第５章　ファイル入出力

指導目標

本章では、ファイル入出力機能とはどのようなものであるかを学習する。

ファイル入出力機能は、オペレーティング・システムのデータ管理の役割とＣＯＢＯＬの機能（文法）を理解させることが望ましい。

市販のＣＯＢＯＬ解説書、および、各々のメーカのマニュアル等は、個々の文法について詳細に説明されているが、はじめてＣＯＢＯＬの入出力機能を使用する場合に問題となるのは、どこにどのような記述（定義）するのかが一目でわからないことである。

また、一般的な「データ管理（ファイルシステム）」の基本的な知識がないと、ＣＯＢＯＬの入出力機能を理解できない。
最近は、ＰＣやＵＮＩＸなどのオペレーティング・システムを中心に「ハードウェア」、「ソフトウェア」を学んでから、プログラミングとしてＣＯＢＯＬ言語に入った場合、ＰＣやＵＮＩＸのファイルシステムでは、データ管理機能の「ファイル編成」の概念が理解できず索引編成、相対編成等でつまずくケースが見受けられる。
また、逆に汎用コンピュータから学んだ者は、ＰＣやＵＮＩＸのファイルシステムの単純な機能に戸惑うケースもある。

これらを考慮して、ＣＯＢＯＬプログラムにおける入出力定義の全体像を解説してファイル入出力とオペレーティング・システムのデータ管理機能を理解、整理させる。

内容のあらまし

内　　容	説　明	議　論	机上実習	計算機実習
COBOL入出力	COBOL入出力に必要な記述形式の全体像を説明する。	ﾌﾟﾛｸﾞﾗﾑで入出力に必要な情報とは何かを考えさせる	図解を用いて概念を整理させる。	簡単な例題を実行してみる
ファイルの定義	SELECT句を中心にＯＳとの関連を説明する。	ﾌﾟﾛｸﾞﾗﾑと物理ﾌｧｲﾙとの関連付けの必要性を議論する		JCLとの関連を実習する。
ﾚｺｰﾄﾞとﾌﾞﾛｯｸの定義	FD句とﾚｺｰﾄﾞ記述について説明する。	ﾌｧｲﾙとﾚｺｰﾄﾞの構造について議論する。	ﾚｺｰﾄﾞ数とﾌﾞﾛｯｸｻｲｽﾞでﾌｧｲﾙ領域の計算を行う。	
入出力命令	COBOLの入出力命令文の全体像を整理させる。	ﾌｧｲﾙのｱｸｾｽに必要は命令文にはどんなものが必要であるか考えさせる		ﾌｧｲﾙの概念とﾃﾞｨｽｸ、ﾌﾟﾘﾝﾀ装置等の関係を理解させる
入力ファイル	入力ファイルの概念と使用できる命令文について説明する。	入力ﾌｧｲﾙに必要な命令文と利用方法を議論する	入力装置の種類や特徴を整理させる。	
出力ファイル	出力ファイルの概念と使用できる命令文について説明する。	出力ﾌｧｲﾙに必要な命令文と利用方法を議論する	出力装置の種類や特徴を整理させる。	
入出力両用ファイル	入出力両用ファイルの概念と使用できる命令文用について説明する。	入出力両用ﾌｧｲﾙの更新について命令の機能と実行順序の重要性を考えさせる。	入出力装置の種類や特徴を整理させる。	
ファイル編成	順、相対、索引ファイルについて定義と使用方法を説明する。	ﾌｧｲﾙに必要な概念と物理ﾌｧｲﾙの関わりについて考えさせる。	ﾌｧｲﾙの概念とﾃﾞｰﾀ構造を整理させる。	応用事例を実習させる。
呼出し法	順、乱、動的呼出し法について使用方法を説明する。	ﾌｧｲﾙのｱｸｾｽの応用について事例で考えさせる。		

内　　容	説　明	議　論	机上実習	計算機実習
ﾌｧｲﾙｽﾃｰﾀｽ	ﾌｧｲﾙｽﾃｰﾀｽの基本的な意味を学ぶ。	ﾌｧｲﾙｽﾃｰﾀｽの意味からｱｸｾｽ結果の情報との関連を考えさせる。		簡単な事例でﾌｧｲﾙｽﾃｰﾀｽの状況を実習させる。
入出力例外／ｴﾗｰ宣言	EXCEPTION／ERROR処理の動作について学ぶ	ｴﾗｰ処理の実行順序の考え方を事例で考える	ｴﾗｰの種類を整理させる。	ﾃﾞｰﾀ管理機能のｴﾗｰ時の具体例を実習させる。
順ファイル	COBOLの順ﾌｧｲﾙの概念とﾃﾞｰﾀ構造やその処理方法を充分に指導する。	順ﾌｧｲﾙの応用事例を挙げさせてﾌｧｲﾙの基本適な操作を議論する		磁気ﾃｰﾌﾟ、ﾃﾞｨｽｸ、ﾌﾟﾘﾝﾀ等で入出力操作を実習させる

５．１　ＣＯＢＯＬ入出力

ＣＯＢＯＬプログラムにおける入出力定義の全体像を解説する。

入出力定義の概要

```
IDENTIFICATION  DIVISION.

ENVIRONMENT  DIVISION.

INPUT-OUTPUT  SECTION.
FILE-CONTROL.
    SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語                          ① ②
            ORGANIZATION  IS  ﾌｧｲﾙ編成                           ③
            ACCESS  MODE  IS  呼出し法                           ④
            FILE  STATUS  IS  ﾌｧｲﾙｽﾃｰﾀｽ.
     :
I-O-CONTROL.
    RERUN  ON  ･･･                                               ⑤
    SAME  RECORD  AREA  FOR  ﾌｧｲﾙ名  ･･
     :

DATA  DIVISION.

FILE  SECTION.
FD  ﾌｧｲﾙ名
      BLOCK  CONTAINS  整数2  RECORDS
01  ﾚｺｰﾄﾞ記述項                                                  ⑥
    03  ﾚｺｰﾄﾞｷｰ  PIC  X(10).                                      ⑦


WORKING-STORAGE  SECTION.
01  ﾌｧｲﾙｽﾃｰﾀｽ      PIC  XX.                                     ⑧
01  相対ﾚｺｰﾄﾞ番号  PIC  99999.                                   ⑨

PROCEDURE  DIVISION.

DECLARATIVES.
  USE AFTER STANDARD EXCEPTION/ERROR PROCEDURE ON  ﾌｧｲﾙ名/INPUT/･･
                                                                 ⑩
END  DECLARATIVES.

    OPEN  入出力ﾓｰﾄﾞ  ﾌｧｲﾙ名                                      ⑪

    READ  ﾌｧｲﾙ名  AT END/INVALID KEY                             ⑫
    WRITE  ﾚｺｰﾄﾞ名  INVALID KEY
    REWRITE  ﾚｺｰﾄﾞ名  INVALID  KEY
    DELETE  ﾌｧｲﾙ名  INVALID KEY
    START  ﾌｧｲﾙ名  KEY  IS EQUAL/GREATER/LESS  ﾃﾞｰﾀ名  INVALID KEY

    CLOSE  ﾌｧｲﾙ名
```

入出力定義の説明

①ファイル名

ＣＯＢＯＬプログラム上の論理ファイル名を定義する。

②作成者語

ＣＯＢＯＬプログラムの論理ファイルとオペレーティング・システムのデータ管理（ファイルシステム）の物理ファイルとの関連付けを定義する。

③ファイル編成

ＣＯＢＯＬがサポートしているファイル編成を定義する。

SEQUENTIAL	－	順ファイル（順次ファイル）
INDEXED	－	索引ファイル
RELATIVE	－	相対ファイル

④呼出し方式

ファイルの呼出し方式を定義する。

SEQUENTIAL	－	順呼出し法
RANDOM	－	乱呼出し法
DYNAMIC	－	動的呼出し法

⑤入出力制御

チェックポイントをとるタイミングや、入出力バッファー領域の共用を定義する。

⑥レコード記述

入出力命令で指定するレコード名と、そのデータ構造とデータ属性を定義する。
索引ファイルの場合は、レコードキーを含めなければならない。

⑦レコードキー

索引ファイルのレコードキーの位置とデータ属性を定義する。

⑧ファイルステータス

ＣＯＢＯＬでの入出力の実行結果（状況キー）を返す領域（２ﾊﾞｲﾄ）を定義する。
基本的な意味は、次のとおり。「？」には補助情報がコード化されている。

０？	正常終了
１？	ファイルの終了（AT　END状態）
２？	無効なキー（INVALID　KEY状態）
３？	永続エラー
９？	その他エラー

⑨相対レコード番号

相対ファイルを使用する場合、WORKING-STORAGE　SECTIONに相対レコード番号の領域を数字項目（整数）で定義する。

⑩入出力例外／エラー処理

入出力例外やエラーが発生した場合に実行する手続きを宣言する。
どのファイルを対象にするかは、ファイル名や入出力モードで指定する。

⑪入出力モード

ファイルの処理モードを指定する。

INPUT	－	入力ファイル
OUTPUT	－	出力ファイル
I-O	－	入出力両用ファイル
EXTEND	－	拡張ファイル（出力ファイル）

⑫入出力命令文

入出力の動作を指定する。OPEN命令文は最初に実行して、その後、ファイル編成と入出力モードに対応した命令文（READ、WRITE、REWRITE、DELETE、START）を実行する。最後はCLOSE文を実行する。

C言語の入出力定義の概要

```
#include  <stdio.h>

main(){

FILE  *stream;
char  buffer[250];

  stream  =  fopen(ﾌｧｲﾙ名,ｵｰﾌﾟﾝﾓｰﾄﾞ);

  size  =  fread(buffer,ｻｲｽﾞ,count,stream);

  fseek(stream,offset,origin);

  size  =  fwrite(buffer,ｻｲｽﾞ,count,stream);

  fprintf(stream,書式制御,argument,..);

  fclose(stream);

}
```

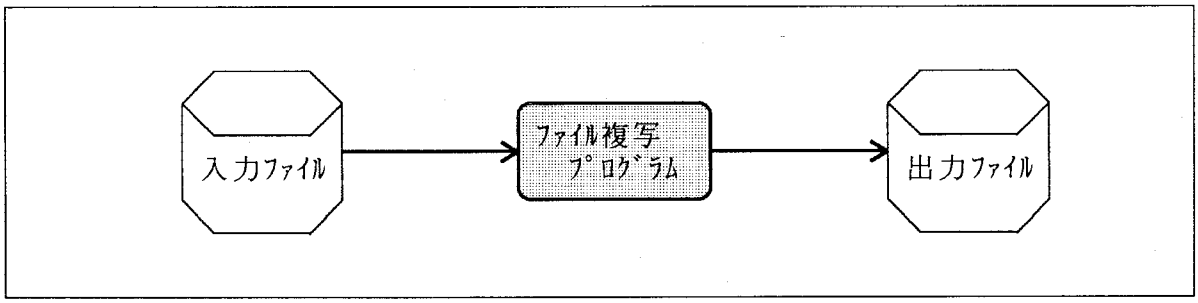
ＦＯＲＴＲＡＮの入出力定義の概要

```
    CHARACTER  buffer*ｻｲｽﾞ

    OPEN(UNIT=装置番号,ACCESS='DIRECT/SEQUENTIAL',FILE=ﾌｧｲﾙ名,
+       FORM='FORMATTED/UNFORMATTED',STATUS='OLD/NEW/UNKNOWN')

    READ(UNIT=装置番号,FMT=書式,END=終了番号,ERR=ｴﾗｰ番号,
+       REC=ﾚｺｰﾄﾞ番号)  buffer

    WRITE(UNIT=装置番号,FMT=書式,ERR=ｴﾗｰ番号,IOSTAT=戻り値,
+       REC=ﾚｺｰﾄﾞ番号)  buffer

    BACKSPACE(UNIT=装置番号,ERR=ｴﾗｰ番号)

    REWIND(UNIT=装置番号,ERR=ｴﾗｰ番号)

    INQUIRE(UNIT=装置番号,EXIST=...)

    ENDFILE(UNIT=装置番号,ERR=ｴﾗｰ番号)

    CLOSE(UNIT=装置番号,ERR=ｴﾗｰ番号,STATUS='KEEP/DELETE'))

    END
```

重要な解説部分は、次の通り。

- 装置番号の考え方と物理ﾌｧｲﾙとの関連付け
- ﾌｧｲﾙ編成としてのDIRECTとSEQUENTIALについて
- ﾚｺｰﾄﾞ形式としてFORMATTEDとUNFORMATTEDについて
- 入出力ﾓｰﾄﾞとしてOLD/NEW/UNKNOWN/SCRATCHについて

