ISSC Shanghai, AMS, GCG

PLI Programming

For Entry Level Training

ISSC SH
Walker JIA

Version 1.1

2004/03

日程表

	第一天	第二天	第三天
前半	**1. 编程基础** A. 记述规则 B. 变量,文件的定义 C. 基本命令 练习一	**3. 程序流程** A. 数据处理的基本方法 B. 输出的处理 **4. 编程技巧** A. 过程与函数 B. 匹配的处理 C.结构化编程	练习三 练习四
后半	**2. 调试** A. 编译 B. 调试方法 练习二	练习三	练习四 **5. AMS标准介绍** **Appedix. Using DB2**

Table of contents

1. 编程基础

本章内容

- 记述规则
- 变量,文件的定义
- 基本命令

学习目标

能够说明PL/I的基本记述规则和命令

1. 编程基础

A. 记述规则 – PL/I 的特征

PL/I

Programming
Language
I (No.1)

§ **通用性**
适用于科学计算,事务处理等等各方面

§ **容易理解性**
使用日常生活相近的语言(英语)
(例)　IF TOKUTEN >= 60
　　THEN HANTEI = “ｺﾞｳｶｸ”;
　　ELSE HANTEI =“ﾌｺﾞｳｶｸ”;

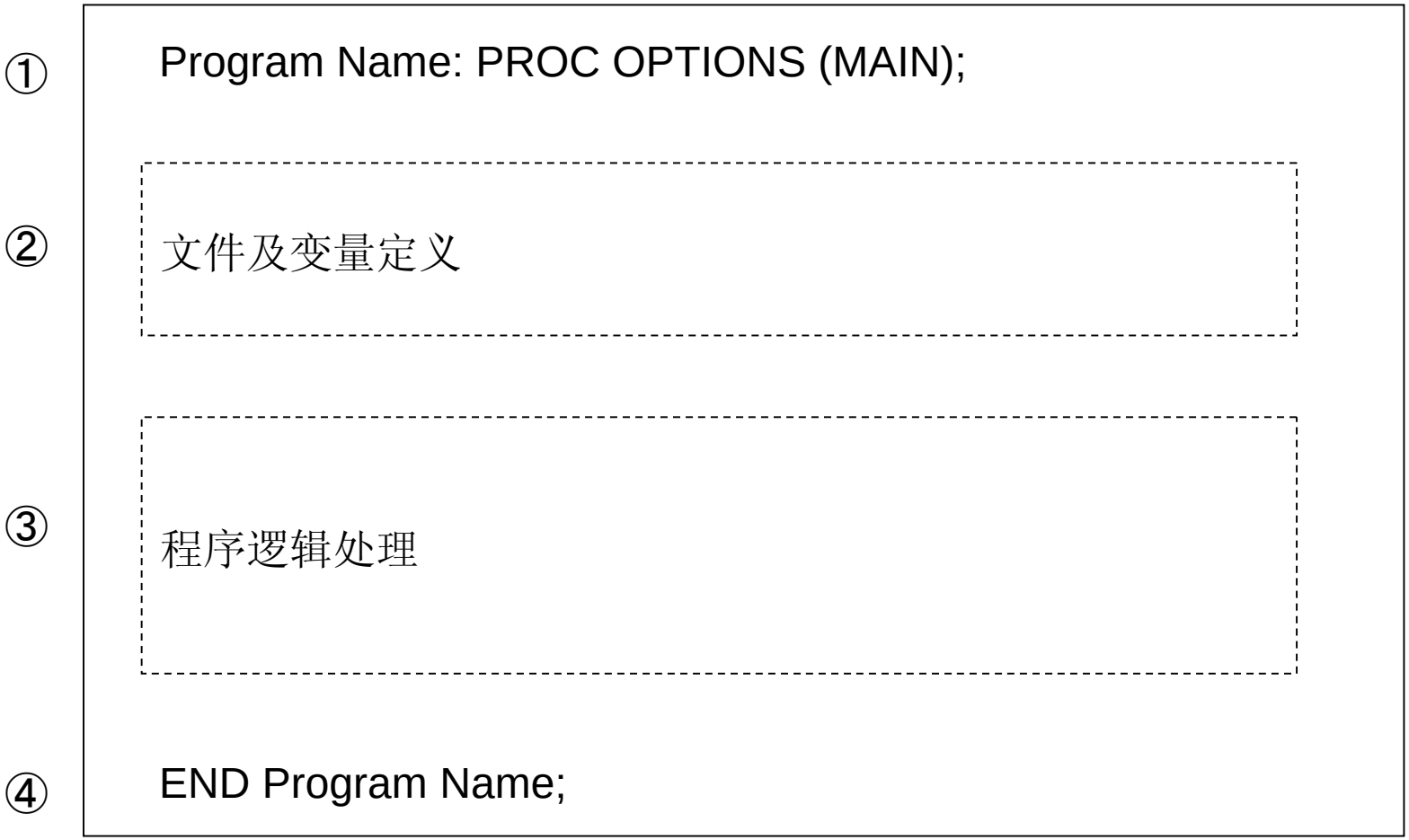
§ **书写简单性**
形式自由 – 记述的位置没有限定

可用省略形式性 – 记述变得简单
(例)　DECLARE　->　DCL　　CHARACTER　-> CHAR

省略时自动解释 – 能够自动判断省略的内容
(例)　PUT FILE(SYSPRINT) EDIT(“THIS IS A TEST!”) (A);
　　-> PUT EDIT(“THIS IS A TEST!”) (A);

1. 编程基础

A. 记述规则 – 程序的构成



PROC是PROCEDURE的省略

1. 编程基础

A. 记述规则 – 程序的构成(例)

①

②

③

④

```
REI01: PROC OPTIONS (MAIN);

   DCL
      SHOHIN_BANGO   CHAR( 5),
      SHOHIN_MEI     CHAR(10),
      TANKA          FIXED DEC( 5),
      SURYO          FIXED DEC( 7),
      URIAGE         FIXED DEC(11);

   /*** GET INPUT DATA  ****/
   GET EDIT(SHOHIN_BANGO,SHOHIN_MEI,TANKA,SURYO)
           (A(5),A(10),F(5),F(6));
   /*** COMPUTE ****/
   URIAGE = TANKA * SURYO;
   /*** OUTPUT  ****/
   PUT EDIT(SHOHIN_BANGO,SHOHIN_MEI,URIAGE)
           (A(5),X(3),A(10),X(2),F(11));

END REI01;
```

程序开始

变量定义

逻辑处理

程序结束

1. 编程基础

A. 记述规则 – 规则

①程序名

- **英文字母或者数字**
- **以英文字母开头**
- **7文字以内**
- **与其它的程序不能同名**

②记述的位置

1. 编程基础

A. 记述规则 – 规则

③语句的形式

- 变量,保留字,符号等之间用一个或以上的空格分隔
- 以分号结束一条语句

④PL/I里可以使用的文字

- 数字
- 英文字母
- 特殊记号 = + - * / ( ) ; : . ‘ & | > < _ % ? HEX(‘5F’)
- 空格

⑤注释 用/* */

1. 编程基础

B.变量,文件的定义 – 变量的定义

a.变量的定义(DECLARE语句)

```
DCL   变量名        属性;
```

- DCL　　DECLARE的省略
- 变量名　31个文字以内;英文字母,数字或者下划线组成;以字母开头
- 属性　　变量的类型及长度

如果几个变量具有相同的属性,可以用类似下面的方式定义:

```
DCL  (A,B,C,D)  CHAR(5);
```

1. 编程基础

B.变量,文件的定义 – 变量的定义

属性

1) CHARACTER (文字)　　CHAR(n): n列的文字

2) FIXED DECIMAL (固定10进制数)　FIXED DEC(p,q)

总长度为p的数字，q位小数

数值	精度			最小值	最大值
3.1416	(5,4) →	FIXED DEC (5,4) →		-9.9999	9.9999
425	(3,0) →	FIXED DEC (3,0) →		- 999	999
5908.3	(5,1) →	FIXED DEC (5,1) →		-9999.9	9999.9
007	(3,0) →	FIXED DEC (3,0) →		-999	999
-3510	(4,0) →	FIXED DEC (4,0) →		-9999	9999
.0048	(4,4) →	FIXED DEC (4,4) →		-0.9999	0.9999

最大精度　FIXED DEC(15)

1. 编程基础

B.变量,文件的定义 – 变量的定义

属性

3) FIXED BINARY (固定2进制数) FIXED BIN(p,q)
总长度为2^p的数字, 2^q位小数

数值	精度			最小值	最大值
10110B	(5,0) →	FIXED BIN (5)	→	-32	31
11111B	(5,0) →	FIXED BIN (5)	→	-32	31
-101B	(3,0) →	FIXED BIN (3)	→	-8	8
1011.111B	(7,3) →	FIXED BIN (7,3)	→	-16	15

最大精度 FIXED BIN(31) -2^p 2^p-1

4) **其它数据**类型

BIT	bit data
GRAPHIC	**双字**节
FLOAT DECIMAL	**浮**动10进制数
FLOAT BINARY	浮动2进制数

1. 编程基础

B.变量,文件的定义 – 变量的定义

属性

5) PICTURE(编辑用文字)　PIC'ZZZ9'

	要编辑的数值	PIC 指定	结果
	01111	99999	01111
	01111	ZZZZZ	1111
	01111	ZZ,ZZ9	1,111
	00222	999B99	002_22
	00222	ZZBBZZ	2__22
	0	ZZZ9	0
	0	ZZZZ	
	01111	$$,$$9	$1,111

1. 编程基础

B.变量,文件的定义 – 结构的定义

```
DCL   1    结构名
          层次      变量名1          属性,
          层次      变量名2          属性,
          层次      变量名3          属性,
          ....        .....          .... ,
          层次      变量名n          属性;
```

层次: 2-255之间的整数(最大15层)

(例)

```
DCL   1   A1,
          3   A11   CHAR(3),
          3   A12   FIXED DEC(9),
          3   A13   CHAR(4);
```

1	A1		
3	A11 C(3)	A12 F(9)	A13 C(4)

1. 编程基础

B.变量,文件的定义 – 指针的定义

DCL　变量名　PTR;　　*PTR　POINTER的省略*

例1:

```
DCL  P  PTR;
DCL  A  CHAR(10) INIT( 'ABCDEFGHIJ' );
DCL  1  B BASED(P),
        3  B1  CHAR(7),
        3  B2  CHAR(3);
DCL  ADDR  BUILTIN;

P = ADDR(A);
```

例2:

```
DCL  (P,Q)  PTR;
DCL  A  CHAR(26) ;
DCL  1  B,
        3  B1  CHAR(4),
        3  B2  CHAR(5);
……
P = ADDR(A);
Q = ADDR(B);
CALL ASUB(P,Q);
……
*********************************
ASUB: PROC(X,Y);
    DCL (X,Y) PTR;

    DCL P1 CHAR(26) BASED(X);
    DCL P2 BASED(Y),
           3 P21 CHAR(4),
           3 P22 CHAR(5);
    … …
```

1. 编程基础

B.变量,文件的定义 – 初期值的设定

DCL 变量名 属性 INIT (常量);

INIT INITIAL的省略
常量 文字用单引号括起

```
DCL A11 CHAR(6)        INIT('SAMPLE');
DCL A12 FIXED DEC(9) INIT(0);
```

1. 编程基础

B.变量,文件的定义 –文件的定义

```
DCL    文件名     FILE  [ STREAM|RECORD ]  [ INPUT|OUTPUT|UPDATE ]  [ PRINT ];
```

