**主机软件开发系列丛书**

**COBOL程序设计思想**

**孙树斌 阮利 编著**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*如您阅读本电子文稿，则视同您将遵守中国知识产权相关法律\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**目 录（自动生成）**

[第1章 COBOL85程序设计概述 8](#_Toc275157303)

[知识要点： 8](#_Toc275157304)

[§1.1 COBOL语言简介 8](#_Toc275157305)

[1.1.1 什么是COBOL 8](#_Toc275157306)

[1.1.2 COBOL语言的特点 8](#_Toc275157307)

[1.1.3 COBOL语言的历史与发展 8](#_Toc275157308)

[第2章 Cobol编程的基本元素 10](#_Toc275157309)

[知识要点 10](#_Toc275157310)

[§2.1 第一个COBOL程序——Hello COBOL！ 10](#_Toc275157311)

[§2.2 程序的基本结构 10](#_Toc275157312)

[2.2.1 部(DIVISION) 10](#_Toc275157313)

[2.2.2 节(SECTION) 11](#_Toc275157314)

[2.2.3 句子、语句和子句 11](#_Toc275157315)

[2.2.4 COBOL85程序结构及书写格式 11](#_Toc275157316)

[2.2.5 COBOL程序的数据分层 13](#_Toc275157317)

[§2.3 COBOL字符集 13](#_Toc275157318)

[2.3.1 COBOL字 14](#_Toc275157319)

[2.3.2 保留字 14](#_Toc275157320)

[2.3.3 表意常量 14](#_Toc275157321)

[2.3.4 用户字 15](#_Toc275157322)

[2.3.4.1 数据名 15](#_Toc275157323)

[2.3.4.2 常量 15](#_Toc275157324)

[§2.4 COBOL所处理的数据的特点 16](#_Toc275157325)

[§2.5 认识数据部 16](#_Toc275157326)

[2.5.1 了解数据部 16](#_Toc275157327)

[2.5.2 数据部的节 16](#_Toc275157328)

[2.5.3 COBOL数据 17](#_Toc275157329)

[2.5.4 数据组成层次结构 17](#_Toc275157330)

[§2.6 工作单元节（WORKING-STORAGE SECTION） 18](#_Toc275157331)

[§2.7 字型语句（PICTURE子句） 18](#_Toc275157332)

[2.7.1 数值型数据描述 18](#_Toc275157333)

[2.7.1.1 “9”描述符 19](#_Toc275157334)

[2.7.1.2 “S”描述符 19](#_Toc275157335)

[2.7.1.3 “V”描述符 19](#_Toc275157336)

[2.7.1.4 “P”描述符 20](#_Toc275157337)

[2.7.1.5 数值型变量小结 20](#_Toc275157338)

[2.7.2 文字型数据描述 20](#_Toc275157339)

[2.7.2.1 “A”描述符 20](#_Toc275157340)

[2.7.2.2 "X"描述符 21](#_Toc275157341)

[§2.8 赋初值子句（VALUE子句） 21](#_Toc275157342)

[§2.9 认识过程部 23](#_Toc275157343)

[2.9.1 了解过程部 23](#_Toc275157344)

[2.9.2 接收语句(ACCEPT语句) 24](#_Toc275157345)

[2.9.3 显示语句(DISPLAY语句) 27](#_Toc275157346)

[2.9.4 数据项初始化（INITIALIZE） 28](#_Toc275157347)

[2.9.5 传送语句(MOVE语句) 29](#_Toc275157348)

[2.9.6 停止语句(STOP语句) 30](#_Toc275157349)

[§2.10 算术运算语句 30](#_Toc275157350)

[2.10.1 加法语句(ADD语句) 30](#_Toc275157351)

[2.10.2 减法语句(SUBTRACT语句) 32](#_Toc275157352)

[2.10.3 乘法语句(MULTIPLY语句) 32](#_Toc275157353)

[2.10.4 除法语句(DIVIDE语句) 32](#_Toc275157354)

[2.10.5 计算语句(COMPUTE语句) 33](#_Toc275157355)

[§2.11 逻辑与控制语句 34](#_Toc275157356)

[2.11.1 条件表达式 34](#_Toc275157357)

[2.11.2 条件语句(IF语句) 35](#_Toc275157358)

[2.11.3 IF嵌套与分析 36](#_Toc275157359)

[2.11.4 转移语句(GOTO语句) 36](#_Toc275157360)

[§2.12 执行语句（PERFORM 语句） 38](#_Toc275157361)

[2.12.1 概述 38](#_Toc275157362)

[2.12.2 基本形 38](#_Toc275157363)

[2.12.3 PERFORM 语句与其他语句的区别 38](#_Toc275157364)

[2.12.4 PERFORM 语句的嵌套 39](#_Toc275157365)

[2.12.5 PERFORM语句序列中的转移 39](#_Toc275157366)

[2.12.6 PERFORM 语句语句计数循环 39](#_Toc275157367)

[2.12.7 PERFORM 语句UNTIL循环 40](#_Toc275157368)

[2.12.8 出口语句（EXIT） 40](#_Toc275157369)

[§2.13 复写语句 40](#_Toc275157370)

[§2.14 顺序文件操作 41](#_Toc275157371)

[2.14.1 顺序文件的定义 41](#_Toc275157372)

[2.14.2 顺序文件的读取 41](#_Toc275157373)

[2.14.3 顺序文件的写出 41](#_Toc275157374)

[第3章 数据处理 42](#_Toc275157375)

[知识要点 42](#_Toc275157376)

[§3.1 数据在计算机内的表示形式 42](#_Toc275157377)

[3.1.1 计算机内存的组织形式 42](#_Toc275157378)

[3.1.2 数值型数据在内存中的存储数据 42](#_Toc275157379)

[3.1.3 数据描述与存储形式的关系 42](#_Toc275157380)

[§3.2 用法子句（USAGE子句） 42](#_Toc275157381)

[§3.3 重定义子句(REDEFINES子句) 44](#_Toc275157382)

[§3.4 重命名子句(RENAMES子句) 45](#_Toc275157383)

[§3.5 遇零置空子句（BLANK WHEN ZERO） 45](#_Toc275157384)

[§3.6 对齐子句(JUSTIFIED子句) 46](#_Toc275157385)

[§3.7 同步安置子句（SYNCHRONIZED） 46](#_Toc275157386)

[§3.8 编辑型数据描述 47](#_Toc275157387)

[3.8.1 “.”描述符 47](#_Toc275157388)

[3.8.2 “+”描述符 47](#_Toc275157389)

[3.8.3 “-”描述符 47](#_Toc275157390)

[3.8.4 “，”描述符 48](#_Toc275157391)

[3.8.5 “$”描述符 49](#_Toc275157392)

[3.8.6 “\*”描述符 49](#_Toc275157393)

[3.8.7 插入空格、“0” 49](#_Toc275157394)

[3.8.8 “DB”、“CR” 50](#_Toc275157395)

[§3.9 描述符知识点整理 51](#_Toc275157396)

[§3.10 算术运算精度处理 52](#_Toc275157397)

[3.10.1 四舍五入处理子句（ROUNDED子句） 52](#_Toc275157398)

[3.10.2 长度溢出处理子句（ON SIZE ERROR） 52](#_Toc275157399)

[3.10.3 除法语句的余数子句(REMAINDER) 53](#_Toc275157400)

[§3.11 对应传送子句（或称作同名传送CORR） 53](#_Toc275157401)

[§3.12 对应项间的运算（CORR） 55](#_Toc275157402)

[第4章 字符串处理 56](#_Toc275157403)

[知识要点 56](#_Toc275157404)

[§4.1 字符串连接（STRING） 56](#_Toc275157405)

[§4.2 字符串分解语句（UNSTRING） 57](#_Toc275157406)

[§4.3 检测语句（INSPECT语句） 58](#_Toc275157407)

[本章小结 60](#_Toc275157408)

[第5章 逻辑控制 60](#_Toc275157409)

[**学习提示** 60](#_Toc275157410)

[§5.1 逻辑控制语句 60](#_Toc275157411)

[§5.2 条件表达式 60](#_Toc275157412)

[§5.3 IF-ELSE语句 60](#_Toc275157413)

[§5.4 EVALUATE语句 61](#_Toc275157414)

[§5.5 CONTINUE语句 62](#_Toc275157415)

[§5.6 GO TO 62](#_Toc275157416)

[§5.7 条件关系 62](#_Toc275157417)

[§5.8 复合条件 63](#_Toc275157418)

[§5.9 符号条件 63](#_Toc275157419)

[§5.10 类型条件 64](#_Toc275157420)

[§5.11 条件名条件 64](#_Toc275157421)

[§5.12 循环处理 65](#_Toc275157422)

[§5.13 带有循环控制变量的PERFORM 语句 67](#_Toc275157423)

[5.13.1 一重循环 67](#_Toc275157424)

[5.13.2 三重循环的PERFORM 语句 68](#_Toc275157425)

[本章小结 69](#_Toc275157426)

[第6章 表与检索 69](#_Toc275157427)

[**学习提示** 69](#_Toc275157428)

[§6.1 表的基本概念 69](#_Toc275157429)

[§6.2 表的建立 71](#_Toc275157430)

[§6.3 表元素的引用 73](#_Toc275157431)

[§6.4 表元素赋初值 73](#_Toc275157432)

[§6.5 表的应用 74](#_Toc275157433)

[§6.6 位标法 75](#_Toc275157434)

[6.6.1 位标名 75](#_Toc275157435)

[6.6.2 位标法引用表元素规则 75](#_Toc275157436)

[6.6.3 设置语句 75](#_Toc275157437)

[§6.7 表的顺序检索 76](#_Toc275157438)

[§6.8 表的对分检索 77](#_Toc275157439)

[§6.9 用PERFORM语句对表进行检索 78](#_Toc275157440)

[本章小结 79](#_Toc275157441)

[第7章 子程序 79](#_Toc275157442)

[**学习提示** 79](#_Toc275157443)

[§7.1 子程序概述 79](#_Toc275157444)

[§7.2 主程序与子程序连接 80](#_Toc275157445)

[§7.3 子程序返回语句 81](#_Toc275157446)

[§7.4 公用区数据 81](#_Toc275157447)

[§7.5 子程序举例 82](#_Toc275157448)

[本章小结 83](#_Toc275157449)

[第8章 程序部详解 83](#_Toc275157450)

[**学习提示** 83](#_Toc275157451)

[§8.1 标识部 83](#_Toc275157452)

[§8.2 环境部 84](#_Toc275157453)

[8.2.1 环境部概述 84](#_Toc275157454)

[8.2.2 配置节 85](#_Toc275157455)

[8.2.3 文件组织方式 90](#_Toc275157456)

[8.2.4 输入输出节 92](#_Toc275157457)

[8.2.4.1 输入输出节概述 92](#_Toc275157458)

[8.2.4.2 文件控制段 93](#_Toc275157459)

[8.2.4.3 输入输出控制段 99](#_Toc275157460)

[§8.3 数据部 100](#_Toc275157461)

[8.3.1 数据部概述 100](#_Toc275157462)

[8.3.2 文件节 105](#_Toc275157463)

[8.3.3 工作单元节（WORKING-STORAGE SECTION） 108](#_Toc275157464)

[§8.4 过程部 116](#_Toc275157465)

[本章小结 117](#_Toc275157466)

[第9章 文件处理 118](#_Toc275157467)

[**知识要点：** 118](#_Toc275157468)

[§9.1 文件的基本概念 118](#_Toc275157469)

[§9.2 文件处理常用语句概述 118](#_Toc275157470)

[9.2.1 打开语句（OPEN语句） 118](#_Toc275157471)

[9.2.2 读语句（READ语句） 119](#_Toc275157472)

[9.2.3 写语句（WRITE语句） 120](#_Toc275157473)

[9.2.4 重写语句REWRITE 121](#_Toc275157474)

[9.2.5 删除语句DELETE 121](#_Toc275157475)

[9.2.6 关闭语句（CLOSE语句） 121](#_Toc275157476)

[9.2.7 开始语句START 121](#_Toc275157477)

[9.2.8 文件操作汇总 122](#_Toc275157478)

[§9.3 磁带文件 122](#_Toc275157479)

[§9.4 磁盘顺序文件 124](#_Toc275157480)

[9.4.1 顺序文件的概念 124](#_Toc275157481)

[9.4.2 COBOL中与顺序文件有关的部分 124](#_Toc275157482)

[9.4.3 顺序文件实例 126](#_Toc275157483)

[§9.5 磁盘索引文件 129](#_Toc275157484)

[9.5.1 索引文件的概念 129](#_Toc275157485)

[9.5.2 COBOL中与索引文件有关的部分 129](#_Toc275157486)

[9.5.3 建立索引文件 130](#_Toc275157487)

[9.5.4 检索索引文件记录 131](#_Toc275157488)

[9.5.5 新增索引文件记录 133](#_Toc275157489)

[9.5.6 修改索引文件记录 134](#_Toc275157490)

[9.5.7 删除索引文件记录 134](#_Toc275157491)

[9.5.8 综合实例 135](#_Toc275157492)

[§9.6 磁盘相对文件 137](#_Toc275157493)

[9.6.1 相对文件的概念 137](#_Toc275157494)

[9.6.2 COBOL中与相对文件有关的成分 137](#_Toc275157495)

[9.6.3 建立相对文件 137](#_Toc275157496)

[§9.7 动态存取 138](#_Toc275157497)

[§9.8 文件综合应用 138](#_Toc275157498)

[9.8.1 与表操作结合实例 138](#_Toc275157499)

[本章小结 139](#_Toc275157500)

[第10章 排序 140](#_Toc275157501)

[**学习提示** 140](#_Toc275157502)

[§10.1 基本概念 141](#_Toc275157503)

[§10.2 环境部中排序文件的描述 141](#_Toc275157504)

[§10.3 数据部中排序文件描述 141](#_Toc275157505)

[§10.4 排序语句（SORT）单纯排序 142](#_Toc275157506)

[§10.5 排序语句（SORT）加工排序 143](#_Toc275157507)

[本章小结 144](#_Toc275157508)

[第11章 合并 144](#_Toc275157509)

[学习提示 144](#_Toc275157510)

[§11.1 基本概念 144](#_Toc275157511)

[知识小结 149](#_Toc275157512)

[第12章 画面 150](#_Toc275157513)

[学习提示 150](#_Toc275157514)

[本章小结 150](#_Toc275157515)

[第13章 报表 150](#_Toc275157516)

[**学习提示** 150](#_Toc275157517)

[§13.1 报表概述 151](#_Toc275157518)

[§13.2 报表定义 151](#_Toc275157519)

[本章小结 155](#_Toc275157520)

[第14章 嵌入式数据SQL 155](#_Toc275157521)

[**学习提示** 155](#_Toc275157522)

[§14.1 ORACLE PRO\*COBOL编程 156](#_Toc275157523)

[本章小结 156](#_Toc275157524)

[第15章 COBOL与其他编程语言 156](#_Toc275157525)

[**学习提示** 156](#_Toc275157526)

[§15.1 C语言的接口 156](#_Toc275157527)

[本章小结 156](#_Toc275157528)

[第16章 COBOL内部函数 156](#_Toc275157529)

[学习提示 156](#_Toc275157530)

[§16.1 函数简介 157](#_Toc275157531)

[本章小结 157](#_Toc275157532)

[第17章 COBOL程序设计思想 157](#_Toc275157533)

[**学习提示** 157](#_Toc275157534)

[§17.1 结构化程序设计 157](#_Toc275157535)

[§17.2 基于面向对象思想的程序设计 158](#_Toc275157536)

[§17.3 匹配处理 158](#_Toc275157537)

[**本章小结** 158](#_Toc275157538)

[附录 1](#_Toc275157539)

[附录1 COBOL关键字 1](#_Toc275157540)

[附录2 内部函数一览 1](#_Toc275157541)

# COBOL85程序设计概述

知识要点：

1.了解COBOL发展简史及应用领域，为理解COBOL85相关语句含义作基础

2．通过“hello Cobol！”小程序介绍COBOL85的基本概念。

## COBOL语言简介

### 什么是COBOL

COBOL是Common Business Oriented Language (商业通用语言)的缩写 .不仅是商业数据处理的理想语言，而且广泛用于数据管理领域，如财会工作，统计报表，作业调度，情报检索，人事管理，证券银行等金融领域。 据资料统计，世界上约有一半程序使用COBOL编写的。

请注意COBOL的应用领域，大部分都是一些表格多的管理信息系统，这和COBOL数据的层次关系、表、文件应用紧密相连，对于这些应用领域业务的熟悉程度对于理解领会COBOL语句和程序设计方法有一定好处。

### COBOL语言的特点

描述性好，能根据需要描述各种形式数据，打印的帐表可以不加处理，直接作为商业单据。

最适于数据处理领域 ，能够对数据进行严密的组织。数据通常以文件的形式，存放在外部介质上，能够对大量数据很方便的进行添加、更新、删除处理。

比较接近于自然语言（英语），成文自明。

通用性强 ，移植方便。

COBOL格式固定，有较好的模块化结构，层次分明，使程序易于编制和维护。便于程序间数据交换。

COBOL缺点是比较繁琐，四个部必须定义和说明。

### COBOL语言的历史与发展

1946年在美国炭夕法尼亚大学莫尔电气工程学院诞生了世界上第一台电子计算机，标志着计算机时代的到来和信息社会的来临。计算机是二十世纪最伟大．具有最深远和深刻影响的科学发明，它的出现极大地改变了人类社会的面貌，极大地加快了科学、技术和文明的进步过程。在有关新的世界技术革命的种种议论中，有一种共同的，一致的认识．即计算机一信息在这场革命中将作为带头的．核心的学科而发挥无可估量的作用；当然，计算机本身也是处于日新月异的发展过程中的。初期的计算机主要用于计算科学与工程技术中的数值计算问题，不能用于数据处理。这主要有两个原因：

1.当时计算机只会处理数字不会处理字母、符号；

2.当时计算机上没有大容量的存贮器。

这两个问题到50年代切就很快得到了解决．1951年．sperry Rand公司研制的UNAVAC-1第一次成功地解决了在计算机上表示相处理字母及其它符号的问题(UNAVAC-1也是世界上第一个批量生产的商用计算机型号）．同年，LEOI计算机第一次用既能正转，也能反转的磁带作为外贮存器；这两个技术进展为计算机的非数值应用奠定了基础。1953年．美国通用电气公司在UNAVAC-1上研制成一个用于处理发放职工工资．支票往来．帐目支付．分类帐和总帐的系统．被认为是世界上第一个汁算机辅助企业管理系统。1956年，IBM公司和Reminton rean公司几乎同时推出磁盘存贮器，为电子数据处理注入了新的血液．

与此相适应，各计算机公司研制与建立了一些适台于数据处理需要的系统，如，IBM公司的COmmercial Transltor,SPerry Rand公司的FLOW MATlC，Honeywcll公司的FAct等．它们各有特色，但都不够理想。

1959年5月，由美国国防部(计算机的最大用户)出面召集了一个会议,叫“数据系统语言会议” (conference on Data systems Longuage)．专门讨论创建一种用于数据处理的语言的问题，对这种语言的主要要求如下：简单英语动词，易用性优先于功能，数据与过程分离。这样COBOL的语法就借用了段、句、词的概念。会议的结果是成立了一个九人的CODASYL委员会，由它负责起草语言文本．委员会经过紧张的工作，于1959年12月完成了COBOL语言初稿．并于1960年4月正式发表，COBOL语言从此诞生。这第一个COBOL文本称为“COBOL-60”。

COBOL创始人致力于扩大潜在的编程队伍，是新程序的开发要求商业知识而不要求对特定计算机的过细了解。COBOL创始人认为这种编程语言应能在多种计算机系统间移植。

COBOL语言诞生后不断改进、完善、发展；上述CODASYL委员会不但继续存在，并且不断扩充它的活动范围。第一个COBOL文本公布以来的，先后产生过许多修改本。比较重大的有61、65．68、69．70、73、85、2002等文本。

1959年COBOL-61(最早的版本)

1965年COBOL-65

1974年COBOL-74

1985年COBOL-85 (应用最广泛的版本)

2000年Fujitsu推出COBOL For .Net（在.net平台上进行可视化编程，加入了面向对象成分,可直接生成ASP.Net的WEB页面）

2002年COBOL02002

国内行业的一些应用案例：

建行: COBOL + IMS +CICS + xpeditor (compuware产品，Debug工具）

农行: C + DB2 + CICS (其他不详）

工行：COBOL + DB2 + CICS (其他不详）

中行：COBOL + DB2 + CICS

常见的编译器有IBM、NEC、Fujitsu、HITACHI等等

# Cobol编程的基本元素

知识要点

本章通过一个简单的COBOL程序，介绍了COBOL程序设计的最基本结构。介绍了COBOL的最基本的核心概念，包括COBOL最基本的常用语句（输入输出、逻辑转移、循环、顺序文件）。

## 第一个COBOL程序——Hello COBOL！

程序代码

例：

**000010\*COBOL85**

**000020 IDENTIFICATION DIVISION.**

**000030 PROGRAM-ID. HELLO.**

**000040 ENVIRONMENT DIVISION.**

**000050 CONFIGURATION SECTION.**

**000060 DATA DIVISION.**

**000070 WORKING-STORAGE SECTION.**

**000080 PROCEDURE DIVISION.**

**000090 PRINT.**

**000100 DISPLAY "HELLO COBOL!".**

**000110 STOP RUN.**

**000120 END PROGRAM HELLO.**

注释

段

节

部

过程语句

程序出口

过程部结束

## 程序的基本结构

### 部(DIVISION)

具体的程序可读性取决于编程人员对程序中使用的各个元素名称的选择。过去的COBOL标准要求全部用大写字母，但COBOL85允许采用小写字母。

COBOL程序有着严谨的结构。四大部缺一不可，即使部内并无内容也要写上“部首”。每个COBOL程序，不管是解决复杂问题的大程序还是极其简单的小程序，都由以下四个部分所组成：

标识部(IDENTIFICATION DIVISION)

标识部的功能是提供程序的一般性文档说明，主要用来指定源程序名字，也可以写入其它用作备忘的某些信息(如编写和编译源程序日期、作者、程序说明、修改履历等)。

环境部 (ENVIRONMENT DIVISION)

环境部提供与程序外部有关的项目，指出程序中用到的数据文件名与计算机硬件系统的设备的对应关系，制定目标程序中使用的专门控制方法，确定并指明该程序所用计算机内存区的大小。

数据部(DATA DIVISION)

    提供程序变量描述程序中用到的全部数据（输入、输出、中间数据）的类型（数值型、字符型）、结构（数值型、字符型）以及内存中占用情况。

COBOL85中对不同数据类型分成不同的节：文件数据（File Section）,静态数据（Working-Storage Section）、参数（Linkage Section）和其他。

过程部(PROCEDURE DIVISION)

书写程序中进行一系列运算和数据处理所必须的指令，使计算机产生相应操作。源程序中由程序编写者自行定义的过程部的节名或段名统称为过程名。

### 节(SECTION)

在标识部下面不设节而直接设段。除标识部外，在每个部下面可以有若干个节(SECTION)，每一个节以节头作为标识。每个节下面可以包括若干段。

COBOL提供了一些固定的节，我们也可以根据自己的需要定义自己的节。(定义方法： 节名 SECTION.)

在数据部内可以设节，也可以不设节，但不能设段。节下面可以有若干描述体。数据部中若不设节，则下面就不能出现描述体。

过程部中可以设节，下面再设段，在简单的程序中也可以直接设段而不设节。

### 句子、语句和子句

   每一个段由若干个句子组成，句子中又由若干语句组成，语句中又包含若干个子句。

   过程部中的一个节或者一个段由若干个句子(SENTENCE)组成，一个句子是以‘.’和空格结束的。一个句子又可以由若干个子句(CLAUSE)组成

### COBOL85程序结构及书写格式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 标号区 | 指示符区 | A区 | B区（正文区） | 注  解 |
| 1    6 | 7 | 8 11 | 12                       72 | 73       80 |
| 000010  000020  000030  000040  000050  000006  000007  000008 | \* | COBO  IDEN  PROG  ENVI  DATA  WORK  PROC  PRIN | L85  TIFICATION   DIVISION.  RAM-ID.      HELLO-COBOL.  RONMENT      DIVISION.  DIVISION.  ING-STORAGE SECTION.  EDURE        DIVISION.  T.  DISPLAY     'HELLO COBOL！'  STOP        RUN. | 注释行  标识部    程序ID  环境部  数据部  工作单元节  过程部    PRINT段    显示内容  结束程序 |

1. 行号区，1—6列。

行号区用于编写程序的页号．行号。通常前3列为页号，后3列为行号，由于COBOL程序中的转栘是由节名．段名来标识的，而不是以语句标号来标识的，所以COBOL程序的行号区可以全空。若给语句编行号。必须从小到大排列．但不一定连续，这时编译程序将俭查行号的递增性，遇有行号颠倒时将给出出错信息(但也有系统不作此种检查．

 2.指示符区。

用于放置续行标志“-”，即如果一非数值常量太长．在一行中写不下而需转至下行时．在下行的第7列需加“—”以作标志，同时在续行B区(见下)的始端添加引号；有些系统也允许数值常量保留字换行，以续行区中加“—”标志．但不推荐这类换行；

在续行区还可放以下不同符号而有不同的用途：

(1)”\*”表示该行是注解行，编译程序将不予处理；

(2）“/”也表示该行是注解行．同时还通知编译程序，在编译后生成的列表文件（listing file）中从本行起应换页打印．

(3)"D"表示该行为调试行．

在COBOL85中指示符区只允许5个字符：空格、\*、/、-和D.

3.A区 8～11列

A区用以书写部、节、段的标题和标志过程部中直陈性节开始和结束的DECLARATIVES和END DECLARATIVES以及层指示符FD、SD、RD、CD和层号01及77。这些内容必须从A区开始书写，但为了整齐划一，使程序清晰易读，建议一律写在A区的始界第8列上。

1. B区，12一72列：

B区以书写程序正文及从02一49的各层数据．可从任意一列开始书写．最好也一律从B区边界第12列开始书写（不同层号最好缩进以加区别）； 过程部语句必须从B区开始

5．注解区，73—80列

这个区用以写注释性内容．通常可写入程序名，该区内容不予编译，所以千万注意正文内容不要延伸到注解区（包括语句结束的语句）

书写时应注意的几点：

所有字母通常应大写（根据编译器要求），引号内的字符串例外。

两个COBOL字之间留一个空格，逗号，句号，分号左侧不留空格，右侧留空格。

允许一行写几个语句，也允许一个语句写在多行上。

句号用于下列场合：

* 1. 部、节、段的标题之后；
  2. 标识部和环境部中每段正文结束处；
  3. 数据部中每层数据描述结束；
  4. 过程部中每一命令语句结束。

在COBOL中，必须用英文句号结束部标题、节标题、段名、项目和语句。如果省略英文句号，则可能程序无法编译。用句号作为小数点时，两边都要有数字，否则编译器会把它看成句子结束。

逗号可用于下列场台作分隔符：

1. 两个操作数之间；
2. 两个下标之间；
3. 一系列语句或子句之间。

分号可用于一系列语句或子句之间作分隔符。

    实际上，作为分隔符的逗号和分号常常是可以省略的，也就是说，可以用空格作分隔符来代替。根据经验．可用可不用的标点符号以不用为好，避免增加出错的可能。

### COBOL程序的数据分层

COBOL程序中的数据也有一个层次关系。大体说来，COBOL中的数据有如下层次文件(file)  记录（record）组合数据硕(group item） 初等数据项（elementary item）。

文件是存贮在外部没备上的某类信息(通常是同类记录)的集合。在高级程序设计语言中，COBOL是第一个明确提出文件慨念的。

    记录是描写某类实体(entity)的若干属性(attribute)的集合．实体的属性很多，在不同的应用中视需要取不同的属性。

数据项是属性在计算机上的表示，在其它程序设计浯言中有叫做字段(field）的；数据项有两种，最原始．最基本．不能再分的叫初等数据项．如人的年龄．性别等，它表示—个确定的属性。另一类叫组合数据项，简称组项。组项又分两种，一种叫”向量组项”．是若干相关的初等数据项的一维的、有序的集合。另一种组项叫“重复组”(repeating group)是同类数据项或组项的多次重复出现。

在COBOL程序中，数据之间的层次关系通过不同的层号来表示；除了文件用层指示苻FD(file description)标志以外，其后的层次规定记录用层号01，记录中的组项，初等项用02—49层次愈低，层号愈大，但不一定连续．另有几个特殊层号，77用于孤立数据项，66用于重定义数据项，88用于条件名，在关于数据部的章节会对层号进行详细描述。

## COBOL字符集

系统字符集 ：指的是在输入输出操作中允许出现的字符的集合。

COBOL字符集：指的是在COBOL程序中允许出现的字符（用引号引起来的字符串中的字符除外）其中包括：数字（0～9），大写字母（A～Z），小写字母：a～z ，以及+ -\*/=,.;’()<>$等，共十五个专用字符（根据系统可能不同）

|  |  |  |
| --- | --- | --- |
| 字符 | 名称 | 功能 |
| 0……9 | 数字 | 1．组成数值常量； 2.0和9可用于PICTURE中。 |
| A……Z | 英文字母 | 1.组成保留字，字符串，用户字；2.A、X、Z、B、CR、DB可用于PICTURE中。 |
|  | 空格 | 分割符 |
| + | 加号 | 1．运算表达式中用作加号；2.数字符号；3.PICTURE中作描述符 |
| - | 减号 | 1.运算表达式中用作减号;2.连字符;3.数字符号;4.PICTURE中作描述符 |
| \* | 星号 | 1.运算表达式中用作乘号;2.用于第7列标志注释行;3.PICTURE中标志遇零置星 |
| / | 斜线 | 1.运算表达式中用作除号;2.PICPICTURE中作插入符 |
| = | 等号 | 1.运算表达式中用作等号 |
| $ | 美元符号 | 1．钱币符号；2. PICPICTURE中作插入符 |
| ， | 逗号 | 1．标点符号；2. PICPICTURE中作插入符 |
| ； | 分号 | 1．标点符号 |
| . | 句号 | 1．标点符号；2.小数点；3. PICPICTURE中作插入符 |
| “ | 引号 | 1．非数值直接量的前后限界符 |
| （ | 左括号 | 用于表达式 |
| ） | 右括号 | 用于表达式 |
| 〉 | 大于号 | 用于表达式中表示大于 |
| < | 小于号 | 用于表达式中表示小于 |

### COBOL字

### 保留字

保留字是在COBOL中已规定做专门用途的字，他们代表特定的含义。这些保留字不能另做它用。如：MOVE代表传送，ADD代表加操作，DISPLAY代表显示等

1.关键字(key word)，即必写字，使程序语句中必写的成分。

2.任选字(optional word)。是语句中可写可不写的成份；写上可使语句较为完整，没有也不影响语义，

3.连接词(connective)，起连接作用的保留字；有三种连接词：

(1)限定词连接字(qualifier-connective),用以连接数据名、段名和它们的限定词。在COBOL程序中，数据名和过程名应该是唯—定位的，不允许重复；在特殊情况下，两个记录中出现同名的数据项，或过程部的两个节中出现同名的段．则在引用时需加以限定，如：

NAME IN PERSONAL

TEST-1 OF DO-ACCOUT SECTION

 其中的IN，OF就是限定词连接词；

(2）逻辑连接词(logical-connective)，用以连接条件语句中的两个或更多个条件，包括AND和OR．

(3)序列连接词(series—connective)，用以连接两个或多个并列的语法成分。实际上它们是标点符号”.”或“；”．

### 表意常量

表意常量是用某些英文字（保留字）来代表某些特定的常值。

|  |  |
| --- | --- |
| 表意常量 | 代表的值 |
| ZERO , ZEROS , ZEROES | 表示一个或多个零 |
| SPACE , SPACES | 表示一个或多个空格字符 |
| HIGH-VALUE , HIGH-VALUES | 表示一个或多个具有“最高值”的字符 |
| LOW-VALUE , LOW-VALUES | 表示一个或多个具有“最小值”的字符 |
| QUOTE , QUOTES | 表示一个或多个引号字符 |
| ALL 常量 | 表示一个或多个由该常量组成的字符串 |

其中：表意常量的单复数形式是等价的，表意常量的长度由数据长度所决定，表意常量不和数据发生联系时则认为其长度为1。

只有ZERO、ZEROS和ZEROES可以同时作为非数值常量和数值常量使用。

### 用户字

用户字包括数据名，文件名，条件名，助记名，过程名(分节名和段名）以及层号六种，是由程序员按下列规则自行选定的任意字苻串(层号例外）。

用户字构成规则：

* 1. 由字母、数字符和连接号“-”组合而成；
  2. 字符组合中至少含有一个字母或连字符，但段名可以由数字组成，数字应为正整数；
  3. 连接号只能出现在用户字的中间任何位置，而不能出现在用户字的两端；
  4. 字长不允许超过30个字符；
  5. 用户字中不能出现空格符（空格符在COBOL中是分隔符，表示一个COBOL字的结束）。

