**主机软件开发系列丛书**

**COBOL学习实践**

**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](#_Toc274506344)

[§1.1 思考题 8](#_Toc274506345)

[第2章 Cobol编程的基本元素 8](#_Toc274506346)

[§2.1 思考题 8](#_Toc274506347)

[§2.2 实验 8](#_Toc274506348)

[2.2.1 Hello Cobol 8](#_Toc274506349)

[2.2.2 逻辑与控制语句 8](#_Toc274506350)

[§2.3 实例操练二：九九乘法表 9](#_Toc274506351)

[**思考与练习** 12](#_Toc274506352)

[**思考与练习解答** 12](#_Toc274506353)

[第3章 数据处理 15](#_Toc274506354)

[§3.1 算术运算精度处理 16](#_Toc274506355)

[3.1.1 四舍五入处理子句（ROUNDED子句） 16](#_Toc274506356)

[3.1.2 长度溢出处理子句（ON SIZE ERROR） 16](#_Toc274506357)

[3.1.3 除法语句的余数子句(REMAINDER) 16](#_Toc274506358)

[§3.2 对应传送子句（或称作同名传送CORR） 17](#_Toc274506359)

[§3.3 对应项间的运算（CORR） 19](#_Toc274506360)

[§3.4 本章小结 19](#_Toc274506361)

[思考与练习 19](#_Toc274506362)

[思考与练习解答 20](#_Toc274506363)

[第4章 字符串处理 20](#_Toc274506364)

[§4.1 字符串连接（STRING） 20](#_Toc274506365)

[§4.2 字符串分解语句（UNSTRING） 21](#_Toc274506366)

[§4.3 检测语句（INSPECT语句） 22](#_Toc274506367)

[第5章 COBOL85逻辑控制 24](#_Toc274506368)

[§5.1 逻辑控制语句 24](#_Toc274506369)

[§5.2 条件表达式 24](#_Toc274506370)

[§5.3 IF-ELSE语句 24](#_Toc274506371)

[§5.4 EVALUATE语句 25](#_Toc274506372)

[§5.5 CONTINUE语句 25](#_Toc274506373)

[§5.6 GO TO 26](#_Toc274506374)

[§5.7 条件关系 26](#_Toc274506375)

[§5.8 复合条件 26](#_Toc274506376)

[§5.9 符号条件 27](#_Toc274506377)

[§5.10 类型条件 27](#_Toc274506378)

[§5.11 条件名条件 28](#_Toc274506379)

[§5.12 循环处理 28](#_Toc274506380)

[§5.13 带有循环控制变量的PERFORM 语句 30](#_Toc274506381)

[5.13.1 一重循环 30](#_Toc274506382)

[5.13.2 三重循环的PERFORM 语句 31](#_Toc274506383)

[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。 32](#_Toc274506384)

[§5.14 本章小结 33](#_Toc274506385)

[思考与练习 33](#_Toc274506386)

[思考与练习解答 33](#_Toc274506387)

[第6章 COBOL85表与检索 33](#_Toc274506388)

[§6.1 表的基本概念 33](#_Toc274506389)

[§6.2 表的建立 35](#_Toc274506390)

[§6.3 表元素的引用 37](#_Toc274506391)

[§6.4 表元素赋初值 37](#_Toc274506392)

[§6.5 表的应用 38](#_Toc274506393)

[§6.6 位标法 39](#_Toc274506394)

[6.6.1 位标名 39](#_Toc274506395)

[6.6.2 位标法引用表元素规则 39](#_Toc274506396)

[6.6.3 设置语句 39](#_Toc274506397)

[§6.7 表的顺序检索 40](#_Toc274506398)

[§6.8 表的对分检索 41](#_Toc274506399)

[§6.9 用PERFORM语句对表进行检索 42](#_Toc274506400)

[§6.10 本章小结 43](#_Toc274506401)

[思考与练习 43](#_Toc274506402)

[思考与练习解答 43](#_Toc274506403)

[第7章 COBOL85子程序 43](#_Toc274506404)

[§7.1 子程序概述 43](#_Toc274506405)

[一、概述 43](#_Toc274506406)

[§7.2 主程序与子程序连接 44](#_Toc274506407)

[一、 子程序结构 44](#_Toc274506408)

[（一） 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。 44](#_Toc274506409)

[（二） 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。 44](#_Toc274506410)

[（三） 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。 44](#_Toc274506411)

[二、 调用语句语法： 44](#_Toc274506412)

[三、 联接节 45](#_Toc274506413)

[§7.3 子程序返回语句 45](#_Toc274506414)

[§7.4 公用区数据 46](#_Toc274506415)

[§7.5 子程序举例 46](#_Toc274506416)

[§7.6 习题答案演示系统 47](#_Toc274506417)

[7.6.1 需求分析 47](#_Toc274506418)

[7.6.2 系统设计 47](#_Toc274506419)

[§7.7 本章小结 47](#_Toc274506420)

[思考与练习 47](#_Toc274506421)

[思考与练习解答 48](#_Toc274506422)

[第8章 COBOL85程序部 48](#_Toc274506423)

[§8.1 标识部 48](#_Toc274506424)

[§8.2 环境部 49](#_Toc274506425)

[8.2.1 环境部概述 49](#_Toc274506426)

[8.2.2 配置节 50](#_Toc274506427)

[8.2.3 文件组织方式 55](#_Toc274506428)

[8.2.4 输入输出节 57](#_Toc274506429)

[8.2.4.1 输入输出节概述 57](#_Toc274506430)

[8.2.4.2 文件控制段 58](#_Toc274506431)

[2． 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。 60](#_Toc274506432)

[FILE-CONTROL 64](#_Toc274506433)

[SELECT 文件名 ASSIGN TO 设备名1，… 64](#_Toc274506434)

[RANDOM 64](#_Toc274506435)

[DYNAMIC 64](#_Toc274506436)

[ORGANAZATION IS RELATIVE 64](#_Toc274506437)

[ACCESS MODE IS SEQUENTIAL 64](#_Toc274506438)

[8.2.4.3 输入输出控制段 64](#_Toc274506439)

[§8.3 数据部 65](#_Toc274506440)

[8.3.1 数据部概述 65](#_Toc274506441)

[8.3.2 文件节 70](#_Toc274506442)

[8.3.3 工作单元节（WORKING-STORAGE SECTION） 73](#_Toc274506443)

[§8.4 过程部 81](#_Toc274506444)

[§8.5 本章小结 82](#_Toc274506445)

[思考与练习 83](#_Toc274506446)

[思考与练习解答 83](#_Toc274506447)

[第9章 COBOL85文件处理 83](#_Toc274506448)

[§9.1 文件的基本概念 83](#_Toc274506449)

[§9.2 文件处理常用语句概述 84](#_Toc274506450)

[9.2.1 打开语句（OPEN语句） 84](#_Toc274506451)

[9.2.2 读语句（READ语句） 84](#_Toc274506452)

[9.2.3 写语句（WRITE语句） 85](#_Toc274506453)

[9.2.4 重写语句REWRITE 86](#_Toc274506454)

[9.2.5 删除语句DELETE 86](#_Toc274506455)

[9.2.6 关闭语句（CLOSE语句） 86](#_Toc274506456)

[9.2.7 开始语句START 87](#_Toc274506457)

[9.2.8 文件操作汇总 87](#_Toc274506458)

[§9.3 磁带文件 87](#_Toc274506459)

[§9.4 磁盘顺序文件 90](#_Toc274506460)

[9.4.1 顺序文件的概念 90](#_Toc274506461)

[9.4.2 COBOL中与顺序文件有关的部分 90](#_Toc274506462)

[9.4.3 顺序文件实例 91](#_Toc274506463)

[§9.5 磁盘索引文件 94](#_Toc274506464)

[9.5.1 索引文件的概念 94](#_Toc274506465)

[9.5.2 COBOL中与索引文件有关的部分 94](#_Toc274506466)

[9.5.3 建立索引文件 95](#_Toc274506467)

[9.5.4 检索索引文件记录 97](#_Toc274506468)

[9.5.5 新增索引文件记录 98](#_Toc274506469)

[9.5.6 修改索引文件记录 100](#_Toc274506470)

[9.5.7 删除索引文件记录 100](#_Toc274506471)

[9.5.8 综合实例 100](#_Toc274506472)

[§9.6 磁盘相对文件 102](#_Toc274506473)

[9.6.1 相对文件的概念 102](#_Toc274506474)

[9.6.2 COBOL中与相对文件有关的成分 102](#_Toc274506475)

[9.6.3 建立相对文件 103](#_Toc274506476)

[§9.7 动态存取 103](#_Toc274506477)

[§9.8 文件综合应用 103](#_Toc274506478)

[9.8.1 与表操作结合实例 104](#_Toc274506479)

[§9.9 本章小结 104](#_Toc274506480)

[思考与练习 104](#_Toc274506481)

[思考与练习解答 105](#_Toc274506482)

[第10章 排序 107](#_Toc274506483)

[§10.1 基本概念 107](#_Toc274506484)

[§10.2 环境部中排序文件的描述 107](#_Toc274506485)

[§10.3 数据部中排序文件描述 108](#_Toc274506486)

[§10.4 排序语句（SORT）单纯排序 108](#_Toc274506487)

[§10.5 排序语句（SORT）加工排序 109](#_Toc274506488)

[第11章 合并 111](#_Toc274506489)

[§11.1 基本概念 111](#_Toc274506490)

[知识小结 116](#_Toc274506491)

[思考与练习 116](#_Toc274506492)

[思考与练习解答 116](#_Toc274506493)

[第12章 画面 116](#_Toc274506494)

[第13章 报表 117](#_Toc274506495)

[§13.1 报表概述 117](#_Toc274506496)

[§13.2 报表定义 118](#_Toc274506497)

[二、报表编制功能在COBOL程序中的描述 119](#_Toc274506498)

[第14章 嵌入式数据SQL 121](#_Toc274506499)

[§14.1 ORACLE PRO\*COBOL编程 122](#_Toc274506500)

[第15章 COBOL与其他编程语言 122](#_Toc274506501)

[第16章 COBOL内部函数 123](#_Toc274506502)

[第17章 COBOL综合实例 123](#_Toc274506503)

[§17.1 综合实例一：个人股票交易管理 123](#_Toc274506504)

[§17.2 综合实例二：学生管理信息系统 124](#_Toc274506505)

[17.2.1 系统分析 124](#_Toc274506506)

[17.2.2 系统概况 124](#_Toc274506507)

[17.2.3 需求分析 125](#_Toc274506508)

[17.2.3.1 理解需求 125](#_Toc274506509)

[17.2.3.2 分析需求 126](#_Toc274506510)

[(1) 记录学生情况 126](#_Toc274506511)

[(2) 修改学生信息 查询学生信息→修改学生信息。 126](#_Toc274506512)

[(3) 统计查询学生情况 126](#_Toc274506513)

[(4) 记录学生考勤情况 126](#_Toc274506514)

[(5) 修改学生考勤情况 126](#_Toc274506515)

[(6) 统计查询学生考勤情况 126](#_Toc274506516)

[(7) 记录学生奖惩情况 127](#_Toc274506517)

[(8) 修改学生奖惩情况 127](#_Toc274506518)

[(9) 统计查询学生奖惩情况 127](#_Toc274506519)

[(10) 设置院系情况 127](#_Toc274506520)

[§17.3 UML系统建模 127](#_Toc274506521)

[17.3.1 学生管理信息系统的用例分析 127](#_Toc274506522)

[17.3.2 学生管理信息系统的领域分析 128](#_Toc274506523)

[§17.4 系统配置 131](#_Toc274506524)

[17.4.1 软件配置 131](#_Toc274506525)

[17.4.2 硬件配置 132](#_Toc274506526)

[17.4.3 网络配置 133](#_Toc274506527)

[§17.5 数据库分析 134](#_Toc274506528)

[17.5.1 E-R图分析 134](#_Toc274506529)

[17.5.2 表与字段分析和PowerDesign数据库建模 135](#_Toc274506530)

[§17.6 数据库设计 136](#_Toc274506531)

[§17.7 界面设计 141](#_Toc274506532)

[§17.8 学生管理主界面设计 141](#_Toc274506533)

[17.8.1 学生基本信息界面的设计 142](#_Toc274506534)

[17.8.2 学生基本信息窗体要完成信息的录入查询功能，因此需要设计菜单。该窗体的设计如图9-18所示。 142](#_Toc274506535)

[§17.9 连接数据库 145](#_Toc274506536)

[§17.10 功能模块设计 145](#_Toc274506537)

[§17.11 代码分析与实现 147](#_Toc274506538)

[附录 1](#_Toc274506539)

[附录1 COBOL关键字 1](#_Toc274506540)

[ACCEPT 1](#_Toc274506541)

[附录2 内部函数一览 1](#_Toc274506542)

[附录3 开发工具 2](#_Toc274506543)

[应用EDITPLUS编辑COBOL程序 2](#_Toc274506544)

[应用UESTUDIO编辑COBOL程序 2](#_Toc274506545)

[NET COBOL开发环境配置 3](#_Toc274506546)

[COBOL85单机软件开发环境配置概述 3](#_Toc274506547)

[OS 390 编译Cobol程序 33](#_Toc274506548)

[附录4 COBOL开发补充 33](#_Toc274506549)

[COBOL85语法汇总 33](#_Toc274506550)

[COBOL编码规约 33](#_Toc274506551)

[COBOL CHECK LIST 33](#_Toc274506552)

[COBOL的网络资源 33](#_Toc274506553)

[**中文** 33](#_Toc274506554)

[**英文** 33](#_Toc274506555)

[**日文** 33](#_Toc274506556)

[附录5 索引 34](#_Toc274506557)

[附录6 自我测试 34](#_Toc274506558)

[参考文献 34](#_Toc274506559)

[IBM———————— 34](#_Toc274506560)

[COBOL上机实验 34](#_Toc274506561)

[Average record unit 35](#_Toc274506562)

[（三）建立一个存放LOAD的PDS：YOURID.COBOL.LOAD 35](#_Toc274506563)

[Average record unit 35](#_Toc274506564)

[//YOURIDXX JOB ACCT#,YOURID,CLASS=A,MSGCLASS=H, 35](#_Toc274506565)

[//COBOL.SYSLIB DD DISP=SHR,DSN= YOURID.COBOL.SOURCE 35](#_Toc274506566)

[//STEPLIB DD DSN= YOURID.COBOL.LOAD,DISP=SHR 36](#_Toc274506567)

[//STEPLIB DD DSN= YOURID.COBOL.LOAD,DISP=SHR 37](#_Toc274506568)

# COBOL85程序设计概述

## 思考题

# Cobol编程的基本元素

## 思考题

## 实验

### Hello Cobol

程序代码：

### 逻辑与控制语句

**编写程序，输入3个数，按从大到小排序显示。**

求1+2+3+……+100的和。 EXAMPLE45\_80\_2

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. |

## 实例操练二：九九乘法表

通过各种循环实现九九乘法表

IF-PERFORM-2-SSB-99T

代码如下：

000010\*>

000020 IDENTIFICATION DIVISION.

000030 PROGRAM-ID. IF-PERFORM.

000040 ENVIRONMENT DIVISION.

000050 CONFIGURATION SECTION.

000060 DATA DIVISION.

000070 WORKING-STORAGE SECTION.

000080 77 PERFORM-N PIC 9 VALUE 9.

000090\*

000100 01 COM-99.

000110 03 M1 PIC 99.

000111 03 FILLER PIC X VALUE "\*".

000120 03 M2 PIC 99.

000121 03 FILLER PIC X VALUE "=".

000130 03 MM PIC 99.

000140\*

000150 01 SHOW-99.

000160 03 SHOW-M1 PIC 99.

000170 03 FILLER PIC X .

000180 03 SHOW-M2 PIC 99.

000190 03 FILLER PIC X .

000200 03 SHOW-MM PIC 99.

000240 PROCEDURE DIVISION.

000250\* L-1

000260 S-MAIN.

000270\*

000280 DISPLAY "HELLO PERFORM".

000290\*

000300 PERFORM S-PERFORM THRU S-PERFORM-END

000310 UNTIL PERFORM-N = 0.

000320\*

000330 DISPLAY "99 SHOW END!".

000331 STOP 1.

000340 STOP RUN.

000350\* L-2

000360 S-PERFORM.

000370 PERFORM MENU-SHOW.

000380 DISPLAY "PERFORM-N=" WITH NO ADVANCING.

000390 ACCEPT PERFORM-N.

000391 DISPLAY " ".

000400 INITIALIZE COM-99.

000410 EVALUATE PERFORM-N

000420 WHEN 1 PERFORM PERFORM-TIMES

000430 WHEN 2 PERFORM PERFORM-UNTIL

000440 WHEN 3 PERFORM PERFORM-VARYING

000450 WHEN OTHER GO TO S-PERFORM-END

000460 END-EVALUATE.

000470

000480 S-PERFORM-END.

000490 EXIT.

000500\* L-3

000510 MENU-SHOW.

000520 DISPLAY "PERFORM-N=0 STOP RUN".

000530 DISPLAY "PERFORM-N=1 PERFORM TIMES".

000540 DISPLAY "PERFORM-N=2 PERFORM UNTIL" .

000550 DISPLAY "PERFORM-N=3 PERFORM VARYING".

000560\*

000570 PERFORM-TIMES.

000580 DISPLAY "PERFORM TIMES".

000590 PERFORM PERFORM-TIMES-SUB THRU PERFORM-TIMES-END 9 TIMES.

000600\*

000610 PERFORM-UNTIL.

000620 DISPLAY "PERFORM UNTIL".

000630 PERFORM PERFORM-UNTIL-SUB THRU PERFORM-UNTIL-END

000640 UNTIL M1 > 8.

000650\*

000660 PERFORM-VARYING.

000670 DISPLAY "PERFORM VARYING".

000680 PERFORM PERFORM-VARYING-SUB THRU PERFORM-VARYING-END

000690 VARYING M1 FROM 1 BY 1 UNTIL M1 > 9

000700 AFTER M2 FROM 1 BY 1 UNTIL M2 > M1.

000710\* L-4

000720 PERFORM-TIMES-SUB.

000730 ADD 1 TO M1.

000731 MOVE 1 TO M2.

000740 PERFORM M1 TIMES

000750 COMPUTE MM = M1 \* M2

000770 MOVE COM-99 TO SHOW-99

000780 DISPLAY SHOW-99 WITH NO ADVANCING

000781 DISPLAY " " WITH NO ADVANCING

000782 ADD 1 TO M2

000791 END-PERFORM.

000800 DISPLAY " ".

000810 PERFORM-TIMES-END.

000820 EXIT.

000830\*

000840 PERFORM-UNTIL-SUB.

000850 ADD 1 TO M1.

000851 MOVE 1 TO M2.

000860 PERFORM UNTIL M2 > M1

000870 COMPUTE MM = M1 \* M2

000890 MOVE COM-99 TO SHOW-99

000900 DISPLAY SHOW-99 WITH NO ADVANCING

000901 DISPLAY " " WITH NO ADVANCING

000902 ADD 1 TO M2

000910 END-PERFORM.

000920 DISPLAY " ".

000930 PERFORM-UNTIL-END.

000940 EXIT.

000950\*

000960 PERFORM-VARYING-SUB.

000970 COMPUTE MM = M1 \* M2.

000980 MOVE COM-99 TO SHOW-99.

000990 DISPLAY SHOW-99 WITH NO ADVANCING.

000991 DISPLAY " " WITH NO ADVANCING.

001000 IF M1 = M2 DISPLAY " ".

001010 PERFORM-VARYING-END.

001020 END PROGRAM IF-PERFORM.

**思考与练习**

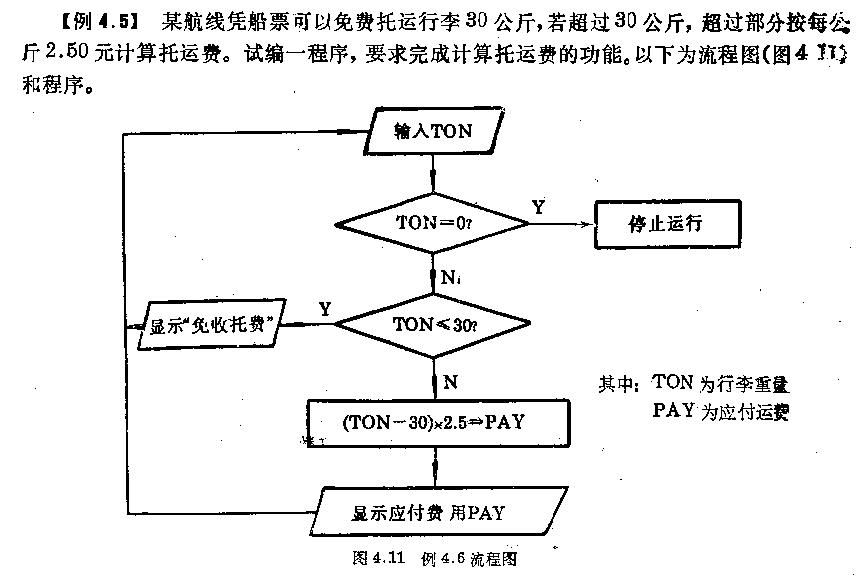
1. **编写一个程序，录入10个数，找出其中最大和最小值**
2. **某航线凭船票可以免费托运行李30公斤，若超过30公斤，超过部分按每公斤2.50元计算托运费，编写一个程序完成计算托运费的功能**
3. **请用多种语句编写一个程序，实现1到100的求和。**

**思考与练习解答**

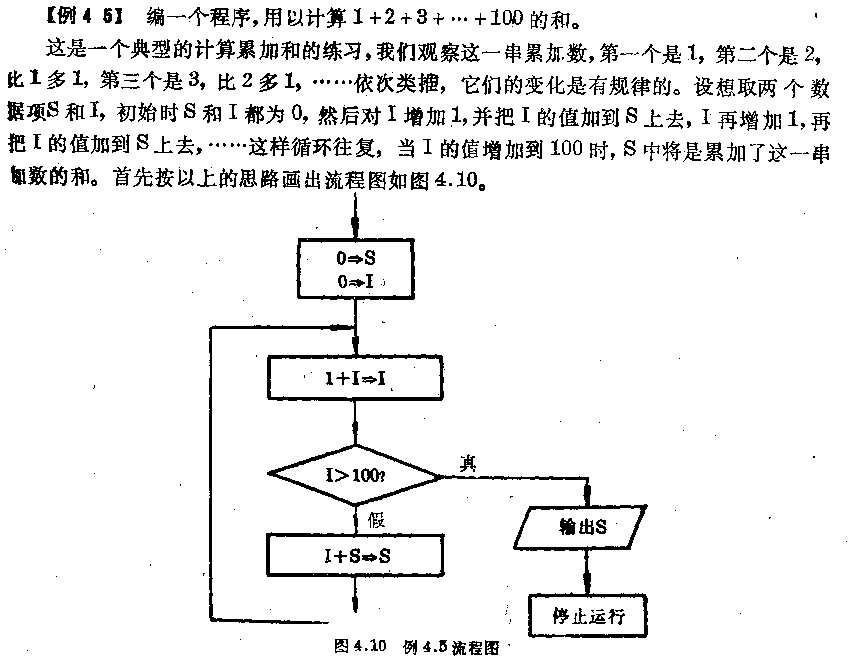


|  |  |
| --- | --- |
|  | **IDENTIFICATION DIVISION.**  **PROGRAM-ID. COBOLSAMPLE.**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **\***  **\*程序ID ：COBOLSAMPLE**  **\*程序名 ：演示程序**  **\*处理概要 ：**  **\***  **\*变量 IO 变量名**  **\***  **\*日期 作成者 概要**  **\*2007/03/18 孙树斌 作成**  **\***  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **ENVIRONMENT DIVISION.**  **DATA DIVISION.**  **WORKING-STORAGE SECTION.**  **77 X PIC 9(3)V9.**  **77 MAX-M PIC 9(3)V9.**  **77 MIN-M PIC 9(3)V9.**  **77 MAX-E PIC Z(3).9.**  **77 MIN-E PIC Z(3).9.**  **77 I PIC 9 VALUE 0.**  **PROCEDURE DIVISION.**  **MAIN-PROCESS SECTION.**  **DISPLAY "ENTER X".**  **ACCEPT X.**  **MOVE X TO MAX-M,MIN-M.**  **ADD 1 TO I.**  **MAIN.**  **DISPLAY 'ENTER X '**  **ACCEPT X.**  **IF X > MAX-M MOVE X TO MAX-M.**  **IF X < MIN-M MOVE X TO MIN-M.**  **ADD 1 TO I.**  **IF I < 10 GO TO MAIN.**  **MOVE MAX-M TO MAX-E.**  **MOVE MIN-M TO MIN-E.**  **DISPLAY "THE MAXIMUM NUMBER IS ",MAX-E.**  **DISPLAY "THE MINIMUM NUMBER IS ",MIN-E.**  **STOP 0.**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **END PROGRAM COBOLSAMPLE.**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** |

