IBM iSeries (AS/400) 开发从入门到精通教程

前言

IBM iSeries（前身为 AS/400、System i）是 IBM 开发的集成商业计算平台，以其卓越的稳定性、安全性和集成性闻名于世。自1988年首次发布以来，它已经成为全球众多企业核心业务系统的基石，尤其在金融、制造、零售和物流等行业。

本教程旨在为 IBM iSeries 开发提供一份全面、系统、深入的指南，涵盖从基础概念到高级应用的各个方面。无论您是初次接触 IBM i 的新手，还是希望提升技能的经验开发者，本教程都将为您提供宝贵的知识和实践经验。

教程特色

- **系统性**：从基础概念到高级主题，循序渐进
- **实用性**：大量实战案例和代码示例
- **全面性**：涵盖 RPG IV、CLLE、SQL、Personal Communication 等核心技术
- **图文并茂**：详细截图和图表说明

目标读者

- 初学 IBM i 开发的程序员
- 希望从 RPG III 迁移到 RPG IV/ILE 的开发者
- 系统管理员和运维人员
- 对遗留系统现代化感兴趣的技术人员

目录

第五部分：高级主题与实战案例

- 第20章：DB2 for i 数据库编程
- 第21章：Web 服务与现代化开发
- 第22章：系统集成与 API
- 第23章：性能调优与最佳实践
- 第24章：完整项目实战：ERP 订单管理系统

第一部分：IBM i 基础入门

第1章：IBM i 系统概述

1.1 IBM i 的历史与演进

IBM i 操作系统有着悠久而辉煌的历史，其发展历程反映了企业计算技术的不断演进：

AS/400 时代（1988-2000）

1988年，IBM 推出了 Application System/400（AS/400），这是一个革命性的系统，它将硬件、操作系统、数据库和中间件紧密集成在一起。AS/400 采用了独特的技术架构：

- **Technology Independent Machine Interface (TIMI)**：允许应用程序独立于底层硬件运行
- **Integrated DB2 数据库**：内置关系型数据库管理系统
- **单一级别存储**：简化了存储管理，自动处理物理和虚拟存储之间的映射

iSeries 时代（2000-2006）

2000年，IBM 将 AS/400 更名为 iSeries，标志着系统向 e-business 和互联网时代的转型：

- 增强了对 Web 应用的支持
- 引入了更多的开放标准
- 改进了 Java 和 Linux 支持

System i 时代（2006-2008）

2006年，IBM 推出了 System i 品牌，进一步强调了系统的集成特性：

- 统一了 i5/OS 操作系统品牌
- 增强了虚拟化能力

- 改进了 POWER 处理器支持

IBM i 时代（2008至今）

2008年至今，系统正式更名为 IBM i：

- 每两年发布一个新版本
- 持续增强现代化开发能力
- 强化了开源技术支持
- 增强了云部署能力

1.2 IBM i 的核心特性

1.2.1 集成性（Integration）

IBM i 最显著的特点是其高度集成性：

这种集成架构带来了以下优势：

1. **简化管理**：单一供应商支持，减少了兼容性问题
2. **降低成本**：无需额外购买数据库、安全软件等
3. **提高可靠性**：各组件经过优化配合，运行更加稳定
4. **简化开发**：开发人员可以专注于业务逻辑

1.2.2 稳定性与可靠性

IBM i 以其卓越的稳定性著称：

- **99.997% 的可用性**：年平均停机时间不超过15分钟

- **自动内存管理**：防止内存泄漏和溢出
- **硬件错误恢复**：自动检测和纠正硬件错误
- **日志与恢复机制**：完善的数据保护和恢复能力

1.2.3 安全性

IBM i 提供企业级的安全保护：

安全特性	说明
对象级安全	每个对象都有独立的权限控制
用户认证	支持多因素认证、LDAP 集成
加密支持	内置 SSL/TLS、字段级加密
审计功能	全面的系统活动日志记录
病毒防护	内置病毒扫描和防护机制

1.2.4 技术独立性

TIMI（Technology Independent Machine Interface）是 IBM i 的核心技术之一：

这种架构的优势：

- 应用程序可以跨硬件平台迁移

- 无需重新编译即可在新硬件上运行
- 保护了企业的软件投资

1.3 IBM i 的应用场景

IBM i 广泛应用于以下领域：

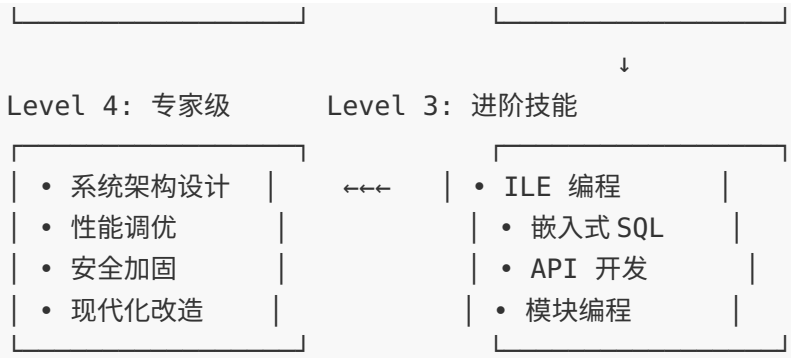
金融行业
- 核心银行系统
- 证券交易系统
- 保险理赔系统
- 支付处理平台

制造业
- ERP 系统（SAP、Oracle JD Edwards）
- 供应链管理
- 库存控制
- 生产计划系统

零售行业
- 销售点系统（POS）
- 库存管理
- 客户关系管理
- 电商平台后端

物流行业
- 仓储管理系统
- 运输调度系统
- 订单追踪平台
- 供应链可视化

1.4 学习路径规划



第2章：系统架构与核心概念

2.1 IBM i 系统架构

2.1.1 分层架构详解

IBM i 采用分层架构设计，每一层都有明确的职责：

应用层（Application Layer）

应用层是用户直接交互的层面，支持多种编程语言和开发环境：

- **传统语言**：RPG IV、COBOL、CL
- **现代语言**：C、C++、Java
- **脚本语言**：PHP、Python、Node.js、Ruby
- **Web 技术**：JavaScript、HTML5、CSS3

中间件层（Middleware Layer）

提供应用程序运行所需的各种服务：

```
│  集成服务      │ WebSphere Message Broker                    │
└──────────────┴─────────────────────────────────────────────┘
```

数据库层（Database Layer）

Db2 for i 是系统内置的关系型数据库：

- **完全集成**：与操作系统深度集成
- **高性能**：优化的查询处理器
- **高可用性**：支持数据复制和高可用配置
- **安全性**：字段级加密、审计功能

操作系统层（OS Layer）

IBM i 操作系统提供：

- 进程和内存管理
- 文件系统管理
- 安全控制
- 网络通信
- 设备管理

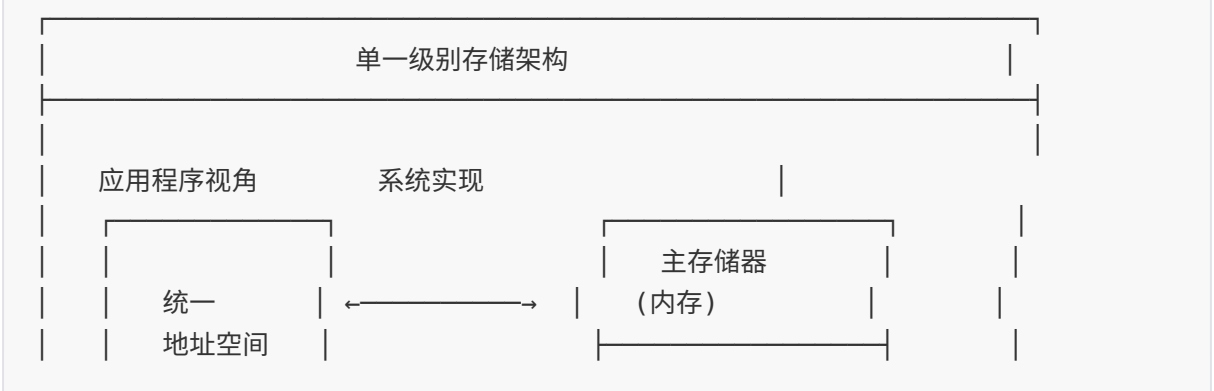
硬件层（Hardware Layer）

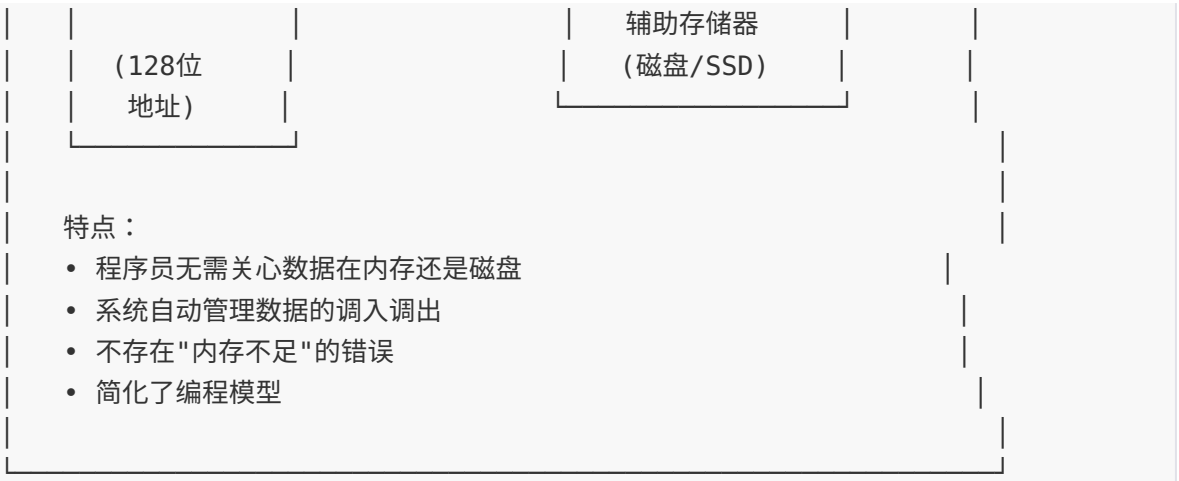
基于 IBM Power Systems：

- **POWER 处理器**：高性能、低功耗
- **虚拟化支持**：PowerVM 虚拟化技术
- **I/O 子系统**：专用 I/O 处理器

2.1.2 单一级别存储（Single-Level Storage）

单一级别存储是 IBM i 的核心创新之一：





2.2 核心概念

2.2.1 库（Library）

库是 IBM i 中对象的基本组织单元，类似于其他系统中的"文件夹"或"目录"：

库类型：

库类型	说明	示例
系统库	包含系统对象	QSYS, QHLPSYS

库类型	说明	示例
产品库	安装软件时创建	QGPL, QTEMP
当前库	当前操作的默认库	可更改
用户库	用户创建的库	自定义名称

库列表（Library List）

库列表定义了系统查找对象的顺序：

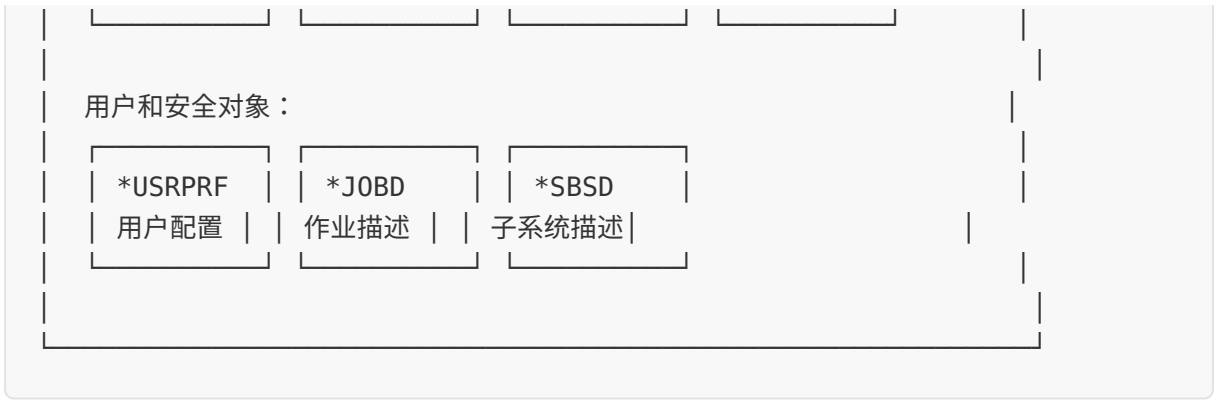
库列表结构：

```
┌──────────────────────────────────────────────────────┐
│  库列表类型    │  库名称          │  说明             │
├──────────────────────────────────────────────────────┤
│  系统部分      │  QSYS            │  系统对象         │
│                │  QHLPSYS         │  帮助系统         │
│                │  QUSRSYS         │  用户系统对象     │
├──────────────────────────────────────────────────────┤
│  用户部分      │  QTEMP           │  临时对象         │
│                │  PRODLIB         │  应用程序库       │
│                │  *CURLIB         │  当前库           │
└──────────────────────────────────────────────────────┘
```

2.2.2 对象（Object）

IBM i 中的所有资源都是对象，每种对象类型都有特定的用途：

```
┌──────────────────────────────────────────────────────┐
│                    IBM i 对象类型                    │
├──────────────────────────────────────────────────────┤
│                                                      │
│  编程相关对象：                                      │
│  ┌──────────┐ ┌──────────┐ ┌──────────┐ ┌──────────┐ │
│  │ *PGM     │ │ *MODULE  │ │ *SRVPGM  │ │ *BNDDIR  │ │
│  │ 程序     │ │ 模块     │ │ 服务程序 │ │ 绑定目录 │ │
│  └──────────┘ └──────────┘ └──────────┘ └──────────┘ │
│                                                      │
│  数据库对象：                                        │
│  ┌──────────┐ ┌──────────┐ ┌──────────┐ ┌──────────┐ │
│  │ *FILE    │ │ *DTAARA  │ │ *DTAQ    │ │ *JRN     │ │
│  │ 文件     │ │ 数据区   │ │ 数据队列 │ │ 日志     │ │
```

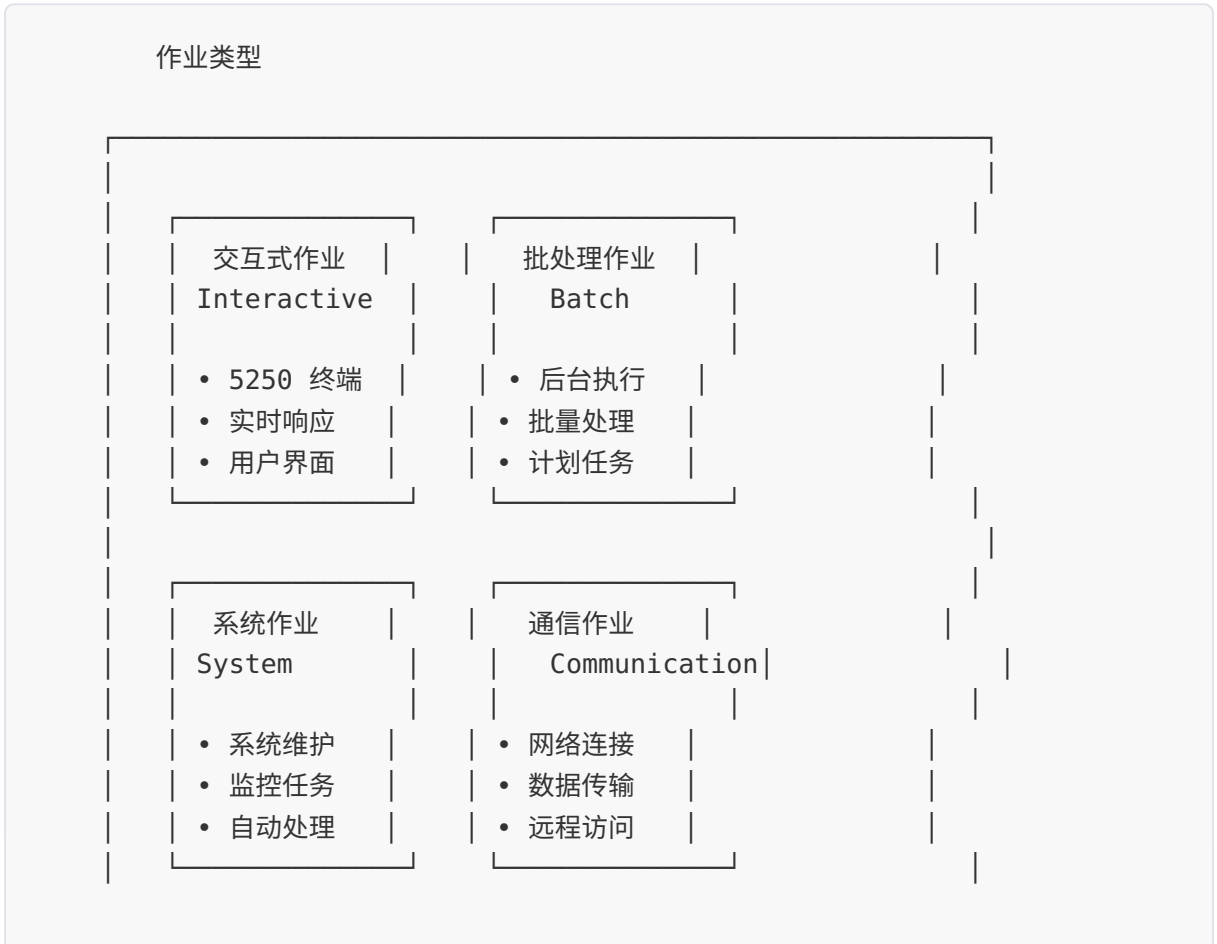


对象命名规则：

- 最大长度：10 个字符
- 可包含：字母、数字、特殊字符（@、#、$、_）
- 必须以字母或特殊字符开头
- 不区分大小写（但系统会转换为大写）

2.2.3 作业（Job）

作业是 IBM i 中工作的基本单元：



```
    │                                                          │
    └──────────────────────────────────────────────────────────┘
```

作业结构：

```
作业组成部分：
┌──────────────────────────────────────────────────────────────┐
│                          作业 (Job)                          │
├──────────────────────────────────────────────────────────────┤
│  作业定义：                                                  │
│  ├── 作业名 (Job Name)                                       │
│  ├── 用户名 (User Name)                                      │
│  ├── 作业号 (Job Number)                                     │
│  └── 作业描述 (Job Description)                              │
│                                                              │
│  运行时环境：                                                │
│  ├── 库列表 (Library List)                                   │
│  ├── 输出队列 (Output Queue)                                 │
│  ├── 消息队列 (Message Queue)                                │
│  └── 运行优先级 (Run Priority)                               │
│                                                              │
│  资源：                                                      │
│  ├── 内存分配                                                │
│  ├── 处理器时间                                              │
│  └── I/O 设备                                                │
└──────────────────────────────────────────────────────────────┘
```

2.3 文件系统

IBM i 支持多种文件系统：

2.3.1 库文件系统（QSYS.LIB）

传统的 IBM i 文件系统，存储所有 IBM i 对象：

```
QSYS.LIB 结构：
/QSYS.LIB/库名.LIB/对象名.对象类型

示例：
/QSYS.LIB/QSYS.LIB/QCMD.PGM
/QSYS.LIB/MYLIB.LIB/CUSTOMER.FILE
/QSYS.LIB/MYLIB.LIB/MYPGM.PGM
```

2.3.2 集成文件系统（IFS）

兼容 POSIX 的文件系统，支持目录层次结构：

```
IFS 结构：
/
├── QSYS.LIB/           # 库文件系统
├── QOpenSys/           # 开放系统接口
│   ├── usr/
│   ├── bin/
│   └── lib/
├── home/               # 用户主目录
│   ├── user1/
│   └── user2/
├── tmp/                # 临时文件
└── dev/                # 设备文件
```

2.3.3 其他文件系统

文件系统	说明
QOPT	光学设备文件系统
QNTC	Windows 网络文件系统
NFS	网络文件系统
UDFS	用户定义文件系统

第3章：对象与库管理

3.1 库管理命令

3.1.1 创建库

```
/* 创建库的基本命令 */
CRTLIB LIB(MYLIB) TEXT('我的应用程序库')

/* 创建库并指定辅助存储池 */
```

```
CRTLIB LIB(PRODLIB) ASP(2) TEXT('生产库在 ASP 2')

/* 创建库并指定类型 */
CRTLIB LIB(TESTLIB) TYPE(*TEST) TEXT('测试库')
```

CRTLIB 参数说明：

参数	说明	默认值
LIB	库名称	必需
ASP	辅助存储池	*SYSTEM
TYPE	库类型 (PROD/TEST)	*PROD
TEXT	描述文本	空白

3.1.2 管理库

```
/* 显示库信息 */
DSPLIB LIB(MYLIB)

/* 显示库内容 */
DSPLIBD LIB(MYLIB)

/* 更改库 */
CHGLIB LIB(MYLIB) TEXT('更新后的描述')

/* 删除库 */
DLTLIB LIB(TESTLIB)

/* 添加库到库列表 */
ADDLIBLE LIB(MYLIB) POSITION(*LAST)

/* 从库列表中移除 */
RMVLIBLE LIB(MYLIB)

/* 更改当前库 */
CHGCURLIB CURLIB(MYLIB)
```

3.2 对象管理

3.2.1 对象的复制与移动

```
/* 复制对象 */
CRTDUPOBJ OBJ(MYPGM) FROMLIB(Srclib) OBJTYPE(*PGM) +
          TOLIB(Dstlib) NEWOBJ(MYPGM2)

/* 移动对象 */
MOVOBJ OBJ(MYPGM) OBJTYPE(*PGM) TOLIB(NEWLIB)

/* 重命名对象 */
RNMOBJ OBJ(MYPGM) OBJTYPE(*PGM) NEWOBJ(MYPGMNEW)
```

3.2.2 对象的保存与恢复

```
/* 保存库 */
SAVLIB LIB(MYLIB) DEV(TAP01)

/* 保存对象 */
SAVOBJ OBJ(MYPGM) LIB(MYLIB) OBJTYPE(*PGM) DEV(TAP01)

/* 恢复库 */
RSTLIB SAVLIB(MYLIB) DEV(TAP01)

/* 恢复对象 */
RSTOBJ OBJ(MYPGM) SAVLIB(MYLIB) DEV(TAP01) +
       RSTLIB(MYLIB) OBJTYPE(*PGM)
```

3.3 库列表管理

3.3.1 查看库列表

```
/* 显示库列表 */
DSPLIBL

/* 显示库列表输出示例 */
                      Display Library List

  Library      Type      Description
  ____________ __________ ____________________________________
```

```
  QSYS         SYS       System Library
  QHLPSYS      SYS       Help System Library
  QUSRSYS      SYS       User System Library
  QTEMP        USR       Temporary Library
  PRODLIB      USR       Production Library
  QGPL         USR       General Purpose Library
```

3.3.2 修改库列表

```
/* 修改系统值中的初始库列表 */
CHGSYSVAL SYSVAL(QUSRLIBL) VALUE('MYLIB PRODLIB')

/* 在作业级别修改库列表 */
CHGLIBL LIB(MYLIB PRODLIB QTEMP QGPL)

/* 添加到库列表 */
ADDLIBLE LIB(NEWLIB) POSITION(*FIRST)

/* 从库列表中移除 */
RMVLIBLE LIB(OLDLIB)
```

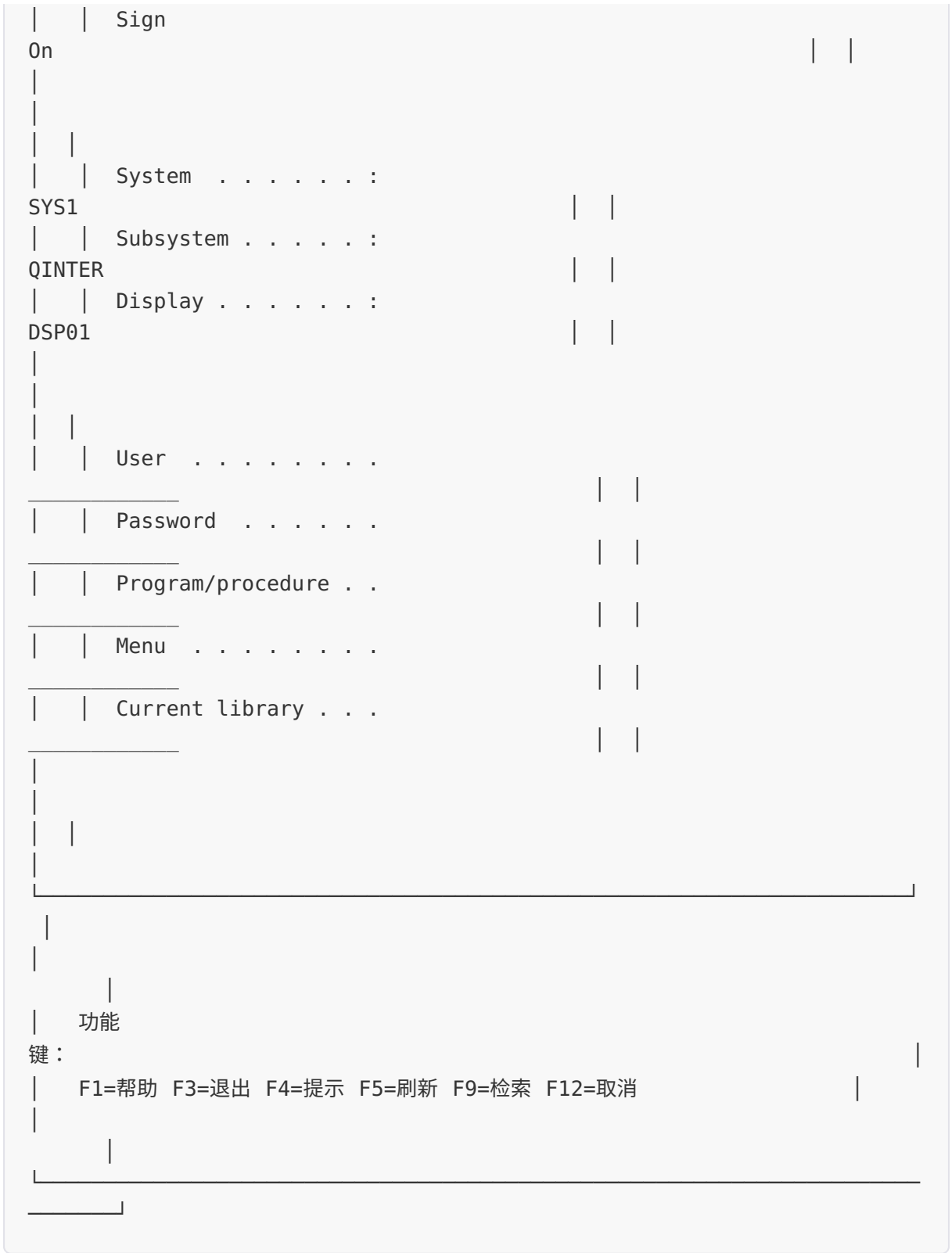
第4章：用户界面与基本操作

4.1 5250 工作站界面

4.1.1 5250 终端介绍

5250 工作站界面是 IBM i 传统的字符型用户界面：

```
┌──────────────────────────────────────────────────────────────────────────────
──────┐
│                              5250 工作站界
面                                    │
├──────────────────────────────────────────────────────────────────────────────
──────┤
│
      │
│
┌──────────────────────────────────────────────────────────────────────────────┐
 │
```