次の例は、ファイルの複写（コピー）を行うＣＯＢＯＬ原始プログラムである。




```
IDENTIFICATION DIVISION.
PROGRAM-ID.   複写.
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER    計算機名.
OBJECT-COMPUTER    計算機名.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT 入力ﾌｧｲﾙ名   ASSIGN TO 作成者語1.
    SELECT 出力ﾌｧｲﾙ名   ASSIGN TO 作成者語1.
```

```
DATA DIVISION.
FILE SECTION.
FD  入力ﾌｧｲﾙ名
    LABEL RECORD OMITTED.
01  入力ﾚｺｰﾄﾞ          PICTURE  X(80).

FD  出力ﾌｧｲﾙ名
    LABEL RECORD OMITTED.
01  出力ﾚｺｰﾄﾞ          PICTURE  X(80).

WORKING-STORAGE SECTION.
01  件数     PIC 9(5)  VALUE ZERO.
```

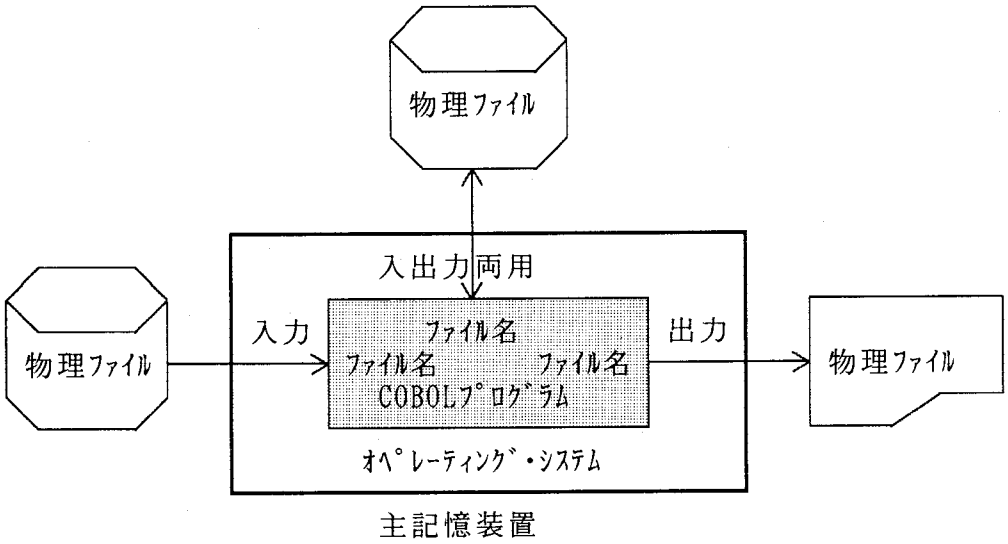
```
PROCEDURE DIVISION.
HAJIME.
    OPEN INPUT  入力ﾌｧｲﾙ名, OUTPUT 出力ﾌｧｲﾙ名.
KURIKAESI.
    READ 入力ﾌｧｲﾙ名 AT END GO TO OWARI.
    WRITE 出力ﾚｺｰﾄﾞ FROM 入力ﾚｺｰﾄﾞ.
    COMPUTE  件数  =  件数 + 1.
    GO TO KURIKAESI.
OWARI.
    CLOSE 入力ﾌｧｲﾙ名, 出力ﾌｧｲﾙ名.
    DISPLAY   件数   " ﾚｺｰﾄﾞ複写しました。".
    STOP RUN.
```

５．２　ファイルの定義

ＳＥＬＥＣＴ文でＣＯＢＯＬプログラムのファイル（論理）の定義と、実行時の外部ファイル（物理）の関連を定義する。

```
ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT [OPTIONAL] ﾌｧｲﾙ名1  ASSIGN TO 作成者語1／定数1
           [RESERVE 整数1 AREAS]
           [ORGANIZATION IS SEQUENTIAL／INDEXED／RELATIVE]
           [ACCESS MODE IS SEQUENTIAL／RANDOM／DYNAMIC]
           [FILE STATUS IS ﾃﾞｰﾀ名2].
```

（１）　ファイル名と物理ファイルとの関係



主記憶装置

（２）　物理ファイルとの関連付け

パソコンの例

```
ENVIRONMENT  DIVISION.
  SELECT  ﾌｧｲﾙ名  ASSIGN  TO  FILENAME.

DATA  DIVISION.
01  FILENAME    PIC  X(64).

PROCEDURE  DIVISION.
    MOVE  "ﾄﾞﾗｲﾌﾞ名:¥ﾃﾞｨﾚｸﾄﾘ名¥ﾌｧｲﾙ名.拡張子"  TO  FILENAME.
    OPEN  INPUT  ﾌｧｲﾙ名
```

汎用コンピュータの例

```
COBOLｿｰｽ･ﾌﾟﾛｸﾞﾗﾑ

ENVIRONMENT  DIVISION.
  SELECT  ﾌｧｲﾙ名  ASSIGN  TO  S-FILENAME.

ｼﾞｮﾌﾞ制御文

//FILENAME  DD  DSN=ﾃﾞｰﾀｾｯﾄ名,DISP=SHR
```

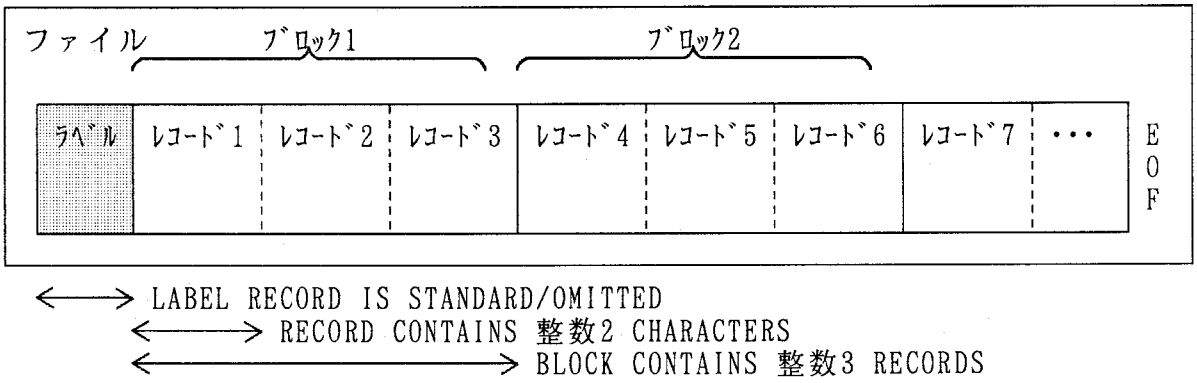
５．３　レコードとブロックの定義

ＦＩＬＥ　ＳＥＣＴＩＯＮでファイルのレコードとブロックの構造を定義する。

```
DATA DIVISION.
FILE SECTION.

FD   ﾌｧｲﾙ名1
     [LABEL  RECORD  IS  STANDARD／OMITTED]
     [BLOCK CONTAINS 整数2 RECORDS／CHARACTERS]
     [RECORD CONTAINS 整数3 CHARACTERS]

 [ﾚｺｰﾄﾞ記述項]
01   ﾚｺｰﾄﾞ名
      [ﾚｺｰﾄﾞｷｰ記述]
```

（１）　ファイル構造との関連

ＦＤ句でファイルのラベル情報、レコードサイズ、ブロックサイズ等を定義する。



（２）　レコード記述

レベル番号でレコードの構造を定義し、レコードに含まれるデータの形式（属性）とサイズをＰＩＣＴＵＲＥ句で定義する。

```
01  ﾚｺｰﾄﾞ名.
  nn  ﾚｺｰﾄﾞｷｰ名   属性/ｻｲｽﾞ
  nn  ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
  nn  ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
  nn  ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
  nn  ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
  nn  ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
  nn  ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
  nn  ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
  nn  ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
```

ﾚｺｰﾄﾞ名はﾚﾍﾞﾙ番号01で定義する。

索引ﾌｧｲﾙの場合は、ﾚｺｰﾄﾞｷｰをﾚｺｰﾄﾞ中に英数字項目で定義する。

注：　nnはレベル番号

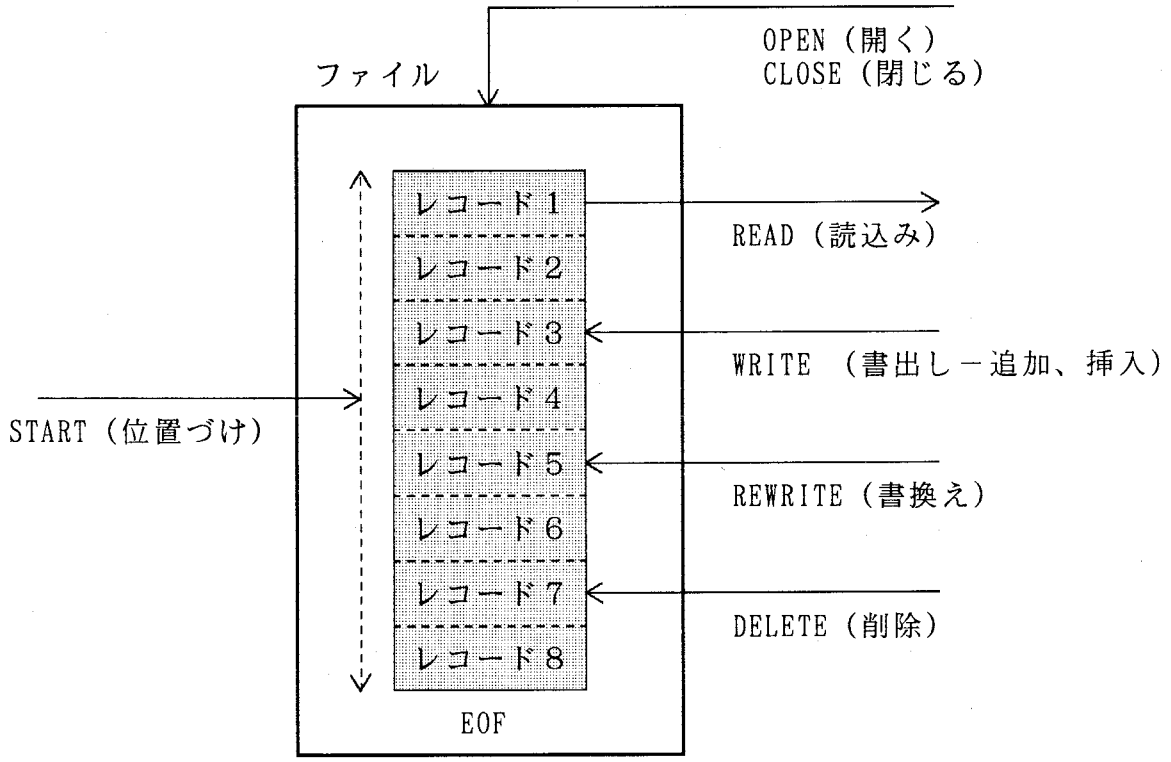
５．４　入出力命令文

ファイル編成及び処理モードで使用できる入出力命令文の関係は、次のとおり。

命令文種類		順ファイル	索引ファイル	相対ファイル
OPEN	ﾌｧｲﾙ名	○	○	○
CLOSE	ﾌｧｲﾙ名	○	○	○
READ	ﾌｧｲﾙ名	○ INPUT	○ INOUT　I-O	○ INPUT　I-O
START	ﾌｧｲﾙ名	×	○ INPUT　I-O	○ INPUT　I-O
WRITE	ﾚｺｰﾄﾞ名	○ OUTPUT	○ OUTPUT　I-O	○ OUTPUT　I-O
DELETE	ﾚｺｰﾄﾞ名	×	○ I-O	○ I-O
REWRITE	ﾚｺｰﾄﾞ名	○ I-O	○ I-O	○ I-O