1. **某航线凭船票可以免费托运行李30公斤，若超过30公斤，超过部分按每公斤2.50元计算托运费，编写一个程序完成计算托运费的功能。**

****

**4．**

****

# 数据处理

# 字符串处理

**介绍字符串的拼接，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：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | { | 标识符1 | | }[{ | ，标识符2 | | }] |  |  |
| STRING | |  | |  | |  |  |
|  | | 常量1 | | ，常量2 | |  |  |
|  | |  |  | |  |  | |  |  |  |
|  |  |  |  | { | 标识符3 | | }[，{ | 标识符4 | }] |  |
|  | DELIMITED BY | | | 常量3 | |  |  |
|  |  |  |  | SIZE | | 常量4 |  |
|  |  |  |  | |  |  | |  |  |  |
|  |  |  |  | { | 标识符6 | | } |  |  |  |
|  | DELIMITED BY | | | 常量6 | |  |  |
|  |  |  |  | SIZE | |  |  |
|  |  |  |  | |  |  | |  |  |  |
|  | INTO |  | 标识符7 | | [WITH POINTER 标识符8] | | | |  |  |
|  |  |  |  | |  |  | |  |  |  |
|  |  |  | [：ON OVERFLOW 强制语句] | | | | | |  |  |
|  |  |  |  | |  |  | |  |  |  |
|  |  |  |  | |  |  | |  |  |  |

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）

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

|  |
| --- |
|  |

# COBOL85逻辑控制

**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

COBOL85语法格式：

|  |
| --- |
| 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：

|  |
| --- |
| GO TO 过程名 |

语法2：

|  |
| --- |
| GO TO 过程名1，过程名2 DEPENDING ON 标示符. |

|  |
| --- |
|  |

## 条件关系

条件关系由关系表达式来表示

关系表达式格式

|  |  |  |
| --- | --- | --- |
| 标识符1 |  | 标识符2 |
| 常量1 | 关系比较符 | 常量2 |
| 算术表达式1 |  | 算术表达式 |

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

语法格式：

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

CH23\_33

**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：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | | | | | |
| PERFORM | 起始过程名 | | THRU | | 结束过程名 | | | | | |
|  |  | |  | |  | | | | | |
|  |  | |  | | | | | | | |
|  | VARYING | | 循环变量标识符 | | | | | | | |
|  |  | | { | 整形直接数 | | } |  | { | 整形直接数 | } |
|  |  | FROM | 初值标识符 | | BY | 步长标识符 |
|  |  | |  | |  |  |
|  |  | |  | | | | | | | |
|  | UNTIL | | 结束条件表达式 | | | | | | | |
|  |  | |  | | | | | | | |

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

思考与练习

思考与练习解答

# COBOL85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. 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 DOWN BY 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 | } |
|  |  |  | NEXT SENTENCE |
|  |  |  |  |  |  |

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

思考与练习

* 音像目录文件，MUSIC.DAT文件组织形式为顺序文件，记录格式为

磁带编号 六位整数

曲名 二十字符

演奏者 二十字符

乐曲类型 十五字符

并且记录按曲名升序排列，记录个数小于200，用表处理技术编制程序，由给定的曲名查出该乐曲存放的磁带编号、作者、乐曲类别。

CH106-188

思考与练习解答

# COBOL85子程序

**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. |

## 习题答案演示系统

### 需求分析

### 系统设计

## 本章小结

思考与练习

综合练习题：分别用执行语句PERFORM和IF语句作两个99乘法表的子程序，然后用主程序调用。实现功能如下：

主程序要求显示一个菜单，如

请输入您的选择：

1、用PERFORM实现的99表；

2、用IF—GO TO

0、退出程序

菜单要求是动态的，看过一个后自动从菜单删除，不再显示

子程序显示的效果参考下面两个

其一

1\* 1= 1

2\* 1= 2 2\* 2= 4

……

其二

9\*9=81 9\*8=72 9\*7=63 9\*6=54 9\*5=45

8\*8=64 8\*7=56 8\*6=48 8\*5=40

思考与练习解答

# COBOL85程序部

**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。

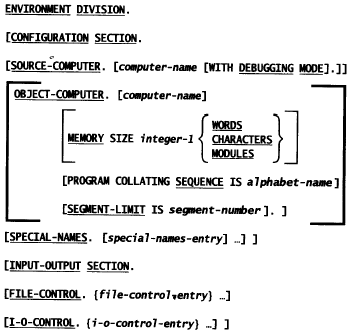
PROGRAM-ID 在这个必需的段落中，program-name是用户定义字，表示程序名，可以选择一个有意义的名称。在应用程序中，应建立一个程序命名规则，并坚持按这个规则命名。在说明行中建档命名规则。如果公司已经有现成的命名规则，则按公司的规则办。

## 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。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [CONFIGURATION SECTION.] | | | | | | | |
| [SOURCE-COMPUTER. 源计算机名 [WITH DEBUGGING MODE]] | | | | | | | |
| [OBJECT-COMPUTER. 目标计算机名 | | | | | | | |
|  |  | { | WORDS | | | } |  |
|  | [MOMORY SIZE 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-POINT IS COMMA] | | | | | | |
|  | [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

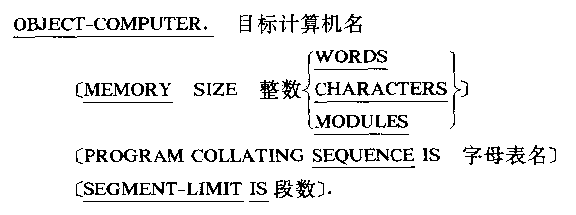
RESET TEACE

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.



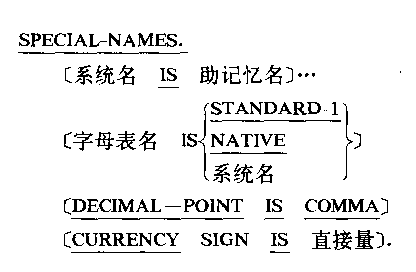
这个段的作用主要是给出执行本程序的计算机名，其内容也是注释性的。此外有三个任选语句1.MEMORY SIZE子句  
这个子句提供有关目标程序运行时所需内存大小的信息，可有字（WORDS）、字符(CHARACTERS）模块（MODULES）三种形式；但本子句内容只作为注释；

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。

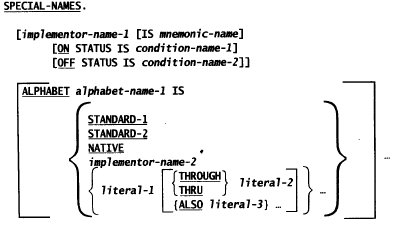
专用名段

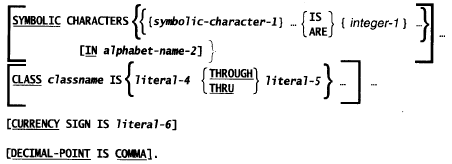
专用名段用来通知系统把系统中原规定的一些设备名或功能名或符号改为用户自己指定的名字或符号 ，也就是定义助记名

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-￥

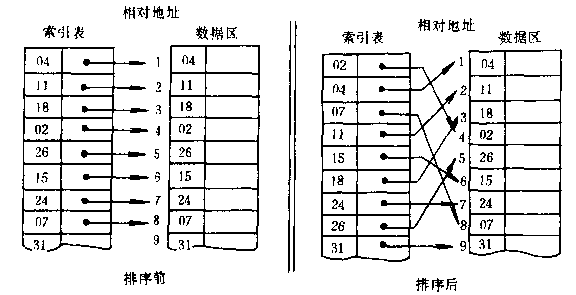
DECIMAL-POINT IS COMMA. 小数点-，

### 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

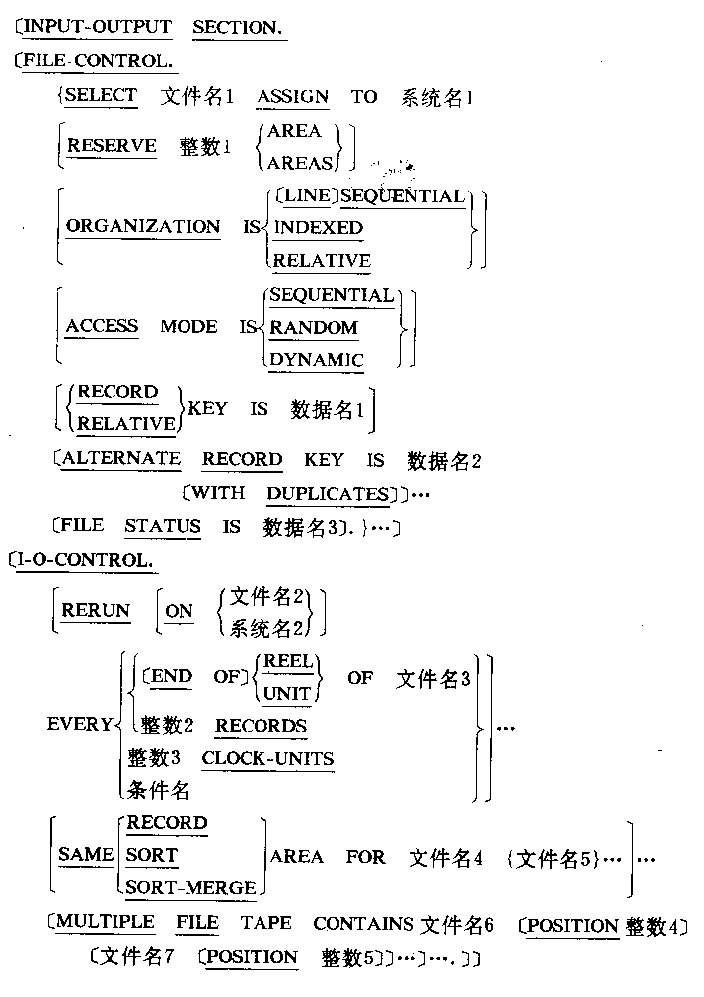
输入-输出节用以描写本程序中各输入输出文件的特性以及它们与外部设备的联系，有关输入输出文件缓冲区方面的规定，以上两部分内容分别在文件控制段和输入输出控制段中。只要程序中用到文件．就一定有输入—输出节，但可以没有其中的输入输出控制段。

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

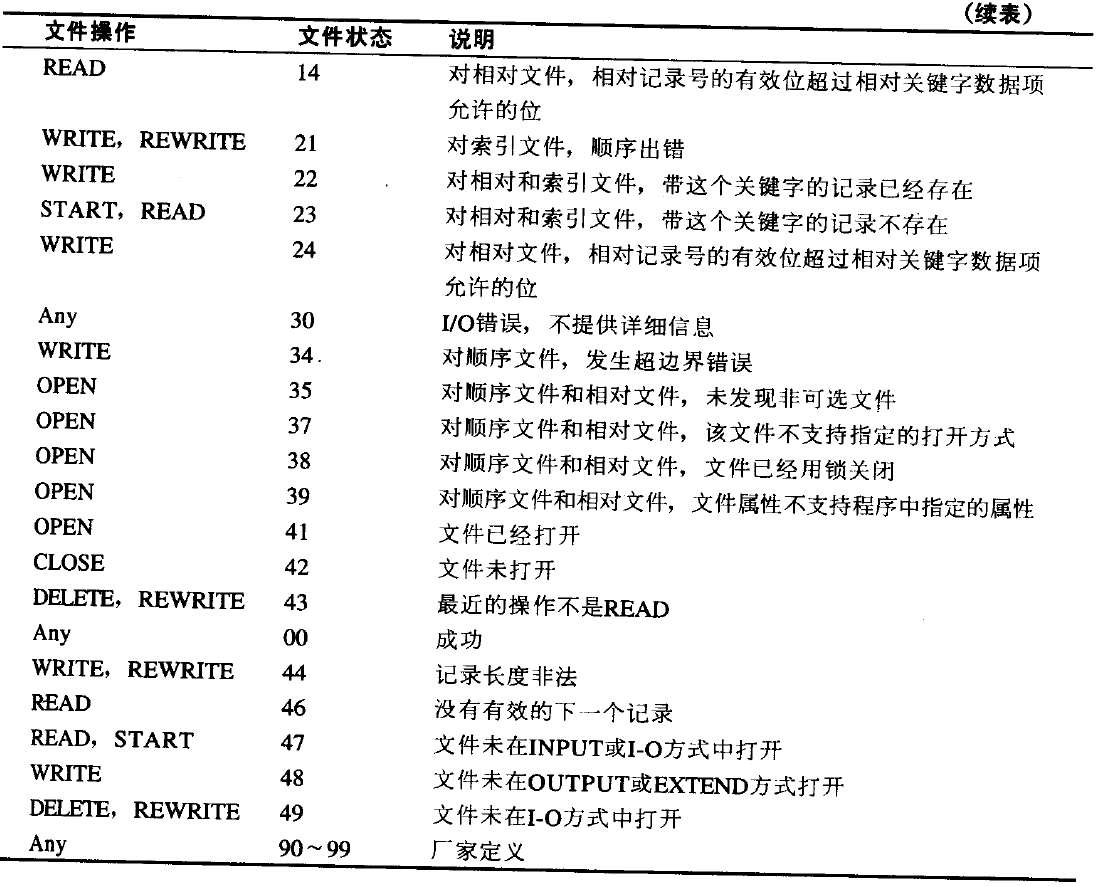
这是一个任选语司。在选用这一语句时，为该文件指定一个两字符的独立数据项，用以存放每次文件操作执行结果的状态码．前一字符称状态码1，用以指示存取操作是否成功地执行：

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

RELATIVE KEY 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）

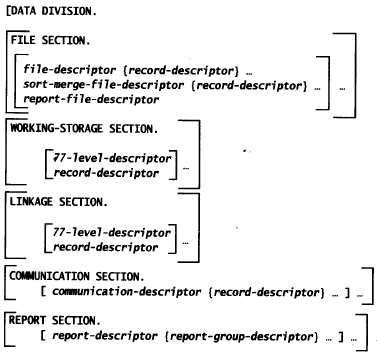
数据部中的节

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

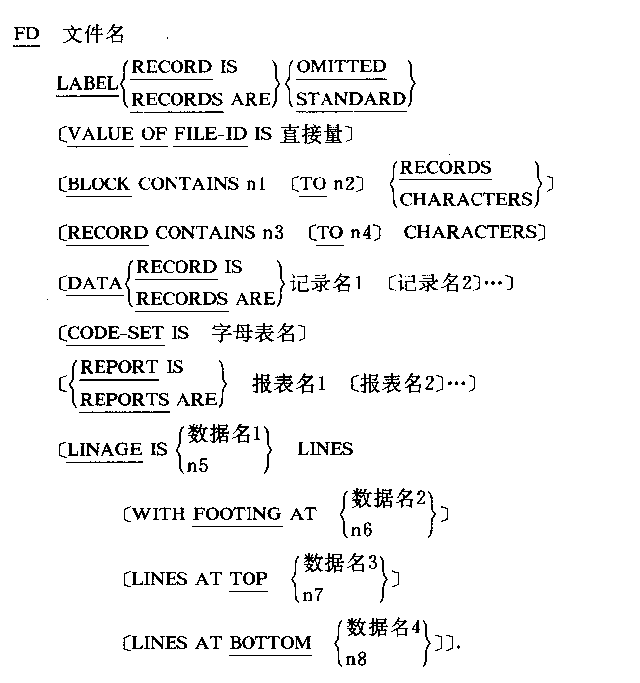
定义该文件中记录的名称

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

|  |
| --- |
| DATA DIVISION.  FILE SECTION.  FD 文件名 LABEL RECORD IS STANDARD  RECORDS ARE OMITTED  VALUE OF FILE-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。

思考与练习

标识部的作用是什么？  
标识部唯一必写部分是什么？

环境部的作用  
在环境部中说明计算机的硬件环境有什么好处？

思考与练习解答

# COBOL85文件处理

学习提示：

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，通过小综合实例了解文件使用方法。

## 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：（语法格式需要更新）

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：

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

### 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.

## 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

思考与练习

随机读取练习

CH41

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

CH42

* 将2000户的登记情况通过打印机打印出来

步骤：

1.打开磁盘文件及打印文件

2.打印报表题头

3.执行打印输出段

4.关闭文件，结束程序

打印输出段

1.读取记录

2.数据传送，并打印输出

CH43

某集团公司计算中心存有一个磁盘文件，数据为该公司下属10个分公司某年外贸出口计划金额和实际完成金额，文件名为SUM.DAT,记录格式为：

公司名：20个字节

出口计划金额：6位整数2位小数

实际完成金额：6位整数2位小数

编写COBOL程序，使用该文件统计总公司全年总计划金额，各公司实际与计划的金额，完成的百分比及各公司完成情况对集团的影响度。

* 相对文件的相对键为记录的底号，不属于记录中的数据项，而索引文件的记录键本身是记录中的数据项，相对文件记录的存取按记录号进行，存取速度快。
* 相对文件必须顺序建立，但允许有空记录
* 举例：建立学生档案，以学号为记录号，记录内容包括姓名，各学科学习成绩

思考与练习解答

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

DATA RECORD 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

# 合并

## 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

RELEASE SORTREC.

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

思考与练习

* 输入文件是一个学生高考的成绩记录，将考生总分>500的打印出来

思考与练习解答

# 画面

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 .

# 报表

## 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。**

# 嵌入式数据SQL

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

# COBOL与其他编程语言

# COBOL内部函数

**第十一讲函数简介**

函数调用：

语法：

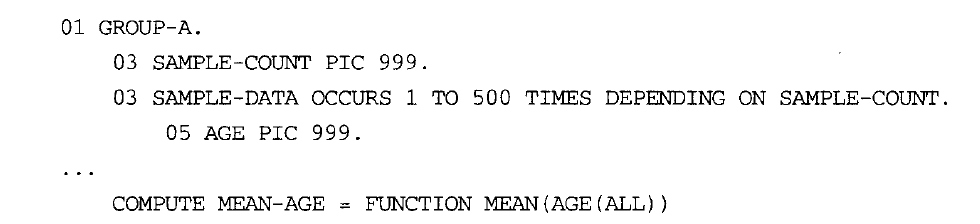
FUNCTION function-name [({argumengt})…………]

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。



# COBOL综合实例

## 综合实例一：个人股票交易管理

## 综合实例二：学生管理信息系统

### 系统分析

### 系统概况

学生管理信息系统广泛地应用于各大中院校的学生管理工作过程中，几乎每个进行了信息化建设的学校都建设了学生管理信息系统。图9-1和图9-2是两个典型的学生管理信息系统的界面。

学生管理信息系统的界面



         典型的学生管理信息系统的操作界面

从界面可以看到，一个典型的学生管理信息系统应该提供包括院系设置管理、学科管理、以及学生出勤管理和奖惩管理等内容。其中，

(1) 院系设置主要用于设置本学校的院系情况。

(2) 学科管理用于设置本院系的学科情况。

(3) 学生的出勤奖惩管理主要是录入和查询学生的出勤和奖惩情况。

(4) 后台管理(系统维护)。对数据库表的备份和恢复，数据库表的维护等。

学生管理信息系统作为数据库项目应用的一种，其开发流程包括需求分析、UML系统建模、确定系统集成方案、数据库分析和设计以及各功能模块的开发等。

### 需求分析

开发学生管理信息系统的第一步是进行需求分析。需求分析的好坏直接决定着系统能否真正满足用户的需要。可以将需求分析分为两个过程：一是理解需求，二是分析需求，下面分别讲解。

#### 理解需求

下面是一份典型的文本需求说明，是通过对学生管理信息系统的终端用户和客户进行调研后编写的。因为篇幅所限，这里只列出比较关键的部分。

(1) 要求系统可以准确地记录和查询学生信息，包括学生的姓名、单位、年龄、性别以及身份证号码等。

(2) 要求系统可以准确地记录学生地每一次奖惩情况以及每次的缺勤情况。

(3) 系统可以对学校的院系情况进行管理，包括设置学院名称、修改某学院某专业方向的名称等。

(4) 系统应该可以对基础数据进行维护。

(5) 系统还应该提供强大数据统计、查询、报表生成以及打印等功能。

(6) 系统客户端运行在Windows平台下，服务端可以运行在Windows平台或者UNIX平台下。系统还应该有一个较好的图形用户界面。

(7) 系统应该有很好的可扩展性。

#### 分析需求

分析需求就是描述系统的需求，通过定义系统中的关键域类来建立模型。分析的根本目的是在开发者和提出需求的人之间建立一种理解和沟通的机制。因此，学生管理信息系统的需求分析也应该是开发人员和用户或者客户一起完成的。

分析需求的第一步描述学生管理信息系统的功能，即定义用例，以此确定系统的功能需求。学生管理信息系统的用例分析主要涉及规格说明的阅读和分析，需要和学生管理信息系统的潜在用户进行讨论。

学生管理系统的角色是管理者和系统管理员。管理者是管理学生的人员，如教务处或者学生处的员工，他也是系统的用户，而系统管理员是系统管理者，虽然系统管理员也是一个员工，但这并不影响本系统的功能。同样，管理者也可能是一个学生，但这也不影响系统的功能。

学生管理信息系统的用例主要包括如下内容。

● 记录学生情况

● 修改学生信息

● 统计查询学生情况

● 记录学生考勤情况

● 修改学生考勤情况

● 统计查询学生考勤情况

● 记录学生奖惩情况

● 修改学生奖惩情况

● 统计查询学生奖惩情况

● 设置院系情况

学生管理信息系统的分析可以用UML的用例图来描述。如图9-3所示。在此之前，需要将每个用例以文本的方式描述，描述的内容包括用例以及用例与角色交互的更详细的信息。文本的内容是通过与用户讨论后确定的。下面给出上述用例的描述。

(1) 记录学生情况

查询院系情况→记录学生信息。

(2) 修改学生信息  
  
查询学生信息→修改学生信息。

(3) 统计查询学生情况

设置统计查询条件→统计查询学生信息→打印统计查询结果。

(4) 记录学生考勤情况

查询学生基本信息→记录学生考勤情况。

(5) 修改学生考勤情况

查询学生基本信息→查询学生考勤情况→修改学生考勤情况。

(6) 统计查询学生考勤情况

查询学生基本信息→输入查询统计条件→统计查询学生考勤情况→打印统计查询学生考勤结果。

(7) 记录学生奖惩情况

查询学生基本信息→记录学生奖惩情况。

(8) 修改学生奖惩情况

查询学生基本信息→查询学生奖惩情况→修改学生奖惩情况。

(9) 统计查询学生奖惩情况

查询学生基本信息→输入查询统计条件→统计查询学生奖惩情况→打印统计查询学生奖惩结果。

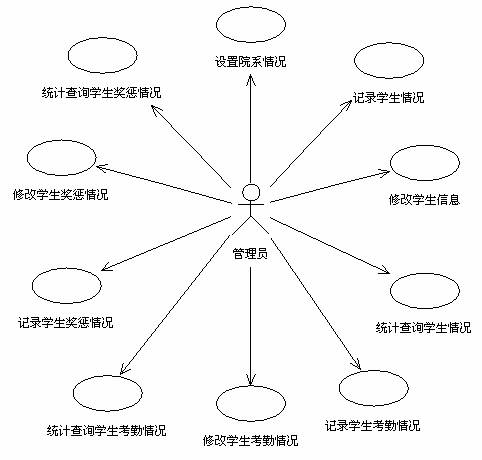
(10) 设置院系情况

增加院系情况。

## UML系统建模

需求分析后，就可以对系统进行UML建模了。下面分别讲解学生管理信息系统的UML建模过程。

### 学生管理信息系统的用例分析

在需求分析中对列出了学生管理信息系统的全部用例，包括记录学生情况、修改学生信息、统计查询学生情况、记录学生考勤情况、修改学生考勤情况、统计查询学生考勤情况、记录学生奖惩情况、修改学生奖惩情况、统计查询学生奖惩情况以及设置院系情况。这里，使用Rose软件进行UML建模。如图所示。  


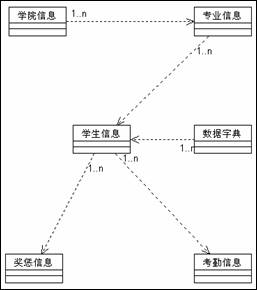
**学生管理信息系统用例图**

**该用例图标记了所有的用例，从中可以得知，基本的学生管理信息系统的角色只有管理员，他将承担所有用例的使用。**

### 学生管理信息系统的领域分析

**UML建模的第二步就是领域分析。实际开发学生管理信息系统时，领域分析是建立在用例分析基础上的。要了解系统要处理的概念，这时最好将院校的有关人员组织起来开一个讨论会，了解概念和概念之间的关系。**

**学生管理信息系统中的域主要包括：学生信息、学院信息、专业信息、奖惩信息、考勤信息和数据字典。可以在类图中将上面这些域以及它们之间的关系表示出来。如图9-4所示。**

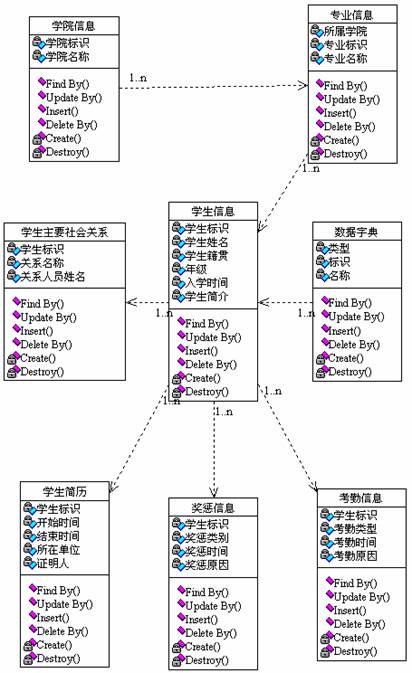


**学生管理信息系统域略图**

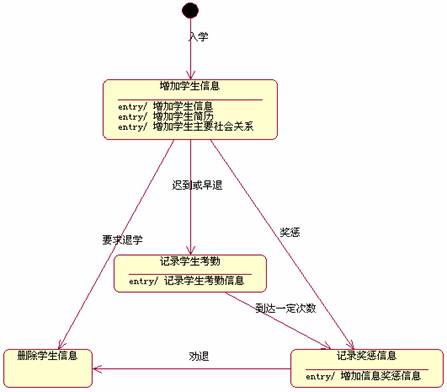
**如图所示，首先给出了学生管理信息系统的域略图，在该图中，仅给出了各个域的名称以及各个域间的关系，在此基础上，对域再进行分析，可以得到域草图。如图所示。**