#### 数据名

数据名相当于其他编程语言（如C、java）中的变量名，代表具体的数据项，COBOL中指的数据是广义的，可以是数值或字符、字符串。

数据名对应计算机中一段存储区，代表的数据项应在数据部进行定义；

数据名、段名、程序名不能重名。

#### 常量

常量分为数值常量和非数值常量。

数值常量即数学上的常数，具有数据类型。

数值常量要点：

1. 小数点不能多于一个，而且不能出现在常数的最右边。
2. 数值常量的长度不能超过18位数字（视编译系统而不同）。
3. 至少有一个数字，至多有一个正或负号，且只能出现在最左边。
4. 数字之间和正负号后不能有空格。

举例：123，+3.14159，-12.6等

非数值常量：用引号（双引号或单引号）括起来的字符串。

非数值常量要点：

1. 可以包括字母、数字、空格和其他字符
2. 由纯数字字符组成的非数值常量不能用于计算，只能用于显示，要用引号括起来，‘123’与123是不同的
3. COBOL保留字也可以用引号括起来作为非数值常量
4. 不应超过120个字符，包括常量中的空格。
5. 把引号也包含在字符串内，可写作：MOVE QUOTE”SHANGHAI”QUOTE TO X

有的系统可以用单引号嵌套双引号，或者反之来实现。

举例：’ABCD’,’123’,’$123’，“‘1’”

## COBOL所处理的数据的特点

层次：在数据处理中，数据之间并不是互相孤立的，而是存在从属关系，这种关系称为层次。

记录：是具有一定层次关系的一组数据项的最大集合，它是内存中具有独立逻辑含义的的最大可存储项。

文件：多个记录可以组成文件，文件是被记录在外部介质上的记录的集合。

库：若干个文件组织成一个库。

初等项：是数据的基本单位，不可再分。

组合项：由若干个初等项和低一层组合项组成。

层号：由两位整数表示，用不同的层号表示它们之间的从属关系，层号小的表示高的层次。

## 认识数据部

### 了解数据部

数据部是整个COBOL源程序中唯一描述数据的部分。凡是在程序中涉及的数据都要在数据部中加以说明（描述）。 如文件记录、变量、表格和报表。数据项的定义包括名称、特征与其他数据项的关系。

数据部包含的每一节针对COBOL程序使用的每种数据。

### 数据部的节

    数据部包含程序使用的所有局部数据的定义（数据描述）。针对COBOL程序使用的每种数据，每个节前面有节标题。每个节都是可选的，但程序使用的所有数据都应在数据部的某个节中定义。尽管每个节都是可选的，但程序中的节应按指定顺序出现：

文件节（FILE SECTION）：用来描述程序中用到的输入文件和输出文件及其记录中各数据项的属性。 包括打印文件和存储文件。每个文件描述中包括所含记录的定义。文件节中的数据或者从文件读取，或者由要写入文件的程序产生。

工作单元节（WORKING-STORAGE SECTION）：用来描述程序中用到的 临时结果和初始化静态数据的数据。用以描写文件以外的（也就是和文件无关的）独立的数据项和记录，通常用来保存程序运行中的中间结果、固定的标题栏的内容．控制变量，统计计 数等等。

联接节（LINKAGE SECTION）： 只用于子程序，用来描述调用程序间发生数据传递的数据项。 描述主调程序和被调程序之间的结合参数（子程序使用主程序的数据，主程序便用子程序运行的结果数据）。

通讯节（COMMUNICATION SECTION）：描述作为程序间接口的数据和消息控制系统，用以实现与通信设备相关的消息分析的COBOL方法。通信节属COBOL通信模块的一部分，可以处理COBOL程序和通信设备间的消息。

报表节（REPORT SECTION）:为了完成报表编制功能，此节用来规定欲输出的报表的“体裁”，设计各报表栏的打印形式和方法等。

### COBOL数据

COBOL中最基本的数据项称为基本数据项

数据分类：

初等数据项：没有内在联系，不可再分，具有独立的逻辑意义，包括孤立的和具有从属关系的两种

组合项：由若干孤立项（数据类型可以不同）组成，互相关联，具有一定内在联系和从属关系

### 数据组成层次结构

可以将基本项目组成层次结构，称为组数据项或组，组可以分成子组，最终分成基本项目不能再划分。

数据的从属关系用层次关系来描述，COBOL程序中数据项的定义从层号开始。层号只能是整数。01层是最高的，49层是描述数据的最底层，数据层次结构的层号从01开始，到49结束，记录定义为01层，为最高层。从属项层号比上属层层号大，即01层包含02层。77层项目用于某个组的数据项（孤立数据项，如程序中的临时变量），66层、88层后面讲到相关内容具体介绍。

数据的层次结构是：记录→组合项→初等项

举例：

|  |
| --- |
| 77 A PIC 99. ————————孤立项  01 A. ————————记录  03 A1 PIC 99. ————————初等项  03 A2. ————————组合项  05 B1 PIC XX. ——————初等项 |

几个数据项从属于同一个组合项，但互不从属，则位于同一层，具有相同的层号。处于同一层次，但分别从属于不同组合项的那些数据项的层号可以不相同。

记录和文件的关系：记录是具有一定层次关系的一组数据项的最大集合。是内存中具有独立逻辑含义的最大可存取项，具有最高的层次，层号为01。多个记录可以组成数据文件。

  引用组数据项，可以不考虑其中的组件而整个作为单元，从一个地方移到另一地方。只有基本项目可以表示数学处理所用的数据。

## 工作单元节（WORKING-STORAGE SECTION）

工作单元节：程序中用到的中间数据应在本节中描述，还可以利用工作单元节为某些数据赋以初值。

工作单元节中描述的数据项分为两种：

孤立的数据，以层号77开头

组合项：以01到49之间的一个数字作为开头

通常应先写77层，再写01到49层。

66用于重命名、88用于条件名。

## 字型语句（PICTURE子句）

PIC子句用来描述每一个初等数据项的一般的特征和编辑要求。

用于说明：

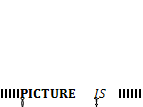
1.数据项是什么类型的

2.数据项占用多大的内存域

3.是否需要按打印的要求准备相关字符

组合数据项不允许用PIC子句

语法：



数据描述

PICTURE可以缩写成PIC，IS可以省略

用法如前面所示例子。

### 数值型数据描述

数值型数据项描述语法：



#### “9”描述符

9表示该位置上可以放入一个0～9之间的数字

举例：

77 A PIC 999. 表示放入000～999之间三位数字

只能放入0～9之间的数字，不能放入空格，有几个9表示内存单元长度是几个字节，实际存放时按右对齐原则存放，长度多余在左端补零，长度不足，则高位数截去。只能表示正整数，如果输入小数，则小数部分被舍去，如输入负数，则负号被舍去（或者使用S描述符描述数值的符号，后面介绍）

举例：

|  |  |  |
| --- | --- | --- |
| 描述 | 数值 | 内存中内容 |
| 02 A1 PIC 99. | 12 | 12 |
| 02 A2 PIC 9（6）. | 4567 | 004567 |
| 02 A3 PIC 9（3）. | 1.3 | 001 |
| 02 A3 PIC 9（3）. | -12 | 012 |

#### “S”描述符

用于描述带符号的数，尤其是负数。S不计入数据项的长度中，数值的正负号信息通常记录在最后一个字节中，根据编译器不同也可能不同。

S必须是PIC语句中数据类型描述中最左边的一个字符，在内存中S不占位，在运算时S起作用，但数据符号不会被输出。

举例：

|  |  |  |
| --- | --- | --- |
| **描述** | **数值** | **内存内容** |
| **02 C1 PIC S9（3）.** | **+26** | **026** |
| **02 C2 PIC S99.** | **-3.5** | **03** |

#### “V”描述符

指出在数值数据结构中隐含的小数点的位置，对V描述符不分配存储空间，故在V出现的位置上并没有一个真正的小数点存在，只是在计算中，根据V的位置作为小数点位置与其它数据进行对位或运算。

一个数据描述中只能有一个V，V可以是数据描述中最后的字符，则等于无小数点，由于V不占内存，只在传送和运算时有效，显示时只将内存中各字节内容显示，小数点不被显示，因此用V描述的数值型数据项不能直接用于显示和打印。如果想打印小数点应使用编辑型数据。

举例： 03 M PIC 999V99.

如果 MOVE 123.45 TO M.

则内存中内容是 123|45 红色表示隐含的小数点的位置

举例：

|  |  |  |
| --- | --- | --- |
| **描述** | **数值** | **内存中内容** |
| **02 B1 PIC 9V99** | **4.59** | **4|59** |
| **02 B2 PIC 9(4)V99.** | **1766** | **1766|00** |
| **02B3 PIC S9(3)V99.** | **-40.3** | **040|30** |

#### “P”描述符

又称比例因子，主要用于十分大或十分小数。用P描述不分配存储空间，但在计算机处理时会用零替代每一个P。

P必须出现在全部9之前或之后。在前零前有一小数点，9后有N个P，表示\*10n，9前有N个P，表示\*10-m，由于P不占内存，显示或打印时P不出现，只显示出内存中实际存放的数字而不显示隐含的零。

如1\*109描述 02 B PIC 9PPPPPPPPP或02 B PIC 9P(9)

如0.000065描述02 B PIC VP(4)99或02 B PIC P(4)99

#### 数值型变量小结

|  |  |  |
| --- | --- | --- |
|  | 说明 | 举例 |
| ‘9’ | 在变量中的每一位只能存放0-9之间的数字。 | 02  A  PIC   9999.  4568  02  B  PIC   9(4). |
| ‘V’ | 指出在定义的变量中隐含的小数点的位置。 | 02 A PIC  999V99.  123.82  02  B PIC  9(3)V9(2). |
| ‘P’ | 变量低位的若干的零可以用‘P’来表示。 | 02 A PIC  9PPPP.  30000  02 B PIC  9P(4). |
| ‘S’ | 定义一个带符号的数。 | 02 A PIC  S99.  -30  02 B PIC  S9(2). |

### 文字型数据描述

#### “A”描述符

字母型数据项中只允许存放字母或空格。按A的个数分配存储单元。字母数据送入字母型数据项时，按“左对齐原则”存放，若数据过长，则按数据项的长度截断右边多余的字母；若数据不足，则右边补空格。

举例：

|  |  |  |
| --- | --- | --- |
| **描述** | **数值** | **内存内容** |
| **02 D1 PIC AAA.** | **ABC** | **ABC** |
| **02 D2 PIC A(5).** | **NAME** | **NAME＿** |
| **02 D3 PIC A(4).** | **YES** | **YES ＿** |

#### "X"描述符

在字符型数据项中存放由任意的字符组成的数据。存放在字符型数据项中的数字不能参加运算，字符数据存入字符型数据项时，也按"左对齐原则"存放，字符数据不足数据项长度时，右边补空格，字符数据超过数据项长度时，从右边截断多余的长度。字符型数据项可用X描述符来描述，但在某些固定位置上总是遇到字母或数据字符也可用A和9描述符描述

举例：

|  |  |  |
| --- | --- | --- |
| **描述** | **字符数据** | **内存内容** |
| **02 E1 PIC X(6).** | **PAGE** | **PAGE＿＿** |
| **02 E2 PIC X(4).** | **4054** | **4054** |
| **02 E3 PIC X(9).** | **IBM/PC-XT** | **IBM/PC-XT** |
| **02 E4 PIC X(8).** | **COBOL-74** | **COBOL-74** |
| **02 E5 PIC A(5)X9(2).** | **COBOL-74** | **COBOL-74** |

## 赋初值子句（VALUE子句）

基本语法：



对数据项赋以初值可以使用VALUE子句，注意，只有对工作单元节中的数据项可以赋初值，不能对文件节中的输入输出文件的数据项赋值。

如果在组合项的描述体中使用VALUE子句，初值只能是表意常量或非数值型常量，对于组合项整体而言，一律按字符型数据项处理。

VALUE子句给出的值应适合PIC子句描述的范围，否则会出现截断或产生错误。当用一个带符号的数据做初值时，相应的PIC子句中应该有"S"描述符，否则符号无效。赋值时应注意类型的一致性，表意常量中只有ZERO和ZEROS即可作为非数值常量，又可作为数值常量使用。

举例：

|  |
| --- |
| 77 A PIC 99 VALUE IS 0.  77 B PIC XX VALUE SPACE.  01 T.  02 T1 PIC S99V99 VALUE 1.2.  02 T2 PIC S99 VALUE -21.  02 T3 PIC 9(3) VALUE ZERO.  02 T4 PIC X(4) VALUE ZEROS. |

## 认识过程部

### 了解过程部

过程部是COBOL程序的核心部分，指出数据处理的步骤，即决定计算机应做什么操作 。

　 部首以PROCEDURE DIVISION开头，由各种结构组成，从而构成COBOL程序的程序逻辑。过程部这些结构包括语句、段和节，过程部下面分若干节，节下面分段，段由若干句子组成，句子由若干语句组成，以句号“.”和空格结束。过程部简单格式可以不设节，而直接由段组成，甚至可以不设段而直接由句子组成。句子可以只含一条语句。过程部复杂格式用于过程部需要DECLARATIVES部分时要把段组合成节。

节名和段名是过程名的一种形式，是标示一组组语句的用户定义组。其他COBOL语句可以按这个名称引用段，如GO TO语句的目标或由PERFORM语句作为子过程执行。

过程部部首、节名、段名应从A区开始书写，过程部的语句一律从B区开始书写，一个语句可以在一行或多行上 。

过程部的语句都是以动词开头的单词、直接数和分隔符组合。动词是标识语句的COBOL保留字。动词，如MOVE,OPEN,READ,WRITE等，它表示计算机应执行的操作。语句中的动词后面一般要跟以一个操作的对象，操作对象可以是数据名、文件名或过程名。语句应从一行的B区开始，语句可以和段名放在同一行，但最好将语句另起一行。

### 接收语句(ACCEPT语句)

基本语法：



说明：

ACCEPT语句要求从外界输入数据，如不写FROM部分，表示从系统隐含指定设备上接收一个数据给数据项。

标识符指的是能唯一的标识一个数据项的数据名。只能接收同数据名定义的长度类型相一致的数据。

ACCEPT后面只能跟一个标识符，不能出现两个或两个以上标识符，但可以用组合项。

几种常用的形式：

|  |  |
| --- | --- |
| ACCEPT 标示符 FORM DATE | DATE 表示为6个字符的数字基本项目，指出当前日期。年年月月日日 |
| ACCEPT 标示符 FORM DAY | DAY表示为5个字符的数字基本项目，前两位是年份，后三位为当年的天数 |
| ACCEPT 标示符 FORM DAY-OF-WEEK | DAY-OF-WEEK标示当前星期几，1表示星期一，7表示星期日 |
| ACCEPT 标示符 FORM TIME | TIME——8位时间 时时|分分|秒秒|百分秒 |
|  |  |

例

|  |  |
| --- | --- |
|  | \* COBOL sample  IDENTIFICATION DIVISION.  PROGRAM-ID. COBOLMAIN.  AUTHOR. SSB-RL.  REMARKS. CH2.  \*---------------------------------------  ENVIRONMENT DIVISION.  CONFIGURATION SECTION.  SOURCE-COMPUTER. IBM-PC.  OBJECT-COMPUTER. IBM-PC.  \*---------------------------------------  DATA DIVISION.  WORKING-STORAGE SECTION.  77 NAME-IN PIC A(6).  77 BIRTH-IN PIC 9(6).  77 ADDR-IN PIC X(20).  \*---------------------------------------  PROCEDURE DIVISION.  000-BEGIN-RTN.  DISPLAY " " LINE 1 POSITION 1 ERASE.  DISPLAY "NAME" LINE 5 POSITION 5.  ACCEPT NAME-IN LINE 5 POSITION 17.  DISPLAY "BIRTH" LINE 8 POSITION 5.  ACCEPT BIRTH-IN LINE 8 POSITION 17.  DISPLAY "address" LINE 11 POSITION 5.  ACCEPT ADDR-IN LINE 11 POSITION 17.  DISPLAY NAME-IN LINE 15 POSITION 6.  DISPLAY BIRTH-IN LINE 15 POSITION 10.  DISPLAY ADDR-IN LINE 15 POSITION 20.  STOP RUN.  END PROGRAM COBOLMAIN. |

例，从控制台接收数据，输入A，A1=12，A2=AB，A3=56，并显示A1

|  |  |
| --- | --- |
|  | IDENTIFICATION DIVISION.  PROGRAM-ID. COBOLSAMPLE.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*程序ID ：COBOLSAMPLE  \*程序名 ：演示程序  \*处理概要 ：演示ACCEPT  \*  \*变量 IO 变量名  \*  \*日期 作成者 概要  \*2007/03/18 孙树斌 作成  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ENVIRONMENT DIVISION.  CONFIGURATION SECTION.  SPECIAL-NAMES.  CONSOLE IS ABC.  DATA DIVISION.  WORKING-STORAGE SECTION.  01 A.  02 A1 PIC 99.  02 A2 PIC 9(6).  02 A3 PIC 9(5).  02 A4 PIC 9.  02 A5 PIC 9(8).  PROCEDURE DIVISION.  MAIN-PROCESS SECTION.  ACCEPT A1 FROM ABC.  DISPLAY A1 LINE 5 COLUMN 5.  ACCEPT A2 FROM DATE.  DISPLAY A2 LINE 7 COLUMN 5.  ACCEPT A3 FROM DAY.  DISPLAY A3 LINE 9 COLUMN 5.  ACCEPT A4 FROM DAY-OF-WEEK.  DISPLAY A4 LINE 11 COLUMN 5.  ACCEPT A5 FROM TIME.  DISPLAY A5 LINE 13 COLUMN 5.  STOP RUN.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  END PROGRAM COBOLSAMPLE.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

### 显示语句(DISPLAY语句)

基本语法：



说明：

如果使用WITH NOADVANCING 则不自动换行，否则每执行一个DISPLAY语句，总是从一个新行开始显示的。显示项目可以是标示符，常量，字符串，如果没有UPON可选项，则在计算机隐含的指定输出设备上显示指定数据项。如果常量为表意常量，则按一个字符输出。

当应用DISPLAY语句在一行中输出几个数据项的值，两个相邻数据之间应插入输出几个空格字符，以把数据隔开，通常对输出的数据给予文字说明。

输出数据时是按数据项在内存中实际内容显示的。

在运行正式程序时，一般不用ACCEPT和DISPLAY，以提高计算机效率，减少程序员的干预

假设组合项A，下分A1=12，A2=34，A3=56

|  |  |
| --- | --- |
| 显示语句 | 显示结果 |
| DISPLAY A1. | 12 |
| DISPLAY A1，A2，A3. | 123456 |
| DISPLAY “A=”，A. | A=123456 |
| DISPLAY “MY AGE IS”，A2. | MY AGE IS 34 |
| DISPLAY ZERO. | 0 |
| 77 X PIC 9(4) VALUE 17.  DISPLAY X. | 0017 |

例：

|  |  |
| --- | --- |
|  | IDENTIFICATION DIVISION.  PROGRAM-ID. COBOLMAIN.  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  77 LINE-OUT PIC X(78) VALUE ALL "-".  01 INPUT-RE.  03 NO-IN PIC 9(5).  03 NAME-IN PIC X(20).  03 CITY-IN PIC X(20).  03 ADDRESS-IN PIC X(20).  01 OUTPUT-RE.  03 FILLER PIC X(2) VALUE "|".  03 NO-OUT PIC 9(5).  03 FILLER PIC X(3) VALUE "|".  03 NAME-OUT PIC X(20).  03 FILLER PIC X(3) VALUE "|".  03 CITY-OUT PIC X(20).  03 FILLER PIC X(3) VALUE "|".  03 ADDRESS-OUT PIC X(20).  03 FILLER PIC X(2) VALUE "|".  PROCEDURE DIVISION.  BEGIN.  DISPLAY "CUSTOMER LIST".  ACCEPT INPUT-RE.  MOVE NO-IN TO NO-OUT.  MOVE NAME-IN TO NAME-OUT.  MOVE CITY-IN TO CITY-OUT.  MOVE ADDRESS-IN TO ADDRESS-OUT.  DISPLAY LINE-OUT.  DISPLAY OUTPUT-RE.  DISPLAY LINE-OUT.  DISPLAY "END LIST".  STOP 6.  END PROGRAM COBOLMAIN. |

### 数据项初始化（INITIALIZE）

基本语法：



说明：对用于计算的数据项进行初始化，即置零，字符型就是空。

例：如A=1234，

INITIALIZE A ， B.

那么A所在内存0000.

### 传送语句(MOVE语句)

语法：



传送语句MOVE将数据从发送项目复制到另一个或几个接收项目中。

数值型数据传送，按小数点对齐

字母或字符型数据传送，按左对齐

MOVE语句用来实现数据的传送，将一个数据从一个内存域送到另一个内存域中。可以将常量（包括数值常量、非数值常量、表意常量）或一数据项的内容传送给另一数据项。

例：如果A=3456，B=1234，

MOVE A TO B.

则A=3456,B=3456.

传送语句(MOVE语句)的传送原则：

如果接收项和发送项在数据部中描述的类型和长度相同，则按字节一一对应的传送

如果接收项和发送项长度不相同，而两者都是数值数据项，则按“小数点对齐”原则处理

对字母或字符数据（非数值型数据）的传送，按“左对齐”原则处理

MOVE语句可以将一初等项内容传送给另一初等数据项，也可以将一组合项内容传送给一初等项，也可以将一初等项内容传送给一组合项.

非法的传送 ：

数值编辑项、字符编辑项、SPACE、字母数据项不能传送给数值数据项或数值编辑项

数值常量、ZERO，数值数据项、数值编辑项不能传送给字母数据项

非整数的数值数据项或数值常量不能传送给字符数据项或字符编辑数据项

合法的传送 ：

接收项为字符数据项或字符编辑项，而发送项的长度大于接收项的，按“对齐原则”，超过部分截断，如果长度小于接收项的，多余位置补空格。

接收项是数值项或数值编辑项（初等项），可以接受数值型的数据以及内容全为数字的字符型数据。

接收项是字母型，按左对齐原则接受字母字符，多余位置补空格。但它不能接受非字母的字符。

组合项的传送：

组合项的传送是将发送项的内容不加转换的一个字节一个字节的顺序传送到接收项

发送项和接收项都是组合项，而且其结构和描述均相同，则可看作将各初等项一一对应传递

如果发送项与接收项长度相同，但数据形式不同，则将发送项的内容原样不变的自左而右顺序的传送到接收项

数据名的受限和受限名的传送：

用上属层的数据名来对下属层的数据名加以“限定”，使其成为唯一，这种方法，称为数据名的受限

数据名和限定符之间用OF 或IN 来连接，但通常只使用OF。

如果限定一次不能成为唯一，则可以限定多次，数据名在受限后构成一个整体称为受限名

举例：A1 OF A

### 停止语句(STOP语句)

基本语法：



其中：

当用STOP RUN时，执行此语句将程序停止运行，停止后不能再接着向下运行   
用STOP 常量 ， 表示程序暂时挂起不往下执行，显示出此常量

## 算术运算语句

一个语句只能进行一种单一的运算，不能在一个语句中实现两种不同的运算,加法和减法可以进行两个以上数值量的计算,四种算术运算，都有两种形式，即带GIVING和不带GIVING的.

### 加法语句(ADD语句)

基本语法一：



基本语法语法二：



几种应用形式

|  |  |
| --- | --- |
| ADD A TO B. | A + B => B |
| ADD A , B TO C. | A + B + C => C |
| ADD A , B GIVING C. | A + B => C |
| ADD A , B TO C , D | A + B + C => C , A + B + D => D |

其中：

在TO 和GIVING 后面只能跟数据名，而不能跟常量

TO 前后的数据名的次序不要随便改换

GIVING的后面可以跟几个数据名

参加运算的只能是数值量，它们的长度不应超过18位数字

例：

|  |  |
| --- | --- |
|  | \* A=A+B+C  IDENTIFICATION DIVISION.  PROGRAM-ID. COBOLMAIN.  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  77 A1 PIC 999.  77 A2 PIC 999.  77 A3 PIC 999.  77 A PIC 9999.  PROCEDURE DIVISION.  BEGIN.  INITIALIZE A.  DISPLAY "A1=".  ACCEPT A1.  DISPLAY "A2=".  ACCEPT A2.  DISPLAY "A3=".  ACCEPT A3.  ADD A1 TO A.  ADD A2 TO A.  ADD A3 TO A.  DISPLAY "A=" , A1 , "+" , A2 , "+" , A3 , "=" , A.  STOP 1.  END PROGRAM COBOLMAIN. |

### 减法语句(SUBTRACT语句)

语法一：



几种应用形式

|  |  |
| --- | --- |
| SUBTRACT B FROM A | A - B => A |
| SUBTRACT B , C FROM A | A - B - C => A |
| SUBTRACT B , C FROM A , T | A - B – C => A  T – B – C => T |
| SUBTRACT B , C FROM A GIVING T | A - B - C => T |

其中：

GIVING后面不能跟常量

如不带GIVING部分，则FROM后面也不能跟常量

### 乘法语句(MULTIPLY语句)

基本语法：



几种应用形式

|  |  |
| --- | --- |
| MULTIPLY A BY B | A \* B =>B |
| MULTIPLY A BY B GIVING C | A \* B => C |
| MULTIPLY A BY B , C | A \* B => B , A \* C => C |

其中：

不带GIVING部分时，BY后面不能跟常量

带GIVING部分时，BY后面可以是常数，而GIVING后面不能是常量

存放值的项只能是数据名，不能是常量

### 除法语句(DIVIDE语句)

基本语法：



几种应用形式

|  |  |
| --- | --- |
| DIVIDE A INTO B | B / A => B |
| DIVIDE A INTO B GIVING C | B / A => C |
| DIVIDE A BY B GIVING C | A / B => C |

其中：

当用GIVING部分时，第一个运算量可以除第二个运算量，此时用INTO

允许GIVING后有几个标识符

如除不尽则多余的位数被截去

### 计算语句(COMPUTE语句)

基本语法：



其中：

算术表达式是由算术初等量（数值常量、数值型数据项），算术运算符(+ 、-、\*、\*\*、/等)，括号，所组成的有意义的式子。所有运算符两侧均应留一空格 ，括号的外侧应留空格，内侧不要留空格

表达式优先级（依次递减）：

括号（），单目运算符正负号，乘方\*\*，乘除，加减

## 逻辑与控制语句

### 条件表达式

关系运算符

|  |  |  |  |
| --- | --- | --- | --- |
| COBOL关系运算符 | 简写 | 意义 | 相当数学上的符号 |
| IS GREATER THAN | > | 大于 | > |
| IS LESS THAN | < | 小于 | < |
| IS EQUAL TO | = | 等于 | = |
| NOT GREATER THAN | NOT > | 不大于 | ≦ |
| NOT LESS THAN | NOT < | 不小于 | ≥ |
| NOT EQUAL TO | NOT = | 不等于 | ≠ |

比较原则：

数值型数据项比较，按代数值进行，不考虑其长度

字母型数据项比较，从各自左起第一个字母开始，逐个进行。两个不等长数据项相比，应将较短的数据项，用空格填充，空格比任何字母都小

字符型数据比较，按其ASCII码值大小相比

### 条件语句(IF语句)

语法格式一：



处理

处理

处理

条件表达式

真

假

**例：编写程序，输入3个数，按从大到小排序显示。**

### IF嵌套与分析

|  |
| --- |
| IF……  IF……  ELSE……  END-IF  ELSE……  END-IF. |

或

|  |
| --- |
| IF ……  ELSE ……  IF ……  ELSE ……  END-IF  END-IF. |

1.IF与ELSE配对，从前向后检查各语句，遇到ELSE后，应从前面的语句中查找最近的IF，用以组成一条语句。

2.当嵌套的IF语句第一部分（或第二部分）不再为IF语句时，转到IF语句的下一个语句。语句系列中含有GOTO语句除外。

3.在嵌套的选择结构中，只能在最外层的选择结构中使用一个句点

### 转移语句(GOTO语句)

基本语法：



说明：这是一个无条件转移语句，使用不当会破坏结构化程序的结构。若用GOTO语句从一个模块中间跳到另一模块入口或中间是不允许的。

多分枝GO TO语句：



说明：可以控制程序转移到不同的过程名，当ON后面的标识符值为1时，则转移到过程名1，值为2，转移到过程名2，类推，标识符必须是整数型数据项，超出范围，程序不转移，继续执行下一句。

例：

**例：编写程序，要求从键盘上键入若干学生每门课的成绩，计算学生总成绩及平均成绩，列出最高分和最低分。**

|  |  |
| --- | --- |
|  | IDENTIFICATION DIVISION.  PROGRAM-ID. COBOLSAMPLE.  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  77 TOTAL PIC 9(5) VALUE 0.  77 I PIC 99.  77 SCORE PIC 9(3).  77 TOTAL1 PIC 9(4).99.  77 AVERAGE PIC Z(3).99.  PROCEDURE DIVISION.  MAIN-PROCESS SECTION.  DISPLAY "INPUT SCORE 999 I9S THE END."  ACCEPT SCORE.  IF SCORE = 999 GO TO END-OUT.  ADD SCORE TO TOTAL.  ADD 1 TO I.  GO TO MAIN-PROCESS.  END-OUT SECTION.  DIVIDE TOTAL BY I GIVING AVERAGE.  MOVE TOTAL TO TOTAL1.  DISPLAY "THE NUMBER OF STUDENTS IS " .  DISPLAY "THE TOTAL IS ", TOTAL1 .  DISPLAY "THE AVERAGE IS ", AVERAGE .  STOP 0.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  END PROGRAM COBOLSAMPLE.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

## 执行语句（PERFORM 语句）

### 概述

在一个COBOL程序中过程部中往往有一部分语句是需要多次执行的，这时需要使用执行语句。

本身并不直接进行数据处理，但是COBOL语言中最强有力的控制语句。

可以减少内容相同的程序段重复编写，有多种循环功能，可以使程序结构模块化，从而使程序设计易于分工、调试和维护，同时程序层次分明，易于阅读。

由PERFORM调用的程序段通常总是放在STOP RUN语句之后。

模块化设计就是把各个功能段写成彼此独立的模块，在建立一个主模块，由主模块调用各功能模块。

### 基本形

执行语句最基本的形式：



过程部中的节名、段名称为过程名。用PERFORM语句只能转到指定的节或段的开头，执行完该节（段）的全部语句后返回。

需要执行的不是一个段（节），而是若干个段（节），则应指明从哪一段（节）起到哪一段（节）止。则PERFORM语句的一般格式为



THROUGH可以缩写成THRU

举例：PERFORM A THRUB。

### PERFORM 语句与其他语句的区别

* 与子程序比较

二者都具有调用功能。

PERFORM 语句被调用的模块与调用模块在同一个文件中，可以直接使用数据，而子程序模块写在另一个文件中，并且子程序中四个部完备，调用子程序有时需要定义数据联系。

* 与GOTO比较

二者都具有转移功能。

PERFORM 语句有返回功能，而GOTO则一去不复返。

### PERFORM 语句的嵌套

嵌套不能交叉，后一个被“调用”的语句序列或者全部套在前一个被“调用”的语句序列之中，或者全部在它之外是正确的。不允许二者交叉或者有共同的终点。具体是否允许语句序列交叉，查看系统手册。

### PERFORM语句序列中的转移

在PERFORM语句中所执行的语句序列中，可以含有转移语句，可以使流程转到语句序列之外，但一般应转回到此语句序列，以便最后能执行此语句序列的最后一个句子。

### PERFORM 语句语句计数循环

基本形式：



说明：

（1）标识符应为整数数据型。

（2）如果此标识符的值在执行语句序列中有变化，不会影响执行的次数。即以它开始时的值决定执行的次数

例如：PERFORM T1 THRU T5 N TIMES.