1) DCL　　DECLARE的省略
2) 文件名　　以字母开头,7位以下的字母与数字组合
3) FILE　　表示定义的是文件
4) 数据传送类型　默认为STREAM(具体见下页)
5) 输入输出类型　指定文件的输入输出种类
6) 印刷属性　　印刷输出时指定

(例)　DCL　FTOUR　FILE　RECORD ;
　　DCL　FOUT　　FILE STREAM PRINT OUTPUT;

(注1) SYSIN 和SYSPRINT等的定义可以省略
(注2)上记4,5,6的选项可以在文件打开时指定

1. 编程基础

B.变量,文件的定义 –文件的定义

数据传送的种类

RECORD方式

文件是由记录(RECORD)组成的;
用READ,WRITE命令进行处理

STREAM方式

文件是连续的数据组成;
用GET,PUT命令进行处理

变量的默认值

PL/I里如果没有对变量进行定义,则有以下的规则:
字母I~N开头的变量默认为FIXED BIN(15)
I~N以外字母开头的变量默认为FLOAT DEC(6)

1. 编程基础

C.基本命令

a) OPEN
b) CLOSE
c) PUT
d) GET
e) DO
   (1)DO GROUP
   (2)DO WHILE
f) END
g) 赋值语句
h) ON STATEMENT
i) READ
j) WRITE
k) IF

1. 编程基础

C.基本命令 – a) OPEN

```
OPEN FILE(文件名)  [ STREAM|RECORD ] [SEQUENTIAL|DIRECT]
                   [ INPUT|OUTPUT|UPDATE ]  [ PRINT LINESIZE(nn) PAGESIZE(nn)];
```

i. FILE(文件名)　　文件名:程序中使用的必须与JCL中的DD名一致
ii. 数据传送类型　　默认为STREAM
iii. RECORD方式传送数据时使用的读取属性,默认为SEQUENTIAL
iv. 输入输出类型
v. 印刷属性　　STREAM,OUTPUT时有效
　　LINESIZE: 一行的长度,默认为120
　　PAGESIZE: 一页的行数,默认为60

(例)

```
OPEN FILE(FOUTPUT) RECORD OUTPUT;
OPEN FILE(SYSPRINT) OUTPUT PRINT LINESIZE(132) PAGESIZE(55);
```

1. 编程基础

C.基本命令 – b) CLOSE

```
CLOSE   FILE(文件名);
```

FILE(文件名)　　文件名:与OPEN的文件名一致

(例)

```
CLOSE FILE(FOUTPUT);
CLOSE FILE(SYSPRINT);
```

1. 编程基础

C.基本命令 – c) PUT

```
PUT   FILE(文件名) [SKIP] EDIT(变量1,变量2….)(格式1,格式2….);
PUT   FILE(文件名) [SKIP] EDIT(常量1,常量2….)(格式1,格式2….);
PUT   FILE(文件名) [SKIP] EDIT(变量1,常量2….)(格式1,格式2….);
```

PUT EDIT: 编辑输出
SKIP: 换行输出
格式: A(n) n位的文字输出—可省略
F(n) n位的数字输出
X(n) n位空格输出
P'…' 数字的编辑输出

(例)

```
DCL  A1  FIXED DEC(3) INIT( 123 ),
         A2  FIXED DEC(5) INIT( 45678 );
PUT FILE( FOUT ) EDIT ( A1, 'ABC', A2 ) (F(3),X(3),A(5),P'ZZZ,ZZ9');
->  123_ _ABC_ _45,678
```

1. 编程基础

C.基本命令 – d) GET

```
GET   FILE(文件名) [SKIP] EDIT(变量1,变量2….)(格式1,格式2….);
```

GET EDIT: 编辑输入
SKIP: 换行输入
格式: A(n) n位的文字输出—可省略
F(n) n位的数字输出
X(n) n位空格输出

(例)

```
DCL  H1  FIXED DEC(5) ,
     H2  CHAR(5),
     H2  FIXED DEC(7);
GET FILE( FIN ) EDIT ( H1, H2, H3 ) (F(3),X(3),A(3),F(5));
```

输入 123_ _ABC45,678
H1=123,H2='ABC',H3=45678

1. 编程基础

C.基本命令 – d) GET

```
GET   FILE(文件名) [SKIP] EDIT(变量1,变量2….)(格式1,格式2….);
```

GET EDIT: 编辑输入
SKIP: 换行输入
格式: A(n) n位的文字输出—可省略
F(n) n位的数字输出
X(n) n位空格输出

(例)

```
DCL  H1  FIXED DEC(5) ,
     H2  CHAR(5),
     H2  FIXED DEC(7);
GET FILE( FIN ) EDIT ( H1, H2, H3 ) (F(3),X(3),A(3),F(5));
```

输入 123_ _ABC45,678
H1=123,H2='ABC',H3=45678

1. 编程基础

C.基本命令 – e) DO

(1) DO GROUP
(DO与END之间的语句作为一个STEP执行)
(例)

```
IF A > 20
  THEN DO;
     B = X * 1.25;
     C = Y * 1.10;
   END;
   ELSE DO;
     B = X * 0.90;
     C = Y * 0.75;
   END;
```

(2) DO WHILE (循环处理)

```
DO WHILE(条件);
        循环处理;
END;
```

(3) DO 变量 = 初期值 TO 终了值 [BY 增减值]

1. 编程基础

C.基本命令 – f) END

BLOCK或者GROUP的终止标志

END [LABEL]

(例)

```
LBLX:  DO WHILE ( EOF = 0 );
           OUT_AREA.XX = IN_AREA.AA;
           PUT EDIT ( OUT_AREA.XX) ( A(80));
           GET EDIT ( IN_AREA.AA) ( A(80));
       END LBLX;
```

1. 编程基础

C.基本命令 – g) 赋值语句

(1) 变量名1 = 变量名2 | 常量

(例) X = Y;

EOF = 0;

(2) 变量名n = [变量名1 | 常量1] 操作符

[变量名2 | 常量2] ……;

操作符: + - * / **

(例) X = Y * (Z + 10);

URIAGE = URIAGE * 0.95;

GOHKEI_URI = GOHKEI_URI + URIAGE;

(3)变量名1 = 变量名2 , BY NAME;

(例) DCL 1 AA,

3 YY CHAR(2) INIT('04'),

3 MM CHAR(2) INIT('02'),

3 DD CHAR(2) INIT('11');

DCL 1 BB,

3 YY CHAR(2) ,

3 F CHAR (1) INIT('/'),

3 MM CHAR(2) ,

3 G CHAR(1) INIT('/'),

3 DD CHAR(2) ;

BB = AA, BY NAME;

1. 编程基础

C.基本命令 – h) ON STATEMENT

ON 条件 处理逻辑;

条件:
ERROR
ENDFILE
ENDPAGE
CONVERSION(CONV)
ZERODIVIDE(ZDIV)
FIXEDOVERFLOW(FOFL)

(例)

```
ON ENDFILE(FIN)  EOF = 1 ;

ON ENDPAGE (SYSPRINT)
    DO;
        PUT PAGE EDIT ( OUT_AREA.XX) ( A(80));
        PUT SKIP EDIT ( OUT_AREA.YY) ( A(80));
    END ;
```

1. 编程基础

C.基本命令 – i) READ

READ FILE(文件名)　INTO (变量名);

(例)
READ FILE(FINPUT) INTO (IN_AREA) ;

1. 编程基础

C.基本命令 – j) WRITE

WRITE FILE(文件名)　FROM (变量名);

(例)
WRITE FILE(FOUTPUT) FROM (OUT_AREA) ;

1. 编程基础

C.基本命令 – k) IF

IF 条件
  THEN 满足条件的处理;
   ELSE 不满足条件的处理;

比较符
= > < ~

逻辑运算符
() & |

QUIZ 1

请找出程序中的错误(共10处)

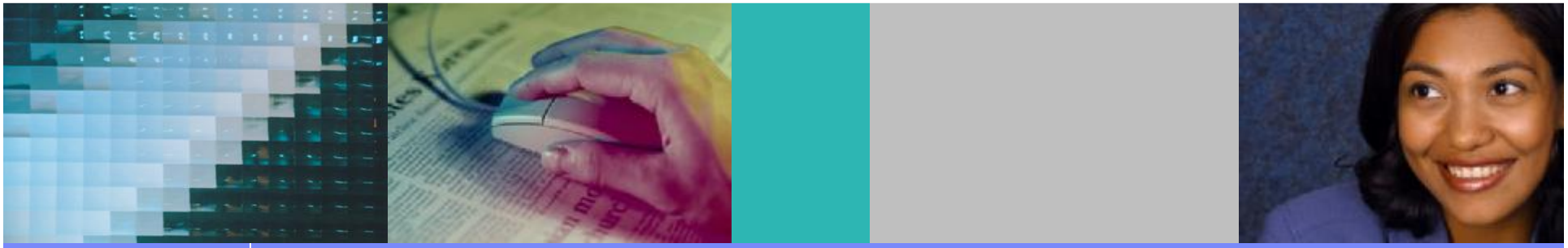


Table of contents

2. 调试

本章内容

- 开发流程
- 编译JCL
- 执行JCL
- 编译错误LEVEL
- 错误返回码
- CHECK LIST

学习目标

能够掌握PL/I编译,执行过程并学会处理简单的错误

2. 调试

A.开发流程

1. PLI程序编辑
2. PLI程序保存
3. 编译JCL编辑
4. PLI程序编译
5. 编译错误处理
6. 执行JCL编辑
7. PLI程序执行
8. 执行结果确认

2. 调试

B.编译JCL例

```
 表示     WD01I.EV6098.JCLM.BK0611(FTPXXXC) - 01.06    行  00000000  桁  001 080
  コマンド  ===>                                              スクロール  ===> CSR
******************************** データの始め ********************************
//EV6098CP JOB (F9500B,SA00X,31),                                        00001004
//            E27153YOSHI,                                               00002000
//            CLASS=M,                                                   00003000
//            MSGCLASS=R,                                                00004000
//            TIME=5,NOTIFY=EV6098                                       00006004
//*****************************************************************     00006100
//** THIS JCL IS TO COMPILE AND LINK EDIT PL/I PROGRAM INTO              00006200
//**          <<< PLI FOR MVS FOR LE ENVIRONMENT >>>                     00006300
//*****************************************************************     00006400
/*JOBPARM S=ANY                                                          00006500
//*                                                                      00006600
//IEL1CL  PROC LNGPRFX='SYS1.IEL',LIBPRFX='SYS1',                        00006700
//             SYSLBLK=3200,MBR=,                                        00006800
//             SOUT=*,                                                   00006900
//             CRD='SYS1.CRD', =============> CRD MASTER                00007000
//             TCRD='SYS1.TESTCRD', ========> CRD MASTER(TEST)           00008000
//             URD='SYS1.URD1', ===========> URD MASTER                  00009000
//             TURD='SYS1.TESTURD1', =======> URD MASTER(TEST)           00010000
//             SYSLIB=SYS1.DCL,                                          00020000
```