**在从略图到详图的过程中，可以看到增加了一些对象。在实际开发过程中，这种情况比较常见。这主要是因为，在将略图细化的过程中，可能会发现，为更好地描述系统需要增加一些类，这些工作可以在设计详图时完成。**

**有些类可以用UML状态图来显示类的对象的不同状态以及改变状态的事件。在本系统中有状态图的类只有学生信息。该类的状态图将在随后介绍。**

**为了描述域类的动态行为，可以使用UML的时序图、协作图或者活动图。这里，选用时序图。时序图的基础是用例。在时序图中，要说明域类是如何协作以操作系统中的用例。当然，在建立时序图时，将会发现新的操作，并将其加入类中，将在后面看到所建立的时序图模型。此外，操作仅仅是草案，同样也要用说明详细描述。分析的目的是同企业的工作人员进行沟通过以对要建立的系统有更好的了解，而不是要设计一个完整详细的学生管理信息系统的设计方案。**

**学生管理信息系统域草图**

用时序图建模时，需要窗口或对话框作为角色的接口。显然，这里需要窗口的有记录学生情况、修改学生信息、统计查询学生情况、记录学生考勤情况、修改学生考勤情况、统计查询学生考勤情况、记录学生奖惩情况、修改学生奖惩情况、统计查询学生奖惩情况以及设置院系情况等，此外维护也需要一个窗口。当然，这里还没有定义详细的用户接口。用户接口所包含的内容也仅仅是一个草案。详细的用户接口，将在设计阶段对其进行定义。  
  
学生管理信息系统的设计  
在设计阶段，首先要设计类的状态图。不是所有的类都有状态图。在本系统中，有状态图的类只有学生信息。  
  
学生信息的状态图如图所示。  
  
  
客户预定的状态图

## 系统配置

学生管理系统的系统配置包括软件、硬件和网络的配置。

### 软件配置

软件配置主要包括数据库的选择和操作系统的选择。学生管理信息系统的软件配置要根据用户对系统的稳定性要求、系统的容量以及用户的维护水平来确定。

(1) 操作系统选择

服务器端的选择如表所示。

学生管理信息系统操作系统的选择

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用    户 | 每秒数据库并发数据访问记录数 | 稳定性  要求 | 维护  水平 | 备选操作系统 |
| 社区小学 | 10~50 | 低 | 差 | Windows 2000 Professional,  Windows XP |
| 公立小学或社区中学 | 50~500 | 中 | 一般 | Windows NT, Windows 2000 Server, Windows Server 2003 |
| 公立中学或社区大学 | 500~2000 | 较高 | 较好 | Windows NT, Windows 2000 Server, Windows Server 2003, Linux, UNIX |
| 专业型大学 | 500~2000 | 较高 | 较好 | UNIX |
| 综合型大学 | 2000以上 | 高 | 好 | UNIX |

可以选择Windows 2000 Server操作系统等。

(2) 数据库选择

可以根据数据量的大小选择不同的数据库，如表所示。

学生管理信息系统数据库的选择

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用    户 | 数据量(记录数) | 稳定性要求 | 维护水平 | 备选数据库 |
| 社区小学 | 100~500 | 低 | 差 | Access, FoxPro |
| 公立小学或社区中学 | 500~5000 | 低 | 差 | Access, FoxPro |
| 公立中学或社区大学 | 5000~50 000 | 中 | 一般 | SQL Server |
| 专业型大学 | 50000~200 000 | 较高 | 较好 | SQL Server, Oracle |
| 综合型大学 | 200 000以上 | 高 | 好 | Oracle |

当然，这种选择并不是绝对的。有时，一个学校规模不大，但是学生流动数量大，要求数据库的吞吐量和稳定性较高，这就需要选择更好的数据库。也有些情况，由于建设经费的限制，只能选择较为便宜的数据库，这要根据用户的需要而定。数据库的选择还要考虑与操作系统的配合。

### 硬件配置

硬件配置主要包括客户端硬件的选择和服务器端硬件的选择。学生管理信息系统的硬件配置要根据用户对系统的稳定性要求、系统的容量、系统的吞吐量以及用户的维护水平来确定。

(1) 客户端硬件选择

可以根据稳定性要求选择不同的客户端。一般情况下，对客户端的要求不高。如表所示。

表9-3  学生管理信息系统客户端硬件的选择

|  |  |  |
| --- | --- | --- |
| 用    户 | 稳定性要求 | 备选客户端 |
| 查询客户端 | 低 | PentiumCPU/64MB/4GB |
| 管理客户端 | 中 | PentiumⅢ/256MB/20GB |

(2) 服务器端硬件选择

可以根据数据量和吞吐量的大小选择不同的服务器硬件。服务器端的选择如表9-4所示。

学生管理信息系统服务器端硬件的选择

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用    户 | 每秒数据库并发数据访问记录数 | 稳定性  要求 | 维护  水平 | 备选服务器 |
| 社区小学 | 10~50 | 低 | 差 | PentiumⅢ/256MB/20GB个人计算机 |
| 公立小学或社区中学 | 50~500 | 中 | 一般 | XEON(Xeon)双CPU/1GB/RAID5 3×72GB |
| 公立中学或社区大学 | 50~500 | 中 | 一般 | XEON(Xeon)双CPU/1GB/RAID5 3×72GB |
| 专业型大学 | 500~2000 | 较高 | 较好 | 小型计算机/双机热备  或者XEON(Xeon)4CPU/4GB/RAID5 6×72GB |
| 综合型大学 | 2000以上 | 高 | 好 | 小型计算机/双机热备  或者XEON(Xeon)4CPU/4GB/RAID5 6×72GB |

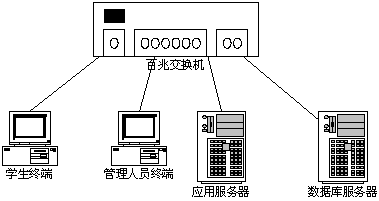
当然，这种配置不是绝对的，这里只给出参考配置的范围。具体的配置需要根据用户的需求精确计算。

### 网络配置

学生管理信息系统的网络配置一般有如下几种类型。

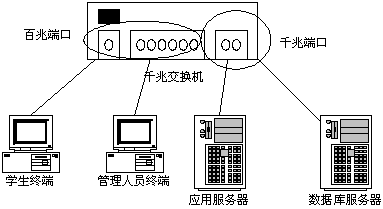
(1) 单机模式：即服务器和客户端在一台计算机上。

(2) 百兆模式：计算机网络的速度为百兆。如图所示。



学生管理信息系统网络百兆配置

(3) 千兆模式：计算机网络的速度为千兆。实际上，没有必要给每个节点都配备千兆网络接口，只需要给服务器端配备千兆接口即可。如图所示。



学生管理信息系统网络千兆配置

可以根据用户情况选择不同的网络配置，如表所示。

 学生管理信息系统网络配置的选择

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用    户 | 每秒数据库并发 数据访问记录数 | 稳定性  要求 | 维护  水平 | 备选网络方案 |
| 社区小学 | 10~50 | 低 | 差 | 单机模式 |
| 公立小学或社区中学 | 50~500 | 中 | 一般 | 百兆模式 |
| 公立中学或社区大学 | 50~500 | 中 | 一般 | 百兆模式 |
| 专业型大学 | 500~2000 | 较高 | 较好 | 百兆模式 |
| 综合型大学 | 2000以上 | 高 | 好 | 千兆模式 |

## 数据库分析

数据库分析是数据库项目开发周期中的一个重要的阶段，也是工作量比较大的一项活动。随着现代软件的发展，手工分析方式已经很难满足数据库系统数据库分析的要求，必须借助相应的工具。在开发学生管理信息系统时，可以先进行E-R图分析，然后对表字段进行分析，最后进行数据库建模。PowerDesign是数据库Sybase公司提供的数据库分析工具，具备了上述全部功能，本节将利用该工具完成E-R图分析和数据库建模。

### E-R图分析

    对学生管理信息系统的E-R图分析是建立在UML系统模型基础上的。这里，直接给出E-R分析的结果。实体关系图的分析结果非常复杂，一般情况下使用从简到繁的方式进行设计。首先从大的方面中设计出各个实体之间的关系，然后可以在这个关系的基础上进行细化。

如图9-9所示，为学生管理信息系统的实体关系简图。

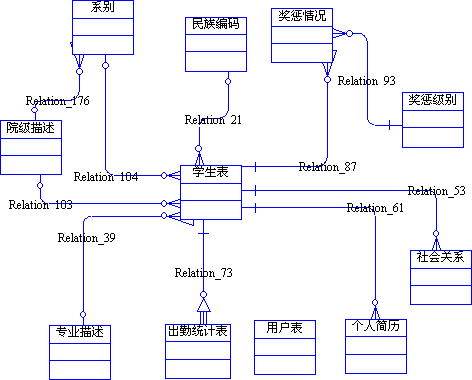


图9-9  学生管理信息系统的实体关系简图

在这个简图的基础上进一步设计实体关系的详细结构。如图9-10所示，为学生管理信息系统的实体关系详图。

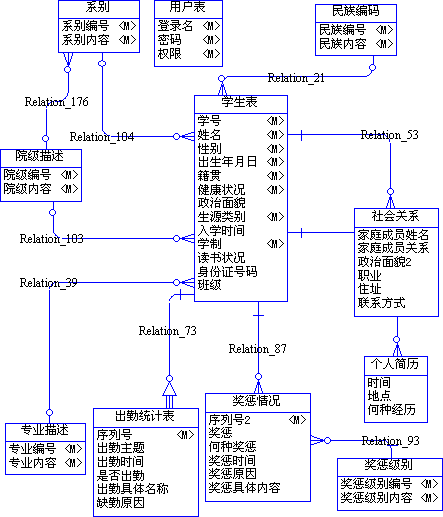


图9-10  学生管理信息系统的实体关系图

为便于编程，将所有的属性都使用中文全拼的首字母表示，用汉字在实体关系图中显示。当然，现在有些数据库也支持中文字段名，但是考虑到系统的兼容性，推荐大家都使用英文编写数据表名和字段。实际上，图9-10只是学生管理信息系统最基本元素的实体关系图，读者可以在此基础上根据用户的不同需要进行扩展。考虑到篇幅所限，这里就不再详细介绍其他扩展了。

### 表与字段分析和PowerDesign数据库建模

表与字段分析是建立在实体关系图基础上的。表与字段分析后就可以利用PowerDesign建立数据库模型了。以图9-10的学生管理信息系统的实体关系图为基础，可设计表字段，然后建立数据库模型。当然，如果是使用PowerDesign设计的E-R模型，则可以使用PowerDesign自带的工具产生表与字段并建立了数据库模型。此外，ERWin 4.0也可以实现这个功能。对于比较复杂的数据库最后使用上面提到的设计工具进行表与字段分析，以减轻工作量并提高设计质量。

建立的数据库模型如图9-11所示。

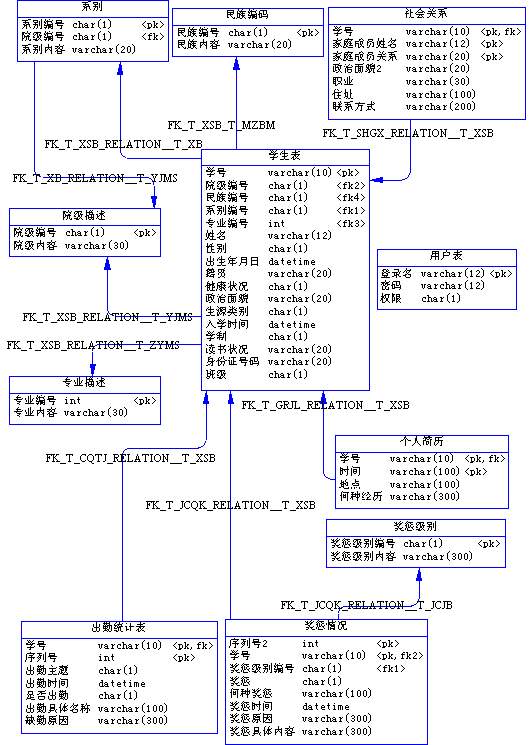


图9-11  学生管理信息系统的数据库模型

## 数据库设计

    有了数据库模型，数据库的设计就简单多了。在学生管理信息系统中，首先要创建学生管理信息系统数据库，然后在数据库中创建需要的表和字段。如果有需要，还可以设计视图、存储过程和触发器。下面分别讲述在示例系统中数据库的设计。

一 创建数据库

**1 创建数据库（ORACLE9i）**

（1） 启动数据库

（2）启动企业管理器

（3）新建数据库

　创建数据库（SQL Server 2000）

    示例中使用SQL Server 2000为基础数据库，可以使用SQL Server 2000自带的数据库工具创建数据库。这里使用SQL Server 2000的工具创建数据库，步骤如下：

(1) 启动SQL Server 2000数据库。

首先在服务器端操作界面上单击“开始”|“程序”| Microsoft SQL Server |“服务管理器”，然后单击“启动”按钮即启动SQL Server2000。

(2) 启动企业管理器。

再单击“开始”|“程序”| Microsoft SQL Server |“企业管理器”，启动数据库的管理界面。

(3) 新建数据库。

展开Microsoft SQLServer | local |“数据库”后右击“数据库”，在弹出菜单中选择“新建数据库”命令后弹出新建数据库窗口。如图9-12所示。



图9-12  新建数据库窗口

在“名称”项中输入新数据库的名称XSGL。单击“数据文件”标签页后输入数据库文件的存放位置，单击“事务日志”标签页后输入数据库日志的存放位置。如图9-13所示。

设置完成后单击“确定”按钮即完成数据库的创建。

图9-13  数据库文件和日志文件的存放位置

**3 也可以直接使用SQL语句创建数据库**

创建销售信息管理数据库的SQL语句代码如下：

CREATE DATABASE XSGL

ON

( NAME =XSGL\_dat,

   FILENAME = 'c:\program files\microsoft sql server\mssql\data\ XSGL \_dat.mdf',

   SIZE = 10,

   MAXSIZE = 50,

   FILEGROWTH = 5 )

LOG ON

( NAME = ' XSGL\_log',

   FILENAME = 'c:\program files\microsoft sql server\mssql\data\ XSHL\_log.ldf',

   SIZE = 5MB,

   MAXSIZE = 25MB,

   FILEGROWTH = 5MB )

也可以直接使用ERWin或者PowerDesign的数据库生成工具完成数据库的创建。读者可以参考相关的技术手册，这里不再赘述。

创建表

创建XSGL数据库后，再在该数据库中创建数据表。

image015

图9-14   创建表

可以在这里输入数据表信息以完成数据表的创建。

创建数据表的SQL语句如下：

create table T\_CQTJ (

XH                varchar(10)               not null,

CQ\_ID             int                       not null,

CQZT              char(1)                   null default '1'

constraint CKC\_CQZT\_T\_CQTJ check (CQZT is null or ( CQZT in ('1','2','3') )),

CQSJ                 datetime                    null,

SFCQ                char(1)                     null default '1'

constraint CKC\_SFCQ\_T\_CQTJ check (SFCQ is null or ( SFCQ in ('1','2') )),

CQJTMC             varchar(100)                 null,

QQYY               varchar(300)                 null,

constraint PK\_T\_CQTJ primary key  (XH, CQ\_ID)

)

create   index RELATION\_73\_FK on T\_CQTJ (

XH

)

create table T\_GRJL (

XH                    varchar(10)                  not null,

SJ                   varchar(100)                 null,

DD                varchar(100)                 null,

HZJL              varchar(300)                 null

)

create   index RELATION\_61\_FK on T\_GRJL (

XH

)

create table T\_JCJB (

JCJBBH              char(1)                     not null,

JCJBNR            varchar(300)                 not null,

constraint PK\_T\_JCJB primary key  (JCJBBH)

)

create table T\_JCQK (

JC\_ID                 int                        not null,

XH                varchar(10)                  not null,

JCJBBH            char(1)                     not null,

JC                   char(1)                     null

     constraint CKC\_JC\_T\_JCQK check (JC is null or ( JC in ('1','0') )),

HZJC                 varchar(100)            null,

JCSJ               datetime                    null,

JCYY                        varchar(300)                 null,

JCMS                       varchar(300)                 null,

constraint PK\_T\_JCQK primary key  (JC\_ID)

)

create   index RELATION\_93\_FK on T\_JCQK (

JCJBBH

)

create   index RELATION\_87\_FK on T\_JCQK (

XH

)

create table T\_MZBM (

MZBH               char(1)                     not null,

MZNR               varchar(20)                  not null,

constraint PK\_T\_MZBM primary key  (MZBH)

)

create table T\_SHGX (

XH                varchar(10)                  not null,

F\_XM               varchar(12)                  null,

F\_GX                varchar(20)                  null,

F\_ZZMM          varchar(20)                  null,

F\_ZY               varchar(30)                  null,

F\_ZZ              varchar(100)                 null,

F\_LXFS            varchar(200)                 null

)

create   index RELATION\_53\_FK on T\_SHGX (

XH

)

create table T\_USER (

YHM                varchar(12)                  not null,

MM                        varchar(12)                  not null,

QX                  char(1)                     not null default '3'

     constraint CKC\_QX\_T\_USER check (QX in ('1','2','3')),

constraint PK\_T\_USER primary key  (YHM)

)

create table T\_XB (

XBBH               char(1)                     not null,

YJBH              char(1)                     null,

XBNR                varchar(20)                  not null,

constraint PK\_T\_XB primary key  (XBBH)

)

create   index RELATION\_176\_FK on T\_XB (

YJBH

)

create table T\_XSB (

XH                   varchar(10)                  not null,

YJBH                 char(1)                     null,

MZBH               char(1)                     null,

XBBH               char(1)                     null,

ZYBH               int                        null,

XM                 varchar(12)                  not null,

XB                 char(1)                     not null

     constraint CKC\_XB\_T\_XSB check (XB in ('1','2')),

CSNY                datetime                    not null,

JG                           varchar(20)                  not null,

JKZK                 char(1)                     not null

     constraint CKC\_JKZK\_T\_XSB check (JKZK in ('1','2','3')),

ZZMM                      varchar(20)                  null,

SYLB                char(1)                     not null

     constraint CKC\_SYLB\_T\_XSB check (SYLB in ('1','2','3')),

RXSJ                datetime                    null,

XZ                char(1)                     not null default '4',

DSZK                varchar(20)                  null

     constraint CKC\_DSZK\_T\_XSB check (DSZK is null or ( DSZK in ('1','2','3','4','5','6') )),

SFZ\_ID               varchar(20)                  null,

BJ                   char(1)                     null,

constraint PK\_T\_XSB primary key  (XH)

)

create   index RELATION\_104\_FK on T\_XSB (

XBBH

)

create   index RELATION\_103\_FK on T\_XSB (

YJBH

)

create   index RELATION\_39\_FK on T\_XSB (

ZYBH

)

create   index RELATION\_21\_FK on T\_XSB (

MZBH

)

create table T\_YJMS (

YJBH                 char(1)              not null,

YJNR                 varchar(30)          not null,

constraint PK\_T\_YJMS primary key  (YJBH)

)

create table T\_ZYMS (

ZYBH                 int                  not null,

ZYNR                 varchar(30)          not null,

constraint PK\_T\_ZYMS primary key  (ZYBH)

)

## 界面设计

学生管理信息系统的界面应该分为4大部分。

●     学生管理主界面

●     学生基本信息界面

●     奖惩界面

●     考勤界面

## 学生管理主界面设计

学生管理主界面主要是为其他功能，例如学生基本信息、出勤情况和奖惩情况，提供一个入口。窗体设计如图9-16所示。



图9-16  学生管理主窗体的设计

这里，分别为学生基本信息、出勤情况、奖惩情况以及系统帮助设置了菜单项。如图9-17所示。



图9-17  学生管理主窗体的菜单项

### 学生基本信息界面的设计

### 学生基本信息窗体要完成信息的录入查询功能，因此需要设计菜单。该窗体的设计如图9-18所示。



图9-18  学生基本信息主窗体设计

录入个人信息的窗体设计如图9-19所示。在这个窗体中，要完成学生基本信息、学生的社会关系以及学生个人简历的录入，这里使用不同的标签页完成这些功能。



图9-19  学生基本信息录入窗体设计

单击“社会关系”标签页，将显示社会关系录入窗体。如图9-20所示。

单击“个人简历”标签页，将显示个人简历录入窗体。如图9-21所示。

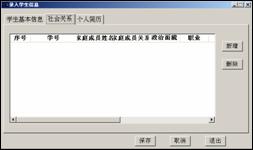
  

图9-20  学生社会关系录入窗体设计         图9-21  个人简历信息录入窗体设计

录入外部信息的窗体设计如图9-22所示。在这个窗体中，要完成院级信息、系别信息、专业信息以及民族编码的录入，这里使用不同的标签页完成这些功能。



图9-22  学院信息录入窗体设计

奖惩界面设计

奖惩界面需要完成学生奖惩的录入、查询和修改，因此该窗口也使用MDI格式。该窗体的设计如图9-23所示。



图9-23  学生奖惩界面的设计

在这个窗体中，可以录入学生奖惩的情况。界面如图9-24所示。



图9-24  奖惩录入界面的设计

在奖惩处理的主窗体菜单中，单击查询或者修改菜单项应该可以调出奖惩的查询和修改界面。如图9-25和9-26所示。

图9-25  奖惩的查询界面的设计               图9-26  奖惩的修改界面的设计

学生考勤界面设计

学生考勤包括录入出勤、出勤查询和出勤修改，这些窗体的设计如图9-27、图9-28和图9-29所示。



图9-27  考勤录入界面的设计

图9-28  考勤查询界面的设计                   图9-29  考勤修改界面的设计

至此，学生管理信息系统管理系统的主要界面介绍完毕。以上描述的窗体只是实际应用系统的一部分，读者可以根据实际用户的需求设计用户满意的窗体。总之，窗体的设计应该以用户的要求为主。

此外，除了上述的功能窗体，还应该为系统管理员设计一个用于维护数据库和数据库表的窗体，考虑到这类窗体的通用性比较强，在前几章已经介绍过类似的功能窗体，这里不再赘述。

## 连接数据库

## 功能模块设计

    界面设计后就可以在此基础上进行功能模块的设计了。学生管理信息系统各功能模块的划分实际上已经体现在界面设计之中了。其功能模块设计的核心实际上就是数据窗口的设计。

(1) 学生基本信息显示数据窗口

学生基本信息数据窗口主要用于学生的增加、显示和检索，可以以t\_xsb表为基础创建学生基本信息数据窗口。如图9-30和图9-31所示。

image031

图9-30  学生基本信息显示数据窗口(左部分)

image032

图9-31  学生基本信息显示数据窗口(右部分)

该数据窗口的代码为：

    SELECT  "T\_XSB"."XH" ,

           "T\_XSB"."XM" ,

           "T\_XSB"."XB" ,

           "T\_XSB"."YJBH" ,

           "T\_XSB"."XBBH" ,

           "T\_XSB"."ZYBH" ,

           "T\_XSB"."BJ" ,

           "T\_XSB"."CSNY" ,

           "T\_XSB"."MZBH" ,

           "T\_XSB"."JG" ,

           "T\_XSB"."JKZK" ,

           "T\_XSB"."ZZMM" ,

           "T\_XSB"."SYLB" ,

           "T\_XSB"."RXSJ" ,

           "T\_XSB"."XZ" ,

           "T\_XSB"."DSZK" ,

           "T\_XSB"."SFZ\_ID"

        FROM "T\_XSB"

(2) 用于考勤数据窗口设计

学生考勤信息数据窗口主要用于学生考勤情况的增加、显示和检索。如图9-32所示，为处理考勤的数据窗口。

image033

image034

图9-32  用于处理考勤的数据窗口

该数据窗口的语法为：

SELECT  "v\_all\_cqtj"."XH" ,

           "v\_all\_cqtj"."XM" ,

           "v\_all\_cqtj"."XB" ,

           "v\_all\_cqtj"."CQZT" ,

           "v\_all\_cqtj"."CQSJ" ,

           "v\_all\_cqtj"."SFCQ" ,

           "v\_all\_cqtj"."CQJTMC" ,

           "v\_all\_cqtj"."QQYY" ,

           "v\_all\_cqtj"."YJNR" ,

           "v\_all\_cqtj"."XBNR" ,

           "v\_all\_cqtj"."ZYNR"

        FROM "v\_all\_cqtj"