入出力命令文でファイル名を指定するのか、レコード名を指定するのか注意する。

（１）　入出力命令文と機能



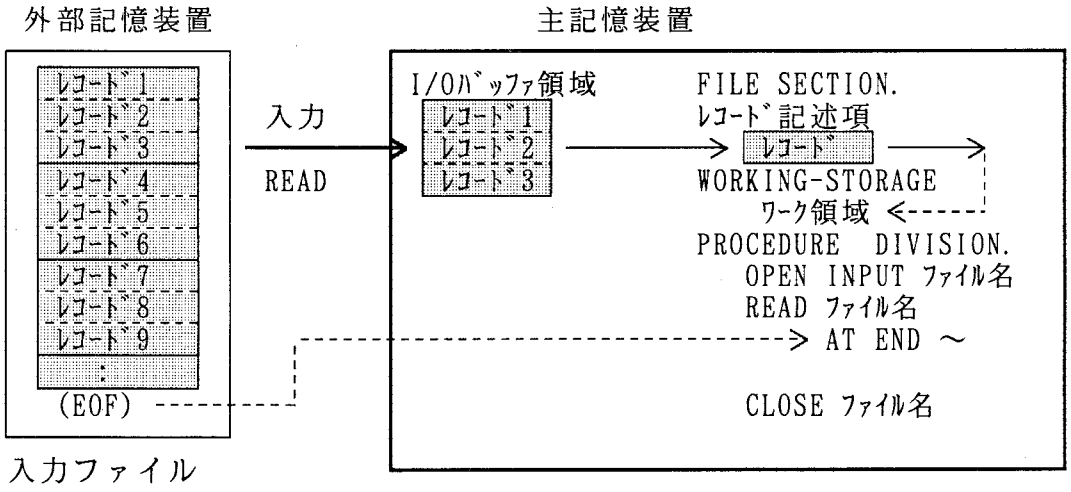
他に、ACCEPT（小入力）とDISPLAY（表示）がある。

５．５　入力ファイル

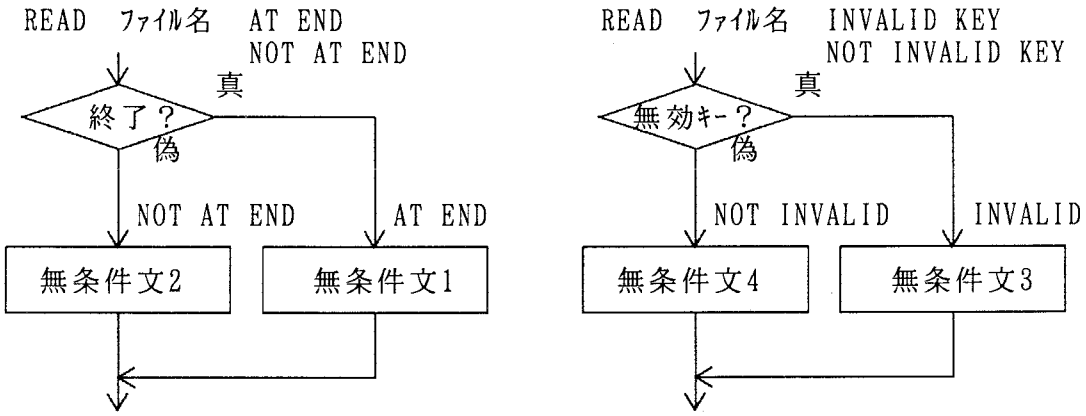
レコードの取り出し（読み込み）を行うファイルを入力ファイルという。

```
OPEN INPUT ﾌｧｲﾙ名

READ ﾌｧｲﾙ名 [NEXT] RECORD [INTO 一意名]
            [AT END 無条件文1]
            [NOT AT END 無条件文2]
[END-READ]

READ ﾌｧｲﾙ名       RECORD [INTO 一意名]
            [INVALID KEY 無条件文3]
            [NOT INVALID KEY 無条件文4]
[END-READ]

 CLOSE  ﾌｧｲﾙ名
```

（１）　入力（READ）の機能



（２）　AT ENDとINVALID KEY条件



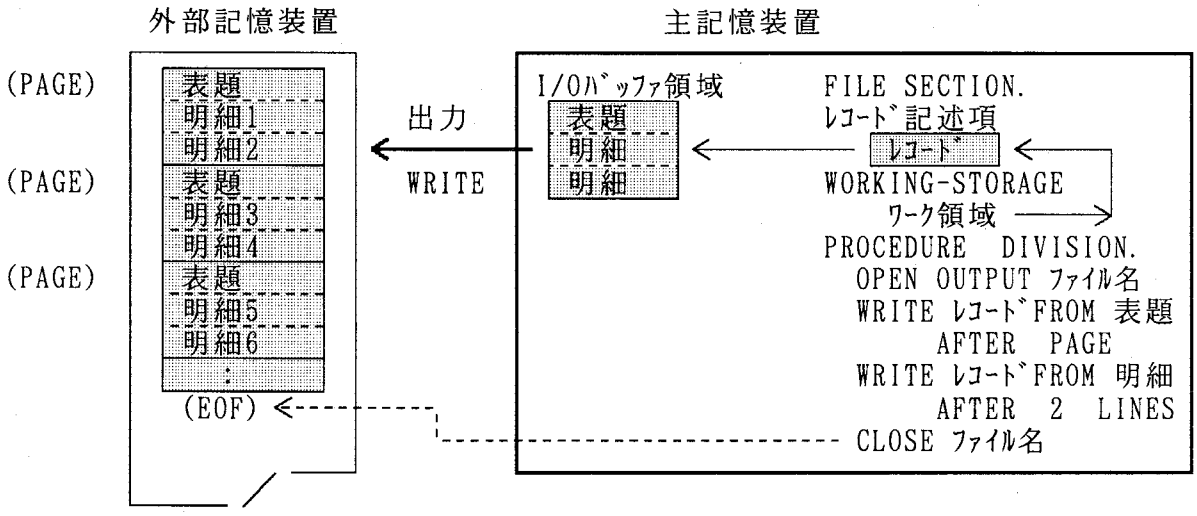
５．６　出力ファイル

レコードの書き出しを行うファイルを出力ファイルという。

```
OPEN OUTPUT ﾌｧｲﾙ名

WRITE ﾚｺｰﾄﾞ名 [FROM 一意名1]
        [BEFORE／AFTER  ADVANCING{ 整数1／一意名2  [LINES] } ]
                                 { PAGE                    }
        [AT  END-OF-PAGE  無条件文1]
        [NOT  AT  END-OF-PAGE  無条件文2]
 [END-WRITE]

WRITE ﾚｺｰﾄﾞ名 [FROM 一意名1]
        [INVALID  KEY  無条件文1]
        [NOT  INVALID  KEY  無条件文2]
 [END-WRITE]

CLOSE  ﾌｧｲﾙ名
```

（１）　出力（WRITE）の機能

指定したﾚｺｰﾄﾞ名のﾃﾞｰﾀを書き出す。

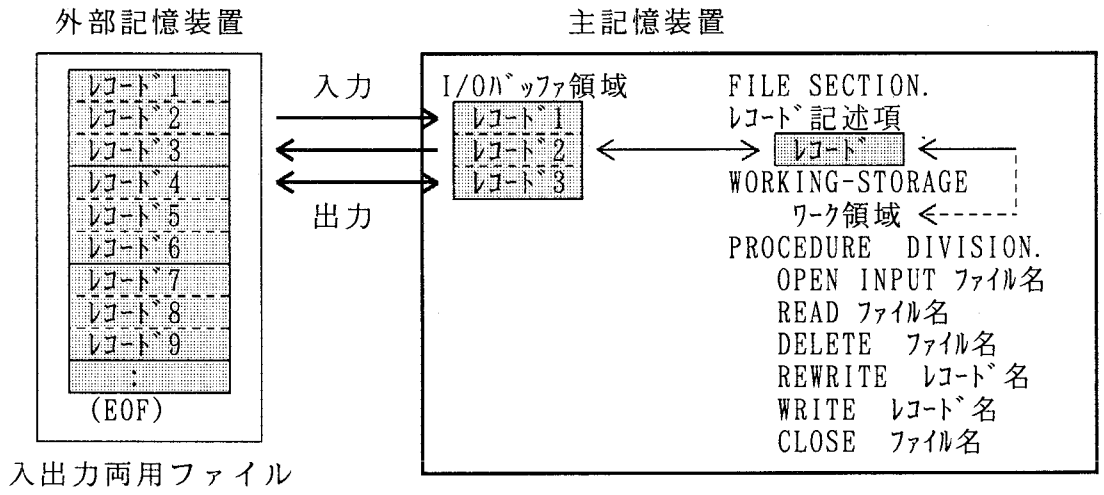


５．７　入出力両用ファイル

レコードの取り出しと更新を行うファイルを入出力両用ファイルという。

```
OPEN I-O ﾌｧｲﾙ名

READ  ﾌｧｲﾙ名  RECORD  [INTO  一意名1]
      [INVALID  KEY  無条件文3]
      [NOT  INVALID  KEY  無条件文4]
 [END-READ]

DELETE  ﾌｧｲﾙ名  RECORD  [INTO  一意名1]
      [INVALID  KEY  無条件文3]
      [NOT  INVALID  KEY  無条件文4]
 [END-DELETE]

REWRITE  ﾚｺｰﾄﾞ名1  [FROM  一意名1]
      [INVALID  KEY  無条件文1]
      [NOT  INVALID  KEY  無条件文2]
 [END-REWRITE]

WRITE  ﾚｺｰﾄﾞ名1  [FROM  一意名1]
      [INVALID  KEY  無条件文1]
      [NOT  INVALID  KEY  無条件文2]
 [END-WRITE]

CLOSE  ﾌｧｲﾙ名
```

（１）　入出力の機能



入出力両用ファイル

５．８　ファイル編成

本を例に、ファイルの編成（organization）と呼出し法（access mode）を具体的に説明する。

（１）　順編成（順次編成）

本には目次もページもないので、本を最初のページから、順に読んで行く。

入出力	順次ﾌｧｲﾙ	相対ﾌｧｲﾙ	索引ﾌｧｲﾙ	～
ａａａａａａａａａａ。	ＸＸＸＸＸＸＸＸＸＸ。	ｙｙｙｙ。ｙｙｙｙｙｙｙｙｙｙ。	ＺＺＺＺＺＺＺＺＺＺ。	～

読出し順序 ―――――――――――――――――→

（ｷｰなし）

必要なﾍﾟｰｼﾞ（情報）を探す場合は、最初から順に各々のﾍﾟｰｼﾞを見ていく。

（２）　相対編成

本のページ番号を基本に、読んで行く。

－目次－ 頁	入出力	順次ﾌｧｲﾙ	相対ﾌｧｲﾙ	索引ﾌｧｲﾙ	～
入出力 1 順次ﾌｧｲﾙ 2 相対ﾌｧｲﾙ 3 索引ﾌｧｲﾙ 4 ： 5	ａａａａａａａａａａ。	ＸＸＸＸＸＸＸＸＸＸ。	ｙｙｙｙ。ｙｙｙｙｙｙｙｙｙｙ。	ＺＺＺＺＺＺＺＺＺＺ。	～
	-1-	-2-	-3-	-4-	～

読出し順序 ↑―――↑―――↑―――↑―――↑

（ｷｰはﾍﾟｰｼﾞ番号）

必要な情報のﾍﾟｰｼﾞが前もって判っていれば、直接そのﾍﾟｰｼﾞを開くことができる。

（３）　索引編成

本の目次（索引）情報を基本に、読んで行く。

－目次－ 頁	入出力	順次ﾌｧｲﾙ	相対ﾌｧｲﾙ	索引ﾌｧｲﾙ	～
入出力 1 順次ﾌｧｲﾙ 2 相対ﾌｧｲﾙ 3 索引ﾌｧｲﾙ 4	ａａａａａａａａａａ。	ＸＸＸＸＸＸＸＸＸＸ。	ｙｙｙｙ。ｙｙｙｙｙｙｙｙｙｙ。	ＺＺＺＺＺＺＺＺＺＺ。	～
	-1-	-2-	-3-	-4-	～

