第８章　ファイルの整列と併合

指導目標

本章では、ファイルの整列併合機能について指導する。

整列併合とは、利用者が指定したキーに従って、いくつかのファイルのレコードの順序をそろえたり、そろえられた複数のファイルを併合する処理をいう。
これらの処理は、ＣＯＢＯＬのＳＯＲＴ文やＭＥＲＧＥ文で簡単に記述することができる。
少量のデータ（レコード）は、第５章の内部整列法の応用で処理できるが、大量のデータでは主記憶容量の制限等があり難しい。
事務処理をコンピュータ上で実現するためには、処理するデータを特定の順序に並べて置くことが処理の基本となる。
これらデータの順序づけの知識をＣＯＢＯＬのＳＯＲＴ文とＭＥＲＧＥ文で学習する。

本章で学習する命令文は、次のとおり。

・ＳＯＲＴ文
・ＭＥＲＧＥ文
・ＲＥＬＥＡＳＥ文
・ＲＥＴＵＲＮ文

内容のあらまし

内　　容	説　明	議　論	机上実習	計算機実習
整列機能	具体的なﾃﾞｰﾀで整列を説明し、ﾃﾞｰﾀ形式やｺｰﾄﾞ体系での注意点を明確にする。	数字を文字とした場合の比較について議論する	ﾃﾞｰﾀ形式別の並び順を具体的な値を使用して実習させる。	ﾃﾞｰﾀ量やﾚｺｰﾄﾞの並び順で整列処理時間がどのように変化するか実習する。 汎用ﾕｰﾃｨﾘﾃｨで整列処理ができるように実習する。
ＳＯＲＴ	COBOLの全体像を説明する。 SORT、MERGEでの入力手続きや出力手続きの使い方を説明する			
整列併合用ﾌｧｲﾙの定義	ＦＤとＳＤの使い方が明確になるよう説明する			
ＳＯＲＴ文	SORTのｷｰ指定と入力、出力手続き指定を中心に制御の流れについて説明する。	SORT最終ﾌｪｰｽﾞ処理がどの段階で行われるか議論する。		内部整列法とSORTの処理時間を比較してみる。
ＲＥＬＥＡＳＥ	SORTにﾚｺｰﾄﾞを渡す事によるそのﾚｺｰﾄﾞの移動について説明する。	WRITE文との違いについて議論する。		
ＲＥＴＵＲＮ	整列、併合操作からﾚｺｰﾄﾞを取り出す動作について説明する。	READ文との違いについて議論する。		
併合機能	具体的なﾃﾞｰﾀで併合の動作を説明する。	ﾌｧｲﾙ中のｷｰが並んでいない場合について議論する。	複数ｷｰ項目の併合処理のﾌﾛｰﾁｬｰﾄを書いてみる。	汎用ﾕｰﾃｨﾘﾃｨで併合処理ができるように実習する。
ＭＥＲＧＥ文	MERGEでのｷｰ指定や出力手続きの動作について説明する。	SORTとMERGEの違いについて議論する。		

第８章　ファイルの整列と併合

８．１　整列機能

整列（ＳＯＲＴ）機能は、ユーザが指定したレコードの特定項目をキーとして、ファイル中のレコードの順序を昇順や降順に並べかえることである。

施設名	電話番号	開始年月日
北海道.室蘭	0143-43-1001	1988.04.11
北海道.美唄	01266-8-8668	1989.04.14
青森　.青森	0177-39-1311	1989.04.14
岩手　.北上	0197-64-6551	1991.04.11
福島　.いわき	0246-56-0711	1991.04.12
栃木　.真岡	0285-83-1234	1989.04.10
新潟　.小千谷	0258-82-8282	1989.04.20
兵庫　.西播磨	07915-8-0011	1991.04.09
山口　.周南	0833-72-8000	1991.04.17
愛媛　.今治	0898-22-8000	1989.04.17
福岡　.久留米	0942-32-3311	1989.04.18
福岡　.直方	09492-3-0200	1990.04.23
長崎　.諫早	0957-25-2131	1988.04.15
大分　.中津	0979-22-1122	1990.04.16
宮崎　.延岡	0982-37-7788	1990.04.16

並べかえのキー項目

開始年月日の昇順に並べかえる　←　整列機能

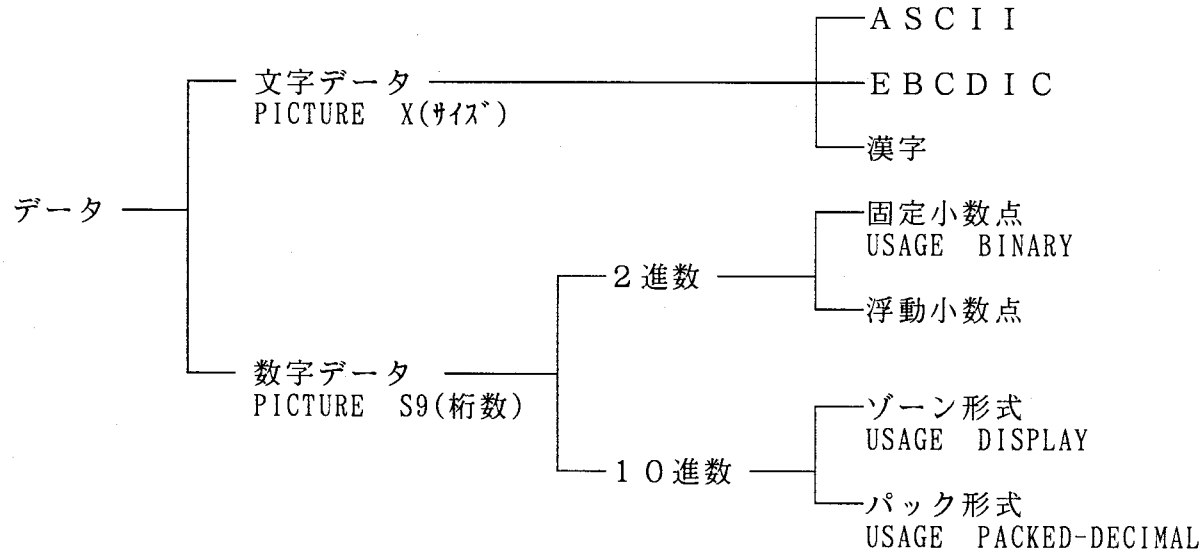
施設名	電話番号	開始年月日
北海道.室蘭	0143-43-1001	1988.04.11
長崎　.諫早	0957-25-2131	1988.04.15
栃木　.真岡	0285-83-1234	1989.04.10
北海道.美唄	01266-8-8668	1989.04.14
青森　.青森	0177-39-1311	1989.04.14
愛媛　.今治	0898-22-8000	1989.04.17
福岡　.久留米	0942-32-3311	1989.04.18
新潟　.小千谷	0258-82-8282	1989.04.20
大分　.中津	0979-22-1122	1990.04.16
宮崎　.延岡	0982-37-7788	1990.04.16
福岡　.直方	09492-3-0200	1990.04.23
兵庫　.西播磨	07915-8-0011	1991.04.09
岩手　.北上	0197-64-6551	1991.04.11
福島　.いわき	0246-56-0711	1991.04.12
山口　.周南	0833-72-8000	1991.04.17

並べかえには、次の種類がある。

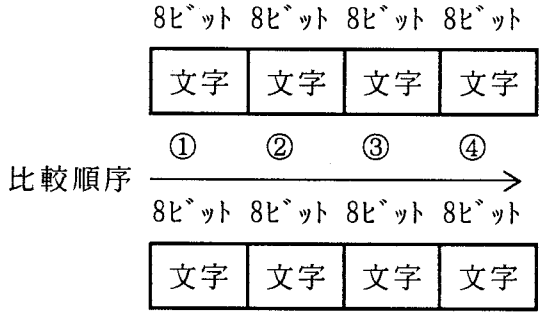
①昇順（ASCENDING）　例えば数字では、０、１、２から８、９であり、
英字では、Ａ、Ｂ、ＣからＹ、Ｚの順になる。

②降順（DESCENDING）　例えば数字では、９、８、７から１、０であり、
英字では、Ｚ、Ｙ，ＸからＢ、Ａの順になる。

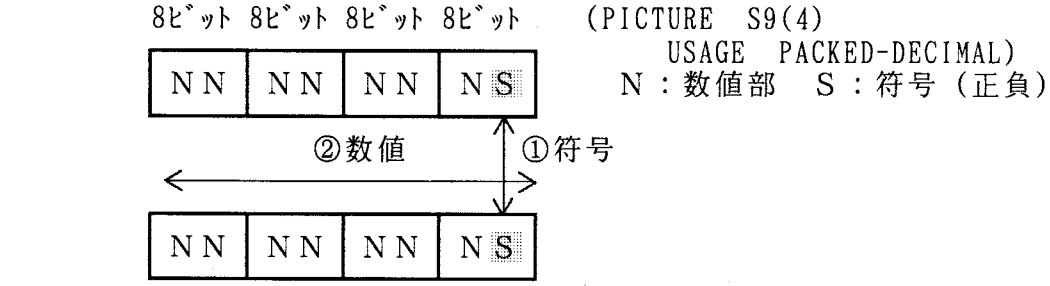
並べかえる項目のデータを分類すると次のようになる。



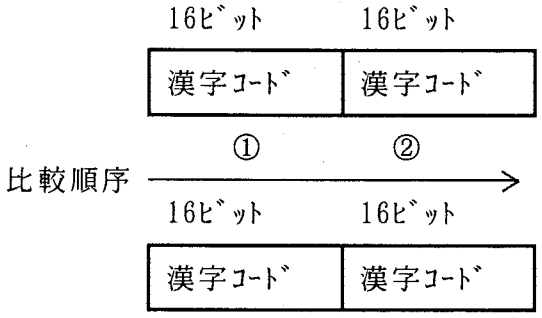
（１）　文字データは、単にビット列の比較を行う。



（２）　数字データは、符号を考慮して数値を比較する。



（３）　漢字データは、漢字コードをビット列として比較する。



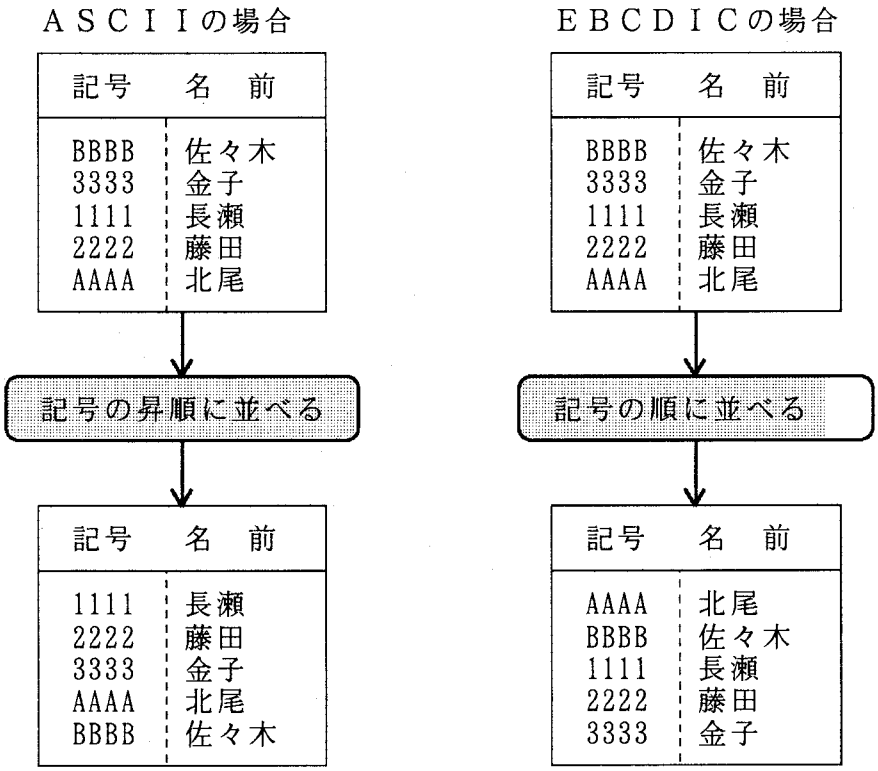
現在、使用されているコンピュータの文字データのコード体系は、概ね次の２種類に分類できる。