(3) 用于奖惩的数据窗口设计

学生奖惩信息数据窗口主要用于学生奖惩情况的增加、显示和检索。如图9-33所示，为处理奖惩的数据窗口。

image035

图9-33  处理奖惩的数据窗口

该数据窗口的语法为：

  SELECT  "T\_XSB"."XH" ,

           "T\_XSB"."XM" ,

           "T\_XSB"."XB" ,

           "T\_MZBM"."MZNR" ,

           "T\_YJMS"."YJNR" ,

           "T\_XB"."XBNR" ,

           "T\_ZYMS"."ZYNR" ,

           "T\_JCQK"."JC" ,

           "T\_JCQK"."HZJC" ,

           "T\_JCQK"."JCSJ" ,

           "T\_JCJB"."JCJBNR"

 FROM      "T\_JCJB" ,

           "T\_JCQK" ,

           "T\_XB" ,

           "T\_XSB" ,

           "T\_YJMS" ,

           "T\_ZYMS" ,

           "T\_MZBM"

        WHERE ( "T\_JCQK"."JCJBBH" = "T\_JCJB"."JCJBBH"  )

              and  ( "T\_XSB"."XBBH"  =  "T\_XB"."XBBH"  )

              and  ( "T\_XSB"."XH"    =  "T\_JCQK"."XH"  )

              and  ( "T\_YJMS"."YJBH" =  "T\_XSB"."YJBH" )

              and  ( "T\_YJMS"."YJBH" =  "T\_XB"."YJBH"  )

              and  ( "T\_ZYMS"."ZYBH" =  "T\_XSB"."ZYBH" )

              and  ( "T\_MZBM"."MZBH" =  "T\_XSB"."MZBH" )

其他数据窗口的创建比较简单（略）。

## 代码分析与实现

    学生管理信息系统的代码实现包括学生信息的录入、查询和修改；学生考勤信息的录入、查询和修改；学生奖惩信息的录入、查询和修改以及系统的维护等。由于这些模块的实现比较类似，这里只介绍查询代码实现。

(1) 在应用中连接数据库

首先要在应用中创建事务对象并连接数据库。

(2) 在学生信息查询界面的代码

在学生查询界面中，列出了可以进行查询的字段。输入字段后，即可完成查询。窗口如图9-34所示。



图9-34  查询界面

在查询界面的“查询”按钮编写代码如下：

string wheresql

string oldsql,newsql

string ls\_temp

string ls\_zjbl

ls\_temp = trim(sle\_1.text)

if ls\_temp = '' then

  messagebox('提示','请输入查询条件',StopSign!)

  sle\_1.setfocus()

  return;

end if

if ddlb\_1.text = '学号' then

  wheresql="where v\_all\_jbxx.xh='"+ls\_temp+"'"

end if

if ddlb\_1.text = '姓名' then

  wheresql="where v\_all\_jbxx.xm='"+ls\_temp+"'"

end if

if ddlb\_1.text = '系别' then

  wheresql="where v\_all\_jbxx.xbnr='"+ls\_temp+"'"

end if

if ddlb\_1.text = '专业' then

  wheresql="where v\_all\_jbxx.zynr='"+ls\_temp+"'"

end if

if ddlb\_1.text = '民族' then

  wheresql="where v\_all\_jbxx.mznr='"+ls\_temp+"'"

end if

if ddlb\_1.text = '读书状况' then

  CHOOSE CASE ls\_temp

           CASE "在校"

                    ls\_zjbl='1'

           case "已毕业"

                    ls\_zjbl='2'

           case "肄业"

                    ls\_zjbl='3'

           case "降级"

                    ls\_zjbl='4'

           case "跳级"

                    ls\_zjbl='5'

           case "其他"

                    ls\_zjbl='6'

  END CHOOSE

  wheresql="where v\_all\_jbxx.dszk="+ls\_zjbl+""

end if

if ddlb\_1.text = '生源类别' then

  CHOOSE CASE ls\_temp

           CASE "统招"

                    ls\_zjbl='1'

           case "委培"

                    ls\_zjbl='2'

           case "其他"

                    ls\_zjbl='3'

  END CHOOSE

  wheresql="where v\_all\_jbxx.sylb="+ls\_zjbl+""

end if

//合成新的条件

dw\_1.settransobject(sqlca)

oldsql = dw\_1.getsqlselect()

newsql=oldsql+wheresql

//数据窗口进行精确查询

dw\_1.setsqlselect(newsql)

dw\_1.retrieve()

//复位，便于继续查询

dw\_1.setsqlselect(oldsql)

    执行后将会在界面下方的数据窗口显示查询结果。考勤信息和奖惩信息的查询与此类似，这里不再赘述。信息的插入和存盘的代码比较简单，请读者自己参考本书光盘所带示例。

(3) 学生基本信息录入窗口的代码分析

学生基本信息录入窗口的代码要实现学生基本信息数据库表的插入，该窗口如图9-35所示。



图9-35  学生基本信息录入窗口

在“学生基本信息”标签页的Clicked事件编写代码如下：

string ls\_xh

string oldsql1,newsql1,wheresql1

string oldsql2,newsql2,wheresql2

int li\_row

select max(shgx\_id) into :il\_shgx\_maxrow from t\_shgx;

select max(grjl\_id) into :il\_grjl\_maxrow from t\_grjl;

li\_row=tabpage\_1.dw\_1.getrow()

if li\_row<=0 then

  return

end if

ls\_xh=tabpage\_1.dw\_1.getitemstring(li\_row,"xh")

wheresql1="where t\_shgx.xh="+ls\_xh+""

wheresql2="where t\_grjl.xh="+ls\_xh+""

//合成新的条件

tabpage\_2.dw\_2.settransobject(sqlca)

oldsql1 = tabpage\_2.dw\_2.getsqlselect()

newsql1=oldsql1+wheresql1

//数据窗口进行精确查询

tabpage\_2.dw\_2.setsqlselect(newsql1)

tabpage\_2.dw\_2.retrieve()

//复位，便于继续查询

tabpage\_2.dw\_2.setsqlselect(oldsql1)

//合成新的条件

tabpage\_3.dw\_3.settransobject(sqlca)

oldsql2 = tabpage\_3.dw\_3.getsqlselect()

newsql2=oldsql2+wheresql2

//数据窗口进行精确查询

tabpage\_3.dw\_3.setsqlselect(newsql2)

tabpage\_3.dw\_3.retrieve()

//复位，便于继续查询

tabpage\_3.dw\_3.setsqlselect(oldsql2)

在Create事件编写代码如下：

this.tabpage\_1=create tabpage\_1

this.tabpage\_2=create tabpage\_2

this.tabpage\_3=create tabpage\_3

this.Control[]={this.tabpage\_1,&

this.tabpage\_2,&

this.tabpage\_3}

在Destroy事件编写代码如下：

destroy(this.tabpage\_1)

destroy(this.tabpage\_2)

destroy(this.tabpage\_3)

在窗口“检索条件”按钮下编写代码如下：

dw\_1.y=dw\_1.y + 100

dw\_1.height=dw\_1.height - 100

st\_1.visible=true

cb\_10.enabled=true

this.visible=false

dw\_1.reset()

dw\_1.modify('datawindow.querymode=yes')

在窗口“检索”按钮下编写代码如下：

if st\_1.visible=true then

    st\_1.visible=false

   dw\_1.y=dw\_1.y - 100

    dw\_1.height=dw\_1.height + 100

    cb\_13.visible=true

end if

dw\_1.retrieve()

dw\_1.modify('datawindow.querymode=no')

this.enabled=false

在“保存”按钮下编写代码实现对多个标签页中数据的保存，代码如下：

string ls\_err

if tab\_1.tabpage\_1.dw\_1.update(true,false)<>1 then

   ls\_err=sqlca.sqlerrtext

   rollback;

   messagebox('错误信息','数据存盘失败！！~r~n'+ls\_err)

   return

end if

commit;

tab\_1.tabpage\_1.dw\_1.resetupdate()

if tab\_1.tabpage\_2.dw\_2.update(true,false)<>1 then

   ls\_err=sqlca.sqlerrtext

   rollback;

   messagebox('错误信息','数据存盘失败！！~r~n'+ls\_err)

   return

end if

commit;

tab\_1.tabpage\_2.dw\_2.resetupdate()

if tab\_1.tabpage\_3.dw\_3.update(true,false)<>1 then

   ls\_err=sqlca.sqlerrtext

   rollback;

   messagebox('错误信息','数据存盘失败！！~r~n'+ls\_err)

   return

end if

commit;

tab\_1.tabpage\_3.dw\_3.resetupdate()

select max(shgx\_id) into :il\_shgx\_maxrow from t\_shgx;

select max(grjl\_id) into :il\_grjl\_maxrow from t\_grjl;

(4) 考勤情况统计窗口的代码分析

考勤情况统计窗口要按照条件对考勤情况进行统计查询，该窗口如图9-26所示。在该窗口的“检索”按钮编写代码如下：

string oldsql,newsql

string wheresql=''

date   ld\_begin,ld\_end

string ls\_temp

//处理学号

if cbx\_1.checked = true then

  ls\_temp = trim(sle\_1.text)

  if ls\_temp <> '' then

           if wheresql = '' then

                    wheresql = "where v\_all\_cqtj.xh='"+ls\_temp+"'"

           else

                    wheresql = wheresql + " and v\_all\_cqtj.xh= ' "+ls\_temp+" ' "

           end if

  end if

end if

//处理姓名

if cbx\_2.checked = true then

  ls\_temp = trim(sle\_2.text)

  if ls\_temp <> '' then

           if wheresql = '' then

                    wheresql = "where v\_all\_cqtj.xm='"+ls\_temp+"'"

           else

                    wheresql = wheresql + " and v\_all\_cqtj.xm= ' "+ls\_temp+" ' "

           end if

  end if

end if

//出勤的内容

if cbx\_3.checked = true then

  if ddlb\_1.text = '上课' then

           ls\_temp = '1'

  else if ddlb\_1.text = '出操' then

           ls\_temp = '2'

  else

           ls\_temp = '3'

  end if

  if wheresql = '' then

           wheresql = "where v\_all\_cqtj.cqzt='"+ls\_temp+"'"

  else

           wheresql = wheresql + " and v\_all\_cqtj.cqtj= '"+ls\_temp+"'"

  end if

end if

//出缺勤

if cbx\_4.checked = true then

  if rb\_2.checked = true then

           ls\_temp = '2'

  else

           ls\_temp = '1'

  end if

  if wheresql = '' then

           wheresql = "where v\_all\_cqtj.sfcq='"+ls\_temp+"'"

  else

           wheresql = wheresql + " and v\_all\_cqtj.sfcq= '"+ls\_temp+"'"

  end if

end if

//出勤开始时间

if cbx\_5.checked = true then

  ls\_temp = em\_1.text

  if ls\_temp <> '' then

           if wheresql = '' then

                    wheresql = "where v\_all\_cqtj.cqsj>='"+em\_1.text+" ' "

           else

                    wheresql = wheresql + " and v\_all\_cqtj.cqsj>= ' "+em\_1.text+" ' "

           end if

  end if

end if

//出勤结束时间

if cbx\_6.checked = true then

  ls\_temp = em\_2.text

  if ls\_temp <> '' then

           if wheresql = '' then

                    wheresql = "where v\_all\_cqtj.cqsj<='"+em\_2.text+" ' "

           else

                    wheresql = wheresql + " and v\_all\_cqtj.cqsj<= ' "+em\_2.text+" ' "

           end if

  end if

end if

//出勤具体名称

if cbx\_7.checked = true then

  ls\_temp = trim(sle\_3.text)

  if ls\_temp <> '' then

           if wheresql = '' then

                    wheresql = "where v\_all\_cqtj.cqjtmc='"+ls\_temp+" ' "

           else

                     wheresql = wheresql + " and v\_all\_cqtj.cqjtmc= ' "+ls\_temp+" ' "

           end if

  end if

end if

//所在院名

if cbx\_8.checked = true then

  ls\_temp = trim(sle\_4.text)

  if ls\_temp <> '' then

           if wheresql = '' then

                    wheresql = "where v\_all\_cqtj.yjnr='"+ls\_temp+" ' "

           else

                    wheresql = wheresql + " and v\_all\_cqtj.yjnr= ' "+ls\_temp+" ' "

           end if

  end if

end if

//所在系名

if cbx\_9.checked = true then

  ls\_temp = trim(sle\_5.text)

  if ls\_temp <> '' then

           if wheresql = '' then

                    wheresql = "where v\_all\_cqtj.xbnr='"+ls\_temp+" ' "

           else

                    wheresql = wheresql + " and v\_all\_cqtj.xbnr= ' "+ls\_temp+" ' "

           end if

  end if

end if

//所读专业

if cbx\_10.checked = true then

  ls\_temp = trim(sle\_6.text)

  if ls\_temp <> '' then

           if wheresql = '' then

                    wheresql = "where v\_all\_cqtj.zynr='"+ls\_temp+" ' "

           else

                    wheresql = wheresql + " and v\_all\_cqtj.zynr= ' "+ls\_temp+" ' "

           end if

  end if

end if

connect using sqlca;

//合成新的条件

dw\_1.settransobject(sqlca)

oldsql = dw\_1.getsqlselect()

newsql=oldsql+wheresql

//数据窗口进行精确查询

dw\_1.setsqlselect(newsql)

dw\_1.retrieve()

//统计

sle\_7.text = string(dw\_1.Rowcount())

//复位，便于继续查询

dw\_1.setsqlselect(oldsql)

(5) 奖惩统计窗口的代码分析

奖惩统计窗口要实现按照条件对奖惩情况进行统计查询，该窗口如图9-23所示。在该窗口的“检索”按钮编写代码如下：

string lookforcondition

string lookforitem1,lookforitem2,lookforitem3

string lookforhow1,lookforhow2,lookforhow3

string lookforwhat1,lookforwhat2,lookforwhat3

string lookforoperator1,lookforoperator2

string ls\_temp

integer li\_temp

//查询条件的选择

lookforhow1=ddlb\_4.text

lookforhow2=ddlb\_5.text

lookforhow3=ddlb\_6.text

//操作符号1的选择

if ddlb\_7.text="与" then

  lookforoperator1="and"

else if ddlb\_7.text="或" then

  lookforoperator1="or"

else if ddlb\_7.text="非" then

  lookforoperator1="not"

end if

//操作符号2的选择

if ddlb\_8.text="与" then

  lookforoperator2="and"

else if ddlb\_8.text="或" then

  lookforoperator2="or"

else if ddlb\_8.text="非" then

  lookforoperator2="not"

end if

//查询段1和查询内容1的选择

if ddlb\_1.text="学号" then

  lookforitem1="xh"

  lookforwhat1="'"+trim(sle\_1.text)+"'"

else if ddlb\_1.text="姓名" then

  lookforitem1="xm"

  lookforwhat1="'"+trim(sle\_1.text)+"'"

else if ddlb\_1.text="性别" then

  lookforitem1="xb"

  if trim(sle\_1.text) = '男' then

           ls\_temp = '1'

  else

           ls\_temp = '2'

  end if

  lookforwhat1="'"+ls\_temp+"'"

elseif ddlb\_1.text="民族" then

  lookforitem1="mznr"

  lookforwhat1="'"+sle\_1.text+"'"

elseif ddlb\_1.text="所在院名" then

  lookforitem1="yjnr"

  lookforwhat1="'"+sle\_1.text+"'"

else if ddlb\_1.text="所在系名" then

  lookforitem1="xbnr"

  lookforwhat1="'"+sle\_1.text+"'"

else if ddlb\_1.text="所读专业" then

  lookforitem1="zynr"

  lookforwhat1="'"+sle\_1.text+"'"

else if ddlb\_1.text="奖惩类别" then

  lookforitem1="jc"

  if trim(sle\_1.text) = '奖励' then

           ls\_temp = '1'

  else

           ls\_temp = '0'

  end if

  lookforwhat1="'"+ls\_temp+"'"

else if ddlb\_1.text="奖惩级别" then

  lookforitem1="jcjbnr"

  lookforwhat1="'"+trim(sle\_1.text)+"'"

else if ddlb\_1.text="奖惩名称" then

  lookforitem1="hzjc"

  lookforwhat1="'"+sle\_1.text+"'"

end if

//查询段2和查询内容2的选择

if ddlb\_2.text="学号" then

  lookforitem2="xh"

  lookforwhat2="'"+trim(sle\_2.text)+"'"

else if ddlb\_2.text="姓名" then

  lookforitem2="xm"

  lookforwhat2="'"+trim(sle\_2.text)+"'"

else if ddlb\_2.text="性别" then

  lookforitem2="xb"

  if trim(sle\_2.text) = '男' then

           ls\_temp = '2'

  else

           ls\_temp = '2'

  end if

  lookforwhat2="'"+ls\_temp+"'"

else if ddlb\_2.text="民族" then

  lookforitem2="mznr"

  lookforwhat2="'"+sle\_2.text+"'"

else if ddlb\_2.text="所在院名" then

  lookforitem2="yjnr"

  lookforwhat2="'"+sle\_2.text+"'"

else if ddlb\_2.text="所在系名" then

  lookforitem2="xbnr"

  lookforwhat2="'"+sle\_2.text+"'"

else if ddlb\_2.text="所读专业" then

  lookforitem2="zynr"

  lookforwhat2="'"+sle\_2.text+"'"

else if ddlb\_2.text="奖惩类别" then

  lookforitem2="jc"

  if trim(sle\_2.text) = '奖励' then

           ls\_temp = '1'

  else

           ls\_temp = '0'

  end if

  lookforwhat2="'"+ls\_temp+"'"

else if ddlb\_2.text="奖惩级别" then

  lookforitem2="jcjbnr"

  lookforwhat2="'"+trim(sle\_2.text)+"'"

else if ddlb\_2.text="奖惩名称" then

  lookforitem2="hzjc"

  lookforwhat2="'"+sle\_2.text+"'"

end if

//查询段3和查询内容3的选择

if ddlb\_3.text="学号" then

  lookforitem3="xh"

  lookforwhat3="'"+trim(sle\_3.text)+"'"

else if ddlb\_3.text="姓名" then

  lookforitem3="xm"

  lookforwhat3="'"+trim(sle\_3.text)+"'"

else if ddlb\_3.text="性别" then

  lookforitem3="xb"

  if trim(sle\_3.text) = '男' then

           ls\_temp = '1'

  else

           ls\_temp = '2'

  end if

  lookforwhat3="'"+ls\_temp+"'"

else if ddlb\_3.text="民族" then

  lookforitem3="mznr"

  lookforwhat3="'"+sle\_3.text+"'"

else if ddlb\_3.text="所在院名" then

  lookforitem3="yjnr"

  lookforwhat3="'"+sle\_3.text+"'"

else if ddlb\_3.text="所在系名" then

  lookforitem3="xbnr"

  lookforwhat3="'"+sle\_3.text+"'"

else if ddlb\_3.text="所读专业" then

  lookforitem3="zynr"

  lookforwhat3="'"+sle\_3.text+"'"

else if ddlb\_3.text="奖惩类别" then

  lookforitem3="jc"

  if trim(sle\_3.text) = '奖励' then

           ls\_temp = '1'

  else

           ls\_temp = '0'

  end if

  lookforwhat3="'"+ls\_temp+"'"

else if ddlb\_3.text="奖惩级别" then

  lookforitem3="jcjbnr"

  lookforwhat3="'"+trim(sle\_3.text)+"'"

else if ddlb\_3.text="奖惩名称" then

  lookforitem3="hzjc"

  lookforwhat3="'"+sle\_3.text+"'"

end if

//查询条件的综合

if ddlb\_1.text="" or ddlb\_4.text="" then

else if ddlb\_2.text="" or ddlb\_5.text="" then

  lookforcondition= lookforitem1 +lookforhow1 +lookforwhat1

else if ddlb\_3.text="" or ddlb\_6.text ="" then

  lookforcondition=lookforitem1 + lookforhow1 + "" + lookforwhat1 + "" + lookforoperator1 + "  " + lookforitem2 + lookforhow2 + "" + lookforwhat2 + ""

else

  lookforcondition=lookforitem1 + lookforhow1 + "" + lookforwhat1 + "" + lookforoperator1 + " " + lookforitem2 + lookforhow2 + "" + lookforwhat2 + "" + lookforoperator2 + " " + lookforitem3 + lookforhow3 + "" + lookforwhat3 + ""

end if

connect using sqlca;

dw\_1.settransobject(sqlca)

//messagebox('',lookforcondition)

if dw\_1.setfilter(lookforcondition)=-1 then return

dw\_1.filter()

dw\_1.retrieve()

以上即为学生管理系统的主要代码分析，其他代码比较简单，这里不再赘述。总的来说，小型的学生管理信息系统比较简单，但是，用户如果将学生管理信息系统和其他系统联合在一起，就比较复杂了。这些要根据客户的需求而定，请读者试着按照本书介绍的系统开发方法自行解决。

实例演示

   学生管理系统的使用比较简单，首先打开学生管理系统的主窗口，然后选择需要操作的功能，例如要录入学生基本信息，首先要打开学生基本信息录入的主窗口。如图9-36所示。



图9-36  学生基本信息录入主窗口

选择“文件”|“录入”|“录入个人信息”命令后，打开“录入学生信息”窗口。如图9-37所示。



图9-37  学生基本信息录入窗口

    在这个窗口中录入学生基本信息后单击“保存”按钮即可完成学生基本信息的录入。如果要录入学生的社会关系信息，则单击“社会关系”标签页。如图9-38所示为学生社会关系的录入。



图9-38  学生社会关系录入窗口

如果要录入学生的简历信息，则单击“个人简历”标签页。如图9-39所示为个人简历的录入。



图9-39  学生个人简历录入窗口

单击“保存”按钮即可完成录入。

其他功能的操作与此类似(略)。

**知识小结**

**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** |
|  |  |  |

# 开发工具

## 应用EDITPLUS编辑COBOL程序

## 应用UESTUDIO编辑COBOL程序

## NET COBOL开发环境配置

## COBOL85单机软件开发环境配置概述

以下环境配置为日文操作系统，中文、英文可参考进行，部分配置为进行PL/COBOL数据库操作编程使用的，做简单程序并非必须配置，这里先介绍最常用的配置，其它配置在讲解到相关语句时候再进行详细介绍。

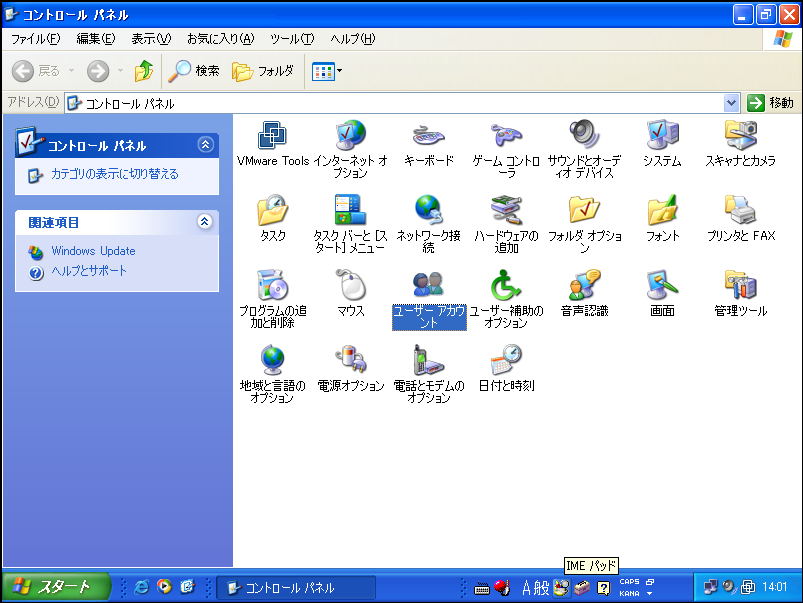
Windows环境设置

本机用户配置（如不联网操作可略过）

配置本机用户名：用户名必须是英文字母，不能有汉字

操作方法：

选中“控制面板”的“管理工具”，再选中“计算机管理”



选中“本地用户和组”的用户(ユーザー)，然后选择“建立新用户”