読出し順序 ↑―――↑―――↑―――↑―――↑

（ｷｰは目次、ﾚｺｰﾄﾞ位置は目次についているﾍﾟｰｼﾞを使う）

必要な情報のｷｰﾜｰﾄﾞがあれば、まず目次（索引）をみて、その情報の有無が判別する。

情報を取り出す場合、目次にあるﾍﾟｰｼﾞを使ってそのﾍﾟｰｼﾞを開くことができる。

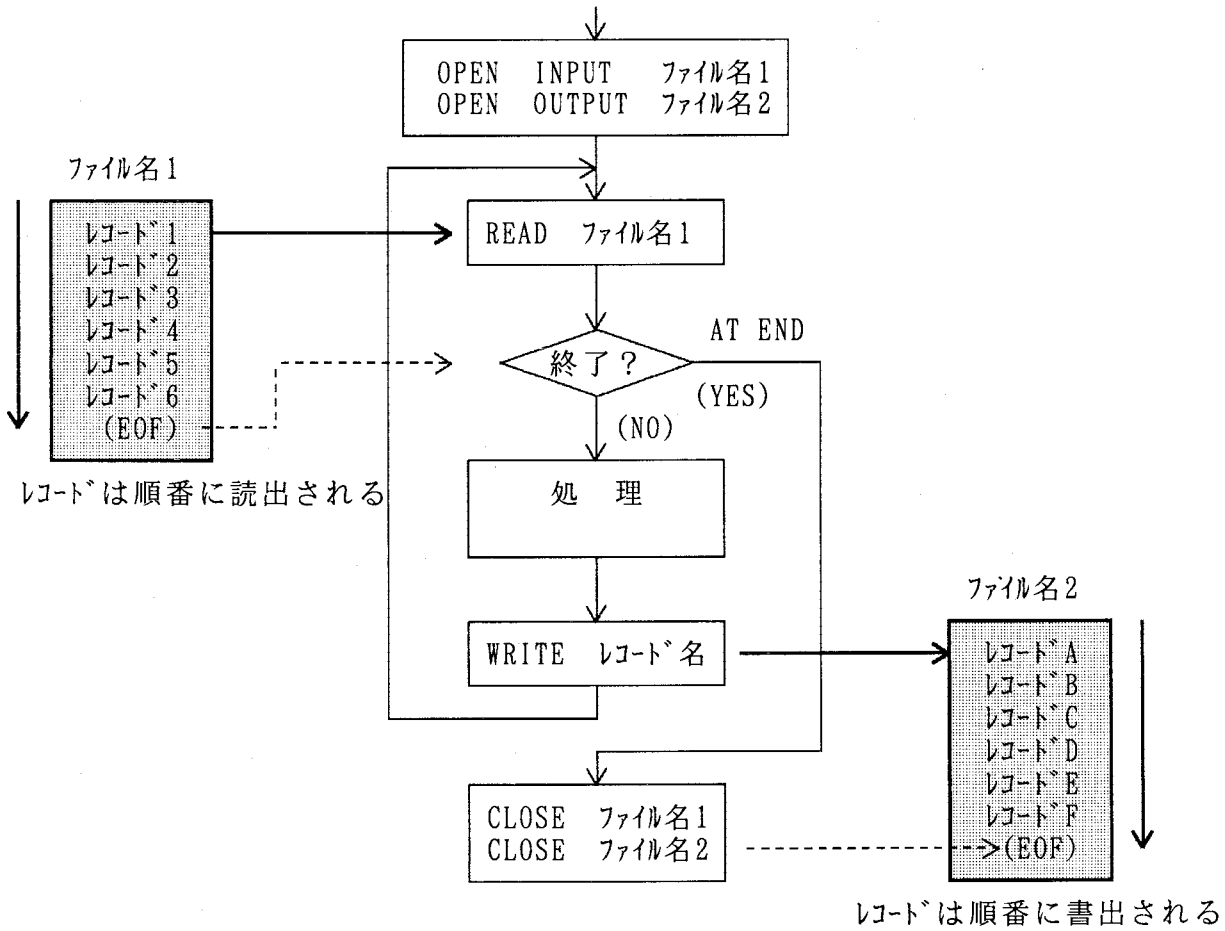
５．８．１　順編成

順編成ファイルは、レコードの前後関係はファイルを生成する時のＷＲＩＴＥ文の実行順序によって決まり、ファイルの終わりにレコードを追加する以外に変わることはない。

順編成ファイルが大記憶ファイルの場合は、更新可能であるが、順次記憶媒体（磁気テープ、プリンタ等）では制約がある。

```
SELECT [OPTIONAL] ﾌｧｲﾙ名  ASSIGN TO 作成者語1
       [RESERVE 整数1 AREAS]
       [[ORGANIZATION IS] SEQUENTIAL]
       [ACCESS MODE IS SEQUENTIAL]
       [FILE STATUS IS ﾃﾞｰﾀ名2].
```

（１）　順編成ﾌｧｲﾙの基本的な処理

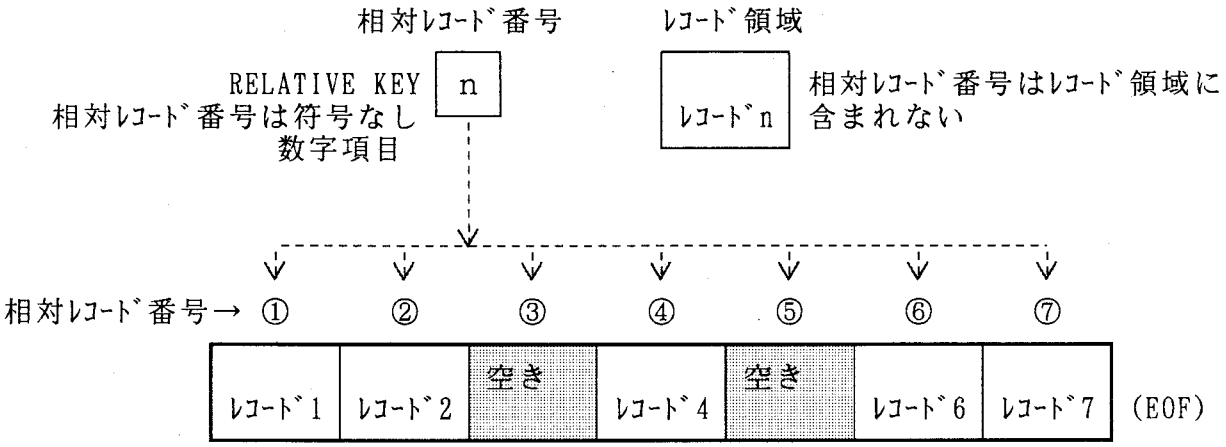


５．８．２　相対編成

相対編成ファイルは、大記憶装置にだけ作成できてレコードは相対レコード番号で識別される。
相対編成ファイル中の各レコード領域は、相対レコード番号をもち、レコードが書かれているか否かにかかわらずレコード領域は確保される。

```
SELECT   [OPTIONAL]   ﾌｧｲﾙ名   ASSIGN TO 作成者語1
         [RESERVE 整数1 AREAS]
         [ORGANIZATION IS RELATIVE]
         [ACCESS MODE IS  ( SEQUENTIAL  [RELATIVE KEY IS ﾃﾞｰﾀ名1] )  ]
                          ( RANDOM       RELATIVE KEY IS ﾃﾞｰﾀ名1   )
                          ( DYNAMIC      RELATIVE KEY IS ﾃﾞｰﾀ名1   )
         [FILE STATUS IS ﾃﾞｰﾀ名2] .
```

（１）　相対ﾌｧｲﾙの入出力構成



（２）　相対ファイルのOPENﾓｰﾄﾞと入出力命令の組合せ

呼出し法	命　令	ＯＰＥＮ　モード			
		ＩＮＰＵＴ	ＯＵＴＰＵＴ	Ｉ－Ｏ	ＥＸＴＥＮＤ
順呼出し法	READ	○	×	○	×
	WRITE	×	○	×	○
	REWRITE	×	×	○	×
	DELETE	×	×	○	×
	START	○	×	○	×
乱呼出し法	READ	○	×	○	×
	WRITE	×	○	○	×
	REWRITE	×	×	○	×
	DELETE	×	×	○	×
	START	×	×	×	×

５．８．３　索引編成

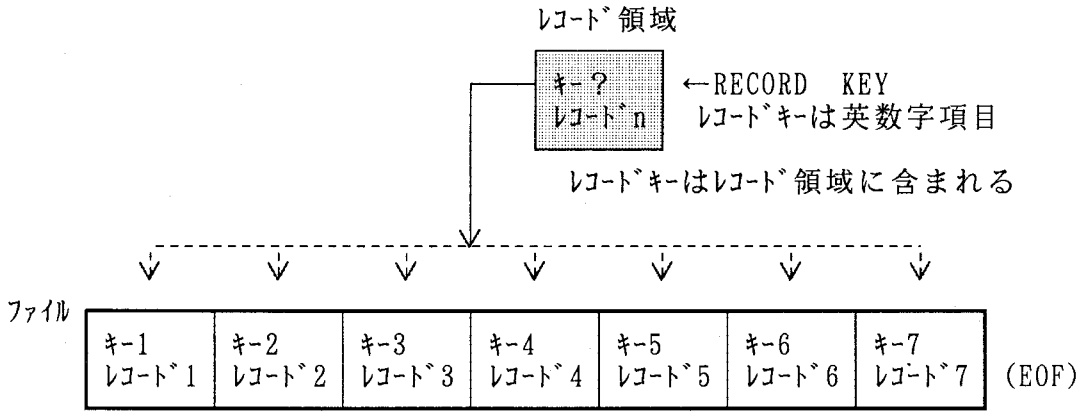
索引編成ファイルは、レコード中の指定されたキーの値によって対応するレコードを呼び出すことができる大記憶ファイルである。ファイル中のレコードのキーデータ項目で索引が作成される。

各レコードは、キーを表す主レコードキー（record　key）をもち、主レコードキーの値により、挿入、変更、削除できる。

副レコードキー（alternate　record　key）は、ファイルのレコードを検索するための別の手段のためのキーである。各レコードの主レコードキーの重複は許されないが、副レコードキーは重複していてもよい。

```
SELECT   [OPTIONAL]   ﾌｧｲﾙ名   ASSIGN TO 作成者語1
         [RESERVE 整数1 AREAS]
         [ORGANIZATION IS]   INDEXED
         [ACCESS MODE IS  ( SEQUENTIAL )  ]
                          ( RANDOM     )
                          ( DYNAMIC    )
         RECORD KEY IS ﾃﾞｰﾀ名1
         [ALTERNATE RECORD KEY IS ﾃﾞｰﾀ名2  [WITH DUPLICATES] ]
         [FILE STATUS IS ﾃﾞｰﾀ名3] .
```

（１）　索引ﾌｧｲﾙの入出力構成



（２）　索引ファイルのOPENﾓｰﾄﾞと入出力命令の組合せ

呼出し法	命　令	ＯＰＥＮ　モード			
		ＩＮＰＵＴ	ＯＵＴＰＵＴ	Ｉ－Ｏ	ＥＸＴＥＮＤ
順呼出し法	READ	○	×	○	×
	WRITE	×	○	×	○
	REWRITE	×	×	○	×
	DELETE	×	×	○	×
	START	○	×	○	×
乱呼出し法	READ	○	×	○	×
	WRITE	×	○	○	×
	REWRITE	×	×	○	×
	DELETE	×	×	○	×
	START	×	×	×	×

５．９　呼出し法

ファイル記述項の「ACCESS　MODE」句でファイル中のレコードの操作方法を指定する。

５．９．１　順呼出し法

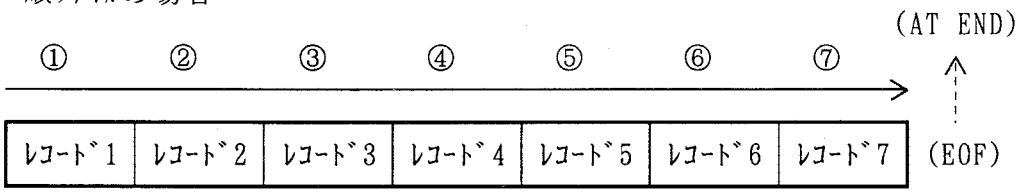
レコードの入出力を、各々のファイル編成で決められた順序で行う。
ファイル編成に関係なく全てのファイルに指定できる。

```
ACCESS MODE IS SEQENTIAL
```

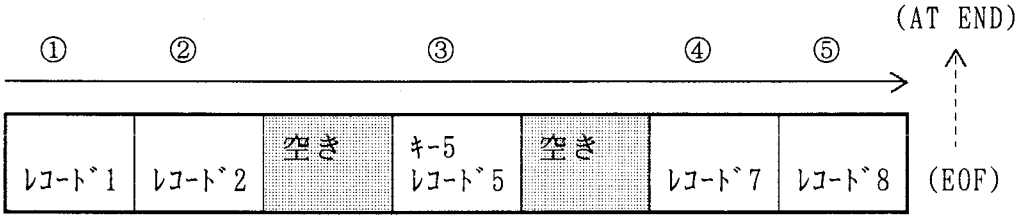
順編成　　－　レコードが作成（拡張）された順序。
索引編成　－　レコードキーの順序（昇順）
相対編成　－　相対レコード番号の順序（昇順）

```
SELECT  ﾌｧｲﾙ名 ASSIGN  TO  作成者語1
              ORGANIZATION  IS  SEQUENTIAL/INDEXED/RELATIVE
              ACCESS  MODE  IS  SEQUENTIAL.


    OPEN  INPUT  ﾌｧｲﾙ名.
LOOP.
    READ  ﾌｧｲﾙ名  AT  END  GOTO  OWARI.

    読み込んだﾚｺｰﾄﾞ（①～⑦）の処理　（相対ﾌｧｲﾙは①～⑤）

    GOTO  LOOP.
OWARI.
    CLOSE  ﾌｧｲﾙ名.
```

