数弹出对话框，让用户选择是否退出系统。

2. 关键技术

使用 MsgBox 函数可打开一个对话框，在对话框中显示一个提示信息，并让用户单击对话框中的按钮，使程序继续执行。

MsgBox 函数语法格式如下：

Value=MsgBox(prompt[,buttons][,title][,helpfile,context])

通过函数返回值可获得用户单击的按钮，并可根据按钮的不同而选择不同的程序段来执行。

该函数共有 5 个参数，除第 1 个参数外，其余参数都可省略。各参数的意义与 Inputbox 函数参数的意义基本相同，不同的地方是多了一个 buttons 参数，用来指定显示按钮的数目及形式、使用提示图标样式、默认按钮以及消息框的强制响应等。其常数值如表 3-1 所示。

表 3-1 按钮常数值

常 量	值	说 明
vbOkOnly	0	只显示“确定”(Ok) 按钮
vbOkCancel	1	显示“确定”(Ok) 及“取消”(Cancel) 按钮
vbAbortRetryIgnore	2	显示“异常终止”(Abort)、“重试”(Retry) 及“忽略”(Ignore) 按钮
vbYesNoCancel	3	显示“是”(Yes)、“否”(No) 及“取消”(Cancel) 按钮

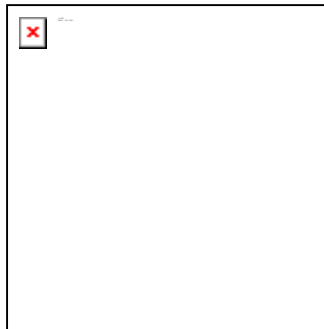
续表

常 量	值	说 明
vbYesNo	4	显示“是”(Yes) 及“否”(No) 按钮
vbRetryCancel	5	显示“重试”(Retry) 及“取消”(Cancel) 按钮
vbCritical	16	显示 Critical Message 图标 
vbQuestion	32	显示 Warning Query 图标 
vbExclamation	48	显示 Warning Message 图标 
vbInformation	64	显示 Information Message 图标 
vbDefaultButton1	0	以第一个按钮为默认按钮
vbDefaultButton2	256	以第二个按钮为默认按钮
vbDefaultButton3	512	以第三个按钮为默认按钮
vbDefaultButton4	768	以第四个按钮为默认按钮
vbApplicationModal	0	进入该消息框，当前应用程序暂停
vbSystemModal	4096	进入该消息框，所有应用程序暂停

表 3-1 中的数值（或常数）可分为四组，其作用分别为：

— 第一组值（0～5）用来决定对话框中按钮的类型与数量。

- 第二组值（16，32，48，64）用来决定对话框中显示的图标。
- 第三组值（0，256，512）设置对话框的默认活动按钮。活动按钮中文字的周转有虚线，按回车键可执行该按钮的单击事件代码。
- 第四组值（0，4096）决定消息框的强制响应性。



buttons 参数可由上面 4 组数值组成，其组成原则是：从每一类中选择一个值，把这几个值累加在一起就是 buttons 参数的值（大部分时间里都只使用前三组数值的组合），不同的组合可得到不同的结果。

3. 编写代码

（1）在 VBE 中，双击“工程”子窗口中的“ThisWorkbook”打开代码窗口，如图 3-4 所示。

（2）在代码窗口左上方的对象列表中选择“Workbook”，如图 3-5 所示。

（3）在代码窗口右上方的事件列表中选择“BeforeClose”，如图 3-6 所示。代码窗口中将自动生成事件过程结构如下：

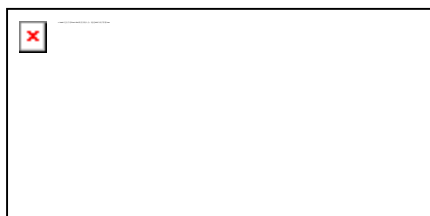


图 3-5 对象列表



图 3-6 事件列表

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
```

```
End Sub
```

（4）在上面生成的事件过程中输入以下代码：

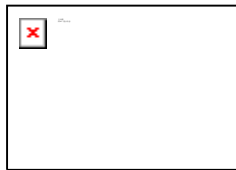
```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
```

```
Dim intReturn As Integer
```

```
intReturn = MsgBox("真的退出系统吗？", vbYesNo + vbQuestion, "提示")
```

```
If intReturn <> vbYes Then Cancel = True
```

```
End Sub
```



(5) 保存 Excel 工作簿。

(6) 关闭 Excel 工作簿时，将弹出如图 3-7 所示的对话框。单击“是”按钮将退出 Excel，单击“否”按钮将返回 Excel 工作簿。

3.2 分支结构

分支结构，又叫选择结构。这种结构的程序将根据给定的条件来决定执行哪一部分代码，而跳过其他代码。

例 020 突出显示不及格学生

1. 案例说明

本例判断学生成绩表中的成绩，如果成绩不及格（低于 60 分），则将该成绩着重显示出来。如图 3-8 所示（左图为原成绩，右图突出显示不及格成绩）。

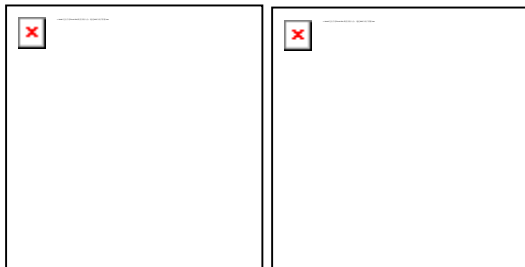


图 3-8 突出显示不及格学生

2. 关键技术

在本例中，需要进行一个判断（成绩是否低于 60 分），这时可使用 If...Then 语句。用 If...Then 语句可有条件地执行一个或多个语句。其语法格式如下：

```
If 逻辑表达式 Then
```

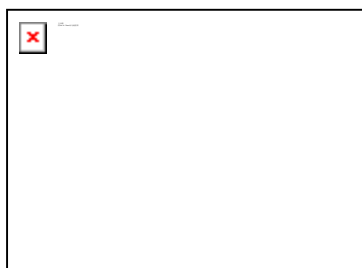
```
    语句 1
```

```
    语句 1
```

```
... ..
```

语句 n

End If



逻辑表达式也可以是任何计算数值的表达式，VBA 将为零（0）的数值看做 **False**，而任何非零数值都被看做 **True**。

该语句的功能为：若逻辑表达式的值是 **True**，则执行位于 **Then** 与 **End If** 之间的语句；若逻辑表达式的值是 **False**，则不执行 **Then** 与 **End If** 之间的语句，而执行 **End If** 后面的语句。其流程图如图 3-9 所示。

If...Then 结构还有一种更简单的形式：单行结构条件语句。其语法格式如下：

If 逻辑表达式 Then 语句

该语句的功能为：若逻辑表达式的值是 **True**，则执行 **Then** 后的语句；若逻辑表达式的值是 **False**，则不执行 **Then** 后的语句，而执行下一条语句。

3. 编写代码

- (1) 打开“学生成绩表”。
- (2) 按快捷键“Alt+F11”进入 VBE 环境。
- (3) 单击菜单“插入/模块”命令向工程中插入一个模块，并编写以下代码：

Sub 显示不及格学生()

Dim i As Integer

For i = 3 To 11

If Sheets(1).Cells(i, 2).Value < 60 Then

Sheets(1).Cells(i, 2).Select

Selection.Font.FontStyle = "加粗"

Selection.Font.ColorIndex = 3

End If

Next

End Sub

(4) 关闭 VBE 开发环境返回 Excel。

(5) 在功能区“开发工具”选项卡的“控件”组中，单击“插入”按钮弹出“表单控件”面板，如图 3-10 所示。

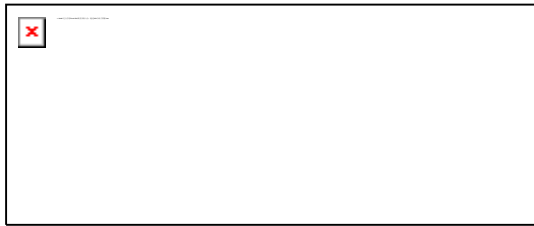


图 3-10 插入按钮

(6) 在“表单控件”面板中单击“按钮”，拖动鼠标在工作表中绘制一个按钮。当松开鼠标时，将弹出“指定宏”对话框，如图 3-11 所示。

(7) 在“指定宏”对话框中，单击选中“显示不及格学生”宏，单击“确定”按钮。

(8) 右击工作表中的按钮，弹出快捷菜单如图 3-12 所示，单击“编辑文字”菜单，修改按钮中的提示文字为“显示不及格学生”。

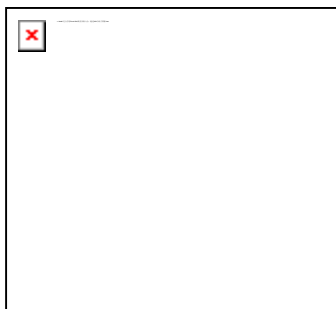


图 3-11 指定宏



图 3-12 编辑文字

(9) 单击“显示不及格学生”按钮，执行宏代码，成绩表中不及格成绩将突出显示为粗体、红色，如图 3-13 所示。

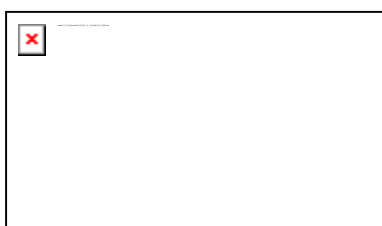


图 3-13 执行程序

例 021 从身份证号码中提取性别

1. 案例说明

在很多信息系统中都需要使用到身份证号码，身份证号码中包含有很多信息，如可从其中提取性别。我国现行使用的身份证号码有两种编码规则，即 15 位居民身份证和 18 位居民身份证。

15 位的身份证号的编码规则。

dddddd yymmdd xx p

18 位的身份证号的编码规则。

dddddd yyyymmdd xx p y

其中：

- dddddd 为地址码（省地县三级）18 位中的和 15 位中的不完全相同。
- yyyymmdd yymmdd 为出生年月日。
- xx 序号类编码。
- p 性别。
- 18 位中末尾的 y 为校验码。

2. 关键技术

在 If...Then 语句中，条件不成立时不执行任何语句。在很多时候需要根据条件是否成立分别执行两段不同的代码，这时可用 If...Then...Else 语句，其语法格式如下：

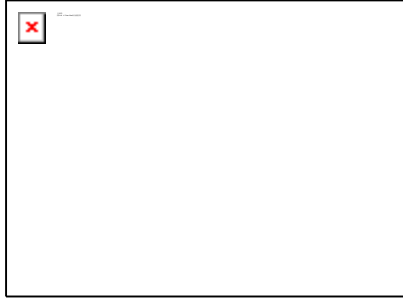
If 逻辑表达式 Then

语句序列 1

Else

语句序列 2

End If



VBA 判断“逻辑表达式”的值，如果它为 **True**，将执行“语句序列 1”中的各条语句，当“逻辑表达式”的值为 **False** 时，就执行“语句序列 2”中的各条语句。其流程图如图 3-14 所示。

3. 编写代码

(1) 新建 Excel 工作簿，在 VBE 中插入一个模块。

(2) 在模块中编写以下代码：

Sub 根据身份证号码确定性别()

 sid = InputBox("请输入身份证号码：")

 i = Len(sid)

 If i <> 15 And i <> 18 Then '判断身份证号长度是否正确

 MsgBox "身份证号码只能为 15 位或 18 位！"

 Exit Sub

End If

 If i = 15 Then '长度为 15 位

 s = Right(sid, 1) '取最右侧的数字

 Else '长度为 18 度

 s = Mid(sid, 17, 1) '取倒数第 2 位数

End If

 If Int(s / 2) = s / 2 Then '为偶数

 sex = "女"

 Else

```
sex = "男"
```

```
End If
```

```
MsgBox "性别: " + sex
```

```
End Sub
```

(3) 切换到 Excel 环境，添加一个按钮“从身份证号码提取性别”，并指定执行上步创建的宏。

(4) 单击“从身份证号码提取性别”按钮，弹出如图 3-15 所示对话框。

(5) 输入身份证号码后单击“确定”按钮，将在如图 3-16 所示对话框中显示性别。

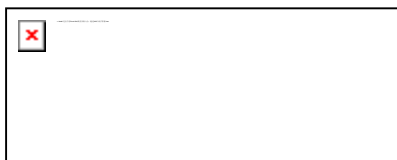


图 3-15 输入身份证号码



图 3-16 显示性别

例 022 评定成绩等级

1. 案例说明

本例将成绩表中的百分制成绩按一定规则划分为 A、B、C、D、E 五个等级，如图 3-17 所示。

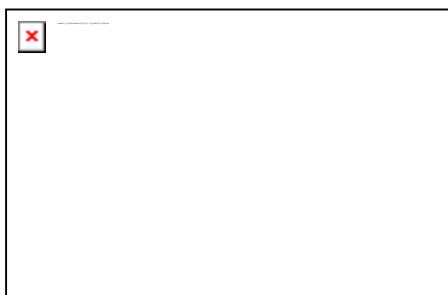


图 3-17 评定成绩等级

其中各等级对应的成绩分别为：

— A：大于等于 90 分；

- B: 大于等于 80 分, 小于 90 分;
- C: 大于等于 70 分, 小于 80 分;
- D: 大于等于 60 分, 小于 70 分;
- E: 小于 60 分。

2. 关键技术

本例共有五个分支, 使用 If...Then...Else 这种二路分支结构也可完成, 但需要复杂的嵌套结构才能解决该问题。其实 VBA 中提供了一种 If...Then...Elseif 的多分支结构, 其语法格式如下:

```
If 逻辑表达式 1 Then  
  
    语句序列 1  
  
Elseif 逻辑表达式 2 Then  
  
    语句序列 2.  
  
Elseif 逻辑表达式 3 Then  
  
    语句序列 3  
  
    ... ..  
  
Else  
  
    语句序列 n  
  
End If
```

在以上结构中, 可以包括任意数量的 Elseif 子句和条件, Elseif 子句总是出现在 Else 子句之前。

VBA 首先判断“逻辑表达式 1”的值。如果它为 False, 再判断“逻辑表达式 2”的值, 依此类推, 当找到一个为 True 的条件, 就会执行相应的语句块, 然后执行 End If 后面的代码。如果所有“逻辑表达式”都为 False, 且包含 Else 语句块, 则执行 Else 语句块。其流程图如图 3-18 所示。

图 3-18 If...Then...Elseif 语句流程图

3. 编写代码

- (1) 在 Excel 中打开成绩表。
- (2) 按快捷键“Alt+F11”进入 VBE 开发环境。
- (3) 单击“插入/模块”命令向工程中插入一个模块，并编写以下 VBA 代码：