①ＡＳＣＩＩ（American Standard Code for Information Interchange）コード
②ＥＢＣＤＩＣ（Extended Binary Coded Decimal Interchange Code）コード

基本的な文字と数字（USAGE DISPLAY）の比較順序は、次の通り。

コード体系	小 <---------- 比較条件 ----------> 大
ＡＳＣＩＩ	０ ＜ １ ＜ ２ ～ ７ ＜ ８ ＜ ９ ＜ Ａ ＜ Ｂ ＜ Ｃ ～ Ｙ ＜ Ｚ
ＥＢＣＤＩＣ	Ａ ＜ Ｂ ＜ Ｃ ～ Ｘ ＜ Ｙ ＜ Ｚ ＜ ０ ＜ １ ＜ ２ ～ ８ ＜ ９

データ項目が数字のみ、または、英字のみで構成されている場合は、どちらのコード体系であっても問題はないが、数字と英字が混在する場合には注意が必要である。



ＣＯＢＯＬでは、コード体系を変えて比較処理（文字）ができる機能を持っている

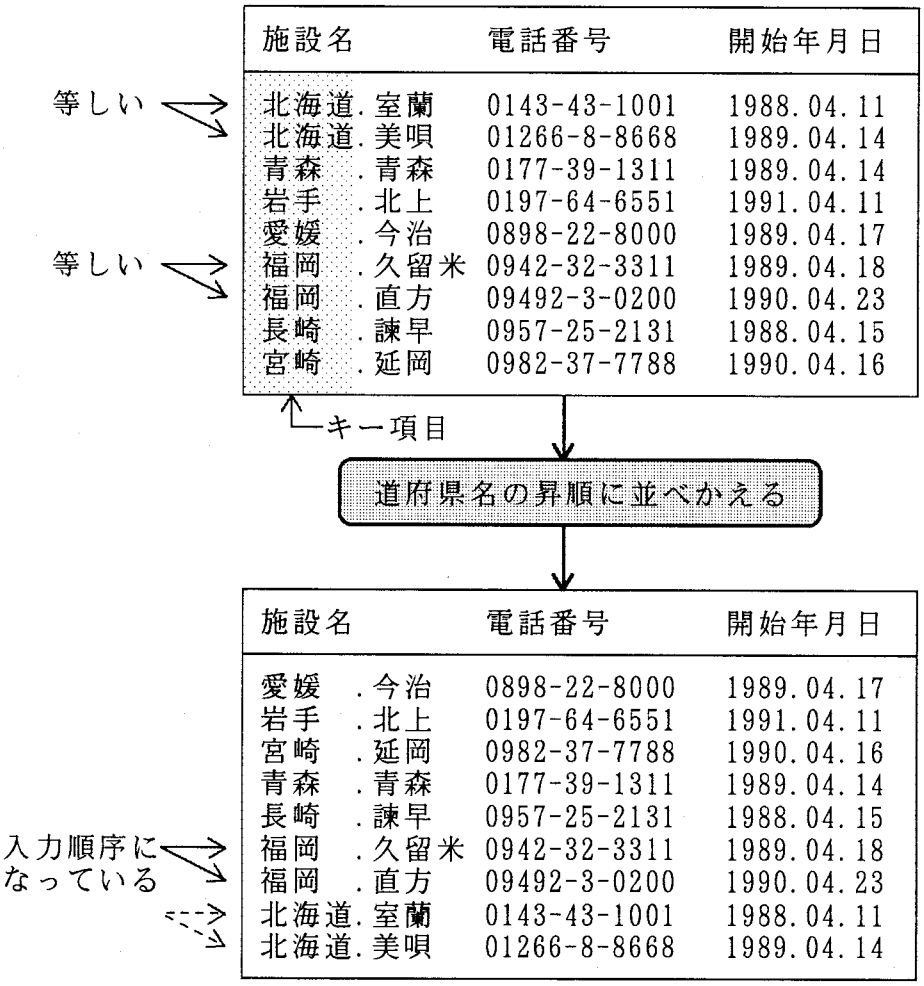
```
ENVIRONMENT  DIVISION.
CONFIGURATION  SECTION.
    :
OBJECT-COMPUTER.   計算機名
                   PROGRAM  COLLATING  SEQUENCE  IS  符号系名.
```

また、整列処理（ＳＯＲＴ）の時だけコード体系を変えて比較することもできる。

```
SORT  ﾌｧｲﾙ名  ON  ASCENDING  KEY  ﾃﾞｰﾀ名  ･･･
              COLLATING  SEQUENCE  IS  符号系名
                :
```

整列処理でレコードを並べかえる時、キー項目のデータが等しい場合に、そのレコードの順序を指定することもできる。

①比較結果が等しい場合、入力の順序に並べる。
②比較結果が等しい場合、順序は不定とする。



ＣＯＢＯＬのＳＯＲＴ文でキー項目が等しい場合に、レコードの入力順序を保つ指定は次の通り。この句を省略すると等しい場合のレコード順序は不定となる。

```
SORT ﾌｧｲﾙ名 ON ASCENDING KEY ﾃﾞｰﾀ名 ･･･
          WITH DUPLICATES IN ORDER
            :
```

８．２　ＳＯＲＴの定義

ＣＯＢＯＬプログラムにおけるＳＯＲＴ（整列機能）の全体像を解説する。

ＳＯＲＴの定義の概要

```
IDENTIFICATION  DIVISION.

ENVIRONMENT  DIVISION.
INPUT-OUTPUT  SECTION.
FILE-CONTROL.
    SELECT  整列ﾌｧｲﾙ名  ASSIGN  TO  作成者語.                    ①
      :

I-O-CONTROL.
    SAME  SORT  AREA  ﾌｧｲﾙ名     ‥                             ②

DATA  DIVISION.
FILE  SECTION.

SD  整列ﾌｧｲﾙ名.
01  整列ﾚｺｰﾄﾞ記述項                                           ③
    03  整列ｷｰ     PIC  X(15).                                ④
      :

PROCEDURE  DIVISION.
主処理  SECTION..
      :
    SORT  整列ﾌｧｲﾙ名                                          ⑤
        ON  ASCENDING/DESCENDING  KEY  整列ｷｰ                 ⑥
        INPUT  PROCEDURE  前処理                              ⑦
        OUTPUT  PROCEDURE  後処理                             ⑧
      :
    STOP RUN

前処理  SECTION.
      :
    RELEASE  整列ﾚｺｰﾄﾞ  FROM  一意名                           ⑨
      :
    EXIT.

後処理  SECTION.
      :
    RETURN  整列ﾌｧｲﾙ  AT  END  GO  TO  後処理終了.            ⑩
      :
後処理終了.
    EXIT.
```

①ファイル名
整列ファイル名を定義する。

②入出力制御
整列ファイルでバッファを共有する場合に定義する。

③整列レコード記述
整列レコードのデータ構造とデータ属性を定義する。

④整列キー
整列に必要なキー項目を定義する。
この項目は、SORTの「ASCENDING/DESCENDING　KEY」で明示的に指定することで整列ｷｰとして認識される。

⑤ＳＯＲＴ文
整列処理の実行を指定する。

⑥ASCENDING／DESCENDING　ｷｰ
ここで、整列ｷｰの項目と昇順、降順を指定する。それぞれ複数の指定ができる。
昇順　－　ASCENDING
降順　－　DESCENDING

⑦INPUT　PROCEDURE
整列処理の前処理を指定する。
前処理が必要ない場合は、「USING　入力ﾌｧｲﾙ名」にする。
INPUT　PROCEDUREの考え方は、PERFORMと同じである。

```
INPUT　PROCEDURE　前処理
　　↓
PERFORM　前処理
```

⑧OUTPUT　PROCEDURE
整列処理の後処理を指定する。
後処理が必要ない場合は、「GIVING　出力ﾌｧｲﾙ名」にする。
OUTPUT　PROCEDUREの考え方は、PERFORMと同じである。

```
OUTPUT　PROCEDURE　後処理
　　↓
PERFORM　後処理
```

⑨RELEASE文
整列の前処理で、整列レコードを整列処理へ受け渡す。
RELEASE文の考え方は、WRITE文と同じである。

```
RELEASE　整列ﾚｺｰﾄﾞ　[FROM　一意名]
　　↓
WRITE　　ﾚｺｰﾄﾞ名　　[FROM　一意名]
```

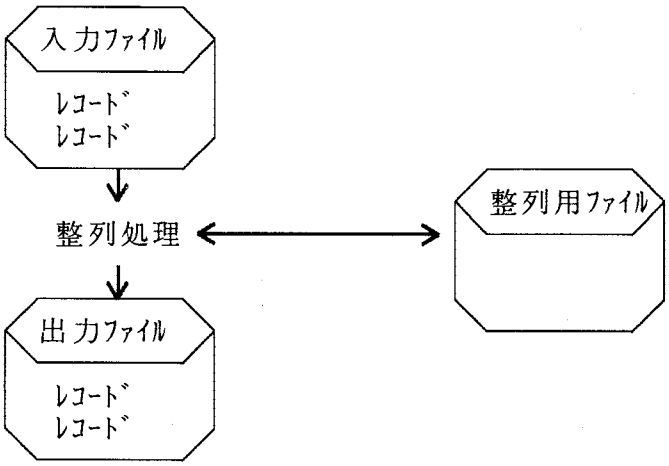
⑩RETURN文
整列の後処理で、整列済みレコードを受け取る。
RETURN文の考え方は、READ文と同じである。

```
RETURN　整列ﾌｧｲﾙ名　AT　END　～
　　↓
READ　　ﾌｧｲﾙ名　　　AT　END　～
```