例：求1+2+3+……+100的和。

过程部如下：

|  |  |
| --- | --- |
| 000001  000002  000003  000004  000005  000006  000007  000008  000009  000010  000011  000012 | PROCEDURE DIVISION.  MOVE 0 TO S.  MOVE 0 TO I.  PERFORM A 100 TIMES.  DISPLAY ‘S=‘ S.  STOP RUN.  A.  ADD 1 TO I.  ADD I TO S. |

### PERFORM 语句UNTIL循环

基本语法：



作用：反复执行指定的语句序列 ，直到给定的条件满足为止。

注意：

当型循环：在执行PERFORM语句时，先判断指定的“条件”是否满足，若满足则执行，如果一开始“条件”不满足，则一次也不执行。

直到型循环：在执行PERFORM语句时，先判断指定的“条件”是否满足，不满足则执行，满足则退出循环，至少执行一次。

### 出口语句（EXIT）

语法：



EXIT提供了一组公共过程的公共出口，或者说提供了被调用过程的逻辑终点。

EXIT必须是该段的唯一一条语句。

## 复写语句

COBOL程序中的DATA DIVISION部分往往是很长的，因为其中包含许多的数据项的描述。可以利用COPY语句使某些记录描述和数据描述为不同的程序共用。为此要建立一个“源程序库”，将上述这些共同使用的程序中的某一部分事先存入库中。

语法：



“字”的含义是不超过30个字符组成的字符序列。它包括数据名、条件名、过程名、表意常量、助忆名或保留字等。复写语句可以用在除标识部以外的其他三个部中，但在环境部中使用复写语句的意义不大。使用复写语句的目的是有利于在不同文件中用标准化的记录书写程序，减少写程序的工作量，便与程序的保存和修改。

## 顺序文件操作

### 顺序文件的定义

### 顺序文件的读取

### 顺序文件的写出

# 数据处理

## 知识要点

**Cobol数据的定义，使用方法**

## 数据在计算机内的表示形式

### 计算机内存的组织形式

计算机是以二进制形式存放数据的，内存的最小单位是二进制位。8位构成一个字节

数据在内存中的存储形式

不论是字母还是数字，都按一个字节存放一个字符存放

### 数值型数据在内存中的存储数据

* 外部十进制：一个数字在内存中占一个字节
* 外部浮点数形式：

一般形式： 数符 数值部分 E 阶码符 阶码

例： 77 A PIC +9.99999E+99 表示 +1.23456E+59

注意：外部符点形式的数据不能用VALUE语句赋初值。

* 内部十进制数：只能存放0—9十个数字，每两个数字占一个字节。
* 定点二进制形式：将十进制数转换成定点二进制数，存入内存。

### 数据描述与存储形式的关系

字母型、字符型、编辑型、外部十进制数据和以外部浮点形式表示的数据用标准数据形式来存放

数值型数据可以由程序员任意选定存放形式。

数据在计算机内进行运算，都是化成二进制数以后再进行的。

## 用法子句（USAGE子句）

用法 (USAGE) 子句可以使程序设计者自由选择数据在内存中的存放形式。

语法：



USAGE IS可以省写，默认为DISPLAY

“USAGE IS DISPLAY”的意思是“显示型的用法”，即此数据适合打印、显示，它采用标准数据类型。USAGE IS可以省写

BINARY、COMPUTATIONAL、PACKED-DECIMAL只能用于数字数据项。

“USAGE IS BINARY”的意思是指定项目存放成二进制数。

COMPUTATIONAL和COMP是同一意思，表示“计算型的用法”表示此数据类型适合计算，它采用适用于计算用的定点二进制形式或内部浮点形式。由厂家定义。

COMP-1 表示以内部短浮点形式

COMP-2 表示以内部长浮点形式

COMP-3表示以内部十进制形式存放

PACKED-DECIMAL指定数字按10进制存放，

INDEX 指定该项目作为表格索引。

说明：

（1）USAGE子句是用来指定数据项在内存中的存储形式的；

（2）如果省略USAGE子句，则隐含表示用DISPLAY形式；

（3）如果对组合项描述为一种存储形式，则表示这个组合项的下属各初等项都是这种形式；

（4）USAGE子句指定的数据存储形式不应与PIC子句指定的数据类型矛盾；

（5）长、短浮点形式已确定内存长度，不应再用PIC子句；

（6）在传送或运算时不同存储形式的数据型数据间可互相转换；

（7）在用DISPLAY语句显示数据项的内容时，如果数据项的USAGE“用法中”不指定DISPLAY，则在显示前，计算机自动将内存中的数据形式转换（EBCDIC码或ASCII码，视计算机系统而定），然后以字符形式显示；

（8）如果用WRITE语句，则直接输出，不进行转换。

举例：

77 A PIC 9（2）USAGE IS COMP.

77 A PIC 9（2）COMP.

表示数值数据项A用定点二进制形式存放数据，这种存放形式便于计算，不便于显示，故称为计算型。

77 A PIC 9（6）USAGE IS DISPLAY.

77 A PIC 9（6） DISPLAY.

77 A PIC 9（6）.

显示型用法，即此数据项适宜于显示、打印，它采用标准数据形式（一个字节放一个字符）存放。子母型、字符型、编辑型、外部十进制数据必须指明为显示型。

01 T.

03 T1 USAGE COMP.

05 X PIC S9（3）.

05 Y PIC S9（3）.

用于对组合项的描述，表示该组合项的下属各初等项都以同一种形式存放。

## 重定义子句(REDEFINES子句)

REDEFINES可以用不同的数据描述共用内存中同一段内存空间。 如果在时间上并不需要同时占用内存空间，那么这样就节省了内存。适用于基本数据项和组合数据项。

语法：



举例:

01 A.

02 A1 PIC X(6).

02 B1 REDEFINES A1.

03 B11 PIC X(4).

03 B12 PIC 99.

02 C1 REDEFINES A1 PIC 9(6).

说明：

（1）数据名2与数据名1的层号必须相同。即它们应是同一层次的。REDEFINES子句不能用于88层和66层。

（2）用REDEFINES子句的描述体应紧跟在被重新定义的数据项的描述之后，中间不能插入其他项的描述说明。

（3）可以多次重定义，但必须紧跟出现，而且要求使用最初定义的数据名。

（4）REDEFINES子句不能用于文件节的01层中，因为文件节中01层描述的是记录，但工作单元节中的01层是可以用REDEFINES子句重新定义的，因为这里的01层不是值输入输出文件的记录，而是指组合项。

（5）用REDEFINES子句**可以改变数据的结构**，但两个数据名的长度应相同。

（6）REDEFINES子句应在其它子句之前。

（7）内存中的值为数据名1和数据名2共享。也就是说，重定义后两个数据名的名称和两种数据结构同时存在，都有效。程序中可使用其中任何一个。他们在内存中为同一段存储单元。如果改变了内存内容，则二者的值都因而改变。

（8）重定义子句所在的数据描述体中不能使用初值子句赋初值。

不适用REDEFINES子句时的情况：

数据项长度不一致

使用REDEFINES子句的数据项有VALUE子句

文件节和报表

原数据项含有OCCURS子句

## 重命名子句(RENAMES子句)

用REDEFINES子句可以在不改变数据项的长度的前提下，重新定义数据区的名称和数据结构的形式（包括重新定义初等项的类型和长度）。用重命名（RENAMES）子句可以把原来已定义的某些数据项重新组合成一个新项，并以一个新名字来代表它。但重命名子句不能改变原来初等项的类型、长度和属性。

语法：



举例：

|  |  |
| --- | --- |
|  | 01 A.  02 B.  03 G PIC X(2).  03 H PIC 99.  02 C  03 I PIC 99V99.  03 J PIC S999.  02 D PIC 9(5).  02 E PIC XX.  66 M RENAMES B THRU C.  66 N RENAMES E. |

说明：

（1）层号只能用66，它必须紧跟在01层记录中最后一个数据描述体之后，因为它是对记录中有关部分重新组合和命名的。

（2）如无THRU部分，则数据名1和数据名2代表的是同一内容。

（3）用THRU时，数据名2在记录中的位置应在数据名3之前，而且数据名3不应包括在数据名2之中。

（4）RENAMES子句只能用在工作单元节中，不能用于文件节中。

## 遇零置空子句（BLANK WHEN ZERO）

BLANK子句的作用是：当数据项的值为零时，使它的内容改变为空白。这个子句只能用于数值型或编辑数值型的初等项。可以将格式字符串通常不允许为空的项目变为空。

例： 03 A PIC $(5).99 BLANK WHEN ZERO.

## 对齐子句(JUSTIFIED子句)

JUSTIFIED子句的一般形式：



说明：JUSTIFIED子句只能用于非编辑字母型和字符型数据，而不能用于数值型数据项和编辑型数值项；因为后者是按小数点位置对齐的方式定位。

数据放入这个项目时，与最右边的字符对齐，数据长出时左边截断，否则左边填充，如果不用，则标准规则为与最左边字符对齐。

举例 77 B PIC X(5) JUST RIGHT.

如B=“XYZ”，则B内容为XYZ

## 同步安置子句（SYNCHRONIZED）

语法：



SYNCHRONIZED （简写SYNC）指定基本数据项在计算机内存自然边界上的对齐，不能用于组合项。即项目再要分配给该数据项的自然边界中对齐，使其他数据项不会占用未用空间。保证项目放在内存边界上，从而最有效的处理。

|  |  |
| --- | --- |
|  | 01 A.  02 A1 PIC 999 SYNC LEFT VALUE 88.  02 A2 PIC XXX SYNC RIVHT VALUE ‘ABC’. |
|  |  |

## 编辑型数据描述

对数据进行编辑，使其满足数据输出的要求,不能用于计算。

分为编辑数值型和编辑字符型

编辑数值型用来对需要显示、打印的数值数据进行编辑加工，不能参加算数运算。一般来说，数值型数据项不能或不宜直接用来显示或打印输出，而必须通过编辑数值型数据项来输出。包括如下描述符：9 $ Z . , \* B 0 + - OR DB

编辑字符型用来对字符数据进行编辑加工。包括如下描述符：A X 9 B 0 编辑描述符从功能上来说可分为替换性和插入性。

### “.”描述符

作用：表示插入小数点的位置

举例：

77 A PIC 9(3)V9.

77 B PIC 9(3).9.

MOVE A TO B.

说明：若A值为148.5，A在内存中占4位，B在内存中占5位（包括小数点），将一个数值型数据项传送给带有“.”描述项数据项，传送按小数点对齐，整数部分向左传送，小数点部分向右传送，多余的数字被截去。

### “+”描述符

作用：在正数前加“+”，负数前加“-”

举例：

77 A PIC S9(3)V9.

77 B PIC +9(3).9.

MOVE A TO B.

说明：若A值为+128.6，B在内存中值为+128.6，占位6位

若A值为-128.6，B在内存中值为-128.6，占位6位

### “-”描述符

作用：在正数前加一个空格，负数前加一个“-”

举例：

77 A PIC S9(3)V9.

77 B PIC -9(3)V9.

MOVE A TO .

说明：若A值为+128.6时，B的值为 128.6，若A值为-128.6时，B的值为-128.6

浮动加入“+”或“-”使用浮动描述，使“+”、“-”可以加在数字前适当的位置上，必须有足够的“浮动范围”。

|  |  |  |  |
| --- | --- | --- | --- |
| A描述 | 赋给A值 | B描述 | B的值 |
| 9(4)V9 | 1248.5 | +(5).9 | +1248.5 |
| 9(4)V9 | 1248.5 | -(5).9 | 1248.5 |
| 9(4)V9 | 1248.5 | -(4).9 | 248.5 |
| 9(4)V9 | 2.5 | +(5).9 | +2.5 |
| 9(4)V9 | 0.5 | +(4)9.9 | +0.5 |
| 9(4)V9 | 0.5 | +(5).9 | +.5 |
| 9(4)V9 | 0.5 | +(5).+ | +.5 |
| S9(4)V9 | -3.2 | -(5).- | -3.2 |
| 9(4)V9 | 0 | +(5).+ |  |

### “，”描述符

插入逗号“，”做分位符

作用：可以使输出数位较多的数字便于阅读

举例：

03 A PIC 9(8)V99  
03 B PIC Z(2),Z(3),Z(3).99.  
MOVE A TO B.

如A的值为03687659.28时，则B内容为3，687，659.28

如果“，”描述符前接收的均为高位无意义零并且置为空格时，对应字节也置为空格，也可以和浮动“+”或“-”描述联用。

### “$”描述符

作用：对于美元金额的输出数据，常常希望在数据前面加一货币符号“$”,此时可以使用“$”描述符。

举例：

02 SUM PIC S99(3)V99.  
02 SUM-O PIC $9(3).99.  
MOVE SUM TO SUM-O.

当SUM的值为359.25时，SUM-O的值为$359.25，当SUM的值为25.25时，SUM-O的值为$ 25.25

为了取消高位无意义数字零，可以使用“$”浮动描述形式，如$(4).99，当使用浮动描述时，前面不能出现其他描述符，如要加正负号，可以将“+”或“-”加在后面，如$(4).99-，但输出数据时，符号也是打印在最后。

### “\*”描述符

作用：将高位无意义数字零代以“\*” 常用于财务处理。

举例：

02 SUM PIC 9(4)V99.  
02 SUM-O PIC \*(4).\*\*.  
MOVE SUM TO SUM-O.

当SUM的值为38.50时，SUM-O的值为\*\*38.50，当SUM的值为0时，SUM-O的值为\*\*\*\*.\*\*

### 插入空格、“0”

每个“B”或“0”描述符占一个字节，该字节得内容固定为空格或数字零，因此使用“B”或“0”可以在数据项的前后或中间加一些空格或数字零。

举例：

03 A PIC 9(3)V9.  
03 B PIC BBZ(3).9B  
03 C PIC Z(3)000  
MOVE A TO B C.

如A取值为128.6,则B内容为 128.6 ,C为128000

“B”和“0”还可以和“A”，“X”联合使用描述字符型编辑数据项，如：ABABA，X0XX等。

### “DB”、“CR”

在银行账目中，有时用“DB”代表借方（credit），而用“CR”表示贷方（credit）紧接在数据之后，这时我们可以使用“DB”和“CR”描述符。

03 SUM PIC S9(3)V99.  
03 SUM-O PIC $(4).99CR.  
MOVE SUM TO SUM-O.

当SUM值为-239.58时，SUM-O内容为$239.58CR,“DB”和“CR”占两个字节位置，“DB”和“CR”对正数编辑结果为两个空格，对负数编辑结果为“DB”或“CR”。

## 描述符知识点整理

在实际开发中，需要对数字用特定的形式输出。（比如：$10，000）

|  |  |
| --- | --- |
| **数值数据项** | **9VSP** |
| **字母数据项** | **A** |
| **字符数据项** | **9AX** |
| **编辑数值数据项** | **9PV.,BZ+-$\*0 CR DB** |
| **编辑字符数据项** | **AX9B0** |
|  |  |

描述符含义一览

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 9 | 表示一个数字位置 |  |
| A | 表示一个字母位置 |  |
| X | 字符位置 |  |
| V | 隐含的小数点 |  |
| S | 数值符号 |  |
| P | 比例换算 |  |
| $ | 货币符号位置 | 02  A  PIC   $9999.  $0045  02  A  PIC   $9(4). |
| . | 小数点 | 02 A PIC 99.99.  02 B PIC 9(2).9(2).  56.82 |
| ， | 逗号 | 02  A  PIC   9,999.  1,000  02  B  PIC   9,9(3). |
| + | 一律加符号 | 02  A  PIC   +999.  +034 |
| - | 正数加空格，负数加负号 | 02  B  PIC   -999.  -044 |
| Z | 取消高位零，加空格 | 02  A  PIC   ZZ99  48 |
| \* | 取消高位零，加\* | 02  A  PIC   \*\*99  \*\*48 |
| B | 插入空格位置 | 02  A PIC  B999B. à  \_123\_  02  B PIC  B9(3)B. |
| 0 | 插入零位置 | 02  A  PIC   9000. à 7000  02  B  PIC   9(3). |
| DB | 借方，数据为负时，在数据后面出现DB，数据为正时后面加两个空格 |  |
| CR | 贷方 |  |
| 浮动 |  | 02  A  PIC   +++99.  à +78 ,  +789 ,  +6789  02  A  PIC   $(4)99. |

## 算术运算精度处理

### 四舍五入处理子句（ROUNDED子句）

语法：

作用：按照截断的第一位数值进行四舍五入进位。

举例：

ADD A , B TO C ROUNDED A + B + C => C

并在截断后的一位四舍五入

### 长度溢出处理子句（ON SIZE ERROR）

语法：



计算结果的整数部分的长度如果比结果数据项描述的整数部分长，应进行长度溢出处理。当发生溢出错误的时候，发生错误的结果不存入结果数据，未发生错误的结果进行保存，然后执行指定的操作。

注意：当ROUNDED 和ON SIZE ERROR 一起使用时，先用ROUNDED做舍入处理 ，再判断是否溢出。

应用举例：

MULTIPLY A BY B GIBING C D

ON SIZE ERROR DISPLAY‘SIZE ERROR.’

STOP RUN.

A \* B => C

其中：

在发生溢出时，错误的结果不存入结果数据项

如果有几个计算结果，未发生溢出的计算结果正常存入结果数据项，只有发生溢出的结果不存入

### 除法语句的余数子句(REMAINDER)

语法：



说明：商和余数的值取决于被除数、除数、数据描述，比如

77 A PIC 9V9 VALUE 6.0。

77 B PIC 99V9 VALUE 16.3。

77 C PIC 9V99。

77 D PIC 9V99。

DIVIDE A INTO B GICING C ROUNDED REMAINDER D 。

商 2.72，余数0.04。

注意:四舍五入、长度溢出只对商有效，不检查余数

DIVIDE 3 INTO 7 GIVING C REMAINDER D

7 \* 3 => 商C=2 余数D=1

其中：

商和余数的值不仅取决于被除数和余数，还取决于数据部中对商和余数的描述

如果用ROUNDED子句，它只对商起作用 长度溢出也只检查商的值是否溢出

## 对应传送子句（或称作同名传送CORR）

语法：



作用：将两个组合项下同名的初等项分别按各对应的项进行传送。

注意：与组合项之间直接传送不同。

带有RENAMES、REDEFINES、OCCURS的数据项不能传送。

对表的元素，只能对表元素进行传送，如MOVE CORR A（1） TO B（1）。

举例：

|  |  |
| --- | --- |
|  | 01 A.  02 B PIC XX.  02 C.  03 C1 PIC XX  03 C2 PIC XX.  01 A1.  03 B PIC XX.  03 C .  05 C3 PIC XXX.  05 C4 PIC XXX.  MOVE CORR A TO A1. |

其中：

对应传送的功能是把一个组合项中的若干项传送给另一组合项中同名的项

如果两个组合项包括的项不同，则只能传送同名的项

传送的两者间必须有成对的同名数据项，而且这一对中必须至少有一个项是初等项

所谓同名，指的是他们有相同的全程受限

带有RENAMES子句或REDEFINES子句或OCCURS子句的数据项不予以传送

编辑传送 将数值型传送给编辑型



1接收项字形由X(n)变为9(n)，n必须小于等于18，否则不允许

2发送项变为USAGE DISPLAY,SING TRAILING SEPARATE再发送

3按字符串处理、发送

4发送项字形X(n)假定为9(n)再发送，n必须小于、等于18，否则不允许。

接收项长度应大于发送项，带符号的数值项传送给字符型时，符号不传送，因为字符型无符号，总之，按照接收项对齐规则进行传递（数字按小数点对齐，字符按左对齐），只要发送项内存内数据符合接收项要求就可以传送。

组合项可看成字符型传送

## 对应项间的运算（CORR）

注意：只能用在加减，且同名数据项的数据类型描述一致。

语法：

ADD CORR 标识符1 TO 标识符 [ROUNDED][ON SIZE ERROR 强制语句]

SUBTRACT CORR 标识符 FORM 标识符 [ROUNDED][ON SIZE ERROR 强制语句]

ADD CORRESPONDING|CORR 标识符1 TO 标识符2 [ROUNDED]

[；ON SIZEERROR 强制语句]

SUBTRACT CORRESPONDING|CORR 标识符1 FROM 标识符2 [ROUNDED]

[；ON SIZEERROR 强制语句]

注意：

只有加法和减法才能进行有CORR的操作，而且对应项除了名字相同外，数据类型必须是数值型。

# 字符串处理

## 知识要点

**字符串的拼接，截断和替换。**

## 字符串连接（STRING）

用途：将多个非数值型的数据项的值连接起来送到一个接收数据项中，在合并过程中可以删出某些制定的字符。

语法：



STRING标识符1，标识符2，----- DELIMITED BY SIZE

标识符3，标识符4，----- DELIMITED BY 标识符

INTO 接收数据项

[WITH POINTER 位置标识符]

[ON OVERFLOW 强制语句].

DELIMITED定界短语，用来控制各个发送项的终止位置。如果写上DELIMITED BY SIZE 表示将整个发送项的字符全部送到接收项

POINTER指针短语，逻辑指针的位置数据项的值表示从左边第几个字符开始接收，位置标识符必须为整形初等数据项。

逻辑指针值小于1或者大于接收项包含的字符个数，不能正常执行字符传送，发生溢出。

OVERFLOW 溢出短语判断是否发生溢出，以及制定当发生溢出时所作的处理。

举例：

如果遇到空格即截止可写为

STRING A，B，C DELIMITED BY SPACE INTO D。

SPACE也可用其他字符

传送中插入其他字符

STRING A，“ ”，B，“A”,C DELIMITED BY SPACE INTO D。

## 字符串分解语句（UNSTRING）

语法：



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| UNSTRING | | 标识符1 | | |  | |  | | |  |  |
|  |  | | | | { | 标识符2 | | | } |  |  |
|  | [DELIMITED BY [ALL] | | | |  |  | |  |  |
|  |  | | | | 常量2 | | |  |  |
|  |  |  |  | |  | | |  | |  |  |
|  | [ |  |  | { | 标识符3 | | | }]……] | |  |  |
|  | ，ON [ALL] | |  | | |  |  |
|  |  |  | 常量3 | | |  |  |
|  |  |  |  | |  | | |  | |  |  |
|  | INTO 标识符4 | | | | | | | | | | |
|  |  |  |  | |  | | |  | |  |  |
|  | [，DELIMITER IN 标识符5，COUNT IN 标识符6][，标识符7] | | | | | | | | | | |
|  |  |  |  | |  | | |  | |  |  |
|  | [，DELIMITER IN 标识符8，COUNT IN 标识符9][，……] | | | | | | | | | | |
|  |  |  |  | |  | | |  | |  |  |
|  | [WITH POINTER 标识符10][ TALLYING IN 标识符11] | | | | | | | | | |  |
|  |  |  |  | |  | | |  | |  |  |
|  | [：ON OVERFLOW 强制语句] | | | | | | | | |  |  |

注意：书写顺序按照语法来写，不能颠倒。

作用： 将一个发送项数据分别传送到多个接收项中，是STRING的逆操作。

用定界短语DELIMITED 作分解定界符，自左而右遇到制定的的定界符为止，定界符左侧的字符为传送内容，按传送家规则进行。定界符不被传送。

[ALL]表示定界符 长度不定的常量，比如ALL “”表示一个或多个空格作为定界符。

OR可以用多个定界符，只要满足其中一个即可。

COUTE 将已发送的字符个数记入用户定义的计数数据项

定界符存储短语DELIMITER 在传送一个字符串到接收项中时将定界符送到一个数据项中存储起来。

指针POINTER 表示希望从发送项第几个字符开始传送，只能有一个指针。

接收项计数TALLYING用来记录实际接收choansong的接收项数量。

|  |
| --- |
|  |

|  |
| --- |
|  |

## 检测语句（INSPECT语句）

语法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INSPECT 标识符1 [TALLYING 标识符 | | | | | | | | | | | | | | | | | | | | | | |  | |
|  |  |  | |  | | | |  | |  | | | |  | | | | |  | | | |  | |
|  |  | { | { | ALL | | | | }{ | | 标识符 | | | | } | | } | | |  | | | |  | |
|  | FOR | LEADING | | | | 直接量1 | | | |  | | | |  | |
|  |  | CHARACTERS | | | | | | | | | | |  | |  | | | |  | |
|  |  | | |  |  | | | | |  | | | |  | | | | |  | | | |  | |
|  | [{ | BEFORE | | | } | | INITIAL | | { | | 标识符 | | | | }]] | | | | |  | | | |  | |
|  | AFTER | | | 直接量2 | | | |  | | | |  | |
|  | [REPLACING | | | { | | { | | ALL | | | | }{ | 标示符 | | | | | } | |  | } |  | |  | |
|  | LEADING | | | | 直接量3 | | | | |  |  | |  | |
|  | FIRST | | | | | |  | | | | | | |  | |  |  | |
|  | CHARACTERS | | | | | |  | | | | | | |  | |  |  | |
|  |  |  | |  | | | |  | |  | |  | | | | | | |  | | | |  | |
|  | BY{ | 标示符 | | | | | | } | |  | |  | | | | | | |  | | | |  | |
|  | 直接量4 | | | | | |  | |  | | | | | | |  | | | |  | |
|  |  |  | |  | | | |  | |  | |  | | | | | | |  | | | |  | |
|  | [{ | BEFIRE | | | } | | | INITIAL{ | | | | 标示符 | | | | | }]] | |  | | | |  | |
|  | AFTER | | | 直接量5 | | | | |  | | | |  | |

用途：用来检测一个字符串中的一个字符出现次数，用一个指定字符代替另一个指定字符，通过指定某些字符来限制上述检查区间

BEFORE/AFTER 用来限制被检查的区域，确定检查的边界，如果找不到指定的定界符，则检查全部字符串。

比如:

检查A-B中句点前有几个空格，将空格数存入N中

INSPECT A-B TALLYING N FOR ALL ”” BEFORE “.”。

TALLYING 用来统计满足条件的字符的个数。

ALL 全部

LEADING 前面的 ，出现多次取左数第一部分的

CHARACTERS 字符

FIRST 左数第一个

|  |
| --- |
|  |

转换语句（TRANSFORM）

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TRANSFORM | | 标示符1 | | |  | | CHARACTERS | |  | |  | |
|  |  | | { | 非数值常量 | | }TO{ | | 非数值常量 | | } | |
|  | FROM | | 表意常量 | | 表意常量 | |
|  |  | | 标示符 | | 标示符 | |

|  |
| --- |
|  |

## 本章小结

# 逻辑控制

**学习提示**

**详细介绍cobol顺序、判断处理和循环三大模块**

## 逻辑控制语句

## 条件表达式

条件表达式是取值为TURE或FALSE的表达式

## IF-ELSE语句

语法：

|  |
| --- |
| IF 条件 语句组.  IF 条件  语句组1  NEXT SENTENOE  ELSE  语句组2  NEXT SENTENOE  END-IF |

IF嵌套与分析

|  |
| --- |
| IF……  IF……  ELSE……  ELSE……  IF ……  ELSE ……  IF ……  ELSE ……。 |

1.IF与ELSE配对，从前向后检查各语句，遇到ELSE后，应从前面的语句中查找距此最近的IF，用以组成一条语句。

2.当嵌套的IF语句第一部分（或第二部分）不再为IF语句时，转到IF语句的下一个语句。语句系列中含有GOTO语句除外。

## EVALUATE语句

根据一个表达式的值，从一组数据中选择要执行的语句。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EVALUATE |  | 表达式1 | [ALSO | 表达式2] |  |  |
|  |  |  |  | ANY |  |  |
|  |  | WHEN 条件 |  | 表达式1 |  |  |
|  |  |  | [NOT] | 表达式3 | THROUGH | 表达式4 |
|  |  | [ ANY | ALSO | 条件表达式2 |  |  |
|  |  |  | [NOT] | 表达式5 | THROUGH | 表达式6] |
|  |  |  | 程序块 |  |  |  |
|  |  |  |  |  |  |  |
|  |  | WHEN OTHER | 程序块 |  |  |  |
| END-EVALUATE. |  |  |  |  |  |  |

表达式1和表达式2可以用标示符、直接数、算术和条件表达式、true与false。

条件表达式1和条件表达式2可以用条件表达式、true与false

表达式3、表达式4、表达式5、表达式6为标示符、直接数、算术表达式。

## CONTINUE语句

跳出语句块

|  |  |  |  |
| --- | --- | --- | --- |
| IF | 条件 |  |  |
| THEN |  |  |  |
|  | CONTINUE |  |  |
| ELSE |  |  |  |
|  | 语句块 |  |  |
| END-IF. |  |  |  |

## GO TO

语法1：



语法2：



|  |
| --- |
|  |

## 条件关系

条件关系由关系表达式来表示

关系表达式格式



关系表达式经过关系运算后产生一逻辑值：“真”或“假”

比较原则：数值型数据项比较按代数值进行，不考虑其长度；字母型数据项比较，从各自左起第一个字母开始，逐个进行。两个不等长数据项相比，应将较短的数据项，用空格填充，空格比任何字母都小；字符型数据比较，按其ASCII码值大小相比

## 复合条件

复合条件是一个逻辑表达式，是由两个或两个以上的简单条件和逻辑运算符AND、OR、NOT所组成。

逻辑运算符优先级：先NOT，后AND，再OR。

复合条件，即逻辑表达式的运算符合下述运算顺序：

1.算术表达式

2.关系表达式

3.NOT运算

4.AND运算（同一级别自左向右运算）

5.OR运算（同一级别自左向右运算）

如果有括号，则括号最优先（自内向外）

复合条件举例

三门课程：MATH、PHY、ENG,分别用三个计数器数据项统计三门课、两门课、一门课不及格和全部及格的人数。

|  |
| --- |
| IF MATH < 60 AND PHY <60 AND ENG < 60  ADD 1 TO COUN1  ELSE IF MATH < 60 AND PHY <60 OR PHY <60  AND ENG < 60 OR MATH < 60 AND ENG < 60  ADD 1 TO COUN2  ELSE IF MATH < 60 OR PHY <60 OR ENG < 60  ADD 1 TO COUN3  ELSE ADD 1 TO COUN4. |

## 符号条件

作用检查数据项的值的符号。

语法：

数据名或算术表达式 IS [NOT] PRSITIVE 相当于X>0 X [NOT] >0

数据名或算术表达式 IS [NOT] NEGATIVE 相当于X<0 X [NOT] <0

数据名或算术表达式 IS [NOT] ZERO 相当于X=O X [NOT] =0

用途：常用来做定型的测定。

## 类型条件

语法：

标识符 IS [NOT] NUMERIC 完全由数字组成

ALPHABETIC 完全由字母和空格组成

选用NOT表示否定的意思

注意：对于数字只能用NUMERIC或NOT NUMERIC进行判断；对于字母只能ALPHABETIC或NOT ALPHABETIC进行测试，对字符型数据项两种都可以用

举例：

01 INVENTORY-RECORD.

03 P-NAME PIC A(10).

MOVE 0 TO ERR.

IF P-NAME IS NOT ALPHABETIC

DISPLAY P-NAME “INCORRECT”

MOVE 1 TO ERR.

## 条件名条件

在IF语句中的条件，以条件名为条件就称为条件名条件，可以用来对以数据项的取值或取值范围进行测试，条件名在数据部说明，并用专用层号88。条件名所对应的数据项的特定值或取值范围用VALUE说明。

定义格式举例

77 A PIC 99.

88 A1 VALUE 6.

88 A2 VALUE 10.

A1代表条件A=6，A2代表条件A=10

一个初等项只能根据某些条件取预定的几个值，或只能在一个预定的范围中取值，则这个变量称为条件变量。用来表示条件变量当前值的名字叫**条件名**

77 X(条件变量) PIC 9(6).

88 X1 VALUE 0 THRU 99.

88 X2 VALUE 100 THRU 999.

88 X3 VALUE 1000 THRU 9999.

… …

IF X1 MOVE A TO B. (在0 <= X < 100 时)

IF X2 MOVE A TO C.

IF X3 MOVE A TO D.