```
Sub 评定等级()
```

```
    Dim i As Integer
```

```
    For i = 3 To 11
```

```
        t = Sheets(1).Cells(i, 2).Value '取得成绩
```

```
        If t >= 90 Then
```

```
            j = "A"
```

```
        ElseIf t >= 80 Then
```

```
            j = "B"
```

```
        ElseIf t >= 70 Then
```

```
            j = "C"
```

```
        ElseIf t >= 60 Then
```

```
            j = "D"
```

```
        Else
```

```
            j = "E"
```

```
        End If
```

```
        Sheets(1).Cells(i, 3) = j
```

Next

End Sub

(4) 返回 Excel 操作界面，在成绩表旁边增加一个按钮，并指定执行宏“评定等级”。

(5) 单击“评定等级”按钮，即可在成绩表的 C 列显示出各成绩对应的等级，如图 3-17 所示。

例 023 计算个人所得税

1. 案例说明

在工资管理系统中，需要计算员工应缴纳的个人所得税。个人所得税税额按 5%至 45% 的九级超额累进税率计算应缴税额，税率表如图 3-19 所示。

个人所得税的计算公式为：

应纳个人所得税税额=应纳税所得额×适用税率-速算扣除数

本例根据工资表中的相应数据计算出应纳税额，并填充在工资表对应的列中。

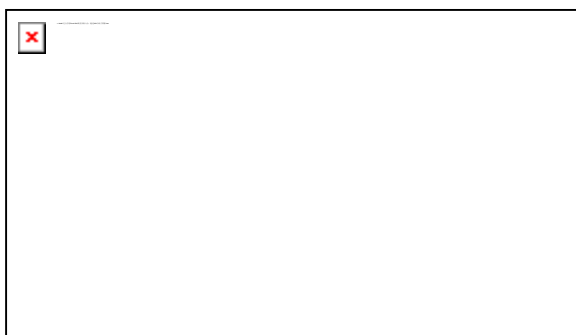


图 3-19 个人所得税税率表

2. 关键技术

本例中计算个人所得税时共有九个分支。这时可在 If...Then...ElseIf 结构中添加多个 ElseIf 块来进行各分支的处理。对于多分支结构，可使用 Select Case 语句。Select Case 语句的功能与 If...Then...Else 语句类似，但在多分支结构中，使用 Select Case 语句可使代码简洁易读。

Select Case 结构的语法格式如下：

Select Case 测试表达式

Case 表达式列表 1

语句序列 1

Case 表达式列表 2

语句序列 2

... ..

Case Else

语句序列 n

End Select

在以上结构中，首先计算出“测试表达式”的值，然后，VBA 将表达式的值与结构中的每个 Case 的值进行比较。如果相等，就执行与该 Case 语句下面的语句块，执行完毕再跳转到 End Select 语句后执行。其流程图如图 3-20 所示。

图 3-20 Select Case 语句流程图

在 Select Case 结构中，“测试表达式”通常是一个数值型或字符型的变量。“表达式列表”可以是一个或几个值的列表。如果在一个列表中有多个值，需要用逗号将各值分隔开。表达式列表可以按以下几种情况进行书写：

- 表达式：表示一些具体的取值。例如：Case 10,15,25。
- 表达式 A To 表达式 B：表示一个数据范围。例如，Case 7 To 17 表示 7~17 之间的值。
- Is 比较运算符表达式：表示一个范围。例如，Case Is>60 表示所有大于 90 的值。
- 以上三种情况的混合。例如，Case 4 To 10, 15, Is>20。

3. 编写代码

- (1) 在 Excel 中打开工资表工作簿。
- (2) 按快捷键“Alt+F11”进入 VBE 开发环境。

(3) 单击菜单“插入/模块”命令插入一个模块。

(4) 在模块中编写以下函数，用来计算所得税：

Function 个人所得税(curP As Currency)

Dim curT As Currency

curP = curP - 1600 '1600 为扣除数

If curP > 0 Then

Select Case curP

Case Is <= 500

curT = curP * 0.05

Case Is <= 2000

curT = (curP - 500) * 0.1 + 25

Case Is <= 5000

curT = (curP - 2000) * 0.15 + 125

Case Is <= 20000

curT = (curP - 5000) * 0.2 + 375

Case Is <= 40000

curT = (curP - 20000) * 0.25 + 1375

Case Is < 60000

curT = (curP - 40000) * 0.3 + 3375

Case Is < 80000

curT = (curP - 60000) * 0.35 + 6375

Case Is < 100000

curT = (curP - 80000) * 0.4 + 10375

Case Else

curT = (curP - 100000) * 0.45 + 15375

End Select

个人所得税 = curT

Else

个人所得税 = 0

End If

End Function

(5) 在模块中编写“计算”子过程，计算工资表中每个员工应缴所得税额，并填写在对应的列中。

Sub 计算()

For i = 4 To 9

Sheets(1).Cells(i, 8).Value = 个人所得税(Sheets(1).Cells(i, 6).Value)

Next

End Sub

(6) 返回到 Excel 环境中，在工资表下方插入一个按钮，为按钮指定宏为“计算”。

(7) 单击“计算”按钮，可计算出每个员工的所得税额，如图 3-21 所示。

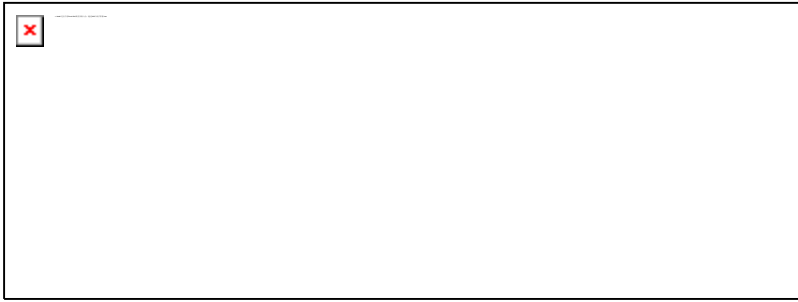


图 3-21 计算所得税

3.3 循环结构

在实际开发的应用系统中,经常需要重复执行一条或多条语句。这种结构称为循环结构。循环结构的思想是利用计算机高速处理运算的特性,重复执行某一部分代码,以完成大量有规则的重复性运算。

VBA 提供了多个循环结构控制语句: Do...Loop 结构、While...Wend 结构、For...Next 结构、For Each...Next 结构。

例 024 密码验证

1. 案例说明

在信息管理系统中,很多时候都需要用户进行登录操作。在登录操作时要求用户输入密码,一般都要给用户三次机会,每次的输入过程和判断过程都相同。

本例使用 Do...Loop 循环完成密码验证过程。

2. 关键技术

在 VBA 中，最常用的循环语句是 Do...Loop 循环。循环结构 Do While...Loop 的语法格式如下：

Do While 逻辑表达式

语句序列 1

[Exit Do]

[语句序列 2]

Loop

其中 Do While 和 Loop 为关键字，在 Do While 和 Loop 之间的语句称为循环体。

当 VBA 执行这个 Do 循环时，首先判断“逻辑表达式”的值，如果为 False（或零），则跳过所有语句，执行 Loop 的下一条语句，如果为 True（或非零），则执行循环体，当执行到 Loop 语句后，又跳回到 Do While 语句再次判断条件。在循环体中如果包含有 Exit Do 语句，当执行到 Exit Do 语句，马上跳出循环，执行 Loop 的下一条语句。其流程图如图 3-22 所示。

图 3-22 Do While...Loop 流程图

VBA 的 Do...Loop 循环有 4 种结构，分别如下：

- Do While...Loop 循环：先测试条件，如果条件成立则执行循环体。
- Do...Loop While 循环：先执行一遍循环体，再测试循环条件，如果条件成立则执行循环体。
- Do Until...Loop 循环：先测试条件，如果条件不成立则执行循环体。
- Do...Loop Until 循环：先执行一遍循环体，再测试循环条件，如果条件不成立则执行循环体。

3. 编写代码

（1）新建 Excel 工作簿，按快捷键“Alt+F11”进入 VBE 开发环境。

（2）单击菜单“插入/模块”命令向工程中插入一个模块。

（3）在模块中编写以下 VBA 代码：

```

Sub login()

    Dim strPassword As String    '保存密码

    Dim i As Integer             '输入密码的次数

    Do

        strPassword = InputBox("请输入密码") '输入密码

        If strPassword = "test" Then '判断密码是否正确

            Exit Do                '退出循环

        Else

            MsgBox ("请输入正确的密码！")

        End If

        i = i + 1

    Loop While i < 3

    If i >= 3 Then '超过正常输入密码次数

        MsgBox "非法用户，系统将退出！"

        Application.Quit

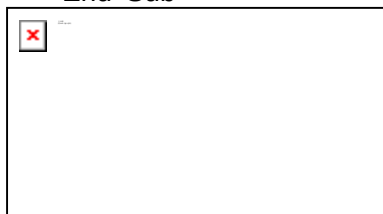
    Else

        MsgBox "欢迎你使用本系统！"

    End If

End Sub

```



(4)返回 Excel 操作界面,在工作表中插入一个按钮,设置提示文字为“密码验证”,并为该按钮指定执行的宏为“login”。

(5)单击“密码验证”按钮,弹出如图 3-23 所示对话框,输入密码后单击“确定”按钮进行密码的验证。

例 025 求最小公倍数和最大公约数

1. 案例说明

几个数公有的倍数叫做这几个数的公倍数，其中最小的一个叫做这几个数的最小公倍数。如 12、18、20 这三个数的最小公倍数为 180。

最大公约数是指某几个整数的共有公约数中最大的那个数。如 2、4、6 这三个数的最大公约数为 2。

本例使用辗转相除法求两个自然数 m 、 n 的最大公约数和最小公倍数。

2. 关键技术

本例首先求出两数 m 、 n 的最大公约数，再将 m 、 n 数的乘积除以最大公约数，即可得到最小公倍数。求最大公约数的算法流程图如图 3-24 所示。

图 3-24 最大公约数算法流程图

本例使用 Do...Loop 循环，并且没有设置循环条件。一般情况下，这种循环是一个死循环（也就是说程序将一直循环下去），因此，在这种循环结构中必须添加一个判断语句，当达到指定的条件时退出循环。如本例中使用以下语句退出循环：

```
If r = 0 Then Exit Do
```

3. 编写代码

（1）新建 Excel 工作簿，按快捷键“Alt+F11”进入 VBE 环境。

（2）单击菜单“插入/模块”命令向工程中插入一个模块。

（3）在模块中编写以下子过程：

```
Sub 最小公倍数和最大公约数()
```

```
    Dim m As Integer, n As Integer
```

```
    Dim m1 As Integer, n1 As Integer
```

```
    Dim t As Integer
```

```
    m = InputBox("输入自然数 m: ")
```

```
    n = InputBox("输入自然数 n: ")
```

```
    m1 = m
```

```
    n1 = n
```

```
    If m1 < n1 Then
```

```
        m1 = n
```

```

n1 = m '交换 m 和 n 的值

End If

Do

    r = m1 Mod n1

    If r = 0 Then Exit Do

    m1 = n1

    n1 = r

Loop

str1 = m & "," & n & "的最大公约数=" & n1 & vbCrLf

str1 = str1 & "最小公倍数=" & m * n / n1

MsgBox str1

End Sub

```

- (4) 返回 Excel 操作环境，向工作表中插入一个按钮，为按钮指定执行上步创建的宏。
- (5) 单击按钮，弹出如图 3-25 所示的输入提示框，分别输入两个数后，得到如图 3-26 所示的结果。



图 3-25 输入数据

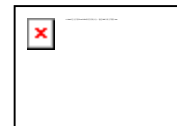
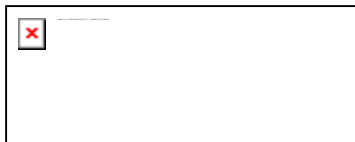


图 3-26 最大公约数和最小公倍数

数

例 026 输出ASCII码表

1. 案例说明

目前计算机中用得最广泛的字符集及其编码，是由美国国家标准局（ANSI）制定的 ASCII 码。ASCII 码由 8 位二进制组成，一共可包含 256 个符号。本例使用循环语句输出 ASCII 中的可见字符，如图 3-27 所示。

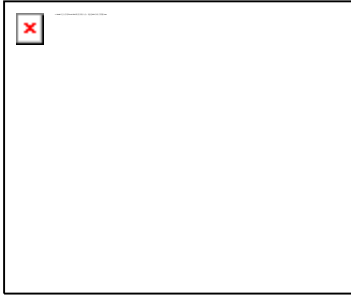


图 3-27 ASCII 码表

2. 关键技术

使用 Do...Loop 循环时，可以不知道循环的具体次数。如果知道循环的次数，可以使用 For...Next 循环语句来执行循环。For 循环的语法如下：

For 循环变量=初始值 To 终值 [Step 步长值]

语句序列 1

[Exit For]

[语句序列 2]

Next [循环变量]

在 For 循环中使用循环变量来控制循环，每重复一次循环之后，循环变量的值将与步长值相加。步长值可正可负，如果步长值为正，则初始值必须小于等于终值，才执行循环体，否则退出循环。如果步长值为负，则初始值必须大于等于终值，这样才能执行循环体。如果没有设置 Step，则步长值默认为 1。For...Next 循环结构的流程图如图 3-28 所示。

For 循环一般都可计算出循环体的执行次数，计算公式如下：

循环次数=[(终值-初值)/步长值]+1

这里用中括号表示取整。

在事先不知道循环体需要执行多少次时，应该用 Do 循环。而在知道循环体要执行的次数时，最好使用 For...Next 循环。

图 3-28 For...Next 流程图

3. 编写代码

(1) 新建 Excel 工作簿，按快捷键“Alt+F11”进入 VBE 环境。

(2) 单击菜单“插入/模块”命令向工程中插入一个模块。

(3) 在模块中编写以下子过程：

```
Sub ascii()  
  
    Dim a As Integer, i As Integer  
  
    i = 3  
  
    For a = 32 To 126  
  
        Sheets(1).Cells(i, 1) = a  
  
        Sheets(1).Cells(i, 2) = Chr(a)  
  
        i = i + 1  
  
    Next  
  
End Sub
```

(4) 返回 Excel 操作环境，向工作表中插入一个按钮，为按钮指定执行上步创建的宏。

(5) 单击按钮，得到如图 3-27 所示的结果。

例 027 计算选中区域数值之和

1. 案例说明

在某些情况下，需要统计工作表中选定区域数值单元格的数值之和（例如，临时查看应发奖金之和），在 Excel 的状态栏就可查看选中单元格的数值之和。本例编写 VBA 代码，使用循环结构来完成该项功能。

2. 关键技术

用户在 Excel 工作表中选定单元格的数量是不固定的，若需统计所选单元格数值之和，这时可使用 For Each 循环来进行处理，对选中区域的每个单元格进行判断，然后再累加数值单元格的值。

For Each...Next 循环语句的语法格式如下：

For Each 元素 In 对象集合

[语句序列 1]

[Exit For]

[语句序列 2]

Next

使用 **For Each** 循环结构，可在对象集合每个元素中执行一次循环体。如果集合中至少有一个元素，就会进入 **For Each** 循环体执行。一旦进入循环，便先针对“对象集合”中第一个元素执行循环中的所有语句。如果“对象集合”中还有其他的元素，则会针对它们执行循环中的语句，当“对象集合”中的所有元素都执行完了，便会退出循环，然后从 **Next** 语句之后的语句继续执行。

在循环体中可以放置任意多个 **Exit For** 语句，随时退出循环。**Exit For** 经常在条件判断之后使用，例如 **If...Then**，并将控制权转移到紧接在 **Next** 之后的语句。

3. 编写代码

(1) 新建 Excel 工作簿，按快捷键“Alt+F11”进入 VBE 环境。

(2) 单击菜单“插入/模块”命令向工程中插入一个模块。

(3) 在模块中编写以下子过程：

Sub 求和()

Dim r

Dim t As Long

For Each r In Selection

If IsNumeric(r.Value) Then

t = t + r.Value

End If

Next

MsgBox "所选区域数值之和为：" & t

End Sub

(4) 返回 Excel 操作环境，向工作表中插入一个按钮，修改按钮的提示字符为“求和”，为按钮指定执行上步创建的宏“求和”。

(5) 在工作表“Sheet1”中输入数据，如图 3-29 左图所示。

(6) 拖动鼠标选中如图 3-29 左图所示数据区域，单击“求和”按钮，求和结果将显示在如图 3-29 右图所示对话框中。

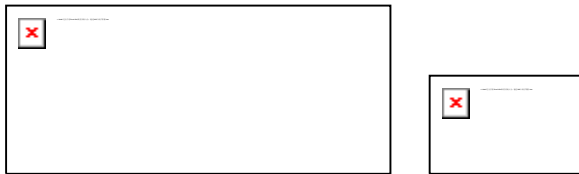


图 3-29 计算选中区域数值之和

例 028 换零钱法（多重循环）

1. 案例说明

将十元钱换成 1 角、2 角、5 角、1 元、2 元、5 元的零钱若干，求出一共有多少种方法进行计算？

2. 关键技术

在 VBA 中，循环结构内的循环体又可以是循环结构，这种情况称为循环的嵌套。VBA 允许在同一过程里嵌套多种类型的循环。

在编写嵌套循环程序的代码时，一定要注意每个循环语句的配对情况。如图 3-30 所示，其中左图是正确的嵌套关系，第一个 Next 关闭了内层的 For 循环，而最后一个 Loop 关闭了外层的 Do 循环。同样，在嵌套的 If 语句中，End If 语句自动与最靠近的前一个 If 语句配对。嵌套的 Do...Loop 结构的工作方式也是一样的，最内圈的 Loop 语句与最内圈的 Do 语句匹配。图 3-30 右图则是错误的嵌套关系。

语句序列 2

图 3-30 正确的嵌套（左）与错误的嵌套（右）

3. 编写代码

- (1) 新建 Excel 工作簿，按快捷键“Alt+F11”进入 VBE 环境。
- (2) 单击菜单“插入/模块”命令向工程中插入一个模块。
- (3) 零钱换法最简单的算法是：使用多重循环，将 10 元钱能换成的各种可能都考虑进去（如 10 可换为 100 个 1 角，可换为 50 个 2 角，等等）。根据这种算法在模块中编写以下子过程：

```
Sub 换零钱 1()  
  
    Dim t As Integer  
  
    For i = 0 To 100                '1 角  
  
        For j = 0 To 50            '2 角  
  
            For k = 0 To 20        '5 角  
  
                For l = 0 To 10    '1 元  
  
                    For m = 0 To 5    '2 元  
  
                        For n = 0 To 2    '5 元  
  
                            If i + 2 * j + 5 * k + 10 * l + 20 * m + 50 * n = 100 Then  
  
                                t = t + 1  
  
                                Sheets(1).Cells(t + 1, 1) = i  
  
                                Sheets(1).Cells(t + 1, 2) = j  
  
                                Sheets(1).Cells(t + 1, 3) = k  
  
                                Sheets(1).Cells(t + 1, 4) = l  
  
                                Sheets(1).Cells(t + 1, 5) = m  
  
                                Sheets(1).Cells(t + 1, 6) = n
```

```

End If

Next

Next

Next

Next

Next

Next

MsgBox "10 元换为零钱共有" & t & "种方法！"

End Sub

```

- (4) 运行该子过程，Excel 工作表中每一行将填写一种可能的换法，如图 3-31 所示。
- (5) 因为换零钱的方法很多，根据计算机的速度不同该程序的运行速度也不同，最后将通过对话框显示出总的换法次数，如图 3-32 所示。



图 3-31 零钱换法



图 3-32 换法总数

- (6) 在循环嵌套中，内层循环体执行的次数等各外层循环数数之积，如本例代码内循环执行次数为：

$$101 \times 51 \times 21 \times 11 \times 6 \times 3 = 21417858 \text{ 次}$$

- (7) 对于嵌套循环，一般都可以对代码进行一定的优化，使程序的执行效率更高。本例最简单的优化代码如下：

