



# 中华人民共和国国家标准

GB/T 4092.4—1992

---

## 程序设计语言 COBOL 顺序 I-O 模块

Programming language COBOL  
Sequential I-O module

1992-08-04 发布

1993-05-01 实施

---

国家技术监督局 发布

程序设计语言 COBOL  
顺序 I-O 模块

GB/T 4092.4—1992

代替 GB 4092.4—83

Programming language COBOL  
Sequential I-O module

## 1 引言

### 1.1 功能

顺序 I-O 模块提供按建立的顺序存取文卷记录的功能。这个顺序是由把记录写到文卷上而建立的。

### 1.2 级别特征

对文卷控制款、文卷描述款及 I-O-CONTROL 段中的各款，1 级顺序 I-O 提供局部功能。在过程中对 CLOSE、OPEN、READ、USE 和 WRITE 语句，1 级顺序 I-O 提供了局部功能，面对 READ 和 REWRITE 语句提供了完整功能。

2 级顺序 I-O 对文卷控制款、文卷描述款及 I-O-CONTROL 段中各款，提供了完整功能。在过程级中，2 级顺序 I-O 对 CLOSE、OPEN、READ、REWRITE、USE 和 WRITE 语句提供了完整功能。

### 1.3 语言概念

#### 1.3.1 组织

顺序文卷是这样组织的，文卷中的每一个记录除最后一个记录外，有唯一的后继记录；并且除第一个记录外，每个记录有唯一的先行记录，这些先行后继关系是在文卷建立时按 WRITE 语句的次序建立的。一旦建立了这种先行后继关系，便不能改变。

顺序组织的大容量存储文卷与任意顺序媒体上的文卷具有相同的逻辑结构；但是，一个顺序大容量存储文卷可能会适当地更改。当使用这种技术时，不能往文卷添加新记录且每个被替换的记录须与原记录长度相同。

#### 1.3.2 存取方式

在顺序存取方式中，存取记录的次序就是原来将记录写到文卷上的次序。

#### 1.3.3 当前卷指针

当前卷指针是在本标准中为了便于确切指明顺序文卷当前的物理卷而使用的概念实体。当前卷指针的状态受到 CLOSE、OPEN、READ 和 WRITE 诸语句的影响。

#### 1.3.4 文卷位置指示符

文卷位置指示符是在本标准中为了便于在某一输入输出操作中确切指明给定的文卷中要存取的一个记录而使用的概念实体。文卷位置指示符的所置值仅受 CLOSE、OPEN 和 READ 语句的影响。对于按输出方式打开的文卷，文卷位置指示符的概念是无意义的。

#### 1.3.5 I-O 状态

I-O 状态是两字符的概念实体，赋给它的值指明了 CLOSE、OPEN、READ、REWRITE 或 WRITE 语句执行期间的状态，指明了与该 I-O 语句相联系的任一命令语句执行之前的状态，或任一可用的 USE AFTER STANDARD EXCEPTION 过程执行之前的状态。在 COBOL 程序中，通过文卷的文卷

控制款中使用 **FILE STATUS** 子句来使用 **I-O** 状态的值。

**I-O** 状态还决定是否执行一可用的 **USE AFTER STANDARD EXCEPTION** 过程。如果出现的条件不是标题为“成功的结束”之下包含的那些条件,则这样的过程就根据其它地方叙述的规则执行。如果出现的条件是在标题为“成功的结束”之下包含的那些条件,则不执行这样的过程(见 4.6**USE** 语句)。

某些类的 **I-O** 状态值指出关键错误条件。它们是:以数字 3 或 4 和任何由实现者作为关键定义的以数字 9 开始的错误条件。如果输入输出操作的 **I-O** 状态值指明了这样的错误条件,那么实现者确定在执行任何可用的 **USE AFTER STANDARD EXCEPTION** 过程之后所要采取的动作;若没有可用的上述过程,则实现者确定输入输出控制系统错误标准处理之后采取什么动作。

**I-O** 状态根据输入输出操作完成的情况表示下列条件之一:

(1) 成功的结束。成功地执行了输入输出语句。

(2) 到末端。由于满足了末端条件,顺序的 **READ** 语句执行不成功。

(3) 永久性错误。由于发生了阻止文卷的进一步处理的错误,输入输出语句执行不成功。执行任何指定的例外过程。如果设有调用实现者定义的技术来改正永久性错误条件,则在余下的输入输出操作中该条件一直有效。

(4) 逻辑错误。由于对文卷执行了一系列不适当的输入输出操作或违反了用户定义的限制而使输入输出语句的执行不成功。

(5) 实现者定义。由于满足了实现者定义的某条件,致使输入输出语句执行不成功。

下面列出上述条件的 **I-O** 状态值,这些条件是在对一顺序文卷进行输入输出操作之后产生的。如果不只一个值适用,由实现者确定往 **I-O** 状态中置入哪一个可用的值。

(1) 成功的结束

a. **I-O** 状态=00。成功地执行了输入输出语句,对有关输入输出操作没有进一步的信息可用。

b. **I-O** 状态=04。成功地执行了 **READ** 语句,但当前处理的记录其长度与相应文卷的固有属性不一致。

c. **I-O** 状态=05。成功在执行了 **OPEN** 语句,但在 **OPEN** 执行时,引用到的任选文卷没有出现。如果打开方式是 **I-O** 或延伸方式,则文卷已被建立。

d. **I-O** 状态=07。成功地执行了输入输出语句,可是,对于带有 **REEL/UNIT** 的 **CLOSE** 语句,被引用的文卷位于非卷/单元媒体上。

(2) 不成功结束的末端条件

a. **I-O** 状态=10。顺序 **READ** 语句正在执行时,由于下列原因下一个逻辑记录不存在:

1) 已达到文卷末端;或

3) 不成功结束的永久性错误条件

a. **I-O** 状态=30。存在一永久性错误,并且没有和输入输出操作有关的进一步可用信息。

b. **I-O** 状态=34。由于边界不合法而出现的永久性错误;试图在顺序文卷的外部定义边界之外写内容。实现者指定这些边界的定义方式。

c. **I-O** 状态=35。如果对一 **REEL/UNIT** 文卷执行带有 **INPUT**、**I-O** 短语的 **OPEN** 语句,而该文卷不存在,则出现一永久性错误。

d. **I-O** 状态=37。由于 **OPEN** 语句要对文卷操作,而该文卷不支持 **OPEN** 语句中指定的打开方式,故产生永久性错误。可能的出错情形是:

1) 指定了 **OUTPUT** 短语但文卷不支持写操作。

2) 指定了 **I-O** 短语但文卷并不支持以 **I-O** 方式打开的顺序文卷所允许的输入输出操作。

3) 指定了 **INPUT** 短语, 但文卷不支持读操作。

f. **I-O 状态=39**。由于在固有文卷属性和程序中为文卷规定的属性之间检测出相互冲突的地方, 故 **OPEN** 不成功。

(4) 不成功结束的逻辑错误条件

a. **I-O 状态=41**。试图对在打开方式下的文卷执行 **OPEN** 语句。

b. **I-O 状态=42**。试图对在不在打开方式下的文卷执行 **CLOSE** 语句。

c. **I-O 状态=43**。对在顺序存取方式下的大容量存储文卷, 在执行 **REWRITE** 语句之前为相关文卷执行的最后一条输入输出语句不是成功执行的 **READ** 语句。

d. **I-O 状态=44**。由于下列原因出现边界不合法:

1) 试图与或里与的记录, 其长度比相关文卷名的 **RECORD IS VARYING** 子句中允许的最大长度要大, 或比允许的最小长度要小。

2) 试图对一顺序文卷重写记录, 而该记录长度与被替代的长度不等。

e. **I-O 状态=46**。试图对以输入输出方式或 **I-O** 方式打开的文卷执行顺序 **READ** 语句, 并且由于下列原因还未建立有效的下一记录:

1) 前面的 **READ** 语句不成功但并不产生末端条件, 或

2) 前面的 **READ** 语句产生末端条件。

f. **I-O 状态=47**。试图对不是以输入方式或 **I-O** 方式打开的文卷执行 **READ** 语句。

g. **I-O 状态=48**。试图对不是以输出方式或 **I-O** 方式打开的文卷执行 **WRITE** 语句。

h. **I-O 状态=49**。试图对不是以 **I-O** 方式打开的文卷执行 **REWRITE** 语句;

(5) 实现者定义的不成功结束条件

a. **I-O 状态=9x**。存在实现者定义条件。该条件不能与 **I-O** 状态值 00 到 49 的任何一个条件重复。  
**x** 的值由实现者定义。

### 1.3.6 末端条件

末端条件能作为执行 **READ** 语句的结果出现 (见 4.4**READ** 语句)。

### 1.3.7 文卷属性冲突条件

执行一条 **OPEN**、**REWRITE** 或 **WRITE** 语句之后, 可能会产生文卷属性冲突条件。当文卷属性冲突条件发生时, 则识别出这个条件的输入输出语句执行不成功而文卷并不受影响 (见 4.3**OPEN** 语句; 4.5**REWRITE** 语句以及 4.7**WRITE** 语句)。

识别出文卷属性冲突条件之后, 这些动作按下列顺序进行:

(1) 对与文卷名相关联的 **I-O** 状态赋值以指明文卷属性冲突条件 (见 1.3.5**I-O** 状态)。

(2) 执行与文卷名相关联的 **USE AFTER EXCEPTION** 过程 (如果有的话)。

### 1.3.8 专用寄存器 **LINAGE-COUNTER**

保留字 **LINAGE-COUNTER** 是在文卷描述款中出现 **LINAGE** 子句时, 所产生的一个行数计数器的名 (见 3.7**LINAGE** 子句)。隐含描述一个无正负号整数, 它的大小等于 **LINAGE** 子句中整数的大小或由数据名 1 引用的数据项的大小。**LINAGE-COUNTER** 只能在过程部的语句中引用; 可是只有输入输出控制系统才能改变 **LINAGE-COUNTER** 的值。

## 2 顺序 **I-O** 模块的环境部

### 2.1 输入输出节

输入输出节位于源程序的环境部中。输入输出节涉及的信息是控制外部媒体与目标程序之间的数据传送和数据处理所需的信息。输入输出节在 **COBOL** 源程序的环境部中是任选的。

输入输出节的一般格式如下所示：

**INPUT-OUTPUT SECTION.**

**FILE-CONTROL.** {文卷控制款} ...

## 2.2 FILE-CONTROL 段

### 2.2.1 功能

**FILE-CONTROL** 段允许指明和文卷相关联的信息。

### 2.2.2 一般格式

**FILE-CONTROL.** {文卷控制款} ...

## 2.3 文卷控制款

### 2.3.1 功能

文卷控制款描述顺序文卷的有关物理属性。

### 2.3.2 一般格式

**SELECT** {**ORGANIZATION**} 文卷名1 **ASSIGN TO**

{实现名1}  
{字值1} .. **RESERVE** 整数1  
[**AREA**]  
[**AREAS**]

[ [**ORGANIZATION IS**] **SEQUENTIAL** ]

**PADDING CHARACTER IS** {数据名1}  
{字值2} ]  
**RECORD DELIMITER IS** {**STANDARD-1**}  
{实现名2}

[ **ACCESS MODE IS** **SEQUENTIAL** ]

[ **FILE STATUS IS** 数据名2 ]