８．２．１　ＳＯＲＴの機能

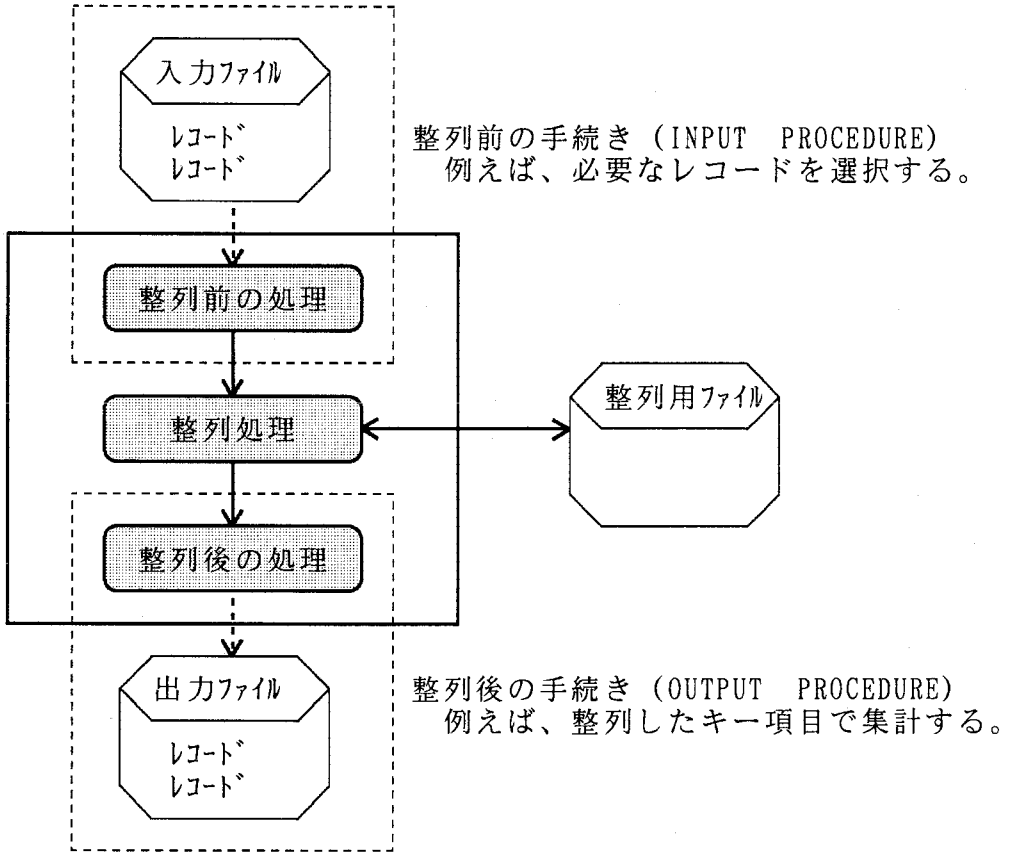
ＳＯＲＴには、次の機能がある。
①一個以上の入力ファイルを指定して整列処理を行い出力ファイルへ書き出す。

```
SORT   整列ﾌｧｲﾙ名
    ON   ASCENDING/DESCENDING   KEY   整列ｷｰ
    USING    入力ﾌｧｲﾙ名
    GIVING   出力ﾌｧｲﾙ名
```



②整列処理に渡す、または、整列処理から渡されたレコードを加工する。
加工にはレコードの選択など整列前に行う手続きと、整列されたレコードを集計するなどの整列後に行う手続きを指定できる。

```
SORT   整列ﾌｧｲﾙ名
    ON   ASCENDING/DESCENDING   KEY   整列ｷｰ
    INPUT    PROCEDURE   整列前処理
    OUTPUT   PROCEDURE   整列後処理
```

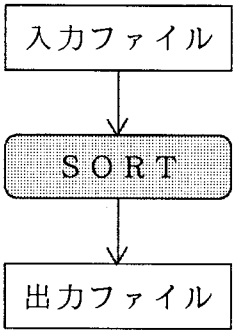


整列前の手続き（INPUT　PROCEDURE）
　例えば、必要なレコードを選択する。

整列後の手続き（OUTPUT　PROCEDURE）
　例えば、整列したキー項目で集計する。

（1）単純ＳＯＲＴ

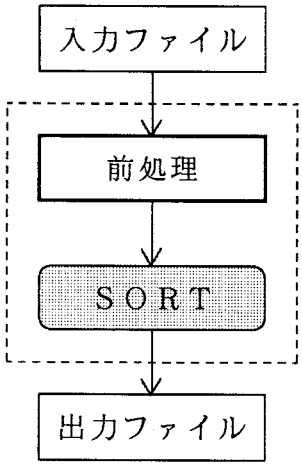
入力ファイルのレコードを指定のキーで並べかえて出力ファイルに書き出す。




```
SORT  ﾌｧｲﾙ名1
    ON  ASCENDING  KEY  ﾃﾞｰﾀ名
    USING   入力ﾌｧｲﾙ
    GIVING  出力ﾌｧｲﾙ
```

（2）ＳＯＲＴ前の処理

入力ファイルのレコードを読み込んで、データチェックなどを行い必要なレコードを指定のキーで並べかえて出力ファイルに書き出す。



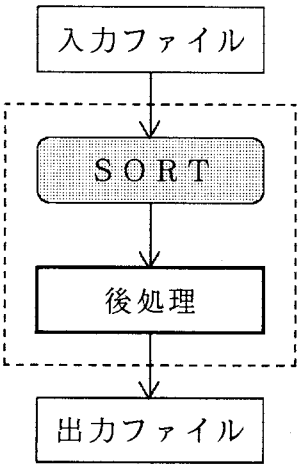

```
      SORT  ﾌｧｲﾙ名
         ON  ASCENDINGKEY  ﾃﾞｰﾀ名
         INPUT  PROCEDURE  前処理
         GIVING  出力ﾌｧｲﾙ

  前処理  SECTION.
      OPEN  INPUT  入力ﾌｧﾙ

      READ  入力ﾌｧｲﾙ
      (ﾃﾞｰﾀﾁｪｯｸ)
      RELEASE  整列ﾚｺｰﾄﾞ
```

（3）ＳＯＲＴ後の処理

ＳＯＲＴ機能が入力ファイルを自動的に読み込んで、指定のキーで並べかえた後、整列レコードを取り出して集計処理などを行う。



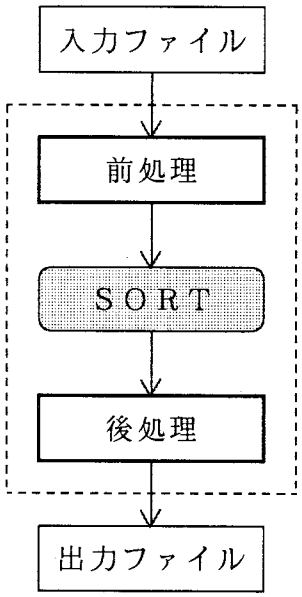

```
  SORT  ﾌｧｲﾙ名1
    ON  ASCENDING  KEY  ﾃﾞｰﾀ名
    USING  入力ﾌｧｲﾙ
    OUTPUT  PROCEDURE  後処理



後処理  SECTION.
    OPEN  OUTPUT  出力ﾌｧｲﾙ
       :
    RETURN  ﾌｧｲﾙ名  AT  END  ～
    WRITE  出力ﾚｺｰﾄﾞ
```

（４）ＳＯＲＴ前後の処理

前処理では、入力ファイルを読み込んでデータチェックなどの処理を行い、必要なレコードを指定のキーで並べかえる。
後処理では、整列されたレコードを取り出して集計処理などを行う。




```
    SORT  ﾌｧｲﾙ名1
      ON  ASCENDING  KEY  ﾃﾞｰﾀ名
      INPUT   PROCEDURE  前処理
      OUTPUT  PROCEDURE  後処理


前処理  SECTION.

    READ  入力ﾌｧｲﾙ
      :
    RELEASE  整列ﾚｺｰﾄﾞ

後処理  SECTION.

    RETURN  ﾌｧｲﾙ名
      :
    WRITE  出力ﾚｺｰﾄﾞ
```

８．２．２　整列併合用ファイルの定義

ＳＯＲＴ文で指定する整列用ファイル、ＭＥＲＧＥ文で指定する併合用ファイルを定義する。

```
ENVIRONMENT  DIVISION.

INPUT-OUTPUT  SECTION.
FILE-CONTROL.

     SELECT  ﾌｧｲﾙ名  ASSIGN  TO  作成者語1.
```

整列併合用ファイルには、順ファイルや索引ファイルなどで記述する「ORGNIZATION IS SEQUENTIAL/INDEXED/RELATIVE」などはいらない。ＡＳＳＩＧＮ句だけを指定する。

整列併合用ファイルは、ＳＯＲＴ／ＭＥＲＧＥ文、ＲＥＴＵＲＮ文で参照する。
ＯＰＥＮ、ＲＥＡＤなどの入出力文で参照されることはない。

ＦＩＬＥ　ＳＥＣＴＩＯＮでの整列併合用ファイルの定義は、「ＦＤ」ではなく、「ＳＤ」でおこなう。

```
     INPUT-OUTPUT SECTION.
     FILE-CONTROL.
          SELECT  ﾌｧｲﾙ名1  ASSIGN  TO  作成者語.    ←順ﾌｧｲﾙ
          SELECT  ﾌｧｲﾙ名2  ASSIGN  TO  作成者語.    ←整列併合用ﾌｧｲﾙ

     FILE  SECTION.
     FD  ﾌｧｲﾙ名1  ～

     SD  ﾌｧｲﾙ名2  ～
```

８．２．３　整列併合用ファイルのレコードの定義

整列併合用ファイルのレコード構造を定義する。

```
DATA  DIVISION.
FILE  SECTION.

SD  ﾌｧｲﾙ名
     [RECORD  CONTAINS  整数1  CHARACTERS]
     [DATA  RECORD  IS  {ﾃﾞｰﾀ名2} ･･･]

01  ﾃﾞｰﾀ名2.
    整列併合ｷｰ記述項
```