```
Sub 换零钱 2()
```

```

Dim t As Long

For j = 0 To 50          '2 角

    For k = 0 To 20      '5 角

        For l = 0 To 10  '1 元

            For m = 0 To 5    '2 元

                For n = 0 To 2    '5 元

                    t2 = 2 * j + 5 * k + 10 * l + 20 * m + 50 * n

                    If t2 <= 100 Then

                        t = t + 1

                        i = 100 - t2

                        Sheets(1).Cells(t + 1, 1) = i

                        Sheets(1).Cells(t + 1, 2) = j

                        Sheets(1).Cells(t + 1, 3) = k

                        Sheets(1).Cells(t + 1, 4) = l

                        Sheets(1).Cells(t + 1, 5) = m

                        Sheets(1).Cells(t + 1, 6) = n

                    End If

                Next

            Next

        Next

    Next

Next

Next

Next

Next

```

Next

MsgBox "10 元换为零钱共有" & t & "种方法！"

End Sub

(8) 以上程序中内循环的执行次数如下：

$51 \times 21 \times 11 \times 6 \times 3 = 212058$ 次

可以看出减少最外层循环的 101 次，可使用内循环体提高 100 倍的执行效率。

本例程序还有很多优化方法，这里就不再介绍。

3.4 使用数组

在程序中，如果要处理大量的数据，为每个数据定义一个变量将使程序变得很难阅读，并且代码很烦琐。

对于大量有序的数据，可以使用数组对其进行存储和处理。在其他程序设计语言中，数组中的所有元素都必须为同样的数据类型，在 VBA 中，数组中各元素可以是相同的数据类型，也可以是不同的数据类型。

例 029 数据排序

1. 案例说明

在 Excel 中可以方便地对单元格区域中的数据进行排序。本例使用 VBA 程序首先让用户输入 10 个数据，然后使用冒泡排序法对这 10 个数进行排序。

2. 关键技术

在程序中处理大量数据时，使用数组来保存是比较好的方法。数组使用之前可以使用 Dim、Static、Private 或 Public 语句来声明。在 VBA 中，数组最大可以达到 60 维，最常用的是一维数组和二维数组。

定义一维数组的语法格式如下：

Dim 数组名([下界 To] 上界) As 数据类型

其中“下界”可以省略，只给出数组的上界（即可以使用的最大下标值），这时默认值为 0，即数组的下标从 0 开始至定义的上界，如：

```
Dim MyArray(10) As String
```

定义了一个名为 **MyArray** 的数组，共有 11 个元素，分别为 **MyArray(0)**、**MyArray(1)**、...、**MyArray(10)**。

如果希望下标从 1 开始，可以通过 **Option Base** 语句来设置，其语法格式如下：

Option Base 1

使用该语句指定数组下标的默认下界，只能设为 0 或 1。



— 该语句只能出现在用户窗体或模块的声明部分，不能出现在过程中，且必须放在数组定义之前。

3. 编写代码

（1）新建 Excel 工作簿，按快捷键 “Alt+F11” 进入 VBE 环境。

（2）单击菜单 “插入/模块” 命令向工程中插入一个模块。

（3）在模块中编写以下代码：

Option Base 1

```
Sub 数据排序()
```

```
    Dim i As Integer, j As Integer
```

```
    Dim k
```

```
    Dim s(10) As Integer
```

```
    For i = 1 To 10
```

```
        s(i) = Application.InputBox("输入第" & i & "个数据: ", "输入数组", , , , , 1)
```

```
    Next
```



```

For i = 1 To 9

    For j = i + 1 To 10

        If s(i) < s(j) Then

            t = s(i)

            s(i) = s(j)

            s(j) = t

        End If

    Next

Next

For Each k In s

    Debug.Print k

Next

End Sub

```

在 VBA 中使用 **Inputbox** 函数接受用户输入数据时，返回的值为文本型。以上代码中使用了 **Application** 对象的 **InputBox** 方法来接受用户输入数据，该方法的语法格式如下：

Application.InputBox(Prompt, Title, Default, Left, Top, HelpFile, HelpContextID, Type)

设置 **Type** 参数可指定返回的数据类型，如本例设置其值为 **2**，则返回的值为数值型。

(4) 运行上面的宏，弹出如图 3-33 所示的对话框，提示用户输入数据。循环程序要求用户输入 10 个数据。

(5) 最后在“立即窗口”输出排序的结果，如图 3-34 所示。



图 3-33 输入数据

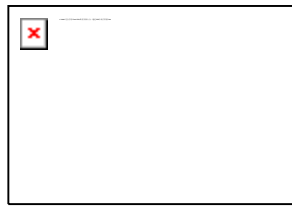


图 3-34 排序结果

例 030 彩票幸运号码

1. 案例说明

本例结合数组和随机函数的知识，生成指定数量的彩票幸运号码。本例生成的彩票号码每注由 7 位数构成，首先让用户输入产生的注数，再使用循环语句生成指定注数的号码。

2. 关键技术

本例代码中使用了两个关键技术：动态数组和随机函数。

(1) 动态数组

本例使用二维数组保存所有的彩票号码，二维数组的定义格式如下：

Dim 数组名(第 1 维上界, 第 2 维上界) **As** 数据类型

或

Dim 数组名(第 1 维下界 To 第 1 维上界, 第 2 维下界 To 第 2 维上界) **As** 数据类型

在本例中，因为生成的彩票数量是由用户输入的数据决定的。因此这里使用动态数组。

动态数组是指在程序运行时大小可以改变的数组，定义动态数组一般分两个步骤：首先用户窗体、模块或过程中使用 **Dim** 或 **Public** 声明一个没有下标的数组（不能省略括号），然后在过程中用 **ReDim** 语句重定义该数组的大小。

ReDim 语句在过程级别中使用，用于为动态数组变量重新分配存储空间。其语法格式如下：

ReDim [**Preserve**] 数组名(下标) [**As** 数据类型]

可以使用 **ReDim** 语句反复地改变数组的元素以及维数的数目，但是不能在将一个数组定义为某种数据类型之后，再使用 **ReDim** 将该数组改为其他数据类型，除非是 **Variant** 所包含的数组。

在默认情况下，使用 **ReDim** 语句重定义数组的维数和大小，数组中原来保存的值将全部消失，如果使用 **Preserve** 关键字，当改变原有数组最后一维的大小时，可以保持数组中原来的数据。

如果使用了 **Preserve** 关键字，就只能重新定义数组最后一维的大小，并不能改变维数的数目。

(2) 随机函数 Rnd

随机函数 **Rnd** 可返回小于 1 但大于或等于 0 的一个小数。其语法格式如下：

Rnd[(number)]

可选的 **number** 参数是 **Single** 或任何有效的数值表达式。根据 **number** 参数值的不同，**Rnd** 函数生成的随机数也不同：

- **number**<0，则每次使用相同的 **number** 作为随机数种得到的相同结果。
- **number**>0，则将生成随机序列中的下一个随机数。
- **number**=0，则将生成最近生成的数。
- 省略 **number**，则生成序列中的下一个随机数。

-  在调用 **Rnd** 之前，先使用无参数的 **Randomize** 语句初始化随机数生成器，该生成器具有根据系统计时器得到的种子。

为了生成某个范围内的随机整数，可使用以下公式：

Int((上限 - 下限 + 1) * **Rnd** + 下限)

3. 编写代码

(1) 新建 Excel 工作簿，按快捷键 “Alt+F11” 进入 VBE 环境。

(2) 单击菜单 “插入/模块” 命令向工程中插入一个模块。

(3) 在模块中编写以下代码:

Option Base 1

Sub 幸运号码()

Dim n As Integer, i As Integer, j As Integer

Dim l() As Integer

n = Application.InputBox("请输入需要产生幸运号码的数量: ", "幸运号码", , , , , 2)

ReDim l(n, 7) As Integer

For i = 1 To n

For j = 1 To 7

Randomize

l(i, j) = Int(10 * Rnd)

Next

Next

For i = 1 To n

For j = 1 To 7

Debug.Print l(i, j);

Next

Debug.Print

Next

End Sub

(4) 运行上面的宏, 弹出如图 3-35 所示的对话框, 提示用户输入数据。输入生成幸运号码的数量。

(5) 单击“确定”按钮后在“立即窗口”输出生成的幸运号码，如图 3-36 所示。



图 3-35 输入数据

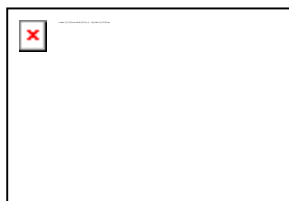


图 3-36 生成幸运号码

例 031 用数组填充单元格区域

1. 案例说明

在 Excel 中要处理大量数据时，可使用循环从各单元格读入数据，经过加工处理后再写回单元格区域中。这种方式比在数组中处理数据的速度要慢。因此，如果有大量的数据需要处理时，可先将数据保存到数组中，经过加工处理后，再将数组的数据填充到单元格区域。

本例演示将二维数组中的数据填充到工作表中的方法。

2. 关键技术

在 Excel 工作表中，工作表是一个二维结构，由行和列组成。这种特性与二维数组类似，因此可以很方便地将工作表单元格区域与二维数组之间进行转换。通过以下语句可将单元格区域赋值给一个二维数组：

```
myarr = Range(Cells(1, 1), Cells(5, 5))
```

反过来，也可将二维数组中的值快速的赋值给一个单元格区域，如以下语句将二维数组 myarr 中的值赋值给单元格区域 Rng：

```
Rng.Value = arr
```

3. 编写代码

(1) 新建 Excel 工作簿，按快捷键“Alt+F11”进入 VBE 环境。

(2) 单击菜单“插入/模块”命令向工程中插入一个模块。

(3) 在模块中编写以下代码：

```
Option Base 1
```

```
Sub 数组填充单元格区域()
```

```
Dim i As Long, j As Long
```

```
Dim col As Long, row As Long
```

```
Dim arr() As Long
```

```

row = Application.InputBox(prompt:="输入行数: ", Type:=2)

col = Application.InputBox(prompt:="输入列数: ", Type:=2)

ReDim arr(row, col)

For i = 1 To row

    For j = 1 To col

        arr(i, j) = (i - 1) * col + j

    Next

Next

Set Rng = Sheets(1).Range(Cells(1, 1), Cells(row, col))

Rng.Value = arr

End Sub

```

(4) 返回 Excel 操作环境，向工作表中添加一个按钮，设置提示文字为“填充数据”，指定该按钮的宏为“数组填充单元格区域”。

(5) 单击“填充数据”按钮，弹出如图 3-37 所示对话框，分别输入数组的行和列。

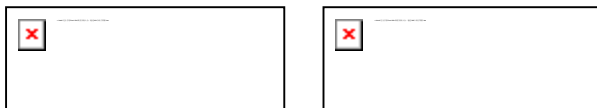


图 3-37 输入行和列

(6) VBA 代码生成一个二维数组，最后填充到工作表中，如图 3-38 所示。



图 3-38 填充数据

通过 Excel 相关对象可对工作表中的数据进行操作，如处理单元格区域的公式、对数据进行查询、排序、筛选等操作。本章演示使用 VBA 进行处理数据的实例。

12.1 处理公式

使用 VBA 代码可对工作表中的公式单元格进行处理，如判断单元格是否包含公式、复制公式、将单元格公式转换为具体的值等。

例 254 判断单元格是否包含公式

1. 案例说明

打开本例工作簿如图 12-1 所示，单击左上角的“公式单元格”按钮，将弹出如图 12-1 右图所示的提示框，显示当前工作表中定义了公式的单元格。

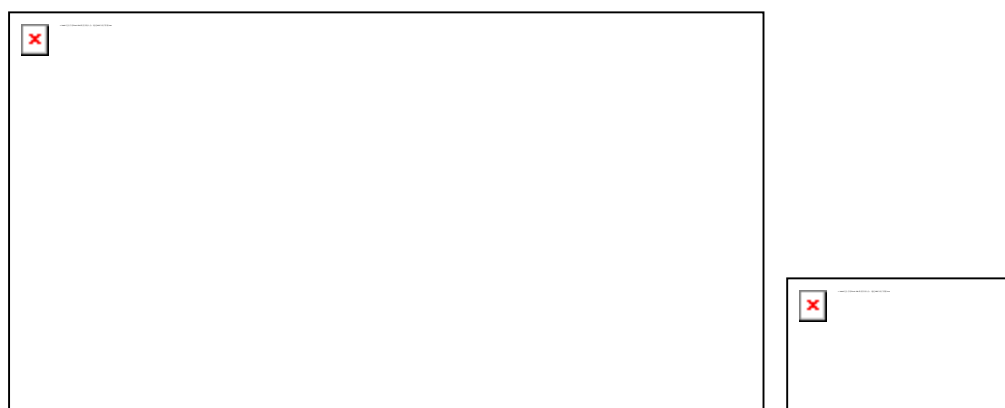


图 12-1 显示有公式的单元格

2. 关键技术

本例使用 Range 对象的 HasFormula 属性来判断指定单元格是否包含公式，如果区域中所有单元格均包含公式，则该属性值为 True；如果所有单元格均不包含公式，则该属性值为 False；其他情况下为 null。

本例对当前单元格区域中的单元格逐个进行判断，并显示出具有公式的单元格。

3. 编写代码

“公式单元格”按钮的 VBA 代码如下：


```
Sub 显示公式单元格()
```

```
    Dim rng As Range
```

```
    Set rng = ActiveSheet.Range("A1").CurrentRegion
```

```
    For Each c In rng.Cells
```

```
        If c.HasFormula Then
```

```
            MsgBox "单元格" & c.Address & " 定义了公式！"
```

```
        End If
```

```
    Next
```

```
End Sub
```

例 255 自动填充公式

1. 案例说明

打开本例工作簿如图 12-2 所示，在如图所示工作表中，单元格 J3 和 D16 定义了公式，单击“填充公式”按钮，单元格 J3 的公式将向下填充，单元格 D16 的公式向右填充，结果如图 12-3 所示。



图 12-2 原工作表

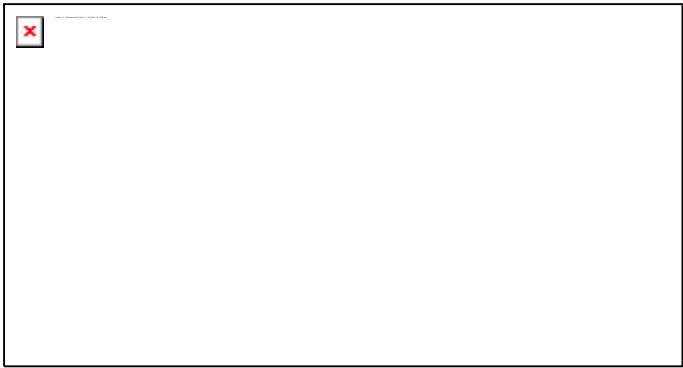


图 12-3 复制公式

2. 关键技术

本例使用 Range 对象的 AutoFill 方法，对指定区域中的单元格执行自动填充。该方法的语法格式如下：

表达式.AutoFill(Destination, Type)

该方法有两个参数，其含义如下：

- Destination：要填充的单元格。目标区域必须包括源区域。
- Type：指定填充类型。该填充类型可使用 xlAutoFillType 枚举类型，其值如表 12-1 所示。

表 12-1 xlAutoFillType 枚举值

名 称	值	描 述
xlFillCopy	1	将源区域的值和格式复制到目标区域，如有必要可重复执行
xlFillDays	5	将星期中每天的名称从源区域扩展到目标区域中。格式从源区域复制到目标区域，如有必要可重复执行
xlFillDefault	0	Excel 确定用于填充目标区域的值和格式
xlFillFormats	3	只将源区域的格式复制到目标区域，如有必要可重复执行
xlFillMonths	7	将月名称从源区域扩展到目标区域中。格式从源区域复制到目标区域，如有必要可重复执行
xlFillSeries	2	将源区域中的值扩展到目标区域中，形式为系列（如，“1, 2”扩展为“3, 4, 5”）。格式从源区域复制到目标区域，如有必要可重复执行
xlFillValues	4	只将源区域的值复制到目标区域，如有必要可重复执行
xlFillWeekdays	6	将工作周每天的名称从源区域扩展到目标区域中。格式从源区域复制到目标区域，如有必要可重复执行
xlFillYears	8	将年从源区域扩展到目标区域中。格式从源区域复制到目标区域，如有必要可重复执行
xlGrowthTrend	10	将数值从源区域扩展到目标区域中，假定源区域的数字之间是乘法关系（如，“1, 2,”扩展为“4, 8, 16”，假定每个数字都是前一个数字乘以某个值的结果）。格式从源区域复制到目标区域，如有必要可重复执行
xlLinearTrend	9	将数值从源区域扩展到目标区域中，假定数字之间是加法关系（如，“1, 2,”扩展为“3, 4, 5”，假定每个数字都是前一个数字加上某个值的结果）。格式从源区域复制到目标区域，如有必要可重复执行

3. 编写代码

“填充公式”按钮的 VBA 代码如下：

```
Sub 填充公式()  
  
    Dim i As Long, j As Long  
  
    With Range("A1").CurrentRegion  
  
        i = .Rows.Count - 1  
  
        j = .Columns.Count - 1  
  
    End With  
  
    Range("J3").AutoFill _  
        Destination:=Range(Cells(3, 10), Cells(i, 10))  
  
    Range("D16").AutoFill _  
        Destination:=Range(Cells(16, 4), Cells(16, j))  
  