### 2.3.3 语法规则

(1) 在文卷控制款中必须首先指出 **SELECT** 子句，接在 **SELECT** 子句后的各个子句可以按任何次序出现。

(2) 在数据部中描述的每一个文卷名在 **FILE-CONTROL** 段中必须且只能命名一次。在 **SELECT** 子句中描述的每一个文卷名在数据部中必须有一个文卷描述款。

(3) 字值1 必须是非数值字值且不能是象征常量。实现名1 允许的内容的含义和规则以及字值1 的值由实现者定义。

### 2.3.4 一般规则

(1) 如果文卷名1 引用的文卷连接符是外部文卷连接符（见 GB/T 4092.12 中 4.5 **EXTERNAL** 子句），则运行单位中引用该文卷连接符的所有文卷控制款必须：

b. 对 **ASSIGN** 子句中实现名1 或字值1 具有一致的说明。实现者对实现名1 或字值1 指定一致规则。

• 对 **RECORD DELIMITER** 子句中的实现名2 具有一致说明。实现者指定实现名2 的一致规则。  
对 **RESERVE** 子句中的整数1 有同样的值

e. 同样的组织。

f. 同样的存取方式。

g. 对 **PADDING CHARACTER** 子句有同样说明。

(2) **OPTIONAL** 短语只对以输入、I-O 或扩展方式打开的文卷适用。对于目标程序每一次运行时并不都要用到的文卷来说，该短语是必需的。

(3) **ASSIGN** 子句指明了文卷名 1 引用的文卷与实现名 1 或字值 1 引用的存储媒体之间的联系。

(4) **ACCESS MODE** 子句、**FILE STATUS** 子句、**ORGANIZATION IS SEQUENTIAL** 子句

**PADDING CHARACTER** 子句、**FILE STATUS** 子句、**ORGANIZATION IS SEQUENTIAL** 子句按字母顺序依次列出。

## 2.4 **ACCESS MODE** 子句

### 2.4.1 功能

**ACCESS MODE** 子句指明文卷中记录的存取次序。

### 2.4.2 一般格式

**ACCESS MODE IS SEQUENTIAL**

### 2.4.3 一般规则

(1) 如果未指定 **ACCESS MODE** 子句，就假定是顺序方式。

(2) 由文卷组织指出文卷中记录的存取顺序。对于顺序文卷而言，这个顺序是由先行后继记录关系所确定的，而该关系又是在文卷产生时由 **WRITE** 语句所建立的。

(3) 如果相关联的文卷连接符是外部文卷连接符，则与运行单位中该文卷连接符相关的每个文卷控制描述款必须指定为同一存取方式。

## 2.5 **FILE STATUS** 子句

### 2.5.1 功能

**FILE STATUS** 子句指定一个包含输入输出状态的数据项。

### 2.5.2 一般格式

**FILE STATUS IS** 数据名 1

### 2.5.3 语法规则

(1) 数据名 1 可以受限。

(2) 数据名 1 必须在数据部中定义为字符类型的两字符数据项，且不能在 **FILE** 节、**REPORT** 节或 **COMMUNICATION** 节中定义。

### 2.5.4 一般规则

(1) 如果指定了 **FILE STATUS** 子句，则一旦 I-O 状态改变，数据名 1 引用的数据项之值也跟着改变以包含 I-O 状态之值。该值指明了该语句执行之后的状态（见 1.3.5 I-O 状态）。

(2) 数据名 1 引用的数据项在执行一条输入输出语句时改变，它是在该语句有关的文卷控制描述款中指定的。

## 2.6 **ORGANIZATION IS SEQUENTIAL** 子句

### 2.6.1 功能

**ORGANIZATION IS SEQUENTIAL** 子句把文卷的逻辑结构指定为顺序组织。

### 2.6.2 一般格式

**[ORGANIZATION IS] SEQUENTIAL**

### 2.6.3 一般规则

(1) **ORGANIZATION IS SEQUENTIAL** 子句指定文卷逻辑结构为顺序组织。文卷组织在文卷产生时建立且不可更改。

(2) 顺序组织是文卷的永久的逻辑结构，通过先行后继关系来标识一个记录在该逻辑结构中的位

置，这个先行后继关系是在记录被置入文卷中时建立的。

(3) 如果 **ORGANIZATION IS SEQUENTIAL** 未指定，就隐含了顺序组织。

## 2.7 PADDING CHARACTER 子句

### 2.7.1 功能

**PADDING CHARACTER** 子句指定顺序文卷填充块中使用的字符。

### 2.7.2 一般格式

**PADDING CHARACTER IS**  $\left\{ \begin{array}{l} \text{数据名 1} \\ \text{字值 1} \end{array} \right\}$

### 2.7.3 语法规则

(1) 字值 1 必须是非字值的单字符字值；

(2) 数据名 1 可以受限；

(3) 数据名 1 在数据部中必须定义为字符类型的单字符数据项，并且不能在 **COMMUNICATION** 节、**FILE** 节或 **REPORT** 节中定义。

### 2.7.4 一般规则

(1) **PADDING CHARACTER** 子句指明顺序文卷的填充块中使用的字符。在输入操作期间，任何超过最后逻辑记录的块或全由填充符组成的块将被全部跳过。在输入操作期间，仅由填充符组成的逻辑记录将被跳过。在输出操作期间，任何超过最后的逻辑记录的块中所有部分都填以填充符。

(2) 如果 **PADDING CHARACTER** 子句对分配给文卷的设备类型不适用，就不产生或不识别填充符。

(3) 字值 1 或数据名 1 引用的数据项，在产生文卷的 **OPEN** 语句执行期间，用作填充符。填充符是文卷的固定属性。

(4) 如果对文卷指定了 **CODE-SET** 子句，在文卷打开时，对字值 1 或数据名 1 的内容所指定的填充符进行转换。

(5) 如果未指定 **PADDING CHARACTER** 子句，用作填充符的值将由实现者来定义。

(6) 如果相关的文卷连接符是外部文卷连接符，则在运行单位中与该文卷连接符有关的所有 **PADDING CHARACTER** 子句具有相同的说明。如果指定了数据名 1，它必须引用一外部数据项。

## 2.8 RECORD DELIMITER 子句

### 2.8.1 功能

**RECORD DELIMITER** 子句指明了在外部媒体上确定变长记录长度的方式。

### 2.8.2 一般格式

**RECORD DELIMITER IS**  $\left\{ \begin{array}{l} \text{STADNARD-1} \\ \text{实现名 1} \end{array} \right\}$

## 2.8.3 语法规则

- (1) **RECORD DELIMITER** 子句只能对变长记录指定。
- (2) 如果指定了 **STANDARD-1** 短语, 外部媒体必须是磁带文卷。

## 2.8.4 一般规则

- (1) **RECORD DELIMITER** 子句用来指明确定外部媒体上变长记录长度的方法。使用的任何方法都不反映程序中用到的记录区或记录的大小。
- (2) 如果指定了 **STANDARD-1** 短语, 则确定变长记录长度的就按 **GB 7574**《信息处理交换用磁带标号和文卷结构》中说明的方法。
- (3) 如果指定实现名 1 短语, 则确定变长记录长度的方法与实现者定义的实现名 1 有关。
- (4) 如果未指定 **RECORD DELIMITER** 子句, 则确定变长记录长度的方法由实现者指定。
- (5) **OPEN** 语句成功地执行后, 记录定界符就是在与 **OPEN** 语句中指定的文卷名有关的文卷控制描述款的 **RECORD DELIMITER** 子句中指定的定界符。
- (6) 如果相关的文卷连接符是外部文卷连接符, 则运行单位中与该文卷连接符相关的所有 **RECORD DELIMITER** 子句必须具有相同的描述。

2.9 **RESERVE** 子句

## 2.9.1 功能

**RESERVE** 子句允许用户指定分配的输入输出区的数目。

## 2.9.2 一般格式。

**RESERVE** 整数 1  $\left[ \begin{array}{l} \text{AREA} \\ \text{AREAS} \end{array} \right]$

## 2.9.3 一般规则

- (1) **RESERVE** 子句允许用户指定分配的输入输出存区的数目。如果指定了 **RESERVE** 子句, 分配的输入输出的存区数目就等于整数 1, 如果未指定 **RESERVE** 子句, 分配的输入输出存区的数目由实现者指定。

2.10 **I-O-CONTROL** 段

## 2.10.1 功能

**I-O-CONTROL** 段指明了重运行的建立点, 由不同文卷共享的存储区及在多文卷卷上各文卷的位置。在标准 **COBOL** 的这一版本中视 **I-O-CONTROL** 段的 **RERUN** 子句和 **MULTIPLE FILE TAPE** 子句是过时成分, 因为在标准 **COBOL** 的以后的修改版中要把它删掉。

## 2.10.2 一般格式

**I-O-CONTROL.**

$\left[ \left[ \text{RERUN} \left[ \text{ON} \left\{ \begin{array}{l} \text{文卷名 1} \\ \text{实现名} \end{array} \right\} \right] \text{EVERY} \right. \right. \\ \left. \left. \left[ \begin{array}{l} \left[ \text{END OF} \right] \left\{ \begin{array}{l} \text{REEL} \\ \text{UNIT} \end{array} \right\} \text{OF 文卷名 2} \\ \text{整数 1 RECORDS} \\ \text{整数 2 CLOCK-UNITS} \end{array} \right\} \right] \dots \right. \right. \\ \left. \left. \text{条件名 1} \right] \right]$

$\left[ \text{SAME} \left[ \text{RECORD} \right] \text{AREA FOR 文卷名 3 (文卷名 4) ...} \right] \dots$

$\left[ \text{MULTIPLE FILE TAPE CONTAINS} \left\{ \begin{array}{l} \text{文卷名} \\ \text{POSITION 整数 3} \end{array} \right\} \dots \right] \dots$



## 2.10.3 语法规则

(1) 子句出现的次序无关紧要。

## 2.10.4 一般规则

(1) **RERUN** 子句和 **SAME** 子句按字母顺序列出。

2.11 **MULTIPLE FILE TAPE** 子句

## 2.11.1 功能

**MULTIPLE FILE TAPE** 子句指定了文卷在多文卷卷上的位置。在标准 **COBOL** 这一版本中视 **MULTIPLE FILE TAPE** 子句是过时成分，因为在标准 **COBOL** 的以后的修改版中要把它删掉。

## 2.11.2 一般格式

**MULTIPLE FILE TAPE CONTAINS** {文卷名 1 [**POSITION** 整数 1]} ...

## 2.11.3 一般规则

(1) 当多于一个文卷共享同一物理带卷时，**MULTIPLE FILE TAPE** 子句就是必需的。只有那些用在目标程序中的文卷需要指定，而不考虑单卷上文卷的数目。如果所有文卷名都连续地列出就不必再给出 **POSITION** 短语。如果文卷序列中有一个未列出，就必须给出带的初始位置，在同一带卷上每次只能打开一个文卷。

2.12 **RERUN** 子句

**RERUN** 子句指明重运行的建立点。在标准 **COBOL** 的这一版本中视 **RERUN** 子句是过时成分，因为在标准 **COBOL** 的以后的修改版中要把它删掉。

## 2.12.1 一般格式

$$\text{RERUN} \left[ \text{ON} \left\{ \begin{array}{l} \text{文卷名 1} \\ \text{实现名 1} \end{array} \right\} \right] \text{EVERY} \left\{ \begin{array}{l} [\text{END OF}] \left\{ \begin{array}{l} \text{REEL} \\ \text{UNIT} \end{array} \right\} \\ \text{整数 1} \quad \text{RECORDS} \\ \text{整数 2} \quad \text{CLOCK-UNITS} \\ \text{条件名 1} \end{array} \right\} \text{OF 文卷名 2} \right\}$$