输入想要新建的用户名和密码，用户名应唯一。

然后设置系统角色、权限、密码等（图片略）。

本机网络环境配置

配置本机IP地址：(通过控制面板的网络连接配置，举例如下）

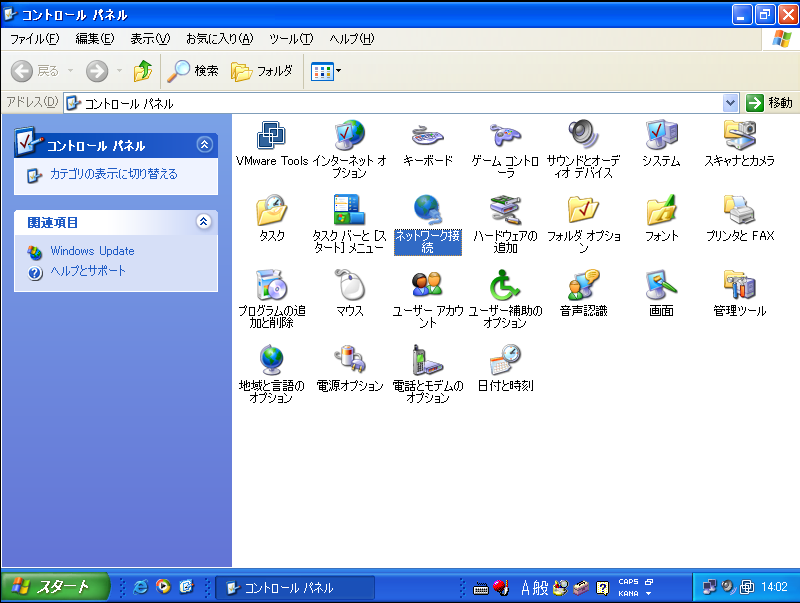
IP地址：192.168.0.XXX

掩码：255.255.255.0

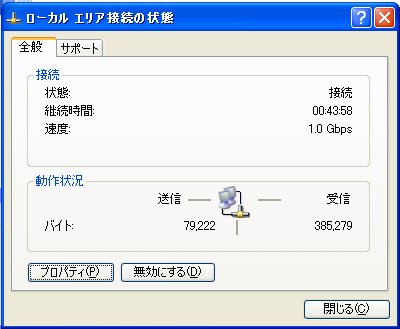
网关：192.168.0.1

DNS：202.96.64.68

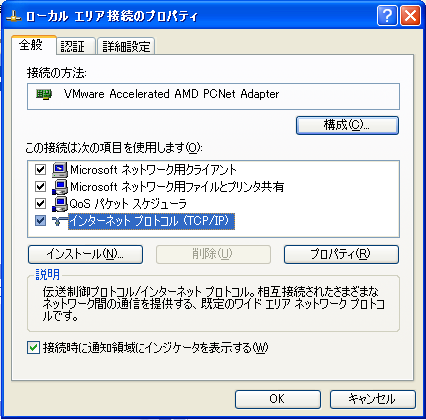
从“开始”、“控制面板”，选择“网络和拨号连接”，打开“本地连接”对话框。



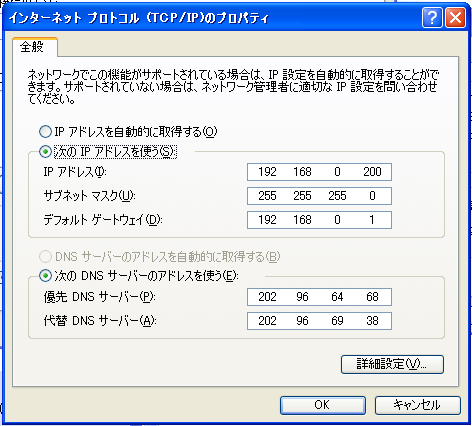
打开本地连接对话框（图片是虚拟机windowsxp日文版的网卡），然后再按“属性按钮”



选中“TCP/IP”，然后点击“属性”按钮



根据网络管理员提供的配置信息设置本机的IP地址。



用Excel比较数据

DB名（D33BUK\_TBL）

更新前

index2、运行程序后，把更新后的数据从数据库中拷贝进Excel文件。

更新後

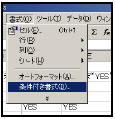
index3、编辑公式，相同的项目显示“YES”，不同的项目显示“YES”，并且标为红色。

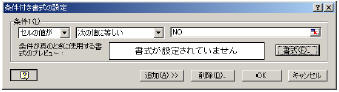
index

4、公式说明。

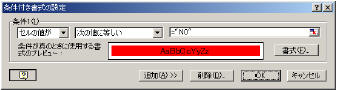
|  |
| --- |
| 在单元各种输入“=IF(B6=B13,"YES","NO")”，其中“B6、B13”表示单元格的位置，这个公式表示如果两个单元格相等则显示"YES"，否则显示"NO"。 |

5、把显示“NO”的单元格显示为红色的方法。

按照左面的图选择。



然后选择红色属性



文本编辑时常用的快捷键

当我们编写COBOL程序的时候，如果能掌握一些快捷键，则能够提高我们的编写源代码的速度，提高工作效率。下面是我在以前的工作中总结的一些快捷的使用方法，供大家参考。

1. Home：移动光标到当前行的行头。

2. End：移动光标到当前行的末尾。

3. Insert：“改写模式”和“插入模式”之间切换，在“改写模式”敲入的字符直接替换当前字符，而“插入模式”敲入的字符插入在当前字符的前面。两种模式的判断可以看光标的形状。

4. Delete：删除当前的字符。

5. Page Up：向上滚动一屏。

6. Page Down：向下滚动一屏。

7. Ctrl + Home：移动光标到第一行的行头。

8. Ctrl + End：移动光标到最后一行的行尾。

9. Ctrl + →：移动光标到右面的下一个单词。

10. Ctrl + ←：移动光标到左面的上一个单词。

11. Ctrl + ↑：向上滚动某些行（具体行数可以在编辑环境中指定）。

12. Ctrl + ↓：向下滚动某些行（具体行数可以在编辑环境中指定）。

13. Shift + →：向右选择一个字符。

14. Shift + ←：向左选择一个字符。

15. Shift + ↑：选择从当前光标位置开始到上一行同一光标位置的一行的内容。

16. Shift + ↓：选择从当前光标位置开始到下一行同一光标位置的一行的内容。（即拷贝正行的内容，拷贝后，可以在要插入行的行头粘贴选择的行进行插入）

17. Shift + Ctrl + →：选择到下一个单词位置的字符串内容。

18. Shift + Ctrl + ←：选择到上一个单词位置的字符串内容。

19. Shift + Ctrl + ↑：选择从当前光标位置开始到上面某一行同一光标位置的某些行的内容（具体行数可以在编辑环境中指定）。

20. Shift + Ctrl + ↓：选择从当前光标位置开始到下面某一行同一光标位置的某些行的内容（具体行数可以在编辑环境中指定）。

21. Shift + Ctrl + Home：选择从当前光标位置开始到第一行的行头的所有内容。

22. Shift + Ctrl + End：选择从当前光标位置开始到最后一行的行尾的所有内容。

23. Ctrl +C：复制被选择的内容。

24. Ctrl +T：剪切被选择的内容。

25. Ctrl +V：粘贴被复制的内容。

26. Shift + F2：查找字符串。查找到第一个字符串后，可以关闭“查找”对话框，按F2 键查找下一个字符串。

27. Shift + F3：查找并且替换字符串。查找到第一个字符串后，可以关闭“替换”对话框，按F2 键查找下一个字符串，按F3 键替换下一个字符串。

28. F4：跳转到指定行。

29. 改变选择模式：选择“編集”菜单项，选择“矩形モード”；或者选择“矩形モード”快捷按钮，则在行选择和列选择模式之间进行切换。

30. 重排行号：选择“編集”菜单项，选择“リナンバ”。重排开始行号和间隔行号都是100。

31. 显示全角空格：选择“表示”菜单项，在选择“表示形式”菜单项，在“日本語空白文字”项输入“■”，再选中“次のウィンドウからも有効とする”（即下一次启动时也生效）。

NETCOBOL环境配置文件编辑

提示：在没用到文件、COPY语句、ORACLE数据操作的练习可以不进行事先配置

在项目文件夹下编辑<程序名.CBI>文件

例如：HELLOCOBOL.CBI

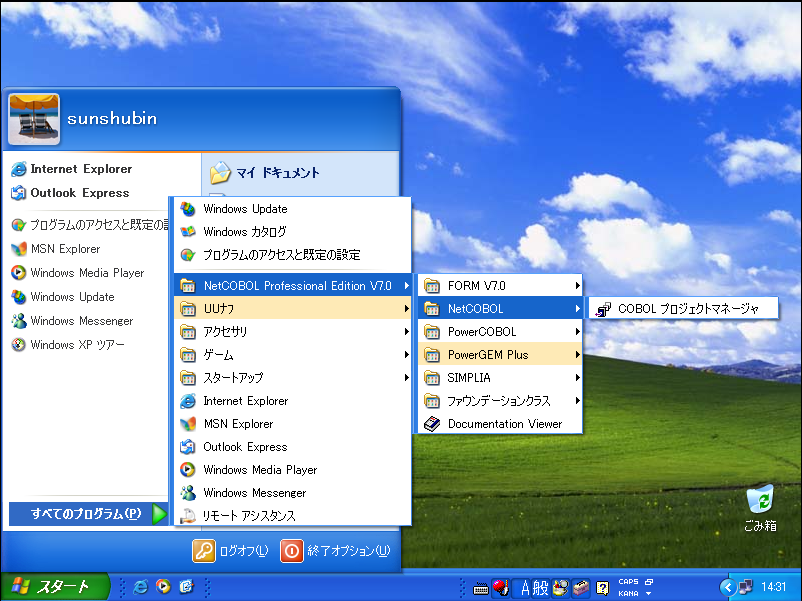
[Compile Options]  
CHECK=Yes 1,ALL  
COPY=Yes  
DLOAD=No  
FLAG=I  
FLAGSW=No STDM  
FORMLIB="Z:\HELLO\OVL"  
GEN=No  
LANGLVL=85  
LIB=Z:\HELLO\COPYLIB  
LINECOUNT=60  
LINESIZE=136  
MAIN=Yes  
MESSAGE=Yes  
NUMBER=No  
OBJECT=Yes  
PRINT=Yes "C:\HELLO\LIST"  
RSV=ALL  
SOURCE=Yes  
SSOUT=LOGFILE  
STD1=JIS1  
TEST=Yes  
TRACE=No  
XREF=No

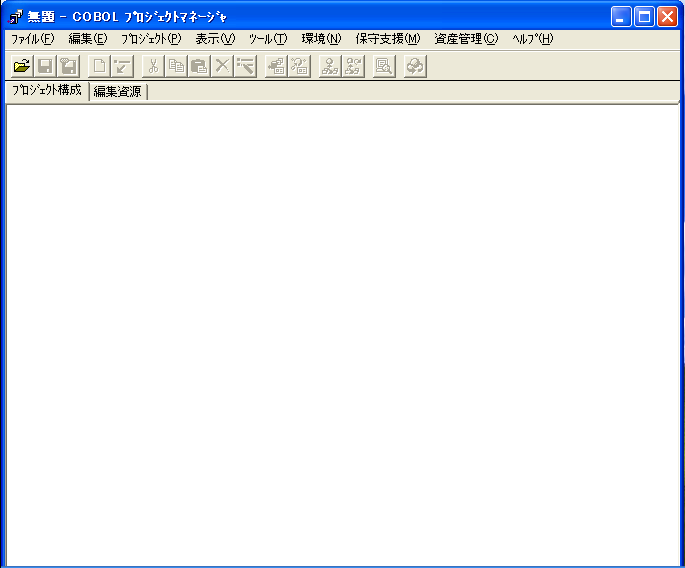
执行参数的设置，在程序文件夹中编辑COBOL85.CBR文件

[HELLOCOBOL.ENTRY]  
CLOCO001=Z:\HELLO\DLL\CLOCO001.DLL  
COBCO001=Z:\HELLO\DLL\COBCO001.DLL  
[HELLOCOBOL]  
@IconName=COB85EXE  
@ScrnSize=(80,24)  
@CnslWinSize=(80,24)  
@CnslBufLine=100  
@WinCloseMsg=ON  
@EnvSetWindow=UNUSE  
@AllFileExclusive=NO  
@NoMessage=YES  
@MessOutFile=C:\HELLO\SOC\HELLOCOBOL\HELLOCOBOL\_SYS.LOG  
DBMSACS=C:\HELLO\SOC\DBMSACS.INI  
U01=C:\HELLO\SOC\HELLOCOBOL\FFUIS47Q.TMP  
U02=C:\HELLO\SOC\HELLOCOBOL\FFUIS47R.TMP  
U11=C:\HELLO\SOC\HELLOCOBOL\FFUIS47T.TMP  
LOGFILE=C:\HELLO\SOC\HELLOCOBOL\HELLOCOBOL.LOG

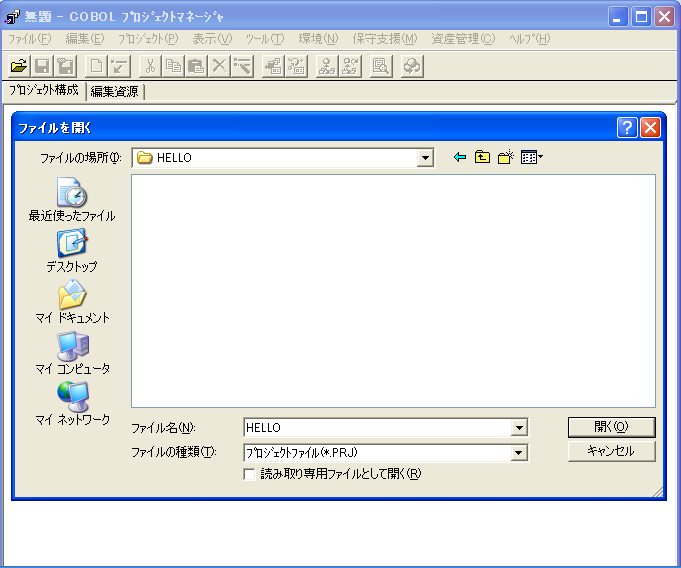
NETCOBOL建立一个项目工程。

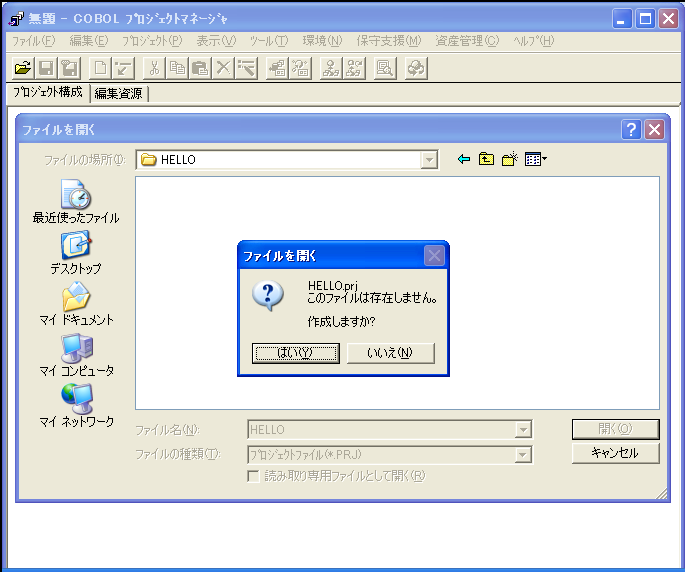
启动NETCOBOL



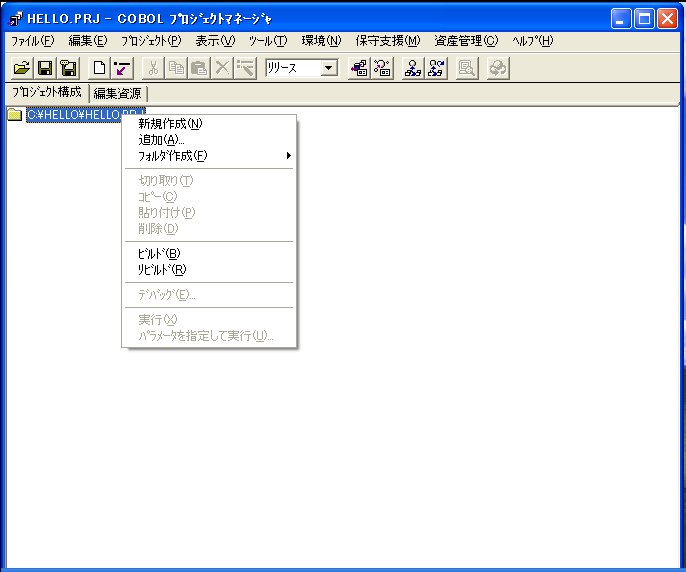


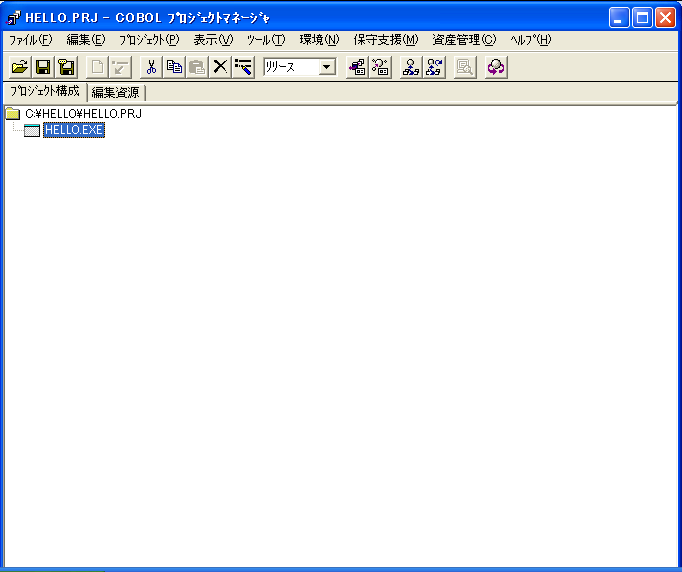
选择打开一个项目文件，这里由于是新建，建立一个项目文件夹，输入项目名称，例如HELLOC，会提示不存在，系统对话框问是否创建，选择创建，建立项目工程。



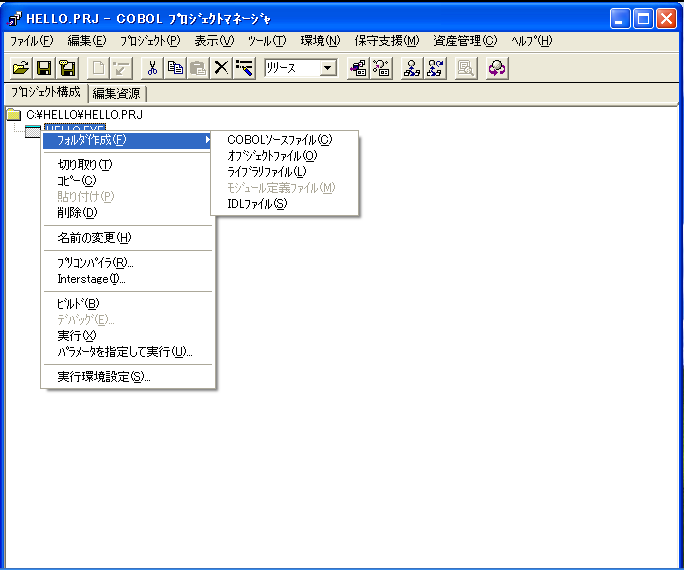


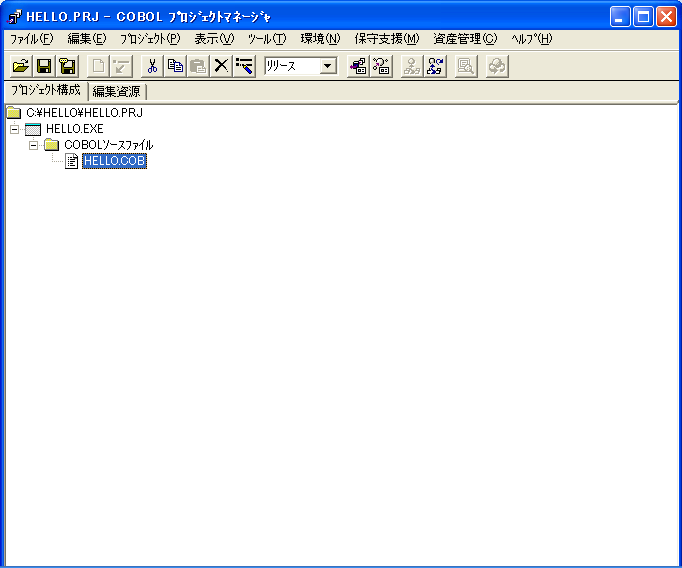
2.在项目上点击右键选择“新规作成”



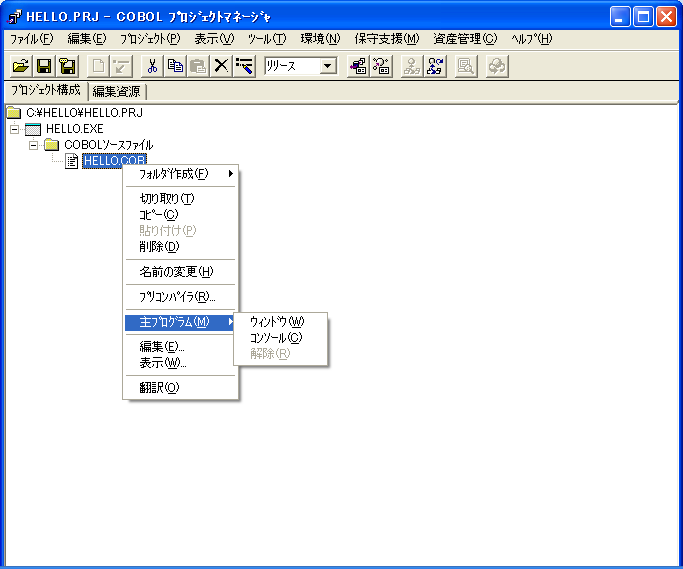
　　3.把文件名改成与项目同名，如HELLOCOBOL.EXE  


　　4.在刚才的文件上点击鼠标右键，选择新建COBOL源程序。

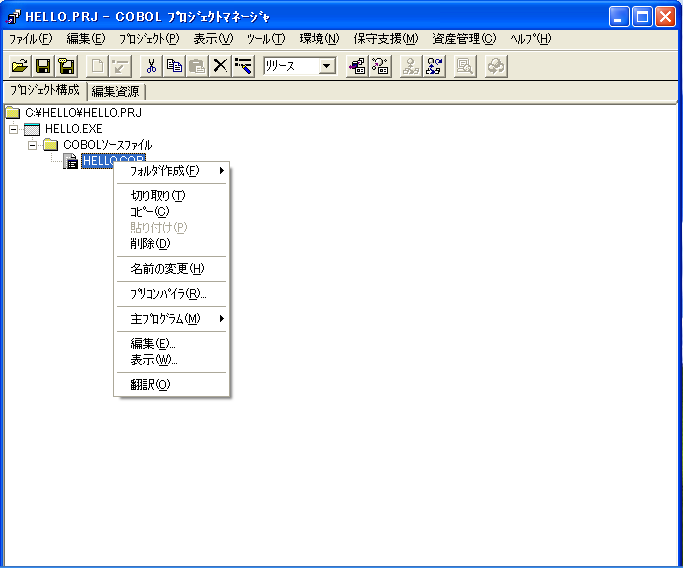
　　5.建立COBOL源程序文件。



6.在源程序上点击鼠标右键，选择这个COB文件为控制台程序主文件，如下图。



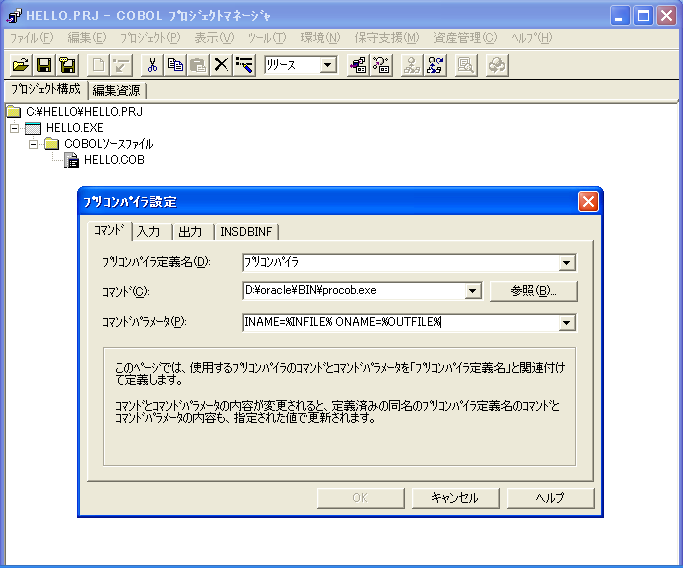
7.如果程序需要访问ORACLE数据库，还需要进行以下操作：  
    (1)建立预编译文件