ＳＤには、順ファイルや索引ファイルなどで記述できる「LABEL　RECORD」句や「BLOCK　CONTAINS」句は記述できない。

ＳＤで定義した整列併合レコード記述は、ＳＯＲＴ／ＭＥＲＧＥ文のＡＳＣＥＮＤ－ＩＮＧ／ＤＥＳＣＥＮＤＩＮＧ　ＫＥＹ指定やＲＥＬＥＡＳＥ文で参照する。

（１）　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整列併合用データの属性とサイズをＰＩＣＴＵＲＥ句で定義する。
ＳＯＲＴ／ＭＥＲＧＥ文で指定（ASCENDING/DESCENDING　KEY）するキーのデータ名はこのレコード中に定義する。
索引ファイルのレコードキー（RECORD KEY句）のデータ名は１個のみであるが、整列併合用キー（ASCENDING/DESCENDING　KEY句）のデータ名は複数定義できる。

整列併合用ﾚｺｰﾄﾞ記述

```
SD  ﾌｧｲﾙ名.
01  ﾚｺｰﾄﾞ名.
    03 ﾃﾞｰﾀ名          属性/ｻｲｽﾞ
    03 整列併合ｷｰ-1    属性/ｻｲｽﾞ
    03 ﾃﾞｰﾀ名          属性/ｻｲｽﾞ
    03 FILLER          属性/ｻｲｽﾞ
    03 整列併合ｷｰ-2    属性/ｻｲｽﾞ
    03 ﾃﾞｰﾀ名          属性/ｻｲｽﾞ
    03 ﾃﾞｰﾀ名          属性/ｻｲｽﾞ
```

索引ﾌｧｲﾙのﾚｺｰﾄﾞ記述

```
FD  ﾌｧｲﾙ名  BLOCK 10 RECORDS.
01  ﾚｺｰﾄﾞ名.
    03 ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
    03 ﾚｺｰﾄﾞｷｰ     属性/ｻｲｽﾞ
    03 ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
    03 FILLER      属性/ｻｲｽﾞ
    03 ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
    03 ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
    03 ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
    03 ﾃﾞｰﾀ名      属性/ｻｲｽﾞ
```

```
SORT  ﾌｧｲﾙ名  ON  ASCENDING   KEY  整列併合ｷｰ-1
                  DESCENDING  KEY  整列併合ｷｰ-2
      INPUT   PROCEDURE  ～
      OUTPUT  PROCEDURE  ～
```

８．２．４　ＳＯＲＴ

ＳＯＲＴ文は、他のファイルからレコードを読んで、または、入力手続きを実行して整列用レコードを作成し、指定されたキーに従ってレコード順序を並べかえる。
その後、整列用レコードを出力ファイルに書き出す。または、出力手続きを実行して整列済みの整列用レコードを取り出して処理することができる。

```
SORT  ﾌｧｲﾙ名1
      {ON  { ASCENDING  }  KEY   {ﾃﾞｰﾀ名1} ･･･} ･･･
           { DESCENDING }
      [WITH  DUPLICATES  IN  ORDER]
      [COLLATING  SEQUENCE  IS  符号系名1]

      { USING   {ﾌｧｲﾙ名2} ･･･                                  }
      { INPUT PROCEDURE, IS   手続き名1   [THRU  手続き名2]   }

      { GIVING   {ﾌｧｲﾙ名3} ･･･                                 }
      { OUTPUT PROCEDURE IS  手続き名1   [THRU  手続き名2]    }
```

（１）　ＡＳＣＥＮＤＩＮＧ／ＤＥＳＣＥＮＤＩＮＧ　ＫＥＹ指定

ＡＳＣＥＮＤＩＮＧ指定をすると、比較条件に従ってキー項目の値の小さい方から大きい方へレコードがそろえられる。
ＤＥＳＣＥＮＤＩＮＧ指定をすると、比較条件に従ってキー項目の値の大きい方から小さい方へレコードがそろえられる。

整列処理の順序（昇順、降順）と、整列レコード記述内のデータ項目を指定する。
キーの比較条件の基本は次の通り。
- 数字項目　　－　符号を考慮
- 非数字項目　－　ｺﾝﾋﾟｭｰﾀの照合順序(COLLATING SEQUENCE)

```
FILE  SECTION.

SD  住所ﾌｧｲﾙ.
01  個人ﾚｺｰﾄﾞ.
    03  生年月日  PIC  9(8).
    03  氏名      PIC  X(20).
        :
    03  住所      PIC  X(50).

PROCEDURE  DVISION.

    SORT  住所ﾌｧｲﾙ
      ON  ASCENDING  KEY  生年月日,  氏名
       :
```

（２）　ＤＵＰＬＩＣＡＴＥＳ指定

複数の整列レコードのキーの値が等しいときの順序を指定する。
- USING指定　　－　入力ファイルの順序
- INPUT PROCEDURE指定　－　RELEASE文で渡した順序

（３）　ＵＳＩＮＧ指定

前処理を必要としない場合に、整列処理に渡す入力ファイル名を指定する。
ここで指定する入力ファイル名のＯＰＥＮ、ＣＬＯＳＥはＳＯＲＴ文で自動的に行われる。

```
SORT  整列ﾌｧｲﾙ名
    ON  ASCENDING  KEY  生年月日,  氏名
    USING  入力ﾌｧｲﾙ1,  入力ﾌｧｲﾙ2,  入力ﾌｧｲﾙ3
      :
```

USINGで指定する入力ファイル名には、順ファイル、索引ファイル、相対ファイルが指定できる。ただし、索引ファイル、相対ファイルでは呼出し法としてACCESS　MODE　IS RANDOM句を書いてはいけない。

入力ファイルのレコードサイズと整列ファイルの関係は次の通り。
・固定長レコード　－　入力ﾌｧｲﾙﾚｺｰﾄﾞｻｲｽﾞ　≧　整列ﾌｧｲﾙﾚｺｰﾄﾞｻｲｽﾞ
・可変長レコード　－　整列ﾌｧｲﾙ最小ｻｲｽﾞ　≦　入力ﾌｧｲﾙｻｲｽﾞ　≦　整列ﾌｧｲﾙ最大ｻｲｽﾞ

（４）　ＩＮＰＵＴ　ＰＲＯＣＥＤＵＲＥ指定

整列処理前にレコードを処理する場合に、その処理の手続き名を指定する。
その場合の実行の順序は、次のとおり。

```
①↓ SORT  整列ﾌｧｲﾙ名
       ON  ASCENDING  KEY  生年月日,  氏名
       INPUT  PROCDURE  IS  前処理
②      :  ←──────────────────────────┐
                                      │
  → 前処理  SECTION.                  │
       OPEN  INPUT  入力ﾌｧｲﾙ.         │
  │ LOOP.                             │
  │    READ  入力ﾌｧｲﾙ  INTO  整列ﾚｺｰﾄﾞ  AT  END  GO  TO  EOF.
③ │    IF  生年月日  <  19980101  GO  TO  LOOP.
  │    RELEASE  整列ﾚｺｰﾄﾞ.            │
  │    GO  TO  LOOP.                  │
  │ EOF.                              │
  │    CLOSE  入力ﾌｧｲﾙ.               │
  ↓ 前処理終了.                  ④    │
       EXIT.  ──────────────────────────┘
```

入力手続き（INPUT PROCEDURE）処理の範囲内では、ＳＯＲＴ文、ＭＥＲＧＥ文、ＲＥＴＵＲＮ文を実行できない。

（５）　ＧＩＶＩＮＧ指定

後処理を必要としない場合、整列機能がデータを書き出す出力ファイル名を指定する。
ここで指定する出力ファイル名のＯＰＥＮ、ＣＬＯＳＥはＳＯＲＴ文で自動的に行われる。

```
SORT  整列ﾌｧｲﾙ名
    ON  ASCENDING  KEY  生年月日,  氏名
      :
    GIVING  出力ﾌｧｲﾙ名
```

ＧＩＶＩＮＧで指定する出力ファイル名には、順ファイル、索引ファイル、相対ファイルが指定できる。ただし、索引ファイル、相対ファイルでは呼出し法としてACCESS MODE IS RANDOM句を書いてはいけない。
ＧＩＶＩＮＧで索引ファイルを指定する場合は、最初のキー項目はASCENDING指定であり、そのデータ項目は索引ファイルの主レコードキーと一致している必要がある。

（６）　ＯＵＰＵＴ　ＰＲＯＣＥＤＵＲＥ指定

整列済みレコードを処理する場合に、その処理の手続き名を指定する。
出力手続き（OUTPUT PROCEDURE）の実行順序は、次のとおり。

```
①  SORT  整列ﾌｧｲﾙ名
       ON  ASCENDING  KEY  生年月日,  氏名
         :
     OUTPUT  PROCEDURE  後処理.
②      :   <-------------------------------------------+
                                                       |
後処理  SECTION.                                       |
    OPEN  OUTPUT  出力ﾌｧｲﾙ.                            |
LOOP.                                                  |
    RETURN  整列ﾌｧｲﾙ  AT  END  GO  TO  EOF.            |
③     :                                                |
    WRITE  出力ﾚｺｰﾄﾞ.                                  |
    GO  TO  LOOP.                                      |
EOF.                                                   |
    CLOSE  出力ﾌｧｲﾙ.                                   |
後処理終了.                                      ④    |
    EXIT.  ----------------------------------------------+
```

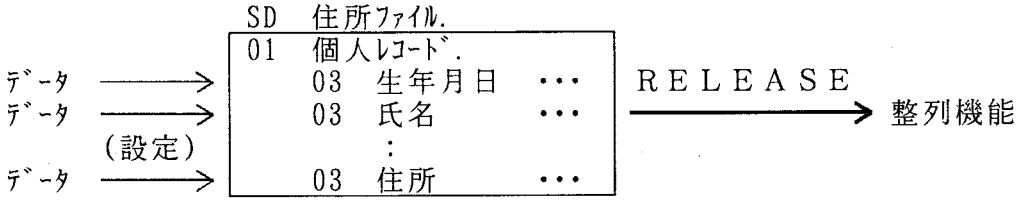
出力手続き処理の範囲内では、ＳＯＲＴ文、ＭＥＲＧＥ文、ＲＥＬＥＡＳＥ文を実行できない。