## 2.12.2 语法规则

(1) 文卷名 1 必须是顺序组织的文卷。

(2) 只有当文卷名 2 是一顺序组织的文卷时，才能使用 **END OF REEL/UNIT** 短语。由实现者确定 **UNIT** 的定义。

(3) 当整数 1 **RECORDS** 短语或整数 2 **CLOCK-UNITS** 短语指定时，实现名 1 必须在 **RERUN** 子句中给出。

(4) 对于给定的文卷名 2，可以指定多个 **RERUN** 子句，其限制条件如下：

- a. 当指定多个整数 1 **RECORDS** 短语时，其中不能有两个短语指定同一个文卷名 2。
- b. 当指定多个 **END OF UNIT** 短语时，其中不能有两个短语指定同一文卷名 2。

(5) 只能指定一个带有 **CLOCK-UNITS** 短语的 **RERUN** 子句。

## 2.12.3 一般规则

(1) **RERUN** 子句指明了在何时及何处记录下重运行的信息。按下列方式记录重运行信息：

a. 如果指定文卷名 1，就把重运行信息写在每一卷或输出文卷的单位上，实现者指明在卷上或文卷中何处记录重运行信息。

b. 如果指定实现名，则把重运行信息写在实现者指定的设备上的独立文卷中。

(2) 根据建立重运行点的不同条件，**RERUN** 子句可分为七类。实现者必须至少提供一类 **RERUN**

子句。

a. 当使用 **END OF REEL** 或 **END OF UNIT** 短语而又不带 **ON** 短语时,在这种情况下,重运行信息要写在文卷名 2 上,且文卷名 2 必须是一输出文卷。

b. 当使用 **END OF REEL** 或 **END OF UNIT** 短语且在 **ON** 短语中指定了文卷名 1。在这种情况下,重运行信息写在文卷名 1 中,且文卷名 1 必须是输出文卷。此外,执行对文卷名 2 的标准卷单位关闭功能。文卷名 2 可为一输入文卷或输出文卷。

c. 当使用 **END OF REEL** 或 **END OF UNIT** 短语且 **ON** 短语中指定了实现名时。在这种情况下,重运行信息写在实现者定义的独立的重运行单位上。文卷名 2 可以是输入文卷也可以是输出文卷。

d. 当使用整数 1 **RECORDS** 短语时。在这种情况下,一旦文卷名 2 的整数 1 个记录处理完后,就把重运行信息写在实现名 1 指定的设备上,这个实现名 1 须在 **ON** 短语中指定。文卷名 2 可以是输入文卷也可以是输出文卷,组织及存取方式不限。

e. 当使用整数 2 **CLOCK-UNITS** 短语时。在这种情况下,一旦由内部时钟计数的一段时间过后,就把重运行信息写在实现名 1 指定的设备上,这个实现名 1 须在 **ON** 短语中指定。

f. 当使用条件名短语且实现名 1 在 **ON** 短语中指定时。在这种情况下,一旦一个开关具有了条件名 1 指定的特定的状态,就把重运行信息写在实现名 1 指定的设备上。在这种情况下,有关的开关必须在环境部配置节的 **SPECIAL-NAMES** 段中定义。实现者指定何时检测开关状态。

g. 当使用条件名 1 短语且 **ON** 短语中指定文卷名 1 时。在这种情况下,一旦一个开关具有了条件名 1 指定的特定状态,就把重运行信息写在文卷名 1 上,且文卷名 1 必须是输出文卷。在这种情况下,和 f 中一样,相关的开关必须在环境部配置节的 **SPECIAL-NAMES** 段中定义。实现者指定何时检测开关状态。

## 2.13 SAME 子句

### 2.13.1 功能

**SAME** 子句指定由不同文卷共享的存储区。

### 2.13.2 一般格式

**SAME** **RECORD AREA** **FOR** 文卷名 1 {文卷名 2} ...

### 2.13.3 语法规则

(1) 文卷名 1 和文卷名 2 必须在同一程序的 **FILE-CONTROL** 段中指定。

(2) 文卷名 1 和文卷名 2 不能引用外部文卷连接符。

(3) 程序中可以出现多个 **SAME** 子句,但有下列限制:

a. 同一文卷名不能出现在多个 **SAME AREA** 子句中。

b. 同一文卷名不能出现在多个 **SAME RECORD AREA** 子句中。

c. 如果 **SAME AREA** 子句中的一个或多个文卷名出现在 **SAME RECORD AREA** 子句中,则该 **SAME AREA** 子句中所有文卷名都要出现在 **SAME RECORD AREA** 子句中。但是,不出现在 **SAME AREA** 子句中的附加文卷名也可以出现在 **SAME RECORD AREA** 子句中。“**SAME AREA** 子句中在任一给定时刻只有一个文卷名能打开”这一规则,比“**SAME RECORD AREA** 子句中所有文卷在任一时刻都能打开”这一规则的优先级要高。

(4) **SAME AREA** **RECORD AREA** 子句中引用的文卷不必都具有同样的组织或存取方式。

### 2.13.4 一般规则

(1) **SAME AREA** 子句指明了由文卷名 1 和文卷名 2 引用的两个或多个文卷在处理时要用到同一存储区,其中文卷名 1 和文卷名 2 都不代表排序或合并文卷。共享的区域包括分配给文卷名 1 或文卷名 2 引用的文卷的所有存区;因此,在同一时刻不可能有多个这样的文卷处于打开方式(见上述语法规则 3c)。

(2) **SAME RECORD AREA** 子句指明了由文卷名 1 和文卷名 2 引用的两个或多个文卷在处理当前逻辑记录时要用到同一存区。所有这些文卷可以同时处于打开方式。**SAME RECORD AREA** 中的逻辑记录被认为是在该 **SAME RECORD AREA** 子句中出现的以输出方式打开的文卷的逻辑记录, 这被认为是在该 **SAME RECORD AREA** 子句中出现的以输入方式打开的最近读的文卷的逻辑记录。这意价于存区的隐含重定义。即记录以最左字符位置对文。

### 3 顺序 I-O 模块的数据部

#### 3.1 文卷节

文卷节位于源程序的数据部。文卷节定义数据文卷的结构。每个文卷都通过一文卷描述款和一个或多个记录描述款来定义。记录描述款紧跟着文卷描述款之后。

顺序 I-O 模块中文卷节的一般格式如下:

#### **FILE SECTION.**

[文卷描述款

{记录描述款} ...] ...

##### 3.1.1 文卷描述款

在 **COBOL** 程序中, 文卷描述款 (**FD**) 表示文卷节中的最高组织, 文卷节首后接一个文卷描述款, 而文卷描述款由层指示符 (**FD**), 文卷名和一系列独立的子句组成。**FD** 子句指明逻辑记录和物理记录的大小, 标号记录是否存在, 由实现者定义的标号项的值, 组成文卷的数据记录名, 以及到逻辑打印的一页上的行号。这个描述款用句号结束。

##### 3.1.2 记录描述结构

记录描述由一组数据描述款组成, 它们描述特定记录的特征。每个数据描述款由一个层号后面接有一个数据名 (如果需要的话), 及一系列所需要的独立子句所组成。记录描述具有层次结构, 因此和一个描述款一起使用的这些子句可能有很大差别, 这依赖于它后面是否接有下属层次的款。记录描述的结构以及记录描述款中允许出现的成分是在“层的概念”及数据描述款中说明的。数据描述款中特定的子句的可用性是由实现中支持的核心模块的级别来决定的。

##### 3.1.3 初值

文卷节中数据项的初值是未定义的。

#### 3.2 文卷描述款

##### 3.2.1 功能

文卷描述款提供关于一个顺序文卷的物理结构, 标识和记录名的信息。

##### 3.2.2 一般格式

**FD** 文卷中 1

```
[BLOCK CONTAINS            整数 2 {RECORDS
CHARACTERS} ]
[RECORD {CONTAINS 整数 3 CHARACTERS
          
S VARYING IN SIZE [ [FROM 整数 4] [TO 整数 5] CHARACTERS
DEPENDING ON 数据名 1 ]
CONTAINS 整数 6 TO 整数 7 CHARACTERS } ]
[LABEL {RECORD IS } {STANDARD }
RECORDS ARE {OMITTED } ]
[VALUE OF { 实现名 1 IS            } ... ]
           字值 1
```

$$\left[ \text{DATA} \left\{ \begin{array}{l} \text{RECORD IS} \\ \text{RECORDS ARE} \end{array} \right\} \left\{ \begin{array}{l} \text{数据名 3} \\ \dots \end{array} \right\} \right]$$

$$\left[ \begin{array}{l} \text{LINAGE IS} \left\{ \begin{array}{l} \text{数据名 4} \\ \text{整数 8} \end{array} \right\} \text{ LINES} \left[ \text{WITH} \right. \\ \quad \left. \text{FOOTING AT} \left\{ \begin{array}{l} \text{数据名 5} \\ \text{整数 9} \end{array} \right\} \right] \left[ \text{LINES AT TOP} \right. \\ \quad \left. \left\{ \begin{array}{l} \text{数据名 6} \\ \text{整数 10} \end{array} \right\} \right] \left[ \text{LINES AT BOTTOM} \left\{ \begin{array}{l} \text{数据名 7} \\ \text{整数 11} \end{array} \right\} \right] \right] \end{array} \right]$$

$$[\text{CODE-SET IS 字母表名 1}].$$

### 3.2.3 语法规则

- (1) 层指示符 **FD** 标识文卷描述的开始，而且必须位于文卷名 1 前面。
- (2) 接在文卷名 1 后的句子出现次序是任意的。
- (3) 在文卷描述款后必须接有一个或多个记录描述款。

### 3.2.4 一般规则

- (1) 文卷描述款使文卷名 1 与一个文卷连接符相联系。
- (2) **BLOCK CONTAINS** 子句、**CODE-SET** 子句、**DATA RECORDS** 子句、**LABEL RECORDS** 子句、**LINAGE** 子句、**RECORD** 子句及 **VALUE OF** 子句按字母顺序依次列出。

## 3.3 BLOCK CONTAINS 子句

### 3.3.1 功能

**BLOCK CONTAINS** 子句指明物理记录的大小。

### 3.3.2 一般格式

$$\text{BLOCK CONTAINS} \left\{ \begin{array}{l} \text{RECORDS} \\ \text{CHARACTERS} \end{array} \right\} \left\{ \begin{array}{l} \text{整数 1} \\ \text{整数 2} \end{array} \right\}$$

### 3.3.3 一般规则

- (1) 除了下列情形外，这个子句总是需要的：
  - a. 一个物理记录包含而且仅包含一个完整的逻辑记录。
  - b. 分配给这个文卷的硬设备有且仅有一个物理记录大小。
  - c. 块中包含的记录数目在操作环境中指明。
- (2) 物理记录的大小可以用记录数来表明。但在下列情形中，一定不能使用 **RECORDS** 短语。
  - a. 在海量存储文卷中，逻辑记录可以跨物理记录。
  - b. 物理记录包含填补区（不包含在逻辑记录中的区）。
  - c. 逻辑记录的组合使得隐含一个不精确的物理记录大小。
- (3) 当指明字 **CHARACTERS** 时，物理记录大小是用存储该物理记录所需的字符位置数来指出，而不考虑表示物理记录的数据项所使用的字符类型。