2. 调试

B.编译JCL例(续)

```
 表示     WD01I.EV6098.JCLM.BK0611(FTPXXXC) - 01.06   行  00000020  桁  001 080
  コマンド  ===>                                          スクロール  ===> CSR
//             SYSLIB2=SYS1.CTRL,                                        00021000
//             SYSLIB3=SYS1.UCTRL                                        00022000
//*            LOADLIB='SA00I.LEMIG.NCAL'   ===> LOAD MODULE LIBRARY     00025000
//*                                                                      00026000
//*********************************************************************  00027000
//*                                                                   *  00028000
//* LICENSED MATERIALS - PROPERTY OF IBM                              *  00029000
//*                                                                   *  00030000
//* 5688-235 (C) COPYRIGHT IBM CORP. 1993, 1995                       *  00040000
//* ALL RIGHTS RESERVED.                                              *  00050000
//*                                                                   *  00060000
//* US GOVERNMENT USERS RESTRICTED RIGHTS - USE,                      *  00070000
//* DUPLICATION OR DISCLOSURE RESTRICTED BY GSA                       *  00080000
//* ADP SCHEDULE CONTRACT WITH IBM CORP.                              *  00090000
//*                                                                   *  00091000
//*                                                                   *  00092000
//*********************************************************************  00093000
//*                                                                      00094000
//* IBM PL/I FOR MVS & VM  VERSION 1 RELEASE 1 MODIFICATION 1            00095000
//*                                                                      00096000
```

2. 调试

B.编译JCL例(续)

```
  表示     WD01I.EV6098.JCLM.BK0611(FTPXXXC) - 01.06   行  00000040  桁  001 080
  コマンド  ===>                                          スクロール  ===> CSR
//*  COMPILE AND LINK EDIT A PL/I PROGRAM                                  00097000
//*                                                                        00098000
//*  RELEASE LEVEL: 01.01.01  (VERSION.RELEASE.MODIFICATION LEVEL)         00099000
//*                                                                        00100000
//*  PARAMETER  DEFAULT VALUE    USAGE                                     00110000
//*   LNGPRFX    IEL.V1R1M1      PREFIX FOR LANGUAGE DATA SET NAMES        00120000
//*   LIBPRFX    CEE.V1R4M0      PREFIX FOR LIBRARY DATA SET NAMES         00130000
//*   SYSLBLK    3200            BLKSIZE FOR OBJECT DATA SET               00140000
//*   GOPGM      GO              MEMBER NAME FOR LOAD MODULE               00150000
//*-------------------------------------------------------------*          00160000
//EXPD    EXEC PGM=SC02ACAT,PARM=P                                         00170000
//STEPLIB  DD  DSN=DMS001A.LMOD2,DISP=SHR                                  00180000
//         DD  DSN=DMS001A.LMOD,DISP=SHR                                   00190000
//SYSPRINT DD  SYSOUT=&SOUT,DCB=BLKSIZE=629                                00200000
//ERRLIST  DD  SYSOUT=&SOUT                                                00201000
//OUT      DD  UNIT=DISK2,SPACE=(CYL,(2,1)),DISP=(,PASS),                  00202000
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)                         00203000
//S2ASW    DD  DSN=&TCRD,DISP=SHR                                          00204000
//         DD  DSN=&CRD,DISP=SHR                                           00205000
//         DD  DSN=&URD,DISP=SHR                                           00206000
```

2. 调试

B.编译JCL例(续)

```
  表示     WD01I.EV6098.JCLM.BK0611(FTPXXXC) - 01.06    行  00000060  桁  001 080
  コマンド  ===>                                          スクロール  ===> CSR
//          DD  DSN=&TURD,DISP=SHR                                      00207000
//S2ASW1    DD  UNIT=DISK2,SPACE=(CYL,(1,1)),                           00208000
//              DCB=(DSORG=PS,RECFM=FB,LRECL=83,BLKSIZE=1079)           00209000
//S2ASW2    DD  UNIT=DISK2,SPACE=(CYL,(1,1)),                           00209100
//              DCB=(DSORG=PS,RECFM=FB,LRECL=83,BLKSIZE=1079)           00209200
//TEXTLIB   DD  DUMMY,DCB=BLKSIZE=80                                    00209300
//IN        DD  DSN=&SRCLIB(&MBR),                                      00209400
//              DISP=SHR                                                00209500
//*-------------------------------------------------------------*       00209600
//*PLI       EXEC PGM=IEL1AA,PARM='OBJECT,NODECK',REGION=512K           00209700
//PLI       EXEC PGM=IEL1AA,REGION=512K,                                00209800
//          PARM='A,X,AG,M,MAR(2,72,1)'                                 00209900
//STEPLIB   DD  DSN=&LNGPRFX..SIELCOMP,DISP=SHR                         00210000
//          DD  DSN=&LIBPRFX..SCEERUN,DISP=SHR                          00220000
//SYSPRINT  DD  SYSOUT=*                                                00230000
//*SYSLIB    DD  DSN=SA00I.LEMIG.MINI.DCL,DISP=SHR                      00240001
//SYSLIB    DD  DSN=&SYSLIB,DISP=SHR                                    00241001
//          DD  DSN=SYS1.SCEESAMP,DISP=SHR                              00242000
//*SYSLIB2   DD  DSN=SA00I.LEMIG.DCL,DISP=SHR       <==ERR MACRO!!    00243001
//SYSLIB2   DD  DSN=&SYSLIB2,DISP=SHR                                   00243101
```

2. 调试

B.编译JCL例(续)

```
  表示     WD01I.EV6098.JCLM.BK0611(FTPXXXC) - 01.06    行  00000080  桁  001 080
  コマンド  ===>                                          スクロール  ===> CSR
//          DD  DSN=&SYSLIB3,DISP=SHR                                    00243200
//SYSLIB2   DD  DSN=&SYSLIB2,DISP=SHR                                    00243300
//SYSLIN    DD  DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,                00243400
//              SPACE=(80,(250,100)),DCB=(BLKSIZE=&SYSLBLK)              00243500
//SYSUT1    DD  DSN=&&SYSUT1,UNIT=SYSDA,                                 00243600
//              SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024     00243700
//SYSIN     DD  DSN=*.EXPD.OUT,DISP=(OLD,DELETE,DELETE)                  00243800
//*-------------------------------------------------------------------* 00243900
//ABEND02 EXEC PGM=UABEND,                                               00244000
//              COND=(9,GT,PLI)                                          00245000
//STEPLIB   DD DSN=SYS1.LINKLIBX,DISP=SHR                                00246000
//* ------------------------------------------------------------------- 00247000
//LKED      EXEC PGM=IEWL,COND=(9,LT,PLI),REGION=512K,                   00248000
//           PARM='LIST,XREF,LET,RMODE=ANY,AMODE=31'                     00249000
//*          PARM='LIST,XREF,LET,RMODE=24,AMODE=31'                      00250000
//*SYSLIB    DD  DSN=SA00I.LEMIG.NCAL,DISP=SHR          <== ERR MACRO    00260001
//*          DD  DSN=SA00I.LEMIG.LMOD,DISP=SHR                           00270001
//*         DD  DSN=SA00.DS.NCAL,DISP=SHR                                00280000
//SYSLIB    DD  DSN=DCP.DCPLIB,DISP=SHR                                  00290001
//           DD  DSN=&LIBPRFX..SCEELKED,DISP=SHR                         00300000
```

2. 调试

B.编译JCL例(续)

```
   表示     WD01I.EV6098.JCLM.BK0611(FTPXXXC) - 01.06   行  00000100  桁  001 080
  コマンド  ===>                                          スクロール  ===> CSR
//          DD  DSN=&LIBPRFX..SCEERUN,DISP=SHR                          00301000
//SYSPRINT DD  SYSOUT=*                                                 00302000
//SYSLIN    DD  DSN=&&LOADSET,DISP=(OLD,DELETE)                         00303000
//          DD  DDNAME=SYSIN                                            00304000
//SYSLMOD   DD  DSN=&LOADLIB(&MBR),DISP=SHR                             00305000
//SYSUT1    DD  DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,(200,20)),          00306000
//              DCB=BLKSIZE=1024                                        00307000
//SYSIN     DD  DUMMY                                                   00308000
//          PEND                                                        00309000
//*-------------------------------------------------------------------* 00310000
//STEP0A  EXEC  IEL1CL,MBR=FTPXXXP,                                     00320005
//           SRCLIB='WD01I.EV6098.PLI',                                 00331006
//           LOADLIB='WD01I.DS.UT.LOAD00'                               00332005
//                                                                      00333000
******************************** データの終り ********************************
```

2. 调试

C.执行JCL例

```
    表示     WD01I.EV6098.JCLM.BK0611(FTPXXXR) - 01.05   行  00000000  桁  001 080
  コマンド   ===>                                          スクロール  ===> CSR
******************************** データの始め ********************************
//EV6098CH JOB (F9500B,XX15X,41,
//          'TY=3,TM=05,TD=10'),
//          EV6098,
//          CLASS=M,MSGCLASS=R,REGION=8M,NOTIFY=EV6098
/*JOBPARM S=ANY
//* ---------------------------------------------------------------
//*  RUN THE MAIN PROGRAM                                          */
//* ---------------------------------------------------------------
//FTPXXX   PROC
//FTPXXXP   EXEC PGM=FTPXXXP,PARM='/HY=0,TY=&TY,TM=&TM,TD=&TD'
//STEPLIB DD DSN=WD01I.DS.UT.LOAD00,DISP=SHR
//SYSPRINT DD SYSOUT=*,
//            DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSIN     DD DUMMY
//INFILE    DD DSN=WD01I.EV6098.JCLM.STDTEST(FTPLLL1),DISP=SHR
//OUTFILE   DD DSN=WD01I.EV6098.JCLM.STDTEST(FTPLLL2),DISP=MOD
//    PEND
//* ---------------------------------------------------------------
//STEP010  EXEC  FTPXXX,TY=&TY,TM=&TM,TD=&TD
******************************** データの終り ********************************
```

2. 调试

D.编译错误LEVEL

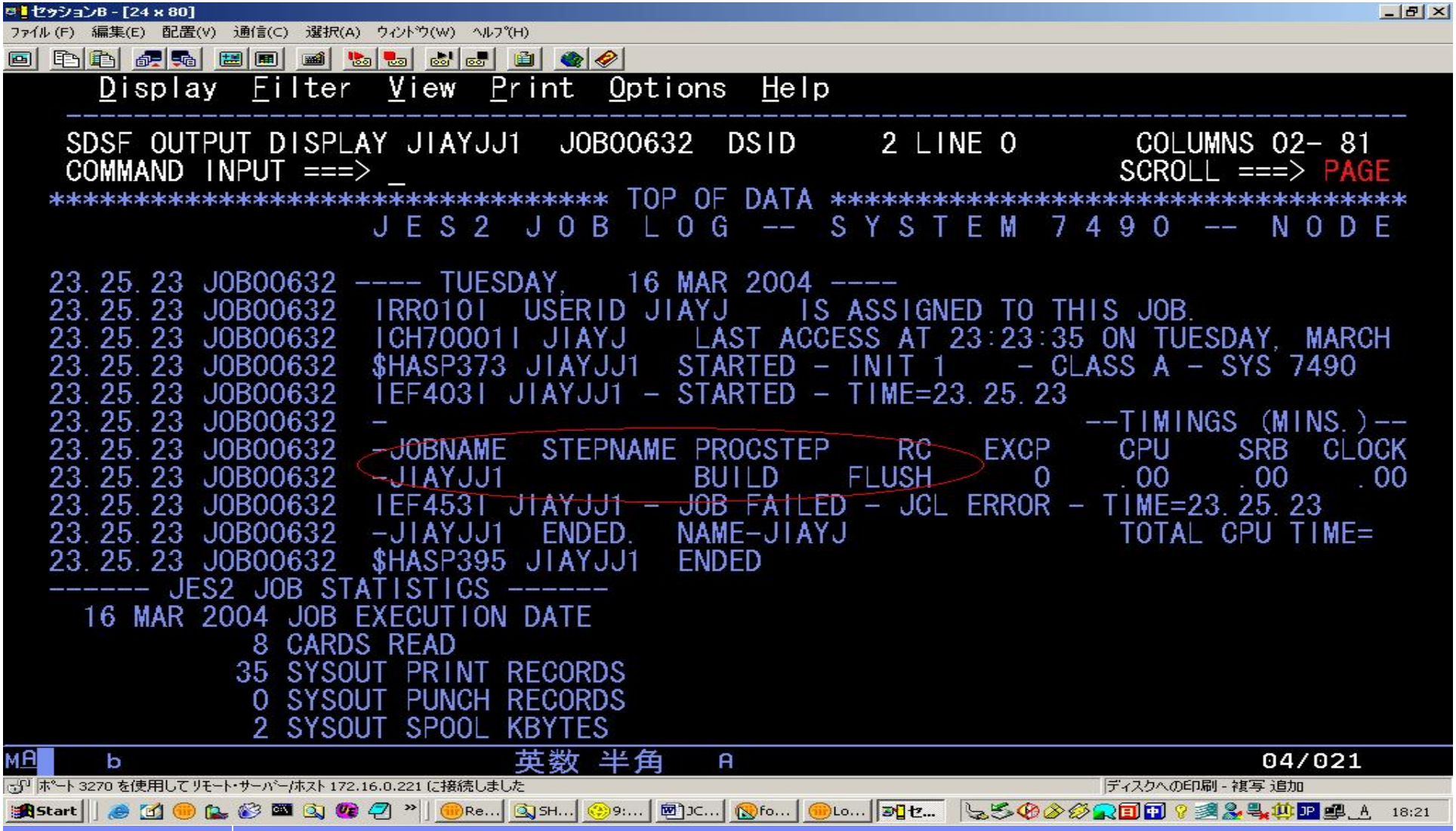
LEVEL	内容及对应
I (Information)	不用对应
W (Warning)	应当对应
E (Error)	必须对应
S (Severity)	必须对应
U (Unrecoverable)	必须对应