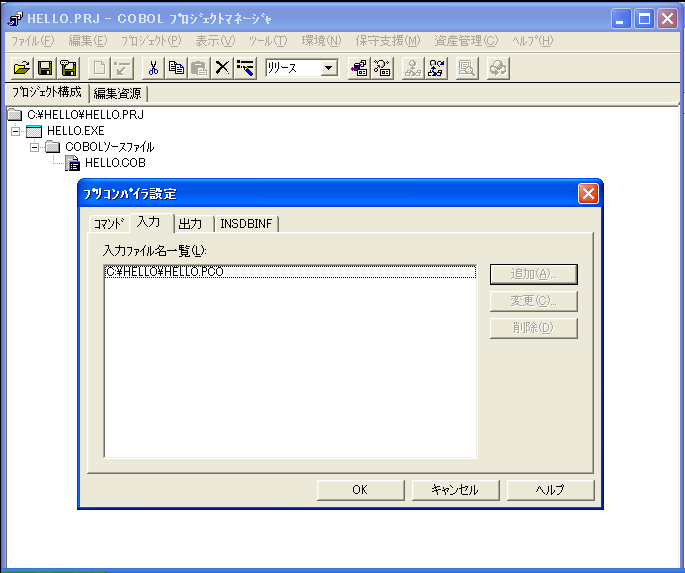
(2)选择预编译程序和参数，如下图所示。

预编译程序：D:\oracle\BIN\procob.exe

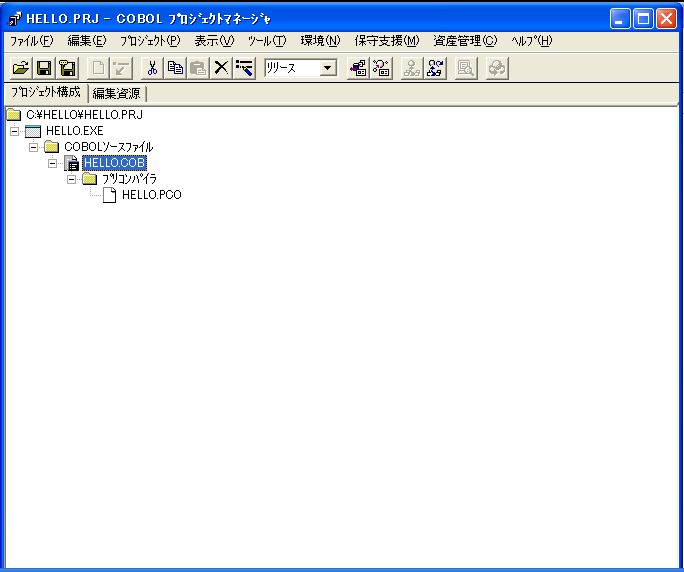
参数：INAME=%INFILE% ONAME=%OUTFILE%



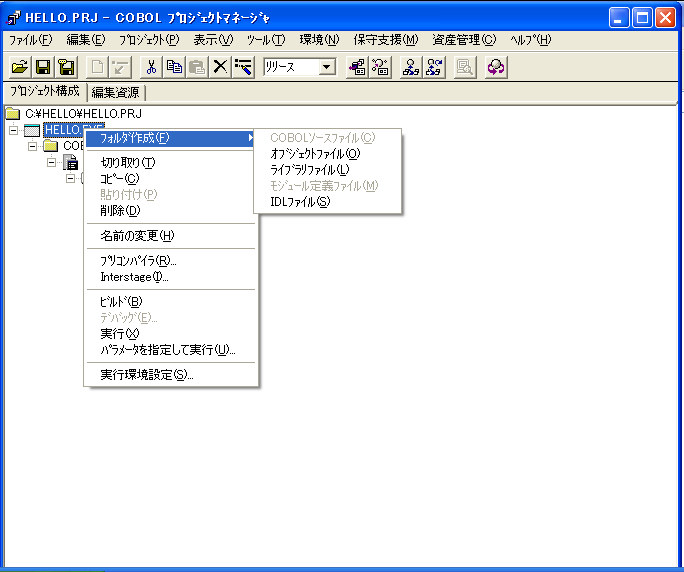
(3)输入预编译文件名



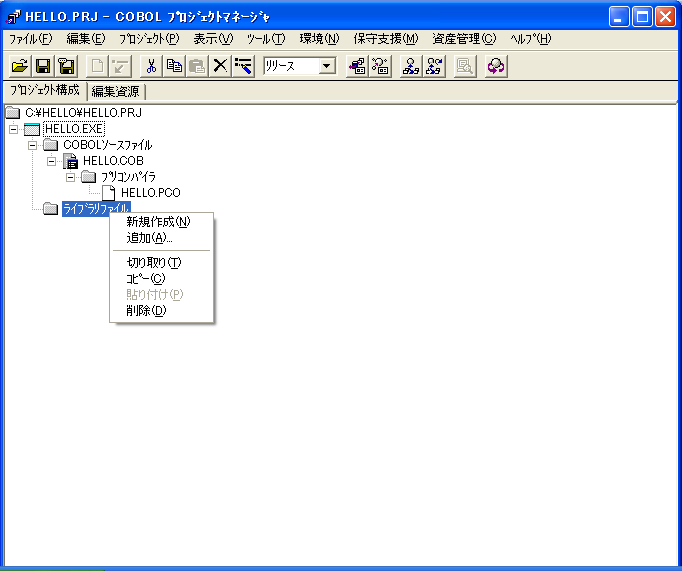
(4)确定后如下图所示：



(5)添加Oracle类库文件。

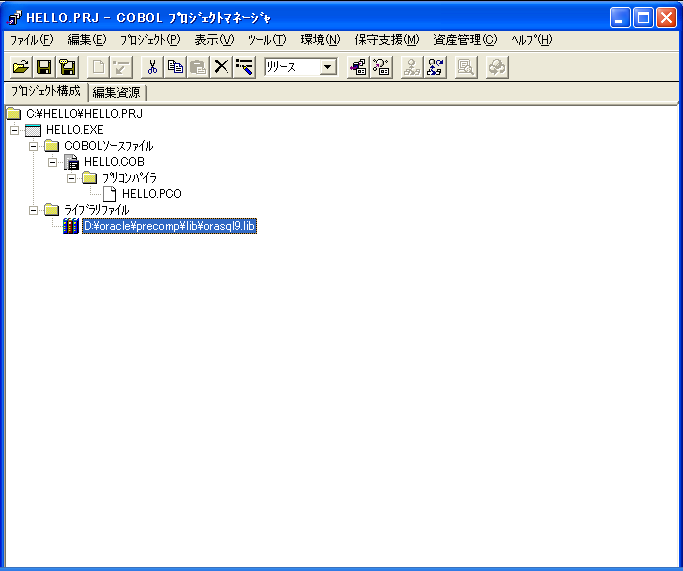


(6)选择具体的类库文件名。



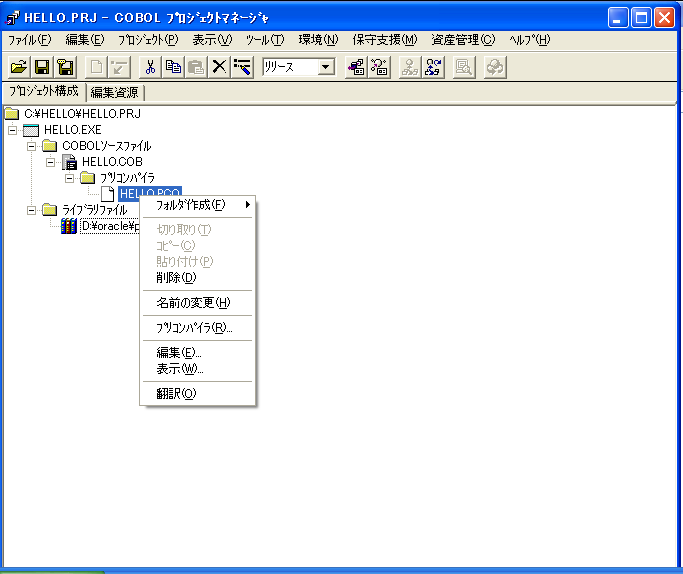
(7)最后如下图所示。

D:\oracle\precomp\lib\orasql9.lib

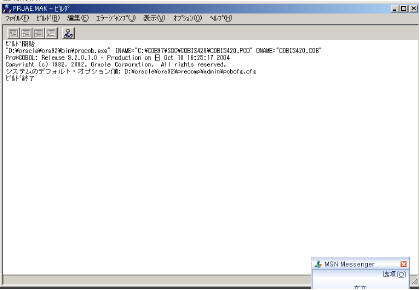


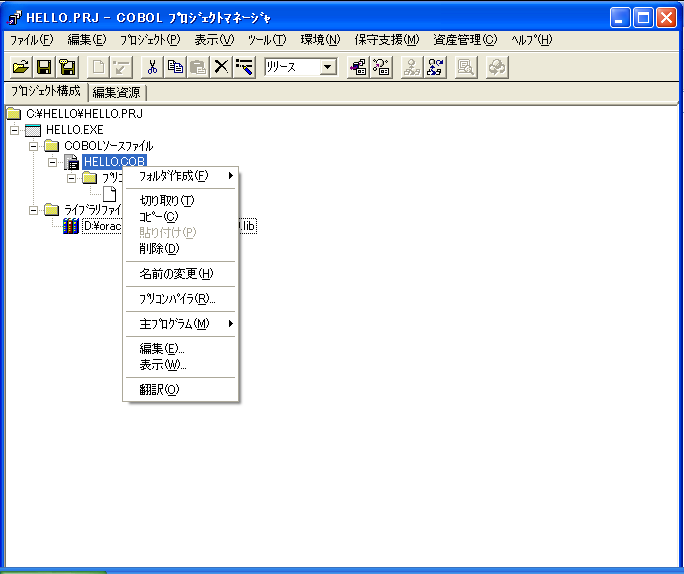
8.编译方法：

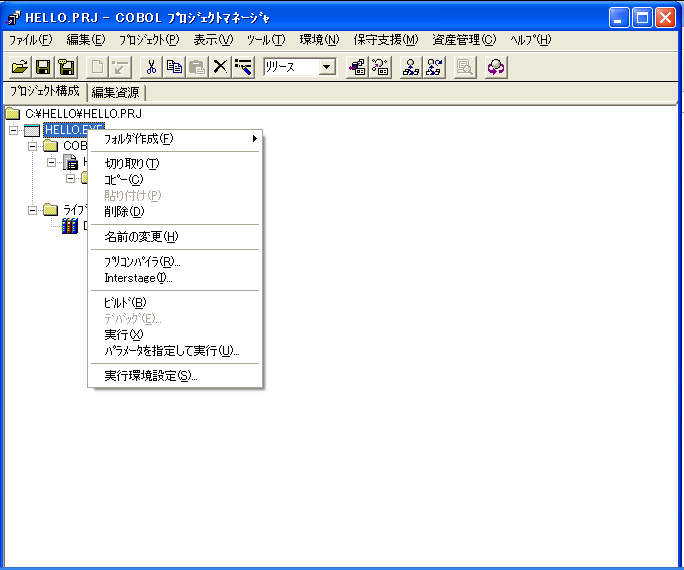
(1)在预编译文件上点击鼠标右键开始编译，如果有错误，会有红色的信息，正确的话，应显示结束。



正确的预编

  
　　(2)在COB文件上点击鼠标右键继续编译



(3)在EXE文件上点击鼠标右键菜单建立可执行文件。  


(4)或者直接选择编译连接按钮。

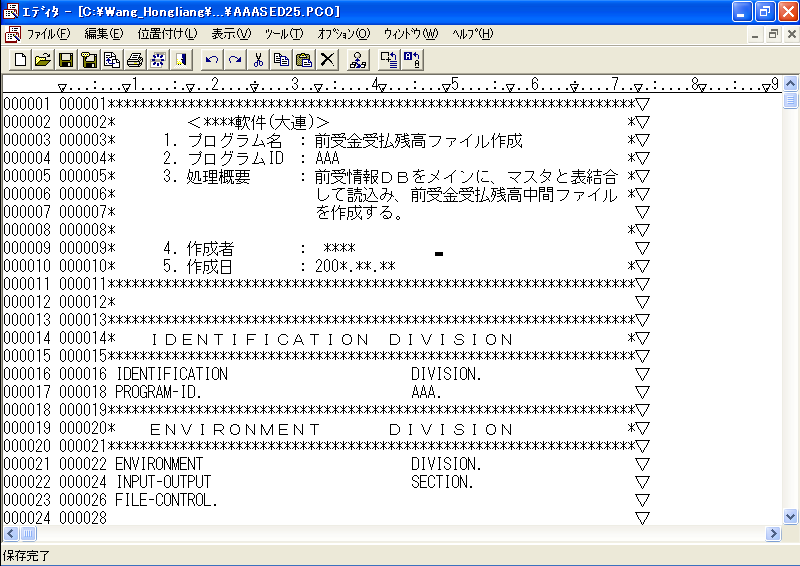
9.执行方法：在EXE文件上直接双击即可；或者在资源管理器中双击运行程序。

PCO格式文件的关联编辑方法。

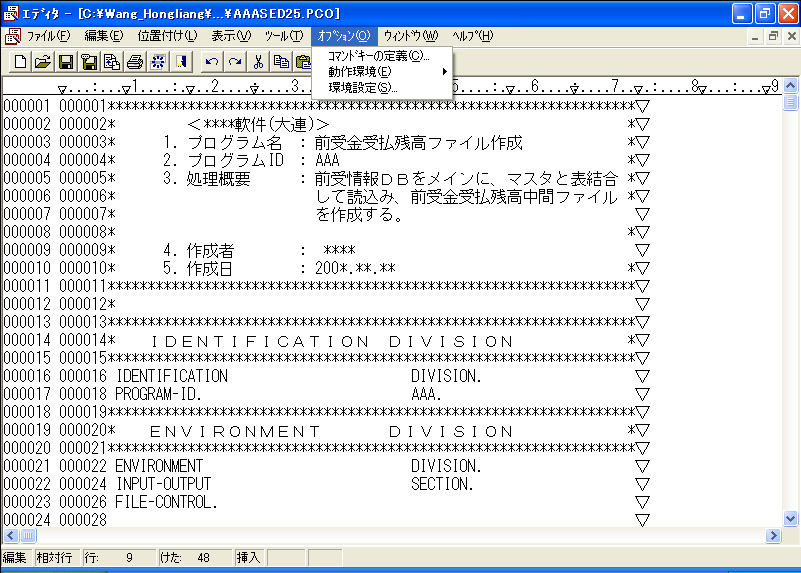
由于PCO格式文件不是默认的COBOL文件，因此关键字没有颜色。因此需要自己设定关联程序：

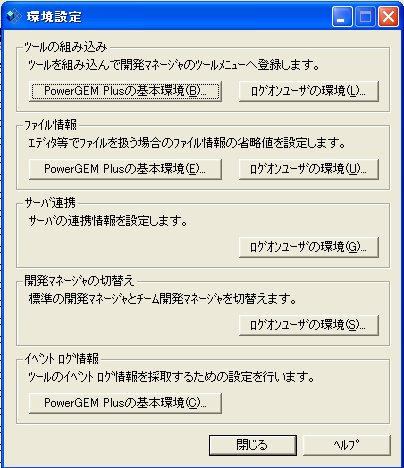
双击PCO文件，然后有对话框提示选择用什么程序打开，选中“PowerGEM Plus Editor”程序，然后打开文件编辑。

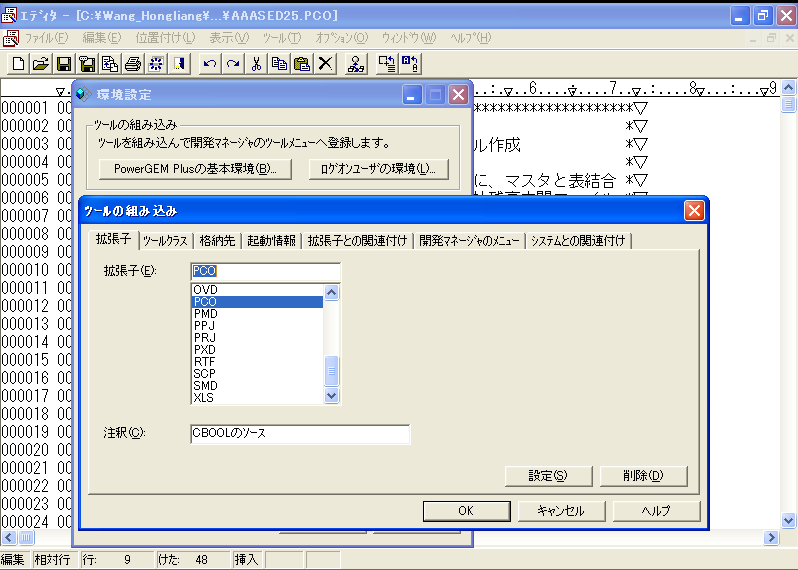
提示：也可以在开发环境中直接鼠标双击PCO文件打开

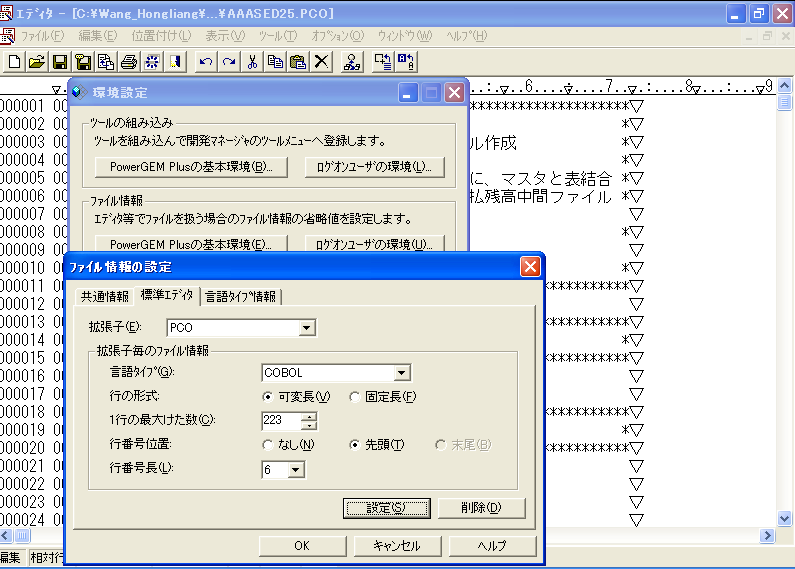


然后再选择“选项”→“环境设定”菜单。





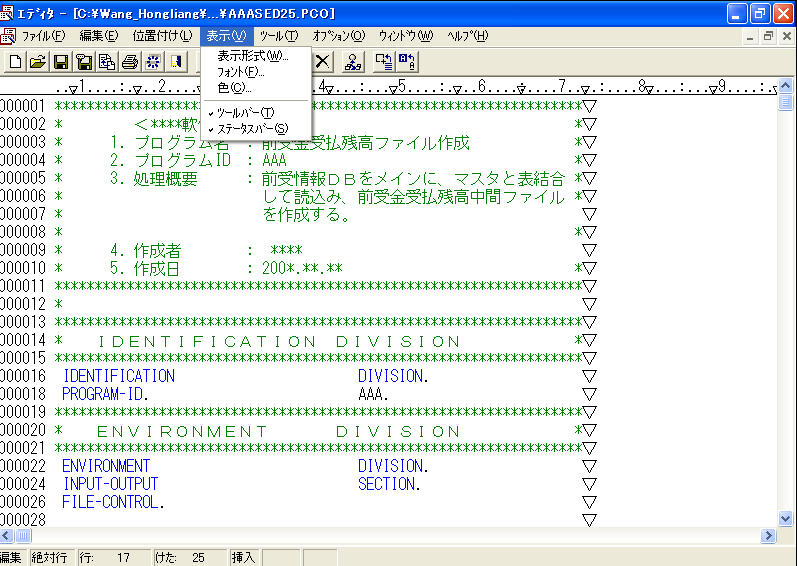
(3)选工具设置，输入下图所示的信息。  


(4)然后选择下面的文件基本环境设定，再输入下图所示信息，“OK”保存。  


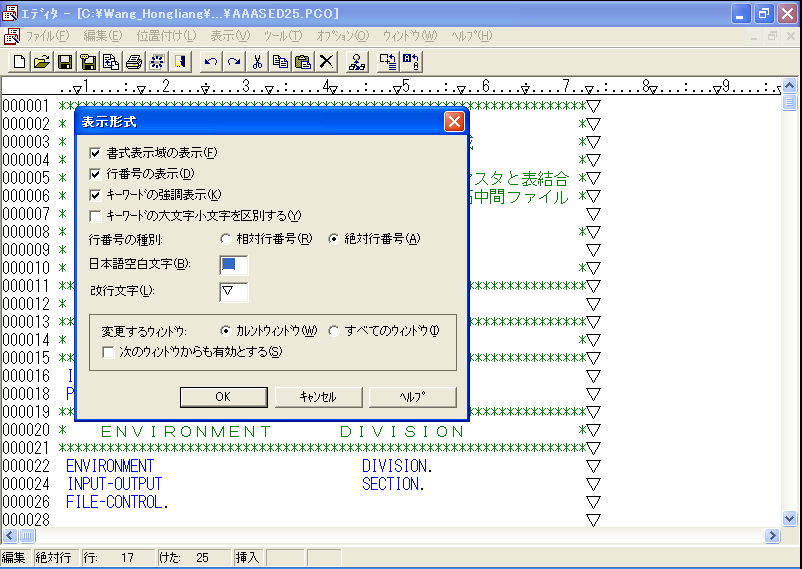
(5) 打开PCO文件后，就可以看到COBOL关键字有颜色了。

输入全角空格时显示的字符

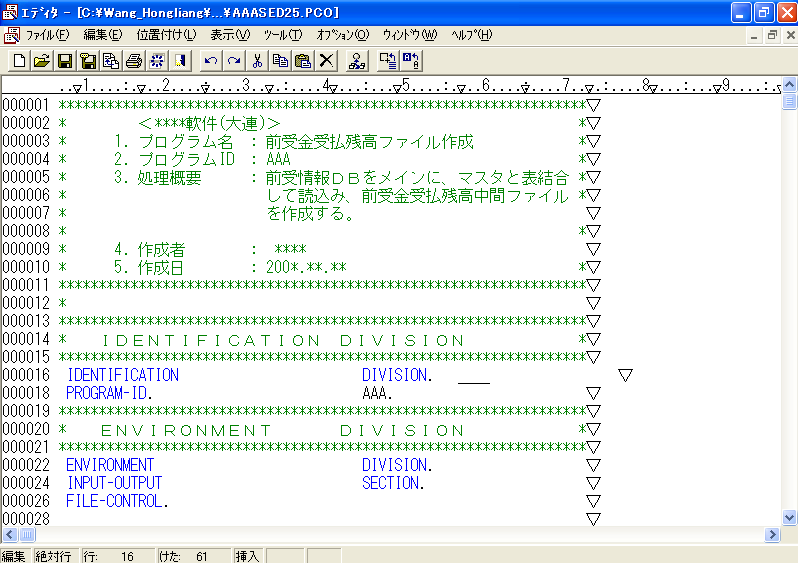
由于编码中有时输入了全角空格而产生编译或者执行错误，用这个字符可以找到全角空格。



可以在那里输入你想让全角空格位置上显示的字符，比如这里输入的是全角下划线



显示效果如下：



## OS 390 编译Cobol程序

# COBOL开发补充

## COBOL85语法汇总

## COBOL编码规约

## COBOL CHECK LIST

**COBOL的网络资源**

**中文**

**英文**

**日文**

# 索引

# 自我测试

**参考文献**

谭浩强 《COBOL程序设计》作者：谭浩强 页数：245  出版日期：1995年7月第1版

《商用COBOL程序设计》对外贸易教育出版社 作者：田文英 页数：275  出版日期：1992年8月第1版

COBOL程序设计　　作者：张令初 页数：295  出版日期：1994年8月第1版

邱仲番 《Cobol85 从入门到精通》【作者】 （美）Carol Baroudi 【译者】 邱仲潘

【出版社】 电子工业出版社

IBM————————

富士通 NetCOBOL for Windows V8.0文法書

**COBOL上机实验**

**实验一：**

内容：熟悉主机系统，建立cobol上机所需数据集，编译运行一个最简单的COBOL程序

要求：

1. 建立相应数据集

（一）建立COBOL源程序PDS，用于存放你的源程序(YOURID. COBOL.SOURCE

参数:

Management class . . .

Storage class . . . .

Volume serial . . . .

Device type . . . . .

Data class . . . . . .

Space units . . . . . TRACK

Average record unit

Primary quantity . . 60

Secondary quantity 20

Directory blocks . . 20

Record format . . . . FB

Record length . . . . 80

Block size . . . . . 8000

Data set name type : PDS

(二)建立存放编译运行JCL程序的PDS: YOURID.COBOL.JCL

参数同上

（三）建立一个存放LOAD的PDS：YOURID.COBOL.LOAD

参数如下：

Management class . . .

Storage class . . . .

Volume serial . . . .

Device type . . . . .

Data class . . . . . .

Space units . . . . . TRACK

Average record unit

Primary quantity . . 80

Secondary quantity 20

Directory blocks . . 20

Record format . . . . U

Record length . . . . 80

Block size . . . . . 3200

Data set name type : PDS

1. 写出一个最简单的COBOL程序，在屏幕上输出诸如“Hello World“的信息
2. 编译运行之。

编译JCL：

//YOURIDXX JOB ACCT#,YOURID,CLASS=A,MSGCLASS=H,

// MSGLEVEL =(1,1),NOTIFY=&SYSUID

//COBCL EXEC IGYWCL

//STEPLIB DD DSN=IGY330.SIGYCOMP,DISP=SHR

//COBOL.SYSIN DD DISP=SHR,DSN=YOURID.COBOL.SOURCE(TEST1)

//COBOL.SYSLIB DD DISP=SHR,DSN= YOURID.COBOL.SOURCE

//LKED.SYSLMOD DD DISP=SHR,DSN= YOURID.COBOL.LOAD(TEST1)

//SYSPRINT DD SYSOUT=\*

运行编译结果的JCL：

// YOURIDXX JOB ACCT#, YOURID,CLASS=A,

// MSGCLASS=H,MSGLEVEL=(1,1),

// NOTIFY=&SYSUID

//RUNCBL EXEC PGM= TEST1

//STEPLIB DD DSN= YOURID.COBOL.LOAD,DISP=SHR

//SYSPRINT DD SYSOUT=\*

//SYSOUT DD SYSOUT=\*

如何在控制台运行程序：

1. ispf首界面选6
2. 运行如下命令：

allocate fi(sysout) fi(sysin) da('\*')

call YOURID.COBOL.LOAD(TEST) (或者直接在YOURID.COBOL.LOAD的TEST前CALL)

**实验二：**

写一个程序，试验不同类型的计算语句和不同数据类型

要求：从控制台读入数据，进行计算，显示出计算结果

同时试验传送语句以及简单选择语句的作用（比如根据不同输入进行不同运算），用程序实现习题4.6和4.8

编译过程同实验一

运行过程采用控制台运行程序方式

**实验三：**

过程部的较高技巧

要求：试验IF与条件名和PERFORM语句

用独立程序完成习题5.8（尝试用if和条件名）

用PERFORM语句完成两个循环，计算1+2+3+…+100=?(in-line 和out-line 各一个)

**实验四：**

文件操作

要求：

一、用程序在磁盘上写一个PS文件，内容不限

二、用程序读入你写的PS程序，然后改变其格式，再写入PS

运行JCL：

// YOURIDXX JOB ACCT#, YOURID,CLASS=A,MSGCLASS=H,

// MSGLEVEL=(1,1),

// NOTIFY=&SYSUID

//RUNCBL EXEC PGM=TEST

//STEPLIB DD DSN= YOURID.COBOL.LOAD,DISP=SHR

//INFILE DD DSN=??????.INPUT,DISP=SHR

//OUTFILE DD DSN=??????,DISP=(NEW,CATLG,DELETE),

// UNIT=3390,SPACE=(CYL,(10,1))

//SYSPRINT DD SYSOUT=\*

//SYSOUT DD SYSOUT=\*

***COBOL & COBOL II***

**Q1) Name the divisions in a COBOL program ?.**

A1) IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, DATA DIVISION, PROCEDURE DIVISION.