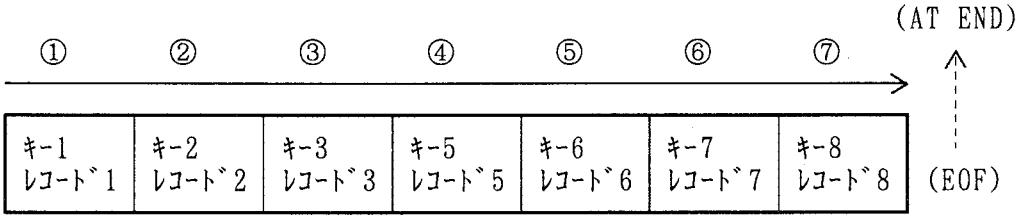
（１）　順ﾌｧｲﾙの場合



（２）　相対ﾌｧｲﾙの場合



（３）　索引ﾌｧｲﾙの場合



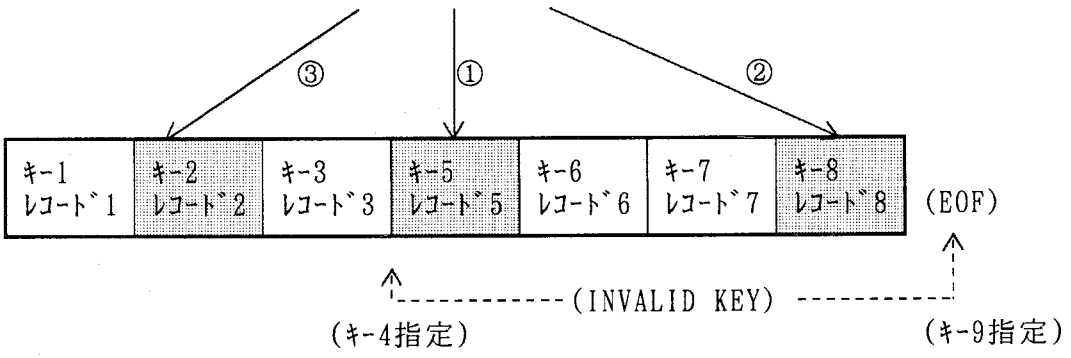
５．９．２　乱呼出し法

乱呼出し法は、索引編成ファイルと相対編成ファイルで使える。
レコードの入出力は、レコードキー（主、副）や相対レコード番号を指定して行う。

```
ACCESS MODE IS RANDOM
```

順編成　　－　使用できない。
索引編成　－　レコードキーに設定した値でアクセスする。
相対編成　－　相対レコード番号（１以上）でアクセスする。

```
SELECT  ﾌｧｲﾙ名 ASSIGN  TO  作成者語1
              ORGANIZATION  IS  INDEXED  RECORD KEY IS  ｷｰﾃﾞｰﾀ
              ACCESS  MODE  IS  RANDOM

01  ﾚｺｰﾄﾞ領域.
    03  ｷｰﾃﾞｰﾀ  PIC  X(5).
    03  ﾃﾞｰﾀ    PIC  X(50).

PROCEDURE  DIVISION.
    OPEN  INPUT  ﾌｧｲﾙ名.

    MOVE  "ｷｰ5"  TO  ｷｰﾃﾞｰﾀ.
    READ  ﾌｧｲﾙ名  INVALID  KEY  DISPLAY  "ｷｰ5がない".       ①
    読み込んだﾚｺｰﾄﾞの処理

    MOVE  "ｷｰ8"  TO  ｷｰﾃﾞｰﾀ.
    READ  ﾌｧｲﾙ名  INVALID  KEY  DISPLAY  "ｷｰ8がない".       ②

    MOVE  "ｷｰ2"  TO  ｷｰﾃﾞｰﾀ.
    READ  ﾌｧｲﾙ名  INVALID  KEY  DISPLAY  "ｷｰ2がない".       ③

    CLOSE  ﾌｧｲﾙ名.
```



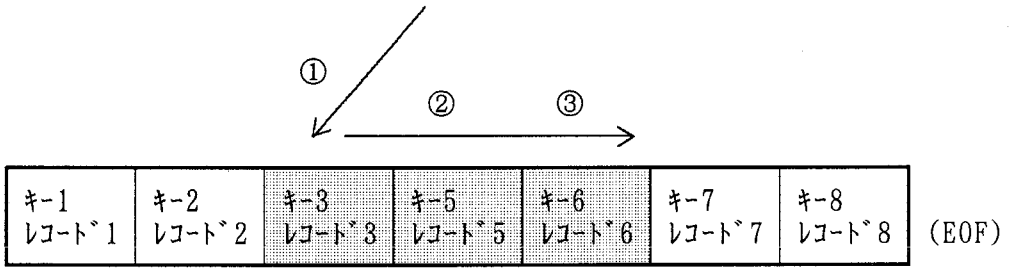
５．９．３　動的呼出し法

動的呼出し法は、索引編成ファイルと相対編成ファイルで使える。
実行時、順呼出し法と乱呼出し法のレコード操作を切り換えて使用できる。

```
ACCESS MODE IS DYNAMIC
```

順編成　　－　使用できない。
索引編成　－　レコード・キーの昇順（SEQUENTIAL）、または、レコード・キーに設定した任意（RANDOM）の値でアクセスする。
相対編成　－　相対レコード番号の昇順（SEQUENTIAL）、または、相対レコード番号に設定した任意（RANDOM）の値でアクセスする。

```
SELECT  ﾌｧｲﾙ名 ASSIGN  TO  作成者語1
              ORGANIZATION  IS  INDEXED  RECORD KEY IS  ｷｰﾃﾞｰﾀ
              ACCESS  MODE  IS  DYNAMIC

01  ﾚｺｰﾄﾞ領域.
    03  ｷｰﾃﾞｰﾀ  PIC  X(5).
    03  ﾃﾞｰﾀ    PIC  X(50).

PROCEDURE  DIVISION.
    OPEN  INPUT  ﾌｧｲﾙ名.

    MOVE  "ｷｰ3"  TO  ｷｰﾃﾞｰﾀ.
    READ  ﾌｧｲﾙ名  INVALID  KEY  DISPLAY  "ｷｰ3がない".
    読み込んだﾚｺｰﾄﾞ①の処理

SEQN.
    READ  ﾌｧｲﾙ名  NEXT  AT  END  DISPLAY  "ﾌｧｲﾙがEOF".
    読み込んだﾚｺｰﾄﾞ（②③）の処理
    IF  ｷｰﾃﾞｰﾀ  <  "ｷｰ7"  GO  TO  SEQN.

    CLOSE  ﾌｧｲﾙ名.
```



５．１０　ファイルステータス

入出力命令文の実行された結果（入出力状態）はファイルステータスとして、２桁の数字を英数字項目のデータ領域に返す。

```
ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.
FILE-CONTROL.

    SELECT [OPTIONAL] ﾌｧｲﾙ名  ASSIGN TO 作成者語1
           FILE STATUS IS ﾃﾞｰﾀ名2.

DATA  DIVISION.

WORKING-STORAGE  SECTION.

01  ﾃﾞｰﾀ名2.                    ←  WORKING-STORAGE  SECTIONに
  03  上位桁    PIC  X.              英数字項目として定義する。
  03  下位桁    PIC  X.
```

ファイルステータスには、基本的には次のコードが割付られている。
これらの値は、入出力命令文の実行後やDECLARATIVESのEXCEPTION／ERROR宣言の中で参照する。

上位桁	意味	下位桁	意味　（例）
0	正常終了	0	情報なし
1	ファイルの終了（AT END）	0	情報なし
2	無効なキー状態（INVALID）	0 1 2 3 4	情報なし 順序が不正 重複キー レコードが存在しない ファイル領域が足りない （相対、索引ﾌｧｲﾙの場合）
3	永続エラーが発生	0 4	情報なし ファイル領域が足りない （順ﾌｧｲﾙの場合）
4	論理エラーが発生	1 2	開かれているﾌｧｲﾙのOPEN 開かれていないﾌｧｲﾙのCLOSE
9	その他のエラー	0	情報なし

ファイルステータスの使用例（ファイルの終了をチェックする）

```
READ  ﾌｧｲﾙ名
    AT END GO TO  OWARI.
```

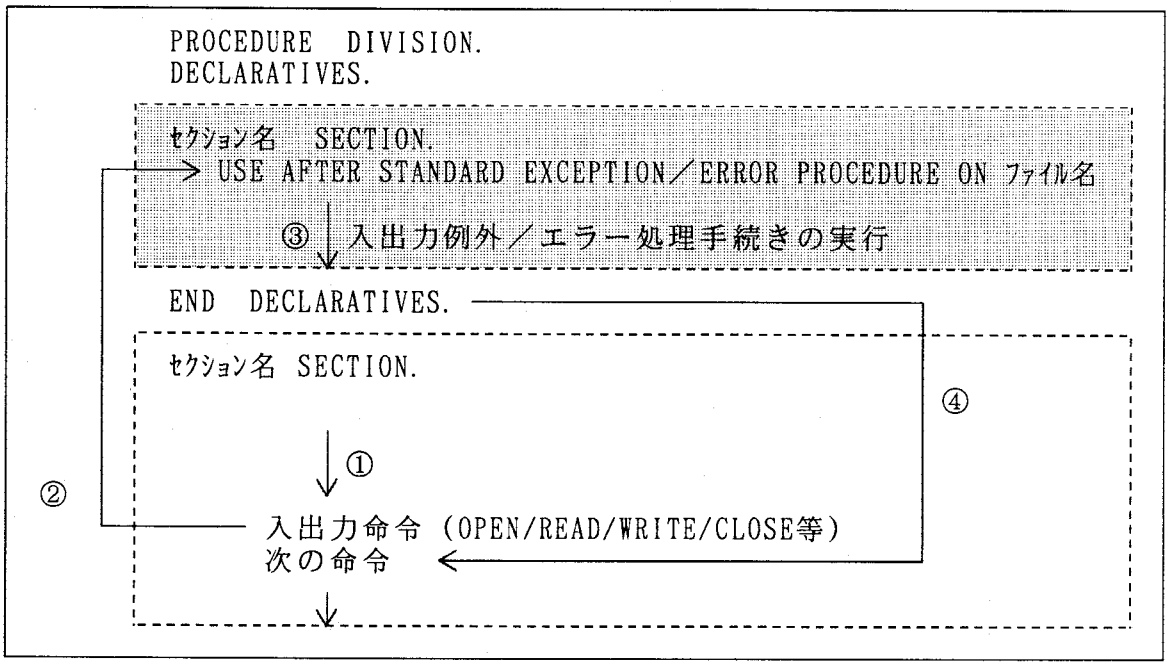
```
READ  ﾌｧｲﾙ名.
IF  ﾌｧｲﾙｽﾃｰﾀｽ = "10"  GO TO  OWARI.
```

５．１１　入出力例外／エラー宣言　　」

ＣＯＢＯＬ入出力命令文の実行で、入出力例外やエラーが発生するとＵＳＥ句で宣言した手続きを実行する。その後、実行は入出力命令文の次の命令文に移る。

```
PROCEDURE  DIVISION.
DECLARATIVES.
ｾｸｼｮﾝ名  SECTION.
    USE AFTER STANDARD EXCEPTION／ERROR PROCEDURE ON ﾌｧｲﾙ名/ﾓｰﾄﾞ指定
      ファイルの入出力例外／エラー処理手続きを記述する。
END  DECLARATIVES.

ｾｸｼｮﾝ名 SECTION.

      入出力命令（OPEN/READ/WRITE/CLOSE等）
```

（１）　実行順序について
入出力例外／エラーが発生した場合の実行順序は次の通り。



次のような場合にも、入出力例外／エラー宣言が実行される。
・AT ENDが指定されない時に、AT END状態が発生した場合
　　READ　ﾌｧｲﾙ名.
・INVALIDKEYが指定されない時に、INVALID KEY状態が発生した場合
　　WRITE　ﾚｺｰﾄﾞ名.