4.1.2 基本导航

常用功能键：

功能键	作用
F1	帮助
F3	退出
F4	提示
F5	刷新
F9	检索上一条命令
F12	返回/取消
F13	重复
F24	显示更多功能键

4.2 命令输入与使用

4.2.1 命令语法

IBM i 命令遵循一致的命名约定：

```
命令格式：动词 + 对象类型

动词（操作）：
  CRT = Create    (创建)
  DLT = Delete    (删除)
  CHG = Change    (更改)
  DSP = Display   (显示)
  WRK = Work with (处理)
  STR = Start     (开始)
  END = End       (结束)
  ADD = Add       (添加)
  RMV = Remove    (移除)

对象类型：
  LIB = Library   (库)
  PF  = Physical File (物理文件)
  LF  = Logical File (逻辑文件)
  PGM = Program   (程序)
  CMD = Command   (命令)
```

```
  USRPRF = User Profile (用户配置)

示例：
  CRTLIB = Create Library
  DLTF   = Delete File
  CHGUSRPRF = Change User Profile
  DSPJOB = Display Job
```

4.2.2 命令输入技巧

```
/* 使用 F4 提示 */
在命令行输入部分命令，按 F4 获得参数提示

/* 命令缩写 */
CRTLIB LIB(MYLIB) → CRTLIB MYLIB

/* 使用 + 续行 */
CRTLIB LIB(MYLIB) +
       TEXT('这是一个长描述文本')

/* 使用 & 重复前一个值 */
/* 在参数位置输入 & 可使用前一个命令的值 */
```

4.3 常用系统界面

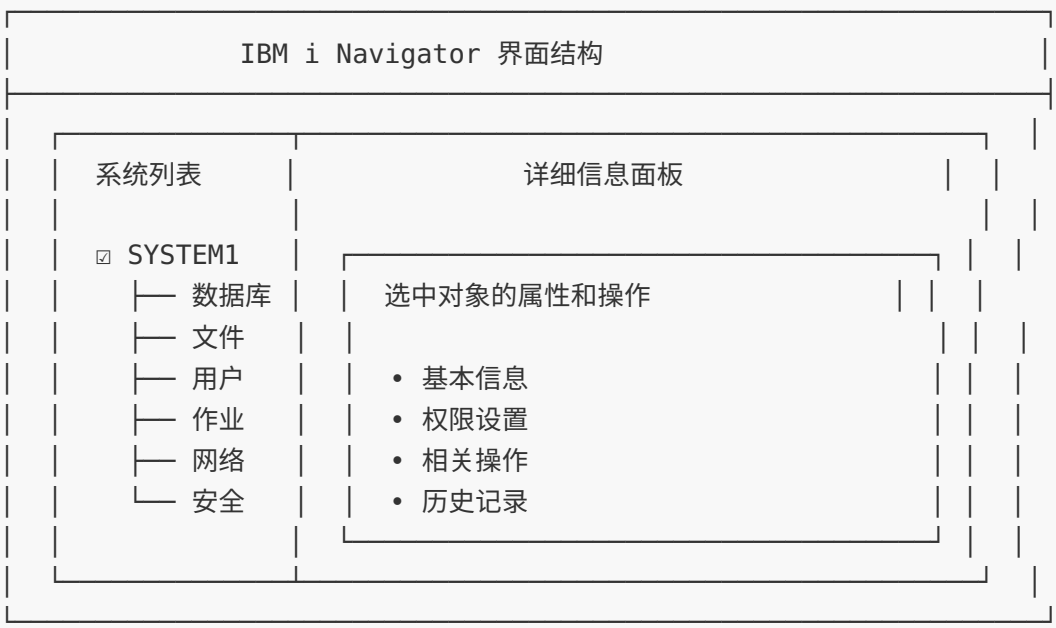
4.3.1 主菜单

```
          主菜单

  Select one of the following:

       1. User tasks
       2. Office tasks
       3. General system tasks
       4. Files, libraries, and folders
       5. Programming
       6. Communications
       7. Configuration
       8. Service
       9. Work management

  Selection or command
```

```
  ===> ____________________________________________________________

  F3=Exit    F4=Prompt    F9=Retrieve    F13=Information Assistant
```

4.3.2 操作导航器（Operations Navigator）

现代 IBM i 推荐使用 IBM i Navigator 或 Access Client Solutions：



第二部分：Personal Communication 终端使用指南

第5章：Personal Communication 安装与配置

5.1 软件概述

IBM Personal Communications（简称 PCOM）是 IBM 提供的 5250/3270 终端仿真软件，用于从 Windows 平台连接到 IBM i 和 IBM z 系统。

5.1.1 主要功能

```
┌──────────────────────────────────────────────────────────────────────────────
┐
```

5.1.2 系统要求

要求项	最低配置	推荐配置
操作系统	Windows 10	Windows 11
内存	4 GB	8 GB
硬盘空间	500 MB	1 GB
网络	TCP/IP 连接	高速网络
显示	1024x768	1920x1080

5.2 安装步骤

5.2.1 获取软件

Personal Communications 可通过以下途径获取：

1. **IBM 官方渠道**：IBM 客户门户
2. **IBM i Access Client Solutions**：包含简化版 PCOM
3. **IBM Developer**：开发者版本

5.2.2 安装过程

安装步骤：

```
┌──────────────────────────────────────────────────────────────┐
│ 1. 运行安装程序                                              │
│    └── setup.exe                                             │
│                                                              │
│ 2. 选择安装类型                                              │
│    ├── 典型安装 (推荐)                                       │
│    ├── 自定义安装                                            │
│    └── 完整安装                                              │
│                                                              │
│ 3. 配置安装目录                                              │
│    └── C:\Program Files\IBM\Personal Communications\         │
│                                                              │
│ 4. 选择组件                                                  │
│    ├── 5250 仿真器                                           │
│    ├── 3270 仿真器                                           │
│    ├── API 支持                                              │
│    └── 文档                                                  │
│                                                              │
│ 5. 完成安装                                                  │
│    └── 重启系统（如需要）                                    │
└──────────────────────────────────────────────────────────────┘
```

5.3 配置 5250 会话

5.3.1 创建新会话

配置新会话步骤：

1. 启动 Personal Communications

```
   Start → All Programs → IBM Personal Communications
   → Start or Configure Sessions

2. 点击 "New Session"

3. 选择连接类型
   ┌─────────────────────────────────────────────┐
   │  Type of Host:  ⊙ IBM iSeries (5250)        │
   │                                             │
   │  Interface:     ⊙ LAN                       │
   │                                             │
   │  Attachment:    ⊙ Telnet5250                │
   └─────────────────────────────────────────────┘

4. 配置连接参数
   ┌─────────────────────────────────────────────┐
   │  Primary Host Name/IP:  [192.168.1.100]     │
   │                                             │
   │  Port:                  [23]                │
   │                                             │
   │  Auto Reconnect:        ☑                   │
   │                                             │
   │  Connection Timeout:    [30] 秒             │
   └─────────────────────────────────────────────┘

5. 配置工作站 ID
   ┌─────────────────────────────────────────────┐
   │  Workstation ID:        [DSP01]             │
   │                                             │
   │  Device Name:           [QPADEV0001]        │
   │                                             │
   │  Device Pool:           [QINTER]            │
   └─────────────────────────────────────────────┘

6. 保存会话
   File → Save As → 输入会话名称（如：PROD_SESSION）
```

5.3.2 配置文件详解

会话配置文件（.WS）包含以下主要部分：

```
; PCOM 会话配置文件示例
[Communications]
; 连接设置
Transport=TCP
```

```
NetworkProtocol=Telnet
HostIP=192.168.1.100
HostPort=23
AutoConnect=1

[5250]
; 5250 特定设置
WorkstationID=DSP01
DeviceName=QPADEV0001
CodePage=935
CharacterSet=65535

[Display]
; 显示设置
ScreenSize=27x132
WindowState=Normal
ColorScheme=Green
FontName=IBM 3270
FontSize=14

[Keyboard]
; 键盘设置
KeyboardMap=IBMDefault
EnterKeyAction=FieldExit

[Printing]
; 打印设置
PrintDestination=HPT
PrintDriver=IBMHPTRedirect
```

5.4 高级配置选项

5.4.1 SSL/TLS 安全配置

```
SSL 连接配置步骤：

1. 获取服务器证书
 联系 IBM i 管理员获取 CA 证书

2. 导入证书
   ┌──────────────────────────────────────────────┐
   │ Edit → Preferences → Cryptography            │
   │                                              │
   │ Click "Import Certificate"                   │
```

```
    | Select certificate file (.cer/.crt/.pem)    |
    | Click "OK"                                  |
    └─────────────────────────────────────────────┘
```

3. 配置 SSL 连接

```
    ┌─────────────────────────────────────────────┐
    | Customize Communications → Link Parameters  |
    |                                             |
    | Enable SSL:              ☑                  |
    | SSL Version:             TLS 1.2            |
    | Certificate Store:       Personal           |
    | Verify Server Cert:      ☑                  |
    └─────────────────────────────────────────────┘
```

4. 更改端口
 默认 Telnet 端口: 23
   SSL Telnet 端口: 992

5.4.2 会话属性优化

性能优化设置：

1. 显示优化

```
    ┌─────────────────────────────────────────────┐
    | Edit → Preferences → Appearance             |
    |                                             |
    | Screen Size:             27 x 132           |
    |   (推荐 27行 x 132列，支持更多内容)          |
    |                                             |
    | Cursor Blink:            ☑                  |
    |                                             |
    | Crosshair Cursor:        □                  |
    |   (禁用可提高性能)                           |
    └─────────────────────────────────────────────┘
```

2. 连接优化

```
    ┌─────────────────────────────────────────────┐
    | Edit → Preferences → Communications         |
    |                                             |
    | Keep Alive Interval:     300 秒             |
    |   (防止空闲断开)                             |
    |                                             |
    | Response Timeout:        60 秒              |
    |                                             |
```

```
    │ Auto Reconnect:          ☑                    │
    │ Retry Count:             10                   │
    └──────────────────────────────────────────────┘
```

3. 键盘优化

```
    ┌──────────────────────────────────────────────┐
    │ Edit → Preferences → Keyboard                │
    │                                              │
    │ Quick Connect Key:       Ctrl+F11            │
    │                                              │
    │ Custom Keymap:           ☑                   │
    │ Map Ctrl+C to Copy:      ☑                   │
    │ Map Ctrl+V to Paste:     ☑                   │
    └──────────────────────────────────────────────┘
```

第6章：5250 终端仿真详解

6.1 5250 工作站界面

6.1.1 屏幕布局

```
┌──────────────────────────────────────────────────────────────────────────────
─────────┐
│  IBM i Main Menu                                          System:
SYS1       │
│
Type . . . . :   │
│
       │
│   Select one of the
following:                                          │
│
       │
│      1. User
tasks                                                    │
│      2. Office
tasks                                                  │
│      3. General system
tasks                                              │
│      4. Files, libraries, and
folders                                         │
```

```
|      5.
Programming                                                          |
|      6.
Communications                                                       |
|      7.
Configuration                                                        |
|      8.
Service                                                              |
|      9. Work
management                                                           |
|     10.
Security                                                             |
|
       |
|  Selection or
command                                                              |
|  ===>
_____________________________________________________________        |
|
       |
|
_____________________________________________________________________
__  |
|  F1=Help   F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve
F12=Cancel      |
|  F13=Information Assistant   F24=More
keys                                  |
└─────────────────────────────────────────────────────────────────────
________┘
```

屏幕元素说明：

标题栏	显示当前菜单名称、系统名称
菜单区	显示可选项列表
命令行	输入命令或选项编号 (===>)
消息行	显示系统消息和错误信息
功能键	显示可用的功能键及其作用

6.1.2 功能键详解

5250 功能键映射：

标准功能键：

按键	功能说明
F1	Help - 显示帮助信息
F3	Exit - 退出当前功能
F4	Prompt - 命令提示
F5	Refresh - 刷新屏幕
F6	Create - 创建（在列表界面）
F7	向上滚动
F8	向下滚动
F9	Retrieve - 检索上一条命令
F10	移动至命令行
F12	Cancel - 取消/返回
F13	Information Assistant
F14	提交作业
F15	工作与服务
F16	主菜单
F17	子菜单
F24	More Keys - 显示更多功能键

在 PCOM 中的等效按键：

PCOM 按键	5250 功能
Escape	Attention 键
Ctrl	功能键前缀（某些配置）
Insert	Insert 切换
Delete	Delete 字符
Home	移至第一个输入字段
End	移至最后一个输入字段
Page Up	F7 (向上滚动)
Page Down	F8 (向下滚动)

6.2 命令行操作

6.2.1 命令输入基础

命令行输入示例：

```
===> CRTLIB LIB(MYLIB) TEXT('My Library')
```

命令结构解析：

参数输入技巧：
1. 使用 F4 获取参数提示
2. 使用 & 引用前一个命令的值
3. 使用 + 续行
4. 使用 ?* 显示所有可能的值

6.2.2 常用系统命令

```
/* 系统信息命令 */
DSPSYSVAL  QSYSVAL(????)     /* 显示系统值 */
DSPJOB     JOB(???????)      /* 显示作业信息 */
DSPUSRPRF  USRPRF(??????)    /* 显示用户配置 */

/* 对象管理命令 */
WRKOBJ     OBJ(QSYS/*ALL) OBJTYPE(*ALL)  /* 处理对象 */
DSPOBJD    OBJ(MYLIB/*ALL) OBJTYPE(*ALL) /* 显示对象描述 */

/* 库和文件命令 */
DSPLIB     LIB(MYLIB)        /* 显示库内容 */
DSPFD      FILE(MYLIB/*ALL)  /* 显示文件描述 */
DSPFFD     FILE(MYLIB/MYFILE)/* 显示文件字段描述 */
```

```
/* 作业管理命令 */
WRKACTJOB                          /* 处理活动作业 */
WRKJOBSCDE                         /* 处理作业计划 */
SBMJOB     CMD(CALL PGM(MYPGM))    /* 提交作业 */

/* 消息命令 */
DSPMSG                             /* 显示消息 */
SNDMSG     MSG('Hello') TOUSR(QSYSOPR)  /* 发送消息 */

/* 系统状态 */
WRKDSKSTS                          /* 工作磁盘状态 */
WRKSYSSTS                          /* 工作系统状态 */
```

6.3 屏幕操作技巧

6.3.1 字段导航

```
屏幕导航方法：

1. Tab 键导航
   Tab       → 移到下一个可输入字段
   Shift+Tab → 移到上一个可输入字段

2. 快捷键导航
   Home      → 移到第一个输入字段
   End       → 移到最后一个输入字段
   F10       → 移到命令行

3. 鼠标导航（在 PCOM 中）
 点击字段 → 直接定位到该字段

4. 快速填充
 在选项字段输入数字后按 Enter → 直接选择
 如：输入 5 后按 Enter → 选择选项 5

字段类型识别：
┌──────────────────────────────────────────────────────────────────────┐
│  _ _ _ _ _ _   下划线字段 = 可输入字符数据                           │
│  0 0 0 0 0 0   数字字段 = 可输入数字数据                             │
│  X X X X X X   高亮字段 = 只读/输出字段                              │
```

```
│  < > < > < >  输入光标 = 当前输入位置                                    │
└──────────────────────────────────────────────────────────────────────┘
```

6.3.2 文本选择与复制

在 PCOM 中进行复制/粘贴：

方法 1：使用鼠标

```
┌──────────────────────────────────────────────────────────────────────┐
│ 1. 点击并拖动鼠标选择文本                                            │
│ 2. 右键点击选择 "Copy"                                               │
│ 3. 定位到目标位置                                                    │
│ 4. 右键点击选择 "Paste"                                              │
└──────────────────────────────────────────────────────────────────────┘
```

方法 2：使用菜单

```
┌──────────────────────────────────────────────────────────────────────┐
│ 1. Edit → Select All 或选择文本                                      │
│ 2. Edit → Copy                                                       │
│ 3. 定位到目标位置                                                    │
│ 4. Edit → Paste                                                      │
└──────────────────────────────────────────────────────────────────────┘
```

方法 3：使用快捷键

```
┌──────────────────────────────────────────────────────────────────────┐
│ Ctrl+C    → 复制（需先选择文本）                                     │
│ Ctrl+V    → 粘贴                                                     │
│ Ctrl+A    → 全选                                                     │
│ Ctrl+X    → 剪切                                                     │
└──────────────────────────────────────────────────────────────────────┘
```

复制特定区域：
Alt+点击拖动 → 矩形区域选择（块选择）

6.4 多会话管理

6.4.1 多会话配置

配置多个会话：

1. 创建会话配置文件

```
   ┌──────────────────────────────────────────────────────────────────┐
```

```
┌──────────────────────────────────────────────────────────┐
│ Session A: PROD_WS.WS  → 生产环境                        │
│ Session B: TEST_WS.WS  → 测试环境                        │
│ Session C: DEV_WS.WS   → 开发环境                        │
└──────────────────────────────────────────────────────────┘
```

2. 启动多个会话

```
┌──────────────────────────────────────────────────────────┐
│ 方法 1：使用 Session Manager                             │
│   - 选中多个会话                                         │
│   - 点击 Start                                           │
│                                                          │
│ 方法 2：使用快捷方式                                     │
│   - 为每个会话创建桌面快捷方式                           │
│   - 分别启动                                             │
│                                                          │
│ 方法 3：使用命令行                                       │
│   PCOMSTRT /S=PROD_WS.WS                                 │
│   PCOMSTRT /S=TEST_WS.WS                                 │
└──────────────────────────────────────────────────────────┘
```

3. 会话切换

```
┌──────────────────────────────────────────────────────────┐
│ Alt+Tab       → 在 Windows 任务间切换                    │
│ Ctrl+F11      → 快速连接（可配置）                       │
│ 任务栏点击    → 直接选择会话窗口                         │
└──────────────────────────────────────────────────────────┘
```

6.4.2 会话属性定制

个性化设置：

1. 颜色方案
   Edit → Preferences → Appearance → Colors

推荐颜色方案：

```
┌──────────────────────────────────────────────────────────┐
│ 传统绿色：                                               │
│   背景：黑色 (#000000)                                   │
│   前景：绿色 (#00FF00)                                   │
│                                                          │
│ 现代配色：                                               │
│   背景：深蓝 (#000033)                                   │
│   前景：白色 (#FFFFFF)                                   │
│   高亮：黄色 (#FFFF00)                                   │
```

```
   └──────────────────────────────────────────────────────────┘

2. 字体设置
   Edit → Preferences → Appearance → Font

 推荐设置：
   ┌──────────────────────────────────────────────────────────┐
   │ 字体：Consolas 或 Courier New                            │
   │ 大小：14-16 磅                                           │
   │ 样式：常规                                               │
   │ 平滑：开启                                               │
   └──────────────────────────────────────────────────────────┘

3. 声音设置
   Edit → Preferences → Sound

 可选操作：
   ┌──────────────────────────────────────────────────────────┐
   │ ☑ 蜂鸣音 (Beep)                                          │
   │ □ 错误提示音                                             │
   │ □ 连接/断开提示                                          │
   └──────────────────────────────────────────────────────────┘
```

第7章：高级功能与宏编程

7.1 宏与脚本编程

7.1.1 宏基础

Personal Communications 支持使用 VBScript 编写宏，实现自动化操作：

```
' PCOM 宏示例：自动登录
' 文件：AutoLogin.mac

[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=自动登录 IBM i
[PCOMM SCRIPT SOURCE]

' 创建与当前会话的连接
Dim autECLSession
```

```
Set autECLSession = CreateObject("PCOMM.autECLSession")
autECLSession.SetConnectionByName(ThisSessionName)

' 获取呈现空间对象
Dim autECLPS
Set autECLPS = autECLSession.autECLPS

' 获取操作员信息区域
Dim autECLIOIA
Set autECLIOIA = autECLSession.autECLIOIA

' 等待登录屏幕出现
autECLIOIA.WaitForInputReady 30

' 输入用户名
autECLPS.SetText "MYUSER", 7, 23

' 移动到密码字段并输入密码
autECLPS.SetText "MYPASS", 8, 23

' 模拟回车键
autECLPS.SendKeys "[enter]"

' 清理对象
Set autECLPS = Nothing
Set autECLIOIA = Nothing
Set autECLSession = Nothing
```

7.1.2 创建宏的步骤

```
创建宏的完整步骤：

1. 打开宏编辑器
   Edit → Preferences → Macro/Script
   Click "Customize"
   File → New → VBScript

2. 编写脚本代码
 在 Script Statements 区域输入 VBScript 代码

3. 保存宏
   File → Save As
 保存为 .mac 文件
 默认位置：C:\Users\<User>\Documents\IBM\PCOMM\Macros\
```

```
4. 运行宏
   Actions → Start Playing Macro/Script
 或配置快捷键

5. 调试宏
 使用 MsgBox 输出变量值
 使用 On Error Resume Next 捕获错误
```

7.1.3 常用宏示例

```
' 示例 1：自动执行命令
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=自动执行 WRKACTJOB
[PCOMM SCRIPT SOURCE]

Dim ps
Set ps = autECLSession.autECLPS

' 等待屏幕就绪
autECLSession.autECLIOIA.WaitForInputReady 10

' 输入命令
ps.SetText "WRKACTJOB", 20, 7
ps.SendKeys "[enter]"

' 等待命令执行完成
autECLSession.autECLIOIA.WaitForAppAvailable 10

Set ps = Nothing
```

```
' 示例 2：数据提取到 Excel
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=提取屏幕数据到 Excel
[PCOMM SCRIPT SOURCE]

Dim ps, excel, wb, ws
Dim row, col, screenData

' 创建 Excel 对象
Set excel = CreateObject("Excel.Application")
```

```
excel.Visible = True
Set wb = excel.Workbooks.Add
Set ws = wb.Worksheets(1)

' 获取呈现空间
Set ps = autECLSession.autECLPS

' 提取屏幕数据
row = 1
For i = 1 To 24  ' 24 行
    screenData = ps.GetText(i, 1, 80)  ' 获取整行数据
    ws.Cells(row, 1).Value = screenData
    row = row + 1
Next

' 自动调整列宽
ws.Columns("A").AutoFit

' 清理
Set ws = Nothing
Set wb = Nothing
Set excel = Nothing
Set ps = Nothing

MsgBox "数据提取完成！"
```

```
' 示例 3：循环处理多个记录
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=循环处理列表
[PCOMM SCRIPT SOURCE]

Dim ps, ioia
Dim counter

Set ps = autECLSession.autECLPS
Set ioia = autECLSession.autECLIOIA

counter = 0

Do While counter < 10
    ' 等待屏幕就绪
    ioia.WaitForInputReady 5

    ' 在选项字段输入 5（假设是修改选项）
```

```
    ps.SetText "5", 8 + counter, 4
    ps.SendKeys "[enter]"

    ' 等待修改屏幕
    ioia.WaitForInputReady 5

    ' 输入修改内容
    ps.SetText "NEWVALUE", 10, 30
    ps.SendKeys "[enter]"

    ' 等待返回列表
    ioia.WaitForInputReady 5

    counter = counter + 1
Loop

MsgBox "处理了 " & counter & " 条记录"

Set ps = Nothing
Set ioia = Nothing
```

7.2 EHLLAPI 编程

7.2.1 EHLLAPI 简介

EHLLAPI（Emulator High-Level Language Application Programming Interface）允许外部程序与 PCOM 交互：

```
EHLLAPI 连接架构：

┌─────────────────────────────────────────────────────────────┐
│                    外部应用程序                              │
│            (C++, C#, VB, Java, Python 等)                   │
├─────────────────────────────────────────────────────────────┤
│                    EHLLAPI 接口                             │
│               (PCOMM 提供的 API)                            │
├─────────────────────────────────────────────────────────────┤
│                    Personal Communications                  │
│                       (PCOM)                                │
├─────────────────────────────────────────────────────────────┤
│                    IBM i 主机系统                           │
└─────────────────────────────────────────────────────────────┘
```

7.2.2 常用 EHLLAPI 函数

函数	功能	参数
ConnectPresentationSpace	连接呈现空间	(PSID, Return)
DisconnectPresentationSpace	断开连接	(PSID, Return)
SendKey	发送按键	(Text, PSID, Return)
Wait	等待	(Time, PSID, Return)
CopyPresentationSpace	复制屏幕内容	(Buffer, PSID, Return)
SearchPresentationSpace	搜索文本	(Text, Position, PSID, Return)
QueryCursorLocation	查询光标位置	(Row, Col, PSID, Return)
SetCursor	设置光标位置	(Row, Col, PSID, Return)