**Q2) What are the different data types available in COBOL?**

A2) Alpha-numeric (X), alphabetic (A) and numeric (9).

**Q3) What does the INITIALIZE verb do? - GS**

A3) Alphabetic, Alphanumeric fields & alphanumeric edited items are set to SPACES. Numeric, Numeric edited items set to ZERO. FILLER , OCCURS DEPENDING ON items left untouched.

**Q4) What is 77 level used for ?**

A4) Elementary level item. Cannot be subdivisions of other items (cannot be qualified), nor can they be subdivided themselves.

**Q5) What is 88 level used for ?**

A5) For condition names.

**Q6) What is level 66 used for ?**

A6) For RENAMES clause.

**Q7) What does the IS NUMERIC clause establish ?**

A7) IS NUMERIC can be used on alphanumeric items, signed numeric & packed decimal items and unsigned numeric & packed decimal items. IS NUMERIC returns TRUE if the item only consists of 0-9. However, if the item being tested is a signed item, then it may contain 0-9, + and - .

**Q8) How do you define a table/array in COBOL?**

A8) ARRAYS.

05 ARRAY1 PIC X(9) OCCURS 10 TIMES.

05 ARRAY2 PIC X(6) OCCURS 20 TIMES INDEXED BY WS-INDEX.

**Q9) Can the OCCURS clause be at the 01 level?**

A9) No.

**Q10) What is the difference between index and subscript? - GS**

A10) Subscript refers to the array occurrence while index is the displacement (in no of bytes) from the beginning of the

array. An index can only be modified using PERFORM, SEARCH & SET. Need to have index for a table in order to

use SEARCH, SEARCH ALL.

**Q11) What is the difference between SEARCH and SEARCH ALL? - GS**

A11) SEARCH - is a serial search.

SEARCH ALL - is a binary search & the table must be sorted ( ASCENDING/DESCENDING KEY clause to be used & data loaded in this order) before using SEARCH ALL.

**Q12) What should be the sorting order for SEARCH ALL? - GS**

A12) It can be either ASCENDING or DESCENDING. ASCENDING is default. If you want the search to be done on an

array sorted in descending order, then while defining the array, you should give DESCENDING KEY clause. (You

must load the table in the specified order).

**Q13) What is binary search?**

A13) Search on a sorted array. Compare the item to be searched with the item at the center. If it matches, fine else repeat the process with the left half or the right half depending on where the item lies.

**Q14) My program has an array defined to have 10 items. Due to a bug, I find that even if the program access the**

**11th item in this array, the program does not abend. What is wrong with it?**

A14) Must use compiler option SSRANGE if you want array bounds checking. Default is NOSSRANGE.

**Q15) How do you sort in a COBOL program? Give sort file definition, sort statement syntax and meaning. - GS**

A15) Syntax: SORT file-1 ON ASCENDING/DESCENDING KEY key.... USING file-2 GIVING file-3.

USING can be substituted by INPUT PROCEDURE IS para-1 THRU para-2

GIVING can be substituted by OUTPUT PROCEDURE IS para-1 THRU para-2.

file-1 is the sort (work) file and must be described using SD entry in FILE SECTION.

file-2 is the input file for the SORT and must be described using an FD entry in FILE SECTION and SELECT

clause in FILE CONTROL.

file-3 is the out file from the SORT and must be described using an FD entry in FILE SECTION and SELECT

clause in FILE CONTROL.

file-1, file-2 & file-3 should not be opened explicitly.

INPUT PROCEDURE is executed before the sort and records must be RELEASEd to the sort work file from the input procedure.

OUTPUT PROCEDURE is executed after all records have been sorted. Records from the sort work file must be RETURNed one at a time to the output procedure.

**Q16) How do you define a sort file in JCL that runs the COBOL program?**

A16) Use the SORTWK01, SORTWK02,..... dd names in the step. Number of sort datasets depends on the volume of data

being sorted, but a minimum of 3 is required.

**Q17) What is the difference between performing a SECTION and a PARAGRAPH? - GS**

A17) Performing a SECTION will cause all the paragraphs that are part of the section, to be performed.

Performing a PARAGRAPH will cause only that paragraph to be performed.

**Q18) What is the use of EVALUATE statement? - GS**

A18) Evaluate is like a case statement and can be used to replace nested Ifs. The difference between EVALUATE and

case is that no 'break' is required for EVALUATE i.e. control comes out of the EVALUATE as soon as one match is

made.

**Q19) What are the different forms of EVALUATE statement?**

A19)

EVALUATE EVALUATE SQLCODE ALSO FILE-STATUS

WHEN A=B AND C=D WHEN 100 ALSO '00'

imperative stmt imperative stmt

WHEN (D+X)/Y = 4 WHEN -305 ALSO '32'

imperative stmt imperative stmt

WHEN OTHER WHEN OTHER

imperative stmt imperative stmt

END-EVALUATE END-EVALUATE

EVALUATE SQLCODE ALSO A=B EVALUATE SQLCODE ALSO TRUE

WHEN 100 ALSO TRUE WHEN 100 ALSO A=B

imperative stmt imperative stmt

WHEN -305 ALSO FALSE WHEN -305 ALSO (A/C=4)

imperative stmt imperative stmt

END-EVALUATE END-EVALUATE

**Q20) How do you come out of an EVALUATE statement? - GS**

A20) After the execution of one of the when clauses, the control is automatically passed on to the next sentence after the

EVALUATE statement. There is no need of any extra code.

**Q21) In an EVALUATE statement, can I give a complex condition on a when clause?**

A21) Yes.

**Q22) What is a scope terminator? Give examples.**

A22) Scope terminator is used to mark the end of a verb e.g. EVALUATE, END-EVALUATE; IF, END-IF.

**Q23) How do you do in-line PERFORM? - GS**

A23) PERFORM ... <UNTIL> ...

<sentences>

END-PERFORM

**Q24) When would you use in-line perform?**

A24) When the body of the perform will not be used in other paragraphs. If the body of the perform is a generic type of code

(used from various other places in the program), it would be better to put the code in a separate Para and use

PERFORM Para name rather than in-line perform.

**Q25) What is the difference between CONTINUE & NEXT SENTENCE ?**

A25) They appear to be similar, that is, the control goes to the next sentence in the paragraph. But, Next Sentence would

take the control to the sentence after it finds a full stop (.). Check out by writing the following code example, one if

sentence followed by 3 display statements (sorry they appear one line here because of formatting restrictions) If 1 > 0

then next sentence end if display 'line 1' display 'line 2'. display 'line 3'. \*\*\* Note- there is a dot (.) only at the end of

the last 2 statements, see the effect by replacing Next Sentence with Continue \*\*\*

**Q26) What does EXIT do ?**

A26) Does nothing ! If used, must be the only sentence within a paragraph.

**Q27) Can I redefine an X(100) field with a field of X(200)?**

A27) Yes. Redefines just causes both fields to start at the same location. For example:

01 WS-TOP PIC X(1)

01 WS-TOP-RED REDEFINES WS-TOP PIC X(2).

If you MOVE '12' to WS-TOP-RED,

DISPLAY WS-TOP will show 1 while

DISPLAY WS-TOP-RED will show 12.

**A28) Can I redefine an X(200) field with a field of X(100) ?**

Q31)1 Yes.

**Q31)2 What do you do to resolve SOC-7 error? - GS**

Q31) Basically you need to correcting the offending data. Many times the reason for SOC7 is an un-initialized numeric item.

Examine that possibility first. Many installations provide you a dump for run time abend’s ( it can be generated also

by calling some subroutines or OS services thru assembly language). These dumps provide the offset of the last

instruction at which the abend occurred. Examine the compilation output XREF listing to get the verb and the line

number of the source code at this offset. Then you can look at the source code to find the bug. To get capture the

runtime dumps, you will have to define some datasets (SYSABOUT etc ) in the JCL. If none of these are helpful, use

judgement and DISPLAY to localize the source of error. Some installation might have batch program debugging

tools. Use them.

**Q32) How is sign stored in Packed Decimal fields and Zoned Decimal fields?**

Q32) Packed Decimal fields: Sign is stored as a hex value in the last nibble (4 bits ) of the storage.

Zoned Decimal fields: As a default, sign is over punched with the numeric value stored in the last bite.

**Q33) How is sign stored in a comp-3 field? - GS**

Q33) It is stored in the last nibble. For example if your number is +100, it stores hex 0C in the last byte, hex 1C if

your number is 101, hex 2C if your number is 102, hex 1D if the number is -101, hex 2D if the number is -102 etc...

**Q34) How is sign stored in a COMP field ? - GS**

Q34) In the most significant bit. Bit is ON if -ve, OFF if +ve.

**Q35) What is the difference between COMP & COMP-3 ?**

Q35) COMP is a binary storage format while COMP-3 is packed decimal format.

**Q36) What is COMP-1? COMP-2?**

Q36) COMP-1 - Single precision floating point. Uses 4 bytes.

COMP-2 - Double precision floating point. Uses 8 bytes.

**Q37) How do you define a variable of COMP-1? COMP-2?**

Q37) No picture clause to be given. Example 01 WS-VAR USAGE COMP-1.

**Q38) How many bytes does a S9(7) COMP-3 field occupy ?**

Q38) Will take 4 bytes. Sign is stored as hex value in the last nibble. General formula is INT((n/2) + 1)), where n=7 in this

example.

**Q39) How many bytes does a S9(7) SIGN TRAILING SEPARATE field occupy ?**

Q39) Will occupy 8 bytes (one extra byte for sign).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | | * [首页](http://www.chinamvs.net/blog/user1/369/index.html) [相册](http://www.chinamvs.net/blog/user1/369/cmd.html?uid=369&do=album) [标签](http://www.chinamvs.net/blog/user1/369/cmd.html?uid=369&do=tags) * [JCL(4)](http://www.chinamvs.net/blog/user1/369/cmd.html?do=blogs&id=15&uid=369) * [COBOL(9)](http://www.chinamvs.net/blog/user1/369/cmd.html?do=blogs&id=16&uid=369) * [DB2(7)](http://www.chinamvs.net/blog/user1/369/cmd.html?do=blogs&id=18&uid=369) * [CICS(0)](http://www.chinamvs.net/blog/user1/369/cmd.html?do=blogs&id=17&uid=369) * [其他(9)](http://www.chinamvs.net/blog/user1/369/cmd.html?do=blogs&id=19&uid=369) | | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | cobol 经典面试集(续) | | By  aireny 发表于 2006-6-19 14:21:45 | | **Q40) How many bytes will a S9(8) COMP field occupy ?**  Q40) 4 bytes.  **Q41) What is the maximum value that can be stored in S9(8) COMP?**  Q41) 99999999  **Q42) What is COMP SYNC?**  Q42) Causes the item to be aligned on natural boundaries. Can be SYNCHRONIZED LEFT or RIGHT. For binary data  items, the address resolution is faster if they are located at word boundaries in the memory. For example, on main  frame the memory word size is 4 bytes. This means that each word will start from an address divisible by 4. If my  first variable is x(3) and next one is s9(4) comp, then if you do not specify the SYNC clause, S9(4) COMP will start  from byte 3 ( assuming that it starts from 0 ). If you specify SYNC, then the binary data item will start from address 4.  You might see some wastage of memory, but the access to this computational field is faster.  **Q43) What is the maximum size of a 01 level item in COBOL I? in COBOL II?**  Q43) In COBOL II: 16777215  **Q44) How do you reference the following file formats from COBOL programs:**  Q44)  Fixed Block File - Use ORGANISATION IS SEQUENTIAL. Use RECORDING MODE IS F,  BLOCK CONTAINS 0 .  Fixed Unblocked - Use ORGANISATION IS SEQUENTIAL. Use RECORDING MODE IS F,  do not use BLOCK CONTAINS  Variable Block File - Use ORGANISATION IS SEQUENTIAL. Use RECORDING MODE IS V, BLOCK  CONTAINS 0. Do not code the 4 bytes for record length in FD ie JCL rec length will be max rec length in pgm + 4  Variable Unblocked - Use ORGANISATION IS SEQUENTIAL. Use RECORDING MODE IS V, do not use  BLOCK CONTAINS. Do not code 4 bytes for record length in FD ie JCL rec length will  be max rec length in pgm + 4.  ESDS VSAM file - Use ORGANISATION IS SEQUENTIAL.  KSDS VSAM file - Use ORGANISATION IS INDEXED, RECORD KEY IS, ALTERNATE RECORD KEY IS RRDS File - Use ORGANISATION IS RELATIVE, RELATIVE KEY IS  Printer File - Use ORGANISATION IS SEQUENTIAL. Use RECORDING MODE IS F, BLOCK  CONTAINS 0. (Use RECFM=FBA in JCL DCB).  **Q45) What are different file OPEN modes available in COBOL?**  Q45) Open for INPUT, OUTPUT, I-O, EXTEND.  **Q46) What is the mode in which you will OPEN a file for writing? - GS**  Q46) OUTPUT, EXTEND  **Q47) In the JCL, how do you define the files referred to in a subroutine ?**  Q47) Supply the DD cards just as you would for files referred to in the main program.  **Q48) Can you REWRITE a record in an ESDS file? Can you DELETE a record from it?**  Q48) Can rewrite (record length must be same), but not delete.  **Q49) What is file status 92? - GS**  Q49) Logic error. e.g., a file is opened for input and an attempt is made to write to it.  **Q50) What is file status 39 ?**  Q50) Mismatch in LRECL or BLOCKSIZE or RECFM between your COBOL pgm & the JCL (or the dataset label). You  will get file status 39 on an OPEN.  **Q51) What is Static and Dynamic linking ?**  Q51) In static linking, the called subroutine is link-edited into the calling program , while in dynamic linking, the subroutine & the main program will exist as separate load modules. You choose static/dynamic linking by choosing either the DYNAM or NODYNAM link edit option. (Even if you choose NODYNAM, a CALL identifier (as opposed to a CALL literal), will translate to a DYNAMIC call).A statically called subroutine will not be in its initial state the next time it is called unless you explicitly use INITIAL or you do a CANCEL. A dynamically called routine will always be in its initial state.  **Q52) What is AMODE(24), AMODE(31), RMODE(24) and RMODE(ANY)? (applicable to only MVS/ESA**  **Enterprise Server).**  Q52) These are compile/link edit options. Basically AMODE stands for Addressing mode and RMODE for Residency  mode.  AMODE(24) - 24 bit addressing;  AMODE(31) - 31 bit addressing  AMODE(ANY) - Either 24 bit or 31 bit addressing depending on RMODE.  RMODE(24) - Resides in virtual storage below 16 Meg line. Use this for 31 bit programs that call 24 bit programs.  (OS/VS Cobol pgms use 24 bit addresses only).  RMODE(ANY) - Can reside above or below 16 Meg line.  **Q53) What compiler option would you use for dynamic linking?**  Q53) DYNAM.  **Q54) What is SSRANGE, NOSSRANGE ?**  Q54) These are compiler options with respect to subscript out of range checking. NOSSRANGE is the default and if chosen,  no run time error will be flagged if your index or subscript goes out of the permissible range.  **Q55) How do you set a return code to the JCL from a COBOL program?**  Q55) Move a value to RETURN-CODE register. RETURN-CODE should not be declared in your program.  **Q56) How can you submit a job from COBOL programs?**  Q56) Write JCL cards to a dataset with //xxxxxxx SYSOUT= (A,INTRDR) where 'A' is output class, and dataset should be  opened for output in the program. Define a 80 byte record layout for the file.  **Q57) What are the differences between OS VS COBOL and VS COBOL II?**  Q57) OS/VS Cobol pgms can only run in 24 bit addressing mode, VS Cobol II pgms can run either in 24 bit or 31 bit  addressing modes.  I. Report writer is supported only in OS/VS Cobol.  II. USAGE IS POINTER is supported only in VS COBOL II.  III. Reference modification e.g.: WS-VAR(1:2) is supported only in VS COBOL II.  IV. EVALUATE is supported only in VS COBOL II.  V. Scope terminators are supported only in VS COBOL II.  VI. OS/VS Cobol follows ANSI 74 stds while VS COBOL II follows ANSI 85 stds.  VII. Under CICS Calls between VS COBOL II programs are supported.  **Q58) What are the steps you go through while creating a COBOL program executable?**  Q58) DB2 precompiler (if embedded SQL used), CICS translator (if CICS pgm), Cobol compiler, Link editor. If DB2  program, create plan by binding the DBRMs.  **Q59) Can you call an OS VS COBOL pgm from a VS COBOL II pgm ?**  Q59) In non-CICS environment, it is possible. In CICS, this is not possible.  **Q60) What are the differences between COBOL and COBOL II?**  A60) There are at least five differences:  COBOL II supports structured programming by using in line Performs and explicit scope terminators, It introduces  new features (EVALUATE, SET. TO TRUE, CALL. BY CONTEXT, etc) It permits programs to be loaded and  addressed above the 16-megabyte line It does not support many old features (READY TRACE, REPORT-WRITER,  ISAM, Etc.), and It offers enhanced CICS support.  **Q61) What is an explicit scope terminator?**  A61) A scope terminator brackets its preceding verb, e.g. IF .. END-IF, so that all statements between the verb and its scope terminator are grouped together. Other common COBOL II verbs are READ, PERFORM, EVALUATE, SEARCH and STRING.  **Q62) What is an in line PERFORM? When would you use it? Anything else to say about it?**  A62) The PERFORM and END-PERFORM statements bracket all COBOL II statements between them. The COBOL equivalent is to PERFORM or PERFORM THRU a paragraph. In line PERFORMs work as long as there are no internal GO TOs, not even to an exit. The in line PERFORM for readability should not exceed a page length - often it will reference other PERFORM paragraphs.  **Q63) What is the difference between NEXT SENTENCE and CONTINUE?**  A63) NEXT SENTENCE gives control to the verb following the next period. CONTINUE gives control to the next verb after the explicit scope terminator. (This is not one of COBOL II's finer implementations). It's safest to use CONTINUE rather than NEXT SENTENCE in COBOL II.  **Q64) What COBOL construct is the COBOL II EVALUATE meant to replace?**  A64) EVALUATE can be used in place of the nested IF THEN ELSE statements.  **Q65) What is the significance of 'above the line' and 'below the line'?**  A65) Before IBM introduced MVS/XA architecture in the 1980's a program's virtual storage was limited to 16 megs. Programs compiled with a 24 bit mode can only address 16 Mb of space, as though they were kept under an imaginary storage line. With COBOL II a program compiled with a 31 bit mode can be 'above the 16 Mb line. (This 'below the line', 'above the line' imagery confuses most mainframe programmers, who tend to be a literal minded group.)  **Q66) What was removed from COBOL in the COBOL II implementation?**  A66) Partial list: REMARKS, NOMINAL KEY, PAGE-COUNTER, CURRENT-DAY, TIME-OF-DAY, STATE, FLOW, COUNT, EXAMINE, EXHIBIT, READY TRACE and RESET TRACE.  **Q67) Explain call by context by comparing it to other calls.**  A67) The parameters passed in a call by context are protected from modification by the called program. In a normal call they are able to be modified.  **Q68) What is the linkage section?**  A68) The linkage section is part of a called program that 'links' or maps to data items in the calling program's working storage. It is the part of the called program where these share items are defined.  **Q69) What is the difference between a subscript and an index in a table definition?**  A69) A subscript is a working storage data definition item, typically a PIC (999) where a value must be moved to the subscript and then incremented or decrements by ADD TO and SUBTRACT FROM statements. An index is a register item that exists outside the program's working storage. You SET an index to a value and SET it UP BY value and DOWN BY value.  **Q70) If you were passing a table via linkage, which is preferable - a subscript or an index?**  A70) Wake up - you haven't been paying attention! It's not possible to pass an index via linkage. The index is not part of the calling programs working storage. Those of us who've made this mistake, appreciate the lesson more than others.  **Q71) Explain the difference between an internal and an external sort, the pros and cons, internal sort syntax etc.**  A71) An external sort is not COBOL; it is performed through JCL and PGM=SORT. It is understandable without any code reference. An internal sort can use two different syntax’s: 1.) USING, GIVING sorts are comparable to external sorts with no extra file processing; 2) INPUT PROCEDURE, OUTPUT PROCEDURE sorts allow for data manipulation before and/or after the sort.  **Q72) What is the difference between comp and comp-3 usage? Explain other COBOL usage’s.**  A72) Comp is a binary usage, while comp-3 indicates packed decimal. The other common usage’s are binary and display. Display is the default.  **Q73) When is a scope terminator mandatory?**  A73) Scope terminators are mandatory for in-line PERFORMS and EVALUATE statements. For readability, it's recommended coding practice to always make scope terminators explicit.  **Q74) In a COBOL II PERFORM statement, when is the conditional tested, before or after the perform execution?**  A74) In COBOL II the optional clause WITH TEST BEFORE or WITH TEST AFTER can be added to all perform statements. By default the test is performed before the perform. | | | | | |

**Q76) What is the default value(s) for an INITIALIZE and what keyword allows for an override of the default.**

A76) INITIALIZE moves spaces to alphabetic fields and zeros to alphanumeric fields. The REPLACING option can be used to override these defaults.

**Q77) What is SET TO TRUE all about, anyway?**

A77) In COBOL II the 88 levels can be set rather than moving their associated values to the related data item. (Web note: This change is not one of COBOL II's better specifications.)

**Q78) What is LENGTH in COBOL II?**

A78) LENGTH acts like a special register to tell the length of a group or elementary item.

**Q79) What is the difference between a binary search and a sequential search? What are the pertinent COBOL**

**commands?**

A79) In a binary search the table element key values must be in ascending or descending sequence. The table is 'halved' to search for equal to, greater than or less than conditions until the element is found. In a sequential search the table is searched from top to bottom, so (ironically) the elements do not have to be in a specific sequence. The binary search is much faster for larger tables, while sequential works well with smaller ones. SEARCH ALL is used for binary searches; SEARCH for sequential.

**Q80) What is the point of the REPLACING option of a copy statement?**

A80) REPLACING allows for the same copy to be used more than once in the same code by changing the replace value.

**Q81) What will happen if you code GO BACK instead of STOP RUN in a stand alone COBOL program i.e. a**

**program which is not calling any other program.**

A81) The program will go in an infinite loop.

**Q82) How can I tell if a module is being called DYNAMICALLY or STATICALLY?**

A82) The ONLY way is to look at the output of the linkage editor (IEWL)or the load module itself. If the module is being called DYNAMICALLY then it will not exist in the main module, if it is being called STATICALLY then it will be seen in the load module. Calling a working storage variable, containing a program name, does not make a DYNAMIC call. This type of calling is known as IMPLICITE calling as the name of the module is implied by the contents of the working storage variable. Calling a program name literal (CALL

**Q83) What is the difference between a DYNAMIC and STATIC call in COBOL.**

A83) To correct an earlier answer: All called modules cannot run standalone if they require program variables passed to them via the LINKAGE section. DYNAMICally called modules are those that are not bound with the calling program at link edit time (IEWL for IBM) and so are loaded from the program library (joblib or steplib) associated with the job. For DYNAMIC calling of a module the DYNAM compiler option must be chosen, else the linkage editor will not generate an executable as it will expect u address resolution of all called modules. A STATICally called module is one that is bound with the calling module at link edit, and therefore becomes part of the executable load module.

**Q84) How may divisions are there in JCL-COBOL?**

A84) Four

**Q85) What is the purpose of Identification Division?**

A85) Documentation.

**Q86) What is the difference between PIC 9.99 and 9v99?**

A86) PIC 9.99 is a FOUR-POSITION field that actually contains a decimal point where as PIC 9v99 is THREE- POSITION numeric field with implied or assumed decimal position.

**Q87) what is Pic 9v99 Indicates?**

A87) PICTURE 9v99 is a three position Numeric field with an implied or assumed decimal point after the first position; the v means an implied decimal point.

**Q88) What guidelines should be followed to write a structured Cobol prg'm?**

A88)

1) use 'evaluate' stmt for constructing cases.

2) use scope terminators for nesting.

3) use in line perform stmt for writing 'do ' constructions.

4) use test before and test after in the perform stmt for writing do-while constructions.

**Q89) Read the following code. 01 ws-n pic 9(2) value zero. a-para move 5 to ws-n. perform b-para ws-n times. b-para.**

**move 10 to ws-n. how many times will b-para be executed ?**

A89) 5 times only. it will not take the value 10 that is initialized in the loop.