End Sub
```

以上代码首先获取当前区域的行和列，接着使用 **AutoFill** 方法在垂直方向和水平方向填充相应的公式。

例 256 锁定和隐藏公式

1. 案例说明

打开本例工作簿如图 12-4 所示，单击“锁定隐藏公式”按钮，当前工作表中的所有公式单元格将被锁定，不允许用户修改，而其他单元格的数据用户可进行修改。同时，公式单元格定义的公式将被隐藏，单击选取具有公式的单元格时，将不会显示公式。



图 12-4 锁定和隐藏公式

2. 关键技术

要锁定和隐藏单元格，可通过 **Range** 对象的以下两个属性来进行设置。

- **Locked** 属性：指明对象是否已被锁定。
- **FormulaHidden** 属性：指明在工作表处于保护状态时是否隐藏公式。

当设置以上两个属性为 **True** 时，对指定区域锁定和隐藏。但要真正锁定和隐藏单元格，必须使用 **Protect** 方法对工作表进行保护。

3. 编写代码

“锁定隐藏公式”按钮的 **VBA** 代码如下：

Sub 锁定和隐藏公式()

If ActiveSheet.ProtectContents = **True** **Then**

MsgBox "工作表已保护！"

Exit Sub

End If

Worksheets("Sheet1").**Range**("A1").**CurrentRegion**.**Select**

Selection.Locked = **False**

```

Selection.FormulaHidden = False

Selection.SpecialCells(xlCellTypeFormulas).Select

Selection.Locked = True

Selection.FormulaHidden = True

Worksheets("Sheet1").Protect DrawingObjects:=True, Contents:=True, Scenarios:=True

Worksheets("Sheet1").EnableSelection = xlNoRestrictions

End Sub

```

例 257 将单元格公式转换为数值

1. 案例说明

打开本例工作簿如图 12-5 所示，在当前工作表中单元格区域“J3:J15”和“D16:I15”中都定义了公式，单击选择这两个区域中的任意一个单元格，编辑栏中将显示该单元格的公式。

单击工作表左上角的“公式转为数值”按钮，当前工作表中所有公式单元格的公式定义都将被具体计算值所替代，这时再修改引用单元格的值，这两个区域的值不会再变化了。



图 12-5 将公式转为数值

2. 关键技术

将单元格公式转换为计算结果的表示方法很简单，只需通过以下的赋值运算即可：

```
rng.Value = rng.Value
```

以上赋值语句中，`rng` 表示 `Range` 对象，该语句首先通过右侧的表达式 `rng.Value` 获取指定单元格的值（如果是公式，则获取公式的计算结果），再将该值赋值给单元格的 `Value` 变量，从而取代单元格原有的内容（公式）。

3. 编写代码

“公式转为数值”按钮的 VBA 代码如下：

```
Sub 公式转为数值()  
  
    Dim rng As Range, c As Range  
  
    Set rng = ActiveSheet.Range("A1").CurrentRegion  
  
    For Each c In rng.Cells  
  
        If c.HasFormula Then  
  
            c.Value = c.Value  
  
        End If  
  
    Next  
  
End Sub
```

以上代码首先获取工作表的当前区域，再逐个单元格判断，如果单元格有公式，则进行转换。

例 258 删除所有公式

1. 案例说明

在 Excel 中，当单元格的数据发生改变后，引用该单元格的公式单元格的值也会随之变化。有时希望经过计算后，具有公式的单元格的值不再随着引用单元格而变化。这时可以删除工作表中的公式，取消与引用单元格的关联。

打开本例工作簿如图 12-6 所示，在如图所示的工作表中部分单元格具有公式，单击选择单元格 I16，在编辑栏中可看到具体的公式。



图 12-6 具有公式的工作表

单击“删除所有公式”按钮，将打开如图 12-7 所示的对话框，询问用户是否删除提示工作簿中的所有公式，单击“是”按钮工作簿中各工作表中的公式都将被删除，如图 12-8 所示选中单元格 I16，编辑栏中可以看到显示的是具体的值，公式已被删除。



图 12-7 确认操作

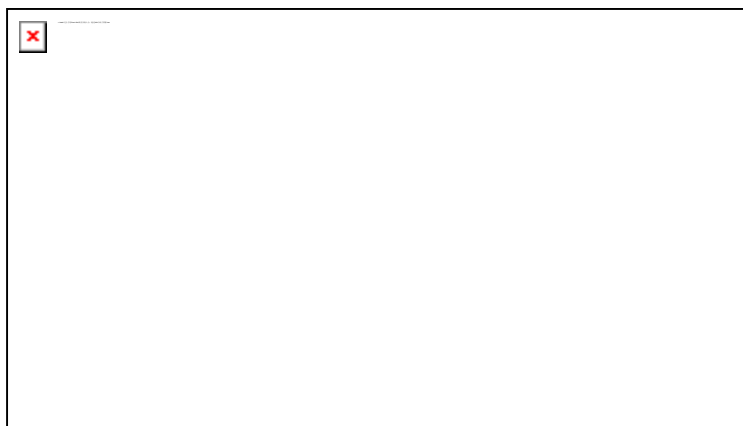


图 12-8 删除公式的工作表

2. 关键技术

本例代码与上例类似，不同的是本例将对所有打开工作簿进行处理，对每个工作簿的每张工作表进行循环，将具有公式的单元格转换为具体的数值。

3. 编写代码

“删除所有公式”按钮的 VBA 代码如下：

```
Sub 删除所有公式()  
  
    Dim wb1 As Workbook, ws1 As Worksheet  
  
    Dim rng As Range, rng1 As Range  
  
    For Each wb1 In Workbooks  
  
        With wb1  
  
            If MsgBox("是否删除工作簿"" & wb1.Name & ""中的所有公式?", _  
                vbQuestion + vbYesNo) = vbYes Then  
  
                For Each ws1 In .Worksheets  
  
                    On Error Resume Next  
  
                    Set rng1 = ws1.UsedRange.SpecialCells(xlCellTypeFormulas)  
  
                    '获取公式单元格区域引用  
  
                    For Each rng In rng1  
  
                        rng.Value = rng.Value '将公式转换成数值  
  
                    Next  
  
                Next  
  
            End If  
  
        End With  
  
    Next  
  
End Sub
```

例 259 用VBA表示数组公式

1. 案例说明

打开本例工作簿如图 12-9 所示。在 Excel 中，可以通过定义数组公式计算销售总金额。但是如果销售日报表中销售商品的数量不确定（占用表格的行是动态的），使用固定的数组公式就不太方便。



图 12-9 销售日报表

本例使用 VBA 动态定义数组公式，在图 12-9 所示工作表中输入数据，然后单击“汇总金额”按钮，在单元格 F5 中将根据录入数据的行数自动生成数组公式，如图 12-10 所示，在编辑栏可看到数组公式为：

```
{=SUM(B4:B9*C4:C9)}
```

2. 关键技术

使用 Range 对象的 FormulaArray 属性，可获取或设置区域的数组公式。如果指定区域不包含数组公式，则该属性返回 null。

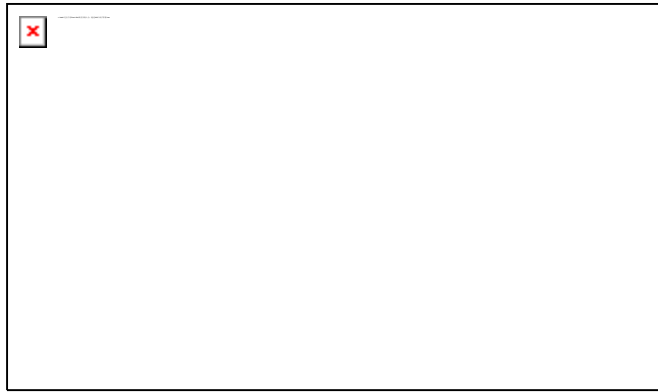


图 12-10 生成数组公式

3. 编写代码

“汇总金额”按钮的 VBA 代码如下：

```
Sub 汇总金额()  
  
    Dim r As Long  
  
    r = ActiveSheet.Range("A3").End(xlDown).Row  
  
    Range("F5").FormulaArray = "=SUM(B4:B" & r & "C4:C" & r & ")"  
  
End Sub
```

12.2 数据查询

在 Excel 中，数据查询是最常用的操作。在“开始”选项卡的“编辑”组中单击“查找和选择”按钮，从下拉的菜单按钮中选择相应的命令即可进行查询操作。在 VBE 中，可使用 Find 方法进行查询相关的操作，本节实例演示查询数据的 VBA 代码。

例 260 查找指定的值

1. 案例说明

打开本例工作簿如图 12-11 所示，单击左上角的“查找”按钮，弹出“查找”对话框如图 12-12 所示，在该对话框中输入要查找的值（如本例中输入 200），单击“确定”按钮，查找的结果显示在如图 12-13 所示的对话框中，同时工作表中对应单元格也加亮显示，如图 12-14 所示。



图 12-11 查找工作表



图 12-12 输入查找值

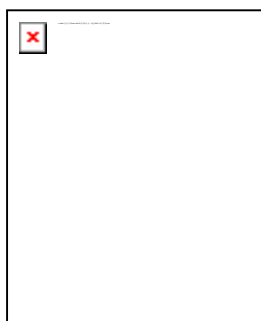


图 12-13 查找结果



图 12-14 加亮显示

2. 关键技术

本例的查找使用了 Range 对象的两个方法：Find 方法和 FindNext 方法。

（1）Find 方法

使用该方法可以在区域中查找特定信息。其语法格式如下：

表达式.Find(What, After, LookIn, LookAt, SearchOrder, SearchDirection, MatchCase, MatchByte, SearchFormat)

该方法的参数很多，其中 What 参数是必须指定的，其余参数都可省略。各参数的含义如下：

- What: 要搜索的数据。可为字符串或任意 Excel 数据类型。
- After: 表示搜索过程将从其之后开始进行的单元格。此单元格对应于从用户界面搜索时的活动单元格的位置。After 必须是区域中的单个单元格。要记住搜索是从该单元格之后开始的；直到此方法绕回到此单元格时，才对其进行搜索。如果不指定该参数，搜索将从区域的左上角的单元格之后开始。
- LookIn: 信息类型。
- LookAt: 设置匹配文本的方式。可为常量 xlWhole（匹配全部搜索文本）或 xlPart（匹配任一部分搜索文本）。
- SearchOrder: 指定搜索区域的次序。可为常量 xlByRows（按行）或 xlByColumns（按列）搜索。

- **SearchDirection**: 搜索的方向。可为常量 **xlNext**（在区域中搜索下一匹配值）或 **xlPrevious**（在区域中搜索上一匹配值）。
- **MatchCase** : 如果为 **True**，则搜索区分大小写。默认值为 **False**。
- **MatchByte**: 只在已经选择或安装了双字节语言支持时适用。如果为 **True**，则双字节字符只与双字节字符匹配。如果为 **False**，则双字节字符可与其对等的单字节字符匹配。
- **SearchFormat**: 搜索的格式。

使用该方法将返回一个 **Range** 对象，它代表第一个在其中找到该信息的单元格。如果未发现匹配项，则返回 **Nothing**。**Find** 方法不影响选定区域或当前活动的单元格。



- 每次使用此方法后，参数 **LookIn**、**LookAt**、**SearchOrder** 和 **MatchByte** 的设置都将被保存。如果下次调用此方法时不指定这些参数的值，就使用保存的值。设置这些参数将更改“查找”对话框中的设置，如果省略这些参数，更改“查找”对话框中的设置将更改使用的保存值。要避免出现这一问题，每次使用此方法时最好明确设置这些参数。

（2）FindNext 方法

FindNext 方法继续由 **Find** 方法开始的搜索。查找匹配相同条件的下一个单元格，并返回表示该单元格的 **Range** 对象。该操作不影响选定内容和活动单元格。其语法格式如下：

表达式.**FindNext**(After)

参数 **After** 指定一个单元格，查找将从该单元格之后开始。此单元格对应于从用户界面搜索时的活动单元格位置。



- **After** 必须是查找区域中的单个单元格。搜索是从该单元格之后开始的；直到本方法环绕到此单元格时，才检测其内容。如果未指定本参数，查找将从区域的左上角单元格之后开始。

当查找到指定查找区域的末尾时，**FindNext** 方法将环绕至区域的开始继续搜索。发生环绕后，为停止查找，可保存第一次找到的单元格地址，然后测试下一个查找到的单元格地址是否与其相同。

3. 编写代码

“查找”按钮的 VBA 代码如下：

```
Sub 查找指定值()  
  
    Dim result As String, str1 As String, str2 As String  
  
    Dim c As Range  
  
    result = Application.InputBox(prompt:="请输入要查找的值：", Title:="查找", Type:=2)  
  
    If result = "False" Or result = "" Then Exit Sub  
  
    Application.ScreenUpdating = False  
  
    Application.DisplayAlerts = False  
  
    With ActiveSheet.Cells  
  
        Set c = .Find(result, , , xlWhole, xlByColumns, xlNext, False)  
  
        If Not c Is Nothing Then  
  
            str1 = c.Address  
  
            Do  
  
                c.Interior.ColorIndex = 4 '加亮显示  
  
                str2 = str2 & c.Address & vbCrLf  
  
                Set c = .FindNext(c)  
  
            Loop While Not c Is Nothing And c.Address <> str1  
  
        End If  
  
    End With  
  
    MsgBox "查找到指定数据在以下单元格中：" & vbCrLf & vbCrLf _  
        & str2, vbInformation + vbOKOnly, "查找结果"  
  
    Application.ScreenUpdating = True  
  
    Application.DisplayAlerts = True
```

End Sub

以上代码首先让用户输入查找的值，接着使用 **Find** 方法查找第一个满足条件的单元格，再使用循环查找当前工作簿中下一个满足条件的单元格，并在循环中对满足条件的单元格设置不同的底纹，以突出显示。

例 261 带格式查找

1. 案例说明

打开本例工作簿如图 12-15 所示，单击左上角的“查找指定格式”按钮，单元格 A2 将被选中，并填上不同的底色。

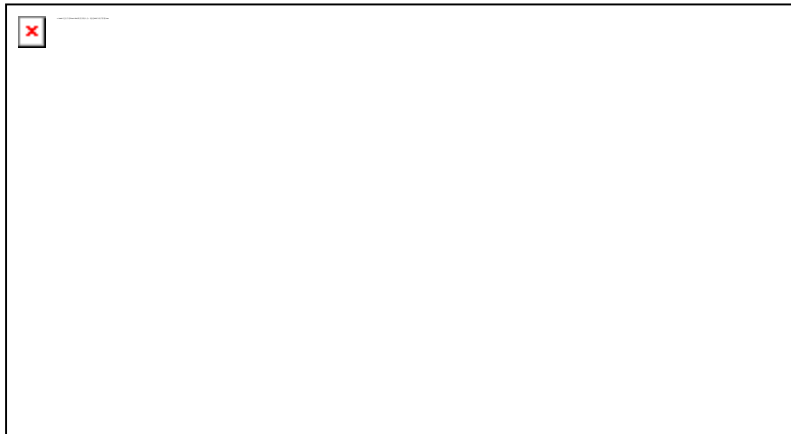


图 12-15 带格式查找

2. 关键技术

本例使用 Application 对象的 FindFormat 属性，设置要查找的单元格格式类型的搜索条件，然后使用 Find 方法按格式进行查找。

3. 编写代码

“查找指定格式”按钮的 VBA 代码如下：

```
Sub 查找指定格式()
```

```
With Application.FindFormat.Font
```

```
.Name = "宋体"
```

```
.FontStyle = "Bold"
```

```
.Size = 11
```

```
End With
```

```
Cells.Find(what:="", SearchFormat:=True).Activate
```

```
Selection.Interior.ColorIndex = 4 '加亮显示
```

```
End Sub
```

以上代码首先使用 **FindFormat** 属性设置查找的格式条件，接着使用 **Find** 方法按格式查找并激活满足条件的单元格，最后加亮显示激活单元格。

例 262 查找上一个/下一个数据

1. 案例说明

打开本例工作簿如图 12-16 所示，单击右上角的“查找”按钮，将弹出输入查找条件对话框，在对话框中输入查找条件单击“确定”按钮，即可在当前工作表中查找满足条件的单元格，找到满足条件的单元格后，选中该单元格。

单击“向前查找”或“向后查找”按钮，可从当前单元格向前或向后查找满足前面设置条件的单元格，并选中该单元格。

如果在使用“查找”按钮输入查找条件之前，就直接单击“向前查找”或“向后查找”按钮，也将弹出如图 12-17 所示的“查找”对话框输入查询条件。



图 12-16 查找上一下/下一个数据



图 12-17 输入查找条件

要重设查找条件，单击“查找”按钮打开如图 12-17 所示对话框即可。

2. 关键技术

(1) FindNext 方法

使用该方法继续由 **Find** 方法开始的搜索。查找匹配相同条件的下一个单元格，并返回表示该单元格的 **Range** 对象。该方法的语法格式如下：

表达式.**FindNext(After)**

参数 **After** 指定一个单元格，查找将从该单元格之后开始。此单元格对应于从用户界面搜索时的活动单元格位置。**After** 必须是查找区域中的单个单元格。搜索是从该单元格之后开始的；直到本方法环绕到此单元格时，才检测其内容。如果未指定本参数，查找将从区域的左上角单元格之后开始。

当查找到指定查找区域的末尾时，本方法将环绕至区域的开始继续搜索。发生环绕后，为停止查找，可保存第一次找到的单元格地址，然后测试下一个查找到的单元格地址是否与其相同。

（2）FindPrevious 方法

该方法继续由 **Find** 方法开始的搜索。查找匹配相同条件的上一个单元格，并返回代表该单元格的 **Range** 对象。其语法格式如下：

表达式.**FindPrevious(After)**

参数 **After** 指定一个单元格，查找将从该单元格之前开始。此单元格对应于从用户界面搜索时的活动单元格的位置。

3. 编写代码

（1）在 **VBE** 中插入一个模块，使用以下代码声明一个模块变量：