7.2.3 EHLLAPI 示例代码

```
// C# EHLLAPI 示例
using System;
using System.Runtime.InteropServices;
using System.Text;

class EHLLAPIExample
{
    // 声明 EHLLAPI 函数
    [DllImport("PCSHLL32.dll")]
    static extern int hllapi(ref int func, StringBuilder data, ref int
length, ref int retc);

    // 功能码常量
    const int HA_CONNECT_PS = 1;
    const int HA_DISCONNECT_PS = 2;
    const int HA_SENDKEY = 3;
    const int HA_WAIT = 4;
    const int HA_COPY_PS = 8;
    const int HA_SETCURSOR = 40;

    static void Main()
    {
```

```
        int func = 0;
        int length = 0;
        int retc = 0;
        StringBuilder data = new StringBuilder(1024);

        // 连接到呈现空间 A
        func = HA_CONNECT_PS;
        data.Length = 0;
        data.Append("A");
        length = 1;
        hllapi(ref func, data, ref length, ref retc);

        if (retc != 0)
        {
            Console.WriteLine("连接失败！");
            return;
        }

        Console.WriteLine("已连接到会话 A");

        // 等待屏幕就绪
        func = HA_WAIT;
        data.Length = 0;
        data.Append("30");  // 等待 30 秒
        length = 2;
        hllapi(ref func, data, ref length, ref refc);

        // 发送命令
        func = HA_SENDKEY;
        data.Length = 0;
        data.Append("WRKACTJOB[enter]");
        length = data.Length;
        hllapi(ref func, data, ref length, ref retc);

        // 复制屏幕内容
        func = HA_COPY_PS;
        data.Length = 0;
        data.Capacity = 1920;  // 24x80 屏幕
        length = 1920;
        hllapi(ref func, data, ref length, ref retc);

        Console.WriteLine("屏幕内容：");
        Console.WriteLine(data.ToString());

        // 断开连接
        func = HA_DISCONNECT_PS;
```

```
        data.Length = 0;
        length = 0;
        hllapi(ref func, data, ref length, ref retc);

        Console.WriteLine("已断开连接");
    }
}
```

```
# Python EHLLAPI 示例 (使用 ctypes)
import ctypes
from ctypes import wintypes

# 加载 DLL
hllapi = ctypes.windll.LoadLibrary("PCSHLL32.dll")

# 定义函数原型
hllapi_func = hllapi.hllapi
hllapi_func.argtypes = [
    ctypes.POINTER(wintypes.INT),      # func
    ctypes.c_char_p,                    # data
    ctypes.POINTER(wintypes.INT),      # length
    ctypes.POINTER(wintypes.INT)       # retc
]
hllapi_func.restype = wintypes.INT

# 功能码
HA_CONNECT_PS = 1
HA_DISCONNECT_PS = 2
HA_SENDKEY = 3
HA_WAIT = 4
HA_COPY_PS = 8

def connect_ps(ps_id):
    """连接到呈现空间"""
    func = wintypes.INT(HA_CONNECT_PS)
    data = ctypes.create_string_buffer(ps_id.encode())
    length = wintypes.INT(len(ps_id))
    retc = wintypes.INT(0)

    result = hllapi_func(ctypes.byref(func), data,
ctypes.byref(length), ctypes.byref(retc))
    return retc.value

def send_key(text):
    """发送按键"""
```

```
    func = wintypes.INT(HA_SENDKEY)
    # 将特殊键转换为 EHLLAPI 格式
    text = text.replace("[enter]", "@E")
    text = text.replace("[tab]", "@T")
    text = text.replace("[pf1]", "@1")
    data = ctypes.create_string_buffer(text.encode())
    length = wintypes.INT(len(text))
    retc = wintypes.INT(0)

    result = hllapi_func(ctypes.byref(func), data,
ctypes.byref(length), ctypes.byref(retc))
    return retc.value

def copy_screen():
    """复制屏幕内容"""
    func = wintypes.INT(HA_COPY_PS)
    data = ctypes.create_string_buffer(1920)
    length = wintypes.INT(1920)
    retc = wintypes.INT(0)

    result = hllapi_func(ctypes.byref(func), data,
ctypes.byref(length), ctypes.byref(retc))
    return data.value.decode('ascii', errors='ignore')

# 使用示例
if __name__ == "__main__":
    # 连接到会话 A
    ret = connect_ps("A")
    print(f"连接结果: {ret}")

    # 发送命令
    send_key("WRKACTJOB[enter]")
    print("已发送 WRKACTJOB 命令")

    # 复制屏幕
    screen = copy_screen()
    print("屏幕内容：")
    print(screen)
```

7.3 数据传输

7.3.1 文件传输配置

```
PCOM 文件传输设置：

1. 启动文件传输
   Actions → Transfer → Send/Receive

2. 选择传输类型
   ┌──────────────────────────────────────────────────────────┐
   │ Send to Host    → 从 PC 传输到 IBM i                     │
   │ Receive from Host → 从 IBM i 传输到 PC                   │
   └──────────────────────────────────────────────────────────┘

3. 配置传输参数
   ┌──────────────────────────────────────────────────────────┐
   │ 文件格式：                                               │
   │   ☑ PC 文件格式 (CSV, TXT)                               │
   │   ☑ IBM i 文件格式 (物理文件)                            │
   │                                                          │
   │ 转换选项：                                               │
   │   ☑ ASCII 到 EBCDIC                                      │
   │   ☑ 去除尾部空格                                         │
   │   □ 包含字段名                                           │
   │                                                          │
   │ 记录格式：                                               │
   │   记录长度: [     ]                                      │
   │   字段分隔符: [,]                                        │
   └──────────────────────────────────────────────────────────┘
```

7.3.2 批量数据传输

```
' 批量数据传输宏示例
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=批量上传数据文件
[PCOMM SCRIPT SOURCE]

Dim fso, folder, files, file
Dim transfer

' 创建文件系统对象
```

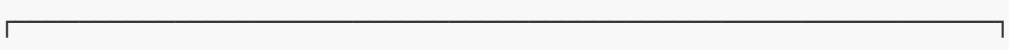
```
Set fso = CreateObject("Scripting.FileSystemObject")
Set folder = fso.GetFolder("C:\Data\Upload")
Set files = folder.Files

' 创建传输对象
Set transfer = CreateObject("PCOMM.autECLTransfer")

For Each file In files
    If LCase(fso.GetExtensionName(file.Name)) = "csv" Then
        ' 配置传输参数
        transfer.HostFile = "UPLOADLIB/" & fso.GetBaseName(file.Name)
        transfer.PCFile = file.Path
        transfer.HostOptions = "REPLACE"
        transfer.PCOptions = "ASCII CRLF"

        ' 执行传输
        transfer.SendFile

        If transfer.TransferStatus = 0 Then
            MsgBox "成功上传: " & file.Name
        Else
            MsgBox "上传失败: " & file.Name & " 错误码: " &
transfer.TransferStatus
        End If
    End If
Next

Set transfer = Nothing
Set files = Nothing
Set folder = Nothing
Set fso = Nothing
```

第8章：故障排除与性能优化

8.1 常见连接问题

8.1.1 连接失败排查

连接问题诊断流程：



```
┌──────────────────────────────────────────────────────────────────┐
│ 问题：无法连接到 IBM i 主机                                      │
├──────────────────────────────────────────────────────────────────┤
│                                                                  │
│  步骤 1：检查网络连接                                            │
│  ├── ping 192.168.1.100  (测试网络连通性)                        │
│  ├── telnet 192.168.1.100 23  (测试端口连通性)                   │
│  └── 检查防火墙设置                                              │
│                                                                  │
│  步骤 2：验证主机配置                                            │
│  ├── 确认 IBM i 系统已启动                                       │
│  ├── 确认 Telnet 服务器正在运行                                  │
│  │   在 IBM i 上执行：STRTCPSVR *TELNET                          │
│  └── 确认端口配置正确                                            │
│                                                                  │
│  步骤 3：检查会话配置                                            │
│  ├── 确认 IP 地址/主机名正确                                     │
│  ├── 确认端口号正确（Telnet: 23, SSL: 992）                      │
│  └── 检查设备名称池配置                                          │
│                                                                  │
│  步骤 4：查看错误日志                                            │
│  ├── PCOM 日志：Help → About → System Information                │
│  └── IBM i 日志：DSPLOG                                          │
│                                                                  │
└──────────────────────────────────────────────────────────────────┘
```

8.1.2 性能问题诊断

性能优化检查清单：

网络性能：
□ 使用有线连接而非无线
□ 检查网络延迟（ping 测试）
□ 确认带宽充足
□ 检查网络拥塞

主机性能：
□ 检查 IBM i CPU 使用率（WRKACTJOB）
□ 检查磁盘 I/O（WRKDSKSTS）
□ 检查内存使用情况
□ 检查交互式作业数量

PCOM 设置：
□ 禁用不必要的视觉效果
□ 调整屏幕刷新率

□ 启用压缩（如果可用）
□ 调整缓冲区大小

8.2 日志与诊断

8.2.1 PCOM 日志配置

启用详细日志：

1. 打开日志配置
   Help → System Information → Logging

2. 配置日志级别

```
┌──────────────────────────────────────────────────────────┐
│ 日志级别：                                               │
│   ☑ 错误 (Error)                                         │
│   ☑ 警告 (Warning)                                       │
│   ☑ 信息 (Information)                                   │
│   □ 调试 (Debug)                                         │
│                                                          │
│ 日志文件位置：                                           │
│   C:\Users\<User>\AppData\Local\IBM\PCOM\Logs\           │
│                                                          │
│ 最大文件大小：10 MB                                      │
│ 保留日志数量：5                                          │
└──────────────────────────────────────────────────────────┘
```

3. 查看日志
 使用文本编辑器打开日志文件
 搜索关键字：ERROR, WARNING, FAIL

8.2.2 诊断工具使用

内置诊断工具：

1. 系统信息
   Help → About Personal Communications
   └── 显示版本、配置、环境信息

2. 跟踪功能
   Actions → Trace → Start
   ├── 通信跟踪 (Communication Trace)

```
    ├── 数据流跟踪 (Data Stream Trace)
    └── API 调用跟踪 (API Trace)

3. 连接测试
   Help → Diagnostic Tools → Connection Test
   └── 自动检测连接问题

4. 屏幕捕获
   Edit → Copy → Screen
   └── 捕获当前屏幕用于故障报告
```

8.3 性能优化最佳实践

8.3.1 显示性能优化

显示优化设置：

1. 减少屏幕更新

```
┌──────────────────────────────────────────────────────┐
│ Edit → Preferences → Appearance                      │
│                                                      │
│ ☑ Use backing store (推荐启用)                       │
│ □ Blink cursor (可禁用以减少闪烁)                    │
│ □ Show crosshair (可禁用)                            │
│ ☑ Fast path updates (推荐启用)                       │
└──────────────────────────────────────────────────────┘
```

2. 优化字体渲染

```
┌──────────────────────────────────────────────────────┐
│ 使用点阵字体而非 TrueType 字体                       │
│ 推荐：Terminal, Consolas, Courier New                │
│ 字号：14-16 磅                                       │
│ 禁用：ClearType 平滑（某些配置）                     │
└──────────────────────────────────────────────────────┘
```

3. 颜色优化

```
┌──────────────────────────────────────────────────────┐
│ 使用较少颜色的配色方案                               │
│ 避免渐变和高亮效果                                   │
│ 推荐：传统绿屏或黑白配色                             │
└──────────────────────────────────────────────────────┘
```

8.3.2 连接稳定性优化

连接稳定性设置：

1. 保持连接

```
┌──────────────────────────────────────────────────────────┐
│ Edit → Preferences → Communications                      │
│                                                          │
│ Keep Alive Interval: 300 (秒)                            │
│   - 每 5 分钟发送保持连接信号                            │
│                                                          │
│ Response Timeout: 60 (秒)                                │
│   - 等待响应的超时时间                                   │
│                                                          │
│ Auto Reconnect: ☑ Enabled                                │
│ Retry Count: 10                                          │
│ Retry Interval: 5 (秒)                                   │
└──────────────────────────────────────────────────────────┘
```

2. 会话恢复

```
┌──────────────────────────────────────────────────────────┐
│ File → Session Preferences → Save Settings               │
│                                                          │
│ ☑ Save window position                                   │
│ ☑ Save connection settings                               │
│ ☑ Auto-save on exit                                      │
│ □ Prompt before saving                                   │
└──────────────────────────────────────────────────────────┘
```

8.3.3 键盘响应优化

键盘优化设置：

1. 减少输入延迟

```
┌──────────────────────────────────────────────────────────┐
│ Edit → Preferences → Keyboard                            │
│                                                          │
│ Key Repeat Delay: Short                                  │
│ Key Repeat Rate: Fast                                    │
│                                                          │
│ ☑ Immediate key response                                 │
│ □ Buffer keystrokes                                      │
└──────────────────────────────────────────────────────────┘
```

```
2. 功能键优化

   ┌──────────────────────────────────────────────────────────┐
   │ 使用 PC 功能键直接映射                                    │
   │ 避免使用 Fn 组合键                                        │
   │ 考虑使用外部程序自定义键盘映射                            │
   └──────────────────────────────────────────────────────────┘
```

第三部分：RPG IV 编程详解

第9章：RPG IV 语言基础

9.1 RPG 语言演进

9.1.1 RPG 历史回顾

RPG（Report Program Generator）自 1959 年诞生以来经历了多次重大演进：

```
RPG 语言演进时间线：

1959 ──────────────────────────────────────────────────────────────>
  │
  ├── RPG I (1960s)
  │    └── 在 IBM 1401 上运行
  │    └── 固定格式，卡片输入
  │
  ├── RPG II (1970s)
  │    └── System/3, System/34, System/36
  │    └── 增加了磁盘文件支持
  │
  ├── RPG III (1978)
  │    └── System/38, 后来的 AS/400
  │    └── 引入了结构化编程概念
  │    └── 计算规范（C-spec）
  │
  ├── RPG/400 (1988)
  │    └── AS/400 早期版本
  │    └── 支持更多数据类型
  │
  ├── RPG IV (1994)
  │    └── ILE（集成语言环境）
```

```
│   └── 自由格式计算
│   └── 内置函数
│   └── 原型定义
│
├── RPG IV with /FREE (2001)
│   └── 完全自由格式
│   └── 摆脱了 80 列限制
│
├── RPG IV Modern (2013+)
│   └── 完全自由格式（所有规范）
│   └── 新操作码和内置函数
│   └── 增强的数据结构
│
└── RPG IV Today
    └── 每 IBM i 版本持续增强
    └── 开源集成支持
```

9.1.2 RPG IV 与 RPG III 对比

特性	RPG III	RPG IV
格式	固定格式	固定/自由格式
列限制	严格 80 列	无列限制（/FREE）
子过程	不支持	完全支持
原型	不支持	支持原型定义
内置函数	有限	丰富的内置函数
指针支持	有限	完整支持
ILE 集成	不支持	完全支持
嵌入式 SQL	有限	完全支持

9.2 RPG IV 程序结构

9.2.1 程序组成

```
**FREE
// ============================================================
// 程序：CUSTINQ
// 功能：客户查询程序
// 作者：[作者名]
// 日期：[日期]
// ============================================================

// ------------------------------------------------------------
// 控制规范 (Control Specification)
// ------------------------------------------------------------
Ctl-Opt DftActGrp(*No) ActGrp(*NEW)
        Option(*SrcStmt:*NoDebugIo)
        BndDir('QC2LE');

// ------------------------------------------------------------
// 文件声明 (File Declaration)
// ------------------------------------------------------------
Dcl-F CUSTMAST Disk(*Ext) Usage(*Input) Keyed;
Dcl-F CUSTDSP WorkStn;

// ------------------------------------------------------------
// 数据声明 (Data Declaration)
// ------------------------------------------------------------
Dcl-S Exit Ind Inz(*Off);
Dcl-S Found Ind Inz(*Off);
Dcl-S CustCount Int(10) Inz(0);

// ------------------------------------------------------------
// 主程序 (Main Procedure)
// ------------------------------------------------------------
Dcl-Pi *N;
    P_CustNo Packed(7:0);
End-Pi;

// 主程序逻辑
Clear CustDSP;
ExSr Init;

Dow Not Exit;
    ExFmt CUSTDSP;
```

```
    Select;
        When *In03;
            Exit = *On;
        When *In05;
            ExSr Refresh;
        Other;
            ExSr Process;
    EndSl;
EndDo;

*InLr = *On;
Return;

// ------------------------------------------------------------
// 子程序 (Subroutines)
// ------------------------------------------------------------
BegSr Init;
    // 初始化代码
    CustCount = 0;
EndSr;

BegSr Refresh;
    // 刷新显示
    Clear CUSTDSP;
EndSr;

BegSr Process;
    // 处理逻辑
    Chain P_CustNo CUSTMAST;
    If %Found(CUSTMAST);
        Found = *On;
        CustCount += 1;
    Else;
        Found = *Off;
    EndIf;
EndSr;
```

9.2.2 规范类型详解

```
RPG IV 规范类型：

┌──────────────────────────────────────────────────────────────┐
│ 控制规范 (H-Spec / Ctl-Opt)                                  │
│ ├── 程序属性设置                                             │
```

```
│ ├── 编译选项                                                    │
│ └── 绑定目录声明                                                │
├──────────────────────────────────────────────────────────────────┤
│ 文件规范 (F-Spec / Dcl-F)                                        │
│ ├── 文件声明                                                    │
│ ├── 设备类型（磁盘、工作站、打印机等）                          │
│ └── 文件属性                                                    │
├──────────────────────────────────────────────────────────────────┤
│ 定义规范 (D-Spec / Dcl-S, Dcl-C, Dcl-DS)                         │
│ ├── 变量声明 (Dcl-S)                                            │
│ ├── 常量声明 (Dcl-C)                                            │
│ └── 数据结构声明 (Dcl-DS)                                       │
├──────────────────────────────────────────────────────────────────┤
│ 输入规范 (I-Spec)                                                │
│ └── 程序描述文件输入（较少使用）                                │
├──────────────────────────────────────────────────────────────────┤
│ 计算规范 (C-Spec / 自由格式)                                     │
│ ├── 操作逻辑                                                    │
│ ├── 控制结构                                                    │
│ └── 内置函数调用                                                │
├──────────────────────────────────────────────────────────────────┤
│ 输出规范 (O-Spec)                                                │
│ └── 程序描述文件输出（较少使用）                                │
├──────────────────────────────────────────────────────────────────┤
│ 过程规范 (P-Spec / Dcl-P, Dcl-Proc)                              │
│ ├── 子过程声明                                                  │
│ ├── 主过程声明                                                  │
│ └── 接口定义                                                    │
└──────────────────────────────────────────────────────────────────┘
```

9.3 基本语法元素

9.3.1 标识符和命名规则

```
// RPG IV 命名规则

// 有效名称：
Dcl-S CustomerName Char(50);     // 字母开头
Dcl-S CUST_NAME Char(50);        // 可包含下划线
Dcl-S #Counter Int(10);          // 可以 # 开头
Dcl-S $Amount Packed(15:2);      // 可以 $ 开头
Dcl-S @Index Int(5);             // 可以 @ 开头
```

```
// 无效名称：
// Dcl-S 1Customer Char(50);     // 错误：数字开头
// Dcl-S Cust-Name Char(50);     // 错误：包含连字符
// Dcl-S VeryLongVariableNameExceeded Char(50);  // 超过 63 字符

// 命名约定建议：
// 1. 变量使用驼峰命名：customerName, orderAmount
// 2. 常量使用全大写：MAX_COUNT, PI_VALUE
// 3. 文件记录格式使用大写：CUSTMASTR
// 4. 子过程使用动词开头：ValidateCustomer, CalculateTotal
```

9.3.2 注释

```
// ============================================================
// 注释方法
// ============================================================

// 1. 行尾注释（双斜杠）
Dcl-S Count Int(10);  // 这是行尾注释

// 2. 整行注释
// 这是整行注释，解释下面的代码块用途

// 3. 多行注释块
/*
 * 这是多行注释
 * 可用于详细的说明
 * 例如描述算法逻辑
 */

// 4. 文档注释（用于自动生成文档）
///
// 计算订单总金额
// @param UnitPrice  单价
// @param Quantity   数量
// @param Discount   折扣率 (0-1)
// @return 计算后的总金额
///
Dcl-Proc CalculateTotal;
    Dcl-Pi *N Packed(15:2);
        UnitPrice Packed(15:2);
        Quantity Int(10);
        Discount Packed(3:2);
    End-Pi;
```

```
    // 过程实现...
End-Proc;
```

9.4 数据类型

9.4.1 基本数据类型

```
// ============================================================
// RPG IV 基本数据类型详解
// ============================================================

// ------------------------------------------------------------
// 1. 字符类型 (Character)
// ------------------------------------------------------------
Dcl-S CustName Char(50);          // 定长字符串，50 字节
Dcl-S Address Varchar(200);       // 变长字符串，最大 200 字节
Dcl-S Flag Ind;                    // 指示器（布尔），*On/*Off

// 字符类型特点：
// Char    - 定长，不足补空格，存储效率高
// Varchar - 变长，存储实际长度 + 2 字节长度前缀
// Ind     - 1 字节，值为 '0' 或 '1'

// ------------------------------------------------------------
// 2. 数值类型 (Numeric)
// ------------------------------------------------------------

// Packed Decimal（压缩十进制）- 推荐用于商业计算
Dcl-S Amount Packed(15:2);        // 15 位数字，2 位小数
Dcl-S Total Packed(31:9);         // 最大精度
// 存储：每两位数字占 1 字节，最后一位占半个字节
// Packed(15:2) 实际占用：(15/2)+1 = 8 字节

// Zoned Decimal（区域十进制）- 用于与外部系统交换
Dcl-S ZonedNum Zoned(7:0);        // 7 位整数
// 存储：每位数字 1 字节，最后一位包含符号

// Integer（整数）- 用于计算和计数
Dcl-S Counter Int(10);            // 10 位整数（4 字节）
Dcl-S BigCount Int(20);           // 20 位整数（8 字节）
Dcl-S SmallCnt Int(3);            // 3 位整数（1 字节）
Dcl-S Unsigned Int(10) Unsigned; // 无符号整数
```

```
// 整数类型对照表：
// Int(3)  = 1 字节, 范围: -128 到 127
// Int(5)  = 2 字节, 范围: -32768 到 32767
// Int(10) = 4 字节, 范围: -2147483648 到 2147483647
// Int(20) = 8 字节, 范围: 非常大

// ------------------------------------------------------------
// 3. 浮点类型 (Floating Point)
// ------------------------------------------------------------
Dcl-S FloatVal Float(4);          // 单精度（4 字节）
Dcl-S DoubleVal Float(8);         // 双精度（8 字节）
// 注意：浮点数不适合精确计算，避免用于货币

// ------------------------------------------------------------
// 4. 日期时间类型 (Date/Time/Timestamp)
// ------------------------------------------------------------
Dcl-S BirthDate Date;             // 日期，格式 *ISO 或指定
Dcl-S StartTime Time;             // 时间
Dcl-S Created Ts;                 // 时间戳（日期+时间+微秒）

// 日期格式选项：
// *ISO   - YYYY-MM-DD (国际标准)
// *USA   - MM/DD/YYYY (美国)
// *EUR   - DD.MM.YYYY (欧洲)
// *JIS   - YYYY-MM-DD (日本)
// *MDY   - MM/DD/YY
// *DMY   - DD/MM/YY
// *YMD   - YY/MM/DD

// 日期声明示例：
Dcl-S OrderDate Date(*ISO);       // 指定 ISO 格式
Dcl-S ShipDate Date(*USA);        // 指定美国格式

// ------------------------------------------------------------
// 5. 指针类型 (Pointer)
// ------------------------------------------------------------
Dcl-S Ptr Pointer;                // 通用指针
Dcl-S ProcPtr Pointer(*Proc);     // 过程指针

// 使用示例：
Dcl-S DataPtr Pointer;
Dcl-S Data Char(100) Based(DataPtr);
// Data 基于 DataPtr 指向的内存位置

// ------------------------------------------------------------
```

```
// 6. 对象类型 (Object)
// ------------------------------------------------------------
Dcl-S JavaObj Object(*Java:'java.lang.String');
Dcl-S ListObj Object;
// 用于与 Java 和其他面向对象系统集成
```

9.4.2 数据类型转换

```
// ============================================================
// 数据类型转换
// ============================================================

// ------------------------------------------------------------
// 1. 使用内置函数转换
// ------------------------------------------------------------

// 数值转字符
Dcl-S NumVal Packed(7:2) Inz(1234.56);
Dcl-S CharVal Char(10);
CharVal = %Char(NumVal);         // 结果: '1234.56'

// 字符转数值
Dcl-S CharNum Char(10) Inz('9876.54');
Dcl-S NumResult Packed(9:2);
NumResult = %Dec(CharNum:9:2);   // 结果: 9876.54

// 日期转换
Dcl-S CharDate Char(10) Inz('2024-03-15');
Dcl-S DateVal Date;
DateVal = %Date(CharDate:*ISO);  // 转换为日期类型

// 时间戳转换
Dcl-S TsVal Timestamp;
TsVal = %Timestamp();            // 当前时间戳

// ------------------------------------------------------------
// 2. 使用 %INT, %UNS 转换
// ------------------------------------------------------------
Dcl-S FloatNum Float(8) Inz(123.456);
Dcl-S IntVal Int(10);
IntVal = %Int(FloatNum);         // 截断小数部分: 123

Dcl-S CharInt Char(5) Inz('456');
IntVal = %Int(CharInt);          // 字符转整数: 456
```

```
// ------------------------------------------------------------
// 3. 日期时间格式化
// ------------------------------------------------------------
Dcl-S Today Date Inz(%Date());
Dcl-S DateStr Char(10);

DateStr = %Char(Today:*ISO);     // '2024-03-15'
DateStr = %Char(Today:*USA);     // '03/15/2024'
DateStr = %Char(Today:*EUR);     // '15.03.2024'

// ------------------------------------------------------------
// 4. 安全转换（避免运行时错误）
// ------------------------------------------------------------
Monitor;
    NumResult = %Dec(CharNum:9:2);
On-Error *Program;
    // 转换失败处理
    NumResult = 0;
EndMon;

// ------------------------------------------------------------
// 5. 截断和填充
// ------------------------------------------------------------
Dcl-S LongChar Char(50) Inz('Hello World');
Dcl-S ShortChar Char(5);

// 截断（从左边开始）
ShortChar = %Subst(LongChar:1:5);  // 'Hello'

// 使用 %Trim 去除空格
Dcl-S Padded Char(20) Inz('  Test  ');
Dcl-S Trimmed Char(20);
Trimmed = %Trim(Padded);           // 'Test'（去除两端空格）
Trimmed = %TrimR(Padded);          // '  Test'（去除右边空格）
Trimmed = %TrimL(Padded);          // 'Test  '（去除左边空格）
```

9.5 操作符与表达式

9.5.1 算术操作符

```
// ============================================================
// 算术操作符
// ============================================================
```

```
Dcl-S A Int(10) Inz(10);
Dcl-S B Int(10) Inz(3);
Dcl-S Result Int(10);
Dcl-S DecResult Packed(7:2);

// 基本算术运算
Result = A + B;          // 加法: 13
Result = A - B;          // 减法: 7
Result = A * B;          // 乘法: 30
Result = A / B;          // 除法: 3 (整数除法截断)
DecResult = A / B;       // 除法: 3.33 (保留小数)
Result = A ** B;         // 幂运算: 1000 (10^3)
Result = %Rem(A:B);      // 余数: 1 (10 % 3)
Result = %Div(A:B);      // 整数除法: 3

// 复合赋值
A += B;                  // A = A + B
A -= B;                  // A = A - B
A *= B;                  // A = A * B
A /= B;                  // A = A / B
A **= B;                 // A = A ** B

// 递增和递减
A += 1;                  // 递增
A -= 1;                  // 递减
```

9.5.2 比较操作符

```
// ============================================================
// 比较操作符
// ============================================================

Dcl-S X Int(10) Inz(10);
Dcl-S Y Int(10) Inz(20);
Dcl-S Equal Ind;

// 比较操作
Equal = (X = Y);         // 等于: *Off (false)
Equal = (X <> Y);        // 不等于: *On (true)
Equal = (X > Y);         // 大于: *Off
Equal = (X < Y);         // 小于: *On
Equal = (X >= Y);        // 大于等于: *Off
Equal = (X <= Y);        // 小于等于: *On
```

```
// 字符比较
Dcl-S Str1 Char(10) Inz('ABC');
Dcl-S Str2 Char(10) Inz('DEF');

Equal = (Str1 = Str2);   // 字符比较 (区分大小写)
Equal = (Str1 < Str2);   // 字母顺序比较: *On ('ABC' < 'DEF')

// 字符串比较技巧
If %Trim(Str1) = %Trim(Str2);
    // 去除空格后比较
EndIf;

If %Upper(Str1) = %Upper(Str2);
    // 不区分大小写比较
EndIf;
```