2. 调试

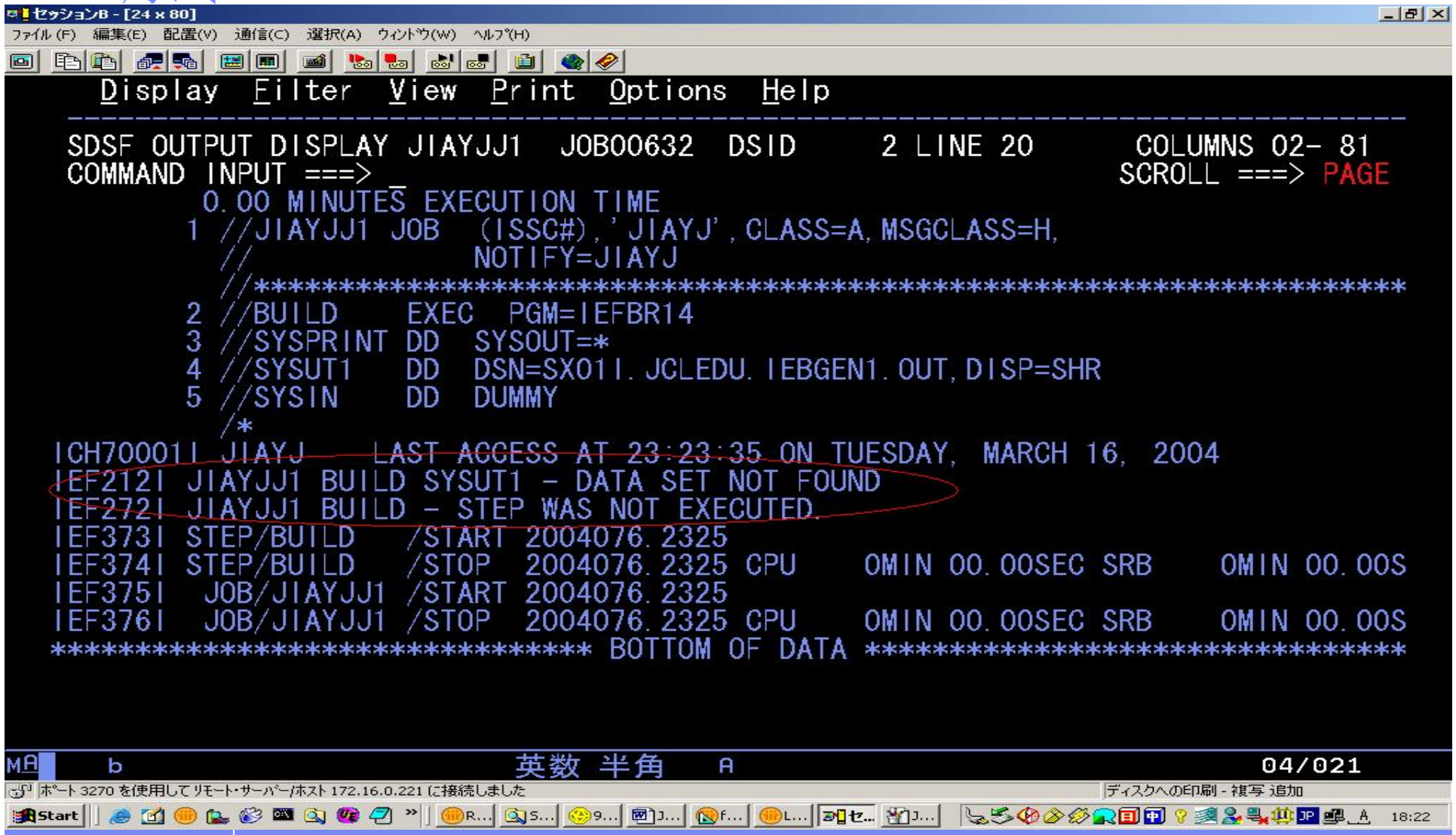
D.编译返回代码

代码	内容及对应
0	无错误
4	有警告,编译成功
8	有错误,编译成功
12	有严重的错误,编译不成功
16	有极其严重的错误,编译不成功

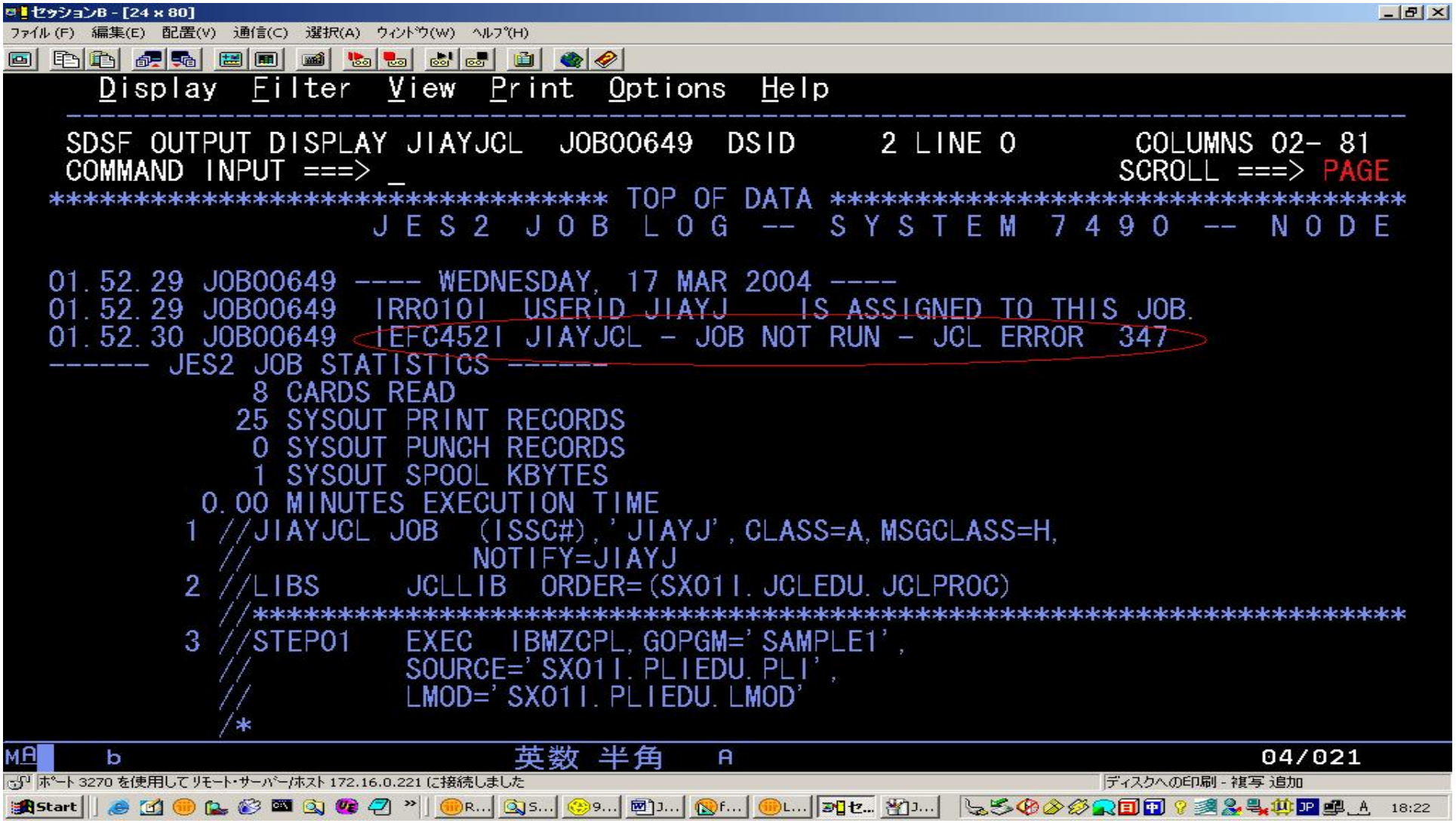
2. 调试 *E.JOB LOG*



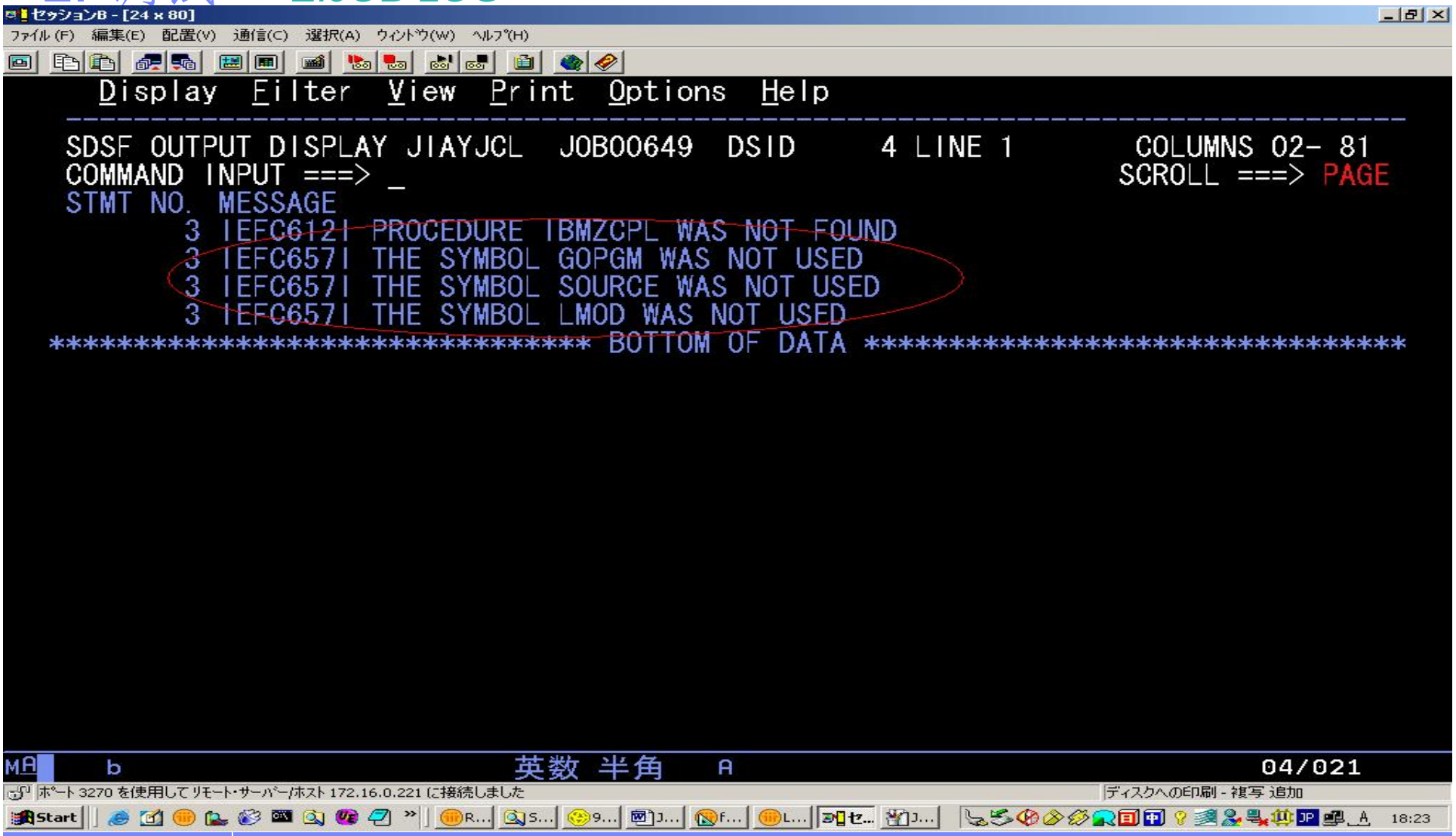
2. 调试 *E.JOB LOG*



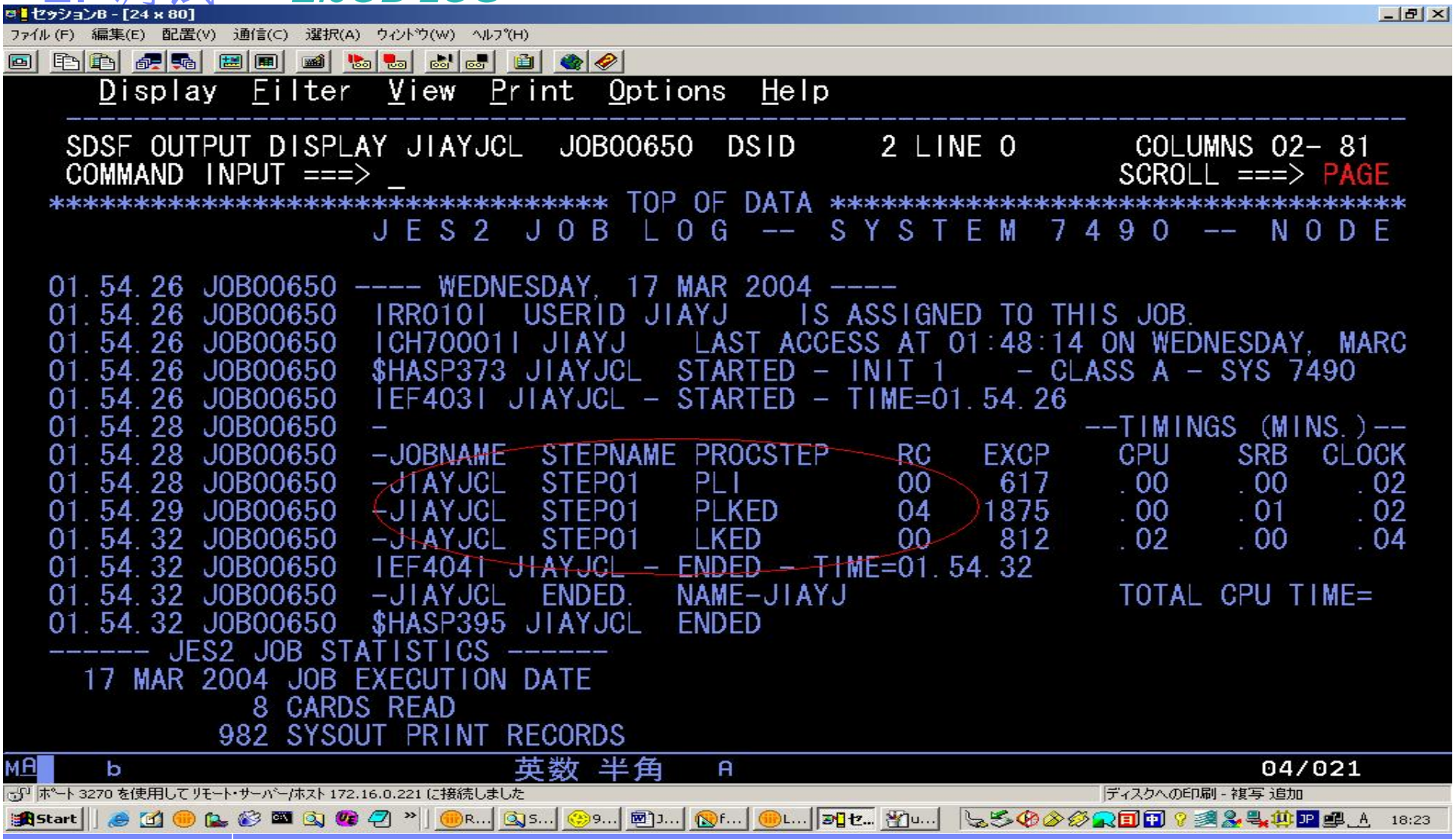
2. 调试 *E.JOB LOG*



2. 调试 *E.JOB LOG*

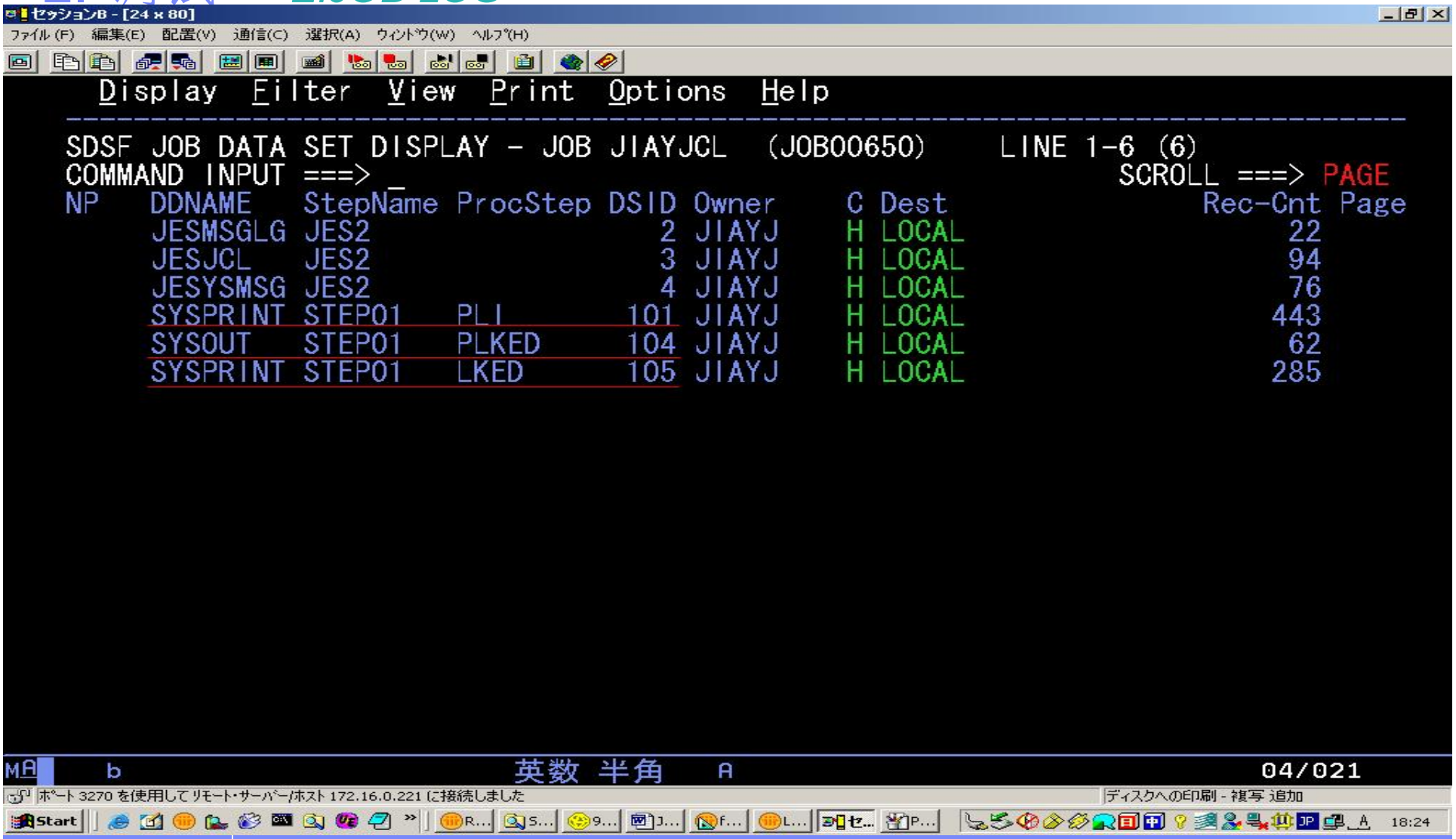


2. 调试 *E.JOB LOG*

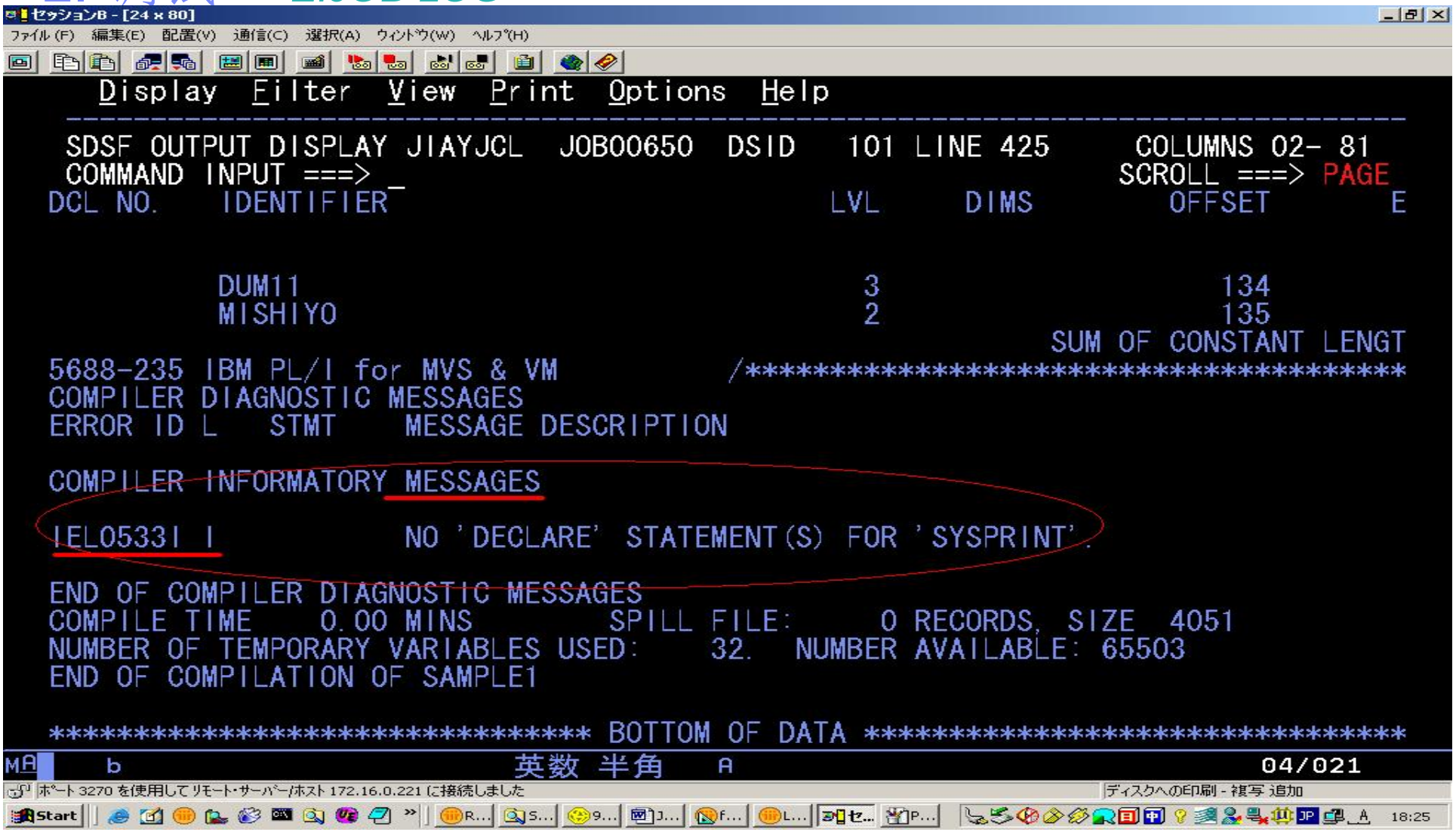


2. 调试 *E.JOB LOG*

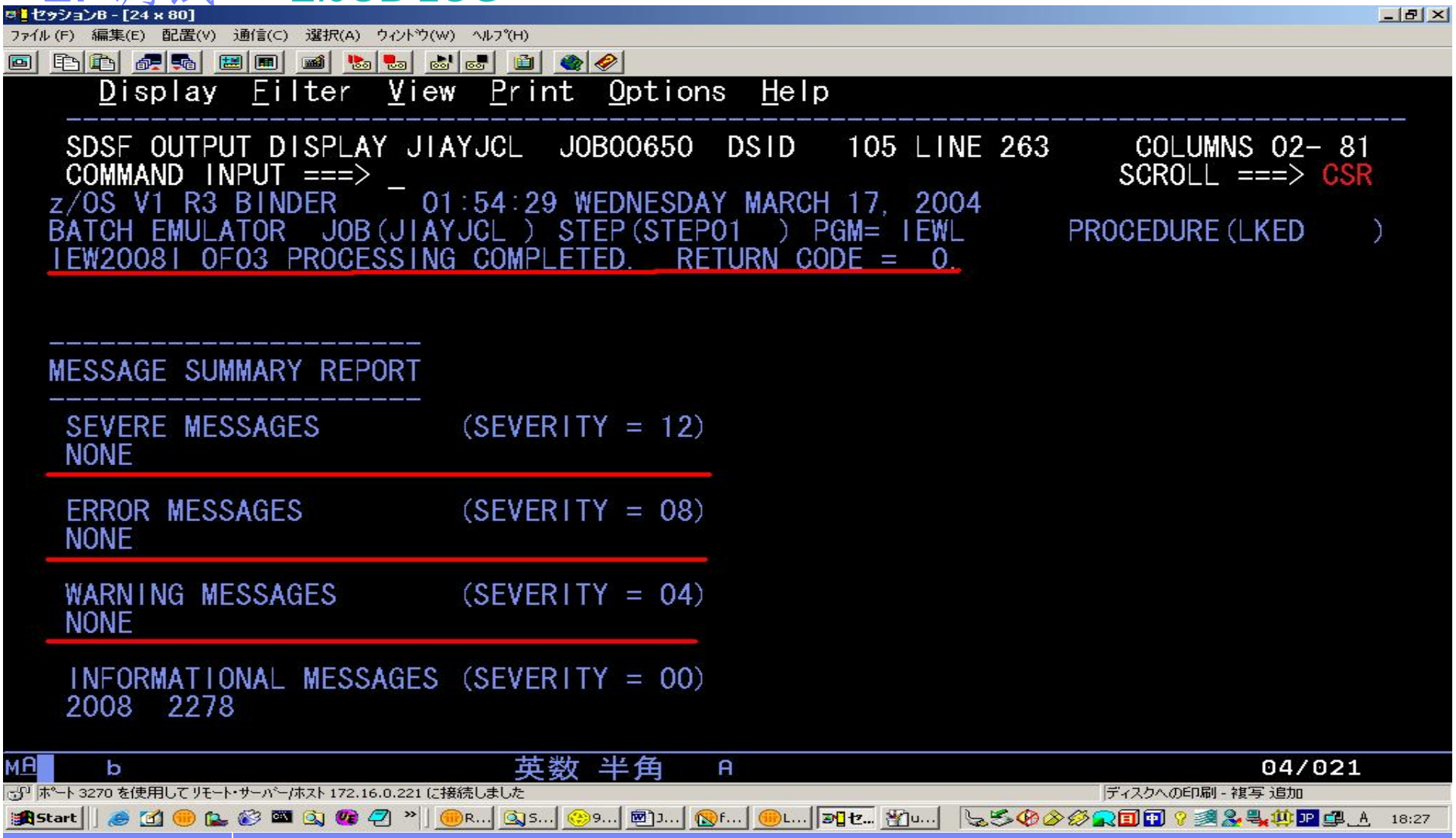
2. 调试 *E.JOB LOG*



2. 调试 *E.JOB LOG*