```
Dim c As Range
```

（2）“查找”按钮的 **VBA** 代码如下：

```
Sub 查找()
```

```
result = Application.InputBox(prompt:="请输入要查找的值：", Title:="查找", Type:=2)
```

```
If result = "False" Or result = "" Then Exit Sub
```

```
Set c = ActiveSheet.Cells.Find(result, , , xlWhole, xlByColumns, xlNext, False)
```

```
If Not c Is Nothing Then
```

```

        c.Activate

    End If

End Sub

```

以上代码首先提示用户输入查询条件，再使用 **Find** 方法向下查找。

(3) “向前查找”按钮的 VBA 代码如下：

```

Sub 向前查找()

    Dim result As String, str1 As String, str2 As String

    If c Is Nothing Then

        result = Application.InputBox(prompt:="请输入要查找的值： ", Title:="查找", Type:=2)

        If result = "False" Or result = "" Then Exit Sub

        Set c = ActiveSheet.Cells.Find(result, , , xlWhole, xlByColumns, xlPrevious, False)

    Else

        Set c = ActiveSheet.Cells.FindPrevious(c)

    End If

    If Not c Is Nothing Then

        c.Activate

    End If

End Sub

```

以上代码首先判断模块变量 **c** 是否为空(判断执行该子过程之前是否设置了查询条件)，若为空，则打开对话框让用户输入查询条件，并使用 **Find** 方法向前查找。若模块变量 **c** 不为空，则调用 **FindPrevious** 方法向前查找。

(4) “向后查找”按钮的 VBA 代码如下：

```
Sub 向后查找()  
  
    Dim result As String, str1 As String, str2 As String  
  
    If c Is Nothing Then  
  
        result = Application.InputBox(prompt:="请输入要查找的值：", Title:="查找", Type:  
=2)  
  
        If result = "False" Or result = "" Then Exit Sub  
  
        Set c = ActiveSheet.Cells.Find(result, , , xlWhole, xlByColumns, xlNext, False)  
  
    Else  
  
        Set c = ActiveSheet.Cells.FindNext(c)  
  
    End If  
  
    If Not c Is Nothing Then  
  
        c.Activate  
  
    End If  
  
End Sub
```

例 263 代码转换

1. 案例说明

打开本例工作簿如图 12-18 所示，在单元格 C3 中输入“101”，按回车键或 Tab 键后，单元格 C3 中输入的值将转换为“财务部”，如图 12-19 所示。

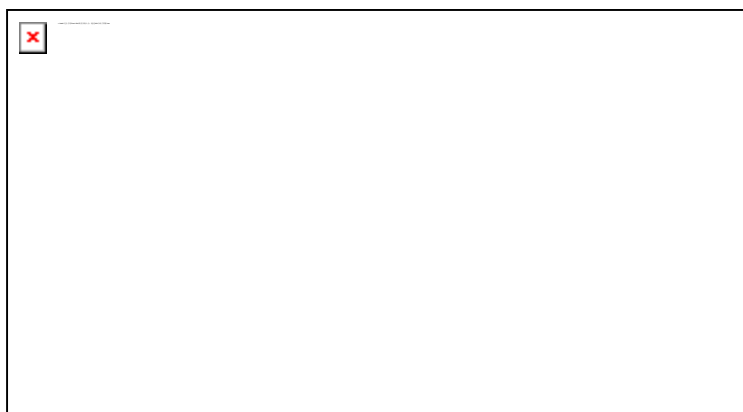


图 12-18 输入代码

单击工作表的“编码”标签，可看到编码表中编码与名称的对应关系，如图 12-20 所示。

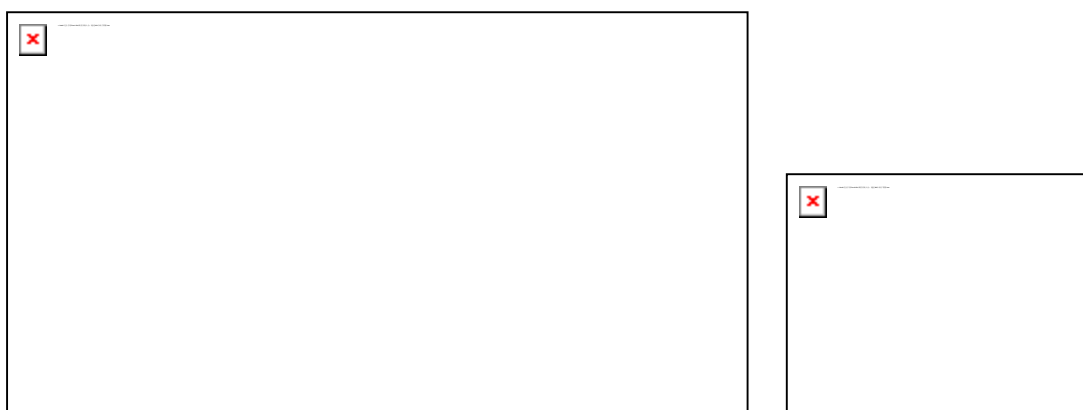


图 12-19 转换代码

图 12-20 编码表

2. 关键技术

本例使用查表的方法，将工作表中指定列中输入的代码转换为对应的值。在如图 12-20 所示的“编码”表中输入编码内容。

本例的关键技术是使用工作表事件 **Change** 事件来进行代码的转换。

当用户更改工作表中的单元格，或外部链接引起单元格的更改时发生 **Change** 事件。该事件的参数 **Target** 为数据正在被更改的区域。

3. 编写代码

在工作表“Sheet1”的 **Change** 事件中编写以下 VBA 代码：

```
Private Sub Worksheet_Change(ByVal Target As Range)
```

```

Dim t, rng As Range, i As Long, c As Range

If Target.Column = 3 And Target.Row > 2 And Target.Value <> "" Then

    t = Target.Value

    With Worksheets("编码")

        i = .Range("A1").End(xlDown).Row

        Set rng = .Range(.Cells(2, 1), .Cells(i, 1))

        Set c = rng.Find(what:=t)

        If c Is Nothing Then Exit Sub

        Target.Value = c.Offset(0, 1).Value

    End With

End If

End Sub

```

以上代码首先对更改单元格的行和列进行判断，如果是第 3 列第 2 行以下单元格，则执行编码转换的代码。在转换代码时先获取更改单元格的值，再从“编码”工作表中查找相应的编码，并将查到的编码对应的名称赋值给当前单元格，完成代码的转换。

例 264 模糊查询

1. 案例说明

打开本例工作簿如图 12-21 所示，单击“模糊查询”按钮，弹出如图 12-22 所示的对话框，在对话框中输入查询条件“刘”，单击“确定”按钮，即可在工作表中查找含有“刘”字的单元格，并为单元格填充底色，如图 12-23 所示。



图 12-21 模糊查询



图 12-22 查询条件

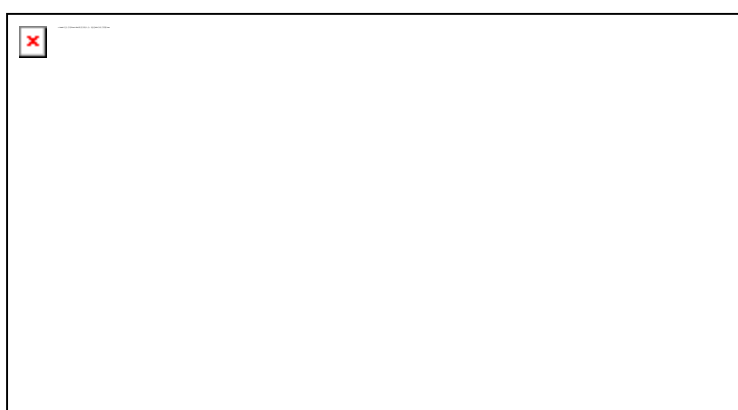


图 12-23 加亮显示查询结果

2. 关键技术

本例使用 **Like** 运算符进行模糊查询。**Like** 运算符可用来比较两个字符串。其使用方法如下：

```
result = string Like pattern
```

Like 运算符的语法具有以下几个部分：

- **result**：运算的结果。
- **string**：被查询的字符串。
- **pattern**：查询字符串，该字符串可建立模式匹配。

如果 **string** 与 **pattern** 匹配，则 **result** 为 **True**；如果不匹配，则 **result** 为 **False**。但是如果 **string** 或 **pattern** 中有一个为 **Null**，则 **result** 为 **Null**。

pattern 中的字符可使用以下匹配模式：

- ?：可为任何单一字符。
- *：零个或多个字符。
- #：任何一个数字(0–9)。
- [charlist]：charlist 中的任何单一字符。
- [!charlist]：不在 charlist 中的任何单一字符。

在中括号([])中，可以用由一个或多个字符(charlist)组成的组与 string 中的任一字符进行匹配，这个组几乎包括任何一个字符代码以及数字。

例如：

```
MyCheck = "张三" Like "张*" ' 返回 True
```

```
MyCheck = "F" Like "[A-Z]" ' 返回 True
```

```
MyCheck = "F" Like "[!A-Z]" ' 返回 False
```

```
MyCheck = "a2a" Like "a#a" ' 返回 True
```

3. 编写代码

“模糊查询”按钮的 VBA 代码如下：

```
Sub 模糊查询()  
  
    Dim result As String, str1 As String  
  
    Dim c As Range, rng As Range  
  
    result = Application.InputBox(prompt:="请输入要查找的值：", _  
        Title:="模糊查找", Type:=2)  
  
    If result = "False" Or result = "" Then Exit Sub
```

```

Application.ScreenUpdating = False

Application.DisplayAlerts = False

Set rng = ActiveSheet.Range("A1").CurrentRegion

str1 = "*" & result & "*"

For Each c In rng.Cells

    If c.Value Like str1 Then

        c.Interior.ColorIndex = 4

    End If

Next

Application.ScreenUpdating = True

Application.DisplayAlerts = True

End Sub

```

以上代码首先让用户输入查询条件，接着使用 **For** 循环逐个单元格进行比较，在比较时使用 **Like** 进行模糊查询，如果单元格中包含有指定条件的值，则设置单元格的底色。

例 265 网上查询快件信息

1. 案例说明

使用本例代码可查询申通快递的快件投递情况。打开本例工作簿如图 12-24 所示，单击“查询快件”按钮打开如图 12-25 所示对话框，在对话框中输入快件编号，单击“确定”按钮，经过一段时间后得到查询结果如图 12-26 所示。



图 12-24 查询工作表



图 12-25 输入快件编号



图 12-26 查询结果

—  本例使用的快件编号进行了处理（虚拟编号），在使用本例代码之前应确保计算已接入互联网。

2. 关键技术

（1）QueryTable 对象

QueryTable 对象代表一个利用从外部数据源（如 SQL Server、Microsoft Access 数据库、网络数据等）返回的数据生成的工作表表格。

QueryTable 对象是 QueryTables 集合的成员。

（2）Add 方法

使用 QueryTables 集合对象的 Add 方法可新建一个查询表。其语法格式如下：

表达式.Add(Connection, Destination, Sql)

该方法参数的含义如下：

- **Connection:** 查询表的数据源。可为连接数据库的连接字符串，也可以是一个 Web 查询。Web 查询字符串的格式如下：

URL;<url>

其中“URL;”是必需的，字符串的其余部分作为 Web 查询的 URL。

- **Destination:** 查询表目标区域（生成的查询表的放置区域）左上角的单元格。目标区域必须位于 QueryTables 对象所在的工作表中。
- **Sql:** 在 ODBC 数据源上运行的 SQL 查询字符串。当使用的数据源为 ODBC 数据源时，该参数可省略。

（3）Refresh 方法

使用 QueryTable 对象的 Refresh 方法可更新外部数据区域(QueryTable)。该方法的语法格式如下：

表达式.Refresh(BackgroundQuery)

参数 BackgroundQuery 如果为 True，则在数据库建立连接并提交查询之后，将控制返回给过程。QueryTable 在后台进行更新。如果为 False，则在所有数据被取回到工作表之后，将控制返回给过程。如果没有指定该参数，则由 BackgroundQuery 属性的设置决定查询模式。

在 Excel 建立一个成功的连接之后，将存储完整的连接字符串，这样，以后在同一编辑会话中调用 Refresh 方法时就不会再显示提示。通过检查 Connection 属性的值可以获得完整的连接字符串。

如果成功地完成或启动查询，则 Refresh 方法返回 True；如果用户取消连接或参数对话框，该方法返回 False。

（4）使用 Web 查询

在申能快递的网站上可查询快件的投递情况，在浏览器中输入以下网址：

<http://www.sto.cn/querybill/webform1.aspx?wen=&Submit2=%B2%E9%D1%AF>

将打开如图 12-27 所示的查询页面，在文本区中输入快件编号，单击“查询”按钮即可在网页上显示指定编号的快件投递情况。



图 12-27 通过网页查询快件投递情况

如果要在 Excel 中通过 VBA 查询快件投递情况，只需要将前面的 URL 地址中的“wen=”字符串后面加上快件编号即可。

3. 编写代码

“查询快件”按钮的 VBA 代码如下：

```
Sub 查询快件()
```

```
Dim str As String, strURL As String
```

```
str = Application.InputBox(prompt:="请输入快件的编号：", _
```

```
Title:="申通快件查询", Type:=2)
```

```
If str = "False" Then Exit Sub
```

```
strURL = "URL;http://www.sto.cn/querybill/webform1.aspx?wen="
```

```
strURL=strURL & str & "&Submit2=%E6%9F%A5%E8%AF%A2"
```

```
With ActiveSheet.QueryTables.Add(Connection:=strURL, Destination:=Range("A2"))
```

```

.Name = "abc"

.FieldNames = True

.WebSelectionType = xlSpecifiedTables    '导入指定表

.WebFormatting = xlWebFormattingNone    '不导入任何格式

.WebTables = "1,2"                      '导入第一个和第二个表格中的数据

.BackgroundQuery = True                  '查询异步执行（在后台执行）

.Refresh BackgroundQuery:=False         '更新数据

End With

End Sub

```

例 266 查询基金信息

1. 案例说明

打开本例工作簿，单击“查询基金信息”按钮，将在当前工作表中显示当前基金的信息如图 12-28 所示。



图 12-28 基金信息

2. 关键技术

在网站 <http://tw.stock.yahoo.com/us/worldinx.html> 中可查询基金的信息，如图 12-29 所示。

在图 12-29 所示的基金信息网页中，上面用 6 个表格显示了一些超链接信息。最下方的表格显示具体各基金的数据，本例通过 Web 查询只需要获取下方的表格即可。通过查看 HT

ML 代码, 可知该表格是第 7 个表格, 所以需要设置 QueryTable 对象的 WebTables 属性为 7。

3. 编写代码

“查询基金信息”按钮的 VBA 代码如下:

```
Sub 查询基金信息()
```

```
Dim strURL As String
```

```
strURL = "URL;http://fund.sohu.com/r/cxo.php"
```

```
With ActiveSheet.QueryTables.Add(Connection:=strURL, Destination:=Range("A2"))
```

```
.Name = "worldinx"
```

```
.FieldNames = True
```

```
.RowNumbers = False
```

```
.FillAdjacentFormulas = False
```

```
.PreserveFormatting = True
```

```
.RefreshOnFileOpen = False
```

```
.BackgroundQuery = True
```

```
.RefreshStyle = xlInsertDeleteCells
```

```
.SavePassword = False
```

```
.SaveData = True
```

```
.AdjustColumnWidth = True
```

```
.RefreshPeriod = 0
```

```
.WebSelectionType = xlSpecifiedTables
```

```
.WebFormatting = xlWebFormattingNone
```

```
.WebTables = "7"
```

```
.WebPreFormattedTextToColumns = True
```

```

.WebConsecutiveDelimitersAsOne = True

.WebSingleBlockTextImport = False

.WebDisableDateRecognition = False

.WebDisableRedirections = False

.Refresh BackgroundQuery:=False

End With

End Sub

```



图 12-29 网站查询基金信息

例 267 查询手机所在地

1. 案例说明

打开本例工作簿如图 12-30 所示，单击“手机所在地”按钮打开如图 12-31 所示对话框，输入手机号码后，单击“确定”按钮即可查询出手机所在地，如图 12-32 所示。

2. 关键技术

本例与前面各例使用的 Web 查询不同。本例使用 <http://www.123cha.com/> 网站来查询手机所在地。其查询的 HTML 代码如下：

```
<form method="post" action="index.php">请输入要查询的手机号码<b>前七位</b>或<b>全部</b>: <input type="text" name="query_mobile" size="18" class="tdc" value="">&n  
bsp;
```

```
<input type="submit" value="查 询">
```

```
</form>
```



图 12-30 查询手机所在地



图 12-31 输入手机号码



图 12-32 手机所在地

从以上 HTML 代码可以看出，查询手机所在地使用的是 POST 方法（另一种方式是 GET 方式，前面两例使用的这种方式），这种方法将传递一个查询变量到目标页面，需要提供以下两个参数：

- 第一个是查询页面，即 QueryTable 对象的 Connection 参数。该参数应该是 <form> 标签中的 action 关键字后面的页面。
- 另一个参数是 POST 方法的字符串，用于向 Web 服务器输入数据以从 Web 查询中返回数据。该参数通过 PostText 属性进行设置，设置该属性的值应该按以下格式：

```
.PostText = "query_mobile=13988888888"
```

其中 query_mobile 为 HTML 页面中用户输入参数的域的名称。

3. 编写代码

“手机所在地”按钮的 VBA 代码如下：