(4)  $\left\{ \begin{array}{l} \text{整数 1} \\ \text{整数 2} \end{array} \right\}$  整数 2 代表物理记录的精确长度。如果整数 1 和整数 2 都指定了，它们就分别代表物理记录的最小长度和最大长度。

(5) 如果相关的文卷连接符是外部文卷连接符，在运行单位中与该文卷连接符相关的所有 **BLOCK CONTAINS** 子句中的整数 1 和整数 2 具有相同的值。

## 3.4 CODE-SET 子句

### 3.4.1 功能

**CODE-SET** 子句指明表示外部媒体上的数据所使用的字符编码集。

### 3.4.2 一般格式

**CODE-SET IS 字母表名 1**

### 3.4.3 语法规则

(1) 当对一个文卷指明了 **CODE-SET** 子句时, 在这个文卷中的所有数据都必须描述为 **USAGE IS DISPLAY**。而且任何有正负号的数值数据必须用 **SIGN IS SEPARATE** 子句描述。

(2) 由 **CODE-SET** 子句引用字母表名子句一定不能指明字值常量短语。

### 3.4.4 一般规则

(1) 如果指明了 **CODE-SET** 子句:

a. 成功地执行 **OPEN** 语句之后, 表示外部媒体上数据的字符集就是由 **OPEN** 语句中指定的文卷名的文卷描述款中的字母表名 1 所引用的字符集 (见 GB/T 4092.2 中 4.5 **SPECIAL-NAMES** 段)。

b. 它指明了从外部媒体上的字符集转换成本原字符集或者从本原字符集转换成外部媒体上字符集的转换算法。这种转换在执行输入输出操作期间发生。

(2) 如果没有指明 **CODE-SET** 子句, 则外部媒体上的数据采用本原字符集。

(3) 如果相关的文卷连接符是外部文卷连接符, 则运行单位中与该文卷连接符相关的 **CODE-SET** 子句具有相同的字符集。

## 3.5 DATA RECORDS 子句

### 3.5.1 功能

**DATA RECORDS** 子句仅用来在程序中列出有关文卷的各个数据记录的名字。在标准 **COBOL** 的这一版本中视 **DATA RECORD** 是过时成分, 因为在标准 **COBOL** 的以后的修改版中要把它删掉。

### 3.5.2 一般格式

$$\text{DATA} \left\{ \begin{array}{l} \text{RECORD IS} \\ \text{RECORDS ARE} \end{array} \right\} \{ \text{数据名 1} \} \dots$$

### 3.5.3 语法规则

(1) 数据名 1 是数据记录名, 它们必须有相应的 01 层号描述, 且在描述中用相同的名。

### 3.5.4 一般规则

(1) 出现多个数据名表明该文卷包含多种数据记录。这些记录可以有不同的大小。不同的格式等。它们在句子中列出的次序是不重要的。

(2) 从概念上说一个文卷中的所有记录共享同一个区, 这个区并不因文卷中出现多种数据类型而改变。

## 3.6 LABEL RECORDS 子句

### 3.6.1 功能

**LABEL RECORDS** 指明标号是否出现。在标准 **COBOL** 的这一版本中视 **LABEL RECORDS** 子句是过时成分, 因为在标准 **COBOL** 的以后的修改版中要把它删掉。

### 3.6.2 一般格式

$$\text{LABEL} \left\{ \begin{array}{l} \text{RECORD IS} \\ \text{RECORDS ARE} \end{array} \right\} \left\{ \begin{array}{l} \text{STANDARD} \\ \text{OMITTED} \end{array} \right\}$$

### 3.6.3 一般规则

(1) **OMITTED** 指明该文卷或分配给该文卷的设备不存在显式的标号。

(2) **STANDARD** 指明该文卷或分配给该文卷的设备上存在标号, 并且该标号与实现者的标号指明一致。

(3) 如果没有对文卷指明 **LABEL RECORDS** 子句, 则该文卷的标号记录应与实现者的标号说明一致。

(4) 如果与该文卷描述款相关的文卷连接符是外部文卷连接符 (见 GB/T 4092.12 中 4.5 **EXTERNAL** 子句), 在运行单位中与该文卷连接符相关的所有 **LABEL RECORDS** 子句应有同样的说明。

## 3.7 LINAGE 子句

### 3.7.1 功能

**LINAGE** 子句提供了一种用行数来指明一个逻辑页的深度的方法。它也用来指明逻辑页的顶界和底界的大小,以及页体中页尾区开始的行号。

### 3.7.2 一般格式

$$\text{LINAGE IS } \left\{ \begin{array}{l} \text{数据名 1} \\ \text{整数 1} \end{array} \right\} \text{ LINES } \left[ \text{WITH FOOTING} \right. \\ \left. \text{AT } \left\{ \begin{array}{l} \text{数据名 2} \\ \text{整数 2} \end{array} \right\} \right] \left[ \text{LINES AT TOP } \left\{ \begin{array}{l} \text{数据名 3} \\ \text{整数 3} \end{array} \right\} \right] \\ \left[ \text{LINES AT BOTTOM } \left\{ \begin{array}{l} \text{数据名 4} \\ \text{整数 4} \end{array} \right\} \right]$$

### 3.7.3 语法规则

- (1) 数据名 1、数据名 2、数据名 3、数据名 4 必须引用初等无正负号整数数据项。
- (2) 数据名 1、数据名 2、数据名 3、数据名 4 可以受限。
- (3) 整数 2 的值必须不大于整数 1。
- (4) 整数 3、整数 4 的值可以是零。

### 3.7.4 一般规则

(1) **LINAGE** 子句提供用行数来指明逻辑页大小的方法,逻辑页的大小是 **FOOTING** 短语除外的每个短语所引用的值的总和。如果未指明 **LINES AT TOP** 或 **LINES AT BOTTOM** 短语,则意味着逻辑页的顶界和底界的行数为 0。如果未指明 **FOOTING** 短语,则不存在与页溢出条件无关的页结束条件。

逻辑页的大小和物理页的大小之间无须有任何联系。

(2) 整数 1 或由数据名 1 引用的数据项的值指出可以往逻辑页上写信息,以及/或在逻辑页上留空的行数。这个值必须大于零。逻辑页上能写信息和/或留空的这些行称作页体。

(3) 整数 2 或由数据名 2 引用的数据项的值指明页体中页尾区开始的行号。该值必须大于零且不大于整数 1 或由数据名 1 引用的数据项的值。

页尾区是由整数 2 或数据名 2 引用的数据项的值表示的行,和由整数 1 或数据名 1 引用的数据项的值(包括它本身)表示的行之间所组成的页体区域。

(4) 整数 3 或由数据名 3 引用的数据项的值指明构成逻辑页底界的行数,该值可以是零。

(5) 整数 4 或由数据名 4 引用的数据项的值指明构成逻辑页底界的行数,该值可以是零。

(6) 如果指明整数 1、整数 3 和整数 4 的值,则这些值将在由执行具有 **OUTPUT** 短语的 **OPEN** 语句在打开文卷时使用,以指出组成各个逻辑页的各节的行数。如果指明了整数 2,则该值将在定义页尾区时使用。在给定的程序执行期间,这些值适用于文卷所写的所有逻辑页。

(7) 如果指明由数据名 1、数据名 3 和数据名 4 引用的数据项的值,则它们将按以下方式使用:

a. 在对该文卷执行有 **OUTPUT** 短语的 **OPEN** 语句时,这些数据项的值用来指明第一个逻辑页上组成所指定的各节的行数。

b. 在执行带有 **ADVANCING PAGE** 短语的 **WRITE** 语句时,或者发生页溢出条件时(见 4.1.1 **WRITE** 语句),这些数据项的值,通常将用来指出下一个逻辑页组成所指定的部与各节的行数(见 4.1.1 **WRITE** 语句)。

(8) 如果指明由数据名 2 引用的数据项的值,则在对这个文卷执行具有 **OUTPUT** 短语的 **OPEN** 语句时,该值将用来定义第一个逻辑页的页尾区。在执行具有 **ADVANCING PAGE** 短语的 **WRITE** 语句或发生页溢出条件时,它将用来定义下一个逻辑页的页尾区。

(9) **LINAGE** 子句的存在导致产生一个 **LINAGE-COUNTER**。在任一给定的时间里 **LINAGE-COUNTER** 的值表示当前页体里设备定位的行号。控制 **LINAGE-COUNTER** 的规则如下：

a. 对在文卷描述款中包含一个 **LINAGE** 子句的文卷节中描述的每个文卷提供一单独的 **LINAGE-COUNTER**。

b. **LINAGE-COUNTER** 仅可由过程部的语句引用，但仅能由输入输出控制系统改变其值。因为在一个程序中可能存在多个 **LINAGE-COUNTER**，当必要时，用户必须用文卷名限定 **LINAGE-COUNTER**。

c. 在相关联的文卷的 **WRITE** 语句执行期间，**LINAGE-COUNTER** 按照下列规则自动修改：

1) 当指明了 **WRITE** 语句的 **ADVANCING PAGE** 短语时，**LINAGE-COUNTER** 自动重置为 1。把 **LINAGE-COUNTER** 重置为 1 时，就隐含地对 **LINAGE-COUNTER** 增值使之超过整数或数据名引用的数据项之值。

2) 当指明了 **WRITE** 语句的 **ADVANCING** 标识符 2 或整数 1 短语时，**LINAGE-COUNTER** 的增量值等于整数 1 或标识符 2 引用的数据项的值。

3) 当未指明 **WRITE** 语句的 **ADVANCING** 短语时，该 **LINAGE-COUNTER** 的增量值为 1（见 4.7WRITE 语句）。

4) 当设备重定位到各个后继逻辑页的第一可写行时，**LINAGE-COUNTER** 的值自动重置为 1（见 4.7WRITE 语句）。

d. 在执行相关联文卷的带有 **OUTPUT** 短语的 **OPEN** 语句时，**LINAGE-COUNTER** 的值自动重置为 1。

(10) 每个逻辑页到下一页是连接的，其间不提供附加的空格。

(11) 如果与该文卷描述款相关的文卷连接符是外部文卷连接符，则运行单位中与该文卷连接符相关的所有文卷描述款必须：

a. 具有一个 **LINAGE** 子句，如果任一文卷描述款有一个 **LINAGE** 子句的话；

b. 整数 1、整数 2、整数 3、整数 4 相应地具有同样的值，如果它们被指定的话。

c. 具有由数据名 1、2、3、4 引用的同样的外部数据项。

### 3.8 RECORD 子句

#### 3.8.1 功能

**RECORD** 子句在定长记录中指出字符位置数，或在变长记录中指明字符位置的范围。如果字符位置数确有变化的话，该子句就指出字符位置的最小数和最大数。

#### 3.8.2 一般格式

格式 1：

**RECORD CONTAINS** 整数 1 **CHARACTERS**

格式 2：

**RECORD IS VARYING IN SIZE** [ **[FROM** 整数 2]

**[TO** 整数 3] **CHARACTERS]**

**[DEPENDENT ON** 数据名 1]

格式 3：

**RECORD CONTAINS** 整数 4 **TO** 整数 5 **CHARACTERS**

#### 3.8.3 语法规则

对格式 1 而言：

(1) 文卷记录描述款中不能指定大于整数 1 的字符位置数。

对格式 2 而言：

- (2) 文卷记录描述款描述的记录含有的字符位置数不得小于整数 2 之值，也不能大于整数 3 之值。
- (3) 整数 3 要大于整数 2。
- (4) 数据名 1 必须在文卷存储节或连接节中描述一非正号是初等数数。