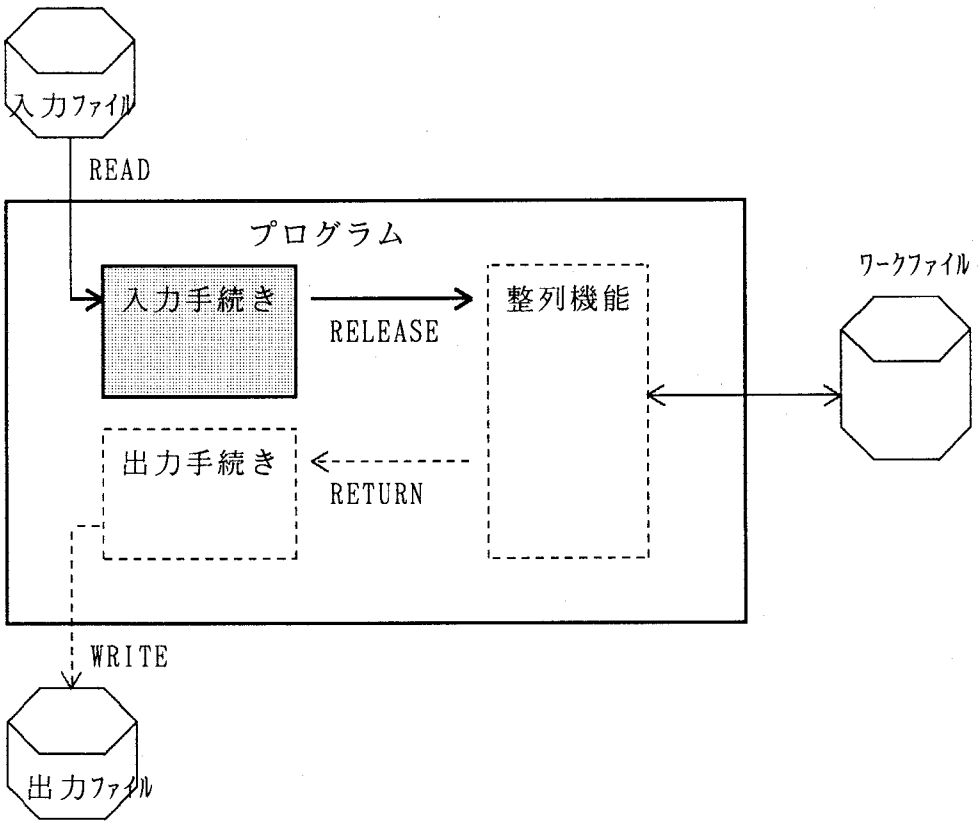
８．２．５　ＲＥＬＥＡＳＥ

ＳＯＲＴの入力手続き（INPUT　PROCEDURE）中で整列機能にレコードを引き渡す。

```
RELEASE  ﾚｺｰﾄﾞ名1   [FROM  一意名1]
```

レコード名には、レベル指示語ＳＤで記述したレコード名を指定する。



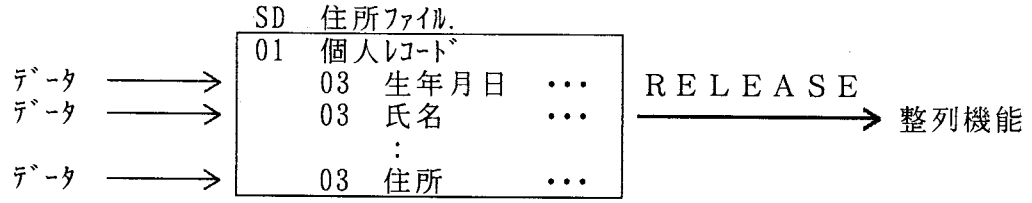


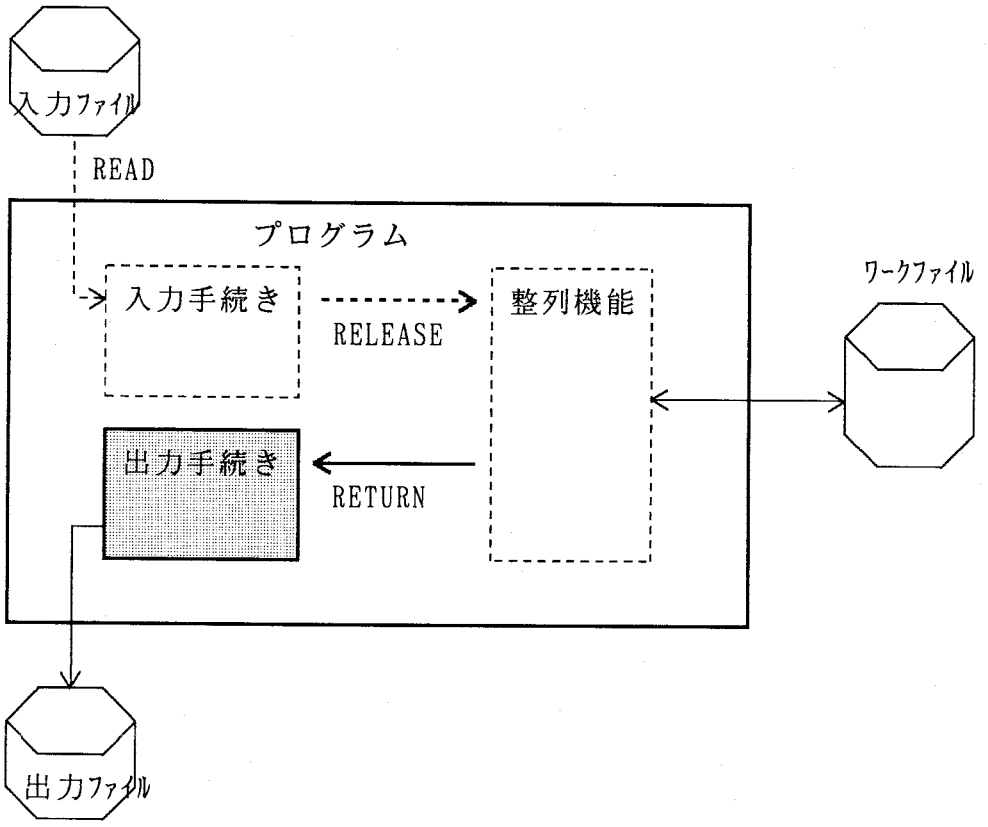
８．２．６　ＲＥＴＵＲＮ

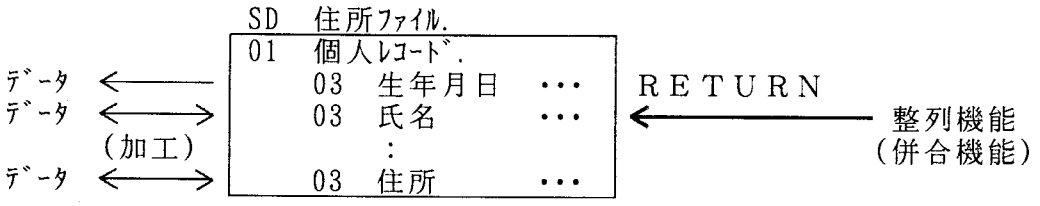
整列（ＳＯＲＴ）の場合は、整列済みの整列レコードを取り出す。
併合（ＭＥＲＧＥ）の場合は、併合されたレコードを取り出す。