```
Sub 查询手机所在地()
```

```
Dim str As String, strURL As String
```

```
str = Application.InputBox(prompt:="请输入手机号码：", _
```

```
Title:="手机所在地查询", Type:=2)
```

```
If str = "False" Then Exit Sub
```

```
If Left(str, 2) <> "13" Then
```

```
MsgBox "请输入正确的手机号码！", vbCritical + vbOKOnly, "提示"
```

```
Exit Sub
```

```
End If
```

```
strURL = "URL;http://www.123cha.com/sj/index.php"
```

```
With ActiveSheet.QueryTables.Add(Connection:=strURL, Destination:=Range("A2"))
```

```
.Name = "cxo"
```

```
.PostText = "query_mobile=" & str
```

```
.FieldNames = True
```

```
.RowNumbers = False
```

```
.FillAdjacentFormulas = False
```

```

.PreserveFormatting = True

.RefreshOnFileOpen = False

.BackgroundQuery = True

.RefreshStyle = xlInsertDeleteCells

.SavePassword = False

.SaveData = True

.AdjustColumnWidth = True

.RefreshPeriod = 0

.WebSelectionType = xlSpecifiedTables

.WebFormatting = xlWebFormattingNone

.WebTables = "8"

.WebPreFormattedTextToColumns = True

.WebConsecutiveDelimitersAsOne = True

.WebSingleBlockTextImport = False

.WebDisableDateRecognition = False

.WebDisableRedirections = False

.Refresh BackgroundQuery:=False

```

End With

End Sub

例 268 使用字典查询

1. 案例说明

打开本例工作簿如图 12-33 所示，在如图所示工作表中列出了员工的姓名，“工资”列为空。单击“查询基础工资”按钮，“工资”列将自动填充员工对应的工资数据，如图 12-34 所示。

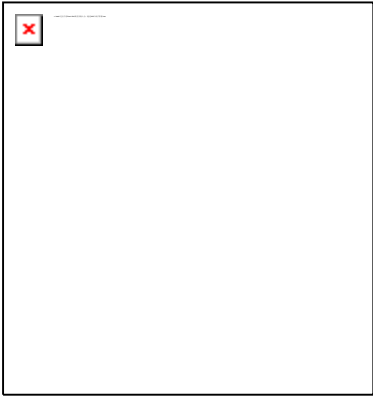


图 12-33 空表

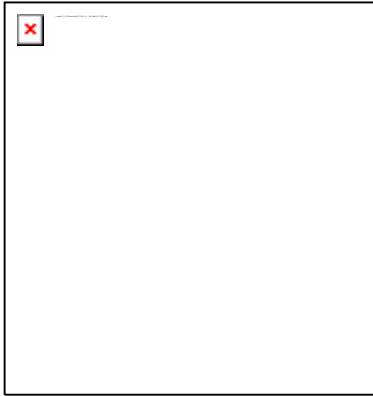


图 12-34 填充基础工资

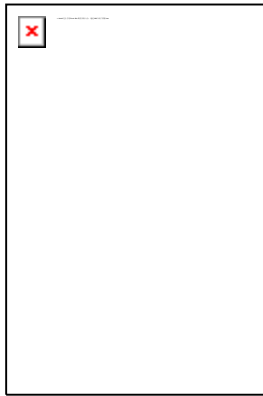


图 12-35 基础工资表

“基础工资表”工作表中的数据如图 12-35 所示，本例根据该工作表中的数据自动填充对应员工的工资。

2. 关键技术

(1) Dictionary 对象

Dictionary 对象用于在结对的名称/值中存储信息（等同于键/项目）。Dictionary 对象看似比数组更为简单，然而，Dictionary 对象却是更令人满意的处理关联数据的解决方案。使用 Dictionary 对象的属性和方法可操作具体的数据项。本例使用以下方法控制字典对象：

- **Add:** 向 Dictionary 对象添加新的键/项目对。
- **Exists:** 返回一个逻辑值，这个值可指示某个指定的键是否存在于 Dictionary 对象中。
- **Items:** 返回 Dictionary 对象中所有项目的一个数组。

(2) Transpose 方法

使用该方法将返回转置单元格区域，即将一行单元格区域转置成一行单元格区域，反之亦然。在行列数分别与数组的行列数相同的区域中，必须将 **TRANSPOSE** 输入为数组公式中。使用 **TRANSPOSE** 可在工作表中转置数组的垂直和水平方向。该方法的语法格式如下：

表达式.Transpose(Arg1)

参数 Arg1 是要进行转置的工作表中的单元格数组或区域。所谓数组的转置就是，将数组的第一行作为新数组的第一列，将数组的第二行作为新数组的第二列，依此类推。

3. 编写代码

“查询基础工资”按钮的 VBA 代码如下：

```
Sub 查询基础工资()
```

```
Dim arr, ds
```

```
Dim j As Long, k As Long, i As Long
```

```
Application.ScreenUpdating = False
```

```
Set ds = CreateObject("Scripting.Dictionary") '创建数据字典对象
```

```
With Worksheets("工资表")
```

```
    j = .Range("B2").End(xlDown).Row
```

```
    .Range("B3:B" & j) = ""           清除“工资”列中的数据
```

```
    k = .Range("A3").End(xlDown).Row
```

```
    arr = .Range("A3:A" & k)         将“姓名”列赋值到数组中
```

```
    For i = 3 To k                   将每个姓名作为一个字典对象的数据项
```

```
        ds.Add arr(i - 2, 1), ""
```

```
    Next
```

```
End With
```

```

With Worksheets("基础工资表")

    j = .Range("A3").End(xlDown).Row

    arr = .Range("A3:B" & j)

End With

On Error Resume Next

For i = 3 To j '在“基础工资表”查询“姓名”，有相同的姓名，则将工资保存到字典对象
中

    If ds.Exists(arr(i - 2, 1)) Then ds(arr(i - 2, 1)) = _

        ds(arr(i - 2, 1)) & arr(i - 2, 2)

Next

Worksheets("工资表").Range("B3").Resize(k - 2, 1) = _

    WorksheetFunction.Transpose(ds.Items)

Set ds = Nothing

Application.ScreenUpdating = True

End Sub

```

12.3 数据排序

在 Excel 2007 中，在“开始”选项卡的“编辑”组中单击“排序和筛选”按钮，从下拉的菜单按钮中选择相应的命令即可进行排序操作。在 VBE 中，可使用 Sort 方法进行排序相关的操作，本节实例演示数据排序的 VBA 代码。

例 269 用VBA代码排序

1. 案例说明

打开本例工作簿如图 12-36 所示，单击左上角的“按姓名排序”按钮，工作表中的数据按姓名升序排列，如图 12-37 所示。

2. 关键技术

在 Excel 2007 操作环境中进行排序时，在单元格中单击作为关键字的列，选择“开始”选项卡“编辑”组中的“排序和筛选”按钮中的相关命令可对工作表中的数据进行排序。但这时参与排序的是所有数据行，在如图 12-36 所示工作表中的数据排序时，最后一行（“合计”）也参与排序，使数据出现不希望的排序结果。

这时使用 VBA 代码可方便地控制排序的区域，Range 对象的 Sort 方法可对值区域进行排序。其语法格式如下：

表达式.Sort(Key1, Order1, Key2, Type, Order2, Key3, Order3, Header, OrderCustom, MatchCase, Orientation, SortMethod, DataOption1, DataOption2, DataOption3)



图 12-36 数据表

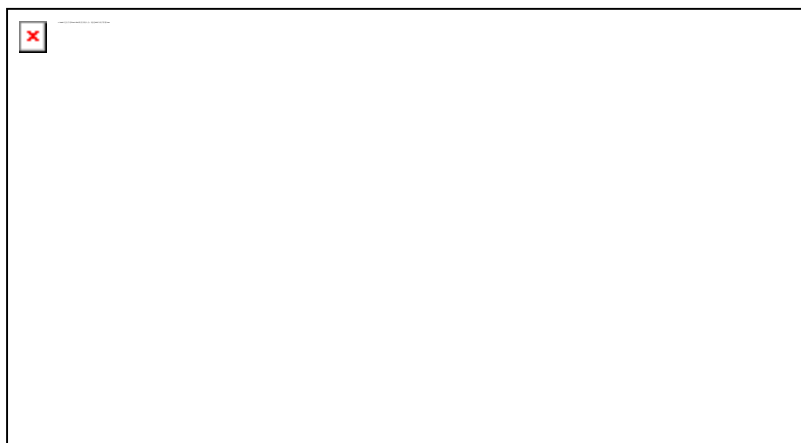


图 12-37 排序后的数据

该方法有很多参数，这些参数都可省略。各参数的含义如下：

- **Key1**: 指定第一排序字段，作为区域名称（字符串）或 **Range** 对象；确定要排序的值。
- **Order1**: 确定 **Key1** 中指定的值的排序次序，可设置为常量 **xlAscending**（升序）或 **xlDescending**（降序）。
- **Key2**: 第二排序字段。
- **Type**: 指定要排序的元素。
- **Order2**: 确定 **Key2** 中指定的值的排序次序。
- **Key3**: 第三排序字段。
- **Order3**: 确定 **Key3** 中指定的值的排序次序。
- **Header**: 指定第一行是否包含标题信息。
- **OrderCustom**: 指定在自定义排序次序列表中的基于 1 的整数偏移。
- **MatchCase**: 设置为 **True**，则执行区分大小写的排序，设置为 **False**，则执行不区分大小写的排序；不能用于数据透视表。
- **Orientation**: 指定以升序还是降序排序。可用常量 **xlSortColumns**（按列排序）或 **xlSortRows**（按行排序，这是默认值）。
- **SortMethod**: 指定排序方法。可用常量 **xlPinYin**（按汉语拼音顺序排序，这是默认值）或 **xlStroke**（按每个字符的笔画数排序）。

- **DataOption1:** 指定 **Key1** 中所指定区域中的文本的排序方式，可使用常量 **xlSortNormal**（分别对数字和文本数据进行排序，这是默认值）或 **xlSortTextAsNumbers**（将文本作为数字型数据进行排序）。
- **DataOption2:** 指定 **Key2** 中所指定区域中的文本的排序方式。
- **DataOption3:** 指定 **Key3** 中所指定区域中的文本的排序方式。

—



— 使用 **Sort** 方法排序时，最多只能按 3 个关键字进行排序。

—

3. 编写代码

“按姓名排序”按钮的 **VBA** 代码如下：

Sub 排序()

```
Dim rng As Range, r As Long, c As Long
```

```
r = ActiveSheet.Range("A1").CurrentRegion.Rows.Count
```

```
c = ActiveSheet.Range("A2").CurrentRegion.Columns.Count
```

```
Set rng = ActiveSheet.Range(Cells(3, 1), Cells(r - 1, c))
```

```
rng.Sort key1:=ActiveSheet.Range(Cells(3, 2), Cells(r - 1, 2))
```

End Sub

以上代码首先获取当前工作表中需要排序的单元格区域，对该区域使用 **Sort** 方法按“姓名”列进行排序。

例 270 乱序排序

1. 案例说明

在很多情况下，希望得到一种无序的数据排列，使用乱序排序的方法可得到这种效果，本例演示这种效果。打开本例工作簿，单击工作表左上角的“乱序排序”按钮，工资表中的数据将呈无序排列，如图 12-38 所示。

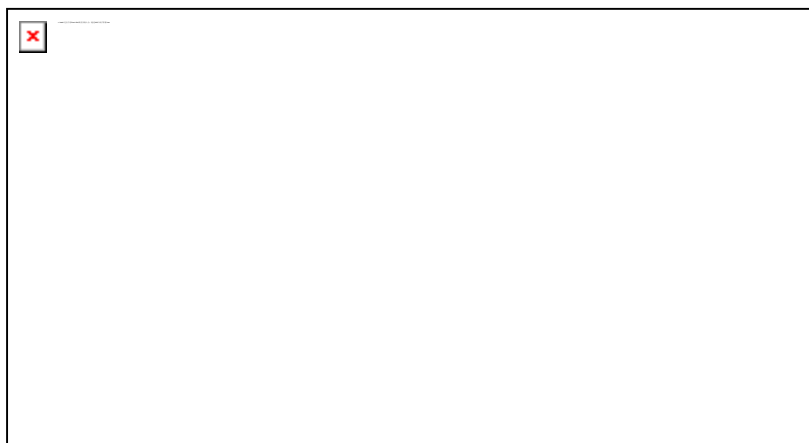


图 12-38 乱序排序

2. 关键技术

使用乱序排序的一种算法是：在需要排序的数据右侧生成一系列随机数据，然后以该随机数的列作为关键字进行排序，即可得到乱序的效果。

3. 编写代码

“乱序排序”按钮的 VBA 代码如下：

```
Sub 乱序排序()
```

```
    Dim rng As Range, r As Long, c As Long
```

```
    Randomize
```

```
    Application.ScreenUpdating = False
```

```
    With ActiveSheet
```

```
        r = .Range("A1").CurrentRegion.Rows.Count
```

```
        c = .Range("A2").CurrentRegion.Columns.Count
```

```
        For i = 3 To r - 1      '添加随机数据
```

```

        .Cells(i, c + 1) = Int((Rnd * 100) + 1)

Next

Set rng = .Range(Cells(3, 1), Cells(r - 1, c + 1))

rng.Sort key1:=.Range(Cells(3, c + 1), Cells(r - 1, c + 1))

.Columns(c + 1).Clear '清除添加的随机数据

End With

Application.ScreenUpdating = True

End Sub

```

以上代码首先在需要排序的数据右列添加随机数据，再使用 **Sort** 方法按该列的数据进行排序，最后删除增加的随机数据列。

例 271 自定义序列排序

1. 案例说明

打开本例工作簿，单击“自定义序列排序”按钮，工作表中的数据将按 C 列（部门）中的数据按自定义序列排序，如图 12-39 所示。自定义序列如图 12-40 所示，在图 12-40 所示工作表中更改数据的排列顺序后，再单击“自定义序列排序”按钮，C 列（部门）又将按新的序列重新排列。

2. 关键技术

本例演示用 VBA 代码创建自定义序列的方法，主要用 **AddCustomList** 方法添加自定义序列，用 **DeleteCustomList** 方法删除自定义序列。

（1）AddCustomList 方法

用该方法为自定义自动填充和/或自定义排序添加自定义列表。其语法格式如下：

表达式.AddCustomList(ListArray, ByRow)

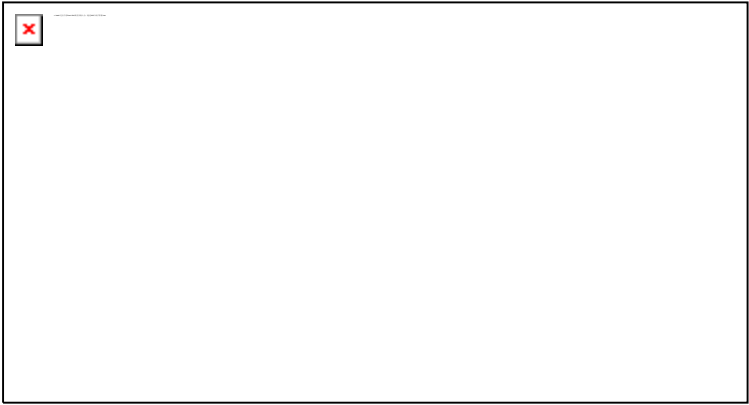


图 12-39 自定义序列排序

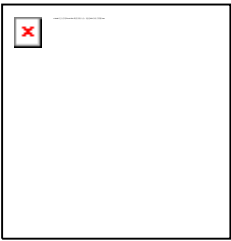


图 12-40 自定义序列

参数的含义如下：

- **ListArray**：将源数据指定为字符串数组或 **Range** 对象。
- **ByRow**：仅当 **ListArray** 为 **Range** 对象时使用。如果为 **True**，则使用区域中的每一行创建自定义列表；如果为 **False**，则使用区域中的每一列创建自定义列表。如果省略该参数，并且区域中的行数比列数多（或者行数与列数相等），则 **Excel** 使用区域中的每一列创建自定义列表。如果省略该参数，并且区域中的列数比行数多，则 **Excel** 使用区域中的每一行创建自定义列表。

—



- 如果要添加的列表已经存在，则本方法不起作用。

（2）GetCustomListNum 方法

使用 **Application** 对象的 **GetCustomListNum** 方法返回字符串数组的自定义序列号。其语法格式如下：

表达式.GetCustomListNum(ListArray)

参数 **ListArray** 为一个字符串数组。

（3）DeleteCustomList 方法

使用 **Application** 对象的 **DeleteCustomList** 方法删除一个自定义序列。其语法格式如下：

表达式.DeleteCustomList(ListNum)

参数 ListNum 为自定义序列数字。此数字必须大于或等于 5（Excel 有 4 个不可删除的内置自定义序列）。

3. 编写代码

“自定义序列排序”按钮的 VBA 代码如下：

Sub 自定义序列排序()

Dim rng As Range, r As Long, c As Long, n As Integer

Dim rng1 As Range, arr1

Application.ScreenUpdating = False

 '获取排序的单元格区域

 r = ActiveSheet.Range("A1").CurrentRegion.Rows.Count

 c = ActiveSheet.Range("A2").CurrentRegion.Columns.Count

 Set rng1 = ActiveSheet.Range(Cells(3, 1), Cells(r - 1, c))

 '添加自定义序列

With Worksheets("Sheet2")

 r = .Range("A1").End(xlDown).Row

 Set rng = .Range(.Cells(1, 1), .Cells(r, 1))

End With

```

With Application

    arr1 = .WorksheetFunction.Transpose(rng)

    .AddCustomList ListArray:=arr1

    n = .GetCustomListNum(arr1)