**Q90) What is the difference between SEARCH and SEARCH ALL? What is more efficient?**

A90) SEARCH is a sequential search from the beginning of the table. SEARCH ALL is a binary search, continually dividing the table in two halves until a match is found. SEARCH ALL is more efficient for tables larger than 70 items.

**Q91) What are some examples of command terminators?**

A91) END-IF, END-EVALUATE

**Q92) What care has to be taken to force program to execute above 16 Meg line?**

A92) Make sure that link option is AMODE=31 and RMODE=ANY. Compile option should never have SIZE(MAX). BUFSIZE can be 2K, efficient enough.

**Q93) How do you submit JCL via a Cobol program?**

A93) Use a file //dd1 DD sysout=(\*, intrdr)write your JCL to this file. Pl some on try this out.

**Q94) How to execute a set of JCL statements from a COBOL program**

A94) Using EXEC CICS SPOOL WRITE(var-name) END-EXEC command. var-name is a COBOL host structure containing JCL statements.

**Q95) Give some advantages of REDEFINES clause.**

A95)

1. You can REDEFINE a Variable from one PICTURE class to another PICTURE class by using the same memory

location.

2. By REDEFINES we can INITIALISE the variable in WORKING-STORAGE Section itself.

3. We can REDEFINE a Single Variable into so many sub variables. (This facility is very useful in solving Y2000

Problem.)

**Q96) What is the difference between static call & Dynamic call**

A96) In the case of Static call, the called program is a stand-alone program, it is an executable program. During run time we can call it in our called program. As about Dynamic call, the called program is not an executable program it can executed through the called program

**Q97) What do you feel makes a good program?**

A97) A program that follows a top down approach. It is also one that other programmers or users can follow logically and is easy to read and understand.

**Q98) How do you code Cobol to access a parameter that has been defined in JCL? And do you code the PARM**

**parameter on the EXEC line in JCL?**

**A98)**

1) using JCL with sysin. //sysin dd \*here u code the parameters(value) to pass in to cobol program /\* and in program

you use accept variable name(one accept will read one row)/.another way.

2) in jcl using parm statement ex: in exec statement parm='john','david' in cobol pgm u have to code linkage section in that for first value you code length variable and variable name say, abc pic x(4).it will take john inside to read next value u have to code another variable in the same way above mentioned.

**Q99) Why do we code S9(4) comp. Inspite of knowing comp-3 will occupy less space.**

A99) Here s9(4)comp is small integer ,so two words equal to 1 byte so totally it will occupy 2 bytes(4 words).here in s9(4) comp-3 as one word is equal to 1/2 byte.4 words equal to 2 bytes and sign will occupy 1/2 byte so totally it will occupy 3 bytes.

**Q100) The maximum number of dimensions that an array can have in COBOL-85 is ----------- ?**

A100) SEVEN in COBOL - 85 and THREE in COBOL - 84

**Q101) How do you declare a host variable (in COBOL) for an attribute named Emp-Name of type VARCHAR(25) ?**

**A101)**

01 EMP-GRP.

49 E-LEN PIC S9(4) COMP.

49 E-NAME PIC X(25).

**Q102) What is Comm?**

A102) COMM - HALF WORD BINARY

**Q103) Differentiate COBOL and COBOL-II. (Most of our programs are written in COBOLII, so, it is good to know,**

**how, this is different from COBOL)**

A103) The following features are available with VS COBOL II:

1. MVS/XA and MVS/ESA support The compiler and the object programs it produces can be run in either

24- or 31-bit addressing mode.

2. VM/XA and VM/ESA support The compiler and the object programs it produces can be run in either

24- or 31-bit addressing mode.

3. VSE/ESA support The compiler and the object programs it produces can be run under VSE/ESA.

**Q104) What is PERFORM ? What is VARYING ? (More details about these clauses)**

A104) The PERFORM statement is a PROCEDURE DIVISION statement which transfers control to one or more specified procedures and controls as specified the number of times the procedures are executed. After execution of the specified procedures is completed (i.e., for the appropriate number of times or until some specified condition is met), control is transferred to the next executable statement following the PERFORM statement. There are 5 types of PERFORM statements:

a) Basic PERFORM

b) PERFORM TIMES

c) PERFORM UNTIL

d) PERFORM VARYING

e) IN-LINE PERFORM

**Q105) How many sections are there in data division?.**

A105) SIX SECTIONS 1.FILE SECTION 2.WORKING-STORAGE SECTION 3. LOCAL-STORAGE SECTION 4.SCREEN SECTION 5.REPORT SECTION 6. LINKAGE SECTION

**Q106) What is Redefines clause?**

A106) Redefines clause is used to allow the same storage allocation to be referenced by different data names .

**Q107) How many bytes does a s9(4)comp-3 field occupy?**

A107) 3Bytes (formula : n/2 + 1))

**Q108) What is the different between index and subscript?**

A108) Subscript refers to the array of occurrence , where as Index represents an occurrence of a table element. An index can only modified using perform, search & set. Need to have an index for a table in order to use SEARCH and SEARCH All.

**Q109) What is the difference between Structured COBOL Programming and Object Oriented COBOL**

**programming?**

A109) Structured programming is a Logical way of programming, you divide the functionalities into modules and code logically. OOP is a Natural way of programming; you identify the objects first, and then write functions, procedures around the objects. Sorry, this may not be an adequate answer, but they are two different programming paradigms, which is difficult to put in a sentence or two.

**Q110) What divisions, sections and paragraphs are mandatory for a COBOL program?**

A110) IDENTIFICATION DIVISION and PROGRAM-ID paragraph are mandatory for a compilation error free COBOL

program.

**Q111) Can JUSTIFIED be used for all the data types?**

A111) No, it can be used only with alphabetic and alphanumeric data types.

**Q112) What happens when we move a comp-3 field to an edited (say z (9). ZZ-)**

A112) the editing characters r to be used with data items with usage clause as display which is the default. When u tries displaying a data item with usage as computational it does not give the desired display format because the data item is stored as packed decimal. So if u want this particular data item to be edited u have to move it into a data item whose usage is display and then have that particular data item edited in the format desired.

**Q113) What will happen if you code GO BACK instead of STOP RUN in a stand-alone COBOL program i.e. a program which is not calling any other program ?**

A113) Both give the same results when a program is not calling any other program. GO BACK will give the control to the system even though it is a single program.

**Q114) what is the difference between external and global variables?**

A114) Global variables are accessible only to the batch program whereas external variables can be referenced from any batch program residing in the same system library.

**Q115) You are writing report program with 4 levels of totals: city, state, region and country. The codes being used can be the same over the different levels, meaning a city code of 01 can be in any number of states, and the same applies to state and region code so how do you do your checking for breaks and how do you do add to each level?**

A115) Always compare on the highest-level first, because if you have a break at a highest level, each level beneath it must also break. Add to the lowest level for each record but add to the higher level only on a break.

**Q116) What is difference between COBOL and VS COBOL II?.**

A116) In using COBOL on PC we have only flat files and the programs can access only limited storage, whereas in VS COBOL II on M/F the programs can access up to 16MB or 2GB depending on the addressing and can use VSAM

files to make I/O operations faster.

**Q117) Why occurs can not be used in 01 level ?**

A117) Because, Occurs clause is there to repeat fields with same format, not the records.

**Q118) What is report-item?**

A118) A Report-Item Is A Field To Be Printed That Contains Edit Symbols

**Q119) Difference between next and continue clause**

A119) The difference between the next and continue verb is that in the continue verb it is used for a situation where there in no EOF condition that is the records are to be accessed again and again in an file, whereas in the next verb the indexed file is accessed sequentially, read next record command is used.

**Q120) What is the Importance of GLOBAL clause According to new standards of COBOL**

A120) When any data name, file-name, Record-name, condition name or Index defined in an Including Program can be referenced by a directly or indirectly in an included program, Provided the said name has been declared to be a global name by GLOBAL Format of Global Clause is01 data-1 pic 9(5) IS GLOBAL.

**Q121) What is the Purpose of POINTER Phrase in STRING command**

A121) The Purpose of POINTER phrase is to specify the leftmost position within receiving field where the first transferred character will be stored

**Q122) How do we get current date from system with century?**

A122) By using Intrinsic function, FUNCTION CURRENT-DATE

**Q123) What is the maximum length of a field you can define using COMP-3?**

A123) 10 Bytes (S9(18) COMP-3).

**Q124) Why do we code s9 (4) comp? In spite of knowing comp-3 will occupy less space?**

A124) Here s9(4)comp is small integer, so two words equal to 1 byte so totally it will occupy 2 bytes(4 words).here in s9(4) comp-3 as one word is equal to 1/2 byte.4 words equal to 2 bytes and sign will occupy 1/2 byte so totally it will occupy 3 bytes.

**Q125) What is the LINKAGE SECTION used for?**

A125) The linkage section is used to pass data from one program to another program or to pass data from a PROC to a program.

**Q126) Describe the difference between subscripting and indexing ?**

A126) Indexing uses binary displacement. Subscripts use the value of the occurrence.

1. What R 2 of the common forms of the EVALUATE STATEMENT ?

2. What does the initialize statement do ?

3. What is the reference modification.

4. Name some of the examples of COBOl 11?

5. What are VS COBOL 11 special features?

6. What are options have been removed in COBOL 11?

7. What is the file organization clause ?

8. What is a subscript ?

9. What is an index for tables?

10. What are the two search techniques ?

11. What is an in-line perform ?

12. What is CALL statement in COBOL?

13. When can the USING phrase be included in the call statement ?

14. In EBCDIC, how would the number 1234 be stored?

15. How would the number +1234 be stored if a PIC clause of PICTUREs9(4) comp-3 were used?

16. What is Alternate Index ? How is it different from regular index ?

提纲

**丛书序**

**第1章 COBOL85程序设计概述**

**§1.1 COBOL语言简介**

**1.1.1 什么是COBOL**

**1.1.2 COBOL语言的特点**

**1.1.3 COBOL语言的历史与发展**

**§1.2 第一个COBOL程序——HELLO COBOL！**

**1.2.1 程序代码**

**1.2.2 部(DIVISION)**

**1.2.3 节(SECTION)**

**1.2.4 句子、语句和子句**

**1.2.5 COBOL85程序结构及书写格式**

**1.2.6 COBOL程序的数据分层**

**1.2.7 COBOL字符集**

**§1.2.8 COBOL字**

**§1.2.8.1 保留字**

**§1.2.8.2 表意常量**

**§1.2.8.3 用户字**

**§1.2.9 COBOL所处理的数据的特点**

**§1.3 本章小结**

**思考与练习**

**思考与练习解答**

**案例一：HELLO COBOL**

**第2章 轻松上手**

**§2.1 认识数据部**

**2.1.1 了解数据部**

**2.1.2 数据部的节**

**2.1.3 COBOL数据**

**2.1.4 数据组成层次结构**

**§2.2 工作单元节（WORKING-STORAGE SECTION）**

**§2.3 字型语句（PICTURE子句）**

**2.3.1 数值型数据描述**

**2.3.1.1 “9”描述符**

**2.3.1.2 “S”描述符**

**2.3.1.3 “V”描述符**

**2.3.1.4 “P”描述符**

**2.3.1.5 数值型变量小结**

**2.3.2 字母型数据描述**

**2.3.2.1 “A”描述符**

**2.3.2.2 "X"描述符**

**§2.4 赋初值子句（VALUE子句）**

**§2.5 认识过程部**

**2.5.1 带有DECLARATIVES的过程部**

**2.5.2 接收语句(ACCEPT语句)**

**2.5.3 显示语句(DISPLAY语句)**

**2.5.4 数据项初始化（INITIALIZE）**

**2.5.5 传送语句(MOVE语句)**

**2.5.6 停止语句(STOP语句)**

**2.5.7 算术运算语句 12**

**2.5.7.1 加法语句(ADD语句)**

**2.5.7.2 减法语句(SUBTRACT语句)**

**2.5.7.3 乘法语句(MULTIPLY语句)**

**2.5.7.4 除法语句(DIVIDE语句)**

**2.5.7.5 计算语句(COMPUTE语句)**

**2.5.8 逻辑与控制语句**

**2.5.8.1 转移语句(GOTO语句)**

**2.5.8.2 条件表达式**

**2.5.8.3 条件语句(IF语句)**

**2.5.8.4 IF嵌套与分析**

**2.5.9 执行语句（PERFORM 语句）**

**2.5.9.1 概述**

**2.5.9.2 基本形**

**2.5.9.3 PERFORM 语句与其他语句的区别**

**2.5.9.4 执行语句（PERFORM 语句）的嵌套**

**2.5.9.5 执行语句（PERFORM 语句）语句序列中的转移**

**2.5.9.6 执行语句（PERFORM 语句）语句计数循环**

**2.5.9.7 执行语句（PERFORM 语句）UNTIL循环**

**2.5.9.8 带有循环控制变量的PERFORM 语句**

**2.5.10 出口语句（EXIT）**

**2.5.11 综合实例**

**2.5.12 编码规约**

**§2.6 本章小结**

**思考与练习**

**思考与练习解答**

**第3章 数据处理及显示**

**§3.1 数据在计算机内的表示形式**

**3.1.1 计算机内存的组织形式**

**3.1.2 数值型数据在内存中的存储数据**

**3.1.3 数据描述与存储形式的关系**

**§3.2 用法子句（USAGE子句）**

**§3.3 重定义子句(REDEFINES子句)**

**§3.4 重命名子句(RENAMES子句)**

**§3.5 遇零置空子句（BLANK WHEN ZERO）**

**§3.6 对齐子句(JUSTIFIED子句)**

**§3.7 同步安置子句（SYNCHRONIZED）**

**§3.8 编辑型数据描述**

**3.8.1 “.”描述符**

**3.8.2“+”描述符**

**3.8.3“-”描述符**

**3.8.4插入逗号“，”做分位符**

**3.8.5“$”描述符**

**3.8.6“\*”描述符**

**3.8.7插入空格、“0”**

**3.8.8“DB”、“CR”**

**3.8.9描述符知识点整理**

**§3.9 算术运算精度处理**

**3.9.1 四舍五入处理（ROUNDED子句）**

**3.9.2 长度溢出处理（ON SIZE ERROR）**

**3.9.3 除法语句的余数子句(REMAINDER子句)**

**§3.10 对应传送（或称作同名传送CORR）**

**§3.11 本章小结**

**思考与练习**

**思考与练习解答**

**第4章 字符串处理**

**§4.4 本章小结**

**思考与练习**

**思考与练习解答**

**第5章 逻辑控制**

**§5.1 条件表达式**

**§5.2 IF-ELSE语句**

**5.2.1**

**5.2.2 IF嵌套与分析**

**§5.3 EVALUATE语句**

**§5.4 CONTINUE语句**

**§5.5 GO TO**

**§5.6 条件关系**

**§5.7 复合条件**

**§5.8 复写语句**

**§5.9 本章小结**

**思考与练习**

**思考与练习解答**

**第6章 表结构与检索**

**§6.1 表的基本概念**

**§6.2 表的建立**

**§6.3 下标法引用表元素**

**§6.4 表元素赋初值**

**§6.5 表的应用**

**§6.6 位标法引用表元素**

**6.6.1 位标的概念**

**6.6.2 位标法引用表元素规则**

**6.6.3 设置语句**

**6.6.4 位标数据项定义**

**§6.7 表的顺序检索**

**§6.8 表的对分检索**

**§6.9 用PERFORM语句对表进行检索**

**§6.10 本章小结**

**思考与练习**

**思考与练习解答**

**第7章 子程序**

**§7.1 子程序概述**

**§7.2 主程序与子程序连接**

**§7.3 子程序结构**

**§7.4 调用语句**

**§7.5 联接节**

**§7.6 子程序返回语句**

**§7.7 公用区数据**

**§7.8 子程序举例**

**§7.9 知识小结**

**思考与练习**

**思考与练习解答**

**案例二：习题答案演示系统**

**第8章 深入了解COBOL的部**

**§8.1 标识部**

**§8.2 环境部**

**8.2.1 环境部概述**

**8.2.2 配置节**

**8.2.3 文件组织方式**

**8.2.4 输入输出节**

**8.2.4.1 输入输出节概述**

**8.2.4.2 文件控制段**

**8.2.4.3 输入输出控制段**

**§8.3 数据部**

**8.3.1 数据部概述**

**8.3.2 文件节**

**8.3.3 工作单元节（WORKING-STORAGE SECTION）**

**§8.4 过程部**

**§8.5 本章小结**

**思考与练习**

**思考与练习解答**

**第9章 COBOL的文件处理**

**§9.1 文件处理常用语句**

**9.1.1 打开语句（OPEN语句）**

**9.1.2 读语句（READ语句）**

**9.1.3 写语句（WRITE语句）**

**9.1.4 重写语句REWRITE**

**9.1.5 删除语句DELETE**

**9.1.6 关闭语句（CLOSE语句）**

**9.1.7 开始语句START**

**§9.2 语句汇总**

**§9.3 文件操作**

**9.3.1 磁带文件**

**9.3.2 磁盘顺序文件**

**9.3.3 磁盘索引文件**

**9.3.4 磁盘相对文件**

**§9.4 动态存取方法简介**

**§9.5 本章小结**

**思考与练习**

**思考与练习解答**

**第10章 排序**

**§10.1 基本概念**

**§10.2 环境部中排序文件的描述**

**三、 数据部中排序文件描述**

**四、 排序语句（SORT）单纯排序**

**五、 排序语句（SORT）加工排序**

**六、 处理过程**

**第11章 合并**

**六、MERGE语句(SAMPLE11-3)**

**知识小结**

**思考与练习**

**思考与练习解答**

**第12章 报表 （未确定是否需要）**

**第13章COBOL内部函数**

**案例三：个人信息管理系统（FILE版）**

**第14章 嵌入式数据SQL**

1. **ORACLE PRO\*COBOL编程**

**案例四 个人信息管理系统（DB版）**

1. **项目实战**

**§16.1 项目一 个人股票操作管理系统**

**附录 1 COBOL关键字**

**附录2 应用文本编辑器COBOL程序（EDITPLUS、UESTUDIO）**

**附录3 RMCOBOL编译、运行Cobol程序**

**附录4 NETCOBOL单机软件开发环境配置**

**§附录2.1 Windows环境设置**

**§附录2.2 本机用户配置（如不联网操作可略过）**

**§附录2.3 文本编辑时常用的快捷键**

**§附录2.4 NETCOBOL环境配置文件编辑**

**附录2.4.1 在项目文件夹下编辑<程序名.CBI>文件**

**附录2.4.2 执行参数的设置，在程序文件夹中编辑COBOL85.CBR文件**

**§附录2.5 NETCOBOL建立一个项目工程。**

**§附录2.6 PCO格式文件的关联编辑方法。**

**§附录2.7 输入全角空格时显示的字符**

**附录5 用EXCEL比较数据**

**附录6 COBOL编码规约**

**附录7 COBOL CHECK LIST**

**附录8 名词解释**

**附录9 COBOL85语法汇总**

**附录10 自我测试**

**参考文献**

**某股票交易记录表**

**STOCK-TABLE（外部文件名STOCK-T）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **字段名** | **流水号** | **客户号** | **股票号** | **交易类别** | **交易数量** | **交易单价** |
| **数据项名** | **BOURSE -NO** | **CLIENT-NO** | **STOCK-NO** | **BOURSE –CALSS** | **BOURSE –** **HOW** | **STOCK –SHOP** |
| **记录** | **000112** | **12459** | **000549** | **1** | **5000** | **1** |
| **000113** | **14587** | **001456** | **0** | **2000** | **1.2** |
| **000114** | **48765** | **001456** | **1** | **8000** | **8.26** |
| **000115** | **12459** | **001146** | **0** | **4000** | **45.5** |
| **000116** | **14587** | **014560** | **0** | **1000** | **1.235** |
| **000117** | **12459** | **001456** | **1** | **3000** | **4.58** |

**股票代码表**

**STOCK-NN-T（外部文件名STOCK-N）**

|  |  |
| --- | --- |
| **股票号** | **股票名称** |
| **STOCK-NO** | **STOCK–NAME** |
| **000549** | **Abc** |
| **001146** | **Eys** |
| **001456** | **Ert** |
| **014560** | **Hsji** |

**客户号**

**CLIENT-NO-T（外部文件名CLIENT-N）**

|  |  |
| --- | --- |
| **客户号** | **客户名** |
| **CLIENT-NO** | **CLIENT-NAME** |
| **12459** | **王** |
| **14587** | **刘** |
| **48765** | **陶** |

**交易类别**

**Bourse-class**

**1                  买进**

**0                      卖出**

**请根据以上文件输出客户<王>当日交易信息文件，并统计买入和卖出股票金额合计**

**客户姓名由屏幕输入**

**WANG-T（外部文件wang）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **股票名称** | **交易数量** | **交易类别** | **交易单价** | **交易金额** |
| **STOCK–NAME** | **BOURSE –** **HOW** | **BOURSE –CALSS** | **STOCK –SHOP** | **STOCK - MONEY** |
|  |  |  |  |  |
|  |  |  |  |  |

**买入股票金额合计：        元**

**卖出股票金额合计：        元**

**要求在屏幕显示输出文件内容及文件操作相关统计信息。**

**文件描述参考：**

**FD  STOCK-TABLE  LABEL RECORD STANDARD.**

**01  STOCK-REC.**

**03 BOURSE-NO           PIC 9(6).**

**03 CLIENT-NO            PIC 9(5).**

**03 STOCK-NO             PIC 9(6).**

**03 BOURSE-CALSS     PIC 9.**

**03 BOURSE-HOW       PIC 9(6).**

**03 STOCK-SHOP         PIC 9(5)V9(3).**

**\***

**FD  STOCK-NN-T     LABEL RECORD STANDARD.**

**01  STOCK-NN-REC.**

**03 STOCK-NO      PIC 9(6).**

**03 STOCK-NAME    PIC X(6).**

**\***

**FD  CLIENT-NO-T    LABEL RECORD STANDARD.**

**01  CLIENT-NO-REC.**

**03 CLIENT-NO     PIC 9(5).**

**03 CLIENT-NAME   PIC X(12).**

**\***

**FD  WANG-T  LABEL RECORD OMITTED.**

**01  WANG-REC.**

**03 STOCK-NAME    PIC X(132).**

**03 BOURSE-HOW    PIC 9(6).**

**03 BOURSE-CALSS  PIC 9.**

**03 STOCK-SHOP    PIC 9(5)V9(3).**

**03 STOCK-MONEY   PIC 9(9)V9(3).**

产品销售记录统计

main.cob

说明：

输入文件结构INFILE

01 IN-REC.

    02 PRODUCT-CODE PIC X(6).         产品号

    02 CUSTOMER-CODE PIC 9(4).        顾客号

    02 QUANTITY     PIC 9(4).         购买数量

输出文件结构 OUTFILE

01 OUT-REC.

    02 PRODUCT-CODE PIC X(6).         产品号

    02 CUSTOMER-COUNT PIC 9(3).       购买该产品的顾客数

    02 TOTAL-PRICE PIC 9(10).         该产品的销售额

主要过程

（1）调用PRODCT-TABLE子程序读入单价表

单价表结构：

02 T VALUE IS ZERO.

    03 PRODUCT-CODE-TABLE OCCURS 20 TIMES

        ASCENDING KEY IS PRODUCT-CODE-T

        INDEXED BY I.

        05 PRODUCT-CODE-T PIC X(6).              产品号

        05 UNIT-PRICE-T PIC 9(4).                单价

（2）调用SEARCH-PRICE子程序查找出产品的单价

（3）统计每件产品的客户数以及销售额，输出文件

MAKE-FILE.Cob

说明：

1.输入文件的结构为（PRODUCT-TABLE-NOINDEXED)

01 IN-REC

    02 PRODUCT-CODE PIC X(6).                    产品号

    02 UNIT-PRICE   PIC 9(4).                    单价

2.输出文件结构为（PRODUCT-TABLE）

01 IN-REC.

    02 PRODUCT-CODE PIC X(6).                    产品号

    02 UNIT-PRICE   PIC 9(4).                    单价

3.该程序的主要过程为将输入文件转换成索引文件并输出。

PRODUCT-TABLE.COB

说明：1.输入文件的结构为（PRODUCT-TABLE）；

01 IN-REC.

    02 PRODUCT-CODE PIC X(6).                    产品号

    02 UNIT-PRICE   PIC 9(4).                    单价

该程序的主要过程为

（1）调用MAKE-FILE.CBL子程序

（2）将文件PRODUCT-TABLE中的产品信息输入表

01 T-S.

    02 PRODUCT-CODE-TABLE-S OCCURS 20 TIMES

        ASCENDING KEY IS PRODUCT-CODE-S

        INDEXED BY I.

        03 PRODUCT-CODE-S PIC X(6).

        03 UNIT-PRICE-S   PIC 9(4).

SEARCH-PRICE-COB

说明：1.该程序的主要过程为

从表T-S.

01 T-S.

    02 PRODUCT-CODE-TABLE-S OCCURS 20 TIMES

        ASSENDING-KEY IS PRODUCT-CODE-S

        INDEXED BY I.

        03 PRODUCT-CODE-S PIC X(6).

        03 UNIT-PRICE-S     PIC 9(4).

中检索中检索出某种产品的单价，为MAIN.CBL所使用

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.