9.5.3 逻辑操作符

```
// ============================================================
// 逻辑操作符
// ============================================================

Dcl-S Flag1 Ind Inz(*On);
Dcl-S Flag2 Ind Inz(*Off);
Dcl-S Result Ind;

// 逻辑与 (AND)
Result = Flag1 And Flag2;    // *Off

// 逻辑或 (OR)
Result = Flag1 Or Flag2;     // *On

// 逻辑非 (NOT)
Result = Not Flag1;          // *Off
Result = Not Flag2;          // *On

// 复杂逻辑表达式
If (Flag1 And Flag2) Or (Not Flag1 And Not Flag2);
    // 异或逻辑（一个且仅一个为真）
EndIf;

// 短路求值
If (X > 0) And (Y / X > 5);
```

```
    // 如果 X <= 0，不会执行 Y / X（避免除零错误）
EndIf;

// 使用指示器
*In01 = *On;                    // 设置指示器 01
If *In01;
    // 检查指示器
EndIf;
```

9.6 控制结构

9.6.1 条件语句

```
// ============================================================
// 条件语句
// ============================================================

// ------------------------------------------------------------
// 1. IF 语句
// ------------------------------------------------------------
If (Count > 0);
    // 单个条件
    Process();
EndIf;

If (Count > 0) And (Count <= MAX_COUNT);
    // 多个条件
    Process();
Else;
    // 否则分支
    ErrorHandler();
EndIf;

If (Status = 'A');
    // 多分支
    ProcessActive();
ElseIf (Status = 'I');
    ProcessInactive();
ElseIf (Status = 'P');
    ProcessPending();
Else;
    ProcessUnknown();
EndIf;
```

```
// 嵌套 IF
If (OrderType = 'WEB');
    If (OrderAmount > 1000);
        ApplyDiscount(10);
    Else;
        ApplyDiscount(5);
    EndIf;
Else;
    ApplyStandardPricing();
EndIf;

// ------------------------------------------------------------
// 2. SELECT 语句
// ------------------------------------------------------------
Select;
    When (Status = 'A');
        // 处理活动状态
        ProcessActive();
    When (Status = 'I');
        // 处理非活动状态
        ProcessInactive();
    When (Status = 'P' Or Status = 'H');
        // 处理待处理或暂停
        ProcessPending();
    Other;
        // 默认处理
        ProcessUnknown();
EndSl;

// SELECT 用于范围检查
Select;
    When (Score >= 90);
        Grade = 'A';
    When (Score >= 80);
        Grade = 'B';
    When (Score >= 70);
        Grade = 'C';
    When (Score >= 60);
        Grade = 'D';
    Other;
        Grade = 'F';
EndSl;

// ------------------------------------------------------------
// 3. 条件表达式
// ------------------------------------------------------------
```

```
// 三元运算符风格
Dcl-S StatusMsg Char(20);
StatusMsg = (Status = 'A') ? 'Active' : 'Inactive';

// 等效于：
If (Status = 'A');
    StatusMsg = 'Active';
Else;
    StatusMsg = 'Inactive';
EndIf;

// 链式条件
Dcl-S Description Char(50);
Description = (Count = 0) ? 'No items' :
              (Count = 1) ? 'One item' :
              'Many items';
```

9.6.2 循环语句

```
// ============================================================
// 循环语句
// ============================================================

// ------------------------------------------------------------
// 1. FOR 循环
// ------------------------------------------------------------

// 基本 FOR 循环
For I = 1 To 10;
    ProcessItem(I);
EndFor;

// 带步长的 FOR 循环
For I = 10 DownTo 1 By -1;
    ProcessReverse(I);
EndFor;

// 遍历数组
Dcl-S Array Int(10) Dim(100);
For I = 1 To %Elem(Array);
    Array(I) = I * 10;
EndFor;

// 带条件的 FOR 循环
```

```
For I = 1 To 100;
    If (Array(I) = 0);
        Iter;  // 跳过当前迭代
    EndIf;
    ProcessArray(I);
    If (Array(I) = 999);
        Leave;  // 退出循环
    EndIf;
EndFor;

// ------------------------------------------------------------
// 2. DOW (Do While) 循环
// ------------------------------------------------------------
Dcl-S Count Int(10) Inz(0);
Dcl-S Continue Ind Inz(*On);

Dow (Count < 100) And Continue;
    Count += 1;
    If (SomeCondition());
        Continue = *Off;
    EndIf;
EndDo;

// 无限循环（需要内部退出）
Dow *On;
    ReadNextRecord();
    If %Eof(File);
        Leave;
    EndIf;
    ProcessRecord();
EndDo;

// ------------------------------------------------------------
// 3. DOU (Do Until) 循环
// ------------------------------------------------------------
// DOU 至少执行一次，然后在条件为真时退出
Dcl-S ValidInput Ind Inz(*Off);
Dcl-S InputValue Char(10);

Dou ValidInput;
    ExFmt InputScreen;
    If (InputValue <> '');
        ValidInput = *On;
    Else;
        ShowError('Input required');
    EndIf;
```

```
EndDo;

// DOU 用于读取文件
Dou %Eof(CustFile);
    Read CustFile;
    If Not %Eof(CustFile);
        ProcessCustomer();
    EndIf;
EndDo;

// ---------------------------------------------------------------
// 4. 循环控制语句
// ---------------------------------------------------------------
For I = 1 To 1000;
    // ITER - 立即跳到下一次迭代
    If (SkipCondition(I));
        Iter;
    EndIf;

    ProcessItem(I);

    // LEAVE - 立即退出循环
    If (ExitCondition(I));
        Leave;
    EndIf;
EndFor;

// ---------------------------------------------------------------
// 5. 循环标签（用于嵌套循环）
// ---------------------------------------------------------------
OuterLoop:
For I = 1 To 10;
    For J = 1 To 10;
        If (EarlyExitCondition());
            Leave OuterLoop;  // 退出外层循环
        EndIf;
    EndFor;
EndFor OuterLoop;
```

第10章：数据定义与数据结构

10.1 变量声明

10.1.1 基本变量声明

```
// ============================================================
// 变量声明详解
// ============================================================

// ------------------------------------------------------------
// 1. Dcl-S (Declare Standalone Variable)
// ------------------------------------------------------------

// 基本语法：
// Dcl-S 变量名 数据类型 [选项];

// 字符变量
Dcl-S CustName Char(50);                    // 定长，50 字符
Dcl-S Address Varchar(200);                 // 变长，最大 200
Dcl-S Code Char(10) Inz('DEFAULT');         // 带初始值

// 数值变量
Dcl-S OrderTotal Packed(15:2);              // 压缩十进制
Dcl-S LineCount Int(10);                    // 整数
Dcl-S TaxRate Zoned(5:4) Inz(0.0825);       // 区域十进制

// 指示器（布尔）
Dcl-S Processed Ind;                        // 指示器
Dcl-S Valid Ind Inz(*Off);                  // 带初始值

// 日期时间
Dcl-S OrderDate Date;                       // 日期
Dcl-S OrderTime Time;                       // 时间
Dcl-S CreatedTs Timestamp;                  // 时间戳

// 常量
Dcl-C MAX_ORDERS 1000;                      // 数值常量
Dcl-C COMPANY_NAME 'ABC Corporation';       // 字符常量
Dcl-C PI_CONST 3.14159;                     // 浮点常量

// ------------------------------------------------------------
// 2. 声明选项
```

```
// ------------------------------------------------------------

// INZ - 初始值
Dcl-S Counter Int(10) Inz(0);
Dcl-S Today Date Inz(D'2024-01-01');        // 使用日期字面量
Dcl-S Status Char(1) Inz('A');

// CONST - 常量
Dcl-S MinValue Int(10) Const(0);            // 只读常量

// DIM - 数组维度
Dcl-S Scores Int(10) Dim(100);              // 100 元素数组

// LIKE - 基于另一个变量
Dcl-S OrderAmt Packed(15:2);
Dcl-S RefundAmt Like(OrderAmt);             // 相同类型

// POS - 指定位置（在数据结构中）
Dcl-S HeaderData Char(256);
Dcl-S RecordType Char(4) Pos(1);            // 从位置 1 开始
Dcl-S RecordLen Int(5) Pos(5);              // 从位置 5 开始

// EXPORT/IMPORT - 跨模块共享
Dcl-S GlobalCounter Int(10) Export;
Dcl-S ImportedVar Int(10) Import;

// STATIC - 静态存储（子过程中保持值）
Dcl-S CallCount Int(10) Static;

// ------------------------------------------------------------
// 3. 数组声明
// ------------------------------------------------------------

// 一维数组
Dcl-S Numbers Int(10) Dim(100);             // 100 个整数

// 二维数组（使用 Overlay）
Dcl-S Matrix Int(10) Dim(100);
Dcl-S Row1 Int(10) Dim(10) Overlay(Matrix:1);
Dcl-S Row2 Int(10) Dim(10) Overlay(Matrix:41);

// 带初始值的数组
Dcl-S DaysOfWeek Char(10) Dim(7)
    Inz('Monday': 'Tuesday': 'Wednesday': 'Thursday':
        'Friday': 'Saturday': 'Sunday');
```

```
// 运行时可变数组（使用 ALLOC）
Dcl-S DynamicArray Pointer;
Dcl-S ArrayElem Char(50) Based(DynamicArray) Dim(1000);
```

10.1.2 基于文件的声明

```
// ============================================================
// 基于文件字段的声明
// ============================================================

// 声明与文件字段同类型的变量
Dcl-F CUSTMAST Disk(*Ext) Usage(*Input);

// 使用 LIKE 基于文件字段
Dcl-S LocalCustNo Like(CustNo);              // 与 CUSTMAST.CustNo 同类型
Dcl-S LocalName Like(CustName);
Dcl-S LocalStatus Like(Status);

// 使用 EXTFLD 基于外部字段
Dcl-S OrderCount Int(10);
Dcl-S LastOrderDate Date ExtFld('ORDDATE'); // 基于外部定义的字段

// 使用 EXTNAME 声明整个记录格式
Dcl-DS CustRecord ExtName('CUSTMAST') Qualified;
End-DS;
// 现在可以使用 CustRecord.CustNo, CustRecord.CustName 等
```

10.2 数据结构

10.2.1 基本数据结构

```
// ============================================================
// 数据结构 (Data Structure)
// ============================================================

// ------------------------------------------------------------
// 1. 程序描述数据结构
// ------------------------------------------------------------
Dcl-DS CustomerDS;
    CustNo Packed(7:0) Pos(1);
    CustName Char(50) Pos(8);
```

```
    Address Char(100) Pos(58);
    City Char(30) Pos(158);
    State Char(2) Pos(188);
    Zip Char(10) Pos(190);
    Phone Char(15) Pos(200);
    Status Char(1) Pos(215);
End-DS;

// 或使用 Len 自动计算位置
Dcl-DS CustomerDS Len(215);
    CustNo Packed(7:0);
    CustName Char(50);
    Address Char(100);
    City Char(30);
    State Char(2);
    Zip Char(10);
    Phone Char(15);
    Status Char(1);
End-DS;

// ------------------------------------------------------------
// 2. 基于外部文件的数据结构
// ------------------------------------------------------------
Dcl-DS CustRecDS ExtName('CUSTMAST') Qualified;
End-DS;

// 使用示例：
// CustRecDS.CUSTNO
// CustRecDS.CUSTNAME
// CustRecDS.STATUS

// 带前缀的数据结构
Dcl-DS Customer ExtName('CUSTMAST') Prefix('CU_');
End-DS;
// 现在字段名是 CU_CUSTNO, CU_CUSTNAME 等

// ------------------------------------------------------------
// 3. 限定的数据结构
// ------------------------------------------------------------
Dcl-DS Order Qualified;
    OrderNo Int(10);
    OrderDate Date;
    Dcl-DS Customer;              // 嵌套数据结构
        CustNo Int(10);
        CustName Char(50);
    End-DS;
```

```
    Amount Packed(15:2);
End-DS;

// 访问嵌套字段：
// Order.OrderNo
// Order.Customer.CustNo
// Order.Customer.CustName

// ------------------------------------------------------------
// 4. 数组数据结构
// ------------------------------------------------------------
Dcl-DS LineItemDS Dim(100) Qualified;
    ItemNo Char(20);
    Description Char(50);
    Quantity Int(10);
    UnitPrice Packed(15:2);
    LineTotal Packed(15:2);
End-DS;

// 访问数组元素：
// LineItemDS(1).ItemNo
// LineItemDS(5).Quantity
```

10.2.2 特殊数据结构

```
// ============================================================
// 特殊用途数据结构
// ============================================================

// ------------------------------------------------------------
// 1. 程序状态数据结构 (PSDS)
// ------------------------------------------------------------
Dcl-DS PSDS PSDS;
    ProcName Char(10) Pos(1);        // 过程名/程序名
    StatusCode Char(5) Pos(11);      // 状态码
    PrvStatus Char(5) Pos(16);       // 前一个状态码
    LineNum Char(8) Pos(21);         // 语句行号
    Routine Char(8) Pos(29);         // 例行程序名
    Parms Int(3) Pos(37);            // 传入参数个数
    Exception Char(7) Pos(40);       // 异常类型
    ExceptionData Char(80) Pos(47);  // 异常数据
End-DS;

// 使用 PSDS 进行错误处理
```

```
Monitor;
    // 可能出错的代码
On-Error;
    // 使用 PSDS 获取错误信息
    Dsply ('Error at line ' + %Trim(LineNum));
    Dsply ('Exception: ' + %Trim(Exception));
EndMon;

// ------------------------------------------------------------
// 2. 文件信息数据结构 (INFDS)
// ------------------------------------------------------------
Dcl-F CUSTMAST Disk(*Ext) InfDS(FileInfo);

Dcl-DS FileInfo;
    FileStatus Char(10) Pos(11);     // 文件状态
    OpCode Char(6) Pos(21);          // 最后操作码
    RowNum Int(10) Pos(33);          // 行号
    RecordLen Int(5) Pos(125);       // 记录长度
End-DS;

// ------------------------------------------------------------
// 3. 指示器数据结构
// ------------------------------------------------------------
Dcl-DS IndicatorDS Len(99);
    *N Char(1) Pos(1);               // *In01
    *N Char(1) Pos(3);               // *In03
    *N Char(1) Pos(5);               // *In05
    ExitKey Char(1) Pos(3);          // 命名指示器
    RefreshKey Char(1) Pos(5);
End-DS;

// ------------------------------------------------------------
// 4. 日期时间数据结构
// ------------------------------------------------------------
Dcl-DS DateDS;
    Century Int(2) Pos(1);
    Year Int(2) Pos(3);
    Month Int(2) Pos(5);
    Day Int(2) Pos(7);
End-DS;

DateDS = %Char(%Date():*ISO0);       // 将日期分解为组件
// 现在可以单独访问 Century, Year, Month, Day
```

10.2.3 数据结构操作

```
// ============================================================
// 数据结构操作技巧
// ============================================================

// ------------------------------------------------------------
// 1. 数据结构赋值
// ------------------------------------------------------------
Dcl-DS SrcDS LikeDS(TgtDS);
Dcl-DS TgtDS Len(100);
    Field1 Char(20);
    Field2 Packed(15:2);
    Field3 Date;
End-DS;

// 整体赋值
TgtDS = SrcDS;

// 清空数据结构
Clear TgtDS;

// 用特定值填充
TgtDS = *All'0';                    // 填充 '0'
TgtDS = *AllX'00';                  // 填充空字符

// ------------------------------------------------------------
// 2. 子字段操作
// ------------------------------------------------------------
Dcl-DS FullName Len(100);
    FirstName Char(30);
    LastName Char(30);
End-DS;

// 使用 %Subst 访问部分
Dcl-S FirstInitial Char(1);
FirstInitial = %Subst(FullName:1:1);

// 使用 %Replace 替换
FullName = %Replace('John': FullName: 1: %Len(%Trim(FirstName)));

// ------------------------------------------------------------
// 3. 数据结构作为参数
// ------------------------------------------------------------
Dcl-Proc ProcessOrder;
```

```
    Dcl-Pi *N;
        OrderDS LikeDS(OrderTemplate) Const;
    End-Pi;

    // 处理订单数据结构
    Dsply OrderDS.OrderNo;
End-Proc;

// ------------------------------------------------------------
// 4. 基于指针的数据结构
// ------------------------------------------------------------
Dcl-S Buffer Pointer;
Dcl-DS MessageDS Based(Buffer) Qualified;
    MsgLen Int(10);
    MsgType Char(10);
    MsgData Char(1000);
End-DS;

// 分配内存
Buffer = %Alloc(%Size(MessageDS));

// 使用数据结构
MessageDS.MsgLen = 100;
MessageDS.MsgType = 'INFORMATION';

// 释放内存
Dealloc Buffer;

// ------------------------------------------------------------
// 5. 数据结构大小
// ------------------------------------------------------------
Dcl-DS MyDS Len(256);
    // ...
End-DS;

Dcl-S SizeOfDS Int(10);
SizeOfDS = %Size(MyDS);              // 获取总大小
SizeOfDS = %Size(MyDS.Field1);       // 获取字段大小
```

第11章：文件操作

11.1 文件声明

11.1.1 文件类型

```
// ============================================================
// 文件声明详解
// ============================================================

// ------------------------------------------------------------
// 1. 磁盘文件 (DISK)
// ------------------------------------------------------------

// 外部描述文件（推荐使用）
Dcl-F CUSTMAST Disk(*Ext) Usage(*Input) Keyed;
Dcl-F CUSTUPD Disk(*Ext) Usage(*Update) Keyed;
Dcl-F CUSTOUT Disk(*Ext) Usage(*Output);

// 组合用法
Dcl-F CUSTFILE Disk(*Ext) Usage(*Input:*Output:*Update) Keyed;

// 特定成员
Dcl-F LOGFILE Disk(*Ext) Usage(*Output) Mbr('JAN2024');

// 带 INFDS
Dcl-F CUSTMAST Disk(*Ext) Usage(*Input) Keyed InfDS(FileInfo);

// ------------------------------------------------------------
// 2. 工作站文件 (WORKSTN)
// ------------------------------------------------------------
Dcl-F DSPFILE WorkStn;
Dcl-F INQFILE WorkStn SFile(SFL01:RRN);

// 带 INFDS
Dcl-F DSPFILE WorkStn InfDS(DspInfo);

// ------------------------------------------------------------
// 3. 打印机文件 (PRINTER)
// ------------------------------------------------------------
Dcl-F RPTFILE Printer Usage(*Output) OfLInd(*InOf);

// ------------------------------------------------------------
// 4. 顺序文件 (SPECIAL)
```

```
// ------------------------------------------------------------
Dcl-F STDIN Disk(*Ext) Usage(*Input) RcvLen(256);
Dcl-F STDOUT Disk(*Ext) Usage(*Output) SndLen(256);
```

11.1.2 文件属性

```
// ============================================================
// 文件属性配置
// ============================================================

// ------------------------------------------------------------
// 1. 磁盘文件属性
// ------------------------------------------------------------
Dcl-F CUSTMAST Disk(*Ext)
    Usage(*Input)           // 用途: *Input, *Output, *Update, *Delete
    Keyed                   // 按键访问
    InfDS(FileInfo)         // 文件信息数据结构
    RenFmt(CUSTREC)         // 重命名记录格式
    Prefix('CU_')           // 字段名前缀
    Commit                  // 启用提交控制
    UsrOpn;                 // 用户控制打开

// ------------------------------------------------------------
// 2. 工作站文件属性
// ------------------------------------------------------------
Dcl-F DSPFILE WorkStn
    SFile(SFL01:RRN01)      // 子文件声明
    InfDS(DspInfo)
    IndDs(DspInds)          // 指示器数据结构
    SFileRcdNbr(SubRrn);    // 子文件记录号

// ------------------------------------------------------------
// 3. 打印机文件属性
// ------------------------------------------------------------
Dcl-F RPTFILE Printer
    OfLInd(*InOf)           // 溢出指示器
    InfDS(PrtInfo)
    UsrOpn;
```

11.2 记录操作

11.2.1 读操作

```
// ============================================================
// 读操作详解
// ============================================================

// ------------------------------------------------------------
// 1. CHAIN - 按键随机读取
// ------------------------------------------------------------
// 读取单条记录
Chain CustKey CUSTMAST;
If %Found(CUSTMAST);
    // 找到记录，处理数据
    ProcessCustomer();
Else;
    // 未找到记录
    ShowError('Customer not found');
EndIf;

// 带锁读取（更新用）
Chain (CustKey) CUSTUPD;
If %Found(CUSTUPD);
    // 记录已被锁定，可以更新
    Status = 'I';
    Update CUSTREC;
EndIf;

// 带不锁选项读取
Chain (N) (CustKey) CUSTUPD;  // (N) = 不锁定

// ------------------------------------------------------------
// 2. SETLL / READE - 范围读取
// ------------------------------------------------------------
// 设置下限
SetLL CustKey CUSTMAST;

// 读取相等键
ReadE CustKey CUSTMAST;
Dow Not %Eof(CUSTMAST);
    ProcessRecord();
    ReadE CustKey CUSTMAST;
EndDo;
```

```
// 部分键匹配
SetLL PartialKey CUSTMAST;
ReadE PartialKey CUSTMAST;

// ---------------------------------------------------------------
// 3. SETGT / READP - 反向读取
// ---------------------------------------------------------------
// 设置上限（大于）
SetGT CustKey CUSTMAST;

// 向前读取
ReadP CUSTMAST;
Dow Not %Eof(CUSTMAST);
    ProcessReverse();
    ReadP CUSTMAST;
EndDo;

// 读取前一个相等
ReadPE CustKey CUSTMAST;

// ---------------------------------------------------------------
// 4. READ - 顺序读取
// ---------------------------------------------------------------
// 从当前位置向前读取
Read CUSTMAST;
Dow Not %Eof(CUSTMAST);
    ProcessRecord();
    Read CUSTMAST;
EndDo;

// ---------------------------------------------------------------
// 5. 文件定位操作
// ---------------------------------------------------------------
// 定位到文件开始
SetLL *Start CUSTMAST;

// 定位到文件结束
SetLL *End CUSTMAST;

// 检查记录是否存在（不读取数据）
SetLL CustKey CUSTMAST;
If %Equal(CUSTMAST);
    // 记录存在
EndIf;
```

11.2.2 写操作

```
// ============================================================
// 写操作详解
// ============================================================

// ------------------------------------------------------------
// 1. WRITE - 写入新记录
// ------------------------------------------------------------
// 写入输出文件
OrderNo = GetNextOrderNo();
OrderDate = %Date();
CustNo = InputCustNo;
Amount = CalculateTotal();

Write ORDREC;

// 写入日志文件
LogTime = %Timestamp();
LogUser = %User();
LogAction = 'CREATE';

Write LOGREC;

// ------------------------------------------------------------
// 2. UPDATE - 更新记录
// ------------------------------------------------------------
// 先读取记录（带锁）
Chain CustKey CUSTUPD;
If %Found(CUSTUPD);
    // 修改字段
    Phone = NewPhone;
    Email = NewEmail;
    LastUpdate = %Timestamp();

    // 更新记录
    Update CUSTREC;
EndIf;

// 使用别名更新
Chain CustKey CUSTUPD Alias;
If %Found(CUSTUPD);
    Update CUSTALIAS;
EndIf;
```

```
// ------------------------------------------------------------
// 3. DELETE - 删除记录
// ------------------------------------------------------------
// 删除记录
Chain CustKey CUSTUPD;
If %Found(CUSTUPD);
    Delete CUSTREC;
EndIf;

// 删除后检查
If %Found(CUSTUPD);
    // 删除成功
EndIf;

// ------------------------------------------------------------
// 4. 批量操作
// ------------------------------------------------------------
// 批量写入
For I = 1 To LineCount;
    LineItem = LineItems(I);
    Write LINREC;
EndFor;

// 批量更新
SetLL LowKey CUSTUPD;
ReadE LowKey CUSTUPD;
Dow Not %Eof(CUSTUPD);
    Status = 'X';  // 标记为删除
    Update CUSTREC;
    ReadE LowKey CUSTUPD;
EndDo;
```

11.3 错误处理

11.3.1 文件状态检查

```
// ============================================================
// 文件错误处理
// ============================================================

// ------------------------------------------------------------
// 1. 使用 %Found, %Eof, %Equal
// ------------------------------------------------------------
// CHAIN 后检查
```

```
Chain CustKey CUSTMAST;
If %Found(CUSTMAST);
    // 处理找到的记录
Else;
    // 处理未找到
EndIf;

// READ 后检查
Read CUSTMAST;
Dow Not %Eof(CUSTMAST);
    ProcessRecord();
    Read CUSTMAST;
EndDo;

// SETLL 后检查
SetLL CustKey CUSTMAST;
If %Equal(CUSTMAST);
    // 记录存在
EndIf;

// ---------------------------------------------------------------
// 2. 使用 INFDS
// ---------------------------------------------------------------
Dcl-F CUSTMAST Disk(*Ext) InfDS(FileInfo);

Dcl-DS FileInfo;
    Status Char(10) Pos(11);
    OpCode Char(6) Pos(21);
    RowNum Int(10) Pos(33);
End-DS;

// 检查文件状态
Monitor;
    Chain BadKey CUSTMAST;
On-Error *Program;
    // 通过 INFDS 获取错误详情
    Dsply ('File error: ' + Status);
EndMon;

// ---------------------------------------------------------------
// 3. 使用文件状态指示器
// ---------------------------------------------------------------
Dcl-F CUSTMAST Disk(*Ext) Usage(*Input)
    Keyed
    InfDS(FileInfo);
```

```
// 在 E 规范中设置错误指示器（固定格式）
// FFilenameIPEAF....RLEN....K...K......
// F           K                       99

// 自由格式中使用 Monitor
Monitor;
    Read CUSTMAST;
On-Error *File;
    // 文件级错误处理
On-Error *Program;
    // 程序级错误处理
EndMon;
```

11.4 高级文件操作

11.4.1 用户控制打开

```
// ============================================================
// 用户控制打开 (USROPN)
// ============================================================

Dcl-F CUSTMAST Disk(*Ext) Usage(*Input) Keyed UsrOpn;
Dcl-F ORDFILE Disk(*Ext) Usage(*Output) UsrOpn;

// 程序开始时文件未打开
// 需要手动打开

// 打开文件
Open CUSTMAST;
Open ORDFILE;

// 正常文件操作
Chain CustKey CUSTMAST;
// ...

// 关闭文件
Close CUSTMAST;
Close ORDFILE;

// ------------------------------------------------------------
// 使用场景：
// ------------------------------------------------------------
// 1. 条件打开
If (ProcessCustomer);
```

```
    Open CUSTMAST;
    // 处理...
    Close CUSTMAST;
EndIf;

// 2. 动态文件名
Dcl-F DATAFILE Disk(*Ext) Usage(*Input) UsrOpn;
// ...
// 使用 Override
// OVRDBF FILE(DATAFILE) TOFILE(MYLIB/MYDATA)
Open DATAFILE;
// ...
Close DATAFILE;
```