2. 调试 *E.JOB LOG*



2. 调试

F.编译时的CHECK LIST

- 编译的选项是否正确,是否有足够的编译信息
- 程序是否写在2—72列之间
- 变量是否有重名
- 所有的保留字是否正确
- 有没有忘记分号
- 引号是否匹配
- 括号是否匹配
- 注释是否用了正确的范围
- END的使用是否正确
- 对输入输出文件的READ/WRITE 是否正确
- 使用RECORD方式存取数据时,RECORD长度是否与物理长度一致
- 计算时使用的变量是否都是数字
- 变量等是否正确的初始化
- 没有明确定义的变量的属性是否满足处理的需要

练习二

输入的程序并编译执行



Table of contents

3. 程序流程

本章内容

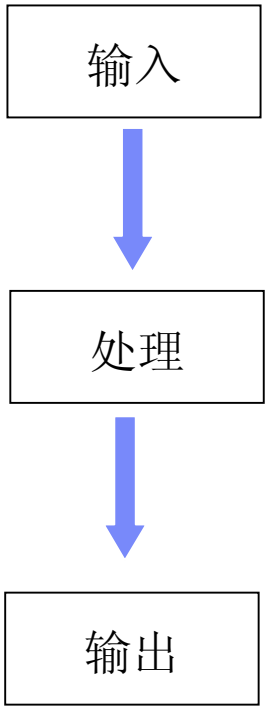
- 数据处理的基本方法
- 输出的处理

学习目标

能够完成简单的对文件的输入输出处理程序

3. 程序流程

A.数据处理的基本方法




```
OPEN FILE(FIN)   INPUT,
     FILE(FOUT) OUTPUT;

READ FILE(FIN)   INTO(IN_AREA);