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

ファイル名には、レベル指示語ＳＤで記述したファイル名を指定する。
整列併合レコードが終了するとＡＴ　ＥＮＤ指定が実行される。
ＡＴ　ＥＮＤ指定は省略できない。

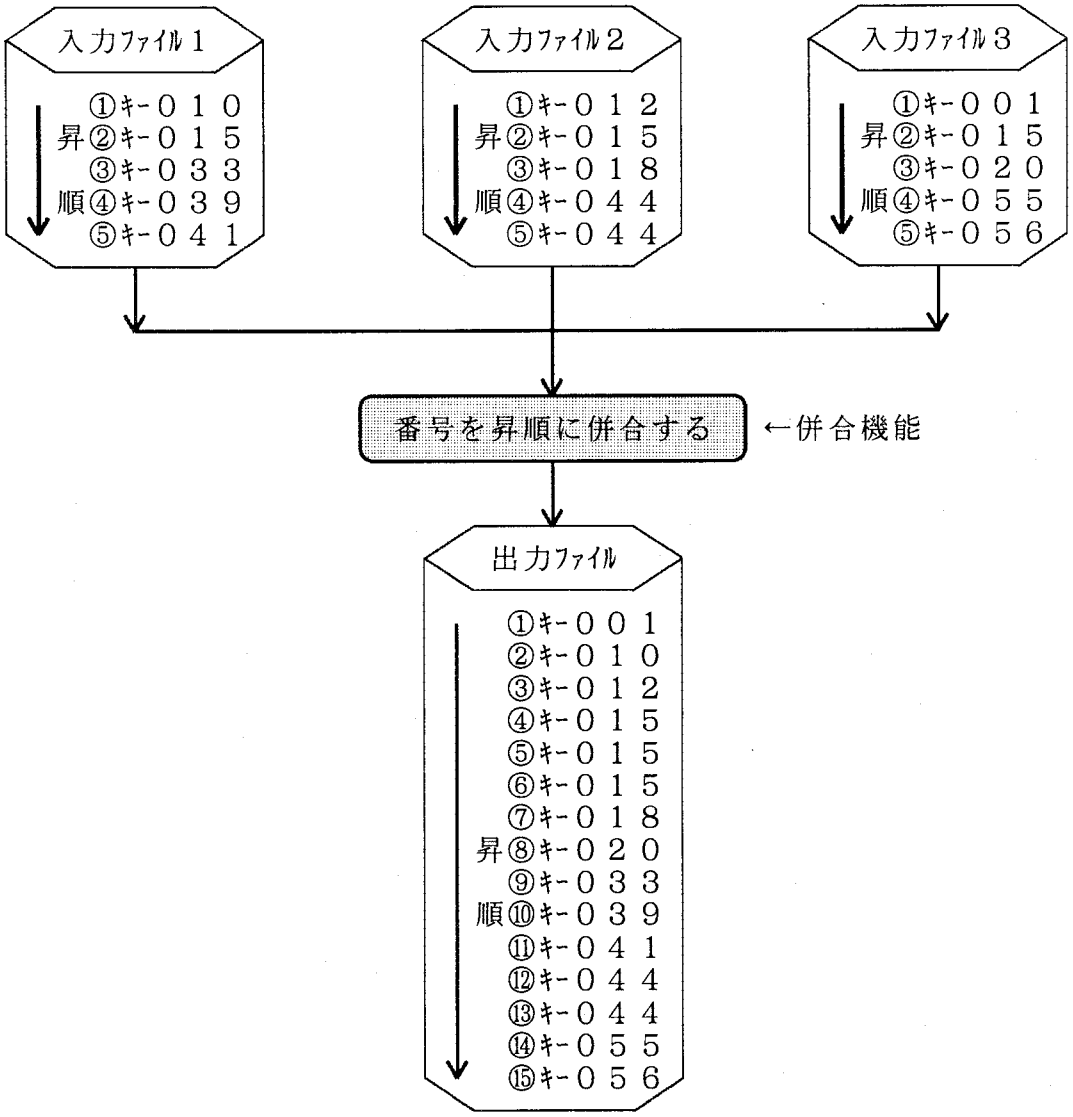






８．３　併合機能

併合（ＭＥＲＧＥ）機能は、ユーザが指定したレコードの特定項目をキーとして、レコードが昇順や降順に並べられた複数のファイルを併合する。

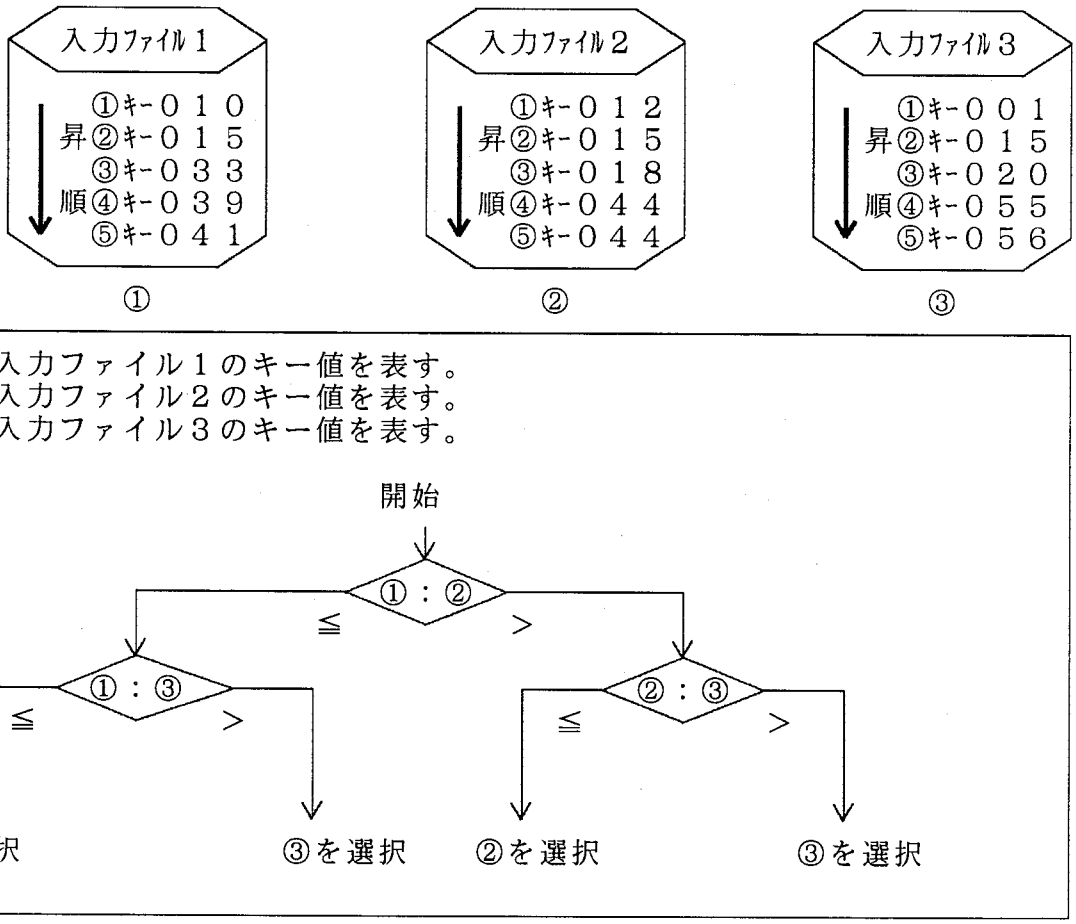


各々の入力ファイルのレコードは、指定したキーに従って昇順、あるいは、降順に整列されている必要がある。

併合キーで昇順（ASCENDING）を指定した場合、各々の入力ファイルの読み込まれたレコードのキー値を比較して一番小さいレコードから順次取り出される。

降順（DESCENDING）を指定した場合は、一番大きなキーを持つレコードから順次取り出される。

前記の３個のファイルの併合処理のキー比較をフローチャートにすると、次のようになる。



このように１個のキー項目でも多くの比較処理が必要である。さらに入力ファイルが増えたり、キー項目が複数指定されるとさらに複雑な比較処理が必要になる。

これらの複雑な処理をＭＥＲＧＥ文は簡単な定義で実行してくれる。

```
MERGE  併合ﾌｧｲﾙ  ON  ASCENDING  KEY  ｷｰ項目1
       USING  入力ﾌｧｲﾙ1  入力ﾌｧｲﾙ2  入力ﾌｧｲﾙ3
       GIVING  出力ﾌｧｲﾙ
```

８．４　ＭＥＲＧＥの定義

ＣＯＢＯＬプログラムにおけるＭＥＲＧＥ（併合機能）の全体像を解説する。

ＭＥＲＧＥの定義の概要

```
IDENTIFICATION  DIVISION.

ENVIRONMENT  DIVISION.
INPUT-OUTPUT  SECTION.
FILE-CONTROL.
    SELECT  併合ﾌｧｲﾙ名    ASSIGN  TO  作成者語.                    ①
    SELECT  入力ﾌｧｲﾙ名1  ASSIGN  TO  作成者語.
    SELECT  入力ﾌｧｲﾙ名2  ASSIGN  TO  作成者語.
    SELECT  入力ﾌｧｲﾙ名3  ASSIGN  TO  作成者語.
      :

I-O-CONTROL.
    SAME  SORT-AREA  併合ﾌｧｲﾙ名     ‥                            ②

DATA  DIVISION.
FILE  SECTION.

SD  併合ﾌｧｲﾙ名.                                                  ③
01  併合ﾚｺｰﾄﾞ記述項
    03  併合ｷｰ     PIC  9(8).                                   ④
      :
FD  入力ﾌｧｲﾙ名1.
01  入力ﾚｺｰﾄﾞ記述項
    03  整列ｷｰ     PIC  9(8).

FD  入力ﾌｧｲﾙ名2.
01  入力ﾚｺｰﾄﾞ記述項
    03  整列ｷｰ     PIC  9(8).

FD  入力ﾌｧｲﾙ名3.
01  入力ﾚｺｰﾄﾞ記述項
    03  整列ｷｰ     PIC  9(8).

PROCEDURE  DIVISION.
主処理  SECTION..
      :
    MERGE  併合ﾌｧｲﾙ名                                            ⑤
        ON  ASCENDING/DESCENDING  KEY  整列ｷｰ                    ⑥
        USING  入力ﾌｧｲﾙ名1  入力ﾌｧｲﾙ名2  入力ﾌｧｲﾙ名3  ‥          ⑦
        OUTPUT  PROCEDURE  後処理                                ⑧
      :
    STOP RUN

後処理  SECTION.
      :
    RETURN  併合ﾌｧｲﾙ  AT  END  GO  TO  後処理終了.               ⑨
      :
後処理終了.
    EXIT.
```

併合機能の説明

①ファイル名
併合ファイル名と複数の入力ファイル名を定義する。必要ならば出力ファイル名の定義も行う。

②入出力制御
併合ファイルでバッファを共有する場合に定義する。

③併合レコード記述
併合レコードのデータ構造とデータ属性を定義する。

④併合キー
併合に必要なキー項目を定義する。
この項目は、MERGEの「ASCENDING／DESCENDING　KEY」で明示的に指定することで併合キーとして認識される。
入力ファイル名のレコードも、併合キーと同じデータ構造とデータ属性を持っていること。

⑤ＭＥＲＧＥ命令
併合処理の実行を指定する。

⑥ＡＳＣＥＮＤＩＮＧ／ＤＥＳＣＥＮＤＩＮＧ　キー
ここで、併合キーの項目と昇順、降順を指定する。それぞれ複数のデータ項目が指定できる。
　昇順　－　ASCENDING
　降順　－　DESCENDING

⑦ＵＳＩＮＧ　ファイル名
併合処理の入力ファイル名として２ファイル以上を指定する。
入力ファイルは、ASCENDING／DESCENDINGｷｰで指定した項目でレコードが並んでいる必要がある。

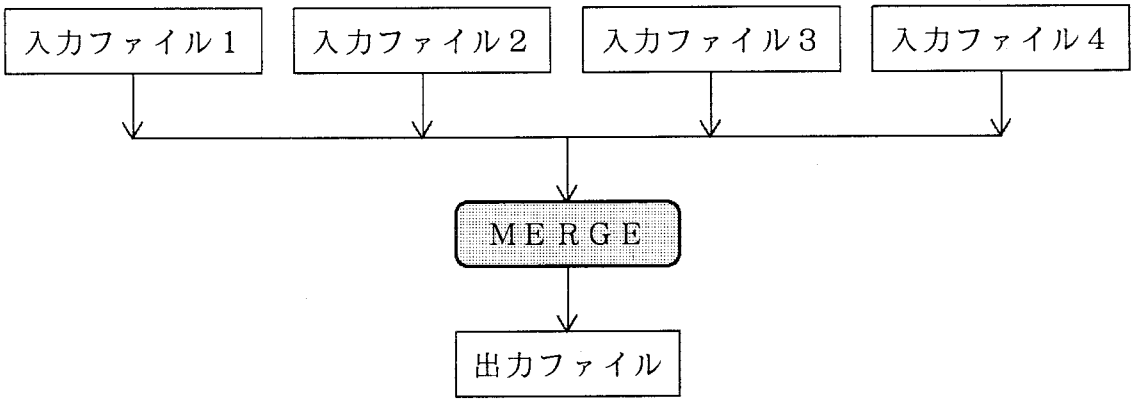
⑧ＯＵＴＰＵＴ　ＰＲＯＣＥＤＵＲＥ
併合処理の後処理を指定する。
OUTPUT　PROCEDUREの考え方は、PERFORMと同じである。
　OUTPUT　PROCEDURE　後処理
　　↓
　PERFORM　後処理

⑨ＲＥＴＵＲＮ文
併合の後処理で、併合済みレコードを受け取る。
ＲＥＴＵＲＮ文の考え方は、ＲＥＡＤ文と同じである。
　RETURN　併合ﾌｧｲﾙ名　AT　END　～
　　↓
　READ　　ﾌｧｲﾙ名　　AT　END　～