其中：

X1,X2,X3为条件名

条件名用层号88，紧跟在条件变量之后说明

## 循环处理

**编写一个COBOL程序，从键盘输入顾客编号、姓名、城市名、地址，从屏幕上显示出以下表格**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **NAME** | **CITY** | **ADDRESS** |
| **00001** | **LI MING** | **TIANJIN** | **BEI JING** |
|  |  |  |  |
|  |  |  |  |

**执行步骤分析：**

**1.输入执行次数；**

**2.显示标题；**

**3.执行输入段；**

**4.显示操作结束信息；**

**5.结束**

**输入段执行内容：1.输入所需数据之值；2.传送；3.显示输入的顾客数据**

求质数

|  |  |
| --- | --- |
|  | IDENTIFICATION DIVISION.  PROGRAM-ID. COBOLMAIN.  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  77 SHU PIC 999 VALUE 0.  77 CHUSHU PIC 999 VALUE 0.  77 SHANG PIC 999V99.  77 SHANG-T PIC 999.  77 T-X PIC 9 VALUE 0.  PROCEDURE DIVISION.  STAT.  DISPLAY "ZHI SHU".  MAIN.  PERFORM MAIN-P THRU MAIN-P-EXT  VARYING SHU FROM 2 BY 1 UNTIL SHU >30 .  STO.  STOP 1.  STOP RUN.  \*  MAIN-P.  INITIALIZE T-X .  PERFORM ZHENGCHU THRU ZHENGCHU-EXT  VARYING CHUSHU FROM 2 BY 1 UNTIL CHUSHU > SHU  IF T-X NOT = 1  DISPLAY SHU, " OK".  MAIN-P-EXT.  EXIT.  \*  ZHENGCHU.  IF T-X = 1 GO TO ZHENGCHU-EXT.  DIVIDE CHUSHU INTO SHU GIVING SHANG.  MOVE SHANG TO SHANG-T.  \* DISPLAY SHANG, " " , SHANG-T.  IF SHANG = SHANG-T AND CHUSHU NOT = SHU  MOVE 1 TO T-X.  ZHENGCHU-EXT.  EXIT.  END PROGRAM COBOLMAIN. |

## 带有循环控制变量的PERFORM 语句

### 一重循环

语法：



说明：循环变量标识符称为循环变量，值会按照一定规律自动变化  
初值标识符称为循环变量初值，可以为正值、负值或零  
步长标识符称为步长，可以是正值、负值但不能为零。  
注意：循环变量的值在每次循环中是自动按步长增长的。在最后的出口条件中，不一定要和循环变量有关。  
例：PERFORM T1 THRU T2 VARYING X FROM A BY B UNTIL X >99

关于循环变量

循环变量赋初值和增加步长都是自动进行的，不需要增加其他语句。

语句条件可以与循环变量有关，也可以无关，当条件与循环变量有关时，循环就具有计数循环的性质，可以自动计数，直到终止循环。

循环变量除了能用以控制循环次数外，主要在循环体内加以利用。

举例：

|  |  |
| --- | --- |
| 000001  000002  000003  000004  000005  000006  000007  000008  000009  000010  000011  000012 | IDENTIFICATION DIVISION.  PROGRAM-ID. SUM  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  77 I PIC 999.  77 S PIC 9999 VALUE 0.  PROCEDURE DIVISION.  PROCEDURE P VARYING I FROM 1 BY 1 UNTIL I>100.  STOP RUN.  P.  ADD I TO S.  DISPLAY ‘I=’,I,’SUM=’,S. |

### 三重循环的PERFORM 语句

语法：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | | | | | |
| PERFORM | 起始过程名 | | THRU | | 结束过程名 | | | | | |
|  |  | |  | |  | | | | | |
|  |  | |  | | | | | | | |
|  | [VARYING | | 第一重循环变量标识符 | | | | | | | |
|  |  | | { | 整形直接数 | | } |  | { | 整形直接数 | } |
|  |  | FROM | 初值标识符1 | | BY | 步长标识符1 |
|  |  | |  | |  |  |
|  |  | |  | | | | | | | |
|  |  | UNTIL | 结束条件表达式1] | | | | | | | |
|  |  | |  | | | | | | | |
|  | [AFTER | | 第二重循环变量标识符 | | | | | | | |
|  |  | | { | 整形直接数 | | } |  | { | 整形直接数 | } |
|  |  | FROM | 初值标识符2 | | BY | 步长标识符2 |
|  |  | |  | |  |  |
|  |  | |  | | | | | | | |
|  |  | UNTIL | 结束条件表达式2] | | | | | | | |
|  |  | |  | | | | | | | |
|  | [AFTER | | 第三重循环变量标识符 | | | | | | | |
|  |  | | { | 整形直接数 | | } |  | { | 整形直接数 | } |
|  |  | FROM | 初值标识符3 | | BY | 步长标识符3 |
|  |  | |  | |  |  |
|  |  | |  | | | | | | | |
|  |  | UNTIL | 结束条件表达式3] | | | | | | | |
|  |  |  |  | | | | | | | |

说明：

1. COBOL 74最多允许三重循环。

2. 循环变量间变化关系：格式中AFTER意思是循环变量1的变化后于循环变量2的变化，而循环变量2的变化又要后于循环变量3的变化。

例：PERFORM A VARYING I FROM 1 BY 1 UNTIL I > 10

AFTER J FROM 1 BY 1 UNTIL J > 10

AFTER K FROM 1 BY 1 UNTIL K > 10

二重循环变化关系

PERFORM TT

VARYING I FROM 1 BY 1 UNTIL I > 3

AFTER J FROM 1 BY 1 UNTIL J > 3

变化关系 I J

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

## 本章小结

# 表与检索

**学习提示**

Cobol特色的表使用方法和检索方法

## 表的基本概念

COBOL语言中的表大体相当与其它高级语言中的数组，但也不完全相同。

把具有相同属性的数据项按一定的逻辑顺序组织在一起，成为一个整体的数据组织，用一个统一的名字来代表它们，这就是“表”。

构成表的各数据项称为表元素。

在建立一个“表”以后，只要指出表名和序号（即指出表中的第几个元素）或表中的相对地址（相对于表中的第一个元素的字节地址），就可以唯一的确定一个表元素。序号称为“下标”，相对地址称为“位标”，“下标”和“位标”统称为“出现号”。

例： 在屏幕显示如下图形 CH101

\*

\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

|  |  |
| --- | --- |
| 000010  000020  000030  000070  000110  000111  000112  000120  000130  000140  000150  000160  000170  000180  000200  000210  000220  000230  000240  000250  000260  000270  000290 | IDENTIFICATION DIVISION.  PROGRAM-ID. CH101.  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  01 P.  03 P-REC PIC X(7).  01 STAR.  03 S1 PIC X(7) VALUE " \* ".  03 S2 PIC X(7) VALUE " \*\*\* ".  03 S3 PIC X(7) VALUE " \*\*\*\*\* ".  03 S4 PIC X(7) VALUE "\*\*\*\*\*\*\*".  PROCEDURE DIVISION.  G.  MOVE S1 TO P-REC.  DISPLAY P-REC.  MOVE S2 TO P-REC.  DISPLAY P-REC.  MOVE S3 TO P-REC.  DISPLAY P-REC.  MOVE S4 TO P-REC.  DISPLAY P-REC.  STOP 1. |

例：用表作上题 CH102-172

|  |  |
| --- | --- |
| 000010  000020  000030  000040  000080  000120  000130  000140  000150  000160  000170  000180  000190  000200  000210  000211  000212  000213  000214  000215  000216  000217  000218  000219  000220  000221  000222  000230  000231  000232  000240  000250  000251  000261 | IDENTIFICATION DIVISION.  PROGRAM-ID. COBOLMAIN.  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  77 N PIC 99 VALUE 0.  01 STAR.  03 S1 PIC X(7) VALUE " \* ".  03 S2 PIC X(7) VALUE " \*\*\* ".  03 S3 PIC X(7) VALUE " \*\*\*\*\* ".  03 S4 PIC X(7) VALUE "\*\*\*\*\*\*\*".  01 STAR-TABLE .  03 STAR-T PIC X(7) OCCURS 4 TIMES.  PROCEDURE DIVISION.  G.  MOVE S1 TO STAR-T(1).  MOVE S2 TO STAR-T(2).  MOVE S3 TO STAR-T(3).  MOVE S4 TO STAR-T(4).  DISPLAY "STAR".  DISPLAY S1.  DISPLAY S2.  DISPLAY S3.  DISPLAY S4.  DISPLAY "STAR-TABLE".  PERFORM P 4 TIMES.  DISPLAY "STAR的内存分布".  DISPLAY STAR.  DISPLAY "STAR-TABLE的内存分布".  DISPLAY STAR-TABLE.  STOP 1.  P.  ADD 1 TO N.  DISPLAY STAR-T(N).  END PROGRAM COBOLMAIN. |

## 表的建立

表的建立就是定义一个表，表名应在数据部中说明

OCCURS子句的最简单的格式为：

|  |
| --- |
| OCCURS 整数 TIMES |

可变长表： OCCURS 1 TO 5 TIMES

可变长表的一般格式：

|  |
| --- |
| OCCURS 整数1 TO 整数2 TIMES DEPENDING ON 数据名1 |

数据名1可以在表所在的记录中描述。也可以不在本记录中描述。但不能是表的一部分。

例： 01 STUDENT-RECORD.

02 NAMES PIC X(20).

02 COURSE PIC 9（3） OCCURS 5 TIMES.

COURSE就是一个表，这个表包含5个类型和地位相同的数据项。

表元素可以是初等项，也可以是组合项。

01 STUDENT-RECORD

02 NAMES PIC X(20)。

02 E。

03 COURSE OCCURS 5 TIMES。

04 COU2 OCCURS 5 TIMES.

05 COU3 OCCURS 5 TIMES PIC 9（3）.

04 COU22 OCCURS 5 TIMES.

05 COU3 OCCURS 5 TIMES PIC 9（3）.

引用表元素是只需一个下标的，是一维表，需要两个下标的则是二维表。

可以直接从数据部的描述看出：如果在数据项描述体中有一个OCCURS子句，而在它的上属数据项的描述体中没有OCCURS子句，则它是一维表，反之，则为二维表。当然还可以定义三维表。

OCCURS使用规则:

（1）OCCURS后面只能为整数。

（2）OCCURS子句不能出现在77层，因为77层是独立的数据项，不和其它数据发生组合关系

（3）OCCURS子句不能用于01层。“表”只能定义在记录内部，可以在01层下面增加一层（例如03层），把表定义在这一层上。

（4）如果OCCURS子句描述的是初等项，则重复出现的是初等项；如果描述的是组合项，则重复出现的是组合项。

（5）在COBOL中表元素可以是组合项。而且在建立一个多维表的同时，也建立了一维（或一、二维）表。这和其他语言的数组是不同的。

（6）只有当OCCURS所说明的数据是初等项时，才能在数据项的描述中使用PIC子句。

（7）不能用VALUE子句对表赋初值。

（8）多维表的元素在内存中是以行序排列的。

多维表描述举例

|  |  |
| --- | --- |
|  | 01 A.  02 AB PIC 9(3).  02 AC OCCURS 2.  03 AC-1 PIC 9(3).  03 AC-2 OCCURS 2.  05 AD-1 OCCURS 3 PIC 9(3).  05 AD-2 PIC 9(3). |

## 表元素的引用

“表”必须先在数据部定义，才能在过程部引用。

各个表元素是按一定规律在内存中顺序存放的，因此，在引用时必须指出表名和这个元素在表中的位置。

举例：

STAR-T（3），AD-1(1,2,3)，AC-2(1,N)

引用表元素注意事项

（1）如果已说明B是一个表，则不能直接引用表名B而不加下标。

（2）如果表元素是组合项，则引用他下属的项（可以是初等项或组合项），也必须用下标指明它是属于哪一个表元素的。

（3）如果表元素是组合项，可以用它对下属的数据项进行限制。

（4）下标只能是整常数或具有整数值的数据名，不可以是表达式。如果下标是数据名，则此数据名可以在数据部的文件节或工作单元节中定义，也可在表所在的记录中定义，但不能属于“表”的一部分。

（5）标准COBOL规定，下标不能是带下标的数据名，即不能是表元素。

（6）带下标的数据名又称下标数据名。它是标识符的一种形式

引用举例

|  |  |
| --- | --- |
|  | 01 A.  02 A-TABLE OCCURS 8 TIMES.  04 A1 PIC 9(3)V99.  04 A2 PIC 9(3)V99.  MOVE A1(2) TO B.  MOVE A1 OF A-TABLE(2) TO B.  MOVE ZEROS TO A-TABLE(2). 不能直接引用表名而不带下标 |

## 表元素赋初值

在工作单元节中给表元素赋初值的办法有两种：

1. 对包括所有表元素的整个表赋初值。这是可以对表的描述体上面一层的数据项赋一个初值即可。

例： 01 TABLE VALUE IS ZERO

03 A OCCURS 20 IS PIC 9(3)

2.联合使用OCCURS和REDEFINES子句给表元素赋初值，它的步骤是：

1. 先在工作单元节中定义一个组合项，它占的内存大小和需赋值的表一样。在该组合项中定义若干个数据项，数据项的描述和表元素的描述相同。
2. 然后对这些项分别用VALUE子句赋值。
3. 把这个组合项重定义成一个表。

例： 01 UNIT-PRICE-TABLE

03 FILLER PIC X(10) VALUE ‘0100000000’

03 FILLER PIC X(10) VALUE ‘0200000000’

01 UNIT-PRICE-TABLE-R REDEFINES UNIT-PRICE-TABLE

02 TABLE OCCUR 2 TIMES

03 PROD-CODE PIC 9(4)

03 PROD-PRICE PIC X(6)

CH103-178

## 表的应用

商品单价表

商品号 单价

0001 3.25

0002 8.27

0003 6.25

0004 18.56

0005 48.50

1. 表的应用—表描述

01 TOCK-PRICE-TABLE.

02 FILLER PIC X(9) VALUE "001000325".

02 FILLER PIC X(9) VALUE "002000027".

02 FILLER PIC X(9) VALUE "003000625".

02 FILLER PIC X(9) VALUE "004001856".

02 FILLER PIC X(9) VALUE "005004850".

01 S-P-T REDEFINES STOCK-PRICE -TABLE.

02 S-P OCCURS 5.

03 N-T PIC 9(3).

03 P-T PIC 9(4)V99.

1. 表的应用—输入与打印

1.4.5.01编写一个交互程序，由键盘输入销售商品的编号，数量，打印售出商品的编号、单价、数量、金额。

CH103-178

1.4.5.02有一个表格，记录为

NUMBERS（商品编号） PIC 9(3).

PRICE (单价) PIC 9(4)V99

CH104-180

## 位标法

### 位标名

表元素的下标指明表元素在表中顺序号

位标是代表一个表元素在表中的相对位置(记录的是内存的相对地址)

表元素相对地址用特定数据项位标名的值来表示，语法格式：

层号 表名 OCCURS 整数 TIMES PIC 描述体

INDEXED BY 位标名。

应用举例：

01 A.

03 B OCCURS 5 PIC X(2) INDEXED BY S.

S就是位标名，由S的值可以指定表B的某一元素的地址，如S的值为4时，指定表中由相对地址4开始的元素，即第三个元素

位标名不需要用PIC进行描述，也不能参加运算或输出。

### 位标法引用表元素规则

使用位标名引用表元素可以在表名之后的括号内直接写上位标名，也可以在括号内用位标名加上或减去一个整数。

位标名的值不能超过表的长度。

表无论是否指定位标，都可以用下标法引用表元素，但是如果要用位标法引用表元素，必须用INDEXED BY指定位标，位标和下标不能混合使用

在IF语句中，位标与非位标数据项比较，系统会自动将位标值转换为对应表元素的下标值，然后进行比较。

### 设置语句

由于位标是特殊的数据类型，只能用SET语句对其赋值

语法格式一：

SET 标识符1 … TO 标识符2

位标1 位标3

整数

举例：

SET I TO 10.

意思是“将位标I置于第十个表元数第一个字节的相对位置上”，是将位标地址送过去。

SET S TO D.

S为位标名，D为数据名，该语句执行的操作为将S的值置为（D值-1）\*表元素的字节数

举例：

A. 01

02 AA OCCURS 8 PIC 9(3) INDEXED S.

02 DD OCCURS 4 PIC 9(6) INDEXED J.

SET S TO J.

先将J转化为DD的下标：6/6+1=2，再计算AA表中第2个元素的相对地址（2-1）\*3=3，将3赋给S

SET D TO J.

D为一般数据项，J为位标名，执行该语句是将位标名J的值，化成对应表元素的下标，传给一般数据项。

由于位标是特殊的数据类型，只能用SET语句对其赋值

语法格式二：

SET 位标1… UP BY 标识符2

DOWN BY 整数

给位标名增加UP BY或减少DOWN BY一个值，其增加或减少的值由其后的标识符或整数的值来确定

举例：

* SET I UP BY 10.

I为位标名，将相对地址为I的元素，其下标加10，再换算为对应的相对地址，结果存入I。

* SET S DOWNBY D.

1. 位标数据项定义

层号 位标数据项 USAGE IS INDEXED.

举例：

77 C USAGE IS INDEXED.

SET C TO I. I为位标名，将I的值存入C中

SET I TO C. I为位标名，将C的值送给I。

位标数据项的值可以传送给一般数据项，但是同位标名一样，先将位标数据项的值转换为对应表元素的下标值。

位标数据项不需要用PIC子句描述，也不能参加运算。

## 表的顺序检索

语法：使用SEARCH 语句检索表要用位标引用表元素

对表元素进行顺序检索，语法：

SEARCH 表名 [VARYING 位标名1] [AT END 强制语句1]

标识符2

WHEN 条件1 强制语句2

NEXT SENTENCE

[WHEN 条件2 强制语句3

NEXT SENTENCE]。

例：SET I TO 1

SEARCHING.

SEARCH SHOP-PRODUCTION (表名)

AT END GO TO FINISH

WHEN QTY(I) < 500 DISPLAY NUM(I),QTY(I)

WHEN GRADE(I) = ‘D’ DISPLAY NUM(I)

SET I UP BY 1

GO TO SEARCHING.

VARYING选项说明

如果不带VARYING可选项，则检索时使用定义表时的第一个位标，其它位标值不变；

如果带VARYING可选项，则使用指定的位标，检索由位标名制定的值开始检索，其它位标值不变；如果使用其它表的位表，则仍使用定义表时的第一个位标，两个位标值同时增长；

当VARYING可选项指定的普通的数据项时，仍使用第一个位标，数据型和位标同时增长。

注意：（1）该语句顺序检索表中的各元素，当未找到满足WHEN条件的记录则转入NEXT SENTENCE。SEARCH语句检索的表必须用位标法引用。在WHEN条件中必须出现位标法引用的表元素。

例：SET I TO 1

SEARCHING.

SEARCH SHOP-PRODUCTION (表名)

AT END GO TO FINISH

WHEN QTY(I) < 500 DISPLAY NUM(I),QTY(I)

WHEN GRADE(I) = ‘D’ DISPLAY NUM(I)

SET I UP BY 1

GO TO SEARCHING.

（2）如果不带VARYING可选项，则检索时使用定义表时的第一个位标，其它位标值不变；如果带VARYING可选项，则使用指定的位标，其它位标值不变；如果使用其它表的位表，则仍使用定义表时的第一个位标，两个位标值同时增长；当VARYING可选项指定的普通的数据项时，仍使用第一个位标，数据型和位标同时增长。

CH105-185

## 表的对分检索

对全表进行对分检索，要求表中数据必须是顺序排列的：

升降序说明

01 LABLE.

02 A OCCERS 100 INDEXED BY K

ASCENDING KEY IS B

DESCEWNDING KEY IS C.

03 B PIC X(10).

03 C PIC X(10).

说明表A以B为键数据名按升序排列，当元素B值相同时再按C降序排列。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SEARCH | ALL | 表名 | [AT END | 强制语句1 | ] |
|  | WHEN | 条件 | { | 强制语句2 | } |
|  |  |  | NEXTSENTENCE |
|  |  |  |  |  |  |

举例：

SEARCH ALL BOOK-TABLE

AT END DISPLAY ‘CAN NOT FIND THE NAME’

WHEN NAME(I) = ‘COMPUTER BOOKS’

DISPLAY NAME(I),NUM(I).

1.简单关系条件只允许使用“EQUAL TO”或“=”

2.作为条件比较的一方，必须是用OCCURS子句指定作为ASCENDING KEY或DESCENDING KEY的数据名

3.复合条件可以使用AND，不能使用OR

4.不需要给位标名初值，SEARCH语句自动控制位标进行对分检索

## 用PERFORM语句对表进行检索

利用PERFORM可以实现多重循环的功能，在多维表中，按照检索条件，对表中的元素进行逐一的顺序检索。

## 本章小结

# 子程序

**学习提示**

**通过一个小例子介绍子程序定义方法和参数传递。**

## 子程序概述

一、概述

如果程序中有的部分比较复杂，会使程序冗长、庞杂，使读者和编者都感觉到不便，则可以将这部分程序另外必编写一个程序，就是子程序，在需要时调用这个程序。也就是说主程序根据需要调用子程序。特别是多个程序需要调用同一段代码时，将这段代码写成子程序更为方便。

执行语句的调用方式，只限于在本程序中使用

子程序调用方式，可以在多个程序中使用。

子程序的优点：1.使程序结构简单；2.使程序调试、编译方便；3.由于每个子程序具有相对独立性，可以由多人分别编制，提高编成效率；4.子程序通用性强，可以减少重复劳动。

注意：（1）调用子程序时就是按名字查找的。

（2）主程序可以调用子程序，子程序不能调用主程序。

（3）子程序可以调用子程序，即子程序可以嵌套。

（4）子程序不能递归调用。

主程序可以调用子程序，但子程序不能调用主程序，子程序可以嵌套。

子程序可以用不同计算机语言编写，也可以被不同语言编写的程序调用

## 主程序与子程序连接

1. 子程序结构

子程序同样由四部分组成：

1. 标识部：在标识部中说明子程序的名字，以供调用程序按名字调用它。此程序名应该是唯一的，即不能与其它程序重名。
2. 环境部：指出子程序运行的环境。子程序中用的设备与主程序相同，也应该在子程序中单独说明。
3. 数据部：在子程序中用到的数据有两种：一是与调用程序无关的数据项；二是与调用程序有关量的数据项。在子程序中专门设一个联接节（LINKAGE SECTION），用来说明与调用程序有数据联系的数据项。联接节中的数据描述体同样分别用77层号或01—49层号开头。

数据部中有三个节：

（1）文件节（FILE SECTION）用来描述输入输出文件中的数据项

（2）工作单元节（WORKING-STORAGE SECTION）用来非输入输出的数据项，譬如中间数据项。

（3）联接节（LINKAGE SECTION）用来描述被调用程序与调用程序有联系的数据项。联接节中只能出现在子程序中，但子程序不一定都要有联接节。如果在联接节中描述的数据项多于USING子句中数据名的个数，则在子程序中实际能用的以过程部部头的USING子句中指出的数据名为限。

1. 过程部：

过程部的部头：

PROCEDURE DIVISION USING 数据名1，数据名2，…。

过程部中应包括一个程序出口语句：

EXIT PROGRAM.

而不应写成STOP RUN，否则不能从子程序返回调用程序。

EXIT PROGRAM应该是逻辑上的最后一个语句，在它后面的语句都不被执行。它可以是过程部的最后一行，也可以不在最后一行。

1. 调用语句语法：

调用语句的一般形式为

CALL “子程序名” [ USING 数据名1 [，数据名2] …]

子程序过程部头

PROCEDURE DIVISION [ USING 数据名1 [，数据名2]…]

子程序名指子程序中由PROGRAM-ID子句定义的程序名

USING后面的数据名是主程序和子程序公用的数据项，通过数据项进行数据的交换，即数据项可以是主程序传送的参数，也是子程序返送的结果

USING后面的公用数据项必须在子程序的数据部连接节加以说明

CALL语句可以不带USING短语，此时没有共享数据，相应子程序过程部部头也不带USING短语。

说明：（1）主程序中除了CALL语句外，其它与从前的介绍没什么不同。

（2）主程序CALL语句中USING子句中用到的参数个数和子程序过程部部头USING子句中用到的参数个数必须相同。

（3）主程序和子程序中USING子句中各参数是依照它们各自次序确定关系的，而不是按照名字相同来确定关系的。

（4）可以由主程序向子程序传递数据，也可以由子程序向主程序送回数据。在执行子程序期间主程序和子程序中相应的数据名共享内存中同一段存储单元，因此它们具有同一个值。

（5）调用程序和被调用程序的相应的数据项的长度应相同。如果长度一样而类型不同则按从前介绍过的传递原则进行类型转换。

（6）也可以无数据的传递。在调用子程序时，子程序按其本身的语句内容执行，而不将数据带回主程序。

（7）每个程序（主程序或子程序）中定义的数据名，只在本程序中有效。

1. 联接节

子程序USING语句中所排列的数据名全部在联接节定义

联接节编写在工作单元节后。

联接节（LINKAGE SECTION）中数据名可以使用77层和01层（可以含有02～49），但决不能使用VALUE子句赋初值，也不能使用88层条件名。联接节定义的数据项只是形式上的数据项，系统不分配存储空间，和主程序中相匹配的数据向公用一个存储空间。子程序是根据主程序定义的数据项地址来调用它的内容。

## 子程序返回语句

语法格式：

段名.

EXIT PROGRAM.

子程序通过返回语句，返回到主程序中CALL语句的下一条语句，并继续执行，同时把相关数据带回主程序。

返回语句一定要有段名，且只有一条返回语句

## 公用区数据

数据名应遵守规则：

主程序CALL语句中USING子句中用到的参数个数和子程序过程部部头USING子句中用到的参数个数必须相同、对应参数的类型和长度应一致

主程序和子程序中USING子句中各参数是依照它们各自次序确定关系的，而不是按照名字相同来确定关系的。

可以由主程序向子程序传递数据，也可以由子程序向主程序送回数据。在执行子程序期间主程序和子程序中相应的数据名共享内存中同一段存储单元，因此它们具有同一个值。

## 子程序举例

例1．5．4．01\_CH111-194

主程序把产品单价和销售数量交给子程序，子程序用产品单价乘上销售数量得到销售额，并送回主程序。

主程序

|  |  |
| --- | --- |
|  | IDENTIFICATION DIVISION.  PROGRAM-ID. CH111.  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  77 PROD-SOLT PIC 9(3)V99.  77 QTY-SOLD PIC 9(4).  77 VALUE-SOLD PIC Z(7).99.  PROCEDURE DIVISION.  BEGIN.  PERFORM AA 10 TIMES.  DISPLAY "END".  STOP RUN.  AA.  DISPLAY "PROD-COST".  ACCEPT PROD-SOLT.  DISPLAY "QTY-SOLD".  ACCEPT QTY-SOLD.  CALL "CALPRO" USING PROD-SOLT QTY-SOLD VALUE-SOLD.  DISPLAY "VALUE-SOLD= " , VALUE-SOLD. |

子程序：

|  |  |
| --- | --- |
|  | IDENTIFICATION DIVISION.  PROGRAM-ID. CALPRO.  ENVIRONMENT DIVISION.  DATA DIVISION.  WORKING-STORAGE SECTION.  LINKAGE SECTION.  77 PROD-SOLD PIC 9(3)V99.  77 QTY-SOLD PIC 9(4).  77 VALUE-CALC PIC Z(7).99.  PROCEDURE DIVISION USING PROD-SOLD QTY-SOLD VALUE-CALC.  BEGIN.  COMPUTE VALUE-CALC = PROD-SOLD \* QTY-SOLD .  BB.  EXIT PROGRAM. |

## 本章小结

# 程序部详解

**学习提示**

**详细介绍cobol程序四大部的作用与语句书写，并进行小结，为学习文件作铺垫**

## 标识部

标识部是COBOL源程序的第一部分，是必要元素，标识部的主要作用是给程序起名，并加入备忘信息。程序标识符段是必写的．在这个段中给出程序名。其它段，包括作者、设备、程宇的编写和编译日期，安全性等等，都是任选的段，可根据需要或程序员的习惯，自由决定写或不写。在书写的情况下．编译系统对这些短的内容是按注解来处理的。

|  |
| --- |
| IDENTIFICATION DIVISION. |
| PROGRAM-ID. 程序名. |
| [AUTHOR. [作者名].] |
| [INSTALLATION. [计算机设置的场所].] |
| [DATE-WRITTEN. [修改日期].] |
| [DATE-COMPILED. [编译日期].] |
| [SECURITY. [保密程度].] |
|  |

COBOL规定，每一个程序（包括主程序或子程序）都必须有一个名字，当系统需要找某一个程序时，就按名字查找，即按名字管理和调用。  
标识部的必写部分  
IDENTIFICATION DIVISION.  
PROGRAM-ID. 程序名.  
    除程序标识段外，标识部可以没有其他段，但允许程序编写者在这部分写入某些信息以备忘，例如作者、日期、保密程度等等。  
标识部的任选部分  
AUTHOR. 作者.  
INSTALLATION.  计算机设置的场所.

DATE-WRITTEN.  程序编写的日期.

DATE-COMPILED. 程序编译的日期.

SECURITY.      保密程度.

标识部举例

IDENTIFICATION DIVISION.

PROGRAM-ID.EX.   
AUTHOR.ZHANG.  
INSTALLATION.DALIAN.   
DATE-WRITTEN.2003/01/01.  
DATE\_COMPLIED.2003/01/01.

SECURITY.HIGH.

程序部中的段标题只能按部描述中指定的顺序出现。除PROGRAM-ID段以外的所有标识部段落都可以省略，但如果出现，则要求按规定顺序出现。

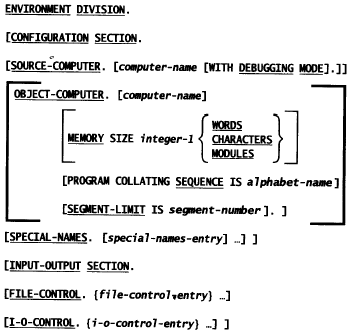
PROGRAM-ID 在这个必需的段落中，program-name是用户定义字，表示程序名，可以选择一个有意义的名称。在应用程序中，应建立一个程序命名规则，并坚持按这个规则命名。在说明行中建档命名规则。如果公司已经有现成的命名规则，则按公司的规则办。

## 环境部

### 环境部概述

环境部是COBOL程序中唯一与计算机硬件设备有关的部分，描写本程序编译和运行时所要求的环境，主要指硬设备（主机与外部设备），以及文件和输入输出条件，将程序中用到的内部文件名与外部文件之间建立起来联系。提高程序的可移植性，只需要修改环境部既可。 它提供下列信息：  
1.编译与运行本程序的系统与主机，  
2.本程序对系统没备的特殊命名(助记忆名)对币号及小数点的特殊规定,  
3.本程序所用到的文件及其特性，以及它门是与哪个外部设备相联系的：  
4.本程序在输入输出方面的特殊观定。

环境部的一般格式

  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.               配置节  
SOURCE-COMPUTER.       源计算机名  
OBJECT-COMPUTER.       目标计算机名  
SPECIAL-NAMES.         专用名描述项  
INPUT-OUTPUT SECTION.                 输入输出节  
FILE-CONTROL.           文件控制描述体  
I-O-CONTROL.            输入输出控制描述体

### 配置节

配置节描述程序运行所需的硬件配置及对设备的某些特殊规定。 配置节是可选的，如果程序是子程序，则这节应省略。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [CONFIGURATIONSECTION.] | | | | | | | |
| [SOURCE-COMPUTER. 源计算机名 [WITH DEBUGGINGMODE]] | | | | | | | |
| [OBJECT-COMPUTER. 目标计算机名 | | | | | | | |
|  |  | { | WORDS | | | } |  |
|  | [MOMORYSIZE 整数1 ] | CHARACTERS | | | ] |
|  |  | MODULES | | |  |
|  | [PROGRAM COLLATING SEQUENCE IS 字母表名].] | | | | | | |
|  |  | | | | | | |
| [SPECIAL-NAMES. | | | | | | | |
|  | [系统名1 IS助忆名]…… | | | | | | |
|  |  | { | | STANDARD-1 | } | |  |
|  | 字母表名 IS | NATIVE | ] |
|  |  | 系统名2 |  |
|  | [DECIMAL-POINTISCOMMA] | | | | | | |
|  | [CURRENCY SIGN IS 直接量].] | | | | | | |
|  |  |  | | | | | |
|  |  |  | | | | | |
|  |  |  | | | | | |

配置节中允许安排三个段，即源计算机段(SOURCE-COMPUTER），目标计算机段(OBJECT-COMPUTER），和特殊名段(或叫专有名段，SPECIAL-NAMES)，三个段各完成特定的描述作用

* 配置节(CONFIGURATION SECTION)是必写的

源计算机段

SOURCE-COMPUTER. 源计算机名.

目标计算机段

OBJECT-COMPUTER. 目标计算机名.

专用名段

专用名段用来通知系统把系统中原规定的一些设备名或功能名或符号改为用户自己指定的名字或符号

其一般形式为：

SPECIAL NAMES.

[DECIMAL-POINT IS COMMA.] 小数点改为，

[CURRENCY SIGN IS 非数值常量.] 货币符号

[专用名 IS 助忆名.]