### 3.8.4 一般规则

对所有格式而言：

- (1) 如果未指定 **RECORD** 子句，则每个数据记录的大小完全在记录描述款中定义。
- (2) 如果相关的文卷连接符是外部连接符，则运行单位中与该文卷连接符有关的所有文卷描述款必须为整数 1 指定相同的值。如果未指明 **RECORD** 子句，与该文卷连接符有关的所有记录描述款必须等长。

对格式 1 而言：

- (3) 格式 1 用来指明定长记录，整数 1 指明文卷中每一记录的字符位置数。

对格式 2 而言：

- (4) 格式 2 用来指明变长记录。整数 2 指明文卷中每一记录的最小字符位置数。整数 3 指明文卷中每一记录的最大字符位置数。
- (5) 与记录描述款相关的字符位置数等于所有初等项（除去重定义和重命名）的字符位置数之和再加上同步所需的隐含的 **FILLER**。如果指明了表，则：
  - a. 记录中表元的最小数目用在上述的相加运算中来确定与记录描述有关的最小字符位置数。
  - b. 记录中描述的表元的最大数目用在上述相加运算中以确定与记录描述有关的最大字符位置数。
- (6) 如果未指明整数 2，文卷中任一记录的最小字符位置数等于该文卷的记录中描述的最小字符位置数。
- (7) 如果未指明整数 3，文卷中任一记录的最大字符位置数等于该文卷的记录中描述的最大字符位置数。
- (8) 如果指明了数据名 1，在对文卷执行 **RELEASE**、**REWRITE** 或 **WRITE** 语句之前，必须把记录的字符位置数放入数据名 1 引用的数据项中。
- (9) 如果指明了数据名 1，则 **DELETE**、**RELEASE**、**REWRITE**、**START** 或 **WRITE** 语句的执行以及 **READ** 或 **RETURN** 语句的不成功的执行都不改变数据名 1 引用的数据项的值。
- (10) 在执行 **RELEASE**、**REWRITE** 或 **WRITE** 语句时，记录中的字符位置数由下列条件确定
  - a. 如果指明了数据名 1，则由数据名 1 引用的数据项的内容确定；
  - b. 如果未指明数据名 1 且记录中不包含有可变出现的数据项，则由记录中的字符位置数确定；
  - c. 如果未指明数据名 1 且记录中确定包含有可变出现的数据项，则由表的固定部分加上执行输出语句时表中用出现号描述的那部分确定；
- (11) 如果指定了数据名 1，则对文卷执行 **READ** 或 **RETURN** 语句成功之后，数据名 1 引用的数据项之值就指出刚读入的记录的字符位置数。
- (12) 如果在 **READ** 或 **RETURN** 语句中指定了 **INTO** 短语，则在隐含的 **MOVE** 语句中充当发送项的当前记录的字符位置数由下列条件确定：
  - a. 如果指明了数据名 1，则由数据名 1 引用的数据项的内容确定；
  - b. 如果未指明数据名 1，则由假定指明数据名 1 时要传送到数据名 1 引用的数据项的值来确定。

对格式 3 而言：



(13) 当使用格式 3 的 **RECORD** 子句时。整数 4 和整数 5 分别指出最小数据记录的最小字符位置数以及最大数据记录的最大字符位置数。可是,在这种情况下,每个数据记录的长度完全在记录描述款中定义。

(14) 不管逻辑记录中数据项的字符类型是什么,数据记录的大小总是用存储逻辑记录所需要的字符位置数来指明的。一个记录的大小是由所有固定长的初等项加上属于这个记录的任何可变长的数据项的最大字符数确定的。这个和数可能与实际的记录长度不同(见 GB/T 4092.1 中 6.4.3.4 字符表示及数基的选择; GB/T 4092.2 中 5.13 **SYNCHRONIZED** 子句和 5.14 **USAGE** 子句)。

### 3.9 VALUE OF 子句

#### 3.9.1 功能

**VALUE OF** 子句详细列出有关文卷标号记录中的数据项的描述。在标准 **COBOL** 的这一版本中视 **VALUE OF** 子句是过时成分,因为在标准 **COBOL** 的以后的修改版中要把它删掉。

#### 3.9.2 一般格式

**VALUE OF** { 实现名 1 IS { 字值 1 } }

#### 3.9.3 语法规则

(1) 必要时数据名 1 应该受限,但不能带下标或位标,也不能是有 **USAGE IS INDEX** 子句描述的数据项。

(2) 数据名 1 必须在工作存储节中定义。

#### 3.9.4 一般规则

(1) 对于输入文卷,相应的标号例行程序核对实现名 1 的值是否等于字值 1 或数据名 1 引用的数据项的值,这取决于何者被指明。

对于输出,在适当的时间实现名 1 的值等于字值 1 或数据名 1 引用的数据项的值,这取决于何者被指明。

(2) 如果相关的文卷连接符是外部文件连接符,则运行单位与该文卷连接符有关的所有 **VALUE OF** 子句必须是一致的。实现者指明这些一致性的规则。

## 4 顺序 I-O 模块的过程部

### 4.1 一般描述

**COBOL** 源程序中出现顺序 I-O 模块里的 **USE** 语句时,过程部中就含有申述过程。下面列出 **USE** 语句出现时过程部的一般格式:

**PROCEDURE DIVISION.**

**DECLARATIVES.**

{节名 **SECTION.**

**USE** 语句.

[段名.

[句子] ...] ...} ...

**END DECLARATIVES.**

{节名 **SECTION.**

[段名.

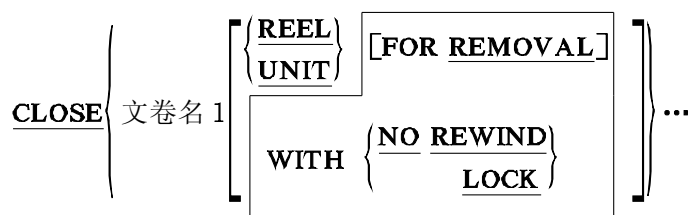
[句子] ...] ...} ...

### 4.2 CLOSE 语句

#### 4.2.1 功能

**CLOSE** 语句终止卷或单位和文卷的处理，而且在可能时也终止具有反绕选用和/或锁选用或撤销选用的文卷的处理。

#### 4.2.2 一般格式



#### 4.2.3 语法规则

(1) 在 **CLOSE** 语句中引用的文卷无须全都具有相同组织或存取方式。

#### 4.2.4 一般规则

除在下面的一般规则中另有说明的情形外，“卷”和“单位”这两个术语是同义的并且在 **CLOSE** 语句中完全可以互用。顺序海量文卷的处理逻辑上等价于磁带或类似的顺序媒体上文卷的处理。

在大多卷带环境中的文卷处理逻辑上等价于顺序的单卷或单单位文卷的处理，只要文卷整个包含在卷中就行。

(1) 只对处于打开方式的文卷才能执行 **CLOSE** 语句。

(2) 为了说明应用于不同存储媒体的各种 **CLOSE** 语句的效果，将所有文卷分成下述几类：

- a. 非卷或非单位文卷。这种输入或输出媒体的文卷对反绕和卷或单位的概念是没有意义的。
- b. 顺序单卷或单单位文卷。全部包含在一个卷或一个单位上的顺序文卷。
- c. 顺序多卷或多单位文卷。包含在多于一卷或多于一个单位上的顺序文卷。

(3) 对各类文卷的每一种 **CLOSE** 语句执行结果概括在表 1 中。

表 1 文卷类型和 **CLOSE** 语句格式关系表

| CLOSE 语句格式                         | 文 卷 类 型     |                |                   |
|------------------------------------|-------------|----------------|-------------------|
|                                    | 非卷或非单位      | 顺序单卷或单单位       | 顺序多卷或多单位          |
| <b>CLOSE</b>                       | <b>C</b>    | <b>C, G</b>    | <b>A, C, G</b>    |
| <b>CLOSE WITH LOCK</b>             | <b>C, E</b> | <b>C, E, G</b> | <b>A, C, E, G</b> |
| <b>CLOSE WITH NO REWIND</b>        | <b>C, H</b> | <b>B, C</b>    | <b>A, B, C</b>    |
| <b>CLOSE REEL/UNIT</b>             | <b>F</b>    | <b>F, G</b>    | <b>F, G</b>       |
| <b>CLOSE REEL/UNIT FOR REMOVAL</b> | <b>F</b>    | <b>D, F, G</b> | <b>D, F, G</b>    |

下面给出表中各个符号的定义。这是根据文卷是输入文卷、输出文卷还是输入-输出文卷而分别给出相应的定义；否则，一个定义就适用于输入文卷、输出文卷、以及输入-输出文卷。

**A. 对前面的卷或单位的影响。**

输入文卷和输入-输出文卷：

在当前卷或单位以前的文卷中所有的卷或单位都被关闭，但由以前的 **CLOSE REEL/UNIT** 语句控制的那些卷或单位则为例外。如果当前卷或单位不是文卷的最末一卷或最末一个单位，则不处理紧接在文卷的当前卷或当前单位之后的那些卷或单位。

输出文卷：

在当前卷或单位以前的文卷中所有的卷或单位都被关闭。但由先前的 **CLOSE REEL/UNIT** 语句控制的那些卷或单位则为例外。

**B. 当前的卷不反绕**

当前的卷或设备在当前的位置离开。

#### C. 关闭文卷

输入文卷和输入-输出文卷：

如果文卷位于它的末端并且对该文卷指明了标号记录，则按照实现者的标准标号约定处理标号。当指明有标号记录但不出现或当未指出标号记录而出现时，**CLOSE** 语句的行为是未定义的。但系统执行由实现者指定的关闭操作。如果文卷定位在它的末端位置，而对该文卷没有指明标号记录，则不处理标号，而执行实现者指出的其他关闭操作。如果文卷未定位在它的末端位置，则执行由实现者指出的关闭操作，但不进行尾标处理。

输出文卷：

如果对文卷指明了标号记录，则按实现者的标准标号约定处理标号。当指明了标号记录但不出现或未指明标号记录而出现时，**CLOSE** 语句的行为是未定义的。系统执行由实现者指出的关闭操作。如果对该文卷没有指明标号记录，则不处理标号，但执行由实现者指出的其他关闭操作。

#### D. 卷或单位的撤销

适用的话反绕当前的卷或单位。该卷或单位逻辑上从该运行单位撤销。然而若跟着执行不带 **REEL** 或 **UNIT** 短语的 **CLOSE** 语句，接着对该文卷执行 **OPEN** 语句则按照该文卷中卷或单位的适当次序，该卷或单位可以再次被存取。