８．４．１　ＭＥＲＧＥ機能

（１）　単純ＭＥＲＧＥ

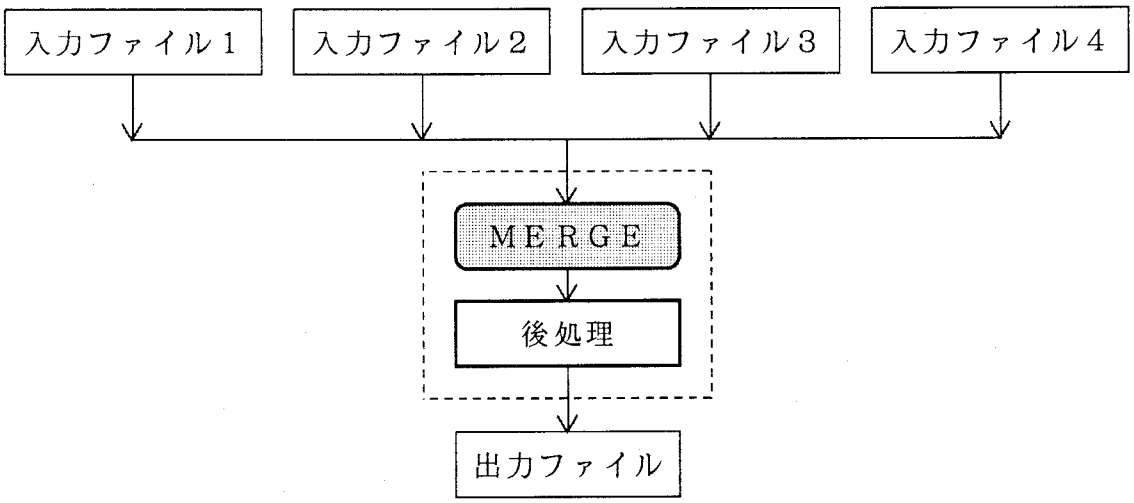
既に指定のキーで並んでいる複数の入力ファイルを読み込んで、指定のキーに従って併合して出力ファイルに書き出す。




```
    MERGE  併合ﾌｧｲﾙ  ON  ASCENDING  KEY  ﾃﾞｰﾀ名1  ﾃﾞｰﾀ名2
        USING   入力ﾌｧｲﾙ1  入力ﾌｧｲﾙ2  入力ﾌｧｲﾙ3  入力ﾌｧｲﾙ4
        GIVING  出力ﾌｧｲﾙ.
```

（２）　ＭＥＲＧＥの後処理

既に指定のキーで並んでいる複数の入力ファイルを読み込んで、指定のキーに従って併合する。
併合した一連のレコードを取り出して集計処理などを行い、その結果を出力する。




```
    MERGE  併合ﾌｧｲﾙ  ON  ASCENDING  KEY  ﾃﾞｰﾀ名1  ﾃﾞｰﾀ名2
        USING   入力ﾌｧｲﾙ1  入力ﾌｧｲﾙ2  入力ﾌｧｲﾙ3  入力ﾌｧｲﾙ4
        OUTPUT  PROCEDURE  後処理.

後処理  SECTION.
    RETURN  併合ﾌｧｲﾙ  AT END  ～
```

８．４．２　ＭＥＲＧＥ

ＭＥＲＧＥ文は、指定したキーに従って、昇順または降順にそろえられた複数の入力ファイルを併合する。
その後、併合されたレコードを出力ファイルに書き出す。または、出力手続きを指定して併合されたレコードを受け取って処理する。

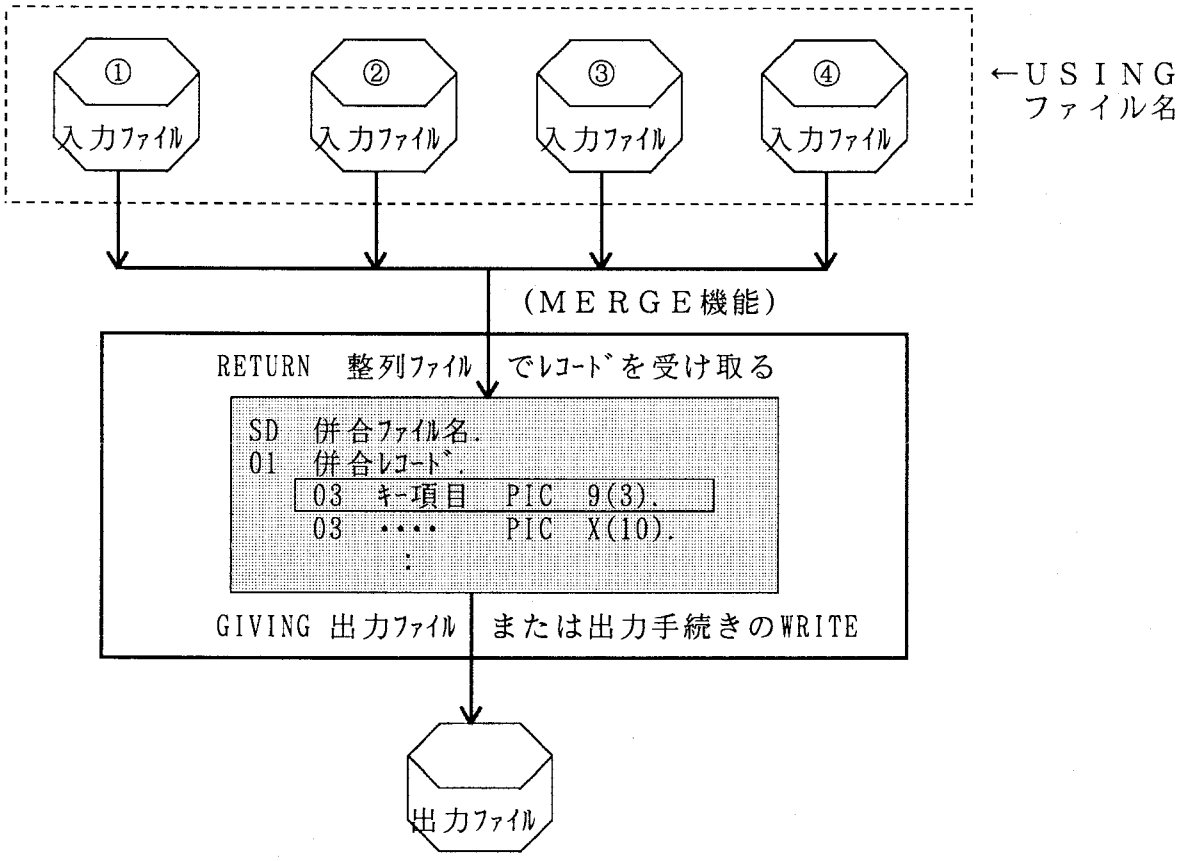
```
MERGE  ﾌｧｲﾙ名1
     {ON  { ASCENDING  }  KEY   {ﾃﾞｰﾀ名1} ･･･} ･･･
          { DESCENDING }
     [COLLATING  SEQUENCE  IS  符号系名1]
      USING  ﾌｧｲﾙ名2   {ﾌｧｲﾙ名3} ･･･
     { GIVING   {ﾌｧｲﾙ名4} ･･･                                  }
     { OUTPUT  PROCEDURE  IS  手続き名1   [THRU  手続き名2]   }
```

ファイル名１は、データ部の整列併合用ファイル記述項（ＳＤ）で定義したファイル名を指定する。ＵＳＩＮＧやＧＩＶＩＮＧで指定するファイル名には、ＦＤで定義したファイル名を指定する。

（１）　ＵＳＩＮＧ指定

併合処理に渡す入力ファイル名を２個以上指定する。ここで指定する入力ファイル名のＯＰＥＮ、ＣＬＯＳＥはＭＥＲＧＥ機能で自動的に行われる。

ＵＳＩＮＧ、または、ＧＩＶＩＮＧに指定するファイル名には、順ファイル、索引ファイル、相対ファイルが指定できる。
索引ファイル、相対ファイルを指定する場合、そのファイルの呼出し法には、ACCESS MODE IS RANDOM句を指定できない。



（２）　ＧＩＶＩＮＧ指定

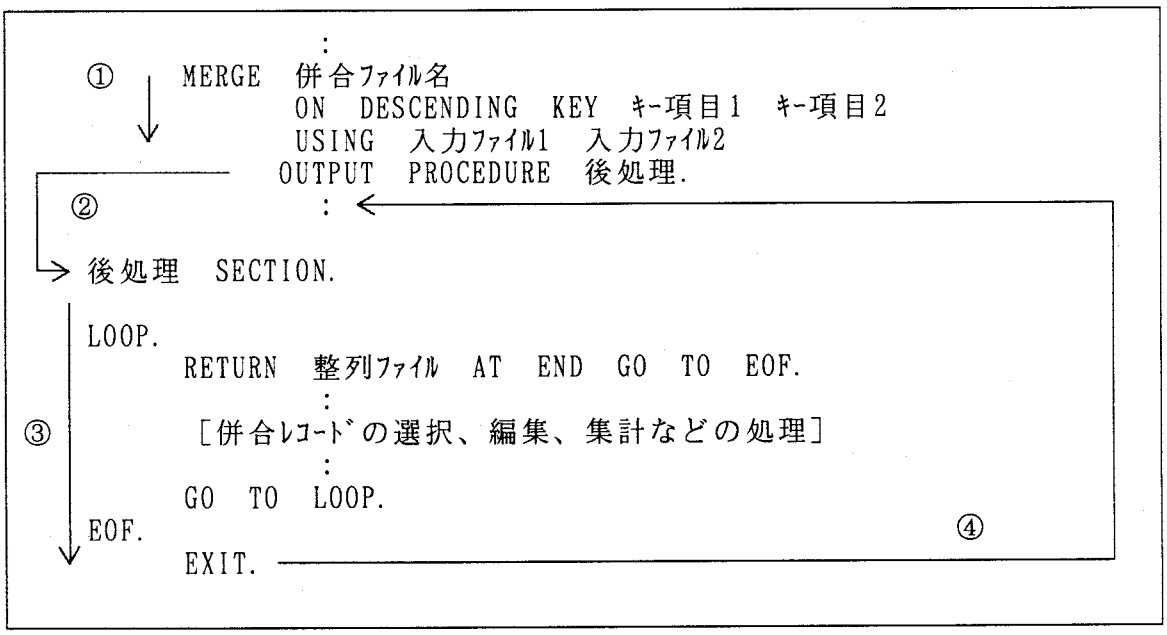
後処理である出力手続きを必要としない場合に、併合された全レコードを書き出すための出力ファイルを指定する。

ＧＩＶＩＮＧに索引ファイルを指定する場合、キー項目はASCENDING指定でなければならない。そのキー項目は索引レコードの主レコードキーと同じ文字位置である必要がある。
入力ファイルのキー項目の値が等しいときは、ＵＳＩＮＧで指定した入力ファイル名の順番でレコードが書き出される。

```
MERGE  併合ﾌｧｲﾙ名
       ON  DESCENDING  KEY  ｷｰ項目  ･･･
       USING  入力ﾌｧｲﾙ名1  入力ﾌｧｲﾙ名2  ･･･
      GIVING  出力ﾌｧｲﾙ名
```

（３）　ＯＵＴＰＵＴ　ＰＲＯＣＥＤＵＲＥ指定

併合されたレコードを受け取る場合に、出力手続きを指定する。
ＲＥＴＵＲＮ文で受け取るレコードはレベル指示語ＳＤのレコード領域に入れられる。
出力手続きの実行順序は、次のとおり。