11.4.2 提交控制

```
// ============================================================
// 提交控制 (Commitment Control)
// ============================================================

// 声明带提交控制的文件
Dcl-F CUSTMAST Disk(*Ext) Usage(*Update) Keyed Commit;
Dcl-F ORDMAST Disk(*Ext) Usage(*Output) Commit;

// 开始提交控制
BegSr StartCommit;
    // 确保日志存在并启动提交控制
    Exec SQL
        SET OPTION COMMIT = *CHG;

    // 或者使用 CL 命令
    // STRCMTCTL LCKLVL(*CHG)
EndSr;

// 业务事务处理
BegSr ProcessTransaction;
    // 开始事务
    BegSr UpdateCustomer;
        Chain CustKey CUSTMAST;
        If %Found(CUSTMAST);
            Balance += OrderAmount;
            Update CUSTREC;
        EndIf;
    EndSr;
```

```
    BegSr CreateOrder;
        OrderNo = GetNextOrderNo();
        OrderDate = %Date();
        // ... 填充其他字段
        Write ORDREC;
    EndSr;

    // 提交事务
    Commit;
On-Error *All;
    // 出错时回滚
    RolBk;
    // 记录错误
    LogError('Transaction failed');
EndSr;

// 结束提交控制
BegSr EndCommit;
    // 使用 CL 命令
    // ENDCMTCTL
EndSr;
```

第12章：子过程与模块化编程

12.1 子过程基础

12.1.1 子过程定义

```
// ============================================================
// 子过程 (Subprocedure) 详解
// ============================================================

// ------------------------------------------------------------
// 1. 基本子过程结构
// ------------------------------------------------------------
Dcl-Proc ProcessOrder;
    Dcl-Pi *N;                      // 过程接口
        OrderNo Int(10) Const;      // 输入参数
        OrderTotal Packed(15:2);    // 输出参数
    End-Pi;
```

```
    // 局部变量
    Dcl-S LineTotal Packed(15:2);
    Dcl-S TaxAmount Packed(15:2);

    // 过程逻辑
    LineTotal = CalculateLines(OrderNo);
    TaxAmount = LineTotal * GetTaxRate();
    OrderTotal = LineTotal + TaxAmount;

    // 记录日志
    LogOrderProcessed(OrderNo, OrderTotal);
End-Proc;

// ------------------------------------------------------------
// 2. 返回值类型
// ------------------------------------------------------------
// 返回整数值
Dcl-Proc GetCustomerCount Int(10);
    Dcl-Pi *N Int(10);
        Status Char(1) Const;
    End-Pi;

    Dcl-S Count Int(10) Inz(0);

    SetLL *Start CUSTMAST;
    Read CUSTMAST;
    Dow Not %Eof(CUSTMAST);
        If (CustStatus = Status);
            Count += 1;
        EndIf;
        Read CUSTMAST;
    EndDo;

    Return Count;
End-Proc;

// 返回指示器
Dcl-Proc IsValidCustomer Ind;
    Dcl-Pi *N Ind;
        CustNo Int(10) Const;
    End-Pi;

    Chain CustNo CUSTMAST;
    Return %Found(CUSTMAST);
End-Proc;
```

```
// ------------------------------------------------------------
// 3. 参数传递模式
// ------------------------------------------------------------
Dcl-Proc UpdateCustomer;
    Dcl-Pi *N;
        // By Reference (默认) - 可修改原始值
        CustRec LikeDS(CustomerDS);

        // By Value - 传递值的副本
        NewStatus Char(1) Value;

        // Constant - 只读，不能修改
        UserId Char(10) Const;

        // By Reference 带选项
        ErrorMsg Char(100) Options(*Varsize);
    End-Pi;

    // 修改 CustRec 会影响调用者的变量
    CustRec.Status = NewStatus;
    CustRec.LastUpdate = %Timestamp();
    CustRec.UpdatedBy = UserId;
End-Proc;

// ------------------------------------------------------------
// 4. 可选参数
// ------------------------------------------------------------
Dcl-Proc CalculatePrice;
    Dcl-Pi *N Packed(15:2);
        BasePrice Packed(15:2) Const;
        Quantity Int(10) Const;
        Discount Packed(3:2) Const Options(*NoPass);  // 可选
    End-Pi;

    Dcl-S Total Packed(15:2);
    Dcl-S Disc Packed(3:2) Inz(0);

    // 检查可选参数是否传递
    If %Parms >= %ParmNum(Discount);
        Disc = Discount;
    EndIf;

    Total = BasePrice * Quantity;
    Total = Total * (1 - Disc);
```

```
    Return Total;
End-Proc;

// 调用示例：
// Price = CalculatePrice(100.00: 5);        // 不使用折扣
// Price = CalculatePrice(100.00: 5: 0.10);  // 使用 10% 折扣
```

12.1.2 子过程作用域

```
// ============================================================
// 子过程作用域和生命周期
// ============================================================

// ------------------------------------------------------------
// 1. 自动存储（默认）
// ------------------------------------------------------------
Dcl-Proc AutoStorageExample;
    Dcl-Pi *N;
    End-Pi;

    Dcl-S Counter Int(10);           // 每次调用重新初始化

    Counter += 1;                    // 总是返回 1
    Return Counter;
End-Proc;

// ------------------------------------------------------------
// 2. 静态存储
// ------------------------------------------------------------
Dcl-Proc StaticStorageExample;
    Dcl-Pi *N Int(10);
    End-Pi;

    Dcl-S Counter Int(10) Static;    // 值在调用间保持

    Counter += 1;                    // 每次调用增加
    Return Counter;
End-Proc;

// ------------------------------------------------------------
// 3. 静态变量初始化
// ------------------------------------------------------------
Dcl-Proc InitializedStatic;
    Dcl-Pi *N;
```

```
    End-Pi;

    // 只在第一次调用时初始化
    Dcl-S Initialized Ind Static Inz(*Off);
    Dcl-S ConfigData Char(100) Static;

    If Not Initialized;
        ConfigData = LoadConfiguration();
        Initialized = *On;
    EndIf;

    // 使用 ConfigData...
End-Proc;
```

12.2 模块化编程

12.2.1 服务程序

```
// ============================================================
// 服务程序 (Service Program) 开发
// ============================================================

// ------------------------------------------------------------
// 1. 创建服务程序模块
// ------------------------------------------------------------
// 文件：UTILITIES.RPGLE

// 导出过程声明
Dcl-Proc GetCurrentDate Export;
    Dcl-Pi *N Date;
        Format Char(10) Const Options(*NoPass);
    End-Pi;

    If %Parms >= %ParmNum(Format);
        Select;
            When Format = '*USA';
                Return %Date():*USA;
            When Format = '*EUR';
                Return %Date():*EUR;
            Other;
                Return %Date();
        EndSl;
    EndIf;
```

```
    Return %Date();
End-Proc;

Dcl-Proc FormatCurrency Export;
    Dcl-Pi *N Varchar(30);
        Amount Packed(15:2) Const;
        Currency Char(3) Const Options(*NoPass);
    End-Pi;

    Dcl-S Curr Char(3) Inz('USD');
    Dcl-S Result Varchar(30);

    If %Parms >= %ParmNum(Currency);
        Curr = Currency;
    EndIf;

    Result = %Trim(Curr) + ' ' + %Char(Amount:*Currency);
    Return Result;
End-Proc;

Dcl-Proc ValidateEmail Export;
    Dcl-Pi *N Ind;
        Email Varchar(256) Const;
    End-Pi;

    // 简单的电子邮件验证
    If %Scan('@':Email) = 0;
        Return *Off;
    EndIf;

    If %Scan('.':Email:%Scan('@':Email)) = 0;
        Return *Off;
    EndIf;

    Return *On;
End-Proc;

// ------------------------------------------------------------
// 2. 绑定语言源文件 (UTILITIES.BND)
// ------------------------------------------------------------
// 用于指定导出哪些过程
/*
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('UTIL001')
    EXPORT SYMBOL('GetCurrentDate')
    EXPORT SYMBOL('FormatCurrency')
    EXPORT SYMBOL('ValidateEmail')
```

```
ENDPGMEXP
*/

// ------------------------------------------------------------
// 3. 创建服务程序
// ------------------------------------------------------------
/*
编译步骤：
1. 编译模块
   CRTRPGMOD MODULE(MYLIB/UTILITIES) SRCFILE(MYLIB/QRPGLESRC)

2. 创建服务程序
   CRTSRVPGM SRVPGM(MYLIB/UTILITIES) MODULE(UTILITIES)
             EXPORT(*SRCFILE) SRCFILE(MYLIB/QSRVSRC)
             BNDDIR(MYLIB/UTILBND)

3. 创建绑定目录（如果不存在）
   CRTBNDDIR BNDDIR(MYLIB/UTILBND)
   ADDBNDDIRE BNDDIR(MYLIB/UTILBND) OBJ((MYLIB/UTILITIES *SRVPGM))
*/

// ------------------------------------------------------------
// 4. 使用服务程序
// ------------------------------------------------------------
// 文件：MAINPGM.RPGLE

Ctl-Opt BndDir('UTILBND');

// 原型声明
Dcl-Pr GetCurrentDate Date ExtProc(*CWIDEN:'GetCurrentDate');
    Format Char(10) Const Options(*NoPass);
End-Pr;

Dcl-Pr FormatCurrency Varchar(30) ExtProc(*CWIDEN:'FormatCurrency');
    Amount Packed(15:2) Const;
    Currency Char(3) Const Options(*NoPass);
End-Pr;

Dcl-Pr ValidateEmail Ind ExtProc(*CWIDEN:'ValidateEmail');
    Email Varchar(256) Const;
End-Pr;

// 主程序
Dcl-S Today Date;
Dcl-S PriceText Varchar(30);
Dcl-S IsValid Ind;
```

```
Today = GetCurrentDate();
PriceText = FormatCurrency(1234.56:'USD');
IsValid = ValidateEmail('test@example.com');
```

12.2.2 绑定目录

```
// ============================================================
// 绑定目录 (Binding Directory)
// ============================================================

// 绑定目录管理命令：

// 创建绑定目录
// CRTBNDDIR BNDDIR(MYLIB/APPLBND) TEXT('Application Binding
Directory')

// 添加条目
// ADDBNDDIRE BNDDIR(MYLIB/APPLBND) +
//            OBJ((MYLIB/UTILITIES *SRVPGM) +
//                (MYLIB/DATABASE *SRVPGM) +
//                (MYLIB/SECURITY *SRVPGM))

// 在程序中使用
Ctl-Opt BndDir('APPLBND');

// 或者使用 H 规范（固定格式）
// H DFTACTGRP(*NO) ACTGRP(*NEW) BNDDIR('APPLBND')

// ------------------------------------------------------------
// 多个绑定目录
// ------------------------------------------------------------
Ctl-Opt BndDir('APPLBND':'QC2LE':'SSLBND');
// QC2LE   - C 运行时库
// SSLBND  - SSL/TLS 支持

// ------------------------------------------------------------
// 模块与服务程序组织建议
// ------------------------------------------------------------
/*
建议的项目结构：

MYLIB/
├── QRPGLESRC/          - RPG 源代码
```

```
│    ├── MAINPGM.RPGLE    - 主程序
│    ├── ORDERMOD.RPGLE   - 订单模块
│    └── CUSTMOD.RPGLE    - 客户模块
│
├── QMODLESRC/           - 模块对象
├── QSRVSRC/             - 服务程序绑定源
│    └── UTILITIES.BND
│
├── BNDDIR/
│    ├── APPLBND         - 应用程序绑定目录
│    └── UTILBND         - 工具绑定目录
│
└── SRVPGM/
     ├── UTILITIES       - 工具服务程序
     ├── ORDERUTIL       - 订单工具服务程序
     └── CUSTUTIL        - 客户服务程序
*/
```

第13章：嵌入式 SQL 编程

13.1 SQL 基础

13.1.1 SQL 简介

```
// ============================================================
// 嵌入式 SQL 编程
// ============================================================

// 嵌入式 SQL 的优势：
// 1. 直接使用 SQL 语句操作数据库
// 2. 支持复杂的查询和数据处理
// 3. 自动优化查询性能
// 4. 支持事务控制

// ------------------------------------------------------------
// 基本 SQL 语句类型
// ------------------------------------------------------------

// 数据查询语言 (DQL)
// SELECT - 检索数据
```

```
// 数据操作语言 (DML)
// INSERT - 插入数据
// UPDATE - 更新数据
// DELETE - 删除数据

// 数据定义语言 (DDL)
// CREATE - 创建对象
// ALTER  - 修改对象
// DROP   - 删除对象

// 事务控制语言 (TCL)
// COMMIT - 提交事务
// ROLLBACK - 回滚事务
```

13.1.2 设置编译选项

```
// ============================================================
// SQL 编译选项
// ============================================================

Ctl-Opt Option(*SrcStmt:*NoDebugIo)
        DftActGrp(*No)
        ActGrp(*NEW);

// 或者使用 SQL SET OPTION
Exec SQL
    Set Option
        Commit = *None,              // 提交控制
        DatFmt = *ISO,               // 日期格式
        TimFmt = *ISO,               // 时间格式
        Naming = *Sys,               // 命名约定 (*Sys 或 *Sql)
        CloSqlCsr = *EndMod;         // 游标关闭选项

// 命名约定说明：
// *Sys (系统命名) - 使用 / 分隔符: LIBRARY/FILE
// *Sql (SQL 命名) - 使用 . 分隔符: LIBRARY.FILE
```

13.2 基本 SQL 操作

13.2.1 SELECT 语句

```
// ============================================================
// SELECT 语句
// ============================================================

// ------------------------------------------------------------
// 1. 单行 SELECT (INTO)
// ------------------------------------------------------------
Dcl-S CustName Char(50);
Dcl-S CustStatus Char(1);
Dcl-S OrderTotal Packed(15:2);

// 简单查询
Exec SQL
    SELECT CUST_NAME, STATUS
    INTO :CustName, :CustStatus
    FROM CUSTOMER
    WHERE CUST_NO = :CustNo;

// 检查查询结果
If SQLCODE = 0;
    // 成功
ElsIf SQLCODE = 100;
    // 未找到记录
Else;
    // 错误
EndIf;

// ------------------------------------------------------------
// 2. 使用数据结构
// ------------------------------------------------------------
Dcl-DS CustDS Qualified;
    CustNo Int(10);
    CustName Char(50);
    Address Char(100);
    City Char(30);
    Status Char(1);
End-DS;

Exec SQL
    SELECT *
    INTO :CustDS
```

```
    FROM CUSTOMER
    WHERE CUST_NO = :InputCustNo;

// ------------------------------------------------------------
// 3. 聚合函数
// ------------------------------------------------------------
Dcl-S OrderCount Int(10);
Dcl-S TotalAmount Packed(15:2);
Dcl-S AvgAmount Packed(15:2);

Exec SQL
    SELECT COUNT(*), SUM(AMOUNT), AVG(AMOUNT)
    INTO :OrderCount, :TotalAmount, :AvgAmount
    FROM ORDERS
    WHERE ORDER_DATE >= :StartDate
      AND ORDER_DATE <= :EndDate;

// ------------------------------------------------------------
// 4. 连接查询
// ------------------------------------------------------------
Dcl-DS OrderDetailDS Qualified;
    OrderNo Int(10);
    OrderDate Date;
    CustName Char(50);
    Amount Packed(15:2);
End-DS;

Exec SQL
    SELECT O.ORDER_NO, O.ORDER_DATE, C.CUST_NAME, O.AMOUNT
    INTO :OrderDetailDS
    FROM ORDERS O
    JOIN CUSTOMER C ON O.CUST_NO = C.CUST_NO
    WHERE O.ORDER_NO = :InputOrderNo;
```

13.2.2 INSERT、UPDATE、DELETE

```
// ============================================================
// 数据修改操作
// ============================================================

// ------------------------------------------------------------
// 1. INSERT
// ------------------------------------------------------------
// 单行插入
Exec SQL
```

```
    INSERT INTO CUSTOMER (CUST_NO, CUST_NAME, STATUS)
    VALUES (:NewCustNo, :NewCustName, 'A');

// 从数据结构插入
Exec SQL
    INSERT INTO CUSTOMER
    VALUES (:CustDS);

// 批量插入（使用数组）
Dcl-S CustNos Int(10) Dim(100);
Dcl-S CustNames Char(50) Dim(100);
Dcl-S i Int(10);

Exec SQL
    INSERT INTO CUSTOMER (CUST_NO, CUST_NAME)
    VALUES (:CustNos, :CustNames);

// ------------------------------------------------------------
// 2. UPDATE
// ------------------------------------------------------------
// 简单更新
Exec SQL
    UPDATE CUSTOMER
    SET STATUS = :NewStatus,
        LAST_UPDATE = CURRENT_TIMESTAMP
    WHERE CUST_NO = :CustNo;

// 带条件的更新
Exec SQL
    UPDATE ORDERS
    SET STATUS = 'C',
        COMPLETE_DATE = CURRENT_DATE
    WHERE ORDER_DATE < :CutoffDate
      AND STATUS = 'P';

// 获取更新行数
If SQLCODE = 0;
    RowsUpdated = SQLERRD(3);  // 更新的行数
EndIf;

// ------------------------------------------------------------
// 3. DELETE
// ------------------------------------------------------------
// 简单删除
Exec SQL
    DELETE FROM TEMP_ORDERS
```

```
    WHERE CREATE_DATE < :ExpireDate;

// 带检查
Exec SQL
    DELETE FROM CUSTOMER
    WHERE CUST_NO = :CustNo
      AND STATUS = 'I';  // 只允许删除非活动客户

// ------------------------------------------------------------
// 4. MERGE (UPSERT)
// ------------------------------------------------------------
Exec SQL
    MERGE INTO CUSTOMER AS T
    USING (VALUES (:CustNo, :CustName, :Status)) AS S(CNo, CName,
CStat)
    ON T.CUST_NO = S.CNo
    WHEN MATCHED THEN
        UPDATE SET CUST_NAME = S.CName,
                   STATUS = S.CStat
    WHEN NOT MATCHED THEN
        INSERT (CUST_NO, CUST_NAME, STATUS)
        VALUES (S.CNo, S.CName, S.CStat);
```

13.3 游标操作

13.3.1 声明和使用游标

```
// ============================================================
// 游标 (Cursor) 操作
// ============================================================

// ------------------------------------------------------------
// 1. 基本游标操作
// ------------------------------------------------------------

// 声明游标
Exec SQL
    DECLARE CustCursor CURSOR FOR
        SELECT CUST_NO, CUST_NAME, STATUS
        FROM CUSTOMER
        WHERE STATUS = :FilterStatus
        ORDER BY CUST_NAME;

// 打开游标
```

```
Exec SQL OPEN CustCursor;

// 获取数据
Exec SQL
    FETCH NEXT FROM CustCursor
    INTO :CustNo, :CustName, :CustStatus;

Dow SQLCODE = 0;
    // 处理当前记录
    ProcessCustomer(CustNo, CustName, CustStatus);

    // 获取下一条
    Exec SQL
        FETCH NEXT FROM CustCursor
        INTO :CustNo, :CustName, :CustStatus;
EndDo;

// 关闭游标
Exec SQL CLOSE CustCursor;

// ------------------------------------------------------------
// 2. 滚动游标
// ------------------------------------------------------------
Exec SQL
    DECLARE ScrollCursor SCROLL CURSOR FOR
        SELECT *
        FROM ORDERS
        ORDER BY ORDER_DATE;

Exec SQL OPEN ScrollCursor;

// 获取第一条
Exec SQL FETCH FIRST FROM ScrollCursor INTO :OrderDS;

// 获取最后一条
Exec SQL FETCH LAST FROM ScrollCursor INTO :OrderDS;

// 获取前一条
Exec SQL FETCH PRIOR FROM ScrollCursor INTO :OrderDS;

// 相对定位
Exec SQL FETCH RELATIVE +5 FROM ScrollCursor INTO :OrderDS;

// 绝对定位
Exec SQL FETCH ABSOLUTE 10 FROM ScrollCursor INTO :OrderDS;
```

```
Exec SQL CLOSE ScrollCursor;

// ------------------------------------------------------------
// 3. 可更新游标
// ------------------------------------------------------------
Exec SQL
    DECLARE UpdateCursor CURSOR FOR
        SELECT CUST_NO, STATUS
        FROM CUSTOMER
        WHERE STATUS = 'P'
        FOR UPDATE OF STATUS;

Exec SQL OPEN UpdateCursor;

Exec SQL
    FETCH NEXT FROM UpdateCursor
    INTO :CustNo, :CustStatus;

Dow SQLCODE = 0;
    // 更新当前行
    Exec SQL
        UPDATE CUSTOMER
        SET STATUS = 'A'
        WHERE CURRENT OF UpdateCursor;

    Exec SQL
        FETCH NEXT FROM UpdateCursor
        INTO :CustNo, :CustStatus;
EndDo;

Exec SQL CLOSE UpdateCursor;
```

13.4 高级 SQL 特性

13.4.1 存储过程调用

```
// ============================================================
// 调用存储过程
// ============================================================

// ------------------------------------------------------------
// 1. 调用外部存储过程
// ------------------------------------------------------------
```

```
// 声明存储过程接口
Dcl-Pr ProcessOrder ExtPgm('PROCESSORDER');
    OrderNo Int(10) Const;
    Result Char(1);
    ErrorMsg Char(100);
End-Pr;

// 调用存储过程
Dcl-S ResultCode Char(1);
Dcl-S ErrorMessage Char(100);

CallP ProcessOrder(OrderNo:ResultCode:ErrorMessage);

// ------------------------------------------------------------
// 2. 使用 CALL 语句调用 SQL 存储过程
// ------------------------------------------------------------
Exec SQL
    CALL PROCESS_ORDER(:OrderNo, :ResultCode, :ErrorMessage);

// ------------------------------------------------------------
// 3. 返回结果集的存储过程
// ------------------------------------------------------------
Exec SQL
    DECLARE ResultCursor CURSOR WITH RETURN FOR
        CALL GET_CUSTOMER_REPORT(:StartDate, :EndDate);

Exec SQL OPEN ResultCursor;
```

13.4.2 错误处理

```
// ============================================================
// SQL 错误处理
// ============================================================

// ------------------------------------------------------------
// 1. 使用 SQLCODE 和 SQLSTATE
// ------------------------------------------------------------
Dcl-S SQLCODE Int(10);
Dcl-S SQLSTATE Char(5);

Exec SQL
    SELECT * INTO :CustDS
    FROM CUSTOMER
    WHERE CUST_NO = :CustNo;
```

```
Select;
    When SQLCODE = 0;
        // 成功
        ProcessCustomer();
    When SQLCODE = 100;
        // 未找到
        ShowMessage('Customer not found');
    When SQLCODE < 0;
        // 错误
        ShowError('SQL Error: ' + %Char(SQLCODE));
    Other;
        // 警告
        ShowWarning('SQL Warning: ' + %Char(SQLCODE));
EndSl;

// ----------------------------------------------------------
// 2. 使用 SQL 诊断区
// ----------------------------------------------------------
Dcl-S ErrMsg Char(256);

Exec SQL
    GET DIAGNOSTICS CONDITION 1
        :ErrMsg = MESSAGE_TEXT;

// ----------------------------------------------------------
// 3. 使用处理程序
// ----------------------------------------------------------
Exec SQL
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        GET DIAGNOSTICS CONDITION 1
            :ErrMsg = MESSAGE_TEXT;
        :HasError = 'Y';
    END;

Exec SQL
    DECLARE EXIT HANDLER FOR SQLWARNING
    BEGIN
        // 警告处理
        :HasWarning = 'Y';
    END;

// ----------------------------------------------------------
// 4. 条件处理
// ----------------------------------------------------------
Exec SQL
```

```
    DECLARE Not_Found CONDITION FOR SQLSTATE '02000';

    DECLARE CONTINUE HANDLER FOR Not_Found
        SET :EndOfData = 'Y';
```

第14章：ILE 概念与程序绑定

14.1 ILE 架构

14.1.1 ILE 概述

```
ILE (Integrated Language Environment) 架构：

┌─────────────────────────────────────────────────────────────┐
│                        应用程序层                            │
│          (程序 PGM 或服务程序 SRVPGM)                        │
├─────────────────────────────────────────────────────────────┤
│                        模块层                               │
│                (*MODULE 对象)                               │
├─────────────────────────────────────────────────────────────┤
│                        编译单元                             │
│          (源代码编译后的中间代码)                           │
├─────────────────────────────────────────────────────────────┤
│                        激活组                               │
│          (Activation Group - 运行时环境)                    │
└─────────────────────────────────────────────────────────────┘

ILE 优势：
1. 代码重用（服务程序）
2. 更好的内存管理
3. 更快的调用速度
4. 支持混合语言编程
5. 更好的错误隔离
```

14.1.2 激活组

```
// ============================================================
// 激活组 (Activation Group)
// ============================================================
```

```
// 激活组类型：
// *NEW       - 每次调用创建新激活组
// *CALLER    - 使用调用者的激活组
// *DFTACTGRP - 默认激活组（向后兼容）
// 命名激活组 - 指定名称，可共享

// ----------------------------------------------------------------
// 1. 创建程序时指定激活组
// ----------------------------------------------------------------
Ctl-Opt DftActGrp(*No) ActGrp(*NEW);
// 或使用
Ctl-Opt DftActGrp(*No) ActGrp('ORDERAPP');

// ----------------------------------------------------------------
// 2. 激活组管理
// ----------------------------------------------------------------
// 在程序中设置返回
Ctl-Opt DftActGrp(*No) ActGrp(*NEW) RTNPARM(*YES);

// 使用 RETURN 代替 SETON LR
Return;

// ----------------------------------------------------------------
// 3. 激活组生命周期管理
// ----------------------------------------------------------------
// 程序正常结束时：
// - 如果是 *NEW 激活组，激活组被删除
// - 如果是命名激活组，激活组保留
// - 如果是 *CALLER，不控制激活组

// RCLACTGRP 命令回收激活组
// RCLACTGRP ACTGRP(ORDERAPP)

// ----------------------------------------------------------------
// 4. 激活组考虑因素
// ----------------------------------------------------------------
// 选择 *NEW 的情况：
// - 独立的应用程序
// - 需要完全隔离
// - 不影响其他程序的状态

// 选择命名激活组的情况：
// - 相关程序组
// - 需要共享资源
```

```
// - 需要在程序间保持状态

// 选择 *CALLER 的情况：
// - 工具程序
// - 需要访问调用者的资源
// - 子过程
```