```

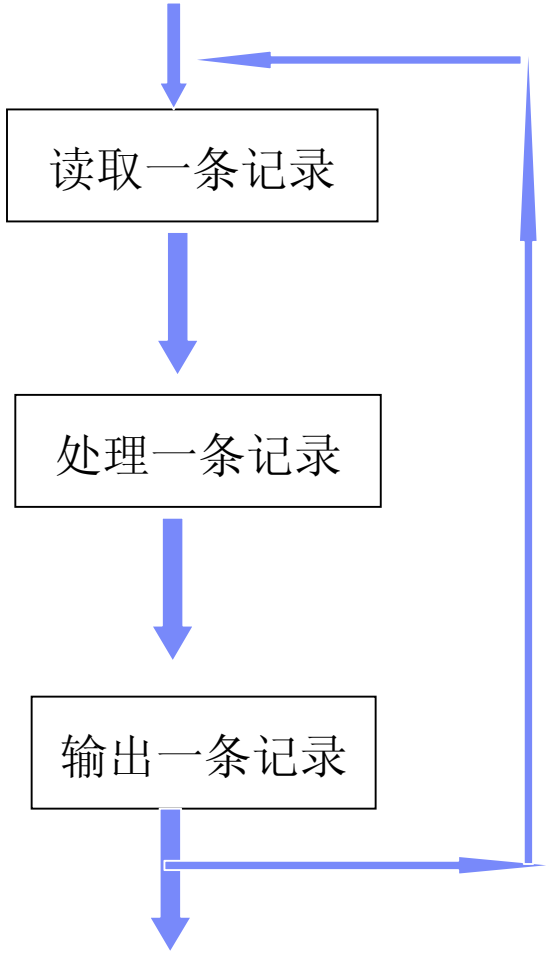
```
处理(计算,……)
```

```
WRITE FILE(FOUT)   FROM(OUT_AREA);

CLOSE FILE(FIN),
      FILE(FOUT);
```

3. 程序流程

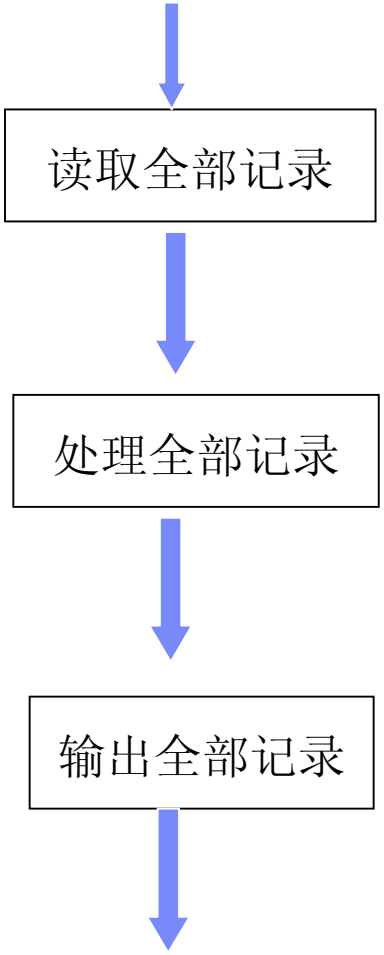
A.数据处理的基本方法-数据处理流程(1)




```
......
OPEN FILE(FIN)   INPUT,
     FILE(FOUT) OUTPUT;
ON ENDFILE(FIN) EOF = 1;
......
READ FILE(FIN)   INTO(IN_AREA);
DO WHILE (EOF = 0) ;
  ......
  WRITE FILE(FOUT)  FROM(OUT_AREA);
  READ FILE(FIN)   INTO(IN_AREA);
END;
CLOSE FILE(FIN),
      FILE(FOUT);
```

3. 程序流程

A.数据处理的基本方法-数据处理流程(2)