入出力例外／エラー宣言の中から、通常の手続きを参照（実行）してはならない。

```
DECLARATIVES.
  手続き1 ──────────┐   （×  PERFORM  手続き2）
END  DECLARATIVES.    │
                      │
  手続き2 ←──────────┘
```

５．１２　順ファイル

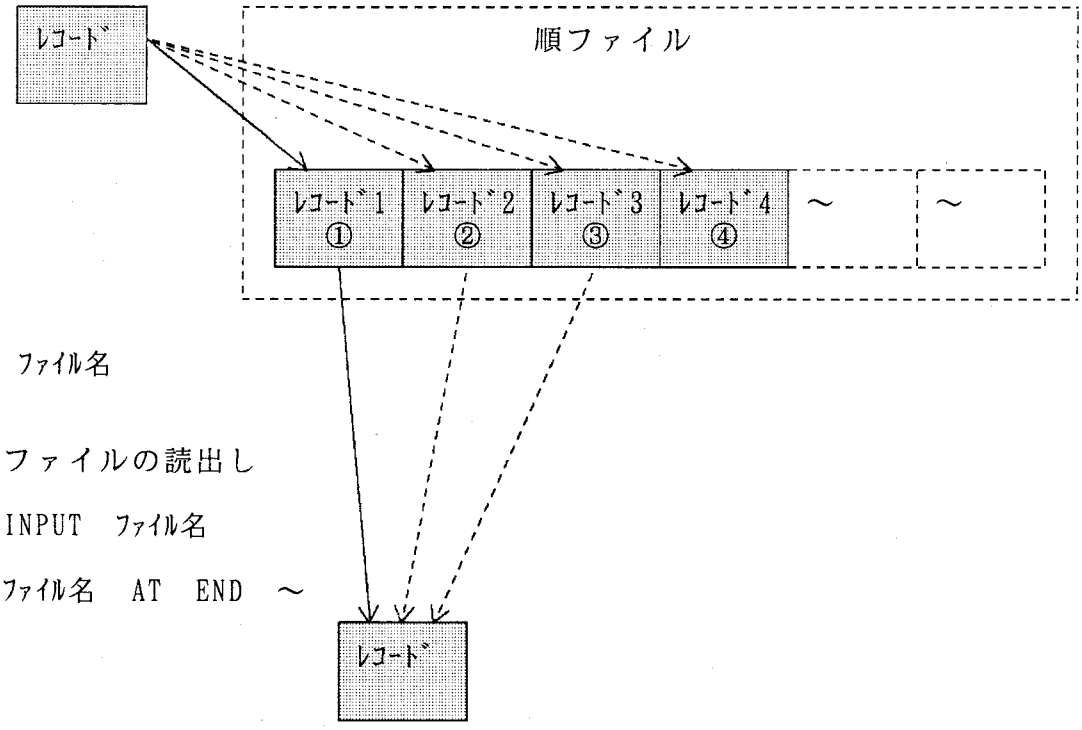
５．１２．１　順ファイルの概念

順ファイルは、ファイルの先頭から順にレコードが記録されているファイルである。
レコードのアクセスはファイルの先頭から順に行う。

（１）　順ファイルの作成

OPEN　OUTPUT　ﾌｧｲﾙ名

WRITE　ﾚｺｰﾄﾞ



CLOSE　ﾌｧｲﾙ名

（２）　順ファイルの読出し

OPEN　INPUT　ﾌｧｲﾙ名

READ　ﾌｧｲﾙ名　AT　END　～

CLOSE　ﾌｧｲﾙ名

順ファイルと入出力命令文の組合せは、次のとおり。

呼出し法	入出力文	ＯＰＥＮ　モード			
		ＩＮＰＵＴ	ＯＵＴＰＵＴ	Ｉ－Ｏ	ＥＸＴＥＮＤ
順呼出し法	READ	○	×	○	×
	WRITE	×	○	×	○
	REWRITE	×	×	○	×

５．１２．２　順ファイルの定義

順ファイルの定義概要は、次のとおり。

```
IDENTIFICATION  DIVISION.

ENVIRONMENT  DIVISION.

INPUT-OUTPUT  SECTION.
FILE-CONTROL.
    SELECT  [OPTIONAL]  ﾌｧｲﾙ名  ASSIGN  TO  作成者語
            ORGANIZATION  IS  SEQUENTIAL
            ACCESS  MODE  IS  SEQUENTIAL
            FILE  STATUS  IS  ｽﾃｰﾀｽ.
     :
I-O-CONTROL.
    RERUN  ON  ･･･
    SAME  RECORD  AREA  FOR  ﾌｧｲﾙ名  ･･
     :

DATA  DIVISION.

FILE  SECTION.
FD  ﾌｧｲﾙ名
      BLOCK  CONTAINS  整数2  RECORDS
      LINAGE  整数5  FOOTING  整数6  TOP  整数7  BOTTOM  整数8
01  ﾚｺｰﾄﾞ記述項.


WORKING-STORAGE  SECTION.
01  ｽﾃｰﾀｽ       PIC  XX.


PROCEDURE  DIVISION.

DECLARATIVES.
  USE AFTER STANDARD EXCEPTION/ERROR PROCEDURE ON  ﾌｧｲﾙ名/INPUT/･･

END  DECLARATIVES.

    OPEN  入出力ﾓｰﾄﾞ  ﾌｧｲﾙ名

    READ  ﾌｧｲﾙ名  AT END ･･
    WRITE  ﾚｺｰﾄﾞ名  AFTER/BEFORE  ADVANCING  ･･
    REWRITE  ﾚｺｰﾄﾞ名


    CLOSE  ﾌｧｲﾙ名
```

5．12．3　SELECT

順ファイルに関係する物理属性を定義する。

```
SELECT [OPTIONAL] ﾌｧｲﾙ名 ASSIGN TO 作成者語1
       [RESERVE 整数1 AREAS]
       [ [ORGANIZATION IS] SEQUENTIAL]
       [ACCESS MODE IS SEQUENTIAL]
       [FILE STATUS IS ﾃﾞｰﾀ名2] .
```

通常の定義は次の通り。

```
SELECT ﾌｧｲﾙ名 ASSIGN TO 作成者語1.
```

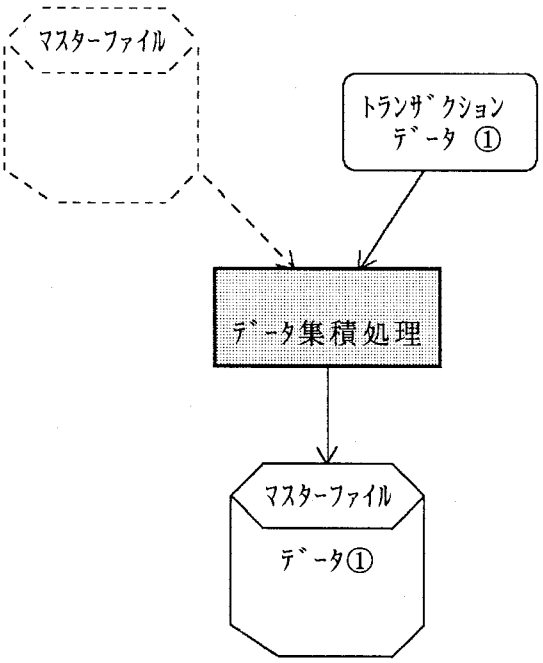
順ファイルのファイルステータス

上位桁	下位桁	意味	
0	0	成功	情報なし
0	4		ﾚｺｰﾄﾞ長がﾌｧｲﾙ属性と合わない
0	5		OPTIONAL指定でﾌｧｲﾙが存在しない
1	0	ﾌｧｲﾙの終了	情報なし
3	0	永続ｴﾗｰ	情報なし
3	4		区域外書き出しによる永続ｴﾗｰ
3	5		OPEN INPUT/I-Oでﾌｧｲﾙが存在しない
3	7		OPENﾓｰﾄﾞと物理ﾌｧｲﾙの属性が一致しない
3	8		CLOSE WITH LOCKのﾌｧｲﾙを再OPENした
3	9		物理ﾌｧｲﾙ属性とﾌｧｲﾙ定義との間で矛盾を検出
4	1	論理ｴﾗｰ	OPEN済みﾌｧｲﾙに再度OPENを実行した
4	2		OPENされていないﾌｧｲﾙにCLOSEを実行した
4	3		REWRITEの前がREADでない
4	4		REWRITEのﾚｺｰﾄﾞが同じ大きさでない
4	6		終了したﾌｧｲﾙにREADを実行した
4	7		OUTPUTﾓｰﾄﾞでREADを実行した
4	8		INPUT/I-OでWRITEを実行した
4	9		I-Oﾓｰﾄﾞ以外でREWRITEを実行した
9	x	その他ｴﾗｰ	作成者が定義する

（１）ＯＰＴＩＯＮＡＬ指定について

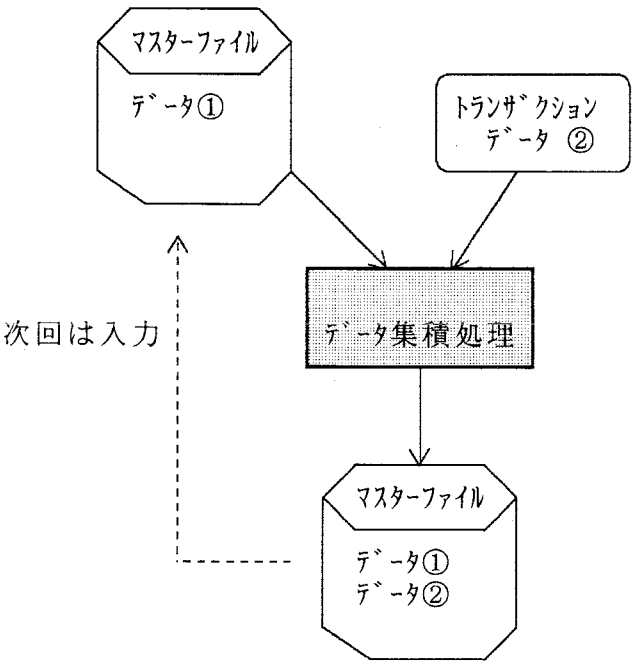
実行時、入力ファイルが状況により存在しなくても実行時エラーにならないように定義する。入力ファイルが存在しなくても「OPEN　INPUT」は成功する。そして、最初の「READ」で入力ファイルの終了（AT　END条件）になる。
次のようなデータ集積処理で使用すれば、実行の都度、入力ファイルの有無をパラメータで指示しなくても済む。

①　最初は、トランザクションのデータのみでマスターファイルは存在しない。




```
SELECT  OPTIONAL  入力ﾏｽﾀｰ
SELECT  ﾄﾗﾝｻﾞｸｼｮﾝ
SELECT  出力ﾏｽﾀｰ

  OPEN  INPUT   入力ﾏｽﾀｰ
        INPUT   ﾄﾗﾝｻﾞｸｼｮﾝ
        OUTPUT  出力ﾏｽﾀｰ
S1.
  READ  入力ﾏｽﾀｰ  AT END  GO S2.
  WRITE  出力ﾏｽﾀｰ･ﾚｺｰﾄﾞ
  GO  TO  S1.
S2.
  READ  ﾄﾗﾝｻﾞｸｼｮﾝ  AT END  GO EE.
  WRITE  出力ﾏｽﾀｰ･ﾚｺｰﾄﾞ
  GO  TO  S2.

EE.
  CLOSE  入力ﾏｽﾀｰ  出力ﾏｽﾀｰ
         ﾄﾗﾝｻﾞｸｼｮﾝ.
```

②　２回目以降は、入力マスターファイルにデータは存在する。データを出力マスターファイルに複写して、その後にトランザクションのデータを追加する。



５．１２．４　ファイル記述項

ファイル節（FILE　SECTION）はデータ部にあり、ファイルの構造を定義する。
ファイル節の一般形式は、次のとおり。

```
FILE　SECTION.

［ファイル記述項
  {レコード記述項}・・・　］・・・
```

ファイル記述項は、ファイルの物理的構造（レコードやブロックのサイズ、ラベルレコードの有無）やファイルのレコード名情報を定義する。

```
FILE　SECTION.

FD　ﾌｧｲﾙ名1
    [BLOCK　CONTAINS　[整数1　TO]　整数2　RECORDS／CHARACTERS]
    [RECORD　CONTAINS　整数3　CHARACTERS]
    [LABEL　RECORD　IS　STANDARD／OMITTED]
    [DATA　RECORD　IS　{ﾃﾞｰﾀ名3}・・・]
    [LINAGE　IS　整数8　LINES　[WITH　FOOTING　AT　整数9]
    [LINES　AT　TOP　整数10]　[LINES　AT　BOTTOM　整数11]]
    [CODE-SET　IS　符号系名1].

01　ﾃﾞｰﾀ名3　{レコード記述項}
```

注：整数8～整数10にはデータ名も指定できる。

ファイル記述は最初にレベル指示語「ＦＤ」を書き、その後にファイル名を書く。
ファイル名の後に書く句は任意である。
ファイル記述項の後に、一つ以上のレコード記述項を書く。

レコード記述は、レコードの形式を定義する。
レコードは、レベル番号とデータ名（FILLER）の後に必要な句を並べてデータ項目を定義する。必要ならばレベル番号の概念でデータ項目の階層構造を定義する。

ファイル名に複数のレコード記述項を定義しても、それらのレコード領域は全て同じ領域である。

```
FD　ﾌｧｲﾙ名
    DATA　RECORDS　ARE　ﾚｺｰﾄﾞ名1,　ﾚｺｰﾄﾞ名2,　ﾚｺｰﾄﾞ名3.
01　ﾚｺｰﾄﾞ名1.
    02　ﾃﾞｰﾀ名11　～
01　ﾚｺｰﾄﾞ名2.
    02　ﾃﾞｰﾀ名21　～
01　ﾚｺｰﾄﾞ名3.
    02　ﾃﾞｰﾀ名31　～
```