14.2 程序创建

14.2.1 单步创建与多步创建

```
// ============================================================
// 程序创建方法
// ============================================================

// ------------------------------------------------------------
// 1. 单步创建（适合简单程序）
// ------------------------------------------------------------
// 命令：CRTBNDRPG
//
// CRTBNDRPG PGM(MYLIB/MYPGM)
//           SRCFILE(MYLIB/QRPGLESRC)
//           SRCMBR(MYPGM)
//           DFTACTGRP(*NO)
//           ACTGRP(*NEW)
//           BNDDIR(MYLIB/MYBND)

// 在代码中指定：
Ctl-Opt DftActGrp(*No) ActGrp(*NEW);

// ------------------------------------------------------------
// 2. 多步创建（适合模块化程序）
// ------------------------------------------------------------
// 步骤 1：编译模块
// CRTRPGMOD MODULE(MYLIB/MODULE1)
//           SRCFILE(MYLIB/QRPGLESRC)
//           SRCMBR(MODULE1)
//
// CRTRPGMOD MODULE(MYLIB/MODULE2)
//           SRCFILE(MYLIB/QRPGLESRC)
//           SRCMBR(MODULE2)

// 步骤 2：绑定程序
// CRTPGM PGM(MYLIB/MYPGM)
```

```
//         MODULE(MYLIB/MODULE1 MYLIB/MODULE2)
//         BNDSRVPGM(MYLIB/UTILS)
//         ACTGRP(*NEW)

// ------------------------------------------------------------
// 3. UPDPGM 更新程序
// ------------------------------------------------------------
// 只替换特定模块而不重新绑定整个程序
// UPDPGM PGM(MYLIB/MYPGM) MODULE(MYLIB/NEWMOD)
```

14.2.2 更新服务程序

```
// ============================================================
// 服务程序版本管理
// ============================================================

// ------------------------------------------------------------
// 1. 创建带版本的导出源
// ------------------------------------------------------------
// UTILITIES.BND 文件内容：

// 当前版本
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('UTIL002')
    EXPORT SYMBOL('GetCurrentDate')
    EXPORT SYMBOL('FormatCurrency')
    EXPORT SYMBOL('ValidateEmail')
    EXPORT SYMBOL('NewFunction')     // 新增函数
ENDPGMEXP

// 上一版本（向后兼容）
STRPGMEXP PGMLVL(*PRV) SIGNATURE('UTIL001')
    EXPORT SYMBOL('GetCurrentDate')
    EXPORT SYMBOL('FormatCurrency')
    EXPORT SYMBOL('ValidateEmail')
ENDPGMEXP

// ------------------------------------------------------------
// 2. 更新服务程序
// ------------------------------------------------------------
// UPDSRVPGM SRVPGM(MYLIB/UTILITIES) +
//           MODULE(MYLIB/UTILITIES) +
//           SRCFILE(MYLIB/QSRVSRC) +
//           SRCMBR(UTILITIES)
```

```
// ------------------------------------------------------------
// 3. 签名验证
// ------------------------------------------------------------
// 程序绑定到服务程序时会记录签名
// 如果签名不匹配，需要在更新后重新绑定程序
```

第四部分：CLLE 编程详解

第15章：CL 控制语言基础

15.1 CL 语言概述

15.1.1 CL 的作用和特点

```
/* ============================================================ */
/* CL (Control Language) 概述                                   */
/* ============================================================ */

/* CL 的主要用途：
   1. 系统管理任务自动化
   2. 程序调度和控制
   3. 对象管理
   4. 错误处理和消息管理
   5. 用户界面创建
*/

/* CL 程序类型：
   - OPM CL (CLP)  : 原始程序模型，源类型 CLP
   - ILE CL (CLLE) : 集成语言环境，源类型 CLLE (推荐)
*/
```

15.1.2 命令结构

```
CL 命令命名规范：

动词 + 对象 = 命令

常用动词：
```

缩写	全称	示例
CRT	Create	CRTLIB, CRTPF, CRTPGM
DLT	Delete	DLTF, DLTUSRPRF
CHG	Change	CHGUSRPRF, CHGJOB
DSP	Display	DSPJOB, DSPLIB
WRK	Work With	WRKACTJOB, WRKOUTQ
STR	Start	STRSBS, STRTCP
END	End	ENDJOB, ENDSBS
ADD	Add	ADDLIBLE, ADDJOBSCDE
RMV	Remove	RMVLIBLE, RMVJOBSCDE
CPY	Copy	CPYF, CPYLIB
MOV	Move	MOVOBJ, MOVJOB
SND	Send	SNDMSG, SNDPGMMSG
RCV	Receive	RCVMSG, RCVF
CALL	Call	CALL, CALLPRC
SBM	Submit	SBMJOB

常用对象类型：

缩写	对象类型
LIB	Library
PF	Physical File
LF	Logical File
PGM	Program
CMD	Command
JOB	Job
SBSD	Subsystem
JOBD	Job Description
OUTQ	Output Queue
MSGQ	Message Queue
USRPRF	User Profile

15.2 CL 程序结构

15.2.1 基本程序框架

```
/* ============================================================ */
/* CL 程序基本结构                                              */
/* ============================================================ */
```

```
             PGM        PARM(&INPUT1 &INPUT2)

/* 声明部分 */
             DCL        VAR(&INPUT1) TYPE(*CHAR) LEN(10)
             DCL        VAR(&INPUT2) TYPE(*DEC) LEN(7 0)
             DCL        VAR(&COUNT) TYPE(*DEC) LEN(5 0) +
                          VALUE(0)
             DCL        VAR(&RTNCDE) TYPE(*CHAR) LEN(10)
             DCL        VAR(&CONTINUE) TYPE(*LGL) VALUE('1')

/* 主要逻辑 */
             CHGVAR     VAR(&COUNT) VALUE(&COUNT + 1)

/* 调用程序 */
             CALL       PGM(PROGRAM1) PARM(&INPUT1 &RTNCDE)

/* 条件处理 */
             IF         COND(&RTNCDE *EQ 'OK') THEN(DO)
                /* 成功处理 */
                SNDPGMMSG  MSG('处理成功')
             ENDDO
             ELSE       CMD(DO)
                /* 错误处理 */
                SNDPGMMSG  MSG('处理失败') TOPGMQ(*PRV)
                RETURN
             ENDDO

/* 结束程序 */
             ENDPGM
```

15.2.2 程序组成部分

```
/* ============================================================ */
/* CL 程序组成部分详解                                          */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* 1. PGM/ENDPGM - 程序开始和结束                               */
/* ------------------------------------------------------------ */
             PGM        [PARM(参数列表)]
             /* 程序主体 */
             ENDPGM
```

```
/* ------------------------------------------------------------ */
/* 2. DCL - 声明命令                                            */
/* ------------------------------------------------------------ */
/* 字符变量 */
             DCL        VAR(&NAME) TYPE(*CHAR) LEN(50) +
                          VALUE('默认值')

/* 数值变量 */
             DCL        VAR(&AMOUNT) TYPE(*DEC) LEN(15 2) +
                          VALUE(0)

/* 逻辑变量 */
             DCL        VAR(&ACTIVE) TYPE(*LGL) VALUE('1')
/* '1' = 真, '0' = 假 */

/* 整数变量 (ILE CL) */
             DCL        VAR(&COUNT) TYPE(*INT) LEN(4)
/* LEN(2) = 2字节, LEN(4) = 4字节, LEN(8) = 8字节 */

/* 无符号整数 */
             DCL        VAR(&POS) TYPE(*UINT) LEN(4)

/* ------------------------------------------------------------ */
/* 3. 文件声明 */
/* ------------------------------------------------------------ */
             DCLF       FILE(MYLIB/MYFILE)

/* 带前缀（处理多个文件时避免字段名冲突） */
             DCLF       FILE(MYLIB/FILE1) PREFIX(F1_)
             DCLF       FILE(MYLIB/FILE2) PREFIX(F2_)
```

15.3 变量和数据类型

15.3.1 数据类型详解

```
/* ============================================================ */
/* CL 数据类型                                                  */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* *CHAR - 字符类型                                             */
/* ------------------------------------------------------------ */
             DCL        VAR(&NAME) TYPE(*CHAR) LEN(10)
/* 特性：
```

```
   - 定长，不足补空格
   - 最大长度：9999
   - 支持单引号或双引号
*/

/* ------------------------------------------------------------ */
/* *DEC - 压缩十进制                                            */
/* ------------------------------------------------------------ */
             DCL        VAR(&AMOUNT) TYPE(*DEC) LEN(15 2)
/* LEN(总位数 小数位数)
   - 总位数：1-15
   - 小数位数：0-9
   - 示例：LEN(7 2) 可表示 -99999.99 到 99999.99
*/

/* ------------------------------------------------------------ */
/* *LGL - 逻辑类型                                              */
/* ------------------------------------------------------------ */
             DCL        VAR(&FLAG) TYPE(*LGL)
/* 值：'1' (真) 或 '0' (假) */

/* ------------------------------------------------------------ */
/* *INT - 整数类型 (ILE CL)                                     */
/* ------------------------------------------------------------ */
             DCL        VAR(&SMALL) TYPE(*INT) LEN(2)
             DCL        VAR(&NORMAL) TYPE(*INT) LEN(4)
             DCL        VAR(&LARGE) TYPE(*INT) LEN(8)
/*
   LEN(2): -32768 到 32767
   LEN(4): -2147483648 到 2147483647
   LEN(8): -9223372036854775808 到 9223372036854775807
*/

/* ------------------------------------------------------------ */
/* *UINT - 无符号整数 (ILE CL)                                  */
/* ------------------------------------------------------------ */
             DCL        VAR(&INDEX) TYPE(*UINT) LEN(4)
/* 只支持正值，范围是 INT 的两倍正数范围 */

/* ------------------------------------------------------------ */
/* 特殊值                                                       */
/* ------------------------------------------------------------ */
/* 可以使用系统定义的特殊值： */
             DCL        VAR(&DATE) TYPE(*CHAR) LEN(6) +
                          VALUE(*SYSVAL)
```

```
/* 常用特殊值：
   *SYSVAL     - 系统值
   *JOB        - 作业默认值
   *USER       - 当前用户
   *DATE       - 当前日期
   *TIME       - 当前时间
   *NULL        - 空值
   *BLANKS      - 空格
   *ZERO        - 零
   *HIVAL       - 最大值
   *LOVAL       - 最小值
*/
```

15.3.2 变量操作

```
/* ============================================================ */
/* 变量操作命令                                                 */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* CHGVAR - 更改变量值                                          */
/* ------------------------------------------------------------ */
/* 简单赋值 */
            CHGVAR     VAR(&NAME) VALUE('John Smith')
            CHGVAR     VAR(&COUNT) VALUE(100)

/* 算术运算 */
            CHGVAR     VAR(&TOTAL) VALUE(&PRICE * &QTY)
            CHGVAR     VAR(&COUNT) VALUE(&COUNT + 1)
            CHGVAR     VAR(&REMAIN) VALUE(&TOTAL - &DISCOUNT)

/* 字符串连接 */
            CHGVAR     VAR(&FULLNAME) VALUE(&FIRST *CAT ' ' +
                         *CAT &LAST)
/* *CAT     - 直接连接
   *BCAT    - 带空格连接（插入一个空格）
   *TCAT    - 截断连接（去除尾部空格后连接）
*/

/* 使用内置函数 */
            CHGVAR     VAR(&UPPER) VALUE(%UPPER(&LOWER))
            CHGVAR     VAR(&SUB) VALUE(%SST(&STRING 1 10))
            CHGVAR     VAR(&BIN) VALUE(%BIN(&CHAR 1 4))
```

```
/* ------------------------------------------------------------ */
/* RTVOBJD - 检索对象描述                                       */
/* ------------------------------------------------------------ */
            RTVOBJD    OBJ(MYLIB/MYPGM) OBJTYPE(*PGM) +
                         RTNLIB(&RTNLIB) CRTDATE(&CRTDATE) +
                         OWNER(&OWNER)

/* ------------------------------------------------------------ */
/* RTVJOBA - 检索作业属性                                       */
/* ------------------------------------------------------------ */
            RTVJOBA    USER(&USER) CURLIB(&CURLIB) +
                         SWS(&SWITCHES) DATFMT(&DATEFMT)

/* ------------------------------------------------------------ */
/* RTVDAT - 检索日期                                            */
/* ------------------------------------------------------------ */
            RTVDAT     DATE(&DATE) DAYOFWEEK(&DAY)
            RTVDAT     DATE(&DATE) DAYOFMONTH(&DOM)
            RTVDAT     DATE(&DATE) MONTH(&MONTH) YEAR(&YEAR)

/* ------------------------------------------------------------ */
/* CVTDAT - 转换日期格式                                        */
/* ------------------------------------------------------------ */
            CVTDAT     DATE(&OLDDATE) TOVAR(&NEWDATE) +
                         FROMFMT(*MDY) TOFMT(*ISO)
/* 日期格式：
   *MDY  - 月/日/年 (06/30/24)
   *DMY  - 日/月/年 (30/06/24)
   *YMD  - 年/月/日 (24/06/30)
   *USA  - MM/DD/YYYY (06/30/2024)
   *ISO  - YYYY-MM-DD (2024-06-30)
   *EUR  - DD.MM.YYYY (30.06.2024)
   *JIS  - YYYY-MM-DD (2024-06-30)
   *CMDY - CMM/DD/YY (106/30/24)
   *CDMY - CDD/MM/YY (130/06/24)
   *CYMD - CYY/MM/DD (124/06/30)
   *LONGJUL - YYYYDDD (2024182)
   *JUL  - YYDDD (24182)
*/
```

15.4 控制流程

15.4.1 条件语句

```
/* ============================================================ */
/* 条件控制语句                                                 */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* IF/ELSE - 条件分支                                           */
/* ------------------------------------------------------------ */
/* 简单 IF */
             IF         COND(&COUNT *GT 0) THEN(DO)
                /* 条件为真时执行 */
                CALL       PGM(PROCESS)
             ENDDO

/* IF/ELSE */
             IF         COND(&STATUS *EQ 'A') THEN(DO)
                SNDPGMMSG  MSG('状态: 活动')
             ENDDO
             ELSE       CMD(DO)
                SNDPGMMSG  MSG('状态: 非活动')
             ENDDO

/* 嵌套 IF */
             IF         COND(&TYPE *EQ '1') THEN(DO)
                IF         COND(&AMT *GT 1000) THEN(DO)
                   CHGVAR     VAR(&DISC) VALUE(10)
                ENDDO
                ELSE       CMD(DO)
                   CHGVAR     VAR(&DISC) VALUE(5)
                ENDDO
             ENDDO
             ELSE       CMD(DO)
                CHGVAR     VAR(&DISC) VALUE(0)
             ENDDO

/* 复杂条件 */
             IF         COND((&COUNT *GE 1 *AND &COUNT *LE +
                           100) *OR &OVERRIDE *EQ '1') THEN(DO)
                /* 处理 */
             ENDDO

/* ------------------------------------------------------------ */
```

```
/* SELECT - 多分支选择                                              */
/* ---------------------------------------------------------------- */
             SELECT
             WHEN       COND(&OPTION *EQ '1') THEN(DO)
                CALL       PGM(PGM1)
             ENDDO
             WHEN       COND(&OPTION *EQ '2') THEN(DO)
                CALL       PGM(PGM2)
             ENDDO
             WHEN       COND(&OPTION *EQ '3' *OR &OPTION *EQ +
                          '4') THEN(DO)
                CALL       PGM(PGM3)
             ENDDO
             OTHERWISE  CMD(DO)
                SNDPGMMSG  MSG('无效选项')
             ENDDO
             ENDSELECT

/* ---------------------------------------------------------------- */
/* DOWHILE/DOUNTIL - 循环                                           */
/* ---------------------------------------------------------------- */
/* DOWHILE - 当条件为真时循环 */
             DOWHILE    COND(&CONTINUE *EQ '1')
                CHGVAR     VAR(&COUNT) VALUE(&COUNT + 1)
                /* 处理逻辑 */
                IF         COND(&COUNT *GE 10) THEN(CHGVAR +
                             VAR(&CONTINUE) VALUE('0'))
             ENDDO

/* DOUNTIL - 直到条件为真时退出（至少执行一次） */
             DOUNTIL    COND(&FOUND *EQ '1')
                RCVF
                MONMSG     MSGID(CPF0864) EXEC(LEAVE)
                IF         COND(&FILEFIELD *EQ &SEARCH) THEN(CHGVAR +
                             VAR(&FOUND) VALUE('1'))
             ENDDO

/* ---------------------------------------------------------------- */
/* DOFOR - 计数循环 (ILE CL)                                        */
/* ---------------------------------------------------------------- */
             DOFOR      VAR(&I) FROM(1) TO(10) BY(1)
                SNDPGMMSG  MSG('循环: ' *CAT %CHAR(&I))
             ENDDO

/* ---------------------------------------------------------------- */
/* GOTO/LEAVE/ITERATE                                               */
```

```
/* ------------------------------------------------------------ */
/* GOTO - 跳转到标签（不推荐过度使用） */
             IF         COND(&ERROR *EQ '1') THEN(GOTO +
                          CMDLBL(ERROR))
             /* 正常处理 */
             RETURN

 ERROR:      SNDPGMMSG  MSG('发生错误')
             RETURN

/* LEAVE - 退出循环 */
             DOWHILE    COND('1')
                RCVF
                MONMSG     MSGID(CPF0864) EXEC(LEAVE)
                /* 如果满足某个条件也退出 */
                IF         COND(&DONE *EQ '1') THEN(LEAVE)
             ENDDO

/* ITERATE - 跳过当前循环迭代 (ILE CL) */
             DOFOR      VAR(&I) FROM(1) TO(100)
                IF         COND(&SKIP *EQ '1') THEN(ITERATE)
                /* 处理 */
             ENDDO
```

第16章：CL 程序结构

16.1 程序和过程

16.1.1 程序调用

```
/* ============================================================ */
/* 程序调用                                                     */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* CALL - 调用程序                                              */
/* ------------------------------------------------------------ */
             PGM        PARM(&INPARM)

             DCL        VAR(&INPARM) TYPE(*CHAR) LEN(50)
             DCL        VAR(&OUTPARM) TYPE(*CHAR) LEN(50)
```

```
             DCL        VAR(&RTNCDE) TYPE(*DEC) LEN(10 0)

/* 带参数调用 */
             CALL       PGM(MYLIB/MYPGM) PARM(&INPARM &OUTPARM +
                          &RTNCDE)

/* 不带参数调用 */
             CALL       PGM(MYLIB/SUBPGM)

/* ------------------------------------------------------------ */
/* CALLPRC - 调用过程 (ILE CL)                                  */
/* ------------------------------------------------------------ */
             CALLPRC    PRC(MYPROC) PARM(&PARM1 &PARM2) +
                          RTNVAL(&RTNVAL)

/* 调用 C 函数 */
             CALLPRC    PRC('printf') PARM('Hello World')

/* ------------------------------------------------------------ */
/* TFRCTL - 转移控制                                            */
/* ------------------------------------------------------------ */
/* 结束当前程序并将控制转移给新程序，不返回 */
             TFRCTL     PGM(MENUPGM)

/* ------------------------------------------------------------ */
/* RETURN - 返回调用者                                          */
/* ------------------------------------------------------------ */
/* 从程序或子程序返回 */
             RETURN
```

16.1.2 子程序

```
/* ============================================================ */
/* 子程序 (Subroutine)                                          */
/* ============================================================ */

             PGM

             DCL        VAR(&VAR1) TYPE(*CHAR) LEN(10)

/* 主程序 */
             CALLSUBR   SUBR(INIT)
             CALLSUBR   SUBR(PROCESS)
             CALLSUBR   SUBR(CLEANUP)
```

```
            RETURN

/* 初始化子程序 */
 SUBROUT:    SUBR(INIT)
             CHGVAR     VAR(&VAR1) VALUE('INITIALIZED')
             SNDPGMMSG  MSG('初始化完成')
             ENDSUBR

/* 处理子程序 */
 SUBROUT:    SUBR(PROCESS)
             /* 处理逻辑 */
             SNDPGMMSG  MSG('处理完成')
             ENDSUBR

/* 清理子程序 */
 SUBROUT:    SUBR(CLEANUP)
             /* 清理逻辑 */
             SNDPGMMSG  MSG('清理完成')
             ENDSUBR

             ENDPGM
```

16.2 命令创建

16.2.1 创建用户命令

```
/* ============================================================ */
/* 创建用户命令 (User-Defined Commands)                         */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* 命令定义源 (CMDSRC)                                          */
/* ------------------------------------------------------------ */
/* 文件：WRKORDCMD.CMDSRC */

             CMD        PROMPT('Work with Orders')

             PARM       KWD(CUSTOMER) TYPE(*CHAR) LEN(10) +
                          MIN(1) PROMPT('Customer number')

             PARM       KWD(STATUS) TYPE(*CHAR) LEN(1) +
                          RSTD(*YES) VALUES(A I P C) +
                          DFT(*ALL) SPCVAL((*ALL ' ')) +
```

```
                          PROMPT('Order status')

             PARM       KWD(DATEFROM) TYPE(*DATE) +
                          DFT(*BEGIN) SPCVAL((*BEGIN 000000)) +
                          PROMPT('From date')

             PARM       KWD(DATETO) TYPE(*DATE) +
                          DFT(*END) SPCVAL((*END 999999)) +
                          PROMPT('To date')

/* ------------------------------------------------------------ */
/* 编译命令 */
/* ------------------------------------------------------------ */
/* CRTCMD CMD(MYLIB/WRKORD) PGM(MYLIB/WRKORDPGM) +
          SRCFILE(MYLIB/QCMDSRC) SRCMBR(WRKORDCMD) */

/* ------------------------------------------------------------ */
/* 命令处理程序 (ILE CL)                                        */
/* ------------------------------------------------------------ */
             PGM        PARM(&CUSTOMER &STATUS &DATEFROM &DATETO)

             DCL        VAR(&CUSTOMER) TYPE(*CHAR) LEN(10)
             DCL        VAR(&STATUS) TYPE(*CHAR) LEN(1)
             DCL        VAR(&DATEFROM) TYPE(*CHAR) LEN(7)
             DCL        VAR(&DATETO) TYPE(*CHAR) LEN(7)

/* 参数验证 */
             IF         COND(&CUSTOMER *EQ ' ') THEN(DO)
                SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) +
                             MSGDTA('Customer number is required') +
                             MSGTYPE(*ESCAPE)
             ENDDO

/* 调用主程序 */
             CALL       PGM(ORDMAIN) PARM(&CUSTOMER &STATUS +
                          &DATEFROM &DATETO)

             ENDPGM
```

第17章：文件与显示设备处理

17.1 数据库文件处理

17.1.1 DCLF 和 RCVF

```
/* ============================================================ */
/* 数据库文件处理                                               */
/* ============================================================ */

             PGM

/* 声明文件 */
             DCLF       FILE(MYLIB/CUSTMAST) +
                          RCDFMT(CUSTREC)

/* 读取文件循环 */
 LOOP:       RCVF
             MONMSG     MSGID(CPF0864) EXEC(GOTO CMDLBL(EOF))

/* 处理记录 - 使用文件字段 */
             IF         COND(&CUSTSTS *EQ 'A') THEN(DO)
                SNDPGMMSG  MSG('处理客户: ' *CAT &CUSTNO)
             ENDDO

             GOTO       CMDLBL(LOOP)

/* 文件结束 */
 EOF:        SNDPGMMSG  MSG('文件处理完成')

             ENDPGM
```

17.1.2 多文件处理

```
/* ============================================================ */
/* 多文件处理                                                   */
/* ============================================================ */

             PGM

/* 声明多个文件，使用前缀区分 */
             DCLF       FILE(MYLIB/ORDMAST) OPNID(ORD) +
```

```
                          RCDFMT(ORDREC)
             DCLF         FILE(MYLIB/CUSTMAST) OPNID(CUS) +
                          RCDFMT(CUSTREC)

/* 读取订单文件 */
 READORD:    RCVF         OPNID(ORD)
             MONMSG       MSGID(CPF0864) EXEC(GOTO CMDLBL(DONE))

/* 查找客户 */
             CHGVAR       VAR(&CUS_CUSTNO) VALUE(&ORD_CUSTNO)
             CHAIN        OPNID(CUS)

/* 处理 */
             SNDPGMMSG    MSG('订单: ' *CAT &ORD_ORDNO *CAT +
                            ' 客户: ' *CAT &CUS_CUSTNAME)

             GOTO         CMDLBL(READORD)

 DONE:       ENDPGM
```

17.2 显示文件处理

17.2.1 SNDRCVF

```
/* ============================================================ */
/* 显示文件处理                                                 */
/* ============================================================ */

             PGM

             DCLF         FILE(MYLIB/LOGIN) RCDFMT(LOGINSCRN)

/* 初始化 */
             CHGVAR       VAR(&IN03) VALUE('0')
             CHGVAR       VAR(&INUSER) VALUE(' ')
             CHGVAR       VAR(&ERRORMSG) VALUE(' ')

/* 显示屏幕 */
 SHOWSCRN:   SNDRCVF      RCDFMT(LOGINSCRN)

/* 检查 F3 */
             IF           COND(&IN03 *EQ '1') THEN(RETURN)

/* 验证输入 */
```

```
             IF         COND(&INUSER *EQ ' ') THEN(DO)
                CHGVAR     VAR(&ERRORMSG) VALUE('用户名不能为空')
                CHGVAR     VAR(&IN30) VALUE('1')
                GOTO       CMDLBL(SHOWSCRN)
             ENDDO

/* 验证密码 */
             CALL       PGM(VALPASS) PARM(&INUSER &INPASS +
                          &VALID)

             IF         COND(&VALID *NE '1') THEN(DO)
                CHGVAR     VAR(&ERRORMSG) VALUE('密码错误')
                CHGVAR     VAR(&IN30) VALUE('1')
                GOTO       CMDLBL(SHOWSCRN)
             ENDDO

/* 登录成功 */
             SNDPGMMSG  MSG('登录成功')
             CALL       PGM(MAINMENU) PARM(&INUSER)

             ENDPGM
```

17.2.2 子文件处理

```
/* ============================================================ */
/* 子文件处理                                                   */
/* ============================================================ */

             PGM

             DCLF       FILE(MYLIB/CUSTDSP) RCDFMT(SFLCTL +
                          SFLMSG SFL)

             DCL        VAR(&RRN) TYPE(*DEC) LEN(4 0) VALUE(0)
             DCL        VAR(&RRNC) TYPE(*DEC) LEN(4 0)

/* 打开查询文件 */
             OPNQRYF    FILE((MYLIB/CUSTMAST)) FORMAT(MYLIB/CUSTDSP) +
                          QRYSLT('CUSTSTS *EQ "A"')

/* 清空子文件 */
             CHGVAR     VAR(&RRN) VALUE(0)
             CHGVAR     VAR(&IN28) VALUE('1')
             SNDRCVF    RCDFMT(SFLCTL)
```

```
             CHGVAR     VAR(&IN28) VALUE('0')

/* 加载子文件 */
 LOAD:       RCVF
             MONMSG     MSGID(CPF0864) EXEC(GOTO CMDLBL(DISPLAY))

             CHGVAR     VAR(&RRN) VALUE(&RRN + 1)
             CHGVAR     VAR(&SFLOPT) VALUE(' ')
             CHGVAR     VAR(&SFLCUST) VALUE(&CUSTNO)
             CHGVAR     VAR(&SFLNAME) VALUE(&CUSTNAME)
             SNDRCVF    RCDFMT(SFL)

             IF         COND(&RRN *LT 9999) THEN(GOTO +
                          CMDLBL(LOAD))

/* 显示子文件 */
 DISPLAY:    CHGVAR     VAR(&SFLSIZ) VALUE(&RRN)
             CHGVAR     VAR(&SFLPAG) VALUE(15)

             IF         COND(&RRN *GT 0) THEN(DO)
                CHGVAR     VAR(&IN30) VALUE('1')
                SNDRCVF    RCDFMT(SFLCTL)
             ENDDO
             ELSE       CMD(DO)
                SNDPGMMSG  MSG('无记录')
             ENDDO

/* 关闭查询文件 */
 CLOSF       CLOF       OPNID(CUSTMAST)
 DLTOVR     FILE(CUSTMAST)

             ENDPGM
```