源计算机段

在配置节CONFIGURATION SECTION中，第一个段是源计算机段SOURCE-COMPUTER  
其格式如下：

SOURCE-COMPUTER段

标示编译程序的计算机，computer-name是COBOL厂家定义的系统名。如果省略这个段（而程序又不包含在另一程序中）或其computer-name项目省略，则编译程序的计算机是源计算机。

如果指定WITH DEBUGGING MODE从句，则编译所有源调试行（调试行就是指示符区为D的行）。否则把调试行作为说明语句。

SOURCE-COMPUTER. 源计算机名 [ WITH DEBUGGING MODE] .这个段给出编译与调试本程序的计算机名．其内容被编译程序当作注释。可任选短浯WITH DEBUGGING MODE作用于过程部直陈性的调试节后，及过程部的三个动态调试语句：

READY TEACE

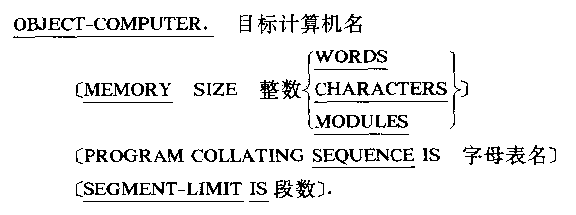
RESET TEACE

EXHIBIT

当在写有这些语句的行的第七列上加”D”标记．而源汁算机名后附以WITH DEBUGGING MODE时，编译程序将处理这些语句使之起调试工具的作用，即将过程部执行时所经过的节、段名顺次显示在终端屏幕上(READY TRACE），或将关键点上关键数据的值显示在终端屏幕上(EXHIBIT）。在程序调试完毕后，可以不去修改过程部，只要将源计算机段中的WITH DEBUGGING MODE这一短语删悼，重新编译一遍程序．这时带“D”的动态调试语句将破作为注释行处理，在执行时不起作用．这对程序的调试无疑带来—定的方便；源计算机段是任选段。

目标计算机段

OBJECT-COMPUTER段描述编译程序运行的计算机。  
OBJECT-COMPUTER. 目标计算机名.



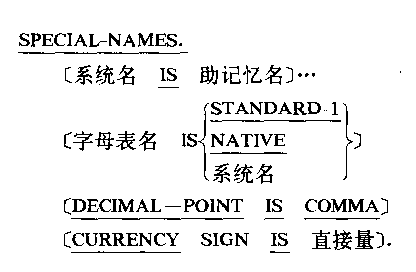
这个段的作用主要是给出执行本程序的计算机名，其内容也是注释性的。此外有三个任选语句1.MEMORYSIZE子句  
这个子句提供有关目标程序运行时所需内存大小的信息，可有字（WORDS）、字符(CHARACTERS）模块（MODULES）三种形式；但本子句内容只作为注释；

1. PROGRAM COLLATING SEQUENCE子句
2. 程序整理顺序子句指明本程序中对数据的表示和排序以哪个字母表为依据．不给出这个子句时隐含采用lSO 8位码．也就是是ASCII码，目前绝大多数系统采用这个际准。这个子句确定了非数值量比较的依据。alphabet-name是用户定义字，引用SPECIAL-NAMES段中定义的名称。3．SECMENT-LIMIT子句   
   段限制于句用于程序采用分段结构的场合。程序分段是在目际程序较大、而内存容量较小时．对过程部加上分段．使其中常用的段常驻内存（根段），不常用的段放在磁盘上，待运行至这部分时，才从库中调入内存(独立段）．段限制子句用以规定常驻内存的根段的段数，使得对内存的需求更加灵活。  
   允许恨段段数为1-50，即若段限制子句中给定段数为n，则意味着。0～n-1号这n个段是根段．分段模块只影响COBOL程序的效率，不影响程序的逻辑。

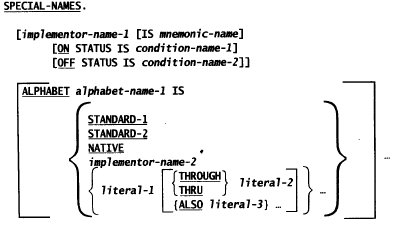
专用名段

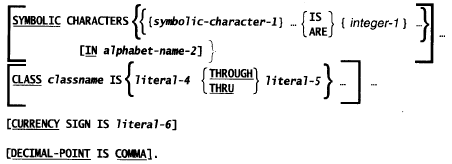
专用名段用来通知系统把系统中原规定的一些设备名或功能名或符号改为用户自己指定的名字或符号 ，也就是定义助记名

其一般形式为：



更详细的



  
第一个从句将用户定义名（mnemonic-name）与外部设备或开关（implementor-name1）相关联,或者提供开关的开/关位置（condition-name-1和condition-name-2）的用户定义名。这个从句的使用取决于COBOL厂家提供哪些设备和外部开关。如果用这个从句将名称与设备相关联，则这个名称可以用于ACCEPT和DISPLAY语句。通常，COBOL厂家提供SISIN or SISOUT之类的名称，引用其标准输入和输出流。

ALPHABET从句将特定字母表与程序其他地方引用的用户定义字（alphabet-name）相关联。

SYMBOLIC CHARACTERS从句可以通过制定字母表中字符的顺序位置而对字母表中的特定字符命名（symbolic-character）.

CLASS从句可以对字符集命名（classname），然后可以在类条件测试中使用。Literal-4为表示字母在字母表中顺序位置的整数或直接指定字符的非数字直接数。

CURRENCY SIGN从句指定数据PICTURE从句中表示货币符号的字符，如果省略这个从句，则使用COBOL的货币符号$

DECIMAL-POINT IS COMMA从句切换数字直接数和PICTURE中逗号与句号的功能。逗号用作小数点，句号用作编辑符

SPECIAL NAMES.  
[DECIMAL-POINT IS COMMA.]  
[CURRENCY SIGN IS 非数值常量.]  
[专用名 IS 助忆名.]

 ”系统名 ls 助记忆名”子句用于把系统对外部设备的专门命名加以变换，从而在本程序的后面用便于记忆的名称呼这些设备。  
字母表名子句用以对目标计算机段在PROGRAM COLLATlNG SEQUENCE中所指明的字母表加以限定，在采用ISO(也就是ASCII）八位码的情况下，可限定为STANDARD-1或NATIVE.  
所谓STANDARD-1，就是国际通用标准，所谓NATIVE，就是适应不同国家和地区的需要而和通用标准略有差异的10个版本（其中第十个版本，即美国．加拿大．澳大利亚版本和通用标准完全一致）。

特殊名段中的第三个子句”DECIMAL-POINT IS COMMA.”用以使子程序中ICTURE字符   
串和数值常量中的逗号与小数点功能互换．即小数点不用句号而用逗号，分隔   
数位的逗号却用句号，这是欧州人的习惯。  
最后一个子句”CURRENCY SIGN IS 常量 ”说明用什么字符作币号。在COBOL程序中，通常用美元号$作币号，若不用美元作货币单位，则通过这个子句来更换。子句中的常量“是单个字符，这个字符   
1.不应是数字  
2．不应是引号．分号或空格；  
3。不应是PICTURE字符串中可能出观的其它任何字符，即A．B，C，D，P．R，S，V．X，Z，+，-，\*，/，（，），．和，。  
配置节中的这三个段都是任选的．如果这三个段都不写．那末配置节的节标题也不写；

　配置节举例

CONFIGURATION SECTION.

SOURCE-COMPUTER.IBM-PC.

OBJECT-COMPUTER.IBM-PC.

SPECIAL NAMES.

CONSOLE IS CON. 控制台-CON

PRINTER IS PRT. 打印机-PRT

CURRENOY SIGN IS “￥” . 货币符号-￥

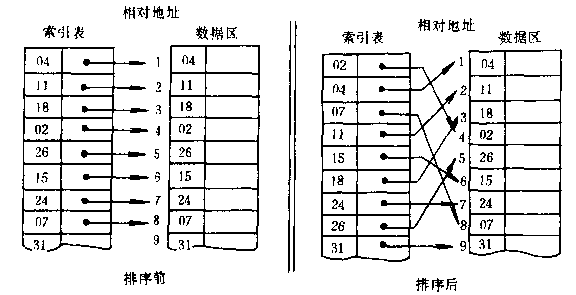
DECIMAL-POINT IS COMMA. 小数点-，

### 文件组织方式

在COBOL中允许三种类型的文件，即顺序文件，索引文件和相对文件．

顺序文件是最简单的一种文件组织方式，即把各记录顺序地安置在物理上相邻接的存贮介质上。这种文件中的记录的写入次序．排列次序和读出次序三者一致、这种文付组织方式最为简单，但存取速度低。在文件中找某一个记录时，必须从头扫描这个文件。因此顺序文件只适用于处理方式为每次处理所有记录的应用场合。比如工资文件．每个月发工资时反正要把所有人的工资记录都打印出釆，组织为顺序文件也就可以了。

对于主要应用方式为随机地查找某一记录的文件，为提高查找效率，可组织为索引文件。这时．逻辑上的一个用户文件，实际上对应于两个部分：除了数据部分外，还有一个索引部分。如图所示。



    索引文件建立时．将数据记录按到来次序存入数据区，同时由软件在索引区建立索引表，索引表中包括相应记录的键值及其在磁盘上的地址。刚建立的索引表是无序的，在输入结束后由软件将索引表排序P。也就是说，在排序前索引表和记录的物理顺序一致，在排序后和记录的逻辑顺序一致。

    在索引文件中查记录时，先根据主关键字值在索引表中找到相应的登记项，然后根据这个登记项中的地址转向数据区找到该记录。由于索引表远比数据区小，所以在索引表中的顺序搜索要比直接在数据区中顺序搜索快得多。

上述索引文件是索引非顺序文件．即数据区中的记录不是按键值顺序悱列的，因此索引表中对于每一记录必须有一个登记项，这叫稠密棠引(dense index）．这种索引文件允许记录以任意次序进入文件．处理增删比较万便、目前大多数系统采用这种方式。但是也有系统采用索引顺序文件的组织万法。在索引顺序文件中，记录在数据区中是按键值顺序徘列的，索引表中对一组己录有一个登记项．登记该组记录的最大键值及该组记录的起始地址．如图所示。这样的索引叫做稀疏索引(sparse index），显然，以顺序方式(即以稀疏索引）组织的索引文件，所占空间较少，但在建立文件时必须采取顺序序取方式，并保证记录严恪按键值的递增次序输入。

1．文件的组织形式

所谓文件的组织形式是指记录在文件中排列的方式。

顺序文件：先入先出。

索引文件：在建立文件时除了开辟一个区域存放记录，还建立一个“目录”以便查找，这个目录称为索引。

直接文件：随机文件，逻辑上相邻的两条记录，物理地址不一定相邻。

相对文件：在建立相对文件时，除了送入记录本身以外，还要求对记录的顺序编号，便于查询。

磁盘上可以建立以上四种文件，磁带上只能建立顺序文件。

2．文件的存取方式

顺序存取方式。

随机存取方式：又叫直接存储方式，这种存储方式比顺序存取方式效率高。

动态存储方式：它是顺序存取方式和随机存取方式的结合。

|  |  |
| --- | --- |
| 文件组织形式 | 存取方式 |
| 顺序文件 | 顺序存取 |
| 索引文件 | 顺序存取、随机存取、动态存储 |
| 直接文件 | 随机存取 |
| 相对文件 | 顺序存取、随机存取、动态存储 |

注意：索引文件在建立时，只能用顺序方式写入，建立后可随机存储。

### 输入输出节

#### 输入输出节概述

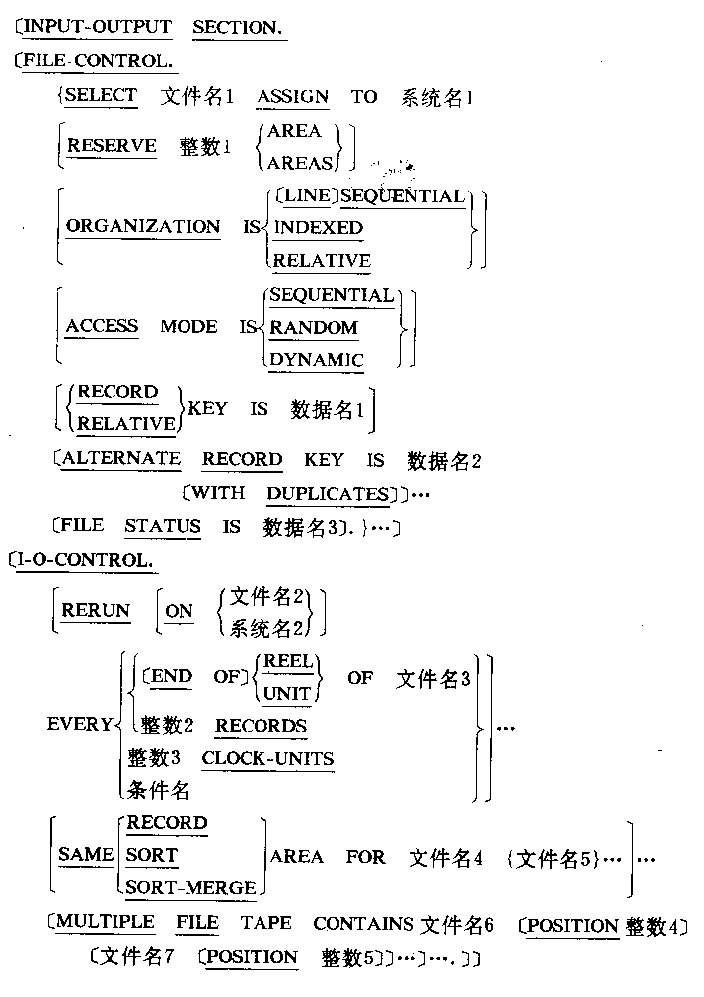
输入-输出节用以描写本程序中各输入输出文件的特性以及它们与外部设备的联系，有关输入输出文件缓冲区方面的规定，以上两部分内容分别在文件控制段和输入输出控制段中。只要程序中用到文件．就一定有输入—输出节，但可以没有其中的输入输出控制段。

输入输出节是任选的，但当程序中涉及到文件时，则该部分是必写的，应在这个节中把程序中的内部文件与外部文件（包括外部设备）联系起来。

包括两个段：

文件控制段FILE-CONTROL.给程序中使用的文件命名指出存放该文件的外部设备及其他的有关文件控制的信息

输入输出控制段I-O-CONTROL.定义目标程序使用内存的特殊控制技术



#### 文件控制段

文件控制段主要有以下作用：  
1.为文件规定文件名，说明该文件驻留于哪个外部设备  
2.规定文件输入输出缓冲区的个数；  
3.文件的类型和对文件的存取方式；  
4 为反映文件存取操作结果的状态码指定工作单元。

文件控制段给程序中使用的文件命名指出存放该文件的外部设备及其他的有关文件控制的信息

INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
[SELECT 文件名 ASSIGN TO 外部文件名或系统设备名]  
[ORGANIZATION IS SEQUENTIAL或INDEXED]  
[ACCESS MODE IS SEQUENTIAL或RANDOM或DYNAMIC]  
[FILE STATUS …]  
[RECORD KEY 数据名.]

 1.SELECT子句和ASSIGN子句  
在对于每个文件的描述中，这两个子句是必写的。SELECT子句指明所选用的文件在本程序中是用什么文件名来标识的，是指源程序中用到的内部数据文件名（在数据部中定义），每一个数据文件都必须用SELECT定义一次，COBOL程序中的READ,OPEN等语句都是用该文件名进行操作 ；ASSIGN必须跟在SELECT子句之后，用以给前面定义的数据文件指定一个输入输出的存储介质或外部设备，ASSIGN子句指示系统应到哪个外部设备上去找这个文件（或建立这个文件）。子句中的”系统名“就是系统对外部设备的命名．应按系统规定书写．微型机一股把磁盘直接写成“DISK”．把打印机直接写成”PRINTER”，如果在配置节的特殊名段中己对有关系统外设规定／助记忆名．则可写成相应的助记忆名。这两个子句必须是每个文件描述体的第一．第二个子句。在这里定义的每个文件必须在数据部中有相应的文件描述体（FD）。

SELECT子句标识文件控制段中所取得文件名，是指源程序中用到的内部数据文件名（在数据部中定义），每一个数据文件都必须用SELECT定义一次，COBOL程序中的READ,OPEN等语句都是用该文件名进行操作

ASSIGN必须跟在SELECT子句之后，用以给前面定义的数据文件指定一个输入输出的存储介质或外部设备。

1. RESERVE子句为文件规定输入输出缓冲区的数目。为了说明这个子句，先简要地叙述一下文件存取操作的执行过程。

文件的存取操作是以逻辑记录为单位进行的．即程序中每发出一个读写命令(在COBOL中是READ和WRITE），就从文件中取一个己录(或往文件中写一个己录)，由于文件在外部设备上(比如说在磁盘上)，而读取的记录先要送到内存用户工作区(UWA—User Work Area)，才能进一步送到中央处理器CPU去加工(或者在写的情况下，所要写的记录是在UWA中的)，这样，就要进行内存——外存的数据交换。我们知道．内存目的大多采用半导体存贮器，工作速度很快，而外存则是机电没备．存取周期相当长，这两者要差几个数量级。而且．在磁盘上读写之前要先寻道(seek),这个时间更长；这样，如果每次存取操作都要启动外存进行内外存的信息交换，系统速度必将大大降低。因此实际上，内外存的信息交换不是以记录为单位进行的，而是以物理记录(页)为单位进行的：也不是在外存和电户工作区之间直接进行的．而是在内存中另开辟系统缓冲区，通过缓冲区的中介进行的。每一程序运行时，在内存中开辟一用户工作区UWA,UWA中有程序区(放程序的目际代码）；数据区和堆栈等，数据区中对于该程序所用的每一文件都有一记录区。除了用户工作区之外，讨于该程序所用的每—文件．在系统缓冲区中要开辟一相应的输入输出缓冲区，其大小是一页（页的典型大小是1024个字节，微型机页面较小，通常是256个宁节）。程序第—次进行读写操作时，要启动磁盘，在相应文件中找到所需汜录．然后把该记录所在的页一次传送到系统缓冲区相应的页，然后再从缓冲区中把所需记录传送到用户工作区相应的记录区中以便进一步处理。以后的存取操作发生时。系统首先要在缓冲区中查找，如果听需记录已在缓冲区中，则直接从缓冲区传送至用户工作区，不再启动磁盘，如果所需记录不在缓冲区，才启动磁盘，找到所需新记录，然后把该记录所在页面调入缓冲区，也就是用文件中新的一页代替老的一页；由于相邻的两次序取操作发生在同一页面中的概率(叫”命中率”一般刚当大，这就大大减少了I：O次数，从而提高了系统速度。

    这个过程是由操作系统中的设备管理部分负责完成的。对于用户，这个过程是透明的。  
    一股情况下，系统只为每个文件开辟一个输入/输出缓冲区。必要时，程序员可通过RESERVE子句使系统为文件保留多个缓冲区以提高处理速度（显然，这是以牺牲内存空间为代价的）。  
注意，RESERVE子句在某些系统中以下达形式出现：

0101_E8

1. 这一形式的RESERVE语句为文件规定除基本的。必需的一个缓冲区之外，是否保留或保留几个附加的缓冲区．这个子句中的整数范围受系统限制，有的允许最大取2，有些系统只允许为磁带文件保留多个(1—7个）缓冲区。

5.ORGANIZATION 子句

说明这个文件的组织形式。若选用SEQUENTIAL，则表示文件组织形式是顺序的，若选用INDEXED则表示是索引的，若省略ORGANIZATION子句则表示是顺序存取。

对于顺序文件，OROANIZATION子句可以不写。

6.ORGANIZATION 子句说明这个文件的组织形式。若选用SEQUENTIAL，则表示文件组织形式是顺序的，若选用INDEXED则表示是索引的，若省略ORGANIZATION子句则表示是顺序存取。

7. ACCESS MODE子句

ACCESS子句指明对数据文件中记录进行存取方式。如选用 SEQUENTIAL则顺序存取，如选用RANDOM则表示随机存取，如选用DYNAMIC则表示动态存取。如果ACCESS被省略则顺序存取。  
这个语句规定了对文件的存取方式．文件有组织方式与存取方式两个不同的概念；一个文件的组织方式只有一种，是固定不变的，而对该文件的存取方式可以有多种(顺序文件除外）。对于顺序文件．只能按记录在文件中排列的次序顺序存取，这个语句可以不写。对于索引文件及相对文件，除了顺序存取方式外，还可以有随机(RANDOM）及动态(DYNAMIC）两种存取方式，随机存取方式就是根据给出的主关键字或辅助关键字值(对索引文件）或记录序号(对相对文件）随机地存取一个记录．动态存取方式的含意是．对这个文件既可用顺序方式存取．也可以用随机方式存取；或者先按顺序方式存取，后来改成随机存取，反之亦然：如随机地指定某一汜录，然后从该记录开始顺序存取．由于动态方式允许在同一处理过程中根据需要交替地采用顺序和随机两种序取方式，可以绐程序员带来很大方便。

ACCESS子句指明对数据文件中记录进行存取方式。如选用 SEQUENTIAL则顺序存取，如选用RANDOM则表示随机存取，如选用DYNAMIC则表示动态存取。如果ACCESS被省略则顺序存取。

RECORD KEY子句为索引文件指明记录键，数据名及时指定的记录键名。记录键可以是组合项或初等项，每个记录应有唯一的记录键值，键的最大长度是60个字符。

 5．RECORD KEY或RElATIVE KEY子句

RECORO KEY子句用于索引文件(必写），用以指明作为主关键字的的是哪个数据项，注意，RECORD KEY后面的数据名必定指定记录中的一个数据项．对这个数据项有以下规定;

(1)是定长的数据(长度受系统限制，从32—355不等），

(2）是字母或字符数据项．不得为数值数据项(即使是数字，也当作亍符）。

(3)不得含有OCCURS子句，或包含在有OCCURS子句的组项内．

(4）PlCTURE的字符串中不得有P或/。

（5）如果文件中有多种格弍的记录，则每种记录对记录键应有相对应的描述．

（6）作为记录键的数据项的值在每个记录中必须唯一而且不得修改(这也是记录键不得规定为蚊值数据项的理田。因为数值数据项可以参与运算，意味着其值可能发生变化，而这是不允许的)。  
某些系统还规定记录键必须是记录中的第一个数据项。

RELATIVE KEY子句用于相对文件，指明存取相对文件时的记录序号要放在何处，其后的数据名是独立于记录的一个77层的数据项(有些系统允许是另一个文件的记录中的一个数据项）。对这个数据项有以下现定：

(1）必须是一个数值数据项，并且不带符号(即永远是正整数）．

（2）PICTURE字符串中不应有P或/。

（3）不包含OCCURS子句或包含在有OCCURS子句的组项中。

(4）包含的值必须唯一，不得重复，值的范围受系统限制(如有些微机最大允许32767，有些允为999999）。

 相对文件若规定存取方式为SEQUENTIAL，则RELATIVE KEY可以不给．这时系统对该文件自动从序号0开始顺序存取各记录。

6．ALTERNATE RECORD KEY子句  
这是用于索引文件的一个任选子句，其作用是给出辅助关键字，用以对文件建立辅助索引，以便实践多关键字检索。—般说来，—个文件的记录中只有一个数据项(主关键字），其值是唯一的；其它数据项的值均可能有重复．用这样的数据项作辅助关键宇位索时，给定一个值往任对应于一批记录．在COBOL中，允许在ALTERNATE RECORD KEY后面以任选短浯WITH DUPLICATES指明该辅助关键字值是否允许重复．

这个语句允许重夏．也就是说可以为索引文件规定多个辅助关键字，建立多个辅助索引。

允许个数是系统约定。微型机一般无辅助索引功能；

7．F1LE STATUS子句

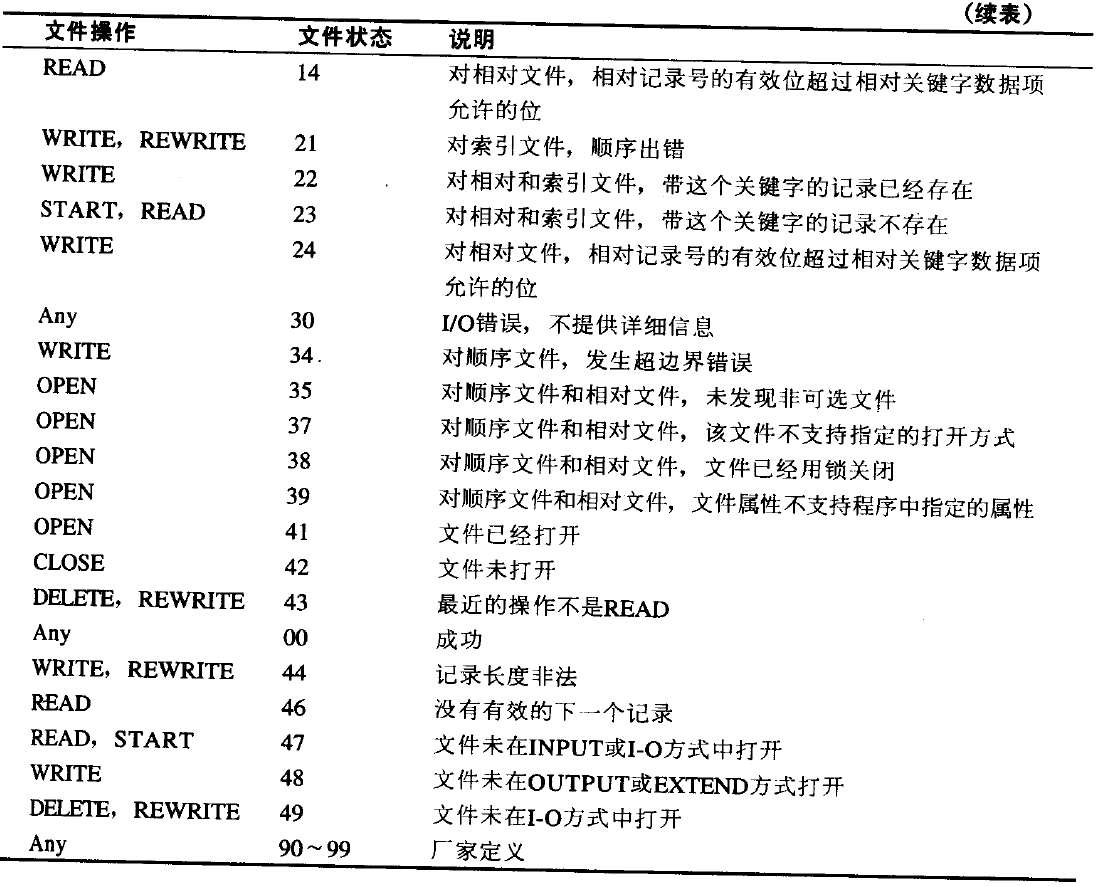
这是一个任选语司。在选用这一语句时，为该文件指定一个两字符的独立数据项，用以存放每次文件操作执行结果的状态码．前一字符称状态码1，用以指示存取操作是否成功地执行：

后一子符称状志码2，用以在存取操作出错的情况下指明出错的性质和原因．各种文件组织可能的状态码组合及其意义见表，表中S代表顺序文件，I代表索引文件．R代寝相对文件。

文件状态码



|  |  |  |
| --- | --- | --- |
| 文件操作 | 文件状态 | 说明 |
| Any | 00 | 成功 |
| Any | 02 | 对索引文件，成功但发现了重复关键字 |
| READ | 04 | 成功，但记录长度不符合程序指定的长度 |
| OPEN | 05 | 成功，但文件是可选的，前面不存在 |
| OPEN,CLOSE | 07 | 对顺序文件，成功，但媒介不是盘 |
| READ | 10 | 文件已到文件末尾 |
| READ | 14 | 对相对文件，相对记录号的有效位超过相对关键字数据项允许的位 |
| WRITE,REWRITE | 21 | 对索引文件，顺序出错 |
| WRITE | 22 | 对相对和索引文件，带这个关键字的记录已经存在 |
| START，READ | 23 | 对相对和索引文件，带这个关键字的记录不存在 |
| WRITE | 24 | 对相对文件，相对记录号的有效位超过相对关键字数据项允许的位 |
| Any | 30 | I/O错误，不提供详细信息 |
| WRITE | 34 | 对循序文件，发生超边界错误 |
| OPEN | 35 | 对顺序文件和相对文件，未发现非可选文件 |
| OPEN | 37 | 对顺序文件和相对文件，该文件不支持指定的打开方式 |
| OPEN | 38 | 对顺序文件和相对文件，文件已经用锁关闭 |
| OPEN |  |  |
| OPEN |  |  |
| OPEN |  |  |
| OPEN |  |  |
| OPEN |  |  |
| OPEN |  |  |
| OPEN |  |  |
|  |  |  |
|  |  |  |



顺序文件文件控制段定义：

FILE-CONTROL

SELECT 文件名 ASSIGN TO 设备名1，…

[ORGANAZATION IS SEQUENTIAL]

[ACCESS MODE IS SEQUENTIAL]

磁盘文件设备名DISK

ORGANAZATION 与ACCESS 是选择项，省略则默认是顺序文件

ORGANAZATION文件的组织形式

ACCESS指存取形式

SQUENTIAL指顺序存取。

索引文件文件控制段定义

FILE-CONTROL

SELECT 文件名 ASSIGN TO 设备名1，…

[ORGANAZATION IS INDEXED]

[ACCESS MODE IS SEQUENTIAL]

RANDOM

DYNAMIC

RECORD KEY 标识符

举例： SELECT 文件名 ASSIGN TO DISK

ORGANAZATION IS INDEXED

ACCESS MODE IS SEQUENTIAL

RECORD KEY CUST-NAMBER. TS163

动态存取既可以用顺序存取也可以用随机存取方式

相对文件定义

FILE-CONTROL

SELECT 文件名 ASSIGN TO 设备名1，…

[ORGANAZATION IS RELATIVE]

[ACCESS MODE IS SEQUENTIAL]

RANDOM

DYNAMIC

RELATIVE KEY 标识符

举例： SELECT 文件名 ASSIGN TO DISK

ORGANAZATION IS RELATIVE

ACCESS MODE IS SEQUENTIAL

RELATIVEKEY CUST-NAMBER.