End With

'用自定义序列排序

rng1.Sort key1:=ActiveSheet.Range(Cells(3, 3), Cells(r - 1, 3)), _

    Order1:=xlAscending, Header:=xlGuess, OrderCustom:=n + 1

Application.DeleteCustomList ListNum:=n '删除自定义序列

Application.ScreenUpdating = True

End Sub

```

以上代码首先获取需要排序的单元格区域，接着将工作表 **Sheet 2** 中的数据添加到自定义序列中，再使用自定义序列进行排序，最后删除自定义序列。

例 272 多关键字排序

1. 案例说明

在 Excel 中对数据进行排序时，最多只能使用 3 个关键字排序，如果 3 个关键字相同时，要使用 4 个或更多关键字排序就比较麻烦。本例演示使用 4 个关键字排序的方法。

打开本例工作簿，单击工作表左上角的“多关键字排序”按钮，工作表中的数据将按 C 列到 F 列（共 4 列）的数据进行排序，得到如图 12-41 所示的结果。从图中可以看出，首先按 C 列（部门）排序，部门相同时再按 D 列（基础工资）排序，基础工资相同再按 E 列（岗位工资）排序，岗位工资相同再按 F 列（工龄工资）排序。如销售部两员工的基础工资、岗位工资都相同，则按工龄工资排序（陈晴工龄工资低，排在前面）。

2. 关键技术

对于超过三个关键字的排序，本例使用的方法时，先将数据按最后一个关键字排序，接着再将数据按倒数第二个关键字排序，……，最后将数据按主要（第一个）关键字排序，即可得到所需要的排列。

使用这种方法，可使用任意数量的关键字进行排序。



图 12-41 多关键字排序

3. 编写代码

“多关键字排序”按钮的 VBA 代码如下：

```
Sub 多关键字排序()  
  
    Dim rng1 As Range, r As Long, c As Long, i As Integer  
  
    Application.ScreenUpdating = False  
  
    '获取排序的单元格区域  
  
    r = ActiveSheet.Range("A1").CurrentRegion.Rows.Count  
  
    c = ActiveSheet.Range("A2").CurrentRegion.Columns.Count  
  
    Set rng1 = ActiveSheet.Range(Cells(3, 1), Cells(r - 1, c))  
  
    With rng1  
  
        For i = 6 To 3 Step -1
```

```
.Sort key1:=ActiveSheet.Range("C3").Offset(, i - 3)
```

```
Next
```

```
End With
```

```
Application.ScreenUpdating = True
```

```
End Sub
```

例 273 输入数据自动排序

1. 案例说明

打开本例工作簿，在 B 列中输入姓名，如图 12-42 所示，当按回车键或 Tab 键完成该列单元格的输入时，输入的数据将自动按顺序排列到工作表的相应行中，如图 12-43 所示。

2. 关键技术

本例需要根据用户对单元格数据的更改及时完成排序，所以需要在工作表的 Change 事件过程中编写代码，有关该事件过程的应用在本书前面多个例子都在使用。

另外本例还使用了 Application 对象的 Intersect 方法，该方法返回一个 Range 对象，该对象表示两个或多个区域重叠的矩形区域。其语法格式如下：

```
表达式.Intersect(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10, Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21, Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)
```

该方法最多可使用 30 个单元格区域作为参数，至少需使用两个参数。



图 12-42 输入数据

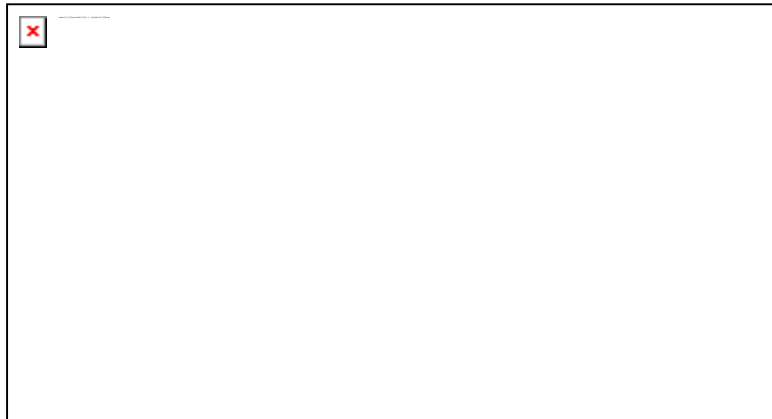


图 12-43 自动排序

在本例中，使用以下表示方法判断 **Target** 和单元格区域[B3:B1000]是否有重叠，若有重叠，则表示 **Target** 包含在区域[B3:B1000]中，否则，则是在该区域之外。

```
Application.Intersect(Target, [B3:B1000])
```

3. 编写代码

要完成本例的功能，需要在工作表的 **Change** 事件过程中编写以下代码：

```
Private Sub Worksheet_Change(ByVal Target As Range)

    If Target.Column <> 2 Then Exit Sub '修改的数据不是第 2 列，退出

    If Not Application.Intersect(Target, [B3:B1000]) Is Nothing Then

        Set rng = ActiveSheet.Range("A1").CurrentRegion

        Set rng = rng.Offset(2, 0).Resize(rng.Rows.Count - 2, rng.Columns.Count)

        rng.Sort Key1:=Range("B3")

    End If

End Sub
```

以上代码首先判断更改数据的单元格是否为第 2 列,接着判断更改数据单元格是否为“B3:B1000”单元格区域中的单元格,然后获取当前区域需要排序的单元格区域,使用 **Sort** 方法对这个区域进行排序即可。

例 274 数组排序

1. 案例说明

打开本例工作簿如图 12-44 所示，单击“生成随机数”按钮，打开如图 12-45 所示对话框，在对话框中输入需要生成的随机数数量，单击“确定”按钮即可生成相应的随机数，如图 12-46 所示。

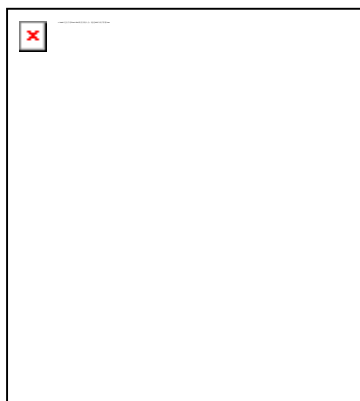


图 12-44 空工作表



图 12-45 输入数量

单击“排序”按钮，将生成的随机数按升序排列，如图 12-47 所示。

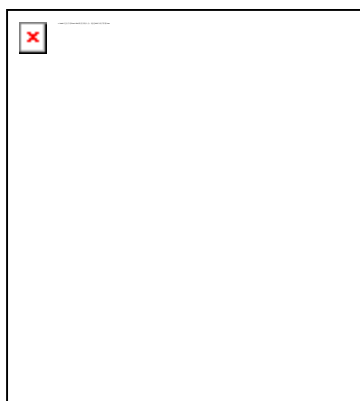


图 12-46 生成随机数

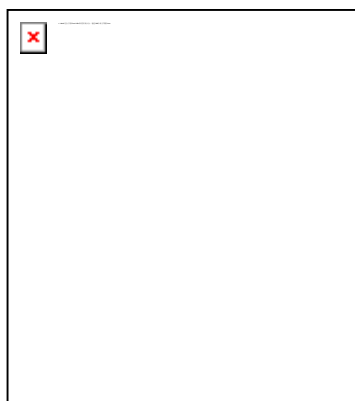


图 12-47 排序

2. 关键技术

Excel 工作表可以方便地和数组进行转换，即单元格区域可以赋值给一个数组，数组也可以通过 Transpose 方法填充到单元格区域中去。

(1) 单元格区域赋值给数组

使用以下方法可将单元格区域赋值给一个数组：

```
arr = ActiveSheet.Range("A1:A10")
```

使用这种赋值将产生一个二维数组，即使单元格区域只选择一行（或一列），得到的也是一个二维数组。

（2）数组填充单元格区域

对于二维数组，可直接使用以下方法将其赋值给单元格区域：

```
ActiveSheet.Range("A1:A" & n) = arr
```

如果是一维数组，则需要使用 **Transpose** 方法对数组进行置换为列或列进行填充。

3. 编写代码

（1）“生成随机数”按钮的 VBA 代码如下：

```
Sub 生成随机数()
```

```
Dim arr(), i As Long, n As Long
```

```
Randomize Timer
```

```
n = Application.InputBox(prompt:="请输入要生成的随机数数量（2-65536）：", _
```

```
Title:="输入数量", Default:=10, Type:=1)
```

```
If n <= 0 Or r > 65536 Then Exit Sub
```

```
ReDim arr(1 To n) '定义动态数组
```

```
For i = 1 To n '循环生成随机数
```

```
arr(i) = Int(Rnd * 10000)
```

```
Next
```

```

With ActiveSheet

    .Columns(1).Clear

    .Range("A1:A" & n) = Application.Transpose(arr) '数组赋值给单元格区域

End With

End Sub

```

(2) “排序”按钮的 VBA 代码如下：

```

Sub 排序()

    Dim arr, t

    Dim i As Long, j As Long, n As Long

    n = ActiveSheet.Range("A1").End(xlDown).Row

    If n <= 1 Then Exit Sub

    arr = ActiveSheet.Range("A1:A" & n) '单元格区域保存到数组中

    For i = 1 To n - 1 '双循环排序

        For j = i + 1 To n

            If arr(j, 1) < arr(i, 1) Then

                t = arr(i, 1) '交换数据

                arr(i, 1) = arr(j, 1)

                arr(j, 1) = t

            End If

        Next j

    Next i

End Sub

```

End If

Next

Next

ActiveSheet.Range("A1:A" & n) = arr '数组赋值给单元格区域

End Sub

例 275 使用Small和Large函数排序

1. 案例说明

打开本例工作簿，在工作表中单击“生成随机数”按钮将打开如图 12-48 所示的对话框，在对话框中输入产生随机数的个数，单击“确定”按钮将在工作表中的 A 列生成指定数量的随机数。

单击“升序排序”按钮，生成的随机数将按从小到大的顺序排列，如图 12-49 所示。单击“降序排序”按钮，生成的随机数将按从大到小的顺序排列。



图 12-48 输入随机数量

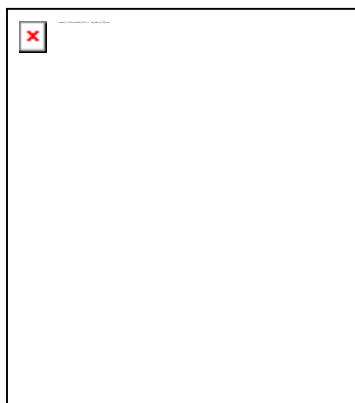


图 12-49 升序排序

2. 关键技术

(1) Small 方法

在 VBA 中通过 WorksheetFunction 对象的 Small 方法可调用 Excel 工作表函数 Small。该方法将返回数据集中第 k 个最小值。其语法格式如下：

表达式.Small(Arg1, Arg2)

参数的含义如下：

- **Arg1**: 需要确定第 k 个最小值的数值数据数组或区域。
- **Arg2**: 要返回的数据在数组或区域中的位置（从最小值开始）。

如果 Arg1 为空，则 Small 将返回错误值#NUM!。

如果 Arg2 ≤ 0 或 Arg2 超过了数据点个数，则 Small 将返回错误值#NUM!。

如果 n 为数组中数据点的个数，则 Small(array,1)等于最小值，Small(array,n)等于最大值。

（2）Large 方法

与 Small 方法类似，Large 方法返回数据集中第 k 个最大值（Small 方法返回第 k 个最小值）。例如，可以使用函数 Large 得到第一名、第二名或第三名的得分。

3. 编写代码

（1）“生成随机数”按钮的 VBA 代码如下：

Sub 生成随机数()

Dim arr(), i As Long, n As Long

Randomize Timer

n = Application.InputBox(prompt:="请输入要生成的随机数数量（2-65536）：", _

Title:="输入数量", Default:=10, Type:=1)

If n <= 0 Or r > 65536 Then Exit Sub

ReDim arr(1 To n) '定义动态数组

For i = 1 To n '循环生成随机数

arr(i) = Int(Rnd * 10000)

Next

```

With ActiveSheet

    .Columns(1).Clear

    .Range("A1:A" & n) = WorksheetFunction.Transpose(arr) '数组赋值给单元格区域

End With

End Sub

```

(2) “升序排序”按钮的 VBA 代码如下：

```

Sub 升序排序()

    Dim arr, arr1(), i As Long, n As Long

    n = ActiveSheet.Range("A1").End(xlDown).Row

    If n <= 1 Then Exit Sub

    arr = ActiveSheet.Range("A1:A" & n)    '单元格区域保存到数组中

    ReDim arr1(1 To n)

    For i = 1 To n                '选出第 i 个最小的数

        arr1(i) = WorksheetFunction.Small(arr, i)

    Next

    ActiveSheet.Range("A1:A" & n) = WorksheetFunction.Transpose(arr1)

    '数组赋值给单元格区域

End Sub

```

(3) “降序排序”按钮的 VBA 代码如下：

```

Sub 降序排序()

    Dim arr, arr1(), i As Long, n As Long

```



```

n = ActiveSheet.Range("A1").End(xlDown).Row

If n <= 1 Then Exit Sub

arr = ActiveSheet.Range("A1:A" & n)    '单元格区域保存到数组中

ReDim arr1(1 To n)

For i = 1 To n                        '选出第 i 个最大的数
    arr1(i) = WorksheetFunction.Large(arr, i)
Next

ActiveSheet.Range("A1:A" & n) = WorksheetFunction.Transpose(arr1)

'数组赋值给单元格区域

End Sub

```

例 276 使用RANK函数排序

1. 案例说明

打开本例工作簿，单击“生成随机数”按钮在工作表中的 A 列生成指定数量的随机数。单击“排序”按钮，生成的随机数将按从小到大的顺序排列，如图 12-50 所示。

2. 关键技术

使用 WorksheetFunction 对象的 Rank 方法，可返回一个数字在数字列表中的排位。数字的排位是其大小与列表中其他值的比值（如果列表已排过序，则数字的排位就是它当前的位置）。

Rank 方法语法的语法格式如下：

表达式.Rank(Arg1, Arg2, Arg3)

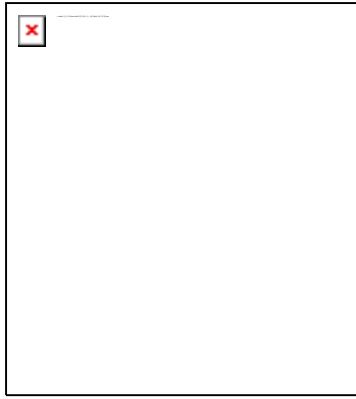
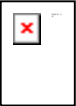


图 12-50 排序

各参数的含义如下：

- **Arg1**: 为要查找其排位的数字。
- **Arg2**: 数字列表数组或对数字列表的引用，为一个 **Range** 对象。
- **Arg3**: 指定数字的排位方式的数字。

如果 **Arg3** 为 0（零）或被省略，Excel 会按照 **Arg2** 为按降序排序的列表对数字排位。如果 **Arg3** 不为零，Excel 会按照 **Arg2** 为按升序排序的列表对数字排位。

-  函数 **RANK** 对重复数的排位相同。但重复数的存在将影响后续数值的排位。例如，在一列按升序排列的整数中，如果整数 **10** 出现两次，其排位为 **5**，则 **11** 的排位为 **7**（没有排位为 **6** 的数值）。

3. 编写代码

（1）“生成随机数”按钮的 VBA 代码如下：

```
Sub 生成随机数()
```

```
Dim arr(), i As Long, n As Long
```

```
Randomize Timer
```

```
n = Application.InputBox(prompt:="请输入要生成的随机数数量（2-65536）：", _
```

```

Title:="输入数量", Default:=10, Type:=1)

If n <= 0 Or r > 65536 Then Exit Sub

ReDim arr(1 To n)                '定义动态数组

For i = 1 To n                    '循环生成随机数

    arr(i) = Int(Rnd * 10000)

Next

With ActiveSheet

    .Columns(1).Clear

    .Range("A1:A" & n) = WorksheetFunction.Transpose(arr) '数组赋值给单元格区域

End With

End Sub

```

(2) “排序”按钮的 VBA 代码如下:

```

Sub 排序()

    Dim arr, rng As Range, t As Long, i As Long

    n = ActiveSheet.Range("A1").End(xlDown).Row

    If n <= 1 Then Exit Sub

    ReDim arr(1 To n)

    Set rng = ActiveSheet.Range("A1:A" & n) '获取单元格区域引用

```

```

For i = 1 To n

    t = WorksheetFunction.Rank(rng(i, 1), rng, 1)

    arr(t) = rng(i, 1)

Next

ActiveSheet.Range("A1:A" & n) = WorksheetFunction.Transpose(arr)