第18章：消息与错误处理

18.1 消息处理

18.1.1 发送消息

```
/* ============================================================ */
/* 消息处理命令                                                 */
/* ============================================================ */
```

```
/* ------------------------------------------------------------ */
/* SNDPGMMSG - 发送程序消息                                     */
/* ------------------------------------------------------------ */
/* 信息消息 */
             SNDPGMMSG  MSG('处理开始')

/* 带变量 */
             SNDPGMMSG  MSG('处理客户 ' *CAT &CUSTNO)

/* 诊断消息 */
             SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) +
                          MSGDTA('验证失败') MSGTYPE(*DIAG)

/* 转义消息（结束程序） */
             SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) +
                          MSGDTA('严重错误') MSGTYPE(*ESCAPE)

/* 状态消息 */
             SNDPGMMSG  MSG('正在处理...') TOPGMQ(*EXT) +
                          MSGTYPE(*STATUS)

/* 发送给特定用户 */
             SNDPGMMSG  MSG('通知消息') TOMSGQ(QSYSOPR)

/* ------------------------------------------------------------ */
/* SNDUSRMSG - 发送用户消息（需要回复）                         */
/* ------------------------------------------------------------ */
             SNDUSRMSG  MSG('确认删除? (Y/N)') TOMSGQ(&USER) +
                          MSGRPY(&REPLY) RPYLQLEN(1)

             IF         COND(&REPLY *NE 'Y') THEN(RETURN)
```

18.1.2 接收消息

```
/* ============================================================ */
/* 接收消息                                                     */
/* ============================================================ */

             PGM

             DCL        VAR(&MSGID) TYPE(*CHAR) LEN(7)
             DCL        VAR(&MSGDTA) TYPE(*CHAR) LEN(256)
             DCL        VAR(&MSGF) TYPE(*CHAR) LEN(10)
             DCL        VAR(&MSGFLIB) TYPE(*CHAR) LEN(10)
```

```
             DCL        VAR(&RTNTYPE) TYPE(*CHAR) LEN(10)

/* 从程序消息队列接收 */
             RCVMSG     PGMQ(*PRV) RMV(*NO) MSGDTA(&MSGDTA) +
                          MSGID(&MSGID) MSGF(&MSGF) +
                          MSGFLIB(&MSGFLIB) RTNTYPE(&RTNTYPE)

/* 从作业日志接收 */
             RCVMSG     MSGQ(*SYSOPR) WAIT(*MAX) RMV(*NO) +
                          MSG(&MSGDTA) MSGID(&MSGID)

             ENDPGM
```

18.2 错误处理

18.2.1 MONMSG

```
/* ============================================================ */
/* 错误监控 (MONMSG)                                            */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* 特定消息监控                                                 */
/* ------------------------------------------------------------ */
             CALL       PGM(MYPGM)
             MONMSG     MSGID(CPF0000) EXEC(DO)
                SNDPGMMSG  MSG('调用程序失败')
                RETURN
             ENDDO

/* ------------------------------------------------------------ */
/* 通用消息监控                                                 */
/* ------------------------------------------------------------ */
/* 监控特定消息 */
             CHGDTAARA  DTAARA(MYLIB/MYDTAARA) VALUE('TEST')
             MONMSG     MSGID(CPF1015) EXEC(DO)
                /* 数据区不存在 */
                CRTDTAARA  DTAARA(MYLIB/MYDTAARA) TYPE(*CHAR) +
                             LEN(50)
             ENDDO

/* 监控多条消息 */
             DLTF       FILE(MYLIB/TEMPFILE)
             MONMSG     MSGID(CPF2105 CPF2110 CPF2111) EXEC(DO)
```

```
                /* 文件不存在或其他错误 */
                SNDPGMMSG  MSG('无法删除文件')
             ENDDO

/* 监控所有消息 */
             CALL       PGM(UNKNOWN)
             MONMSG     MSGID(CPF0000 MCH0000) EXEC(DO)
                SNDPGMMSG  MSG('发生错误')
                RETURN
             ENDDO

/* ------------------------------------------------------------ */
/* 全局监控（程序级别）                                         */
/* ------------------------------------------------------------ */
             PGM

/* 全局错误处理 */
             MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(ERROR))

/* 正常处理 */
             CALL       PGM(PGM1)
             CALL       PGM(PGM2)
             CALL       PGM(PGM3)

             RETURN

/* 错误处理 */
 ERROR:      SNDPGMMSG  MSG('程序遇到错误')
             SNDPGMMSG  MSG(*DIAG)
             RETURN

             ENDPGM

/* ------------------------------------------------------------ */
/* 执行命令 (EXEC)                                              */
/* ------------------------------------------------------------ */
             DLTF       FILE(MYLIB/TEMPFILE)
             MONMSG     MSGID(CPF2105) EXEC(SNDPGMMSG +
                          MSG('文件不存在'))

/* 执行多个命令 */
             CRTLIB     LIB(NEWLIB)
             MONMSG     MSGID(CPF2111) EXEC(DO)
                SNDPGMMSG  MSG('库已存在，使用现有库')
                CHGVAR     VAR(&USEEXIST) VALUE('1')
             ENDDO)
```

第19章：高级 CL 编程技巧

19.1 作业提交和调度

```
/* ============================================================ */
/* 作业提交 (SBMJOB)                                            */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* 基本提交                                                     */
/* ------------------------------------------------------------ */
             SBMJOB     CMD(CALL PGM(MYLIB/NIGHTLY)) +
                          JOB(NIGHTLY) JOBD(MYLIB/NIGHTJOBD)

/* ------------------------------------------------------------ */
/* 带参数提交                                                   */
/* ------------------------------------------------------------ */
             SBMJOB     CMD(CALL PGM(MYLIB/RPTPGM) PARM('A' +
                          '2024-01-01')) JOB(DAILYRPT) +
                          JOBD(MYLIB/RPTJOBD)

/* ------------------------------------------------------------ */
/* 定时提交                                                     */
/* ------------------------------------------------------------ */
/* 延迟提交 */
             SBMJOB     CMD(CALL PGM(MYLIB/DELAYPGM)) +
                          JOB(DELAYED) JOBD(MYLIB/BATCHJOB) +
                          SCDDATE(*NONE) SCDTIME('14:00:00')

/* 调度作业 */
             ADDJOBSCDE JOB(DAILYBKUP) CMD(CALL +
                          PGM(MYLIB/BACKUPPGM)) FRQ(*WEEKLY) +
                          SCDDATE(*NONE) SCDTIME('02:00:00') +
                          JOBD(MYLIB/BATCHJOBD) +
                          SCDDAY(*ALL)

/* ------------------------------------------------------------ */
/* 作业队列管理                                                 */
/* ------------------------------------------------------------ */
/* 提交到特定队列 */
             SBMJOB     CMD(CALL PGM(MYLIB/LONGJOB)) +
                          JOB(LONGJOB) JOBQ(MYLIB/LONGQ)

/* 高优先级作业 */
```

```
             SBMJOB     CMD(CALL PGM(MYLIB/QUICK)) +
                          JOB(QUICKJOB) JOBD(MYLIB/HIPRIJOBD) +
                          RUNPTY(10)

/* ------------------------------------------------------------ */
/* 作业依赖                                                     */
/* ------------------------------------------------------------ */
/* 等待作业完成 */
             SBMJOB     CMD(CALL PGM(STEP1)) JOB(STEP1)
             DLYJOB     DLY(60)

/* 检查作业状态 */
 CHECK:      RTVJOBA    STSJOB(&JOBSTS)
             IF         COND(&JOBSTS *NE 'OUTQ') THEN(DO)
                DLYJOB     DLY(30)
                GOTO       CMDLBL(CHECK)
             ENDDO

/* 提交下一步 */
             SBMJOB     CMD(CALL PGM(STEP2)) JOB(STEP2)
```

19.2 对象操作

```
/* ============================================================ */
/* 对象操作命令                                                 */
/* ============================================================ */

/* ------------------------------------------------------------ */
/* 保存和恢复                                                   */
/* ------------------------------------------------------------ */
/* 保存库 */
             SAVLIB     LIB(MYLIB) DEV(TAP01)

/* 保存对象 */
             SAVOBJ     OBJ(MYPGM) LIB(MYLIB) OBJTYPE(*PGM) +
                          DEV(TAP01)

/* 恢复到不同库 */
             RSTLIB     SAVLIB(MYLIB) DEV(TAP01) RSTLIB(MYLIB2)

/* ------------------------------------------------------------ */
/* 对象锁管理                                                   */
/* ------------------------------------------------------------ */
/* 检查对象锁 */
```

```
             DSPOBJD    OBJ(MYLIB/MYFILE) OBJTYPE(*FILE) +
                          DETAIL(*LOCK)

/* 结束作业释放锁 */
             ENDJOB     JOB(123456/QUSER/MYJOB) OPTION(*IMMED)

/* ------------------------------------------------------------ */
/* 覆盖文件                                                     */
/* ------------------------------------------------------------ */
/* 临时覆盖文件 */
             OVRDBF     FILE(INPUT) TOFILE(MYLIB/MYDATA) +
                          OVRSCOPE(*CALLLVL)

/* 使用成员 */
             OVRDBF     FILE(INPUT) TOFILE(MYLIB/MYFILE) +
                          MBR(JANUARY)

/* 取消覆盖 */
             DLTOVR     FILE(INPUT)

/* ------------------------------------------------------------ */
/* 数据区操作                                                   */
/* ------------------------------------------------------------ */
/* 创建数据区 */
             CRTDTAARA  DTAARA(MYLIB/CONFIG) TYPE(*CHAR) +
                          LEN(100) VALUE('DEFAULT')

/* 读取 */
             RTVDTAARA  DTAARA(CONFIG) RTNVAL(&VALUE)

/* 更改 */
             CHGDTAARA  DTAARA(CONFIG) VALUE('NEWVALUE')

/* 读取子串 */
             RTVDTAARA  DTAARA(CONFIG (1 10)) RTNVAL(&SUB)
```

第五部分：高级主题与实战案例

第20章：DB2 for i 数据库编程

20.1 SQL 编程进阶

20.1.1 高级查询技巧

```
// ============================================================
// 高级 SQL 查询
// ============================================================

// ------------------------------------------------------------
// 1. 公用表表达式 (CTE)
// ------------------------------------------------------------
Exec SQL
    WITH
    SalesSummary AS (
        SELECT
            CUST_NO,
            SUM(AMOUNT) AS TOTAL_SALES,
            COUNT(*) AS ORDER_COUNT,
            AVG(AMOUNT) AS AVG_ORDER
        FROM ORDERS
        WHERE ORDER_DATE >= :StartDate
        GROUP BY CUST_NO
    ),
    TopCustomers AS (
        SELECT CUST_NO, TOTAL_SALES
        FROM SalesSummary
        WHERE TOTAL_SALES > 10000
    )
    SELECT
        C.CUST_NAME,
        S.TOTAL_SALES,
        S.ORDER_COUNT,
        S.AVG_ORDER
    FROM TopCustomers T
    JOIN SalesSummary S ON T.CUST_NO = S.CUST_NO
    JOIN CUSTOMER C ON S.CUST_NO = C.CUST_NO
    ORDER BY S.TOTAL_SALES DESC;

// ------------------------------------------------------------
```

```
// 2. 窗口函数
// ------------------------------------------------------------
Exec SQL
    SELECT
        CUST_NO,
        ORDER_NO,
        AMOUNT,
        SUM(AMOUNT) OVER (PARTITION BY CUST_NO) AS CUST_TOTAL,
        SUM(AMOUNT) OVER (ORDER BY ORDER_DATE) AS RUNNING_TOTAL,
        ROW_NUMBER() OVER (PARTITION BY CUST_NO ORDER BY AMOUNT DESC)
AS RANK_IN_CUST,
        LAG(AMOUNT, 1) OVER (ORDER BY ORDER_DATE) AS PREV_AMOUNT,
        LEAD(AMOUNT, 1) OVER (ORDER BY ORDER_DATE) AS NEXT_AMOUNT
    FROM ORDERS
    WHERE ORDER_DATE >= :StartDate;

// ------------------------------------------------------------
// 3. 递归查询
// ------------------------------------------------------------
Exec SQL
    WITH RECURSIVE EmployeeHierarchy AS (
        -- 锚定成员：顶级经理
        SELECT
            EMP_ID, EMP_NAME, MANAGER_ID, 1 AS LEVEL
        FROM EMPLOYEES
        WHERE MANAGER_ID IS NULL

        UNION ALL

        -- 递归成员：下属员工
        SELECT
            E.EMP_ID, E.EMP_NAME, E.MANAGER_ID, H.LEVEL + 1
        FROM EMPLOYEES E
        JOIN EmployeeHierarchy H ON E.MANAGER_ID = H.EMP_ID
    )
    SELECT * FROM EmployeeHierarchy;

// ------------------------------------------------------------
// 4. 透视查询
// ------------------------------------------------------------
Exec SQL
    SELECT
        CUST_NO,
        SUM(CASE WHEN MONTH(ORDER_DATE) = 1 THEN AMOUNT ELSE 0 END) AS
JAN,
        SUM(CASE WHEN MONTH(ORDER_DATE) = 2 THEN AMOUNT ELSE 0 END) AS
```

```
FEB,
        SUM(CASE WHEN MONTH(ORDER_DATE) = 3 THEN AMOUNT ELSE 0 END) AS
MAR,
        -- ... 其他月份
        SUM(AMOUNT) AS TOTAL
    FROM ORDERS
    WHERE YEAR(ORDER_DATE) = 2024
    GROUP BY CUST_NO;
```

20.1.2 存储过程和触发器

```
-- ============================================================
-- 存储过程
-- ============================================================

-- 创建存储过程
CREATE OR REPLACE PROCEDURE PROCESS_ORDER (
    IN P_ORDER_NO DECIMAL(10, 0),
    IN P_USER_ID CHAR(10),
    OUT P_RESULT CHAR(1),
    OUT P_MESSAGE VARCHAR(100)
)
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
    DECLARE V_CUST_NO DECIMAL(7, 0);
    DECLARE V_ORDER_AMT DECIMAL(15, 2);
    DECLARE V_CUST_BAL DECIMAL(15, 2);

    -- 初始化输出参数
    SET P_RESULT = '0';
    SET P_MESSAGE = '';

    -- 获取订单信息
    SELECT CUST_NO, TOTAL_AMOUNT
    INTO V_CUST_NO, V_ORDER_AMT
    FROM ORDERS
    WHERE ORDER_NO = P_ORDER_NO;

    IF SQLCODE <> 0 THEN
        SET P_RESULT = 'E';
        SET P_MESSAGE = 'Order not found';
        RETURN;
    END IF;
```

```
    -- 检查客户余额
    SELECT CURRENT_BALANCE
    INTO V_CUST_BAL
    FROM CUSTOMER
    WHERE CUST_NO = V_CUST_NO;

    IF V_CUST_BAL + V_ORDER_AMT > V_CREDIT_LIMIT THEN
        SET P_RESULT = 'R';
        SET P_MESSAGE = 'Credit limit exceeded';
        RETURN;
    END IF;

    -- 更新客户余额
    UPDATE CUSTOMER
    SET CURRENT_BALANCE = CURRENT_BALANCE + V_ORDER_AMT,
        LAST_UPDATE = CURRENT_TIMESTAMP,
        UPDATED_BY = P_USER_ID
    WHERE CUST_NO = V_CUST_NO;

    -- 更新订单状态
    UPDATE ORDERS
    SET STATUS = 'P',
        PROCESS_DATE = CURRENT_TIMESTAMP
    WHERE ORDER_NO = P_ORDER_NO;

    SET P_RESULT = 'S';
    SET P_MESSAGE = 'Order processed successfully';
END;

-- ============================================================
-- 触发器
-- ============================================================

-- 审计日志触发器
CREATE OR REPLACE TRIGGER CUSTOMER_AUDIT
AFTER UPDATE ON CUSTOMER
REFERENCING OLD AS O NEW AS N
FOR EACH ROW
MODE DB2SQL
BEGIN
    INSERT INTO CUSTOMER_AUDIT (
        AUDIT_TS, AUDIT_TYPE, CUST_NO,
        OLD_STATUS, NEW_STATUS,
        OLD_BALANCE, NEW_BALANCE,
        CHANGED_BY
    ) VALUES (
```

```
        CURRENT_TIMESTAMP, 'UPDATE', N.CUST_NO,
        O.STATUS, N.STATUS,
        O.CURRENT_BALANCE, N.CURRENT_BALANCE,
        N.UPDATED_BY
    );
END;

-- 自动更新触发器
CREATE OR REPLACE TRIGGER ORDERS_UPDATE_TIMESTAMP
BEFORE UPDATE ON ORDERS
REFERENCING NEW AS N
FOR EACH ROW
MODE DB2SQL
BEGIN
    SET N.LAST_UPDATE = CURRENT_TIMESTAMP;
END;
```

第21章：Web 服务与现代化开发

21.1 Web 服务开发

21.1.1 REST API 开发

```
// ============================================================
// REST API 开发示例
// ============================================================

// 使用 IWS (Integrated Web Services) 或手动处理

// ------------------------------------------------------------
// HTTP 服务器配置
// ------------------------------------------------------------

// 1. 启动 HTTP 服务器
// STRTCPSVR SERVER(*HTTP) HTTPSVR(MYAPP)

// 2. 配置 HTTP 服务器（HTTPADMIN 界面）

// ------------------------------------------------------------
// REST 处理程序示例
// ------------------------------------------------------------
Ctl-Opt BndDir('QC2LE');
```

```
// CGI 处理程序
Dcl-Proc GetCustomer Export;
    Dcl-Pi *N;
        Request Varchar(1024) Const;
        Response Varchar(32767);
    End-Pi;

    Dcl-S CustNo Int(10);
    Dcl-S JsonData Varchar(1000);

    // 解析请求参数
    CustNo = ParseQueryParm(Request:'custno');

    // 查询数据库
    Exec SQL
        SELECT JSON_OBJECT(
            'custno' VALUE CUST_NO,
            'name' VALUE CUST_NAME,
            'status' VALUE STATUS,
            'balance' VALUE CURRENT_BALANCE
        )
        INTO :JsonData
        FROM CUSTOMER
        WHERE CUST_NO = :CustNo;

    If SQLCODE = 0;
        Response = '{"success":true,"data":' + JsonData + '}';
    ElseIf SQLCODE = 100;
        Response = '{"success":false,"error":"Customer not found"}';
    Else;
        Response = '{"success":false,"error":"Database error"}';
    EndIf;

End-Proc;

// ------------------------------------------------------------
// 使用 SQL 发布 REST 服务
// ------------------------------------------------------------
// 创建 REST 服务
Exec SQL
    CREATE OR REPLACE PROCEDURE API_GET_CUSTOMER (
        IN CUSTNO_IN INTEGER,
        OUT JSON_OUT CLOB(10K)
    )
    LANGUAGE SQL
    READS SQL DATA
```

```
    BEGIN
        SELECT JSON_OBJECT(
            'customer' VALUE JSON_OBJECT(
                'number' VALUE CUST_NO,
                'name' VALUE CUST_NAME,
                'address' VALUE JSON_OBJECT(
                    'street' VALUE ADDRESS,
                    'city' VALUE CITY,
                    'state' VALUE STATE,
                    'zip' VALUE ZIP
                )
            )
        )
        INTO JSON_OUT
        FROM CUSTOMER
        WHERE CUST_NO = CUSTNO_IN;
    END;
```

21.1.2 JSON 处理

```
// ============================================================
// JSON 处理
// ============================================================

// ------------------------------------------------------------
// SQL JSON 函数
// ------------------------------------------------------------

// 1. 创建 JSON
Exec SQL
    SELECT JSON_OBJECT(
        'order' VALUE JSON_OBJECT(
            'number' VALUE ORDER_NO,
            'date' VALUE ORDER_DATE,
            'customer' VALUE JSON_OBJECT(
                'name' VALUE CUST_NAME,
                'email' VALUE EMAIL
            ),
            'items' VALUE JSON_ARRAYAGG(
                JSON_OBJECT(
                    'sku' VALUE SKU,
                    'description' VALUE DESCRIPTION,
                    'quantity' VALUE QTY,
                    'price' VALUE PRICE
                )
```

```
            )
        )
    )
    INTO :JsonDoc
    FROM ORDERS O
    JOIN CUSTOMER C ON O.CUST_NO = C.CUST_NO
    JOIN ORDER_ITEMS I ON O.ORDER_NO = I.ORDER_NO
    WHERE O.ORDER_NO = :OrderNo
    GROUP BY O.ORDER_NO, ORDER_DATE, CUST_NAME, EMAIL;

// 2. 解析JSON
Exec SQL
    SELECT
        JSON_VALUE(:JsonDoc, '$.order.number') AS OrderNo,
        JSON_VALUE(:JsonDoc, '$.order.date') AS OrderDate,
        JSON_VALUE(:JsonDoc, '$.order.customer.name') AS CustName
    INTO :OrderNo, :OrderDate, :CustName
    FROM SYSIBM.SYSDUMMY1;

// 3. 查询JSON 数组
Exec SQL
    SELECT
        JSON_VALUE(JT.*, '$.sku') AS SKU,
        JSON_VALUE(JT.*, '$.quantity') AS Qty
    FROM JSON_TABLE(:JsonDoc, '$.order.items[*]'
        COLUMNS (
            SKU VARCHAR(20) PATH '$.sku',
            QTY INT PATH '$.quantity'
        )
    ) AS JT;

// ------------------------------------------------------------
// RPG 中处理JSON
// ------------------------------------------------------------
Dcl-Pr yajl_genOpen Pointer ExtProc('yajl_genOpen');
End-Pr;

Dcl-Pr yajl_genClose ExtProc('yajl_genClose');
    Gen Pointer Value;
End-Pr;

Dcl-Pr yajl_genAddCharString ExtProc('yajl_genAddCharString');
    Gen Pointer Value;
    String Pointer Value Options(*String);
End-Pr;
```

```
// 生成 JSON
Dcl-S Gen Pointer;
Dcl-S JsonString Varchar(1000);

Gen = yajl_genOpen();

// 开始对象
yajl_genOpenObject(Gen);

// 添加字段
yajl_genAddCharString(Gen:'custno');
yajl_genAddInt(Gen:CustNo);

yajl_genAddCharString(Gen:'name');
yajl_genAddCharString(Gen:%Trim(CustName));

// 结束对象
yajl_genCloseObject(Gen);

// 获取结果
JsonString = yajl_genGetBuf(Gen);

// 清理
yajl_genClose(Gen);
```

第22章：系统集成与 API

22.1 调用外部 API

22.1.1 HTTP 客户端

```
// ============================================================
// HTTP 客户端编程
// ============================================================

Ctl-Opt BndDir('QC2LE');

// ------------------------------------------------------------
// 使用 HTTPAPI
// ------------------------------------------------------------

// 定义 HTTP 函数原型
Dcl-Pr http_get Int(10) ExtProc('http_get');
```

```
    Url Pointer Value Options(*String);
    File Pointer Value Options(*String);
End-Pr;

Dcl-Pr http_post Int(10) ExtProc('http_post');
    Url Pointer Value Options(*String);
    File Pointer Value Options(*String);
    ContentType Pointer Value Options(*String);
End-Pr;

// 调用 REST API
Dcl-S HttpStatus Int(10);
Dcl-S ResponseFile Char(256);
Dcl-S ApiUrl Varchar(500);

ApiUrl = 'https://api.example.com/customers/' + %Char(CustNo);
ResponseFile = '/tmp/response.json';

HttpStatus = http_get(ApiUrl:ResponseFile);

If HttpStatus = 200;
    // 读取响应文件
    // 解析 JSON...
Else;
    // 错误处理
EndIf;

// ------------------------------------------------------------
// 使用 Python 集成
// ------------------------------------------------------------
Dcl-Pr system Int(10) ExtProc('system');
    Command Pointer Value Options(*String);
End-Pr;

// 调用 Python 脚本
Dcl-S PyCommand Varchar(1000);
PyCommand = 'python3 /scripts/call_api.py ' +
            %Char(CustNo) + ' > /tmp/result.json';
system(PyCommand);
```

22.1.2 FTP/SFTP 传输

```
/* ============================================================ */
/* FTP/SFTP 文件传输                                            */
```

```
/* ============================================================ */

/* ------------------------------------------------------------ */
/* FTP 客户端                                                   */
/* ------------------------------------------------------------ */
             OVRDBF     INPUT FTPINPUT
             OVRDBF     OUTPUT FTPOUTPUT

/* FTP 输入文件内容：
   open ftpserver.example.com
   user myuser mypassword
   cd /remote/dir
   lcd /local/dir
   put localfile.txt remotefile.txt
   quit
*/
             CALL       PGM(QTMFFTP)

/* ------------------------------------------------------------ */
/* SFTP (使用 PASE)                                             */
/* ------------------------------------------------------------ */
/* 创建 SFTP 脚本文件 */
             ADDENVVAR  ENVVAR(QIBM_MULTI_THREADED) VALUE(Y) +
                          REPLACE(*YES)

/* 使用 QP2TERM 或 QSH */
             QSH        CMD('sftp -b /home/user/sftp_batch.txt +
                          user@server.example.com')

/* SFTP 批处理文件内容：
   cd /remote/directory
   put /local/file.csv
   quit
*/
```