#### 输入输出控制段

输入输出控制段用以说明程序在输入输出方面采用的一些技巧。这些技巧可节省内存空间．对磁带文件实现快速定位，或者在程序执行过程中写检查点记录以便在机器故障后再启动时可从断点继续执行．  
1。RERUN子句  
RERUN子句规定什么时候写检查点记录，写在什么地方。检查点记录内容包括内存各工作单元的数据和程序状态的所有现场信息。在有ON短语的情况下，检查点记录写到规定的文件1（它必须是在文件节中描述过的一个磁带顺序输出文件），或规定的设备，EVERY短语规定写检查点记录的时间：文件2所在的磁带卷结束，（这时可无ON短语，检查点记录就写在该盘磁带的末尾成为文件2的一部分，所以文件2必须是输出文件），或文件2每处理了n1个记录(n1 RECORDS OF 文件2．此时必须有ON短语．且后跟设备名是专用于写检查点记录的，不能被分配给其他文件）“n2 CLOCK-UNITS”和“条件名”方式规定每隔一定的时钟单位（由系统设定）或每遇到条件变量为真时就写一个检查点记录，这两种方式当然必须有ON短语．RERUN子句用于处理大磁带文件的场合；随着磁带文件愈来愈少作为由程序直接处理的对象和计算机系统可靠性的提高，这个子句目前很少有系统实现。  
 2。SAME AREA子句  
SAME AREA子句用来为多个文件规定共用—个记录区(SAME RECORD AREA FOR……）  
或共用一个输入输出缓冲区（SAME AREA FOR…）．在前一种情况下，这几个文件可以同时打开，但对应的记录区只能交替用一个文件；在后—种情况下，文件不能同时打开。在有SORT或SORT—MERGE任选项时，其后的文件中必至少有一个排序或排序一合并文件．  
由于文件的记录区通常远小于文件的输入输出缓冲区，公用记录区意义不在于节省空间．  
而在于可以最少的操作使数椐从公用记录区的一个文件传送到另一个义件；、而排序文件存贮区域的分配，常由系统进行优化分配，所以目前绝大多数系统只实现SAME AREA功能．

 3 MULTIPL FILE TAPE子句  
这个子句用来加速对多文件磁带上文件的定位。若按顺序列出一盘磁带上的所有文件时，可无P0SITION短晤。若只列出部分文件，则不需要按照顺序，但此时要通过POSITION短晤给出文件的相对位置(按POSITION 后各整数的大小顺序）。  
注意：同一磁带卷上的文件不能同时被打开。  
以上是ANSI COBOL74中对I—O—CONTROL段所规定的语法；某些系统还在这个段中加进各种形式的APPLY语句实现各种特殊的输入输出功能，比较常见的有：  
（1）APPLY FOR MASTER—INDEX n KEYS ON 文件1 [文件2]…  
这个语句用以规定将文件l，文件2的部分索引（n KEYS）预先读入内存，以便加快对文件的存取操作。  
(2）APPLY BLOCK—DENSITY n PERCENT ON 文件l [文件2]…  
这个语句用以规定在建立索引文件时原始数据块的充填密度（以百分比方式），以便处理记录的增删(只适用于索引顺序文件的组织万式）。

## 数据部

### 数据部概述

数据部是整个COBOL源程序中唯一描述数据的部分。凡是在程序中涉及的数据都要在数据部中加以说明（描述）。

数据分类：

初等数据项：没有内在联系，不可再分，具有独立的逻辑意义，包括孤立的和具有从属关系的两种

组合项：由若干孤立项（数据类型可以不同）组成，互相关联，具有一定内在联系和从属关系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学生记录 | | | | |
| 姓名 | 学号 | 课程 | | |
| 数学 | 物理 | 化学 |

数据项的属性

* 数据项的类型、存储形式、长度
* 数据项之间相互关系和从属关系
* 记录与文件的关系
* 文件的属性，如记录长度，几个记录组成一个物理块，文件是否有标号

层次关系

从属关系用层次关系来描述，数据的层次结构是：

记录→组合项→初等项

层次规定：

描述数据层次结构的层号从01开始，到49结束，记录定义为01层，为最高层。

从属项层号比上属层层号大，即01层包含02层。

几个数据项从属于同一个组合项，但互不从属，则位于同一层，具有相同的层号。

层次关系示例

COBOL中把有从属关系的数据用层次(level)关系来描述。

公司名称

**个人的基本情报**

**姓名**

**性别**

**生年月日**

**年**

**月**

**日**

**民族**

77 COMPANY\_NAME X(20).

01 PERSON\_INF.

03 NAME-1 PIC X(20).

03 SEX-1 PIC X(2).

03 BIRTHDAY-1.

05 YEAR-1 PIC 9999.

05 MONTH-1 PIC 99.

05 DAY-1 PIC 99.

03 FOLK PIC X(2).

数据部的结构

数据部包含的每一节针对COBOL程序使用的每种数据。

数据部的节

    数据部包含程序使用的所有局部数据的定义（数据描述）。针对COBOL程序使用的每种数据，每个节前面有节标题。每个街都是可选的，但程序使用的所有数据都应在数据部的某个节中定义。尽管每个节都是可选的，但程序中的节应按指定顺序出现：

文件节（FILE SECTION）：用来描述程序中用到的输入文件和输出文件及其记录中各数据项的属性。 包括打印文件和存储文件。每个文件描述中包括所含记录的定义。文件节中的数据或者从文件读取，或者由要写入文件的程序产生。

工作单元节（WORKING-STORAGE SECTION）：用来描述程序中用到的 临时结果和初始化静态数据的数据。用以描写文件以外的（也就是和文件无关的）独立的数据项和记录，通常用来保存程序运行中的中间结果、固定的标题栏的内容．控制变量，统计计数等等  
  
联接节（LINKAGE SECTION）： 只用于子程序，用来描述调用程序间发生数据传递的数据项。 描述主调程序和被调程序之间的结合参数（子程序使用主程序的数据，主程序便用子程序运行的结果数据）。  
  
通讯节（COMMUNICATION SECTION）：描述作为程序间接口的数据和消息控制系统，用以实现与通信设备相关的消息分析的COBOL方法。通信节属COBOL通信模块的一部分，可以处理COBOL程序和通信设备间的消息。

报表节（REPORT SECTION）:为了完成报表编制功能，此节用来规定欲输出的报表的“体裁”，设计各报表栏的打印形式和方法等。

COBOL数据  
COBOL中最基本的数据项称为基本数据项

数据分类：  
初等数据项：没有内在联系，不可再分，具有独立的逻辑意义，包括孤立的和具有从属关系的两种  
组合项：由若干孤立项（数据类型可以不同）组成，互相关联，具有一定内在联系和从属关系

数据组成层次结构  
    可以将基本项目组成层次结构，称为组数据项或组，组可以分成子组，最终分成基本项目不能再划分。  
    数据的从属关系用层次关系来描述，COBOL程序中数据项的定义从层号开始。层号只能是整数。01层是最高的，49层是描述数据的最底层，数据层次结构的层号从01开始，到49结束，记录定义为01层，为最高层。从属项层号比上属层层号大，即01层包含02层。77层项目用于某个组的数据项（孤立数据项，如程序中的临时变量），66层，88层。  
    数据的层次结构是：记录→组合项→初等项  
    几个数据项从属于同一个组合项，但互不从属，则位于同一层，具有相同的层号。

处于同一层次，但分别从属于不同组合项的那些数据项的层号可以不相同。

记录和文件的关系：记录是具有一定层次关系的一组数据项的最大集合。是内存中具有独立逻辑含义的最大可存取项，具有最高的层次，层号为01。多个记录可以组成数据文件。

    引用组数据项，可以不考虑其中的组件而整个作为单元，从一个地方移到另一地方。只有基本项目可以表示数学处理所用的数据。

限定数据  
    COBOL不要求程序中所有数据名唯一，程序中可以对不同数据项多次使用同一数据名，需要引用不唯一的数据名时，要用所属的组名按递减层号限定。组名同一层数据名影唯一。

    保留字OF和IN是等价的。不必列出项目所属的所有组名，只要所有的组名能使用唯一即可（注意，如果要提供更大的清晰性，即使不需要时也可以进行限定）。  
    如果数据项在文件描述符中的记录描述内，则限定符中最外面的名称可以用文件名。除了文件名描述外，记录名必须唯一，因为他们没有更高的限定符。77层项目没有限定符，必须在程序中唯一。

FILLER关键字

    有时需要定义部分记录，但直到这部分用于不会用到，这时不是编一个不用的数据名（让COBOL跟踪），而是用保留字FILLER占用这个记录部分的位置。  
    FILLER不是必需的，因此可以在数据描述项中省略。如果层号后面不插入数据名或FILLER关键字，则编译器像制定了FILLER关键字一样分配这个区域。  
    如果组项目不需要按名称引用，则可以用FILLER作为数据名，而不生成组名。但FILLER组名中定义的每个数据项应当唯一，因为不能用FILLER关键字限定数据项。

88层条件名  
    88层是特殊数据项条件名（condition name），条件名与数据项目相关联。条件名定义的数值可以测试该数据项目的内容。

    条件名定义放在普通数据项定义之后。条件名是个缩写方式，测试数据项与条件中定义的直接数组相等。

数据在计算机内的表示形式   
计算机内存的组织形式   
计算机是以二进制形式存放数据的，内存的最小单位是二进制位。8位构成一个字节

数据在内存中的存储形式  
不论是字母还是数字，都按一个字节存放一个字符存放   
  
数值型数据在内存中的存储数据  
外部十进制：一个数字在内存中占一个字节  
外部浮点数形式：  
一般形式： 数符 数值部分 E 阶码符阶码  
例： 77 A PIC +9.99999E+99 表示 +1.23456E+59  
注意：外部浮点形式的数据不能用VALUE语句赋初值。  
内部十进制数：只能存放0—9十个数字，每两个数字占一个字节。  
定点二进制形式：将十进制数转换成定点二进制数，存入内存。

数据部中要描述的数据项特征

各个数据项之间的相互关系  
每个初等数据项的数据类型，以及在内存中所占的字节长度。  
数据项和文件的关系，即组成文件的记录由哪些数据项组成  
文件的属性（如记录长度，几个记录组成一个物理块，文件是否有标号）

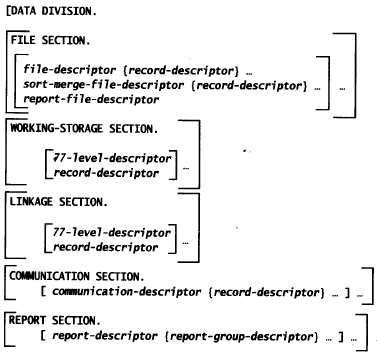
数据部中的节

文件节（FILE SECTION）：用来描述程序中用到的输入文件和输出文件及其记录中各数据项的属性。

工作单元节（WORKING-STORAGE SECTION）：用来描述程序中用到的中间数据项。

联接节（LINKAGE SECTION）：用来描述调用程序间发生数据传递的数据项。

报表节（REPORT SECTION）:为了完成报表编制功能，此节用来规定欲输出的报表的“体裁”，设计各报表栏的打印形式和方法等。



### 文件节

作用：

程序中的每一个输入和输出文件都要在文件节中加以描述。  
文件名，描述文件的结构和格式

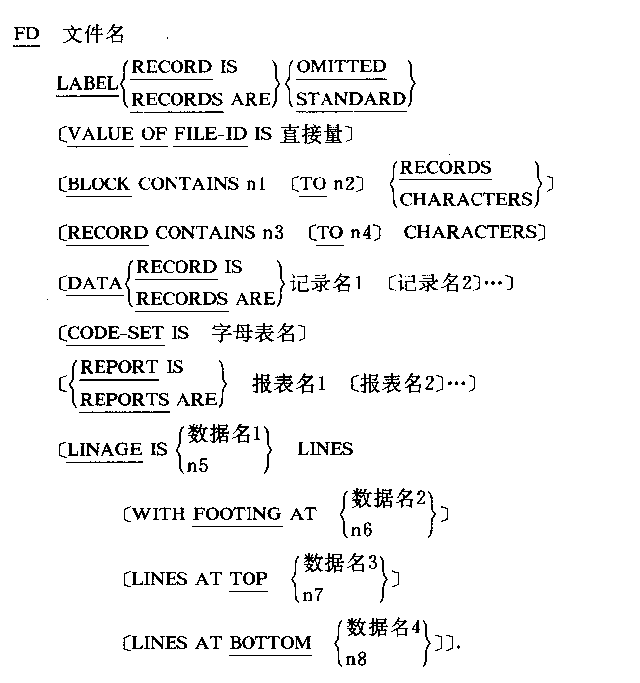
定义该文件中记录的名称

描述记录中的数据层次结构

描述各层中数据项的类型及大小

文件节通常是数据部的第一个节(除非该程序运行无任何文件，，它用来描述文件的状况和文件中各个层次上的数据。对应于各个文件有一独立的描述体，但先后次序是不重要的。对一个文件的描述按层次由文件到记录到初等项。初等项的描述次序也就是它在记录中的顺序  
(既是逻辑顺序也是物理顺序）。  
    记录描述体后的重复号说明一个文件可以有多种记录格式，这时每种记录格式都要逐一描写。在记录描述体中要描写记录及其下各个层次上的的数据，所以可一股地叫做数据描述体。

文件描述体是对整个文件有关情况（届性）的描述，系统通过文件捆述体中的信息才有可能对文件进行定位．



文件节文件描述体格式

|  |
| --- |
| DATADIVISION.  FILESECTION.  FD 文件名 LABELRECORD IS STANDARD  RECORDS ARE OMITTED  VALUEOFFILE-ID “磁盘文件名”. |

记录描述体…….

1.层指示符FD和文件名  
    对文件的描述以层指示符FD后跟文件名开始。FD是FILE DESCRlPTION的缩写。文件名应在环境部，输入输出节．文件控制段的SELECT浯句中出现过，层指示符FD必须写在A区之中。

2．LABEL RECORD标号记录子句

这是每个文件的文件描述体中必须出现的一个子句，说明文件标号记录的情况。

磁盘文件要用LABEL子句说明标号记录为标准（STANDARD ），打印文件可人为区分（控制打印机走纸）无需设置标号，可以省略（ OMITTED ）

3．VALUE OF FILE-ID子句

VALUE OF赋值子句用于定义程序中文件名所对应的磁盘文件名。

    对于规定”LABEL RECORD STANDARD”的文件。必须有这个子句，用以向系统指明本程序中的这个文件到哪个磁盘（带）驱动器上去找，这个文件在盘(带）上叫什么名字；这个语句的格式在微型机的情况下一般如下所示：  
VALUE OF FILE-ID IS "X:磁盘文件名．扩展名”  
其中X可为A．B，C或D，指磁盘驱动器代号．

 4．BLOCK CONTAINS 子句  
    这个子句规定物理存贮块（也就是”页”)的大小．文件操作的逻辑单位是记录，但为了减少I/O次数，物理输入输出的单位是页，包括若千逻辑记录，BLOCK CONTAINS子句即用来规定页面大小。句中n1和n2对应于最小页面和最大页面。不写此子句隐含BLOCK CONTAINS 1 RECORD，对于卡片文件．打印文件，物理记录和逻辑记录是一致的(一张卡片或一个打印行对应于一个己录）。

 5．RECORD CONTAINS子句  
    这个子句说明记录的大小。n3和n4介别是记录的最小和最大长度。无[TO n4]任选时，n3即是记录的精确长度。但这个子句仅起注释作用，系统实际上是根据后面记录描述体中的说明来确定记录长度的．  
对各仲设备上的文件，记录长度最大值一般有如下限制：  
卡片文件：80个字符  
打印文件：132个字符  
盒式磁带文件；256个子符  
磁带文件：32767个子符  
磁盘文件：一个磁道的容量．

 6 DATA RECORDS干句  
    该子句说明本文件的记录名．如果只包括一个记录名，则是单一记录格式文件，如果有一个以上记录名，则是多记录格式文件。但必须注意，无论何种情况，在内存用户工作区中只保留一个记录区。  
由于记录名在随后的01层记录描述体中必定出现，所以这个子句是任选的，可写可不写，写出时仅作为注释。  
一个文件可以规定多少种记录格式?这受系统限制，如有些系琉允许多达20种

记录描述体由01层号开始，后跟记录名，然后用层号表示出组成该记录的所有数据项的层次关系，还要描述每一初等数据项的类型和长度。

文件节举例

DATA DIVISION.

FILE SECTION.

FD DISK-FILE LABEL RECORD IS STANDARD

VALUE OF FILE-ID “IBD.DAT”.

01 HJDJK.

03 HZXM PIC X(20).

……

FD PRINT-FILE LABEL RECORD IS OMITTED.

01 PR-REC PIC X(80).

文件节说明

FD从A区开始写，子句从B区开始

只能在整个文件描述体结束后，方可用句号，各子句间用空格分隔

磁盘文件名用引号括上，在程序中不能直接引用

一个FD只能描述一个文件

### 工作单元节（WORKING-STORAGE SECTION）

工作单元节：程序中用到的中间数据应在本节中描述，还可以利用工作单元节为某些数据赋以初值。  
工作单元节中描述的数据项分为两种：  
孤立的数据，以层号77开头  
组合项：以01到49之间的一个数字作为开头  
通常应先写77层，再写01到49层。

字型子句（PIC子句）

字型子句（PIC子句）  
PIC子句用来描述每一个初等数据项的一半的特征和编辑要求。  
用于说明  
1.数据项是什么类性的  
2.数据项占用多大的内存域  
3.是否需要按打印的要求准备相关字符

数据项分类汇总

字母数据

77 A PIC AA.

字母数字数据

77 A PIC AA999.

* + - 1. 数值型数据项的描述

COBOL语言中用“9”，“V”，“S”，“P”4个描述符来描述数值型数据项的有关特征。  
  
“P”描述符  
又称比例因子，主要用于十分大或十分小数。用P描述不分配存储空间，但在计算机处理时会用零替代每一个P。  
如1\*109描述 02 B PIC 9PPPPPPPPP或02 B PIC 9P(9)  
如0.000065描述02 B PIC VP(4)99或02 B PIC P(4)99  
说明：  
P必须出现在全部9之前或之后。在前零前有一小数点。  
9后有N个P，表示\*10n，9前有N个P，表示\*10-m，m=N+9的个数  
P不占内存，显示或打印时P不出现，只显示出内存中实际存放的数字而不显示隐含的零。  
数值型变量小结

|  |  |  |
| --- | --- | --- |
|  | 说明 | 举例 |
| ‘9’ | 在变量中的每一位只能存放0-9之间的数字。 | 02  A  PIC   9999. ß 4568  02  B  PIC   9(4). |
| ‘V’ | 指出在定义的变量中隐含的小数点的位置。 | 02 A PIC  999V99. ß 123.82  02  B PIC  9(3)V9(2). |
| ‘P’ | 变量低位的若干的零可以用‘P’来表示。 | 02 A PIC  9PPPP. ß 30000  02 B PIC  9P(4). |
| ‘S’ | 定义一个带符号的数。 | 02 A PIC  S99. ß -30  02 B PIC  S9(2). |

字母型数据的描述  
“A”描述符  
字母型数据项中只允许存放字母或空格。按A的个数分配存储单元。  
举例：  
描述 数值 内存内容  
02 D1 PIC AAA. ABC ABC  
02 D2 PIC A(5). NAME NAME＿  
D3 PIC A(4). YES YES ＿  
说明：  
字母数据送入字母型数据项时，按“左对齐原则”存放，若数据过长，则按数据项的长度截断右边多余的字母；若数据不足，则右边补空格。  
“X”描述符  
在字符型数据项中存放由任意的字符组成的数据。  
举例：描述 字符数据 内存内容  
02 E1 PIC X(6). PAGE PAGE＿＿  
02 E2 PIC X(4). 4054 4054  
02 E3 PIC X(9). IBM/PC-XT IBM/PC-XT  
02 E4 PIC X(8). COBOL-74 COBOL-74  
02 E5 PIC A(5)X9(2) COBOL-74  
  
  
注意：  
存放在字符型数据项中的数字不能参加运算  
字符数据存入字符型数据项时，也按“左对齐原则”存放，字符数据不足数据项长度时，右边补空格，字符数据超过数据项长度时，从右边截断多余的长度。  
字符型数据项可用X描述符来描述，但在某些固定位置上总是遇到字母或数据字符也可用A和9描述符描述。

编辑型数据项

对数据进行编辑，使其满足数据输出的要求,不能用于计算。

分为编辑数值型和编辑字符型

编辑数值型用来对需要显示、打印的数值数据进行编辑加工，不能参加算数运算。一般来说，数值型数据项不能或不宜直接用来显示或打印输出，而必须通过编辑数值型数据项来输出。包括如下描述符：9 $ Z . , \* B 0 + - OR DB

编辑字符型用来对字符数据进行编辑加工。包括如下描述符：A X 9 B 0 编辑描述符从功能上来说可分为替换性和插入性。

“.”描述符  
作用：表示插入小数点的位置

举例：

77 A PIC 9(3)V9.  
77 B PIC 9(3).9.  
MOVE A TO B.

说明：若A值为148.5，A在内存中占4位，B在内存中占5位（包括小数点），将一个数值型数据项传送给带有“.”描述项数据项，传送按小数点对齐，整数部分向左传送，小数点部分向右传送，多余的数字被截去。

“+”描述符  
作用：在正数前加“+”，负数前加“-”

举例：

77 A PIC S9(3)V9.  
77 B PIC +9(3).9.  
MOVE A TO B.  
说明：若A值为+128.6，B在内存中值为+128.6，占位6位  
若A值为-128.6，B在内存中值为-128.6，占位6位  
“-”描述符  
作用：在正数前加一个空格，负数前加一个“-”  
举例：

77 A PIC S9(3)V9.  
77 B PIC -9(3)V9.  
MOVE A TO .

说明：若A值为+128.6时，B的值为 128.6，若A值为-128.6时，B的值为-128.6  
浮动加入“+”或“-”使用浮动描述，使“+”、“-”可以加在数字前适当的位置上，必须有足够的“浮动范围”。

|  |  |  |  |
| --- | --- | --- | --- |
| A描述 | 赋给A值 | B描述 | B的值 |
| 9(4)V9 | 1248.5 | +(5).9 | +1248.5 |
| 9(4)V9 | 1248.5 | -(5).9 | 1248.5 |
| 9(4)V9 | 1248.5 | -(4).9 | 248.5 |
| 9(4)V9 | 2.5 | +(5).9 | +2.5 |
| 9(4)V9 | 0.5 | +(4)9.9 | +0.5 |
| 9(4)V9 | 0.5 | +(5).9 | +.5 |
| 9(4)V9 | 0.5 | +(5).+ | +.5 |
| S9(4)V9 | -3.2 | -(5).- | -3.2 |
| 9(4)V9 | 0 | +(5).+ |  |

插入逗号“，”做分位符

作用：可以使输出数位较多的数字便于阅读

举例：

03 A PIC 9(8)V99  
03 B PIC Z(2),Z(3),Z(3).99.  
MOVE A TO B.

如A的值为3687659.28时，则B内容为3，687，659.28

如果“，”描述符前接收的均为高位无意义零并且置为空格时，对应字节也置为空格，也可以和浮动“+”或“-”描述联用。

“$”描述符

作用：对于美元金额的输出数据，常常希望在数据前面加一货币符号“$”,此时可以使用“$”描述符。

举例：

02 SUM PIC 9(3)V99.  
02 SUM-O PIC $9(3).99.  
MOVE SUM TO SUM-O.

当SUM的值为359.25时，SUM-O的值为$359.25，当SUM的值为25.25时，SUM-O的值为$ 25.25

为了取消高位无意义数字零，可以使用“$”浮动描述形式，如$(4).99，当使用浮动描述时，前面不能出现其他描述符，如要加正负号，可以将“+”或“-”加在后面，如$(4).99-，但输出数据时，符号也是打印在最后。

“\*”描述符

作用：将高位无意义数字零代以“\*” 常用于财务处理。

举例：

02 SUM PIC 9(4)V99.  
02 SUM-O PIC \*9(4).99.  
MOVE SUM TO SUM-O.

当SUM的值为38.50时，SUM-O的值为\*\*38.50，当SUM的值为0时，SUM-O的值为\*\*\*\*.\*\*

插入空格、“0”

每个“B”或“0”描述符占一个字节，该字节得内容固定为空格或数字零，因此使用“B”或“0”可以在数据项的前后或中间加一些空格或数字零。

举例：

03 A PIC 9(3)V9.  
03 B PIC BBZ(3).9B  
03 C PIC Z(3)000  
MOVE A TO B C.

如A取值为128.6,则B内容为 128.6 ,C为128000

“B”和“0”还可以和“A”，“X”联合使用描述字符型编辑数据项，如：ABABA，X0XX等。

“DB”、“CR”

在银行账目中，有时用“DB”代表借方（credit），而用“CR”表示贷方（credit）紧接在数据之后，这时我们可以使用“DB”和“CR”描述符。

03 SUM PIC S9(3)V99.  
03 SUM-O PIC $(4).99CR.  
MOVE SUM TO SUM-O.

当SUM值为-239.58时，SUM-O内容为$239.58CR,“DB”和“CR”占两个字节位置，“DB”和“CR”对正数编辑结果为两个空格，对负数编辑结果为“DB”或“CR”。  
  
编辑型变量  
在实际开发中，需要对数字用特定的形势输出。（比如：$10，000）

|  |  |  |  |
| --- | --- | --- | --- |
| ‘.’ | 02  A  PIC   99.99. à 56.82  02  B  PIC   9(2).9(2). | ‘+’  ‘-’ | 02  A  PIC   +999. à +034  02  B  PIC   -999.  à -044 |
| ‘,’ | 02  A  PIC   9,999. à 1,000  02  B  PIC   9,9(3). | ‘$’ | 02  A  PIC   $9999. à $0045  02  A  PIC   $9(4). |
| ‘0’ | 02  A  PIC   9000. à 7000  02  B  PIC   9(3). | ‘Z’  ‘\*’ | 02  A  PIC   ZZ99 à  48  02  A  PIC   \*\*99 à  \*\*48 |
| ‘B’ | 02  A PIC  B999B. à  \_123\_  02  B PIC  B9(3)B. | 浮动 | 02  A  PIC   +++99.  à +78 ,  +789 ,  +6789  02  A  PIC   $(4)99. |

赋初值子句（VALUE子句）  
对工作单元节中的数据赋以初值（VALUE子句）   
只有对工作单元节中的数据项可以赋初值，不能对文件节中的输入输出文件的数据项赋值。  
如果在组合项的描述体中使用VALUE子句，初值只能是表意常量或非数值型常量，对于组合项整体而言，一律按字符型数据项处理。  
当用一个带符号的数据做初值时，相应的PIC子句中应该有“S”描述符，否则符号无效。  
赋值时应注意类型的一致性，表意常量中只有ZERO和ZEROS即可作为非数值常量，又可作为数值常量使用。  
VALUE子句给出的值应适合PIC子句描述的范围，否则会出现截断或产生错误   
  
用法子句（USAGE子句）   
使用用法 (USAGE) 子句可以使程序设计者自由选择数据在内存中的存放形式。  
语法：  
DISPLAY  
[ USAGE IS ] COMPUTATIONAL   
COMP 或COMP-3  
其中“USAGE IS DISPLAY”的意思是“显示型的用法”，即此数据适合打印、显示，它采用标准数据类型。COMPUTATIONAL和COMP是同一意思，表示“计算型的用法”表示此数据类型适合计算，它采用适用于计算用的定点二进制形式或内部浮点形式。COMP-3表示以内部十进制形式存放  
用法子句（USAGE子句）举例  
77 A PIC 9（2）USAGE IS COMP.  
77 A PIC 9（2）COMP.  
表示数值数据项A用定点二进制形式存放数据，这种存放形式便于计算，不便于显示，故称为计算型。  
77 A PIC 9（6）USAGE IS DISPLAY.  
77 A PIC 9（6） DISPLAY.  
77 A PIC 9（6）.  
显示型用法，即此数据项适宜于显示、打印，它采用标准数据形式（一个字节放一个字符）存放。子母型、字符型、编辑型、外部十进制数据必须指明为显示型。  
用法子句（USAGE子句）举例  
01 T.  
03 T1 USAGE COMP.  
05 X PIC S9（3）.  
05 Y PIC S9（3）.  
用于对组合项的描述，表示该组合项的下属各初等项都以同一种形式存放。  
用法子句（USAGE子句）说明  
（1）USAGE子句是用来指定数据项在内存中的存储形式的。  
（2）如果省略USAGE子句，则隐含表示用DISPLAY形式。  
（3）如果对组合项描述为一种存储形式，则表示这个组合项的下属各初等项都是这种形式  
（4）USAGE子句指定的数据存储形式不应与PIC子句指定的数据类型矛盾。  
（5）长、短浮点形式已确定内存长度，不应再用PIC子句。  
（6）在传送或运算时不同存储形式的数据型数据间可互相转换。  
（7）在用DISPLAY语句显示数据项的内容时，如果数据项的USAGE“用法中”不指定DISPLAY，则在显示前，计算机自动将内存中的数据形式转换（EBCDIC码或ASCII码，视计算机系统而定），然后以字符形式显示。  
（8）如果用WRITE语句，则直接输出，不进行转换。  
重定义子句(REDEFINES子句)   
不同的数据项可以共用内存中同一段空间。 如果在时间上并不需要同时占用内存空间，那么这样就节省了内存。  
层号 数据名1 REDEFINES 数据名2   
举例:  
01 A.  
02 A1 PIC X(6).  
02 B1 REDEFINES A1.  
03 B11 PIC X(4).  
03 B12 PIC 99.  
02 C1 REDEFINES A1 PIC 9(6).  
  
重定义子句(REDEFINES子句)  
（1）数据名2与数据名1的层号必须相同。即它们应是同一层次的。REDEFINES子句不能用于88层和66层。  
（2）用REDEFINES子句的描述体应紧跟在被重新定义的数据项的描述之后，中间不能插入其他项的描述说明。  
（3）可以多次重定义，但必须紧跟出现，而且要求使用最初定义的数据名。  
（4）REDEFINES子句不能用于文件节的01层中，因为文件节中01层描述的是记录，但工作单元节中的01层是可以用REDEFINES子句重新定义的，因为这里的01层不是值输入输出文件的记录，而是指组合项。  
（5）用REDEFINES子句可以改变数据的结构，但两个数据名的长度应相同。  
（6）REDEFINES子句应在其它子句之前。  
（7）内存中的值为数据名1和数据名2共享。也就是说，重定义后两个数据名的名称和两种数据结构同时存在，都有效。程序中可使用其中任何一个。他们在内存中为同一段存储单元。如果改变了内存内容，则二者的值都因而改变。  
（8）重定义子句所在的数据描述体中不能使用初值子句赋初值。  
重命名子句(RENAMES子句)  
用REDEFINES子句可以在不改变数据项的长度的前提下，重新定义数据区的名称和数据结构的形式（包括重新定义初等项的类型和长度）。用重命名（RENAMES）子句可以把原来已定义的某些数据项重新组合成一个新项，并以一个新名字来代表它。但重命名子句不能改变原来初等项的类型、长度和属性。   
66 数据名1 RENAMES 数据名2 [ THRU 数据名3]   
重命名子句(RENAMES子句)  
01 A.  
02 B.  
03 G PIC X(2).  
03 H PIC 99.  
02 C  
03 I PIC 99V99.  
03 J PIC S999.  
02 D PIC 9(5).  
02 E PIC XX.  
66 M RENAMES B THRU C.  
66 N RENAMES E.  
重命名子句(RENAMES子句)说明  
（1）层号只能用66，它必须紧跟在01层记录中最后一个数据描述体之后，因为它是对记录中有关部分重新组合和命名的。  
（2）如无THRU部分，则数据名1和数据名2代表的是同一内容。  
（3）用THRU时，数据名2在记录中的位置应在数据名3之前，而且数据名3不应包括在数据名2之中。  
（4）RENAMES子句只能用在工作单元节中，不能用于文件节中。  
遇零置空子句   
BLANK子句的作用是：当数据项的值为零时，使它的内容改变为空白。这个子句只能用于数值型或编辑数值型的初等项。  
例： 03 A PIC $(5).99 BLANK WHEN ZERO.  
对齐子句(JUSTIFIED子句)   
JUSTIFIED子句的一般形式  
JUSTIFIED  
JUST RIGHT  
说明：JUSTIFIED子句只能用于字母型和字符型数据，而不能用于数值型数据项和编辑型数值项；因为后者是按小数点位置对齐的方式定位。   
举例 77 B PIC X(5) JUST RIGHT.  
如B=“XYZ”，则B内容为 XYZ

## 过程部

过程部是COBOL程序的核心部分，指出数据处理的步骤，即决定计算机应做什么操作 。

    部首以PROCEDURE DIVISION开头，过程部下面分若干节，节下面分段，段由若干句子组成，句子由若干语句组成，以句号“.”和空格结束。过程部可以不设节，而直接由段组成，甚至可以不设段而直接由句子组成。句子可以只含一条语句。

* 过程部部首、节名、段名应从A区开始书写，过程部的语句一律从B区开始书写，一个语句可以在一行或多行上 。
* 过程部的语句都以一个动词开始，如MOVE,OPEN,READ,WRITE等。它表示计算机应执行的操作。语句中的动词后面一般要跟以一个操作的对象，操作对象可以是数据名、文件名或过程名。

过程部的组成：   
    简单格式不分节，复杂格式用于过程部需要DECLARATIVES部分时要把段组合成节时。

    简单过程部是一系列段，各有一条或几条句子，段是过程名的一种形式，是标识一组语句的用户定义组。其他COBOL语句可以按这个名称引用段。

带有DECLARATIVES的过程部

    有些程序用特殊的USE语句标识遇到特殊条件要执行的过程。USE语句组合在一起，放在过程部开头，由关键字DECLARATIVES和END DECLARATIVES限定。如果程序中包含这些语句，则要把整个过程部分分一个或几个节,执行COBOL程序时，首先执行DECLARATIVES部分后面第一段的第一条语句。

    段名和节名都标示一组语句，这些标号可以是GO TO语句的目标或由PERFORM语句作为子程序执行。

过程部部件

    过程部由各种结构组成，从而构成COBOL程序的程序逻辑。这些结构包括语句、段和节。

语句和句子

    COBOL语句是以动词开头的单词、直接数和分隔符组合。动词是标识语句的COBOL保留字。语句应从一行的B区开始，语句可以和段名放在同一行，但最好将语句另起一行。

## 本章小结

   环境部集中地说明了本程序运行的特定环境要求：源计算机，目标计算机；对小数点，币号．系统名的特殊规定；本程序有关联的各文件的状况；输入输出方面的特殊安排……等等。  
1.每一文件必对应一个设备，由ASSIGN子句指定。设备名在不同系统中有不同的命名，应按照相应说明书的规定书写．  
2．对应程序中的每个文件，在文件控制段中都有相应的一段描写，从SELECT(必写语句）开始，以句号结束。文件的描述次序则是任意的；  
3。打印文件、卡片文件和磁带文件只能是顺序文件，只能顺序存取。这类文件的描述中只有SELECT子句和ASSlGN子句是必需的．其它ORGANIZATION和ACCESS MODE子句是任选的  
4．磁盘文件可以任意组织为顺序文件．索引文件或相对文件，所以必须通过ORGANIZA-TION子句说明。顺序文件只能顺序存取，ACCESS MODE语句可写可不写。索引文件和相对文件的存取方式又可任选为顺序．随机或动态，必须用ACCESS MODE语句说明。  
5．对于索引文件，必须逋过RECORD KEY子句说明用记录中的哪个数据项作为主关键字，有些系统还允许通过ALTERNATE RECORD KEY子句(一个或多个）规定次关键字．对于相对文件，除非存取方式为顺序，必须通过RELATIVE KEY子句说明用文件外的哪个数据项作相对键。作为键的这些数据项在随后的数据部中必须有相应的定义和描述。

输入/输出节制用于文件。有两个段：

FILE-CONTROL如果程序使用文件，则这个段对每个文件包含一个项目。文件控制项目的格式取决于描述的文件类型。

I-O-CONTROL这个段包含定义程序返回点、不同文件共享的内存区和多文件卷中文件位置的项目。

# 文件处理

**知识要点：**

介绍顺序文件、索引文件、相对文件，打开、读写删、关闭语句语法。

## 文件的基本概念

## 文件处理常用语句概述

### 打开语句（OPEN语句）

语法：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| OPEN | { | INPUT | } |  |  |  |
| OUTPUT | 文件名. |  |  |
| I-O |  |  |  |
| EXTEND |  |  |  |

举例:

|  |  |  |
| --- | --- | --- |
| OPEN INPUT A1，A2.  OPEN OUTPUT A3.  OPEN I-O A5，A6. | 也可以写成 | OPEN INPUT A1，A2，  OUTPUT A3.，  I-O A4，A5. |

OPEN语句说明

1.在开始处理文件之前，必须执行OPEN语句。

2.一条OPEN语句可以打开一个或多个文件，而且可以选择不同的打开方式

3.以INPUT方式打开的文件只有读的权力，而不能将信息写入该文件，即只能用READ语句。

以OUTPUT方式打开的文件只能被实施写操作，常用于建立一个新文件。

以I-O方式打开，可以修改磁盘文件中的数据，需要将原数据读入内存，修改后在重写入磁盘。

EXTEND只适用于顺序文件，用该方式打开文件，记录指针指向文件末尾，在给顺序文件增加记录时使用此打开方式

对索引文件和相对文件可以用OUTPUT、INPUT和I-O三种方式之一打开。

### 读语句（READ语句）

基本语法：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| READ | 文件名 |  | RECORD INTO | 标识符 |  |  |
|  | [AT END …… | | 强制语句] |  |  | 顺序读取 |
|  |  |  |  |  |  |  |
|  | [INVALID KEY | | 强制语句] |  |  | 随机读取 |
|  |  |  |  |  |  |  |

举例：

|  |
| --- |
| READ DISK-FILE  AT END DISPLAY “END OF THE FILE”.  …… |

READ语句说明

执行READ语句就是将当前记录读入文件对应的记录区

当指针已经移到最后一条记录的尾字节之后，再执行READ语句时，已无记录。此时，程序将执行AT END选项后面的语句串。它的作用是防止在文件记录结束后，再读取无效记录。

AT END 任选项中的语句串可以是一条语句，也可以是多条语句。判断语句串结束的标志为句号，每条语句可以由逗号或空格分隔。

INTO 选项：READ 文件名 RECORD INTO 标识符

等价于：READ 文件名

MOVE 记录名 TO 标识符

顺序读取格式

|  |
| --- |
| READ 文件名 [NEXT]RECORD [INTO 标识符] |
| [ AT END 强制语句]. |

顺序读取文件格式中不出现NEXT项，适用于顺序存取方式，对于索引文件或相对文件，在READ之前，可以由START语句指定开始读的记录键值，这样READ语句就由指定的记录开始顺序读取记录

对于动态存取的文件，顺序读时应写出NEXT选择项

选用AT END子句时在读完文件中的最后一条记录后，再执行该操作则认为产生AT END条件，执行AT END之后的强制语句。

随机读取格式

|  |
| --- |
| READ 文件名 RECORD [INTO 标识符] |
| [INVALID KEY 强制语句]. |

随机读取格式适用于随机或动态存取方式，执行该语句，由记录键或相对键的当前值读取文件的记录。对索引文件，由记录键的值从索引表检索该记录的存放地址，然后由该地址读取记录。

当选用INVALID KEY 子句时，文件中如果不存在由键值指定的记录，则执行INVALID KEY之后的强制执行语句，可以防止程序非正常中断。

### 写语句（WRITE语句）

语法：

WRITE 记录名[ FROM 标识符 ]

ADVANCING 强制语句……

举例：

WRITE HJDJK.