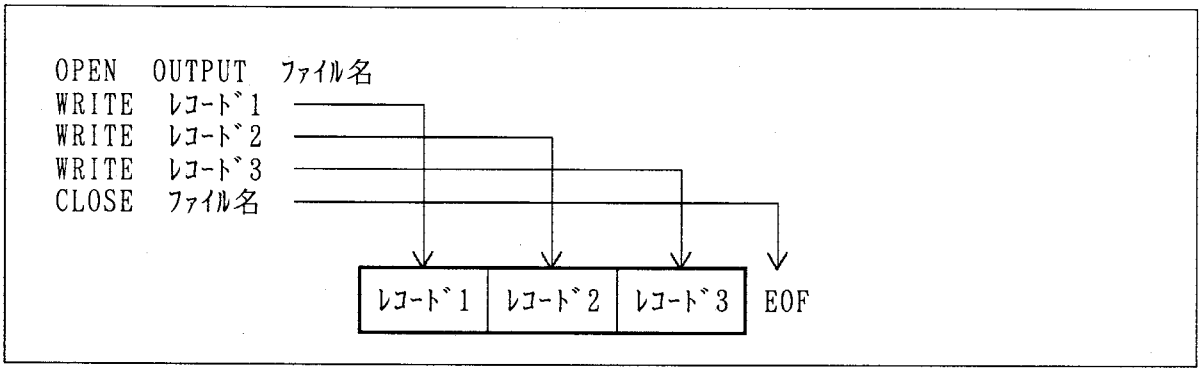
５．１２．５　ＯＰＥＮ

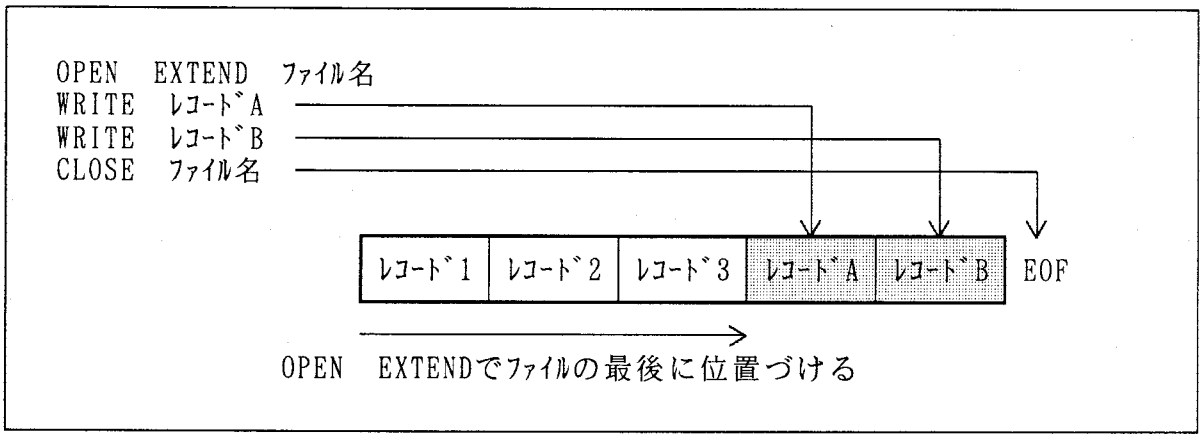
ＯＰＥＮ（開く）文は、ファイルの処理を行うための準備をする。
具体的には、ファイルが物理的に存在しデータ管理システムに認識されて使用可能になる。

```
                ┌ INPUT    {ﾌｧｲﾙ名1   [REVERSED／NO  REWIND] } ･･･ ┐
                │ OUTPUT   {ﾌｧｲﾙ名2   [NO  REWIND] } ･･･            │
OPEN            │                                                   ├ ･･･
                │ I-O      {ﾌｧｲﾙ名3} ･･･                           │
                └ EXTEND   {ﾌｧｲﾙ名4} ･･･                           ┘
```

（１）　ファイルの拡張

新規ファイルの作成は、「OPEN　OUTPUT」でオープンしてWRITE文でレコードを書き出す。ファイルの追加（拡張）は、「OPEN　EXTEND」でオープンしてWRITE文でレコードを書き出す。





（２）　順ファイルの更新

順ファイルの更新は、「OPEN　I-O」で更新ファイルをオープンして、READ文でレコードを読み出す。そのレコードに対してREWRITE文でレコードを書き換える。
更新ファイル（I-O）は、WRITE文でレコードを追加することはできない。
ファイルの記憶媒体（磁気テープ等）、及び、固定長レコード、可変長レコードでは制約がある。

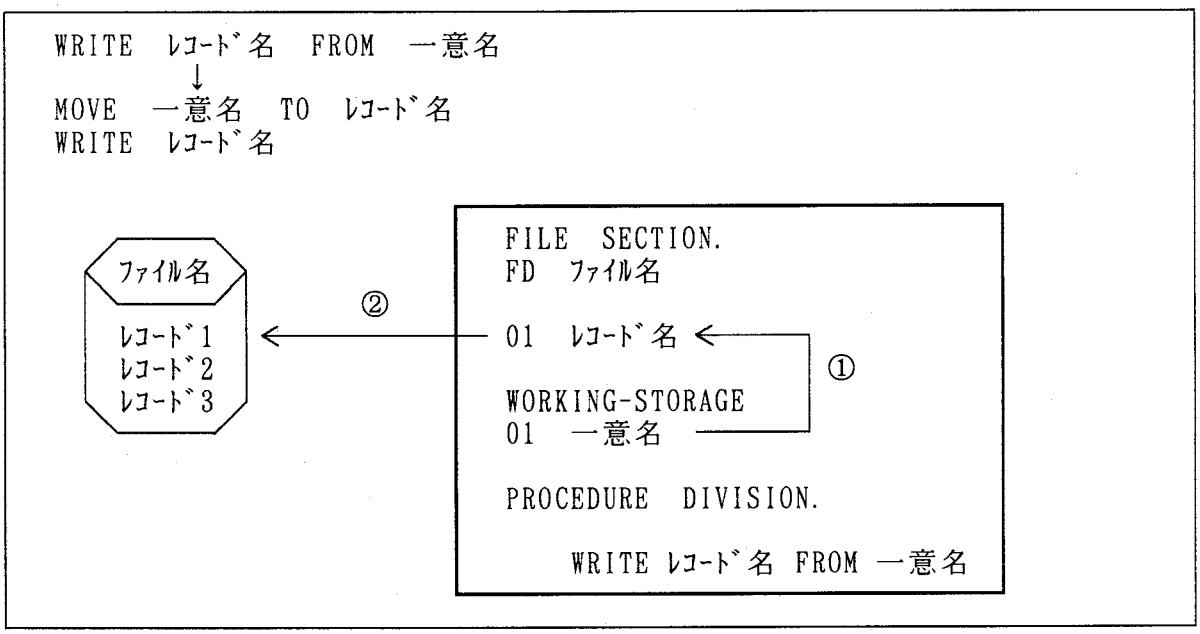
５．１２．６　ＷＲＩＴＥ

ＷＲＩＴＥ（書出し）文は、レコードを出力ファイルに書き出す。
印刷ファイルでは、論理ページの縦方向の行送りもできる。

```
WRITE  ﾚｺｰﾄﾞ名1  [FROM  一意名1]
       [AFTER／BEFORE  ADVANCING  PAGE／整数1／一意名2  LINE／LINES]
       [AT  END-OF-PAGE  無条件文1]
       [NOT  AT  END-OF-PAGE  無条件文2]
 [END-WRITE]
```

（１）　ＦＲＯＭの動作

MOVE命令の規則に従ってFROMのデータをレコード領域に移してから、レコードが書き出される。
WRITE命令実行後のレコード領域の内容は保証されないので、FROMを使用する。



（２）　ＡＤＶＡＮＣＩＮＧ指定

印字ページの縦方向の行制御を行う。
ADVANCING句を指定しないと「AFTER　ADVANCING　1　LINE」が想定され自動的な行送りが行われる。
「AFTER／BEFORE　ADVANCING　PAGE」は、次の論理ページに位置づける前（BEFORE）または、位置づけた後（AFTER）にレコードを論理ページに書き出す。

（３）　ＥＮＤ－ＯＦ－ＰＡＧＥ指定

ＷＲＩＴＥ文の実行中に、印字ページが論理的な終わりになるとEND-OF-PAGEの無条件文1が実行される。論理的なページの定義は、ＦＤのＬＩＮＡＧＥ句で指定する。

５．１２．７　ＲＥＡＤ

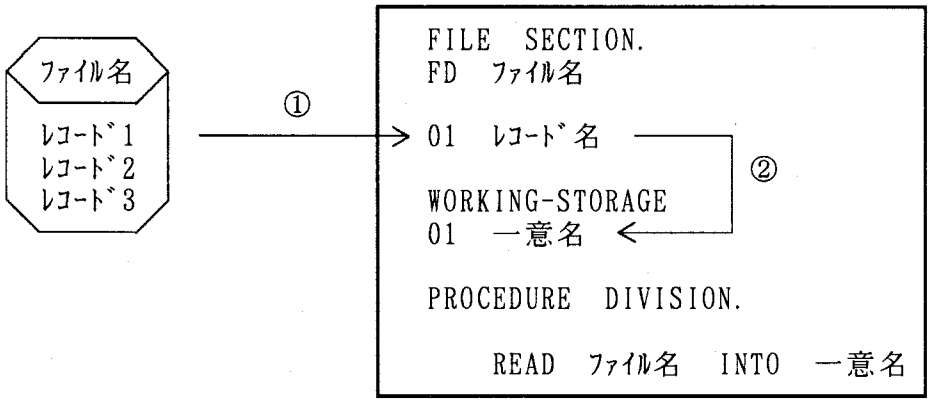
ＲＥＡＤ（読込み）文は、ファイルの次のレコードを使用可能にする。

```
READ  ﾌｧｲﾙ名1  [NEXT]  RECORD  [INTO  一意名1]
      [AT  END  無条件文1]
      [NOT  AT  END  無条件文2]
 [END-READ]
```

（１）　ＩＮＴＯの動作

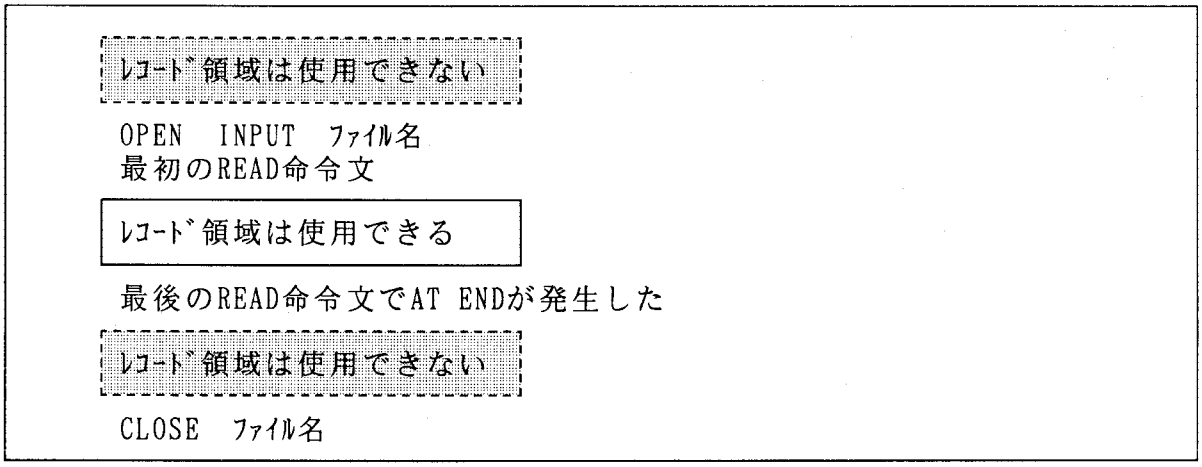
レコード領域に読み込まれてから、INTOで指定した領域にMOVE命令も規則に従って移される。

```
READ  ﾌｧｲﾙ名  INTO  一意名
        ↓
READ  ﾌｧｲﾙ名
MOVE  ﾚｺｰﾄﾞ名  TO  一意名
```