第23章：性能调优与最佳实践

23.1 性能优化技巧

23.1.1 RPG 性能优化

```
// ============================================================
// RPG 性能优化指南
```

```
// ============================================================

// ------------------------------------------------------------
// 1. 文件访问优化
// ------------------------------------------------------------

// 好的做法：使用 KLIST 或 %KDS 进行键访问
Chain %Kds(KeyStruct) CUSTMAST;

// 避免：在循环中重复计算键值
For I = 1 To NumKeys;
    // 不好的做法：每次循环都计算
    Chain GetKey(I) CUSTMAST;
EndFor;

// 好的做法：预计算键值
Dcl-S Keys Dim(1000) Like(CustKey);
// ... 填充 Keys 数组
For I = 1 To NumKeys;
    Chain Keys(I) CUSTMAST;
EndFor;

// ------------------------------------------------------------
// 2. 数组操作优化
// ------------------------------------------------------------
// 使用 %LOOKUP 而非线性搜索
Pos = %Lookup(SearchValue:Array);

// 使用 %LOOKUPLE 进行范围查找
Pos = %Lookuplt(SearchValue:SortedArray);

// ------------------------------------------------------------
// 3. 字符串操作优化
// ------------------------------------------------------------
// 使用 %LEN 获取实际长度，避免 %TRIM 的复制开销
If %Len(%TrimR(String)) > 0;

// 避免频繁的字符串连接
// 不好的做法：
Result = '';
For I = 1 To Count;
    Result = Result + Items(I);
EndFor;

// 好的做法：使用数组和 %SUBST
Result = %Subst(Items:1:TotalLen);
```

```
// ------------------------------------------------------------
// 4. 循环优化
// ------------------------------------------------------------
// 使用 FOR 而非 DOW 进行已知次数的循环
For I = 1 To %Elem(Array);
    Process(Array(I));
EndFor;

// 减少循环内的复杂计算
// 不好的做法：
For I = 1 To Count;
    If ComplexCalculation() > Threshold;
        // ...
    EndIf;
EndFor;

// 好的做法：
CalculatedThreshold = ComplexCalculation();
For I = 1 To Count;
    If CalculatedThreshold > Threshold;
        // ...
    EndIf;
EndFor;

// ------------------------------------------------------------
// 5. 内存管理
// ------------------------------------------------------------
// 使用 STATIC 减少内存分配
Dcl-S Counter Int(10) Static;

// 及时释放基于指针的内存
Dealloc(N) MyPtr;
```

23.1.2 SQL 性能优化

```
-- ============================================================
-- SQL 性能优化
-- ============================================================

-- 1. 使用适当的索引
CREATE INDEX IDX_ORDERS_CUST_DATE ON ORDERS (CUST_NO, ORDER_DATE);

-- 2. 避免在 WHERE 子句中使用函数
```

```
-- 不好的做法：
SELECT * FROM ORDERS WHERE YEAR(ORDER_DATE) = 2024;

-- 好的做法：
SELECT * FROM ORDERS WHERE ORDER_DATE BETWEEN '2024-01-01' AND
'2024-12-31';

-- 3. 使用 EXISTS 而非 IN 进行子查询
-- 不好的做法：
SELECT * FROM CUSTOMER WHERE CUST_NO IN (SELECT CUST_NO FROM ORDERS);

-- 好的做法：
SELECT * FROM CUSTOMER C WHERE EXISTS (SELECT 1 FROM ORDERS O WHERE
O.CUST_NO = C.CUST_NO);

-- 4. 限制返回的行数
SELECT * FROM ORDERS ORDER BY ORDER_DATE DESC FETCH FIRST 100 ROWS
ONLY;

-- 5. 使用覆盖索引
-- 创建包含所有查询字段的索引
CREATE INDEX IDX_ORDERS_COVER ON ORDERS (CUST_NO) INCLUDE (ORDER_NO,
AMOUNT, STATUS);

-- 6. 避免 SELECT *
-- 不好的做法：
SELECT * FROM CUSTOMER WHERE CUST_NO = :CustNo;

-- 好的做法：
SELECT CUST_NAME, STATUS, BALANCE FROM CUSTOMER WHERE CUST_NO
= :CustNo;
```

23.2 调试技巧

23.2.1 调试工具

```
// ============================================================
// 调试技巧
// ============================================================

// ------------------------------------------------------------
// 1. 源语句选项
// ------------------------------------------------------------
Ctl-Opt Option(*SrcStmt:*NoDebugIo);
```

```
// *SrcStmt  - 保留源代码行号信息
// *NoDebugIo - 不调试 I/O 操作

// ------------------------------------------------------------
// 2. 断点设置
// ------------------------------------------------------------
// 使用 STRDBG 命令启动调试器
// STRDBG PGM(MYPGM) UPDPROD(*NO)

// 在代码中设置断点（使用操作码或自由格式）
// 固定格式：
// C           1         DOWEQ1
// C     BREAK     TAG

// 自由格式：
If DebugMode;
    // 调试断点位置
    Dsply 'Breakpoint' ' ';
EndIf;

// ------------------------------------------------------------
// 3. 日志记录
// ------------------------------------------------------------
Dcl-Pr WriteJobLog ExtProc('Qp0zLprintf');
    Format Pointer Value Options(*String);
End-Pr;

// 写入作业日志
WriteJobLog('Debug: CustNo = %d, Status = %s' :
            %Char(CustNo) : Status);

// ------------------------------------------------------------
// 4. DUMP 操作码
// ------------------------------------------------------------
// 输出所有变量值
DUMP;

// 带标签的 DUMP
DUMP(A) 'Before Update';
// ... 更新操作
DUMP(A) 'After Update';
```

第24章：完整项目实战：ERP 订单管理系统

24.1 项目概述

```
项目名称：ERP 订单管理系统
目标：演示完整的 IBM i 开发流程
技术栈：
  - RPG IV ILE
  - SQL
  - CLLE
  - 显示文件
  - 打印文件
  - Web 服务接口

功能模块：
  1. 客户管理
  2. 产品管理
  3. 订单录入
  4. 订单查询
  5. 订单报表
  6. API 接口
```

24.2 数据库设计

```
-- ============================================================
-- 数据库定义
-- ============================================================

-- 客户主文件
CREATE TABLE CUSTOMER (
    CUST_NO DECIMAL(7, 0) NOT NULL PRIMARY KEY,
    CUST_NAME VARCHAR(50) NOT NULL,
    ADDRESS VARCHAR(100),
    CITY VARCHAR(30),
    STATE CHAR(2),
    ZIP CHAR(10),
    PHONE CHAR(15),
    EMAIL VARCHAR(100),
    CREDIT_LIMIT DECIMAL(15, 2) DEFAULT 0,
    CURRENT_BALANCE DECIMAL(15, 2) DEFAULT 0,
    STATUS CHAR(1) DEFAULT 'A',
```

```
    CREATED_TS TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UPDATED_TS TIMESTAMP,
    UPDATED_BY CHAR(10)
);

-- 产品主文件
CREATE TABLE PRODUCT (
    SKU CHAR(20) NOT NULL PRIMARY KEY,
    DESCRIPTION VARCHAR(100) NOT NULL,
    CATEGORY CHAR(10),
    UNIT_PRICE DECIMAL(15, 4) NOT NULL,
    COST_PRICE DECIMAL(15, 4),
    QTY_ON_HAND DECIMAL(10, 0) DEFAULT 0,
    STATUS CHAR(1) DEFAULT 'A',
    CREATED_TS TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 订单主文件
CREATE TABLE ORDERS (
    ORDER_NO DECIMAL(10, 0) NOT NULL PRIMARY KEY,
    ORDER_DATE DATE DEFAULT CURRENT_DATE,
    CUST_NO DECIMAL(7, 0) NOT NULL,
    ORDER_STATUS CHAR(1) DEFAULT 'O', -- O=Open, P=Processed,
C=Complete, X=Cancel
    TOTAL_AMOUNT DECIMAL(15, 2) DEFAULT 0,
    TAX_AMOUNT DECIMAL(15, 2) DEFAULT 0,
    SHIP_AMOUNT DECIMAL(15, 2) DEFAULT 0,
    DISCOUNT_PCT DECIMAL(3, 2) DEFAULT 0,
    NOTES VARCHAR(500),
    CREATED_BY CHAR(10),
    CREATED_TS TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PROCESS_TS TIMESTAMP,
    FOREIGN KEY (CUST_NO) REFERENCES CUSTOMER(CUST_NO)
);

-- 订单明细文件
CREATE TABLE ORDER_ITEM (
    ORDER_NO DECIMAL(10, 0) NOT NULL,
    LINE_NO DECIMAL(5, 0) NOT NULL,
    SKU CHAR(20) NOT NULL,
    DESCRIPTION VARCHAR(100),
    QTY DECIMAL(7, 0) NOT NULL,
    UNIT_PRICE DECIMAL(15, 4) NOT NULL,
    DISCOUNT_PCT DECIMAL(3, 2) DEFAULT 0,
    LINE_TOTAL DECIMAL(15, 2) GENERATED ALWAYS AS
        (QTY * UNIT_PRICE * (1 - DISCOUNT_PCT)),
```

```
    PRIMARY KEY (ORDER_NO, LINE_NO),
    FOREIGN KEY (ORDER_NO) REFERENCES ORDERS(ORDER_NO),
    FOREIGN KEY (SKU) REFERENCES PRODUCT(SKU)
);
```

24.3 源代码示例

24.3.1 订单录入程序

```
// ============================================================
// ORDENTRY - 订单录入程序
// ============================================================

**FREE
Ctl-Opt DftActGrp(*No) ActGrp(*NEW)
        Option(*SrcStmt:*NoDebugIo);

// 文件声明
Dcl-F CUSTOMER Disk(*Ext) Usage(*Input) Keyed;
Dcl-F PRODUCT Disk(*Ext) Usage(*Input) Keyed;
Dcl-F ORDERS Disk(*Ext) Usage(*Output) Keyed;
Dcl-F ORDER_ITEM Disk(*Ext) Usage(*Output) Keyed;
Dcl-F ORDENTRYD WorkStn IndDs(DspInds);

// 数据结构
Dcl-DS DspInds Len(99);
    Exit Ind Pos(3);
    Prompt Ind Pos(4);
    Refresh Ind Pos(5);
    AddItem Ind Pos(6);
    DeleteItem Ind Pos(7);
    SaveOrder Ind Pos(10);
    Cancel Ind Pos(12);
    CancelEnable Ind Pos(31);
    ErrorMsg Ind Pos(30);
End-DS;

Dcl-DS OrderHdrDS Qualified;
    OrderNo Packed(10:0);
    OrderDate Date;
    CustNo Packed(7:0);
    CustName Char(50);
    Total Packed(15:2);
    Status Char(1);
```

```
End-DS;

Dcl-DS ItemDS Dim(20) Qualified;
    LineNo Packed(5:0);
    Sku Char(20);
    Desc Char(50);
    Qty Packed(7:0);
    Price Packed(15:4);
    LineTotal Packed(15:2);
End-DS;

// 变量
Dcl-S LineCount Int(5) Inz(0);
Dcl-S CurrentLine Int(5) Inz(0);
Dcl-S ErrorText Char(50);

// ============================================================
// 主程序
// ============================================================

// 初始化
Clear ORDHEAD;
Clear ORDITEM;
LineCount = 0;
CurrentLine = 0;
OrderHdrDS.OrderNo = GetNextOrderNo();
OrderHdrDS.OrderDate = %Date();

// 主循环
Dow Not Exit;
    ExFmt ORDHEAD;

    Select;
        When Exit;
            Leave;
        When Refresh;
            ExSr ClearScreen;
        When Prompt And %CursorFld = 'CUSTNO';
            ExSr PromptCustomer;
        When Prompt And %CursorFld = 'SKU';
            ExSr PromptProduct;
        When AddItem;
            ExSr AddLineItem;
        When DeleteItem;
            ExSr DeleteLineItem;
        When SaveOrder;
```

```
            ExSr ValidateAndSave;
            If Not ErrorMsg;
                ExSr ClearScreen;
            EndIf;
    EndSl;
EndDo;

*InLr = *On;
Return;

// ============================================================
// 子程序
// ============================================================

BegSr ClearScreen;
    Clear ORDHEAD;
    Clear ORDITEM;
    LineCount = 0;
    CurrentLine = 0;
    OrderHdrDS.OrderNo = GetNextOrderNo();
    OrderHdrDS.OrderDate = %Date();
EndSr;

BegSr PromptCustomer;
    // 调用客户查询程序
    CallP CustLookup(OrderHdrDS.CustNo:OrderHdrDS.CustName);
    If OrderHdrDS.CustNo > 0;
        Chain OrderHdrDS.CustNo CUSTOMER;
        If %Found;
            OrderHdrDS.CustName = CUST_NAME;
        EndIf;
    EndIf;
EndSr;

BegSr PromptProduct;
    Dcl-S TempSku Char(20);
    Dcl-S TempDesc Char(50);

    CallP ProductLookup(TempSku:TempDesc);
    If TempSku <> '';
        SKU = TempSku;
        // 获取产品信息
        Chain TempSku PRODUCT;
        If %Found;
            ITEMDESC = DESCRIPTION;
            UNITPRICE = UNIT_PRICE;
```

```
        EndIf;
    EndIf;
EndSr;

BegSr AddLineItem;
    // 验证输入
    If SKU = '';
        ErrorText = '请输入产品编号';
        ErrorMsg = *On;
        Return;
    EndIf;

    If QTY <= 0;
        ErrorText = '数量必须大于0';
        ErrorMsg = *On;
        Return;
    EndIf;

    // 检查库存
    Chain SKU PRODUCT;
    If Not %Found;
        ErrorText = '产品不存在';
        ErrorMsg = *On;
        Return;
    EndIf;

    If QTY > QTY_ON_HAND;
        ErrorText = '库存不足';
        ErrorMsg = *On;
        Return;
    EndIf;

    // 添加到数组
    LineCount += 1;
    CurrentLine = LineCount;
    ItemDS(CurrentLine).LineNo = CurrentLine;
    ItemDS(CurrentLine).Sku = SKU;
    ItemDS(CurrentLine).Desc = DESCRIPTION;
    ItemDS(CurrentLine).Qty = QTY;
    ItemDS(CurrentLine).Price = UNIT_PRICE;
    ItemDS(CurrentLine).LineTotal = QTY * UNIT_PRICE;

    // 更新订单合计
    OrderHdrDS.Total += ItemDS(CurrentLine).LineTotal;

    // 清空输入字段
```

```
    SKU = '';
    ITEMDESC = '';
    QTY = 0;
    UNITPRICE = 0;

    ErrorMsg = *Off;
EndSr;

BegSr DeleteLineItem;
    // 实现删除逻辑
    If CurrentLine > 0 And CurrentLine <= LineCount;
        // 从合计中减去
        OrderHdrDS.Total -= ItemDS(CurrentLine).LineTotal;

        // 移动后续项目
        For I = CurrentLine To LineCount - 1;
            ItemDS(I) = ItemDS(I + 1);
            ItemDS(I).LineNo = I;
        EndFor;

        LineCount -= 1;
        If CurrentLine > LineCount;
            CurrentLine = LineCount;
        EndIf;
    EndIf;
EndSr;

BegSr ValidateAndSave;
    // 验证订单头
    If OrderHdrDS.CustNo = 0;
        ErrorText = '请选择客户';
        ErrorMsg = *On;
        Return;
    EndIf;

    If LineCount = 0;
        ErrorText = '请添加订单明细';
        ErrorMsg = *On;
        Return;
    EndIf;

    // 开始事务
    Exec SQL SET OPTION COMMIT = *CHG;

    // 保存订单头
    OrderNo = OrderHdrDS.OrderNo;
```

```
    OrderDate = OrderHdrDS.OrderDate;
    CustNo = OrderHdrDS.CustNo;
    OrderStatus = 'O';
    TotalAmount = OrderHdrDS.Total;
    CreatedBy = %User();

    Write ORDREC;

    // 保存订单明细
    For I = 1 To LineCount;
        OrderNo = OrderHdrDS.OrderNo;
        LineNo = ItemDS(I).LineNo;
        SKU = ItemDS(I).Sku;
        Description = ItemDS(I).Desc;
        Qty = ItemDS(I).Qty;
        UnitPrice = ItemDS(I).Price;
        DiscountPct = 0;

        Write ITEMREC;

        // 更新库存
        Exec SQL
            UPDATE PRODUCT
            SET QTY_ON_HAND = QTY_ON_HAND - :Qty
            WHERE SKU = :SKU;
    EndFor;

    // 提交事务
    Exec SQL COMMIT;

    ErrorMsg = *Off;
    SndMsg('订单 ' + %Char(OrderHdrDS.OrderNo) + ' 已保存');
EndSr;

// ============================================================
// 过程定义
// ============================================================

Dcl-Proc GetNextOrderNo Int(10);
    Dcl-Pi *N Int(10);
    End-Pi;

    Dcl-S NextNo Int(10);

    Exec SQL
        SELECT COALESCE(MAX(ORDER_NO), 0) + 1
```

```
        INTO :NextNo
        FROM ORDERS;

    Return NextNo;
End-Proc;

Dcl-Proc SndMsg;
    Dcl-Pi *N;
        MsgText Char(100) Const;
    End-Pi;

    SndPgmMsg Msg(MsgText) ToPgmQ(*Ext);
End-Proc;

Dcl-Proc CustLookup Export;
    Dcl-Pi *N;
        CustNo Packed(7:0);
        CustName Char(50);
    End-Pi;
    // 实现客户查找窗口...
End-Proc;

Dcl-Proc ProductLookup Export;
    Dcl-Pi *N;
        Sku Char(20);
        Desc Char(50);
    End-Pi;
    // 实现产品查找窗口...
End-Proc;
```

24.3.2 订单处理服务程序

```
// ============================================================
// ORDSRV - 订单服务程序
// ============================================================

**FREE
Ctl-Opt NoMain;
Ctl-Opt Export(*All);

// ============================================================
// 过程：获取订单信息
// ============================================================
Dcl-Proc GetOrderInfo Export;
    Dcl-Pi *N Ind;
```

```
        OrderNo Packed(10:0) Const;
        OrderInfo LikeDS(OrderDS);
    End-Pi;

    Dcl-DS OrderDS Qualified Template;
        OrderNo Packed(10:0);
        OrderDate Date;
        CustNo Packed(7:0);
        CustName Char(50);
        Status Char(1);
        Total Packed(15:2);
    End-DS;

    Exec SQL
        SELECT
            O.ORDER_NO, O.ORDER_DATE, O.CUST_NO,
            C.CUST_NAME, O.ORDER_STATUS, O.TOTAL_AMOUNT
        INTO
            :OrderInfo.OrderNo, :OrderInfo.OrderDate,
            :OrderInfo.CustNo, :OrderInfo.CustName,
            :OrderInfo.Status, :OrderInfo.Total
        FROM ORDERS O
        JOIN CUSTOMER C ON O.CUST_NO = C.CUST_NO
        WHERE O.ORDER_NO = :OrderNo;

    Return (SQLCODE = 0);
End-Proc;

// ============================================================
// 过程：更新订单状态
// ============================================================
Dcl-Proc UpdateOrderStatus Export;
    Dcl-Pi *N Ind;
        OrderNo Packed(10:0) Const;
        NewStatus Char(1) Const;
        UserId Char(10) Const;
    End-Pi;

    Exec SQL
        UPDATE ORDERS
        SET ORDER_STATUS = :NewStatus,
            PROCESS_TS = CURRENT_TIMESTAMP,
            CREATED_BY = :UserId
        WHERE ORDER_NO = :OrderNo;

    Return (SQLCODE = 0);
```

```
End-Proc;

// ============================================================
// 过程：获取客户订单列表
// ============================================================
Dcl-Proc GetCustomerOrders Export;
    Dcl-Pi *N;
        CustNo Packed(7:0) Const;
        Orders LikeDS(OrderListDS) Dim(100);
        OrderCount Int(10);
    End-Pi;

    Dcl-DS OrderListDS Qualified Template;
        OrderNo Packed(10:0);
        OrderDate Date;
        Total Packed(15:2);
        Status Char(1);
    End-DS;

    Dcl-S i Int(10) Inz(0);

    OrderCount = 0;

    Exec SQL
        DECLARE OrderCursor CURSOR FOR
        SELECT ORDER_NO, ORDER_DATE, TOTAL_AMOUNT, ORDER_STATUS
        FROM ORDERS
        WHERE CUST_NO = :CustNo
        ORDER BY ORDER_DATE DESC;

    Exec SQL OPEN OrderCursor;

    DoW 1 = 1;
        i += 1;
        If i > %Elem(Orders);
            Leave;
        EndIf;

        Exec SQL
            FETCH OrderCursor INTO
                :Orders(i).OrderNo, :Orders(i).OrderDate,
                :Orders(i).Total, :Orders(i).Status;

        If SQLCODE <> 0;
            Leave;
        EndIf;
```

```
        OrderCount = i;
    EndDo;

    Exec SQL CLOSE OrderCursor;
End-Proc;

// ============================================================
// 过程：计算订单合计
// ============================================================
Dcl-Proc CalculateOrderTotal Export;
    Dcl-Pi *N Packed(15:2);
        OrderNo Packed(10:0) Const;
    End-Pi;

    Dcl-S Total Packed(15:2);

    Exec SQL
        SELECT SUM(LINE_TOTAL)
        INTO :Total
        FROM ORDER_ITEM
        WHERE ORDER_NO = :OrderNo;

    If SQLCODE = 0 And Total <> *Zero;
        Return Total;
    Else;
        Return 0;
    EndIf;
End-Proc;
```

24.4 部署和维护

```
/* ============================================================ */
/* 部署脚本                                                     */
/* ============================================================ */

             PGM

/* 创建库 */
             CRTLIB     LIB(ORDERSYS) TEXT('Order Management System')
             MONMSG     MSGID(CPF2111) EXEC(DO)
                SNDPGMMSG  MSG('库已存在')
             ENDDO
```

```
/* 创建源文件 */
             CRTSRCPF   FILE(ORDERSYS/QRPGLESRC) RCDLEN(112)
             CRTSRCPF   FILE(ORDERSYS/QCLLESRC) RCDLEN(92)
             CRTSRCPF   FILE(ORDERSYS/QDDSSRC) RCDLEN(92)
             CRTSRCPF   FILE(ORDERSYS/QCMDSRC) RCDLEN(92)

/* 创建绑定目录 */
             CRTBNDDIR  BNDDIR(ORDERSYS/ORDERBND)

/* 编译服务程序模块 */
             CRTRPGMOD  MODULE(ORDERSYS/ORDSRV) +
                          SRCFILE(ORDERSYS/QRPGLESRC)

/* 创建服务程序 */
             CRTSRVPGM  SRVPGM(ORDERSYS/ORDSRV) +
                          MODULE(ORDERSYS/ORDSRV) +
                          EXPORT(*ALL)

/* 添加服务程序到绑定目录 */
             ADDBNDDIRE BNDDIR(ORDERSYS/ORDERBND) +
                          OBJ((ORDERSYS/ORDSRV *SRVPGM))

/* 编译程序 */
             CRTBNDRPG  PGM(ORDERSYS/ORDENTRY) +
                          SRCFILE(ORDERSYS/QRPGLESRC) +
                          DFTACTGRP(*NO) +
                          ACTGRP(*NEW) +
                          BNDDIR(ORDERSYS/ORDERBND)

/* 创建命令 */
             CRTCMD     CMD(ORDERSYS/WRKORD) +
                          PGM(ORDERSYS/ORDENTRY) +
                          SRCFILE(ORDERSYS/QCMDSRC)

             SNDPGMMSG  MSG('部署完成')

             ENDPGM
```

附录

附录 A：常用命令速查表

```
系统命令速查表

库管理：
  CRTLIB - 创建库
  DLTLIB - 删除库
  ADDLIBLE - 添加库到库列表
  RMVLIBLE - 从库列表移除
  CHGCURLIB - 更改当前库

文件管理：
  CRTPF - 创建物理文件
  CRTLF - 创建逻辑文件
  DLTF - 删除文件
  WRKF - 处理文件
  CPYF - 复制文件

程序管理：
  CRTBNDRPG - 创建绑定 RPG 程序
  CRTRPGMOD - 创建 RPG 模块
  CRTPGM - 创建程序
  CRTSRVPGM - 创建服务程序
  UPDPGM - 更新程序
  CALL - 调用程序

作业管理：
  SBMJOB - 提交作业
  WRKACTJOB - 处理活动作业
  WRKJOBSCDE - 处理作业调度
  HOLDJOB - 挂起作业
  RLSJOB - 释放作业
  ENDJOB - 结束作业

用户管理：
  CRTUSRPRF - 创建用户配置
  CHGUSRPRF - 更改用户配置
  DLTUSRPRF - 删除用户配置
  WRKUSRPRF - 处理用户配置
```

```
系统管理：
  DSPSYSVAL - 显示系统值
  CHGSYSVAL - 更改系统值
  WRKSYSVAL - 处理系统值
  DSPSYSSTS - 显示系统状态

调试命令：
  STRDBG - 开始调试
  ENDDBG - 结束调试
  ADDBKP - 添加断点
  STEP - 单步执行
  DISPLAY - 显示变量
```

附录 B：RPG IV 操作码参考

```
常用操作码速查

文件操作：
  CHAIN - 按键随机读取
  READ - 顺序读取下一条
  READE - 读取相等键
  READP - 反向读取
  SETLL - 设置下限
  SETGT - 设置上限
  WRITE - 写入记录
  UPDATE - 更新记录
  DELETE - 删除记录

流程控制：
  IF/ELSEIF/ELSE/ENDIF - 条件分支
  SELECT/WHEN/OTHER/ENDSL - 选择分支
  FOR/ENDFOR - 循环
  DOW/ENDDO - Do While 循环
  DOU/ENDDO - Do Until 循环
  ITER - 跳过迭代
  LEAVE - 退出循环
  RETURN - 返回

字符串操作：
  %SUBST - 子串
  %TRIM - 去空格
  %LEN - 长度
```

```
  %REPLACE - 替换
  %SCAN - 扫描
  %XLATE - 转换
  %CHAR - 转字符
  %DEC - 转数值

数组操作：
  %LOOKUP - 查找
  %ELEM - 元素个数
  %SUBARR - 子数组
```

附录 C：错误代码参考

常见错误代码

```
CPF0000 - 通用错误
CPF2105 - 对象不存在
CPF2111 - 对象已存在
CPF2110 - 库不存在
CPF2112 - 对象正在被使用
CPF2207 - 用户未授权
CPF2401 - 对象损坏
CPF3141 - 文件操作错误
CPF502B - 文件打开错误
CPF5032 - 记录锁定
CPF5140 - 数组索引错误
CPF0864 - 文件结束
MCH1202 - 除零错误
MCH0601 - 数组索引超出范围
MCH0603 - 尝试使用空指针
```

总结

本教程全面介绍了 IBM iSeries 开发的各个方面，从基础概念到高级应用，涵盖了：

1. **系统基础**：了解 IBM i 的架构、对象系统和基本操作
2. **终端使用**：掌握 Personal Communications 的配置和使用
3. **RPG IV 编程**：从基础语法到高级特性，包括 ILE、SQL 和模块化编程

4. **CL 编程**：学习控制语言的系统管理和自动化功能
5. **实战案例**：通过完整的 ERP 订单管理系统项目，将所学知识融会贯通

IBM i 是一个强大且成熟的平台，随着技术的发展，它不断融入现代开发实践，如 Web 服务、JSON 处理、开源集成等。掌握 IBM i 开发技能，不仅能够维护传统系统，还能够参与系统现代化改造，为企业的数字化转型贡献力量。

希望本教程能够帮助您从 IBM i 初学者成长为熟练的开发者。持续学习和实践是精通任何技术的关键，祝您在 IBM i 开发之旅中取得成功！

文档版本：1.0
最后更新：2024年
字数统计：约 10 万字