#### E. 文卷锁

给文卷加锁来保证在运行单位的执行期间该文卷不能再次被打开。

#### F. 关闭卷或单位

输入文卷和输入-输出文卷：

进行下列操作：

(1) 如果当前卷或单位是文卷的最后或仅有的一个卷或单位，或者卷位于非卷或非单位媒体上，则不进行卷或单位对换且当前卷指针保持不变。

(2) 如果文卷存在另一卷或单位，就对换卷或单位，当前卷指针值改变以指向文卷中下一卷或单位，执行标准开始卷或单位的标号过程。如果当前不存在数据记录，就再对换一次。

输出文卷（卷或单位媒体）：

进行下列操作：

1) 执行标准结束卷或单位的标号过程。

2) 对换卷或单位。改变当前卷指针值以指向下一卷或单位。

3) 执行标准开始卷或单位的标号过程。

4) 引用该文卷的下一执行执行的 **WRITE** 语句，把下一逻辑数据记录送到该文卷的下一卷或单位。

输出文卷（非卷或非单位媒体）：

该语句的执行被认为是成功的，文卷保持打开方式且除了一般规则 4 中指明的动作以外，不采取其他动作。

#### G. 反绕

当前卷或类似的设备定位在开始物理位置。

##### 1. 未考虑的任选短语

**CLOSE** 语句执行时假定不出现任选短语。

(4) 执行 **CLOSE** 语句使得与文卷名 1 相关的 I-O 状态之值改变（见 1.3.5 I-O 状态）。

(5) 如果不出现任选的输入文卷，则不对文卷进行“文卷末端”或“卷或单位末端”处理，并且文卷位置指示符和当前卷指针保持不变。

(6) 对文卷成功地执行了不带 **REEL** 或 **UNIT** 短语的 **CLOSE** 语句之后，与文卷名 1 相关的记录

区就不再是可用的了。如果这样的 **CLOSE** 语句执行不成功，则记录区的可用性就是未定义的。

(7) 对文卷成功地执行了不带 **REEL** 或 **UNIT** 短语的 **CLOSE** 语句之后，文卷从打开方式下撤销，并且不再与文卷连接符相关联。

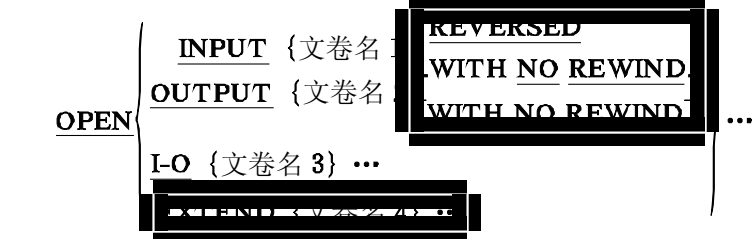
(8) 如果在一个 **CLOSE** 语句中指定的文卷名 1 多于一个，则执行这个 **CLOSE** 语句的结果，就如同对 **CLOSE** 语句中的每个文卷名 1 以同样的顺序分开写出的一串 **CLOSE** 语句一样。

4.3 OPEN 语句

4.3.1 功能

**OPEN** 语句初启文卷的处理。在标准 **COBOL** 的这一版本中视 **REVERSED** 短语是过时成分，因为在标准 **COBOL** 的以后的修改版中要把它删掉。

4.3.2 一般格式



4.3.3 语法规则

- 1) **REVERSED** 短语只能用于顺序文卷中。
- 2) 对多文卷不能指明 **EXTEND** 短语。
- 3) **EXTEND** 短语只能用于指明 **LINEAGE** 子句的文卷。

(4) 在 **OPEN** 语句中引用的文卷无须全都具有相同的组织方式或存取方式。

4.3.4 一般规则

(1) **OPEN** 语句的成功执行决定了文卷的可用性，并且使文卷处于打开方式。**OPEN** 语句的成功执行通过文卷连接符把文卷与文卷名联系起来。

如果一个文卷在物理上出现且由输入输出控制系统识别，则此文卷就是可用的。表 2 显示了对可用及不可用的文卷打开时的结果。

表 2 文卷的可用性

|               | 文卷可用        | 文卷不可用          |
|---------------|-------------|----------------|
| 输入            | 正常打开        | 打开不成功          |
| 输入 (任选文卷)     | 正常打开        | 正常打开；首次读产生末端条件 |
| I-O           | 正常打开        | 打开不成功          |
| I-O (任选文卷)    | 正常打开        | 打开使文卷产生        |
| 输出            | 正常打开；文卷中无记录 | 打开使文卷产生        |
| EXTEND        | 正常打开        | 打开不成功          |
| EXTEND (任选文卷) | 正常打开        | 打开使文卷产生        |

(2) **OPEN** 语句的成功执行使程序可使用相关的记录区。如果与文卷名相关的文卷连接符是外部文卷连接符，则运行单位中只有一个与文卷连接符相关的记录区。

(3) 如果一个文卷不处于打开方式，则任何隐式或显式引用该文卷的语句都不能执行，但带有 **USING** 或 **GIVING** 短语的 **MERGE** 语句、**OPEN** 语句或带有 **USING** 或 **GIVING** 短语的 **SORT** 语句则除外。

(4) 在执行任何可允许的输入-输出语句之前，首先必须成功地执行 **OPEN** 语句。表 3 (可允许的语句) 中，在交点处的 “X” 指出在列的顶部给出的文卷的打开方式下可以使用给定的语句。

表 3 可允许的语句

| 语句             | 打 开 方 式 |    |       |    |
|----------------|---------|----|-------|----|
|                | 输入      | 输出 | 输入-输出 | 扩展 |
| <b>READ</b>    | X       |    | X     |    |
| <b>WRITE</b>   |         | X  |       | X  |
| <b>REWRITE</b> |         |    | X     |    |

(5) 在同一个运行单位中，文卷可以用 **INPUT**、**OUTPUT**、**REEL** 和 **I-O** 短语打开。在对文卷初次执行的 **OPEN** 语句后，对同一个文卷每当要执行下一个 **OPEN** 语句，就必须首先执行一个不带 **REEL**、**UNIT** 短语的 **CLOSE** 语句。

(6) 执行 **OPEN** 语句并不获取或释放第一个数据记录。

(7) 如果对文卷指明了标号记录，则开始的标号处理如下：

a. 当指明 **INPUT** 短语时，执行 **OPEN** 语句引起依据实现者所指明的对输入标号核对的约定核对标号。

b. 当指明 **OUTPUT** 短语时，执行 **OPEN** 语句引起依据实现者所指明书写输出标号的约定书写标号。

当指明了标号记录但未出现，或当未指明而标号记录出现时，**OPEN** 语句的行为是未定义的。

(8) 在 **OPEN** 语句执行期间，如果文卷属性冲突条件发生，则该 **OPEN** 语句的执行是不成功的 (见 1.3.7 文卷属性冲突条件)。

(9) **NO REWIND** 和 **REVERSED** 短语只能用于：

a. 顺序单卷或单单位文卷 (见 4.2 **CLOSE** 语句)。

b. 在多文卷带环境下完全包含在一单卷带上的顺序文卷 (见 2.11 **MULTIPLE FILE TAPE** 语句)。

(10) 如果在存放文卷的存储媒体上，**REVERSED** 和 **WITH NO REWIND** 短语不适用，便跳过此短语。

(11) 如果文卷的存储媒体允许反绕，则应用下列规则：

a. 当指明 **REVERSED** 短语时，执行 **OPEN** 语句后文卷定位在开始位置。

b. 当指明 **NO REWIND** 短语时，执行 **OPEN** 语句不引起该文卷重新定位，在 **OPEN** 语句开始执行之前，文卷必须已经定位在开始位置。

c. 当指明 **REVERSED** 短语时，执行 **OPEN** 语句引起文卷定位在它的末端。

(12) 当指明 **REVERSED** 短语时，后面使用该文卷的 **READ** 语句使该文卷的数据记录是按反序方向可用的，即从文卷的最末一个记录开始。

(13) 如果文卷是用 **INPUT** 短语打开的且该文卷是未出现的任选文卷，**OPEN** 语句对文卷位置指示符赋值以指明任选输入文卷未出现。

(14) 用 **INPUT** 或 **I-O** 短语打开的文卷，**OPEN** 把文卷位置指示符置为 1。

(15) 当指明 **EXTEND** 短语时, **OPEN** 语句将该文卷定位于紧接在该文卷的最末一个逻辑记录之后。

(16) 当指明 **EXTEND** 短语并且 **LABEL RECORDS** 子句指明标号记录出现, 则执行 **OPEN** 语句包括如下几步:

- a. 仅在单卷或单单位文卷的情形处理开始的文卷标号。
- b. 处理最末一个现存的卷或单位的初始的卷或单位的标号, 就如同文卷用 **INPUT** 短语打开一样。
- c. 处理现存的结束的文卷标号, 就如同文卷用 **INPUT** 短语打开一样, 然后就删去这些标号。
- d. 随后, 就如同文卷用 **OUTPUT** 短语打开时一样进行处理。

(17) 带有 **I-O** 短语的 **OPEN** 语句必须引用一个支持输入和输出操作的文卷, 这些操作应对以 **I-O** 方式打开的顺序文卷是允许的。带有 **I-O** 短语的 **OPEN** 语句执行后, 将引用的文卷置于允许输入和输出操作的打开方式之下。

(18) 当指明了 **I-O** 短语并且 **LABEL RECORDS** 子句指明标号记录出现, 执行 **OPEN** 语句包括如下几步:

- a. 依照实现者所指明的对输入输出标号核对的约定核对标号。
- b. 依照实现者对输入输出标号的书写约定书写新标号。

(19) 在多文卷带环境下的文卷的处理逻辑上等价于单文卷带环境下的顺序文卷的处理。

(20) 只要一组文卷位于多文卷上且其中之一被 **OPEN** 语句引用, 则下列规则适用:

- a. 不能有多于一个文卷同时处于打开方式。
- b. 对文卷以输入方式打开的顺序无限制。
- c. 如果文卷名引用的文卷之一是带有 **OUTPUT** 短语的 **OPEN** 语句的主项, 则在执行 **OPEN** 语句时, 在相关的多文卷上比该卷位置号小的所有文卷必须已经存在于卷上, 同时, 卷上不存在位置号比此时将到卷中的该文卷的位置号要大的文卷。
- d. 每个立卷都必须呈顺序立卷。

(21) 对于一个不可用的任选文卷, 在带有 **EXTEND** 或 **I-O** 短语的 **OPEN** 语句执行成功后, 就建立了该文卷。建立过程就如同按序执行下列语句:

**OPEN OUTPUT** 文卷名.

**CLOSE** 文卷名.

上述语句执行后再执行源程序中指明的 **OPEN** 语句。

当成功地执行带有 **OUTPUT** 短语的 **OPEN** 语句时, 文卷就建立了, 该文卷不包含数据记录。

(22) **OPEN** 语句执行成功后, 对当前卷指针赋值, 目的是:

- a. 为一个可用的输入或 **I-O** 文卷指出第一个或仅有的一个卷或单位。

- c. 为一个不可用输出, **I-O** 文卷指出新的卷或单位。

(23) **OPEN** 语句执行后, 改变了与文卷名相关的 **I-O** 状态之值 (见 1.3.5 **I-O** 状态)。

(24) 如果在一个 **OPEN** 语句中指定的文卷名多于一个, 则执行这个 **OPEN** 语句的结果就如同对 **OPEN** 语句中的每个文卷名以同样的顺序分开写出的一串 **OPEN** 语句一样。