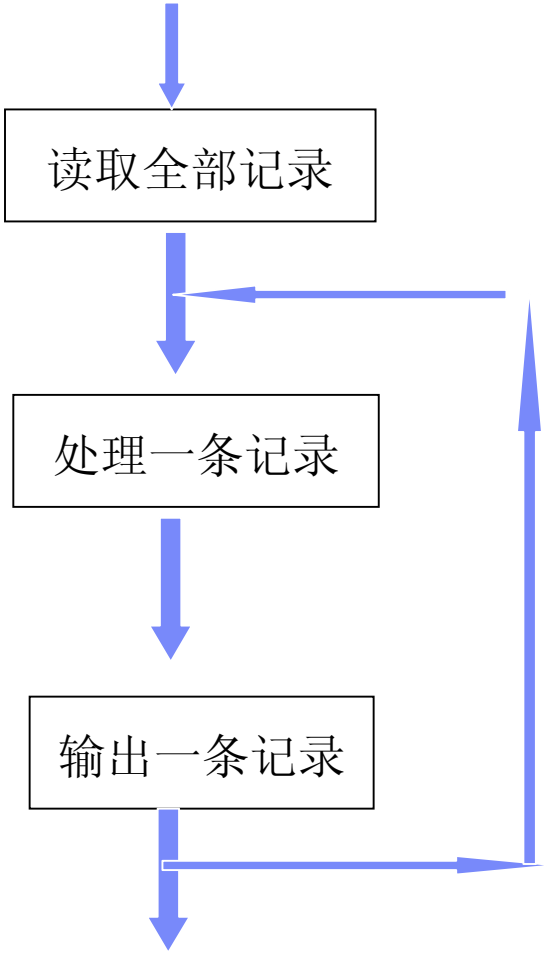

```
OPEN FILE(FIN)  INPUT;
ON ENDFILE(FIN) EOF = 1;
CNT1 = 0;
READ FILE(FIN)  INTO(IN_AREA);
DO WHILE (EOF = 0) ;
  ......
  CNT1 = CNT1 + 1;
  ......
  READ FILE(FIN)  INTO(IN_AREA);
END;
CLOSE FILE(FIN);
```

```
CNT2 = 0;
DO WHILE (CNT1 > 0) ;
  ......
  CNT1 = CNT1 - 1;
  ......
  CNT2 = CNT2 + 1;
END;
```

```
OPEN FILE(FOUT)  OUPUT;
DO WHILE (CNT2 > 0) ;
  ......
  CNT2 = CNT2 - 1;
  WRITE FILE(FOUT)  FROM(OUT_AREA);
END;
CLOSE FILE(FOUT);
```

3. 程序流程

A.数据处理的基本方法-数据处理流程(3)

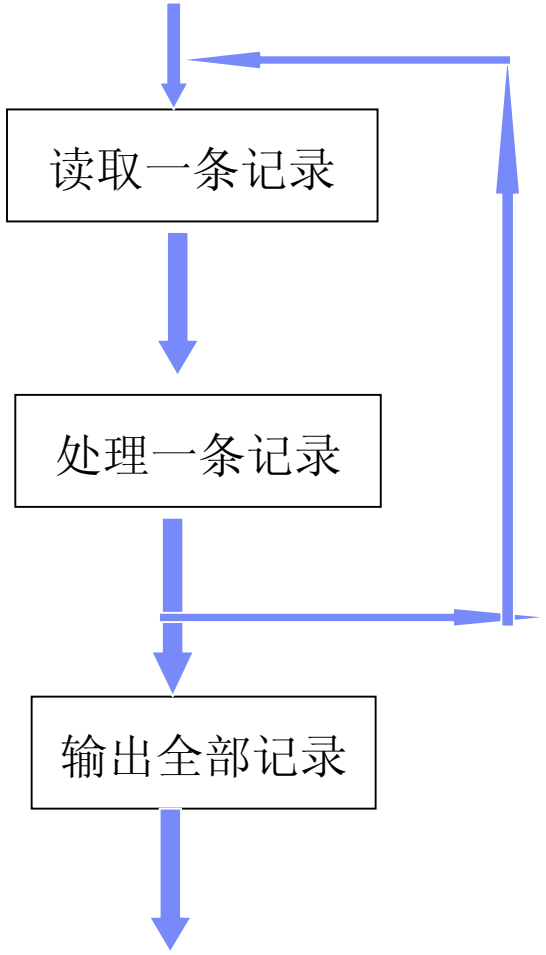



```
OPEN FILE(FIN)  INPUT;
ON ENDFILE(FIN) EOF = 1;
CNT1 = 0;
READ FILE(FIN)  INTO(IN_AREA);
DO WHILE (EOF = 0) ;
   ......
   CNT1 = CNT1 + 1;
   ......
   READ FILE(FIN)  INTO(IN_AREA);
END;
CLOSE FILE(FIN);
```

```
OPEN FILE(FOUT)  OUPUT;
DO WHILE (CNT1 > 0) ;
   ......
   CNT1 = CNT1 - 1;
   ......
   WRITE FILE(FOUT)  FROM(OUT_AREA);
END;
CLOSE FILE(FOUT);
```

3. 程序流程

A.数据处理的基本方法-数据处理流程(4)




```
OPEN FILE(FIN)  INPUT;
ON ENDFILE(FIN) EOF = 1;
CNT1 = 0;
READ FILE(FIN)  INTO(IN_AREA);
DO WHILE (EOF = 0) ;
   ......
   CNT1 = CNT1 + 1;
   ......
   READ FILE(FIN)  INTO(IN_AREA);
END;
CLOSE FILE(FIN);
```

```
OPEN FILE(FOUT)  OUPUT;
DO WHILE (CNT1 > 0) ;
   ......
   CNT1 = CNT1 - 1;
   WRITE FILE(FOUT)  FROM(OUT_AREA);
END;
CLOSE FILE(FOUT);
```

3. 程序流程

B.输出的处理

```
WRITE FILE(文件名) FROM (变量名);
```

ASA选项

通过给变量的第一字节赋上特定的CODE可以控制换行,换页等印刷选择,但这个字节本身并不会被打印出来.此时JCL中的文件格式定义需要加上参数A.

```
//       DCB=(RECFM=FBA,LRECL=167,BLKSIZE=0)
```

<CODE>

1	：换页开始打印
BLANK	：换行开始打印
O	：隔一行开始打印
-	：隔两行开始打印
+	：在同一行打印

3. 程序流程

B.输出的处理(续)

```
PUT FILE(文件名) [PAGE] LINE(n) [SKIP(n)] EDIT(变量名) (格式);
```

PAGE ：换页开始打印
LINE(n)：从第n行开始打印
SKIP(n)：空n行后开始打印

(例1)

```
PUT FILE(FLIST) PAGE LINE(34) EDIT (A,B,C) (A(2),F(3),A(3));
```

在新的一页的34行开始打印A,B,C

(例2)

```
PUT FILE(FLIST) EDIT(D,E,F) ( PAGE,LINE(10),COL(12),A(2),
                              SKIP(3),COL(12), F(3),
                              SKIP(2),COL(12), F(3) );
```

在新的一页的第10行的第12列打印D,
第13行的第12列打印E,
第15行的第12列打印F.

练习三

参照给定的设计文档资料资料,在自己的LIB下编写程序!

Table of contents

4. 编程技巧

本章内容

- 过程与函数
- 匹配的处理
- 结构化编程

学习目标

能够说明简单程序做成时应考虑的问题

4. 编程技巧

A.过程与函数 – 过程

<调用时的形式>

```
[ DCL  过程名    ENTRY ; ]                    <- 内部过程时可省略

 CALL  过程名 [ ( 参数1, 参数2, ……) ];
```

<定义的形式>

```
过程名 : PROC [ ( 参数1, 参数2, ……) ];
……
……
 END  过程名;
```

4. 编程技巧

A.过程与函数 – 过程(内部过程例)

```
PROG1A : PROC OPTIONS(MAIN);
  DCL
    ROUND       BUILITIN,
    NAMAE       CHAR(10),
    KOKUGO      FIXED DEC( 3),
    SANSUH      FIXED DEC( 3),
    SHAKAI      FIXED DEC( 3),
    RIKA        FIXED DEC( 3),
    EIGO        FIXED DEC( 3),
    GOHKEI      FIXED DEC( 3),
    HEIKIN      FIXED DEC( 5,1),
    EOF         FIXED DEC( 1) INIT(0);

    ON ENDFILE( SYSIN ) EOF = 1;
    GET SKIP EDIT ( NAMAE, KOKUGO, SANSUH, SHAKAI, RIKA, EIGO ) ( A(10), 5 F(5) );
    DO WHILE ( EOF = 0 );
       CALL KEISAN;
       PUT SKIP EDIT( NAMAE, GOHKEI, HEIKIN) ( A(10), X(2), F(3), F(5,1) );
       GET SKIP EDIT ( NAMAE, KOKUGO, SANSUH, SHAKAI, RIKA, EIGO ) ( A(10), 5 F(5) );
    END;

    KEISAN : PROC ;
      GOHKEI = KOKUGO + SANSUH + SHAKAI + RIKA + EIGO;
      HEIKIN = ROUND( GOHKEI/5, 1 );
    END KEISAN;
END PROG1A ;
```

4. 编程技巧

A.过程与函数 – 过程(外部过程例)

```
PROG2A : PROC OPTIONS(MAIN);
  DCL
    DEGCONV      ENTRY,
    CONVKEY      CHAR( 4),
    IN_DEG       FIXED DEC( 5),
    OUT_DEG      FIXED DEC( 5),
    EOF          FIXED DEC( 1) INIT(0);

    ON ENDFILE( SYSIN ) EOF = 1;
    GET SKIP EDIT ( CONVKEY, IN_DEG ) ( A( 4), F(5) );
    DO WHILE ( EOF = 0 );
       CALL DEGCONV( CONVKEY, IN_DEG, OUT_DEG);
       PUT LIST( CONVKEY, IN_DEG, 'à' , OUT_DEG);
       GET SKIP EDIT ( CONVKEY, IN_DEG ) ( A( 4), F(5) );
    END;
END PROG2A ;
*************************************************************************************************
DEGCONV : PROC(P1,P2,P3) ;
    DCL   P1   CHAR(4),
          (P2, P3) FIXED DEC( 5);

    IF P1 =  'FTOC'
    THEN P3 = ( 5 * ( P2  -  32 ) )/ 9;
    ELSE P3 = 9 * (P2 / 5) + 32;

END DEGCONV;
```

4. 编程技巧

A.过程与函数 – 函数

<调用时的形式>

```
[ DCL  函数名    ENTRY  RETURNS (返回值属性); ]   <- 内部函数可省略

 变量  =   函数名 [ ( 参数1, 参数2, ……) ];
```

<定义的形式>

```
函数名 : PROC [ ( 参数1, 参数2, ……) ]  RETURNS (返回值属性);
……
……
   RETURN  (值);
END  函数名;
```

4. 编程技巧

A.过程与函数 – 函数(内部函数例)

```
PROG1B : PROC OPTIONS(MAIN);
  DCL
    TEIHEN      FIXED DEC( 5),
    TAKASA      FIXED DEC( 5),
    MENSEKI     FIXED DEC( 9);

    GET LIST ( TEIHEN, TAKASA );
    MENSEKI = TRIAREA( TEIHEN, TAKASA );
    PUT LIST ( MENSEKI );

    TRIAREA : PROC(P1, P2) RETURNS( FIXED DEC(9));
      DCL (P1, P2)  FIXED DEC( 5);
      RETURN ( (P1 * P2) / 2);
    END TRIAREA;
END PROG1B ;
```

4. 编程技巧

A.过程与函数 – 函数(外部函数例)

```
PROG2B : PROC OPTIONS(MAIN);
  DCL
    ONSOKU       ENTRY(FIXED DEC(3) ) EXTERNAL
                 RETURNS FIXED DEC( 5,1)),
    DEGREEC      FIXED DEC( 3);

    GET LIST ( DEGREEC );
    PUT LIST ( DEGREEC, ONSOKU( DEGREEC ) );
END PROG2B ;
*******************************************************************************
ONSOKU : PROC(P1) RETURNS( FIXED DEC(5,1));
      DCL P1  FIXED DEC( 3);
      RETURN ( 331.5 + 0.6 * P1);
END ONSOKU;
```

4. 编程技巧

A.过程与函数 – 系统函数

系统函数的属性定义为BUILTIN,使用方法和自定义函数一样.