'数组赋值给单元格区域

End Sub

```

例 277 姓名按笔画排序

1. 案例说明

在各种会议中，对出席会议（或选举产生）的人员需要列出名单，这些名单一般是按姓名笔画排序。Excel 提供了按笔画排序的方法，但用这种方法排序时也将会出现一些问题，例如：姓名为双字的，一般要在姓和名之间加上一个空格，若为女性或少数民族，还要在姓名后面用括号标明。

本例编写 VBA 代码，对姓名按笔画排序，能自动处理姓名之间有空格、有括号的情况。打开本例工作簿如图 12-51 所示，单击“按姓名笔画排序”按钮，将得到如图 12-52 所示的排序结果。

本例自动生成按笔画排序的汉字库表，如图 12-53 所示。该工作表根据“姓名”工作表中的汉字自动生成。

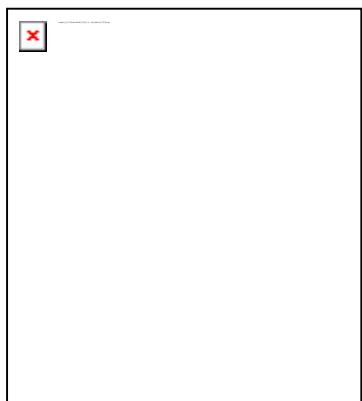


图 12-51 无序姓名

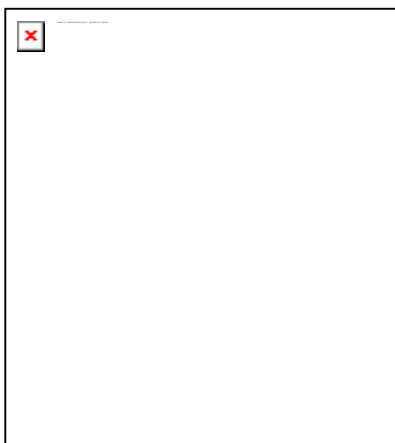


图 12-52 笔画排序

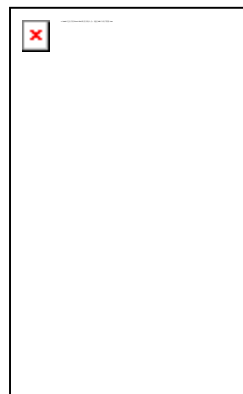


图 12-53 汉字库

2. 关键技术

使用笔画对数据进行排序时，需设置排序方法 Sort 的 SortMethod 属性，该属性指定中文排序方法。可设置为以下值：

- **xlPinYin**：按字符的汉语拼音顺序排序。这是默认值。
- **xlStroke**：按每个字符的笔画数排序。

本例的代码很长，其工作流程如下：

- （1）首先使用字典对象 **Dictionary** 保存姓名中的汉字。
- （2）将字典对象中的汉字填充到“汉字库”工作表的单元格区域。
- （3）使用 **Sort** 方法按笔画排序“汉字库”中的汉字。
- （4）删除字典对象中原有的数据，重新将排序后的“汉字库”工作表中的数据写入字典对象中，并为每个汉字添加顺序号。
- （5）读取“姓名”工作表中每个姓名，从字典对象中查询每个字的顺序号，对每个名字生成一个序列码字符串，将“姓名”和序列码字符串保存到一个二维数组中。
- （6）对二维数组进行排序，得到按笔画排序的姓名。
- （7）将排序后的数组填充到“姓名”工作表中，得到如图 12-52 所示的结果。

3. 编写代码

“按笔画排序”按钮的 VBA 代码如下，该子过程的代码较长，可参考关键技术中介绍的工作程序理解每一部分的作用。

Sub 按笔画排序()

Dim ds As Scripting.Dictionary '字典对象

Dim r As Long, i As Long, j As Integer

Dim c As String, xm As String, c1 As String

Dim str1 As String, n As Long

Dim arr, arr1()

Application.ScreenUpdating = False

Set ds = CreateObject("Scripting.Dictionary") '创建数据字典对象

With Worksheets("姓名")

 r = .Range("A1").End(xlDown).Row

 On Error Resume Next

 For i = 1 To r

 str1 = .Cells(i, 1).Value '获取单元格的姓名

 For j = 1 To Len(str1) '将字符串拆分为单个汉字

 s = Mid(str1, j, 1)

 If s <> " " Then

 ds.Add s, s '添加字典中

 If Err <> 0 Then Err.Clear

 End If

```

Next

Next

On Error GoTo 0

End With

r = ds.Count                '字典中的条目数量

With Worksheets("汉字库")

    .Columns(1).Clear        '清除 A 列

    .Range("A1").Resize(r, 1) = _

        WorksheetFunction.Transpose(ds.Items) '将字典中的数字填充到 A 列

r = .Range("A1").End(xlDown).Row

With .Sort                    '对 A 列按笔画排序

    .SetRange Range("A1:A" & r)

    .Header = xlGuess

    .MatchCase = False

    .Orientation = xlTopToBottom

    .SortMethod = xlStroke

    .Apply

End With

arr = .Range("A1:A" & r)

End With

```

```

ds.RemoveAll                                '删除字典中的所有数据

For i = 1 To r                                '把汉字添加到字典中
    ds.Add arr(i, 1), i
Next

With Worksheets("姓名")
    r = .Range("A1").End(xlDown).Row

    ReDim arr1(1 To r, 1 To 2)                '生定义数组

    For i = 1 To r
        c = .Cells(i, 1)                    '获取单元格的值
        arr1(i, 1) = c                      '保存到数组中

        xm = Replace(Replace(Replace(c, " ", ""), " ", ""), " (", "(")

        '删除空格，全角括号换为半角括号

        xm = Left(xm, InStr(xm & "(", "(") - 1) '去掉括号及括号中的字符

        c1 = ""

        For j = 1 To Len(xm)                '从字典中查询生成序列码字符串
            c1 = c1 & CStr(Format(ds(Mid(xm, j, 1)), "0000"))
        Next

        arr1(i, 2) = c1                    '保存姓名的序列码字符串
    Next
Next

```



```

For i = 1 To r - 1          '双循环排序

    For j = i + 1 To r

        If arr1(i, 2) > arr1(j, 2) Then    '按姓名的序列码字符串比较

            t1 = arr1(i, 1)                '交换数据

            t2 = arr1(i, 2)

            arr1(i, 1) = arr1(j, 1)

            arr1(i, 2) = arr1(j, 2)

            arr1(j, 1) = t1

            arr1(j, 2) = t2

        End If

    Next

Next

.Range("A1:A" & r) = arr1        '将排序后的数组填充到单元格区域

End With

Application.ScreenUpdating = True

End Sub

```

12.4 数据筛选

在 Excel 2007 中，在“开始”选项卡的“编辑”组中单击“排序和筛选”按钮，从下拉的菜单按钮中选择相应的命令即可进行数据筛选操作。在 VBE 中，可使用 `AutoFilter` 方法进行自动筛选操作，使用 `AdvancedFilter` 方法可进行高级筛选操作，本节实例演示数据筛选的 VBA 代码。

例 278 用VBA进行简单筛选

1. 案例说明

打开本例工作簿如图 12-54 所示，单击工作表左上角的“筛选”按钮弹出如图 12-55 所示的对话框，在对话框中输入筛选条件“财务部”，单击“确定”按钮，工作表中将自动出现自动筛选下拉箭头，并且只显示“部门”为“财务部”的数据，如图 12-56 所示。

在如图 12-55 所示的“筛选”对话框中不输入任何值，直接单击“确定”按钮即可显示全部数据。



图 12-54 用 VBA 筛选数据

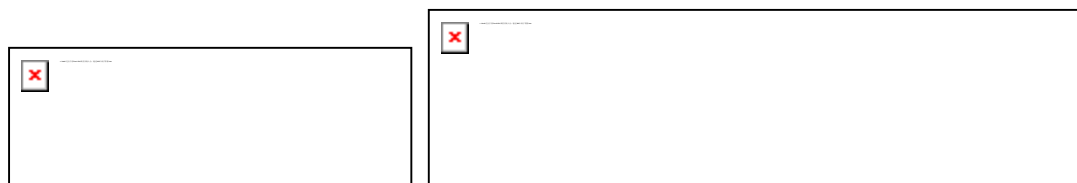


图 12-55 输入筛选条件

图 12-56 筛选结果

2. 关键技术

使用 Range 对象的 AutoFilter 方法，可对 Range 区域的数据中使用“自动筛选”筛选一个列表。该方法的语法如下：

表达式.AutoFilter(Field, Criteria1, Operator, Criteria2, VisibleDropDown)

各参数的含义如下：

- **Field**: 相对于作为筛选基准字段（从列表左侧开始，最左侧的字段为第一个字段）的字段的整型偏移量。

- **Criteria1:** 筛选条件，为一个字符串。使用“=”可查找空字段，或者使用“<>”查找非空字段。如果省略该参数，则搜索条件为 All。如果将 Operator 设置为 xlTop10Items，则 Criteria1 指定数据项个数（例如，“10”）。
- **Operator:** 指定筛选类型，可用常量如表 12-2 所示。

表 12-2 筛选类型

名 称	值	描 述
xlAnd	1	条件 1 和条件 2 的逻辑与
xlBottom10Items	4	显示最低值项（条件 1 中指定的项数）
xlBottom10Percent	6	显示最低值项（条件 1 中指定的百分数）
xlFilterCellColor	8	单元格颜色
xlFilterDynamic	11	动态筛选
xlFilterFontColor	9	字体颜色
xlFilterIcon	10	筛选图标
xlFilterValues	7	筛选值
xlOr	2	条件 1 和条件 2 的逻辑或
xlTop10Items	3	显示最高值项（条件 1 中指定的项数）
xlTop10Percent	5	显示最高值项（条件 1 中指定的百分数）

- **Criteria2:** 第二个筛选条件（一个字符串）。与 Criteria1 和 Operator 一起组合成复合筛选条件。
- **VisibleDropDown:** 如果为 True，则显示筛选字段的自动筛选下拉箭头。如果为 False，则隐藏筛选字段的自动筛选下拉箭头。默认值为 True。

—



- 如果忽略全部参数，此方法仅在指定区域切换自动筛选下拉箭头的显示。
-

3. 编写代码

“筛选”按钮的 VBA 代码如下：

Sub 筛选()

Dim str1 As String

str1 = Application.InputBox(prompt:="请输入要筛选的部门名称（空字符将显示全部数据）：", _

```

Title:="筛选", Type:=2)

If str1 = "False" Then Exit Sub

If str1 = "" Then

Worksheets("Sheet1").Range("A1").AutoFilter field:=3

Else

Worksheets("Sheet1").Range("A1").AutoFilter _

field:=3, _

Criteria1:=str1

End If

End Sub

```

以上代码首先要求用户输入筛选条件，接着判断用户输入的是否为空，若为空，则显示全部数据，若输入的筛选条件不为空，则筛选等于输入条件的数据。

例 279 用VBA进行高级筛选

1. 案例说明

打开本例工作簿如图 12-57 所示，在下方的“条件区域”部分输入条件，再单击左上角的“高级筛选”按钮，即可按条件区域中输入的条件对数据进行高级筛选，得到如图 12-58 所示的结果。

如果在条件区域删除数据（例如删除图 12-57 下方的“财务部”和“>=1400”），再单击“高级筛选”按钮，工作表将显示全部数据（取消高级筛选功能）。



图 12-57 高级筛选



图 12-58 高级筛选结果

若在条件区域不同行输入条件，则将采用逻辑或关系筛选数据（即只要满足一列条件即可），如图 12-59 所示，可显示“人事部”或“基础工资”大于 1400 的数据。

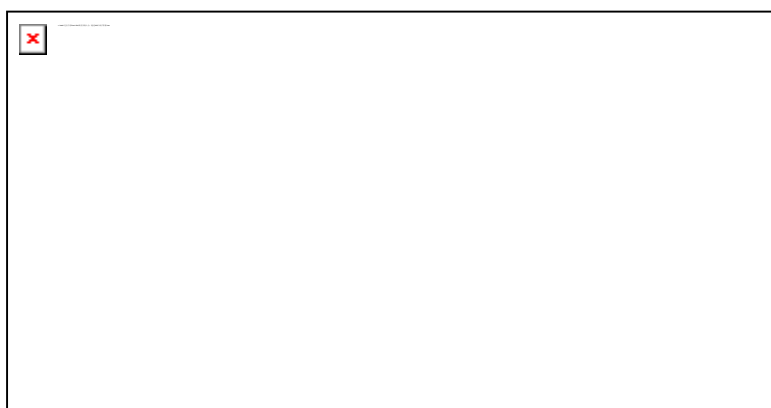


图 12-59 逻辑或筛选

2. 关键技术

Excel 的高级筛选可用 VBA 代码来实现，使用 Range 对象的 AdvancedFilter 方法即可进行高级筛选。

高级筛选必须在工作表中定义一个条件区域，通过该条件从列表中筛选或复制数据。如果初始选定区域为单个单元格，则使用单元格的当前区域。AdvancedFilter 方法的语法格式如下：

表达式.AdvancedFilter(Action, CriteriaRange, CopyToRange, Unique)

该方法各参数的含义如下：

- **Action:** 指定是否就地复制或筛选列表，可使用常量 xlFilterCopy（将筛选出的数据复制到新位置）或 xlFilterInPlace（保留数据不动）。
- **CriteriaRange:** 条件区域。如果省略该参数，则没有条件限制。
- **CopyToRange:** 如果 Action 为 xlFilterCopy，则该参数为复制行的目标区域。否则，忽略该参数。
- **Unique:** 如果为 True，则只筛选唯一记录。如果为 False，则筛选符合条件的所有记录。默认值为 False。

3. 编写代码

“高级筛选”按钮的 VBA 代码如下：

```
Sub 高级筛选()
```

```
Dim rng As Range, rng1 As Range
```

```
Application.ScreenUpdating = False
```

```
Application.Calculation = xlCalculationManual '手动重算
```

```
Set rng = Worksheets("Sheet1").Range("A19").CurrentRegion
```

```

Set rng = rng.Offset(1, 0).Resize(rng.Rows.Count - 1, rng.Columns.Count)

Set rng1 = Worksheets("Sheet1").Range("A1").CurrentRegion

Set rng1=rng1.Offset(1,0).Resize(rng1.Rows.Count-1, rng1.Columns.Count)

rng1.AdvancedFilter Action:=xlFilterInPlace, CriteriaRange:=rng

Application.Calculation = xlCalculationAutomatic '自动重算

Application.ScreenUpdating = True

End Sub

```

以上代码首先获取工作表中条件区域和筛选数据区域的引用，最后使用 **AdvancedFilter** 方法对数据区域进行筛选。

例 280 筛选非重复值

1. 案例说明

打开本例工作簿，单击工作表中的“生成随机数”按钮，将在工作表的 A 列生成 1000 个随机数，再单击“筛选非重复值”按钮，可将左侧生成的 1000 个随机数中的非重复数筛选并复制到 B 列中，如图 12-60 所示。

2. 关键技术

本例使用 **Range** 对象的 **AdvancedFilter** 方法筛选非重复值，有关该方法的介绍参见上例中的内容。

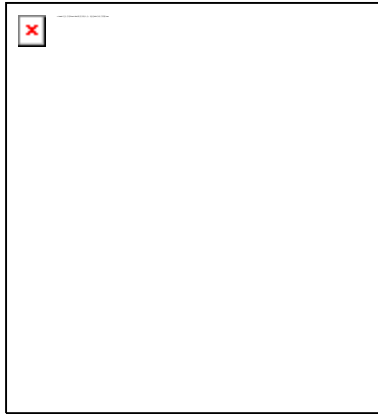


图 12-60 筛选非重复值

3. 编写代码

(1) “生成随机数”按钮的 VBA 代码如下：

```
Sub 生成随机数()  
  
    Dim i As Integer  
  
    Application.ScreenUpdating = False  
  
    Randomize  
  
    With ActiveSheet  
  
        For i = 2 To 1001  
  
            .Cells(i, 1) = Int(Rnd * 1000 + 1)  
  
        Next  
  
    End With  
  
    Application.ScreenUpdating = True  
  
End Sub
```

(2) “筛选非重复值”按钮的 VBA 代码如下：


```

Sub 筛选非重复值()

    Dim i As Long, rng As Range

    Application.ScreenUpdating = False

    With ActiveSheet

        i = .Range("A1").End(xlDown).Row

        If i > 1001 Then Exit Sub

        Set rng = .Range(Cells(2, 1), Cells(i, 1))

        .Columns("B").ClearContents

        rng.AdvancedFilter Action:=xlFilterCopy, _
            CopyToRange:=.Range("B2"), Unique:=True

    End With

    Application.ScreenUpdating = True

End Sub

```

例 281 取消筛选

1. 案例说明

打开本例工作簿如图 12-61 所示，在如图所示工作表中设置了自动筛选，单击“取消筛选”按钮，当前工作簿中每个工作表中的自动筛选都将取消，如图 12-62 所示。

2. 关键技术

如果当前在工作表上显示有“自动筛选”下拉箭头，则 `AutoFilterMode` 属性值为 `True`。设置该属性值为 `False` 可取消自动筛选状态。

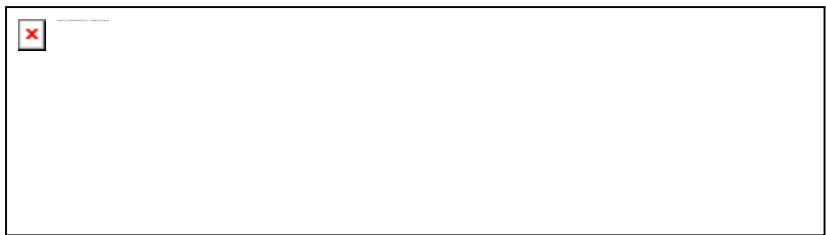


图 12-61 筛选状态的工作表



图 12-62 取消筛选的工作表

—



— 不能将该属性设置为 `True`。使用 `AutoFilter` 方法可筛选列表并显示下拉箭头。

—

3. 编写代码

“取消筛选”按钮的 VBA 代码如下：

```
Sub 取消筛选()  
  
    Dim ws1 As Worksheet  
  
    For Each ws1 In Worksheets  
  
        ws1.AutoFilterMode = False  
  
    Next
```

End Sub