WRITE语句说明

WRITE语句中用的是记录名，而不是文件名

每执行一次WRITE语句，只能输出一条记录，不能用一条WRITE输出多条记录。

在实际应用中，可以用WRITE语句在行式打印机上，打印各种所需格式的数据表格，也可以建立磁盘文件，将一些数据作为资料存于磁盘上保存

FROM选项: WRITE 记录名 FROM 标识符

等价于： MOVE 标识符 TO 记录名

WRITE 记录名

WRITE 记录名 [FROM 标识符] [; INVALID KEY 强制语句]

说明：可以新建磁盘文件和追加记录。新建文件时候应指定顺序存取方式，并以OUTPUT方式打开，追加记录时应指定RANDOM或DYNAMIC存取方式并以I-O方式打开。

执行强制语句条件：

1.顺序存取方式时，键值不是递增的

2.文件中已存在该键值的记录

3.磁盘已满，不能存入该记录

### 重写语句REWRITE

REWRITE记录名 [FROM 标识符] [; INVALID KEY 强制语句]

指定SEQUENTIAL存取方式时，执行之前应当先执行读操作，即将记录读入内存，修改后再重新写入，如果重写的记录其记录键值与原来不同，则产生INVALID KEY 条件。

指定RANDOM或DYNMIC存取方式的文件，可以直接使用REWRITE重写文件中的记录，不必先执行读操作。

如果修改后的记录键值与原键值不同时，则执行INVALID KEY 之后的强制语句

此语句只适用于I-O方式打开的文件

重写顺序文件的记录应注意新记录必须与原记录等长。

### 删除语句DELETE

DELETE 文件名 RECORD [; INVALID KEY 强制语句]

指定顺序存取方式文件，执行该语句之前应先进行读操作

指定RANDOM或DYNMIC存取方式的文件，可以直接删除由记录键或相对键指定的记录，不必先执行读操作。

当所删除的记录键值在文件中找不到时，则执行INVALID KEY 之后的强制执行语句。

DELETE不能用于顺序文件

### 关闭语句（CLOSE语句）

语法：

CLOSE 文件名1，文件名2.

说明：

1.凡是由OPEN语句打开的文件，无论其打开方式如何，处理之后，均用CLOSE语句关闭。

2.一个CLOSE语句可以关闭一个或多个文件

3.OPEN与CLOSE是否一一对应，可根据实际需要加以选择。

### 开始语句START

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| START | 文件名 | KEY | 关系运算符 | 数据名 | [INVALID KEY 强制语句] |

用于顺序读取操作之前制定开始读取的记录，则认为产生INVALID KEY 条件，执行强制语句。

举例：一个名录，姓名为记录键，按照英文排序。

|  |  |
| --- | --- |
|  | MOVE “E” TO NAME.  START NOTABLE-FILE KEY IS NOT < NAME  INVALID KEY PERFORM ERR-P.  READ NOTABLE-FILE AT END  CALSS NOTABLE-FILE  STOP RUN. |

### 文件操作汇总

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **顺序文件** | **索引文件** | **相对文件** | **备注** |
| **Open** | **四种方式** | **三种方式** | **三种方式** |  |
| **Read** | **从开始依次读取** | **按索引键读取** | **按相对键读取** | **动态：顺序读取时可以设置开始记录** |
| **Write** | **追加记录** | **追加记录** | **追加记录** |  |
| **Rewrite** | **无** | **按索引键更新** | **按相对键更新** | **当顺序读取时应该先把原记录读入** |
| **Delete** | **无** | **按索引键更新** | **按相对键更新** | **当顺序读取时应该先把原记录读入** |
|  |  |  |  |  |

## 磁带文件

1．磁带的物理特性：磁带和卡片相比具有句路数据密度大、读写速度快、便于保存数据的优点。

2．磁带记录、块、块间间隙：磁带上的信息是以“块”为单位的，块是磁带存取的最小物理单位。每块中包含的记录数称为块化因子，则显然块化因子越大，输入输出就越少，但是所需的缓冲区就越大。

3．可变长记录：系统在每个记录前增加一个“计数字段”，用来记录本记录中的字节数。

4．磁带文件的组织形式：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BOT | 标号记录 | 数据 | … | … | 数据 | 文件尾记录 | EOT |

说明：（1）BOT：它是“磁带开始标志”的简写

（2）记录标号：也称内部标号，是一个数据块，内存文件标识、保留日期、建立日期、卷序号

（3）记录数据块。

（4）文件结尾标志

（5）EOT：一片铝片，作用是停止磁带驱动器运动防止脱带。

5．COBOL中有关磁带文件的成分

1. 标识部
2. 环境部

在输入输出节中说明：

INPUT-OUTPUT SECTION.

FILE-CONTROL

SELECT 文件名 ASSIGN TO 磁带机1，磁带机2…

[;ORGANAZATION IS SEQUENTIAL]

[;ACCESS MODE IS SEQUENTIAL]

指定多台磁带机时，当处理完一卷磁带时，系统会自动切换磁带机

1. 数据部

要在数据部中对此带文件进行描述，说明其标号记录、块化因子、记录长度等。

FD 文件名

LABEL RECORDS ARE[ RECORD IS ] STANDARD

BLOCK CONTAINS 整数1 TO整数2 RECORDS[CHARACTERS]

[RECORD CONTAINS整数3 TO整数4 CHARACTERS]

DATA RECORDS ARE[ RECORD IS ] 数据名1 ,[ 数据名2]

VALUE OF数据名3 IS数据名4[常量1]…

例： FD MT-FILE LABEL RECORDS IS STANDARD

BLOCK CONTAINS 10 RECORDS

DATA RECORD IS MT-REC

01 MT-REC.

02 PRODUCT-CODE PIC 9(8)

02 PRODUCT-NAME PIC X(5)

02 UNIT-PRICE PIC 9(4)V99.

（4） 过程部

1．OPEN语句

OPEN语句语句的一般格式

OPEN INPUT[OUTPU] 文件名 REVERSED[WITH NO REWIND]

说明：REVERSED表示“反读”，即从最后一个记录开始向前读，在执行这个语句之前应将文件定位到文件尾部，WITH NO REWIND表示磁带不反绕。

2．CLOSE语句

CLOSE语句语句的一般格式

CLOSE 文件名REEL [WITH NO REWIND[LOCK]]

说明：NO REWIND的作用和OPEN语句中的相同，LOCK可选项用于关闭文件，即关闭该文件后程序中不能再打开。REEL可选项表示关闭的是某一磁带卷，不是文件，此时文件仍处于打开状态。只有多卷文件才用REEL。

3．READ语句

READ语句语句的一般格式

READ 文件名 [INTO 数据名] AT END 强制语句

4．WRITE语句

WRITE语句语句的一般格式

WRITE 记录名 [FROM 标识符]

6.磁带文件应用举例

IDENTIFICATION DIVISION

PROGRAM-ID EXAM10-1

ENVIRONMENT DIVISION

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT IN-FILE ASSIGN TO IN-MSD

SELECT MT-FILE ASSIGN TO 磁带文件

DATA DIVISION.

FILE SECTION

FD IN-FILE LABEL RECORDS IS STANDARD

　　VALUE OF IDENTIFICATION IS "D:\DAFILE"

01 IN-REC.

02 PRODUCT-CODE PIC 9(8)

02 PRODUCT-NAME PIC X(15)

02 UNIT-PRICE PIC 9(4)V99.

FD MT-FILE LABEL RECORDS IS STANDARD

BLOCK CONTAINS 10 RECORDS

DATA RECORD IS MT-REC

01 MT-REC.

02 PRODUCT-CODE PIC 9(8)

02 PRODUCT-NAME PIC X(5)

02 UNIT-PRICE PIC 9(4)V99.

PROCEDURE DIVISION.

STA.

OPEN INPUT IN-FILE

OUTPUT MT-FILE.

AT END CLOSE IN-FILE MT-FILE

STOP RUN.

MOVE CORR IN-REC TO MT-REC

WRITE MT-REC

GO TO DISK-TYPE

## 磁盘顺序文件

### 顺序文件的概念

### COBOL中与顺序文件有关的部分

1. OPEN语句：除了用INPUT用OUTPUT方式打开文件外，还可以用I-O方式，即文件即可输入也可输出。

磁盘文件用到的OPEN语句的一般格式：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | { | INPUT | 文件名1 [，文件名2 ]… | } |  |
| OPEN | OUTPUT | 文件名3 [，文件名4 ]… |  |
|  | I-O | 文件名5 [，文件名6 ]… |  |

它没有NO REWIND等可选项

2． CLOSE语句

磁盘文件用到的CLOSE语句的一般格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CLOSE | 文件名 | [WITH LOCK] |  |  |
|  | [，文件名2 | [WITH LOCK]] |  |  |

3．READ语句

磁盘文件用到的READ语句的一般格式：

READ文件名 RECORD [INTO 标识符] [; AT END 强制语句]

4．REWRITE语句：重写语句，用相同长度的记录代替原来的记录，它只能用于用I-O方式打开的文件。在执行REWRITE语句之前，必须先用READ语句读出一条记录，对其进行修改，再用REWRITE语句写回原来位置。不能修改记录的长度和数据的类型。

5． WRITE语句

磁盘文件用到的WRITE语句的一般格式：

WRITE 记录名 [FROM 标识符] [；INVALID KEY强制语句]

当试图输出的记录超过了分配给文件的空间范围时，就执行INVALID KEY子句中的强制语句。

|  |  |  |  |
| --- | --- | --- | --- |
| 读写语句 | OPEN方式 | | |
| INPUT方式 | OUTPUT方式 | I-O方式 |
| READ | YES |  | YES |
| WRITE |  | YES |  |
| REWRITE |  |  | YES |

应用举例：

### 顺序文件实例

例：

CH124301

有一磁盘文件STOCK.DAT,记录内容与格式如下，

货物编号 X(8)

货物名称 X(20)

库存量 9(6)

单价 9(3)V99

文件中每种货物占一条记录，共100种货物，编一程序打印出每种货物的库存量及占用资金

|  |  |
| --- | --- |
|  | \* CH51\_78  IDENTIFICATION DIVISION.  PROGRAM-ID. COBOLMAIN.  \*  ENVIRONMENT DIVISION.  INPUT-OUTPUT SECTION.  FILE-CONTROL.  SELECT STOCK-FILE ASSIGN TO DISK.  SELECT PRINT-FILE ASSIGN TO DISK.  \*  DATA DIVISION.  FILE SECTION.  FD STOCK-FILE LABEL RECORD IS STANDARD  VALUE OF FILE-ID IS "STOCK.DAT".  01 STOCK-REC.  03 NUMBERS PIC X(8).  03 NAMES PIC X(20).  03 AMOUNT PIC 9(6).  03 PRICE PIC 9(3)V99.  FD PRINT-FILE LABEL RECORD IS OMITTED  VALUE OF FILE-ID IS "PRINT.DAT".  01 PRINT-REC PIC X(80).  WORKING-STORAGE SECTION.  01 P-R-W.  03 NUMBERS-O PIC X(8).  03 FILLER PIC X(8) VALUE " ".  03 NAMES-O PIC X(20).  03 FILLER PIC X(8) VALUE " ".  03 AMOUNT-O PIC Z(8).  03 FILLER PIC X(8) VALUE " ".  03 SUM-O PIC Z(8).99.  \*  PROCEDURE DIVISION.  G.  OPEN INPUT STOCK-FILE  OUTPUT PRINT-FILE.  PERFORM A 100 TIMES.  CLOSE STOCK-FILE PRINT-FILE.  STOP RUN.  A.  READ STOCK-FILE.  MULTIPLY PRICE BY AMOUNT GIVING SUM-O.  MOVE NUMBERS TO NUMBERS-O.  MOVE NAMES TO NAMES-O.  MOVE AMOUNT TO AMOUNT-O.  MOVE P-R-W TO PRINT-REC.  WRITE PRINT-REC. |

CH120432

有一个磁盘文件PRT.DAT为某公司100种固定资产帐目，记录为：

财产号 9(4)

购置年度 9(4)

原价 9(6)V99

使用年限 9(3)

残值 9(4)V99

假定所有资产都不到年限，根据文件打印出各项资产的资产号，购置年度，原价，现价值及公司资产总价值。

例：

CH120433

原有某种货物价格记录50条，记录格式为

编号 9(5)

单价 9(4)

说明 X(40)

又有20条新记录。原有记录和新记录分别存于磁盘上。设两个数据文件均已按单价降序排好。编制一个程序把2个数据文件合在一起

|  |  |
| --- | --- |
|  |  |

例

CH120434

磁盘中存放若干外贸公司年销售额，前12条记录为第一个外贸公司1-12月的销售额，接下去12条记录为第二个外贸公司的销售额，每条记录的格式如下：

1-20 C-NAME X(20)

21-25 AMOUNT 9(3)V99

编写程序，求出每个外贸公司全年销售额总和，并打印出如下表格：

2006 YEAR SALE TOTAL PAGE NO.1

PUBLICATIONS CORP $\*\*,\*\*\*.\*\*

TEXTILES CORP $\*\*,\*\*\*.\*\*

## 磁盘索引文件

### 索引文件的概念

磁盘索引文件包括两个文件，由主文件记录构成的数据文件和作为索引用的键文件。在建立索引文件时，应将文件记录按索引排序，之后输入到文件中。

给关键字赋值的方法：

1. 直接给作为RECORD KEY 的数据项赋值。
2. 在建立索引文件时指定RECORD KEY，而在存取检索时，另外指定一个NOMINAL KEY（名义键）。

例：RECORD KEY IS PRODUCT-CODE NOMINAL KEY IS A .

这里A是在数据部中定义的初等项，名字是任意的，只要给A赋值就可以了。

名义键可以在工作单元节中定义，而且它的PIC描述必须和记录键相同。

索引文件分为两类：索引顺序文件和索引非顺序文件。索引顺序文件比索引非顺序文件查找效率高

（1）RECORD KEY指定的数据项必须是索引文件记录的一部分。

（2）在建立索引顺序文件时，应用顺序存储方式，在向文件中写记录时，后一个记录的索引值一定要大于前一个的索引值。

（3）记录键的描述体不能包含OCCURS子句，也不能从属于OCCURS子句的数据项。

（4）RECORD KEY和NOMINAL KEY应当有相同的数据描述。

（5）作为NOMINAL KEY的数据项不能出现在文件的记录中。

### COBOL中与索引文件有关的部分

（1）标识部

（2）环境部：

SELECT 文件名 ASSIGN TO 磁盘机名

ORGANIZATION IS INDEXED

[; ACCESS MODE IS SEQUENTIAL ]

RANDOM

RECORD KEY IS 数据名1

[MONINAL KEY IS数据名2]

说明：（1）ORGANIZATION IS INDEXED说明文件组织形式是是索引文件。

（2）ACCESS MODE IS RANDOM 说明存取方式是随机的。

（3）RECORD KEY IS 数据名1说明以数据名1为“记录键”

（4）MONINAL KEY IS数据名2说明以数据名2为“名义键”

（3）数据部：作为RECORD KEY IS和MONINAL KEY IS的数据应在数据部中说明。

（4）过程部

（一）OPEN和CLOSE语句与磁盘顺序文件相同。

（二）READ语句

READ 文件名RECORD [INTO 数据名] ; INVALID KEY 强制语句

在随机读取记录时，必须先向RECORD KEY （或MONINAL KEY）提供一个值，所以不会遇到文件结束标志。

（三）START语句

START语句只用于顺序读一个索引文件时指定顺序读的起点。

START语句的一般形式：

START 文件名 KEY 关系运算符 数据名 [INVALID KEY 强制语句]

注意：“数据名”应是已被指定为“记录键”的数据项名。事先一定要给数据名赋值。

（四）WRITE语句

WRITE 记录名 [FROM 标识符] [;INVALID KEY 强制语句]

WRITE语句只能向索引文件中写入一个新记录，而不能更新一个记录。

（五）REWRITE语句

REWRITE记录名 [FROM 标识符] [;INVALID KEY 强制语句]

（六）DELETE语句

DELETE 文件名 RECORD [;INVALID KEY 强制语句]

### 建立索引文件

例：CH120531

建立某公司库存商品数据文件，文件组织形式为索引文件，记录包括以下内容

商品编号 6位整数

商品名称 20个字符

库存数量 6位整数

单价 4位整数2位小数

单位名称 8个字符

以商品编号为记录键

操作步骤：

打开文件

输入数据到记录区，写记录

关闭文件

结束

|  |  |
| --- | --- |
|  | \* ch92-159  IDENTIFICATION DIVISION.  PROGRAM-ID. MAIN.  ENVIRONMENT DIVISION.  INPUT-OUTPUT SECTION.  FILE-CONTROL.  SELECT STOCK-FILE ASSIGN TO DISK  ORGANIZATION IS INDEXED  ACCESS MODE IS SEQUENTIAL  RECORD KEY IS NUMBERS.  DATA DIVISION.  FILE SECTION.  FD STOCK-FILE LABEL RECORD IS STANDARD  VALUE OF FILE-ID "STOCK.DAT".  01 STOCK-REC.  03 NUMBERS PIC 9(6).  03 NAME PIC X(20).  03 AMOUNT PIC 9(6).  03 PRICE PIC 9(4)V99.  03 UNITT PIC X(8).  PROCEDURE DIVISION.  G.  OPEN OUTPUT STOCK-FILE.  DISPLAY "INPUT NUMBERS".  ACCEPT NUMBERS.  PERFORM B UNTIL NUMBERS = 999998.  CLOSE STOCK-FILE.  STOP RUN.  B.  DISPLAY "INPUT NAMES".  ACCEPT NAME.  DISPLAY "INPUT AMOUNT".  ACCEPT AMOUNT.  DISPLAY "INPUT PRICS".  ACCEPT PRICE.  DISPLAY "INPUT UNITT".  ACCEPT UNITT.  WRITE STOCK-REC INVALID KEY  DISPLAY "NUMBERS IS ERROR".  DISPLAY "INPUT NUMBERS".  ACCEPT NUMBERS. |

### 检索索引文件记录

检索

从文件记录中查询某一或某些记录，可以用顺序查询，也可以随机查询。

顺序读取可以从第一条记录开始读，和处理顺序文件相同，也可以用START语句指定从中间某一记录开始读。

随机读取可以由给定的键值直接读取记录。

例：CH120541

有一个磁盘索引文件，为某外贸公司商品库存记录，记录键为商品编号，由商品编号查询库存数量

|  |  |
| --- | --- |
|  | \*CH91-153  IDENTIFICATION DIVISION.  PROGRAM-ID. MAIN.  ENVIRONMENT DIVISION.  INPUT-OUTPUT SECTION.  FILE-CONTROL.  SELECT STOCK-FILE ASSIGN TO DISK  ORGANIZATION IS INDEXED  ACCESS MODE IS RANDOM  RECORD KEY IS NUMBERS.  DATA DIVISION.  FILE SECTION.  FD STOCK-FILE LABEL RECORD IS STANDARD  VALUE OF PILE-ID IS "STOCK.DAT".  01 STOCK-REC.  03 NUMBERS PIC 9(6).  03 NAME PIC X(20).  03 AMOUNT PIC 9(6).  03 PRICE PIC 9(4)V99.  03 SUNIT PIC X(8).  WORKING-STORAGE SECTION.  77 EOF PIC X(3) VALUE ALL SPACE.  01 STOCK-OUT.  03 NUMBERS-0 PIC 9(6).  03 FILLER PIC X(5) VALUE ALL SPACES.  03 NAME-0 PIC X(20).  03 FILLER PIC X(5) VALUE ALL SPACES.  03 AMOUNT-0 PIC Z(6).  PROCEDURE DIVISION.  G.  OPEN INPUT STOCK-FILE.  MOVE ZERO TO NUMBERS.  PERFORM A UNTIL NUMBERS = 999999.  CLOSE STOCK-FILE.  STOP RUN.  A.  MOVE SPACE TO EOF.  DISPLAY "INPUT NUMBERS".  ACCEPT NUMBERS.  IF NUMBERS NOT = 999999 PERFORM R.  R.  READ STOCK-FILE INVALID KEY MOVE  "ERR" TO EOF.  IF EOF NOT = "ERR" PERFORM O  ELSE DISPLAY "NUMBERS IS ERROR".  O.  MOVE NUMBERS TO NUMBERS-0.  MOVE NAME TO NAME-0  MOVE AMOUNT TO AMOUNT-0.  DISPLAY STOCK-OUT. |

### 新增索引文件记录

例：新增品种入库

CH120551

|  |  |
| --- | --- |
|  | \* CH93-161  IDENTIFICATION DIVISION.  PROGRAM-ID. MAIN.  ENVIRONMENT DIVISION.  INPUT-OUTPUT SECTION.  FILE-CONTROL.  SELECT STOCK-FILE ASSIGN TO DISK  ORGANIZATION IS INDEXED  ACCESS MODE IS RANDOM  RECORD KEY IS NUMBERS.  DATA DIVISION.  FILE SECTION.  FD STOCK-FILE LABEL RECORD IS STANDARD  VALUE OF FILE-ID IS "STOCK.DAT".  01 STOCK-REC.  03 NUMBERS PIC 9(6).  03 NAME PIC X(20).  03 AMOUNT PIC 9(6).  03 PRICE PIC 9(4)V99.  03 UNIT1 PIC X(8).  WORKING-STORAGE SECTION.  77 EOF PIC X(3) VALUE ALL SPACE.  PROCEDURE DIVISION.  G.  OPEN I-O STOCK-FILE.  DISPLAY "INPUT NUMBERS".  ACCEPT NUMBERS.  PERFORM A UNTIL NUMBERS = 999999.  CLOSE STOCK-FILE.  STOP RUN.  A.  READ STOCK-FILE INVALID KEY MOVE "YES " TO EOF.  IF EOF = "YES" PERFORM A1  ELSE DISPLAY "THIS NUMBERS IS EXIT,RETRY"  DISPLAY "INPUT NUMBERS"  ACCEPT NUMBERS.  A1.  DISPLAY "INPUT NAMES".  ACCEPT NAME.  DISPLAY "INPUT AMOUNT".  ACCEPT AMOUNT.  DISPLAY "INPUT PRICE".  ACCEPT PRICE.  DISPLAY "INPUT UNIT".  ACCEPT UNIT1.  WRITE STOCK-REC.  MOVE SPACE TO EOF. |

### 修改索引文件记录

方法：将记录重新输入到记录缓冲区，使用REWRITE语句重写磁盘文件中具有相同键值的记录

修改库存商品的信息

CH120561

### 删除索引文件记录

对顺序存取方式先用READ语句顺序读取记录，当读出应删除的记录时用DELETE语句删除

对随机或动态存取方式，可给定记录键值，直接删除文件中对应的记录

### 综合实例

CH120406

某公司接到一批客户订单，单据格式如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 订货单 2006231 | | | | | | |
| 年 月 日 | | | | | | |
| 客户名称 |  | | | 地址 |  | |
| 电话 |  | 开户银行 |  | | 账号 |  |
| 产品名称 |  | | | 数量 |  | |

建立数据文件ORDER.DAT,记录为

货单编号 9(7)

客户名称 X(20)

产品名称 X(15)

数量 9(6)

订货日期 9(6)

同时检查客户登记文件是否有该客户登记，如果没有则增加该客户登记，当客户登记文件有增加记录时，输入订货单结束后打印出一张新增客户清单。

IDENTIFICATION DIVISION.

PROGRAM-ID.EXAM10-3.

ENVIROMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT IN-FILE ASSIGN TO IN-FILE。

SELECT DA-FILE ASSIGN TO INDE

ORGANIZATION IS INDEXED

ACCESS MODE IS SEQUENTIAL

RECORD KEY IS DA-PRODUCT-CODE.

DATA DIVISION.

FILE SECTION.

FD IN-FILE LABEL RECORD IS STANDARD

DATA RECORD IS INREC.

01 INREC

02 IN-PEODUCT-CODE PIC X(4).

02 IN-PRODUCT-NAME PIC X(10).

02 IN-UNIT-PRICE PIC 9(7).

FD DA-FILE LABEL RECORD IS STANDARD

DATA RECORD IS DAREC.

01 DAREC

02 DA-PEODUCT-CODE PIC X(4).

02 DA-PRODUCT-NAME PIC X(10).

02 DA-UNIT-PRICE PIC 9(7).

WORKING-STRAGE SECTION.

77 END-SWITCH PIC X(4).

PROCEDURE DIVISION.

START-RUN.

OPEN INPUT INFILE

OUTPUT DA-FILE.

MOVE SPACE TO END-SWITCH.

READ IN-FILE

AT END MOVE HIGH-VALE TO END-SWITCH.

PERFORM RECORD-PROCESSING THRU RECORD-EXIT

UNTIL END-SWITCH = HIGH-VALUE.

CLOSE IN-FILE , DA-FILE.

STOP RUN.

RECORD-PROCESSING.

MOVE IN-PEODUCT-CODE TO DA-PEODUCT-CODE.

MOVE IN-PEODUCT-NAME TO DA-PEODUCT-NAME.

MOVE IN-UNIT-PRICE TO DA- UNIT-PRICE.

WRITE DAREC INVALID KEY

DISLPAY ‘INVALID KY INPUT = ’, DAREC.

READ IN-FILE AT END

MOVE HIGH-VALUE TO END-SWITCH.

RECORD-EXIT.

EXIT.

索引文件应用—建立文件

CH92-159

某公司库存商品数据文件，文件组织形式为索引文件，记录包括以下内容：

商品编号 6位整数

商品名称 20个字符

库存数量 6位整数

单价 4位整数2位小数

单位名称 8个字符

以商品编号为记录键

索引文件应用—检索

* 顺序读取

说明：ACCESS MODE IS SEQUENTIAL

顺序读取可以从第一条记录开始读，和处理顺序文件相同，也可以START语句指定从中间的某一记录开始读。

MOVE 600000 TO NUMBERS.

START STOCK-FILE KEY IS NOT < TO NUMBERS.

* 随机读取

随机读取可以由给定的键值直接读取记录。

MOVE 000128 TO NUMBERS.

READ STOCK-FILE INVALID KEY……

索引文件应用—新增、修改、删除

* 新增记录

CH93-161

* 修改记录

CH94-163

* 删除记录

CH95-165

相对文件应用CH96-168

## 磁盘相对文件

### 相对文件的概念

1．相对文件的概念

所谓相对文件就是在建立文件时，除了记载记录本身之外，还给每一个记录编一个“位置号”。以后按位置号存取记录。还要定义一个“相对键”。

### COBOL中与相对文件有关的成分

环境部：

SELECT 文件名 ASSIGN TO 设备名

ORGANIZATION IS RELATIVE

ACCESS MODE IS SEQUENTIAL [, RELATIVE KEY IS 数据名1]

RANDOM RELATIVE KEY IS数据名2

过程部的使用与索引文件相同。

说明：（1）相对文件可以顺序读，此时可以不指定RELATIVE KEY。

（2）作为RELATIVE KEY的数据项不能是记录中的一项。

（3）随机读时应事先给“相对键”项送入一个值。

（4）WRITE语句可以向文件中写入一条新记录，但也能在空记录中写入

（5）用WRITE建立一个新的相对文件时，即可以顺序写，也可以随机械。

（6）REWRITE语句用于更新一条记录。

（7）DELETE语句用于删除一条记录。

相对文件的处理

读文件记录可以顺序读也可以随机读，顺序读应声明存取方式为 SEQUENTIAL，可以由文件第一条记录开始读，也可以由START语句指定由中间的某一记录开始读。顺序读时如果遇到空记录则自动跳过，直到读出一条实记录。

相对文件的随机读取应声明存取方式为RANDOM或DYNAMIC，执行读操作前应先对相对键赋值。

相对文件记录的增加、修改、删除操作与索引文件类似。

### 建立相对文件

CH120631

## 动态存取

用动态存取方法，应在设备部的SELECT子句中将ACCESS子句改写为：

ACCESS MODE IS DYNAMIC

在读取文件时应使用： READ 文件名 [NEXT] RECORD [INTO 标识符] [; AT END 强制语句]

例： MOVE 10 TO RE-KEY

START DAFILE KEY > RE-KEY

INVALID KEY GO TO ERR.