SUBSTR (X, Y, [Z])

返回由Y和Z指定的X中的SUBSTRING. X是要截取的字符串,Y是开始位置,Z是长度. Z省略时,截取到X的最后一个字符.

```
(例)  DCL   SUBSTR  BUILTIN ,
            A       CHAR(10)  INIT('ABCDEFGHIJ'),
            B       CHAR( 4);
      B = SUBSTR( A, 5, 4) ;              -> B='EFGH'
```

DATE

返回yymmdd格式的6位字符串.yy为年的后两位,mm为月,dd为日

```
(例)  DCL  (DATE, SUBSTR ) BUILTIN ,
           TODAY    CHAR( 6),
           PDATE    CHAR( 8);
      TODAY = DATE;                  <- '040213'
      PDATE = SUBSTR(TODAY,1,2) || '/' || SUBSTR(TODAY,3,2)
                || '/' ||SUBSTR(TODAY,5,2)                  -> '04/02/13'
```

4. 编程技巧

A.过程与函数 – 系统函数(续)

LENGTH (X)

返回X的长度.

```
(例1)  DCL   LENGTH  BUILTIN ,
             A       CHAR(5)  INIT('ABC'),
             B       FIXED BIN(15);
       B = LENGTH( A ) ;              -> B=5
(例2)  DCL   LENGTH  BUILTIN ,
             A       CHAR(5)  VARYING INIT('ABC'),
             B       FIXED BIN(15);
       B = LENGTH( A ) ;              -> B=3
```

SUM (X)

返回数组的合计值

```
(例)  DCL  SUM BUILTIN ,
           A(10)  BIN FIXED(15);
      DO I = 1 TO 10;
        A(I) = I;
      END;
      B = SUM(A);       -> 55
```

4. 编程技巧

A.过程与函数 – 系统函数(续)

MAX (X1, X2,……Xn)

返回参数里最大的值.

```
(例1)  DCL  MAX      BUILTIN ,
            A        FIXED BIN(15);
       A = MAX( 25, 30, 11 ) ;          -> A=30
```

MIN-返回参数里最小的值.

MOD (X, Y)

返回X/Y的余数

```
(例)  DCL  MOD BUILTIN ,
           A   BIN FIXED(15) INIT (10),
           B   BIN FIXED(15) INIT ( 3),
           C   BIN FIXED(15) ;
      C = MOD( A, B);   -> 1
```

4. 编程技巧

B.匹配的处理

匹配的处理在下列情况时会用到:

- 几个相同格式的文件合并成一个文件
- 对某一个文件,根据另外的文件的内容对它进行追加,变更,删除等处理
- 2个文件的关键字一致时,对两个文件的内容进行处理,生成新的文件

匹配处理有两种基本形式:

- 1对1的匹配
- 1对多的匹配

系统函数 – HIGH 和 LOW

- HIGH(x) : PL/I 中的最大值
- LOW(x) : PL/I 中的最小值

使用目的:

匹配处理时文件结束时控制用

4. 编程技巧

C.结构化编程

- SELECT – 多重分支
- CALL 语句 – 内部顺序的控制
- DO GROUP – 使循环处理变的可能

```
SELECT (变量名);
  WHEN (条件1的值) 处理1;
  WHEN (条件2的值) 处理2;
  WHEN (条件3的值) 处理3;
  WHEN (条件4的值) 处理4;
  ……
  WHEN (条件n值)   处理n;
  [OTHERWISE       处理n+1;]
END;
```

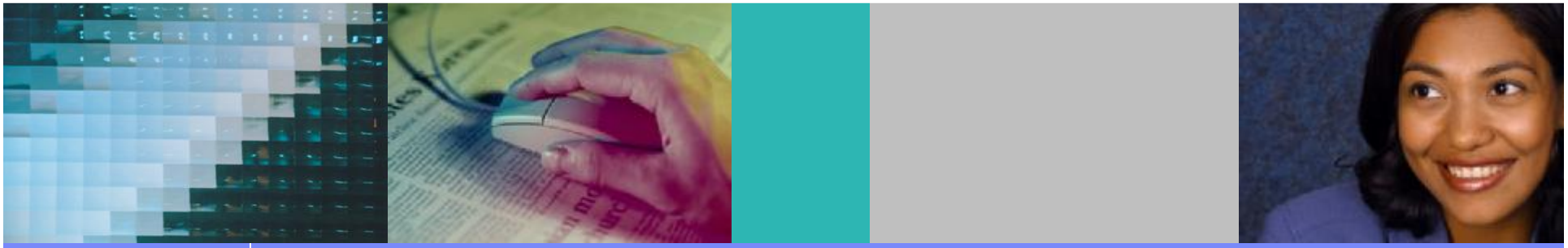
4. 编程技巧

C.结构化编程(续)

- 一个PROC尽可能控制在一页之内
- 使用缩进格式
- 属性定义格式的对齐
- INITIAL部分如果长的话另起一行书写
- 语句的范围使用END来明确
- 文件名,数据名,LABEL,过程名等使用表示内容的名字

练习四

参照给定的设计文档资料资料,在自己的LIB下编写程序!



Appendix - Coding SQL in PLI

Basics of coding SQL

Delimiting an SQL statement

For languages other than REXX, bracket an SQL statement in your program between EXEC SQL and a statement terminator. The terminators for the languages described in this book are:

Language	SQL Statement Terminator
Assembler	End of line or end of last continued line
C	Semicolon (;)
COBOL	END-EXEC
FORTRAN	End of line or end of last continued line
PL/I	Semicolon (;)

For REXX, precede the statement with EXECSQL. If the statement is in a literal string, enclose it in single or double quotation marks.

For example, use EXEC SQL and END-EXEC to delimit an SQL statement in a COBOL program:

```
EXEC SQL
an SQL statement
END-EXEC.
```

Accessing data using host variables

```
EXEC SQL
    SELECT EMPNO, LASTNAME, WORKDEPT
    INTO :CBLEMPNO, :CBLNAME, :CBLDEPT
    FROM DSN8710.EMP
    WHERE EMPNO = :EMPID ;

EXEC SQL
    UPDATE DSN8710.EMP
    SET PHONENO = :NEWPHONE
    WHERE EMPNO = :EMPID;
```

Accessing data using a host structure

```
EXEC SQL
    SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
    INTO :EMPNO, :FIRSTNME, :MIDINIT, :LASTNAME, :WORKDEPT
    FROM DSN8710.VEMP
    WHERE EMPNO = :EMPID;
```

If you want to avoid listing host variables, you can substitute the name of a structure, say :PEMP, that contains :EMPNO, :FIRSTNME, :MIDINIT, :LASTNAME, and :WORKDEPT. The example then reads:

```
EXEC SQL
   SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
   INTO :PEMP
   FROM DSN8710.VEMP
   WHERE EMPNO = :EMPID;
```

Checking the execution of SQL statements

A program that includes SQL statements needs to have an area set apart for communication with DB2 — an *SQL communication area* (SQLCA). When DB2 processes an SQL statement in your program, it places return codes in the SQLCODE and SQLSTATE host variables or corresponding fields of the SQLCA. The return codes indicate whether the statement executed succeeded or failed.

SQLCODE: DB2 returns the following codes in SQLCODE:

- If SQLCODE = 0, execution was successful.
- If SQLCODE > 0, execution was successful with a warning.
- If SQLCODE < 0, execution was not successful.

SQLCODE 100 indicates no data was found.

Checking the execution of SQL statements

The WHENEVER statement

EXEC SQL
WHENEVER *condition action;*

Condition is one of these three values:
SQLWARNING
SQLERROR
NOT FOUND

Action is one of these two values:
CONTINUE
GOTO or GO TO *host-label*

Using a cursor to retrieve a set of rows

The basic steps in using a cursor are:

1. Execute a DECLARE CURSOR statement to define the result table on which the cursor operates. See “Step 1: Declare the cursor”.

2. Execute an OPEN CURSOR to make the cursor available to the application. See “Step 2: Open the cursor” .

3. Specify what the program does when all rows have been retrieved. See “Step 3:Specify what to do at end-of-data”.

4. Execute multiple SQL statements to retrieve data from the table or modify selected rows of the table. See “Step 4: Execute SQL statements”.

5. Execute a CLOSE CURSOR statement to make the cursor unavailable to the application. “Step 5: Close the cursor”

Using a cursor to retrieve a set of rows

Step 1: Declare the cursor

```
EXEC SQL
    DECLARE C2 CURSOR FOR
    SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, SALARY
    FROM DSN8710.EMP X
    WHERE EXISTS
        (SELECT *
         FROM DSN8710.PROJ Y
         WHERE X.EMPNO=Y.RESPEMP
         AND Y.PROJNO=:GOODPROJ);
```

Using a cursor to retrieve a set of rows

Step 2: Open the cursor

```
EXEC SQLEXEC SQL
OPEN C2;
```

Step 3: Specify what to do at end-of-data

```
IF SQLCODE = 100 GO TO DATA-NOT-FOUND;

EXEC SQL
WHENEVER NOT FOUND GO TO DATA-NOT-FOUND;
```

Using a cursor to retrieve a set of rows

Step 4: Execute SQL statements

```
EXEC SQL
FETCH C1 INTO
:HV-EMPNO, :HV-FIRSTNME, :HV-MIDINIT, :HV-LASTNAME, :HV-SALARY;


UPDATE DSN8710.EMP
SET SALARY = 50000
WHERE CURRENT OF C1;


EXEC SQL
DELETE FROM DSN8710.EMP
WHERE CURRENT OF C1;
```

Using a cursor to retrieve a set of rows

Step 5: Close the cursor

```
EXEC SQL
CLOSE C1;
```

Coding SQL statements in a PLI application

Defining the SQL communication area

Defining SQL descriptor areas

Embedding SQL statements

Using host variables & host structures

Determining equivalent SQL and PLI data types

Determining compatibility of SQL and PLI data types

Using indicator variables

Handling SQL error return codes

Developing your application

"Step 1: Process SQL statements"

- One step in preparing an SQL application to run is processing SQL statements in the program. The SQL statements must be replaced with calls to DB2 language interface modules, and a DBRM must be created.

"Step 2: Compile (or assemble) and link-edit the application"

- If you use the DB2 precompiler, your next step in the program preparation process is to compile and link-edit your program.

"Step 3: Bind the application"

- You must bind the DBRM produced by the SQL statement processor to a plan or package before your DB2 application can run. A plan can contain DBRMs, a package list specifying packages or collections of packages, or a combination of DBRMs and a package list. The plan must contain at least one package or at least one directly-bound DBRM. Each package you bind can contain only one DBRM.

"Step 4: Run the application"

- After you have completed all the previous steps, you are ready to run your application.

Any Existing Process Could Be Improved!

Thanks Very Much!