(25) 文卷记录的最小长度和最大长度在文卷生成时建立并且以后不可改变。

#### 4.4 READ 语句

##### 4.4.1 功能

**READ** 语句使文卷的下一逻辑记录成为可用的。

##### 4.4.2 一般格式

**READ** 文卷名 1 **INTO** **RECORD** [ **INTO** 标识符 1 ]

[ **AT END** 命令语句 1 ]

[ **NOT AT END** 命令语句 2 ]

[ **END-READ** ]

#### 4.4.3 语法规则

(1) 与标识符有关的存储区和与文卷名有关的记录区不能是同一个存储区。

(2) 如果对文卷名 1 没有指明可应用的 **USE AFTER STANDARD EXCEPTION** 过程, 则必须指明 **AT END** 短语。

#### 4.4.4 一般规则

(1) 在执行这个语句时, 文卷名 1 引用的文卷必须用输入或 **I-O** 方式打开 (见 4.3 **OPEN** 语句)。

(3) **READ** 语句执行后, 与文卷名 1 有关的 **I-O** 状态之值就被改变 (见 1.3.5 **I-O** 状态)。

(4) 在 **READ** 语句初始执行时对文卷位置指示符赋以确定值, 用它来确定成为可用的记录, 其规则如下所列。顺序文卷中记录的比较与记录号有关。

a. 如果文卷位置指示符指出没有建立有效的下一记录, 则 **READ** 语句的执行是不成功的。

b. 如果文卷位置指示符指出 **RECORD** 标识符, 则按照一般规则 10 继续执行下去。

c. 如果文卷位置指示符是由先前执行的 **OPEN** 语句建立的, 就选择文卷中记录号大于或等于文卷位置指示符值的第一个现存记录。

d. 如果文卷位置指示符是由先前执行的 **READ** 语句建立的, 则选择文卷中记录号大于文卷位置指示符值的第一个现存记录。

如果发现一记录满足上述规则, 它就在与文卷名 1 有关的记录区中成为可用的。

如果没有发现满足上述规则的记录, 则对文卷位置指示符赋值以指明不存在下一逻辑记录且按照一般规则 10 继续执行下去。

一个记录成为可用的之后, 对文卷位置指示符赋值以指向可用记录的记录号。

(5) 如果不考虑存取时间与处理时间重叠时所用的方法, 则 **READ** 语句的概念没有变化。在命令语句 2 (如果指定的话) 执行之前, 记录对目标程序是可用的; 如果未指明命令语句 2, 则在 **READ** 语句之后的任一语句执行之前, 记录对目标程序也是可用的。

(6) 当一个文卷的逻辑记录用多个记录描述来描述时, 这些记录自动地共享同一个存储区, 这便等价于对该区隐含地重定义。在 **READ** 语句执行完毕, 处于当前数据记录范围之外的任何数据项的内容无定义。

(7) **INTO** 短语可以在 **READ** 语句中指定, 条件是:

a. 如果只有一个记录描述款从属于文卷描述款; 或

b. 如果所有与文卷名 1 相关的记录名以及标识符引用的数据项描述了一个组项或一个字符型初等项。

(8) 执行带有 **INTO** 短语的 **READ** 语句的结果等价于按指定顺序应用下列规则的结果:

a. 执行一条不带有 **INTO** 短语的同样的 **READ** 语句;


b. 根据不带 **CORRESPONDING** 短语的 **MOVE** 语句的规则把当前记录从记录区送到标识符指定的区域中。当前记录长度由 “**RECORD** 子句” 中指明的规则确定。

如果 **READ** 语句执行不成功, 则不执行隐含的 **MOVE** 语句。在记录读入之后, 传送到数据项之前对与标识符 1 关联的下标进行求值。无论在记录区还是在标识符 1 引用的数据项中, 该记录都是可用的。

(9) 在执行 **READ** 语句期间, 如果识别出一个卷或一个单位的末端或一个卷或单位中不包含逻辑

记录且并没有到达该文卷的逻辑末端，则执行如下操作：

- a. 标准的结束卷或单位的标号过程。
- b. 对换卷或单位。改变当前指针以指向文卷中现存的下一卷或单位。
- c. 标准的开始卷或单位的标号过程。

(10) 如果文卷位置指示符指出不存在下一逻辑记录， 则按指定顺序执行下列动作：

a. 把赋给文卷位置指示符的值放入与文卷名 1 有关的 I-O 状态中用以指明末端条件（见 1.3.5I-O 状态）。

b. 如果产生末端条件的语句中指明了 AT END 短语，则控制转向 AT END 短语中出现的命令语句 1。不执行与文卷名 1 相关的任何 USE AFTER STANDARD EXCEPTION 过程。

c. 如果未指明 AT END 短语，必须有一 USE AFTER STANDARD EXCEPTION 过程与此文卷名 1 相联系，并执行这个过程。从过程返回后就转向 READ 语句后的下一可执行语句。

当发生了末端条件时，READ 语句的执行就是不成功的。

(11) 如果 READ 语句执行期间未发生末端条件，就跳过 AT END 短语（如果指定的话），并执行下列动作：

a. 对文卷位置指示符赋值，并改变与文卷名 1 相关的 I-O 状态之值。

b. 如果存在的例外条件不是末端条件，根据 USE 语句的规则，在对文卷名 1 适用的 USE AFTER EXCEPTION 过程执行之后进行控制转移（见 4.6USE 语句）。

c. 如果不存在例外条件，就使记录在记录区中成为可用的且执行由 INTO 短语引起的隐含的传送。控制转向 READ 语句结束处或命令语句 2（如果指定的话）。在后一情况下，根据命令语句 2 中各语句的规则执行下去。如果执行到一条产生明显控制转移的过程分支或条件语句，则根据该语句的规则进行控制转移；否则，命令语句 2 执行后，控制转向 READ 语句结束处。

(12) READ 语句执行不成功之后的相关记录区的内容是无定义的。对文卷位置指示符赋值以指明未建立有效的下一记录。

(13) 如果读入记录的字符位置数小于文卷的记录描述款中指定的最小长度，则记录区中位于最后一个有效字符右边的那部分是无定义的。如果读入记录的字符位置数大于文卷的记录描述款中指定的最大长度，对记录右边截断以满足最大长度。在这两种情况下，READ 语句是成功的且对 I-O 状态赋值以指明发生了记录长度冲突（见 1.3.5I-O 状态）。

(14) END-READ 短语限定了 READ 语句的作用域（见 GB/T 4092.1 中 6.6.4.3 语句的作用域）。

## 4.5 REWRITE 语句

### 4.5.1 功能

REWRITE 语句逻辑地替换存在于海量存储文卷中的一个记录。

### 4.5.2 一般格式

REWRITE 记录名 1 [FROM 标识符 1]

### 4.5.3 语法规则

(1) 记录名 1 和标识符 1 不能指称同一个存储区。

(2) 记录名 1 是数据部的文卷节中一个逻辑记录的名并且可以受限。

### 4.5.4 一般规则

(1) 在执行该语句时，与记录名 1 相关联的文卷必须是海量存储文卷并且以 I-O 方式打开（见 4.3 中的 OPEN 语句）。

(2) 在执行 REWRITE 语句之前，执行相应文卷的最末一个输入-输出语句必须是成功执行的 READ 语句。海量存储控制系统（MSCS）逻辑上替换了由 READ 语句存取的记录。



(3) 由一个成功执行的 **REWRITE** 语句所释放的逻辑记录在记录区中不再是可用的,除非与记录名 1 相关的文卷在 **SAME RECORD AREA** 子句中命名。该逻辑记录对该程序仍然是可用的,但它是作为与相关输出文卷同一 **SAME RECORD AREA** 子句中出现的其他文卷的一个记录。这个逻辑记录对于和记录名 1 相关的文卷而言也是可用的。

(4) 执行带有 **FROM** 短语的 **REWRITE** 语句等价于按指定顺序执行下列语句:

a. 根据 **MOVE** 语句的规则执行语句。

**MOVE** 标识符 1 **TO** 记录名 1。

b. 执行不带 **FROM** 短语的同样的 **REWRITE** 语句。

(5) **REWRITE** 语句执行后,标识符 1 引用的区域的信息是可用的,即使记录名 1 引用的区域中的信息并不可用。

(6) 文卷位置指示符不受执行 **REWRITE** 语句的影响。

(7) 执行 **REWRITE** 语句导致更新与记录名 1 相关的文卷名的 **I-O** 状态之值(见 3.1.5 **I-O** 状态)。

(8) **REWRITE** 语句的执行使一个逻辑记录释放给操作系统。

(9) 如果记录名 1 引用的记录中的字符位置数不等于被替换的记录中的字符位置数,则 **REWRITE** 语句的执行是不成功的,且不进行更新操作,记录区中内容不受影响,且与记录名 1 相关的文卷的 **I-O** 状态被赋值以指明条件的原因(见 1.3.5 **I-O** 状态)。

#### 4.6 **USE** 语句

##### 4.6.1 功能

**USE** 语句指明输入-输出错误处理用的过程,这些过程是输入-输出控制系统提供的标准过程之外的过程。

##### 4.6.2 一般格式

```

USE AFTER STANDARD { EXCEPTION
                    ERROR
PROCEDURE ON { {文卷名 1}
              INPUT
              OUTPUT
              I-O
              EXCEPT

```

##### 4.6.3 语法规则

(1) 当 **USE** 语句出现时,必须紧接在过程部的申述节中的节首之后,并且 **USE** 语句出现其中的句子只包含一个语句。该节的其余部分必须由零个、一个或多个用于定义要使用的过程的过程段组成。

(2) **USE** 语句本身从不执行;它只是定义了要求执行 **USE** 过程的条件。

(3) 在 **USE** 语句中出现的文卷名 1 不能同时引起要求执行一个以上的 **USE** 过程。

(4) **ERROR** 和 **EXCEPTION** 是同义字,并且可以交换使用。

(5) 在 **USE** 语句中隐式或显式引用的文卷无须全都具有相同的组织方式或存取方式。

(6) **INPUT**、**OUTPUT**、**I-O** 短语在给定的过程部的申述部分里只能分别出现一次。

##### 4.6.4 一般规则

(1) 一个 **COBOL** 源程序中可以包括申述过程,而不管它包含另一程序或包含于另一程序之中。程序执行过程中,如果申述前的 **USE** 语句中描述的条件发生,则调用该申述。在一分别编译的程序执行时,如果申述前的 **USE** 语句中描述的条件发生,则该分别编译的程序中只有一个申述被调用。这里的分别编译的程序包含引起限制条件的语句。如果分别编译的程序中不存在限定申述,则不执行任何申述。