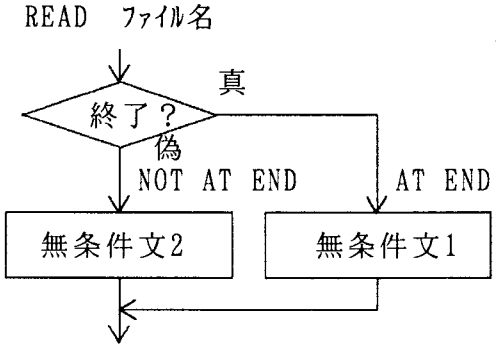


（２）　レコード領域について

OPEN前のレコード領域と、AT　END後のレコード領域は使用できない。



（３）「ＡＴ　ＥＮＤ」と「ＮＯＴ　ＡＴ　ＥＮＤ」について



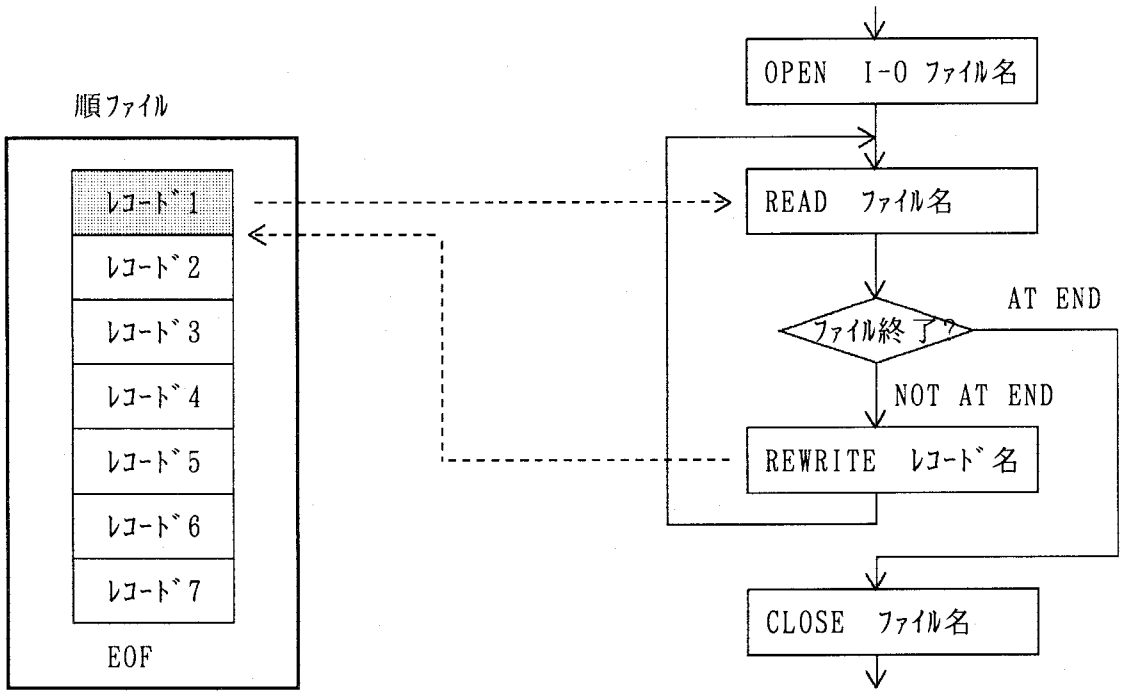
５．１２．８　ＲＥＷＲＩＴＥ

ＲＥＷＲＩＴＥ（書換え）文は、大記憶ファイル中にあるレコードを書き換える。
ＯＰＥＮは入出力両用モード（Ｉ－Ｏ）で開いておくこと。

```
REWRITE  ﾚｺｰﾄﾞ名1  [FROM  一意名1]
 [END-REWRITE]
```

（１）　順ファイルでのREWRITEの使い方

ファイルをI-0でOPENして、READに続いてREWRITEを行うことで順ファイルの更新ができる。



５．１２．９　ＣＬＯＳＥ

ＣＬＯＳＥ（閉じる）文は、ファイルの処理を終了させる。
必要ならば、巻戻し（rewind）、施錠（lock）、取外し（removal）ができる。

```
CLOSE { ﾌｧｲﾙ名1  [REEL／UNIT FOR REMOVAL]
                 [WITH LOCK／WITH NO REWIND] } ...
```

ファイルの施錠とは、その実行単位で再びＯＰＥＮできないようにする。

```
OPEN  I-O  ﾌｧｲﾙ名
  ファイルの更新処理
CLOSE  ﾌｧｲﾙ名  WITH LOCK
OPEN  INPUT  ﾌｧｲﾙ名          ←  OPENは成功しない。
```

指導上の留意点

◎ＣＯＢＯＬ入出力
ＣＯＢＯＬプログラムに於ける入出力定義の全体像を理解、整理させる。
「Ｃ」、「ＦＯＲＴＲＡＮ」と対比させる。

◎ＦＩＬＥ－ＣＯＮＴＲＯＬ
ここでの問題は、ファイル編成と呼出し法が混乱してしまう場合が見受けられることである。

```
ORGANIZATION  IS  SEQUENTIAL        ORGANIZATION  IS  INDEXED
ACCESS  MODE  IS  SEQUENTIAL        ACCESS  MODE  IS  SEQUENTIAL
```

ファイル編成については、順次編成、及び索引編成を中心に指導する。
ＣＯＢＯＬの事務処理では、相対編成はあまり使用されていない。

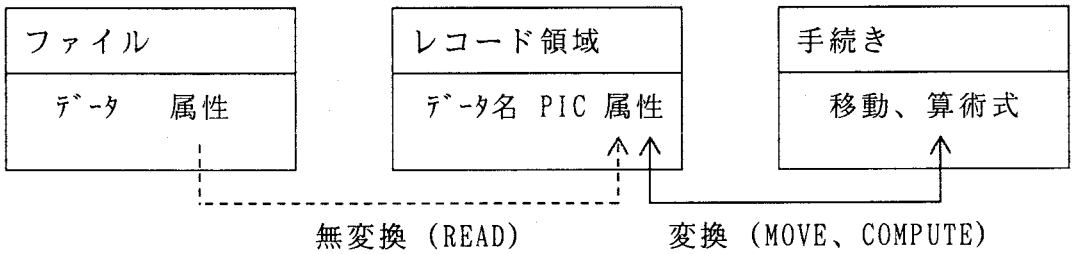
ファイルステータスについて基本的なコードと意味を理解させる。また、これらのコードは処理系により追加、拡張されている。

◎Ｉ－Ｏ－ＣＯＮＴＲＯＬ
通常のプログラムではほとんど使用されない。
この宣言を使用するのは、一度に多数のファイルを使用するとレコードやバッファ領域が大量に取られて主記憶装置の容量が不足することがある。これを解決する方法としてレコード領域を共用する場合などに使用する。

◎ＦＩＬＥ　ＳＥＣＴＩＯＮ
ここでの重要は点は、「BLOCK　CONTAINS　整数　RECORDS」である。
汎用コンピュータでは「整数」を「０」で記述させて、実行時に「ＪＣＬ」で指定させプログラムに自由度を持たせている。
「LABEL　RECORD　IS　STANDARD/OMITTED」は、ほとんどの処理系で注釈としている。

◎レコード記述項
レコードの構造を定義するが、重要な点は各々のデータ名に指定するPICTURE句の属性と入出力の関係である。ＣＯＢＯＬでは入力時には属性に対応してデータ変換が行われないと言うことである。
現れる現象としては、「計算結果が合わない」、「計算の途中でプログラムが異常終了してしまう」などが発生する。

・入出力と属性変換について

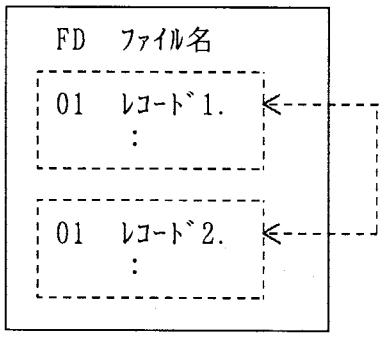


ACCCEPT入力

```
                        01  ﾃﾞｰﾀ名  PIC  S9(4).
+12A4  ---------->  ACCEPT  ﾃﾞｰﾀ名
無変換              COMPUTE  ﾃﾞｰﾀ名  =  ﾃﾞｰﾀ名  +  1.
```

ACCEPTは成功するがCOMPUTEでの結果は予測できない。

相対編成の相対レコード番号、索引編成のレコードキーの定義をどこに指定するのかを整理させる。
レコード記述項のレベル番号「01」には、「REDEFINES」の機能がある。



ﾚｺｰﾄﾞ1とﾚｺｰﾄﾞ2は同一の領域に確保される。
サイズが異なる場合の意味、使用法について解説する。

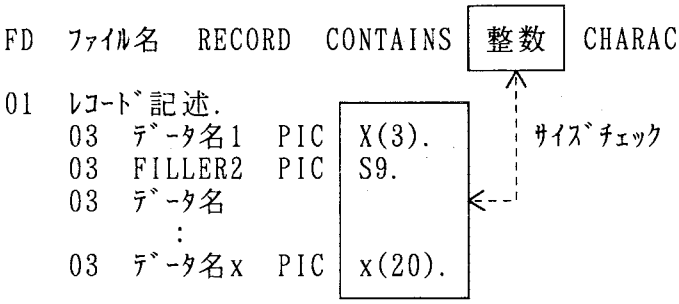
◎ＤＥＣＬＡＲＡＴＩＶＥＳ
通常、ＣＯＢＯＬの実行時入出力エラーが発生すると、実行はＣＯＢＯＬ入出力ライブラリで処理を打ち切られてしまうが、ここでEXCEPTION/ERRORを宣言すると、処理が打ち切られることなく、プログラム中で入出力エラー処理を行うことができる。
一般的な事務処理プログラムではほとんど使用しない機能である。

◎入出力命令文
ファイルの処理として、「追加」、「変更」、「挿入」、「削除」などの意味の理解と、ファイルに対する物理的なこれらの動作を理解させることで、ＣＯＢＯＬの入出力命令を学習する。
ＣＯＢＯＬではファイル編成と処理モードにより入出力命令が規定されることを学習する。

	順次編成	相対編成	索引編成
変更	可能 I-OﾓｰﾄﾞでREAD直後にREWRITEを行う。 ただし、磁気ﾃｰﾌﾟ等ははできない。	可能	可能
挿入	できない	できない	可能
追加	可能 EXTENDﾓｰﾄﾞでOPENしてWRITEで行う。	可能 ただし、空きﾚｺｰﾄﾞの場合	可能
削除	できない	可能 空きﾚｺｰﾄﾞになる	可能

◎ファイルの定義
ファイル編成とＣＯＢＯＬ定義の関係を重点に解説する。
プログラム実行時に必要となるＣＯＢＯＬのファイル名とオペレーティング・システム上の物理ファイル名の関連付けを理解させることも重要である。
ただし、これら関連付けは処理系により異なる。

◎レコードとブロックの定義
「ファイル」、「ブロック」、「レコード」の基本的知識が必要である。
ＦＤの定義では、「ＬＩＮＡＧＥ」を除いて「ＬＡＢＥＬ」、「ＢＬＯＣＫ」などが意味を持たない処理系が多い。
「RECORD　CONTAINS」は、次に記述するレコード記述項との整合性をチェックしてくれるので利用すると便利である。



「BLOCK　CONTAINS」は、ソースプログラムではゼロを指定して置き、実行時にＪＣＬで自由に変更できるようにしている処理系もある。

ＬＩＮＡＧＥ句は、印刷処理で改ページのコントロールを行える点に注目する。

処理系にもよるがレコード記述項（領域）の使用できるタイミングについても誤って理解しやすい点である。

OPEN前　－　ﾚｺｰﾄﾞ領域は使用できない。
WRITE後　－　書出した後の内容は保証されない。
READ後　－　AT END後のﾚｺｰﾄﾞ領域は保証させない。
CLOSE後　－　ﾚｺｰﾄﾞ領域は使用できない。

索引ファイルのレコードキーについては、レコード領域の先頭部分をキーにする場合が多い。レコードキーの位置はレコード内であればどこに存在してもよい。そして、キーとなるデータが複数項目で構成される場合は、それらを連続（集団項目）させることに注意する。

```
01　ﾚｺｰﾄﾞ.
    03　ﾃﾞｰﾀ1
    03　ﾚｺｰﾄﾞｷｰ.
        05　ｷｰ1
        05　ｷｰ2
        05　ｷｰ3
    03　ﾃﾞｰﾀ2
```

汎用コンピュータでは索引ファイルをＶＳＡＭ編成のＫＳＤＳで実現する場合があり、ＶＳＡＭのＫＳＤＳ定義とＣＯＢＯＬプログラムでの定義を一致させないとOPEN命令文で異常終了してしまう。

キーの属性については、次のことも理解しておく必要がある。
索引編成ファイル　－　「＋数字」、と「数字」は別キーになる場合がある。
相対編成ファイル　－　「＋数字」、と「数字」は同じキーである。

◎入出力命令文
各々の命令文の意味と動作を重点に指導する。
ファイル編成と処理モードの組み合わせにより入出力命令文が制限されることを解説する。ACCEPT、DISPLAYについても入出力機能として解説する。