READ DAFILE NEXT RECORD

AT END GO TO TERM.

注意：（1）READ语句有两种形式，而对于WRITE,REWRITE和DELETE语句，没有NEXT可选项，只能用随机方式处理

（2）对索引文件的READ NEXT语句的用法也是一样的。

## 文件综合应用

例：CH120301

建立一个磁盘文件，用以存放2000户的户籍登记卡，磁盘文件名为TBD.DAT，记录格式：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 户籍登记卡（HJDJK） | | | | | |
| 主机姓名  （HZXM） | 性别  （XB） | 出生日期  （CSRQ） | 人口  （RK） | 住址  （ZZ） | 工作单位  (GZDW) |

步骤：

1.打开文件

2.执行输入数据段

3.关闭文件

4.结束程序

输入数据段

1.输入登记卡片项目内容

2.写入磁盘

### 与表操作结合实例

CH120811

在文件中打印

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*\*

## 本章小结

环境部：定义程序用到的输入/输出设备，如数据文件或者打印机等的描述。

语法：

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

\*

FILE-CONTROL.

SELECT 入力ファイル ASSIGN TO U01

FILE STATUS IS Ｗ－状態

ORGANIZATION IS LINE SEQUENTIAL.

\*

SELECT 入力ファイル１ ASSIGN TO U11

FILE STATUS IS Ｗ－状態

ORGANIZATION IS LINE SEQUENTIAL.

\*

SELECT 出力ファイル２ ASSIGN TO PRINTER

FILE STATUS IS Ｗ－状態

ORGANIZATION IS SEQUENTIAL.

ASSIGN TO的文件名可以是磁盘上的实际文件名，也可以是逻辑文件名，然后在执行配置文件中指定实际磁盘文件，使用逻辑文件名的

是改变文件名时不用修改源程序。

数据部：定义程序用到的全部变量，包括访问数据文件的纪录、访问数据库表的变量，程序用的内存变量等。

文件节：定义输入输出文件用到的变量

FILE SECTION.

FD 入力ファイル

LABEL RECORD IS STANDARD

BLOCK CONTAINS 0 RECORDS.

\*

01 入力－レコード.

COPY CISUF470 REPLACING ==()== BY ==入力１－==.

\*

FD 出力ファイル１

LABEL RECORD IS STANDARD

BLOCK CONTAINS 0 RECORDS.

\*

01 出力１－レコード.

COPY CISUF470 REPLACING ==()== BY ==出力１－==.

\*

FD 出力ファイル２.

\*

01 行レコード PIC N(136).

01 注釈レコード PIC N(050).

01 制御レコード PIC X(100).

FD是文件描述的缩写。每个记录表示数据文件的一行数据，一个文件由多个记录组成。（文件→纪录→数据项的层次结构）

# 排序

**学习提示**

结合实例Cobol特有的文件排序语句。

## 基本概念

在COBOL中称排序项为“排序键”。所谓“键”，指的是“关键字”，即按哪个关键字作为排序的依据。除了指定“排序键”以外，还要指出是按升序还是按降序排列。

实现排序的步骤：

1．建立排序中间工作文件：将输入文件中的记录依次输入到中间工作文件中，直到全部输完为止。

2．对排序中间文件的各记录，按指定的排序键和升降序进行排序。排序由计算机自动进行。排序的对象是中间文件，不是输入文件。

3．将排好序的中间文件输出，由于中间文件是程序为了排序而临时建立的文件，当程序运行结束后，该文件被撤销，所以要输出到其它介质上。

## 环境部中排序文件的描述

由于排序需要三个文件（输入文件、排序中间文件，输出文件），所以在环境部中要分别对三个文件进行描述

排序中间文件的组织形式应制定为顺序的，存取方式也应是顺序的

例如：

FILE-CONTROL.

SELECT IN-FILE ASSIGN TO IN-FILE.

SELECT SORT-WORK-FILE ASSIGN TO SW.

SELECT SORTED-FILE ASSIGN TO SORT-FILE.

SORT-WORK-FILE是排序中间文件，名字任意取

## 数据部中排序文件描述

在数据部中，要对每一个文件进行数据描述。输入输出文件的描述不变。排序中间文件的描述体以SD开头。不必指定LABEL RECORD IS STANDARD ，也不用组块。

例： SD SORT-WORK-FILE DATA RECORD IS SORT-REC

语法：

SD 排序中间文件名

DATARECORD IS 数据名1………

RECPRDS ARE

对排序中间文件不用指出LABEL子句，对排序中间文件记录的各数据项的描述一般应当和输入文件中各数据项的描述相同，因为输入文件记录中各数据项的值要直接传送到中间文件中

排序中间文件举例

FD IN-FILE LABEL RECORD IS STANDARD

DATA RECORD IS INREC.

01 INREC.

02 YY-MM-DD PIC 9(6).

02 CUSTOMER-NUM PIC 9(8).

02 CUSTOMER-NAME PIC X(10).

02 PRODUCT-CODE PIC X(6).

02 PRODUCT-NAME PIC X(10).

02 QTY PIC 9(6).

02 AMOUNT PIC 9(8)V99.

SD SORT-WORK-FILE.

01 WORKREC.

02 YY-MM-DD-S PIC 9(6).

02 CUSTOMER-NUM-S PIC 9(8).

02 CUSTOMER-NAME-S PIC X(10).

02 PRODUCT-CODE-S PIC X(6).

02 PRODUCT-NAME-S PIC X(10).

02 QTY-S PIC 9(6).

02 AMOUNT-S PIC 9(8)V99.

## 排序语句（SORT）单纯排序

单纯排序，就是对输入文件的所有记录进行排序，并且每个记录的内容原封不动，即对输入文件的各个记录不作选择，也不改变其结构或内容。

语法：

SORT 排序中间文件名

ON ASCENDING KEY 数据名1 [，数据名2]…

DESCENDING

ON ASCENDING KEY 数据名3 [，数据名4]…

DESCENDING

USING 输入文件名

GIVING 输出文件名

语法说明：

* SORT语句指出排序过程中间文件
* ON指出排序键实现排序的顺序，
* KEY是排序的排序键，数据名1为主排序键，以后按数据名出现先后次序，最多可设12个排序键，升序和降序可以交替出现，排序间可以是初等数据项，也可以是组合数据项，但几个排序间之间不能互相重叠。作为排序键的数据项描述中不能含有OCCURS子句
* USING 指出输入文件名，自动实现输入过程，然后自动传送到中间文件的纪录区，再自动写入排序中间文件中去
* GIVING 指出输出文件名，自动实现输出过程。

单纯排序举例

要求按学生所在系科进行升序排序，同一系科中再按高低分排列。 EXAMPLAE71-249

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据名 | NO | NAME | PACULTY | SCORE1 | SCORE2 | SCORE3 | TOTAL |
| 名称 | 学号 | 姓名 | 系科 | 成绩1 | 成绩2 | 成绩3 | 总分 |
| PIC描述 | 999 | X(30) | 9(2) | 9(3) | 9(3) | 9(3) | 9(4) |

排序语句（SORT）加工排序执行次序

* （1）先执行“输入过程”，按特定要求生成排序中间文件
* （2）再对排序中间文件进行排序
* （3）最后执行“输出过程”，按我们特定要求生成一有序的输出文件。

## 排序语句（SORT）加工排序

如果需要对输入的记录进行某些加工，然后再按指定的某些数据项的顺序排列，则需使用SORT语句的第二种形式。

语法：

SORT 排序中间文件名

ON ASCENDING KEY 数据名1 [，数据名2]…

DESCENDING

ON ASCENDING KEY 数据名3 [，数据名4]…

DESCENDING

INPUT PROCEDURE IS 节名1 THRU 节名2

OUTPUT PROCEDURE IS 节名3 THRU 节名4

INPUT PROCEDURE 和OUTPUT PROCEDURE 具有自动返回功能

注意：“输入过程”和“输出过程”都必须指定节名。

在“输入过程”中可以对尚待排序的记录进行加工，甚至可以改变排序键值；排序好以后可以对记录进行加工。“输入/输出过程”中的语句只能在本过程中控制转移，不能跳出本过程范围之外，也不能从本过程以外转入。

RELEASE释放语句：把中间文件记录区中的记录从内存中送到排序中间文件中去。它的作用相当于WRITE语句。但在这里只能用RELEASE语句。

一般格式： RELEASE 记录名 [FROM 标识符]

若排序中间文件名的记录名是SORT-RECORD，输入文件的记录名是INPUT-RECORD，则该命令形式如下：

RELEASE SORT-RECORD FORM INPUT-RECORD

RETURN回收语句：从排序中间文件读回一个记录到内存。相当于READ语句。

一般形式：RETURN 排序中间文件名 RECORD [INTO标识符] [；AT END 强制语句]

若排序中间文件名为SORT-WORK-FILE，输出文件记录名OUTPUT-RECORD，则该命令可写成

RETURN SORT-WORK-FILE INTO OUTPUT-RECORD

AT END GO TO SORT-END。

RELEASE语句和RETURN语句分别出现在SORT语句指定的输入过程和输出过程中，是在SORT语句的控制下进行的，对其他文件不能使用，他的工作是：从内存到排序中间文件叫“释放”，从排序中间文件到内存叫“回收”。

1. 处理过程

输入过程：

* 按输入方式打开输入文件
* 顺序读入输入文件的记录至内存输入文件记录区
* 对每一读入的输入文件记录进行加工处理
* 把加工处理后的输入文件记录送至排序中间文件的内存记录区
* 把内存中排序中间文件记录区内容用RELEASE语句写入排序中间文件
* 处理完所有输入文件记录以后关闭输入文件

输出过程

* 按输出方式打开输出文件
* 用RETURN语句顺序读入已排好的排序中间文件的各记录至内存排序中间文件记录区
* 对每一个回收的记录进行加工处理，并送入输出文件内存记录区
* 把输出文件记录区内容写入输出文件
* 处理完所有的排序中间文件记录以后关闭输出文件

本章小结

# 合并

学习提示

## 基本概念

MERGE语句

如果有一组已按相同的排序原则排好序的文件，要求将它们合并成为一个文件，这叫“合并”。

MERGE 文件名1

ON ASCENDING KEY 数据名1 [，数据名2]…

DESCENDING

ON ASCENDING KEY 数据名1 [，数据名2]…

DESCENDING

USING 文件名2 ，文件名3 …

GIVING文件名4

例： MERGE MERGE-WORK-FILE

DESCENDING KEY IS AVGE

USING GRADE-FILE1 , GRADE-FILE2 ,

GRADE-FILE3 , GRADE-FILE4

GIVING GRADE-FILE-ALL.

合并语句说明

（1）MERGE语句中各成分的含义于SORT语句中的基本相同。但合并的文件必须已经按照同样的排序要求完成了排序。

（2）MERGE语句不能用输入过程，只能用USING。

（3）输入文件的个数不得小于两个。

（4）“文件名1”是排序合并的中间工作文件，应在数据部中的排序文件描述体中描述，其文件描述方法与排序语句中的排序中间文件相同。

（5）各输入文件和输出文件，排序中间工作文件的记录区大小应该相同。

（6）输出方式有两种：自动输出GIBING和OUTPUT PROCEDURE两种方式。

（7）命令中用到的输入文件，合并中间文件的打开和关闭动作也是通过系统自动执行，不必进行人为的OPEN和CLOSE

排序与合并

一、排序的概念

在COBOL中称排序项为“排序键”。所谓“键”，指的是“关键字”，即按哪个关键字作为排序的依据。除了指定“排序键”以外，还要指出是按升序还是按降序排列。

二、实现排序的步骤

1．建立排序中间工作文件：将输入文件中的记录依次输入到中间工作文件中，直到全部输完为止。

2．对排序中间文件的各记录，按指定的排序键和升降序进行排序。排序由计算机自动进行。排序的对象是中间文件，不是输入文件。

3．将排好序的中间文件输出，由于中间文件是程序为了排序而临时建立的文件，当程序运行结束后，该文件被撤销，所以要输出到其它介质上。

三、COBOL中与排序有关的成分

1. 由于排序需要三个文件，所以在环境部中要分别对三个文件进行描述。
2. 在数据部中，要对每一个文件进行数据描述。输入输出文件的描述不变。排序中间文件的描述体以SD开头。不必指定LABEL RECORD IS STANDARD ，也不用组块。

例： SD SORT-WORK-FILE DATA RECORD IS SORT-REC

1. 过程部中主要是SORT语句。

四、SORT语句的第一种形式(SAMPLE11-1)

如果一个文件需要排序，可用这种方法：

例：IDENTIFICATION DIVISION.

PROGRAM-ID.EXAM11-1.

ENVIROMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT IN-FILE ASSIGN TO IN-FILE.

SELECT SORT-WORK-FILE ASSIGN TO SW.

SELECT SORTED-FILE ASSIGN TO SORT-FILE.

DATA DIVISION.

FILE SECTION.

FD IN-FILE LABEL RECORD IS STANDARD

DATA RECORD IS INREC.

01 INREC.

02 YY-MM-DD PIC 9(6).

02 CUSTOMER-NUM PIC 9(8).

02 CUSTOMER-NAME PIC X(10).

02 PRODUCT-CODE PIC X(6).

02 PRODUCT-NAME PIC X(10).

02 QTY PIC 9(6).

02 AMOUNT PIC 9(8)V99.

SD SORT-WORK-FILE.

01 WORKREC.

02 YY-MM-DD-S PIC 9(6).

02 CUSTOMER-NUM-S PIC 9(8).

02 CUSTOMER-NAME-S PIC X(10).

02 PRODUCT-CODE-S PIC X(6).

02 PRODUCT-NAME-S PIC X(10).

02 QTY-S PIC 9(6).

02 AMOUNT-S PIC 9(8)V99.

FD SORTED-FILE LABEL RECORD IS STANDARD.

01 SORTEDREC PIC X(80).

PROCEDURE DIVISION.

SORTING.

SORT SORT-WORK-FILE

ON ASCENDING KEY CUSTOMER-NUM-S

YY-MM-DD-S

ON DESCENDING KEY AMOUNT-S

USING IN-FILE

GIVING SORTED-FILE.

STOP RUN.

注意：不必在过程部中打开或关闭，输入输出文件，它们是自动实现的。作为排序的项不能含OCCURS项，也不能从属于含有OCCURS子句的数据项。排序项可以是组合项，但几个排序项之间不能互相重叠。输入文件和输出文件必须是顺序文件。三个文件的记录区长度一定要相等。

SORT语句的一般形式1

SORT 排序中间文件名 ON ASCENDING KEY 数据名1 [，数据名2]…

DESCENDING

ON ASCENDING KEY 数据名3 [，数据名4]…

DESCENDING

USING 输入文件名

GIVING 输出文件名

五、SORT语句的第二种形式(SAMPLE11-2)

如果需要对输入的记录进行某些加工，然后再按指定的某些数据项的顺序排列，则需使用SORT语句的第二种形式。

这种排序中要用到的一些辅助语句：

（1）RELEASE语句：把中间文件记录区中的记录从内存中送到排序中间文件中去。它的作用相当于WRITE语句。但在这里只能用RELEASE语句。

一般格式： RELEASE 记录名 [FROM 标识符]

（2）RETURN语句：从排序中间文件读回一个记录到内存。相当于READ语句。

一般形式：RETURN 排序中间文件名 RECORD [INTO标识符]

[；AT END 强制语句]

例：RETURN SORT-WORK-FILE AT END GO TO TERM.

应用举例：IDENTIFICATION DIVISION.

PROGRAM-ID.EXAM11-2.

ENVIROMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT IN-FILE ASSIGN TO IN-FILE.

SELECT SORTFILE ASSIGN TO SW.

SELECT DAFILE ASSIGN TO DA-FILE.

SELECT OUTFILE ASSIGN TO OUT-FILE.

DATA DIVISION.

FILE SECTION.

FD INFILE LABEL RECORD IS STANDARD

DATA RECORD IS INREC.

01 INREC.

02 WORKING-NUM PIC 9(6).

02 SHOP-NUM PIC 9(2).

02 ACTUAL-HOURS PIC 9(3)V9.

02 OVERTIME PIC 9(3)V9.

SD SORTFILE .

01 SORTREC.

02 WORKING-NUM PIC 9(6).

02 SHOP-NUM PIC 9(2).

02 ACTUAL-HOURS PIC 9(3)V9.

02 OVERTIME PIC 9(3)V9.

FD DAFILE LABEL RECORD IS STANDARD.

01 DAREC.

02 WORKING-NUM PIC 9(6).

02 SHOP-NUM PIC 9(2).

02 ACTUAL-HOURS PIC 9(3)V9.

02 OVERTIME PIC 9(3)V9.

FD OUTFILE LABEL RECORD IS STANDARD.

01 OUTREC.

02 FILLER PIC X(4).

02 WORKING-NUM PIC 9(6).

02 FILLER PIC X(4).

02 SHOP-NUM PIC 9(2).

02 FILLER PIC X(4).

02 ACTUAL-HOURS PIC Z(3)V9.

02 FILLER PIC X(4).

02 OVERTIME PIC Z(3)V9.

PROCEDURE DIVISION.

MAIN-PROC SECTION.

SORT-PROC.

SORT SORTFILE

ASCENDING KEY SHOP-NUM OF SORTREC

DESCENDING KEY OVERTIME OF SORTREC

INPUT PROCEDURE IS RECORD-SELECTION

OUTPUT PROCEDURE IS OUT.

STOP RUN.

RECORD-SELECTION SECTION.

OPEN INPUT INFILE.

SELECTION-PROC.

READ INFILE

AT END CLOSE INFILE GO TO SELECTION-END.

IF OVERTIME OF INFILE NOT < 20.0

MOVE CORR INREC TO SORTREC

RELEASESORTREC.

GO TO SELECTION-PROC.

SELECTION-END.

EXIT.

OUT SECTION.

OUT-OPEN.

OPEN OUTPUT DAFILE , OUTFILE.

RETURN-PROC.

RETURN SORTFILE

AT END CLOSE DAFILE , OUTFILE

GO TO OUT-END.

MOVE SPACE TO OUTREC.

MOVE CORR SORTREC TO OUTREC.

WRITE OUTREC AFTER 2.

IF OVERTIME OF SORTREC NOT < 60.0

WRITE DAREC FROM SORTREC.

GO TO RETURN-PROC

OUT-END.

EXIT.

注意：“输入过程”和“输出过程”都必须指定节名。

在“输入过程”中可以对尚待排序的记录进行加工。，甚至可以改变排序键值；排序好以后可以对记录进行加工。

“输入/输出过程”中的语句只能在本过程中控制转移，不能跳出本过程范围之外，也不能从本过程以外转入。

SORT语句的一般形式2

SORT 排序中间文件名 ON ASCENDING KEY 数据名1 [，数据名2]…

DESCENDING

ON ASCENDING KEY 数据名3 [，数据名4]…

DESCENDING

INPUT PROCEDURE IS 节名1 THRU 节名2

OUTPUT PROCEDURE IS 节名3 THRU 节名4

六、MERGE语句(SAMPLE11-3)

如果有一组已按相同的排序原则排好序的文件，要求将它们合并成为一个文件，这叫“合并”。

MERGE语句的一般格式

MERGE 文件名1 ON ASCENDING KEY 数据名1 [，数据名2]…

DESCENDING

ON ASCENDING KEY 数据名3 [，数据名4]…

DESCENDING

USING 文件名2 ，文件名3 …

GIVING文件名4

例： MERGE MERGE-WORK-FILE

DESCENDING KEY IS AVGE

USING GRADE-FILE1 , GRADE-FILE2 , GRADE-FILE3 , GRADE-FILE4

GIVING GRADE-FILE-ALL.

说明：（1）MERGE语句中各成分的含义于SORT语句中的基本相同。但合并的文件必须已经按照同样的排序要求完成了排序。

（2）MERGE语句不能用输入过程，只能用USING。

（3）输入文件的个数不得小于两个。

（4）“文件名1”是排序合并的中间工作文件，应在数据部中的排序文件描述体中描述。

（5）各输入文件和输出文件，排序中间工作文件的记录区大小应该相同。

知识小结

# 画面

学习提示

IDENTIFICATION DIVISION.

PROGRAM-ID.MAINMENU.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

LINKAGE SECTION.

01 CALL-COUNT PIC 99.

01 MENU-NO-X.

03 MENU-NO PIC XX.

SCREEN SECTION.

01 WELCOME-PAGE.

03 BLANK SCREEN.

03 LINE 8 COLUMN 5 VALUE "MAINMENU1".

03 LINE 9 COLUMN 5 VALUE "MAINMENU2".

03 LINE 18 COLUMN 5 PIC XX TO MENU-NO-X.

PROCEDURE DIVISION USING CALL-COUNT, MENU-NO-X.

MAIN-PROCESS.

ADD 1 TO CALL-COUNT.

DISPLAY WELCOME-PAGE.

ACCEPT WELCOME-PAGE.

IF MENU-NO IS NUMERIC

THEN

DISPLAY "您输入的是数字", MENU-NO

ELSE

DISPLAY "您输入的是文字", MENU-NO

END-IF.

DISPLAY MENU-NO-X.

END-PROGRAM.

EXIT PROGRAM .

本章小结

# 报表

**学习提示**

## 报表概述

报表是从一些烈记录产生打印报表，报表的基础是顺序文件，文件可以按一个或几个字段排序和组合，字段称为控制字段。

报告程序打印文件中的数据之前，第一页上有一些信息标示这个报表，称为报表头，可以包括报表名，日期等等。

第一组数据前，先打印标题记录，称为控制头。

然后一次一个记录，格式化报表，处理每个记录时，程序可以根据各个字段累计一些汇总信息。

组最后一个记录时，程序化遇到控制分支，即改变控制字段，打印汇总记录，构成组的末尾，然后打印下一组头记录。

到达文件末尾是，打印报表汇总。

## 报表定义

报表是直接写入顺序输出文件，因此要在环境部和数据部文件中定义。

数据部中，报表节放在工作单元节和连接节后

环境部

数据部

报表节中描述构成报表的每个项目。用RD描述项定义报表的总体页面布局和指定控制字段。

RD下01层项目定义归属的报表部分，通常包括：

Report Heading

Report Footing

Page Heading

Page Footing

然后对报表中每个控制字段，可以有头和尾。

Control Heading

Control Footing

Detail

在01层项目下面是该组的描述：数据在组中的崔志和水平的位置，项目的格式字符串，数据的来源。用COLUMN、LINE、NUMBER、PICTURE、SOURCE等从句。

报表节中定义报表布局语句的通常顺序：

* + 1. Open打开报表文件
    2. Initiate初始化报表
    3. 读取或提供细节记录的下一组数据
    4. 用generate语句生成细节记录
    5. 重复3、4
    6. 用terminate语句完成报表
    7. 用close语句关闭文件

文件控制和文件描述项

COLUMN 定义数据在行上的列号（左边从1开始编号）。行上的字段填充自动实现

PICTURE 只允许display类型数据

VALUE 常数

SOURCE 指定一个标识符，其内容将在报告编写模块打印这个报表时放进报表的这个位置，特殊源PAGE-COUNTER用于页脚组中，包含这个报表的当前页号

SUM 指定一个标识符，数值加进动态汇总中。

过程部

二、报表编制功能在COBOL程序中的描述

1．在数据部中的描述

（1）文件节：一般格式为

RECORDS ARE

LABEL STANDARD

RECORD IS

RECORDS

BLOCK CONTAINS [整数1 TO ] 整数2

CHARACTERS

[RECORD CONTAINS [整数3 TO ] 整数4 CHARACTERS]

RECORDS ARE

DATA 数据名1 [,数据名2]…

RECORD IS

数据名4 数据名5

VALUE OF数据名3 IS ，数据名4 IS

常量1 常量2

[CODE CHARACTERS]

REPORT IS

报表名1 [,报表名2]…

REPORTS ARE

（2）报表节

2．在过程部中的描述

（1）INITIATE语句（初始化语句）

INITIATE 报表名1 [，报表名2]…

此语句的作用是是这些报表的求和计数器和行计数器置零。

（2）GENERATE语句（生成语句）

数据名

GENERATE

报表名

GENERATE语句按照数据部报表节规定的报表描述产生相应的报表栏和其他操作，“数据名”必须是细目报表栏名，并且可以用报表名来对其限定。

（3）TERMINATE语句（终止语句）

TERMINATE 报表名 [，报表名2]…  
此语句使报表控制系统对指定的报表完成所有的结束处理，产生报表尾栏。它不关闭文件。

**报表栏**

* **报表头 整个报表的封面或标题**
* **报表尾 报表的封底**
* **页头 输出每一页的标题**
* **页尾 输出本也的结束信息**
* **控制头 输出统计单位的信息，可以设置不同层次的单位，分别统计。**
* **控制尾 输出本单位统计数字的小计**
* **细目 报表的具体统计数字，即明细。**

**初始化语句**

* **语法：**

**INTIATE 报表名1，[报表名2]……**

**作用：使报表的求和计数器和行计数器置零，页计数器置1**

**生成语句**

* **语法：**

**GENERATE 数据名**

**报表名**

**作用：按照数据部报表节规定的报表描述产生相应的报表栏和其他操作，数据名必须是明细报表栏名，并且可以用报表名来对其限定，如果使用“报表名”，则不打印细目报表栏，仅进行求和等操作。**

**终止语句**

* **语法：**

**TERMINATE 报表名1[，报表名2]**

**作用:使报表编制控制系统对指定的报表完成所有的结束处理，产生报表尾栏，他不关闭文件。**

本章小结

# 嵌入式数据SQL

**学习提示**

访问数据库方法：使用嵌入的SQL 语句，说明如下：

①在SQL 文中使用的变量必须先定义

② COBOL 中的变量名不能有下划线，而SQL 中的变量名不能有中划线（包括全角和半角）。

③在COBOL 中的SQL 语句必须以EXEC SQL 开始，以END-EXEC 结束。

④ SQL 中的变量名在“EXEC SQL BEGIN DECLARE SECTION END-EXEC.”

和“EXEC SQL END DECLARE SECTION END-EXEC.”之间定义。

⑤ DB 的COPY 句用“INCLUDE”命令引入。

⑥“END-EXEC”要上下对齐。

⑦“SQLCOM.CBL”和“SQLCA.CBL”是连接ORACLE 用的COPY 句，在每本DB 程序中都要引用。

⑧ PCO 文件是操作数据库的源程序，需要预编译成COB 文件（COBOL 的源程序），然后再编译和连接

成可执行的程序文件。

⑨预编译文件PCO 经过ORACLE 预编译后把DB 的COPY 句的内容插入到COBOL 程序中。

⑩纪录数据库操作返回状态的变量有SQLSTATE（X(05)）、SQLCODE（S9(9)）、SQLERRMC（X(70)），

其中，这3 个变量需要在SQLCA.COB 中定义。现在我们采用SQLCODE 判断SQL 操作的返回值，值

为ZERO 表示正常，100 表示数据没有检索到，或者游标读取结束。（其他值可以参考Oracle 返回

值说明文件）

􀀀在数据库操作的SQL 语句中，COBOL 的变量前面要加字符‘:’。

**常用的SQL语句**

## ORACLE PRO\*COBOL编程

本章小结

# COBOL与其他编程语言

**学习提示**

## C语言的接口

本章小结

# COBOL内部函数

学习提示

## 函数简介

函数调用：

语法：

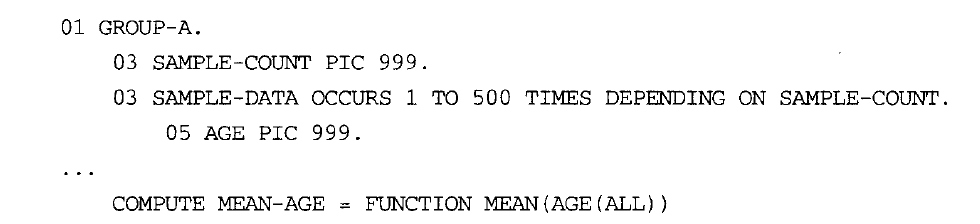
FUNCTION function-name [({argumengt})…………]

函数名可以作为用户定义字复用，函数名前面的单词使其成为内部函数。

函数变量

变量可以是标识符、直接数、算术表达式。包括函数调用。

多个变量间用空格、逗号、空格或分号加空格分隔。



本章小结

# COBOL程序设计思想

**学习提示**

## 结构化程序设计

## 基于面向对象思想的程序设计

## 匹配处理

**本章小结**

**附录**

# COBOL关键字

ACCEPT

# 内部函数一览

|  |  |  |
| --- | --- | --- |
| **函数名** | **变元个数：类型** | **返回值** |
| **ACOS** | **1：Numeric** | **数字，反余弦值** |
| **ANNUITY** | **2:Numeric,integer** | **数字，利率（数字变量）和期限（整数变量）指定时的年支付率** |
| **ASIN** | **1：Numeric** | **数字，反正弦值** |
| **ATAN** | **1: Numeric** | **数字，反正切值** |
| **CHAR** | **1:Integer** | **字母数字，排列顺序中的字符位置（整数变量）** |
| **COS** | **1:Integer** | **数字，余弦值** |
| **CURRENT-DATE** | **0** | **字母数字，当前日期和时间** |
| **DATE-OF-INTEGER** | **1:Integer** | **整数，日数的标准日期（整数变量）** |
| **DAY-OF-INTEGER** | **1:Integer** | **整数，日数的儒勒日期（整数变量）** |
| **FACTORIAL** | **1:Integer** | **整数，阶乘** |
| **INTEGER** | **1:Numeric** | **数字变量的整数部分** |
| **INTEGER-OF-DATE** | **1:Integer** | **整数，标准日期的日数（整数变量）** |
| **INTEGER-OF-DAY** | **1:Integer** | **整数，儒勒日期的日数（整数变量）** |
| **INTEGER-PART** | **1:Numeric** | **数字变量的整数部分** |
| **LENGTH** | **1:任何类型** | **整数，变量长度（字符数）** |
| **LOG** | **1:Numeric** | **数字，自然对数** |
| **LOG10** | **1:Numeric** | **数字，标准对数（基数为10）** |
| **LOWER-CASE** | **1:Alphanumeric** | **字母数字，变量所有字母变成小写** |
| **MAX** | **Variable:任何类型** | **最大变量的值** |
| **MEAN** | **Variable:Numeric** | **数字，变量的算术平均值** |
| **MEDIAN** | **Variable:Numeric** | **数字，变量的中数** |
| **MIDRANGE** | **Variable:Numeric** | **数字，变量最大值和最小值的平均值** |
| **MIN** | **Variable:任何类型** | **最小变量值** |
| **MOD** | **2:integer,integer** | **最小变量的值 有疑问** |
| **NUMVAL** | **1:Alphanumeric** | **整数，第一个整数变量与第二个整数变量求模** |
| **NUMVAL-C** | **1:Alphanumeric 或2：Alphanumeric，Alphanumeric** | **数字，字母数字变量的数字值（用逗号和货币号）** |
| **ORD** | **1:Alphanumeric** | **整数，表示指定折扣率（第一个数字变量）时的期值（第二个及以后的数字变量）** |
| **ORD-MAX** | **Variable:任何类型** | **整数，最大变量的序号** |
| **ORD-MIN** | **Variable:任何类型** | **整数，最小变量的序号** |
| **PRESENT-VALUE** | **2或more:Numeric** | **数字，表示指定折扣率（第一个数字变量）时的期值（第二个及以后的数字变量）** |
| **RANDOM** | **0或1:Integer** | **数字，随机数（0<=N<1）** |
| **RANGE** | **Variable:Numeric** | **数字，（最大值——最小值）数值** |
| **REM** | **2:Numeric,Numeric** | **数字，第一个数字变量除以第二个数字变量的余数。** |
| **REVERSE** | **1:Alphanumeric** | **字母数字，逆转字符顺序的变量** |
| **SIN** | **1:Numeric** | **数字，正弦值** |
| **SQRT** | **1:Numeric** | **数字，平方根** |
| **STANDARD-DEVIATION** | **Variable:Numeric** | **数字，标准偏差** |
| **SUM** | **Variable:Numeric** | **数字和** |
| **TAN** | **1:Numeric** | **数字，正切值** |
| **UPPER-CASE** | **1:Alphanumeric** | **字母数字，所有字符变为大写的变量** |
| **VARINCE** | **Variable:Numeric** | **数字，方差** |
| **WHEN-COMPILED** | **0** | **字母数字，程序编译的日期和时间** |
|  |  |  |

字段长度

LENGTH OF 数据名

SET ADDRESS OF 变量名 TO 变量名

REPLACE =='AAAAAAA'== BY ==BBBBBBBBBBBB==.

COPY \*\*\*\*\*\*.

REPLACE OFF.