(2) 在申述过程中,不能引用非申述过程。

(3) 与 **USE** 语句关联的过程名可以在不同申述节或仅有一个 **PERFORM** 语句的非申述节中引用。

(4) 如果显式指明文卷名 1, 则其他 **USE** 语句不能应用文卷名 1。

(5) 输入输出操作执行不成功时执行标准输入输出例外例行程序, 执行完毕再由输入输出控制系统执行与 **USE** 语句相关的过程。除非 **AT END** 短语处于优先地位。关于何时执行这些过程的规则如下:

a. 如果指定文卷名 1, 则发生 **USE** 语句中描述的状态时执行相关的过程。

b. 如果指定 **INPUT**, 则对以输入方式打开的文卷或正在以输入方式打开过程中的文卷发生了 **USE** 语句中描述的条件之后, 执行相关的过程, 对由指定同一条件的另一 **USE** 语句中的文卷名 1 引用的那些文卷则除外。

c. 如果指定 **OUTPUT**, 则对以输出方式打开的文卷或在处理时以输出方式打开的文卷发生了 **USE** 语句中描述的条件之后, 执行相关的过程, 对由指定同一条件的另一 **USE** 语句中的文卷名 1 引用的那些文卷则除外。

d. 如果指定 **I-O**, 则对以 **I-O** 方式打开的文卷或在处理时以 **I-O** 方式打开的文卷发生了 **USE** 语句中指定的条件之后, 执行相关的过程, 对由指明同一条件的另一 **USE** 语句中的文卷名 1 引用的那些文卷则除外。

e. 如果指明了 **EXTEND**, 则对以扩展方式打开的文卷或在处理时以扩展方式打开的文卷发生了 **USE** 语句中指定的条件后, 执行相关的过程, 对由指明同一条件的另一 **USE** 语句中的文卷名 1 引用的那些文卷则除外。

(6) **USE** 过程执行过后, 控制转到输入输出控制系统的调用程序。如果 **I-O** 状态未指明关键的输入输出错误, 则输入输出控制系统就把控制转回到其执行引起例外情况的输入输出语句之后下一可执行的语句。如果 **I-O** 状态指出关键错误, 则由实现者决定采取什么动作 (见 1.3.5 **I-O** 状态)。

(7) 在 **USE** 过程中, 不能执行这样的语句, 它会调用到以前已调用过的且尚未把控制转回调用程序的 **USE** 过程。

## 4.7 WRITE 语句

### 4.7.1 功能

**WRITE** 语句释放一个逻辑记录到输出文卷, 它还能用于逻辑页中行的垂直定位。

### 4.7.2 一般格式

**WRITE** 记录名 1 [**FROM** 标识符 1]

$$\left[ \begin{array}{l} \left\{ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right\} \text{ADVANCING} \left\{ \begin{array}{l} \text{标识符 2} \\ \text{整数 1} \end{array} \right\} \left[ \begin{array}{l} \text{LINE} \\ \text{LINES} \end{array} \right] \\ \text{PAGE} \end{array} \right]$$

```

AT { END-OF-PAGE } 命令语句 1
EOP
NOT AT { END-OF-PAGE } 命令语句 2
EOP
END-WRITE

```

### 4.7.3 语法规则

(1) 记录名 1 和标识符 1 不能指称同一存储区。

(2) 记录名 1 是数据部的文卷节中逻辑记录的名字, 并且可以受限。

(3) 当与一个记录相关联的包含有 **LINAGE** 子句的文卷描述款相关的文卷上时, 不能指明 **ADVANCING** 助忆名 1 短语。

(4) 标识符 2 必须引用整数数据项。

(5) 整数 1 可为正或零, 但不能为负。

(6) 当指明了助忆名 1 时, 该名字与实现者指定的特定的要点相关联。该助忆名 1 在环境部分 **SPECIAL-NAMES** 段中定义。

(7) 短语 **ADVANCING PAGE** 和 **END-OF-PAGE** 不能同时出现在同一 **WRITE** 语句中。

(8) 如果指定了 **END-OF-PAGE** 或 **NOT END-OF-PAGE** 短语, 则在相关文卷的文卷描述款中必须指明 **LINAGE** 子句。

(9) **END-OF-PAGE** 和 **FOR** 这两个字是等价的。

#### 4.7.4 一般规则

(1) 在执行这个语句时, 与记录名 1 有关的文卷必须是以 **OUTPUT** 方式打开的 (见 4.3 **OPEN** 语句)。

(2) 执行 **WRITE** 语句所释放的逻辑记录, 在该记录区中不再可用。

与记录名 1 有关的文卷名在一个 **SAME RECORD AREA** 子句中被命名时则为例外。该逻辑记录对该程序也是可用的。但这时它是作为与相关输出文卷同一 **SAME RECORD AREA** 子句中出现的其它文卷的一个记录。该逻辑记录对于和记录名 1 相关的文卷同样也是可用的。

(3) 带有 **FROM** 短语的 **WRITE** 语句的执行结果等价于执行:

a. 按照对 **MOVE** 语句所指明的规则执行;

**MOVE** 标识符 1 **TO** 记录名 1

接着执行

b. 不带 **FROM** 短语的同一个 **WRITE** 语句。

(4) 在 **WRITE** 语句执行完毕后, 尽管由记录名 1 引用的区中的信息是不可用的, 但由标识符 1 引用的记录区中的信息却是可用的, **SAME RECORD AREA** 子句中出现的其它文卷的一个记录。该逻辑记录对于和记录名 1 相关的文卷同样也是可用的。

(5) 文卷位置指示符不受 **WRITE** 语句执行的影响。

(6) 执行 **WRITE** 语句引起更新与文卷名 1 相关的文卷的 **I-O** 状态的值 (见 1.3.5 **I-O** 状态)。

(7) **WRITE** 语句的执行把一逻辑记录释放给操作系统。

(8) 记录名 1 引用的记录中的字符位置数不能大于和记录名 1 有关的文卷名相关的 **RECORD IS VARYING** 子句中允许的字符位置的最大数, 也不能小于上述字符位置的最小数。无论哪一种情况下, **WRITE** 语句的执行不成功, 不进行 **WRITE** 操作, 记录区的内容不受影响, 并对与记录名 1 有关的文卷的 **I-O** 状态赋值以指明条件的原因 (见 1.3.5 **I-O** 条件)。

(9) 执行带有 **NOT END-OF-PAGE** 短语的 **WRITE** 语句时, 如果不发生页结束条件, 则控制在适当时刻转到命令语句 2, 这些时刻是:

a. 如果写语句执行成功, 在写过记录并更新过与记录名 1 相关的文卷名的 **I-O** 状态之后。

b. 当 **WRITE** 语句执行不成功, 则在更新过与记录名 1 有关的文卷名的 **I-O** 状态并且执行过应用于与记录名 1 相关的文卷名的 **USE AFTER STANDARD EXCEPTION PROCEDURE** 语句中指定的过程 (如果有的话) 之后。

(10) **END-WRITE** 短语限定了 **WRITE** 语句的作用域 (见 GB/T 4092.1 中 6.6.4.3 语句的作用域)。

(11) 根据执行 **WRITE** 语句时建立文卷的顺序, 建立顺序文卷的后继关系。这个关系不会改变, 但往一文卷末端增加记录时就会改变。

(12) 如果顺序文卷以扩展方式打开, **WRITE** 语句的执行将记录加到文卷末端, 就好像文卷是以输出方式打开的一样。如果文卷中有记录, 则执行带有 **EXTEND** 短语的 **OPEN** 语句之后所写的第一个记录是文卷中最后一个记录的后继记录。

(13) 当试图写到顺序文卷外部定义的边界之外时, 便存在例外条件并且不影响记录区的内容, 进行如下的动作:

- a. 设定与记录名 1 相关文卷的 **I-O** 状态的值指出越界 (见 1.3.5 **I-O** 状态)。
- b. 如果对与记录名 1 有关的文卷显式或隐式地指明有 **USE AFTER STANDARD EXCEPTION** 申述, 便执行这个申述过程。
- c. 如果对与记录名 1 有关的文卷没有显式或隐含地指明有 **USE AFTER STANDARD EXCEPTION** 申述, 则结果无定义。

(14) 识别出卷或单位末端并且没有超过文卷的外部定义边界时, 执行如下操作:

- a. 标准结束卷或单位的标号过程。
- b. 对换卷或单位。更新当前卷指针指出文卷现存的下一卷或单位。
- c. 标准开始卷或单位的标号过程。

(15) **ADVANCING** 短语和 **AFTER ADVANCING 1 LINE** 短语两者允许控制打印页上每行的垂直定位。如果没有使用 **ADVANCING** 短语, 则由实现者提供自动推进, 其作用和用户指出了 **AFTER ADVANCING 1 LINE** 完全一样。如果使用了 **ADVANCING** 短语, 则提供如下推进规则:

- a. 如果整数 1 或标识符 2 引用的数据项之值为正, 则表示打印页上推进的行数等于该值。
- b. 如果标识符 2 引用的数据项之值为负, 则结果无定义。
- c. 如果整数 1 或标识符 2 引用的数据项之值为零, 则表示对打印页不进行重定位。

e. 如果使用 **BEFORE** 短语, 则在按照上述规则 a、b、c 推进打印页前呈现此行。

f. 如果指明了 **AFTER** 短语, 则在按照上述规则 a、b、c 推进打印页后呈现此行。

g. 如果指明 **PAGE** 且 **LINAGE** 子句在相关的文卷描述款中指明, 则该记录出现在该设备里定位到下一个逻辑页之前或之后的 (取决于所用的短语) 逻辑页上。如在 **LINAGE** 子句中指出的那样, 重定位到下一逻辑页上可写的第一行。

h. 如果指明了 **PAGE** 且 **LINAGE** 子句在相关的文卷描述款中指明, 则该记录出现在该设备重定位到下一个物理页之前或之后的 (取决于所用的短语) 逻辑页上。按照实现者定义的方法重定位到下一物理页。如果物理页连同特定的设备无意义, 则推进由实现者提供, 其作用和用户指出的 **BEFORE** 或 **AFTER** (取决于所用的短语) **ADVANCING 1 LINE** 相同。

(16) 在带有 **END-OF-PAGE** 短语的 **WRITE** 语句的执行期间如果到达了打印页的逻辑末端, 则执行 **END-OF-PAGE** 短语中所指定的命令语句 1。逻辑末端在与记录名 1 相关的 **LINAGE** 子句中找出来。

(17) 每当执行一个给出的带有 **END-OF-PAGE** 短语的 **WRITE** 语句而引起打印或在页体的页末区中留出空白时, 便达到页的末端条件。当指出了执行这样的 **WRITE** 语句使 **LINAGE-COUNTER** 等于或超过由整数 2 所指出的值或者由数据名 2 引用的数据项值时, 便产生这个条件。在这种情形下, 执行 **WRITE** 语句。然后执行 **END-OF-PAGE** 短语中的命令语句 1。

每当执行一个给定的 **WRITE** 语句(带或不带 **END-OF-PAGE** 短语的)而在当前的页体中又不能完全容纳时, 则总是产生自动页溢出条件。

当执行 **WRITE** 语句, 导致 **LINAGE-COUNTER** 超过由整数 1 或由 **LINAGE** 子句中数据名 1 引用的数据项值时则发生这种情况。在这种情形, 在 **LINAGE** 子句中指出的设备重定位到下一个逻辑页可写的第一行之前或之后(这取决于所使用的短语), 记录出现在该设备上。如果指明了 **END-OF-PAGE** 子句中的命令语句, 则它在记录被写以后和设备被重新定位以后才执行。

执行给定 **WRITE** 语句引起 **LINAGE-COUNTER** 同时超过 **LINAGE** 中的整数 2 或由数据名 2 引用的数据项之值和整数 1 或数据名 1 引用的数据项之值, 则发生页溢出条件。

#### 附加说明:

本标准由中华人民共和国机械电子工业部提出。

本标准由南京大学负责起草。

本标准主要起草人钱树人、王静英、冯惠、段祥。

本标准由 1983 年 12 月首次发布, 1992 年 8 月第一次修订。