出力手続き内では、ＭＥＲＧＥ、ＳＯＲＴ、ＲＥＬＥＡＳＥ文を実行できない。

指導上の留意点

◎ＳＯＲＴの体験
ＭＳ－ＤＯＳのＳＯＲＴコマンドを使用してファイルの整列処理を体験させる。

ＳＯＲＴ　［／Ｒ］　［／＋ｎ］　＜入力ﾌｧｲﾙ名　＞出力ﾌｧｲﾙ名

［／Ｒ］：降順に並べる場合に指定する。
［／＋ｎ］：キーの位置（１～）を指定する。

入力データの例

```
北海道.室蘭    0143-43-1001 1988.04.11
北海道.美唄    01266-8-8668 1989.04.14
青森　.青森    0177-39-1311 1989.04.14
岩手　.北上    0197-64-6551 1991.04.11
福島　.いわき  0246-56-0711 1991.04.12
栃木　.真岡    0285-83-1234 1989.04.10
新潟　.小千谷  0258-82-8282 1989.04.20
兵庫　.西播磨  07915-8-0011 1991.04.09
山口　.周南    0833-72-8000 1991.04.17
愛媛　.今治    0898-22-8000 1989.04.17
福岡　.久留米  0942-32-3311 1989.04.18
福岡　.直方    09492-3-0200 1990.04.23
長崎　.諫早    0957-25-2131 1988.04.15
大分　.中津    0979-22-1122 1990.04.16
宮崎　.延岡    0982-37-7788 1990.04.16
```

数字や英字は昇順、降順が論理的な意味を持つが、漢字コードでは順序に意味がないことを理解させる。
ＳＯＲＴコマンドで漢字項目を昇順に並べた場合は下記の通り。

```
愛媛　.今治    0898-22-8000 1989.04.17
岩手　.北上    0197-64-6551 1991.04.11
宮崎　.延岡    0982-37-7788 1990.04.16
山口　.周南    0833-72-8000 1991.04.17
新潟　.小千谷  0258-82-8282 1989.04.20
青森　.青森    0177-39-1311 1989.04.14
大分　.中津    0979-22-1122 1990.04.16
長崎　.諫早    0957-25-2131 1988.04.15
栃木　.真岡    0285-83-1234 1989.04.10
福岡　.久留米  0942-32-3311 1989.04.18
福岡　.直方    09492-3-0200 1990.04.23
福島　.いわき  0246-56-0711 1991.04.12
兵庫　.西播磨  07915-8-0011 1991.04.09
北海道.室蘭    0143-43-1001 1988.04.11
北海道.美唄    01266-8-8668 1989.04.14
```

また、プログラミング言語で記述しなくても、通常オペレーティング・システムにはＳＯＲＴコマンド（ユーティリティ）があることも説明する。
ＣＯＢＯＬのＳＯＲＴ文は内部的にそれらのＳＯＲＴユーティリティを呼んで実現している。そのため、キー項目の個数、レコードサイズなどの制限事項はＣＯＢＯＬよりもＳＯＲＴユーティリティに依存している。

◎キー項目のデータ属性
整列キー項目のデータ属性について種類と最大サイズなどを説明する。
下記の図は、汎用コンピュータのＳＯＲＴユーティリティのデータ属性例である。

指定例
ＳＯＲＴ　ＦＩＥＬＤＳ＝（開始位置，サイズ，記号，順序，・・・・）
↑　↑A（昇順）、D（降順）

記号	サイズ（ﾊﾞｲﾄ）	属性の説明
ＣＨ	１～４０９２	ＥＢＣＤＩＣ文字列
ＺＤ	１～３２	ゾーン形式１０進数
ＰＤ	１～３２	パック形式１０進数
ＦＩ	１～２５６	固定小数点
ＢＩ	1ﾋﾞｯﾄ～4092ﾊﾞｲﾄ	２進数（符号なし）
ＦＬ	１～２５６	浮動小数点
ＡＣ	１～２５６	ＡＳＣＩＩ文字列
ＣＳＬ	２～２５６	ＥＢＣＤＩＣ数字　leading　separate　sign
ＣＳＴ	２～２５６	ＥＢＣＤＩＣ数字　trailing separate　sign
ＣＬＯ	１～２５６	ＥＢＣＤＩＣ数字　leading　overpunch　sign
ＣＴＯ	１～２５６	ＥＢＣＤＩＣ数字　trailing overpunch　sign
ＡＳＬ	２～２５６	ＡＳＣＩＩ　数字　leading　separate　sign
ＡＳＴ	２～２５６	ＡＳＣＩＩ　数字　trailing separate　sign

◎重複キーの扱い
キー項目の値が等しいレコードの意味と処理上の不都合について考えさせる。
また、対策として、第１キー項目、第２キー項目、・・・と指定できること、そして、
それぞれのキーに昇順、降順が指定できることも説明する。

・ＣＯＢＯＬ例
```
SORT  整列ﾌｧｲﾙ名  ON  ASCENDING  KEY  項目1  項目2
                      DESCENDING KEY  項目3
                      ASCENDING  KEY  項目4
```

・ＳＯＲＴユーティリティ例
```
SORT  FIELDS=(位置,ｻｲｽﾞ,CH,A,位置,ｻｲｽﾞ,ZD,A,位置,ｻｲｽﾞ,CH,D,位置,ｻｲｽﾞ,PD,A)
              項目１          項目２          項目３          項目４
```

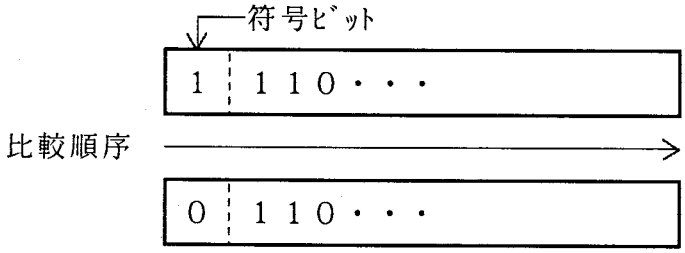
レコードの入力順序の指定は次の通り。

・ＣＯＢＯＬ例
```
SORT  整列ﾌｧｲﾙ  ON  ASCENDING  KEY  ･･･
                WITH  DUPLICATES  IN  ORDER
```

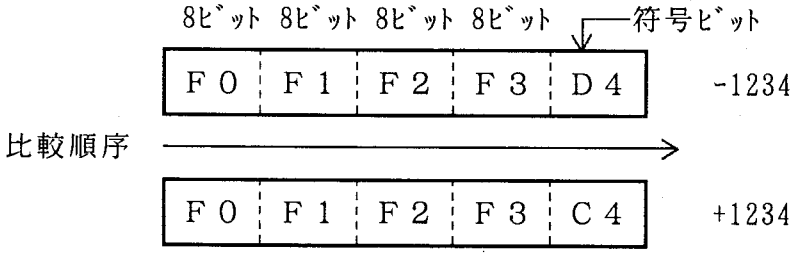
・ＳＯＲＴユーティリティ例
```
SORT=(位置,ｻｲｽﾞ,記号,順序,･･･),  EQUALS | NOEQUALS
```

◎キー項目の制限
ＪＩＳ－ＣＯＢＯＬでは規定していない。
汎用機のＣＯＢＯＬではキー項目は最大１２個、キー項目の合計サイズが４０９２バイトなどとなっている。

◎数字データの比較について
数字項目を英数字項目（文字）として比較した場合の問題点を説明しデータ表現の理解に努める。
数字データの符号について
２進数の符号について、英数字項目として比較すると負の値が大きくなる。



１０進数（USAGE　DISPLAY）で負の値が大きくなる。（EBCDIC表現）



◎符号系名の記述について
プログラムの大小順序を指定すると文字比較で用いられる。

```
OBJECT-COMPUTER.   計算機名
                   PROGRAM  COLLATING  SEQUENCE  IS [符号系名] .

SPECIAL-NAMES.
      ALPHABET [符号系名] IS  STANDARD-1  ←ＡＮＳＩ　Ｘ３．４
                              STANDARD-2  ←国際規格ＩＳＯ　６４６
                              NATIVE      ←固有文字集合
        :
      SORT  整列ﾌｧｲﾙ名  ON  ASCENDING  KEY  ﾃﾞｰﾀ名  ･･･
            COLLATING  SEQUENCE  IS  [符号系名]
```

◎ＳＯＲＴ機能

入力ファイルを整列処理して出力ファイルに書き出す場合は、プログラミングすることなく汎用ユーティリティを使用してできることを説明する。
（会社によっては、会社の標準化の一部としてＣＯＢＯＬのＳＯＲＴ文の使用を禁止しているところもある。）

【ＣＯＢＯＬの場合】

```
IDENTIFICATION  DIVISION.
ENVIRONMENT  DIVISION.
INPUT-OUTPUT  SECTION.
FILE-CONTROL.
    SELECT  入力ﾌｧｲﾙ  ASSIGN  TO  作成者語1.
    SELECT  整列ﾌｧｲﾙ  ASSIGN  TO  作成者語1.
    SELECT  出力ﾌｧｲﾙ  ASSIGN  TO  作成者語1.
DATA  DIVISION.
FILE  SECTION.
FD  入力ﾌｧｲﾙ.
01  入力ﾚｺｰﾄﾞ        PICTURE  X(100).
SD  整列ﾌｧｲﾙ.
01  整列ﾚｺｰﾄﾞ.
    03  施設名       PICTURE  X(15).
    03  電話番号     PICTURE  X(12).
    03  FILLER       PICTURE  X(03).
    03  開始年月日   PICTURE  X(10).
    03  FILLER       PICTURE  X(60).
FD  出力ﾌｧｲﾙ.
01  出力ﾚｺｰﾄﾞ        PICTURE  X(100).
PROCEDURE  DIVISION.
HAJIME.
    SORT  整列ﾌｧｲﾙ  ON  ASCENDING  KEY  開始年月日
                  USING   入力ﾌｧｲﾙ
                  GIVING  出力ﾌｧｲﾙ
    STOP  RUN.
```

【汎用ユーティリティの場合】

```
//ｼﾞｮﾌﾞ名  JOB  ･･･
//STEP      EXEC  PGM=SORT,REGION=4096K
//STEPLIB   DD  DSN=linklib,DISP=SHR
//SORTLIB   DD  DSN=sortlib,DISP=SHR
//SORTIN    DD  DSN=入力ﾃﾞｰﾀﾌｧｲﾙ,DISP=SHR
//SORTOUT   DD  DSN=出力ﾃﾞｰﾀﾌｧｲﾙ,DISP=SHR
//SORTWK01  DD  SPACE=(CYL,(10,10)),UNIT=WORK
//SORTWK02  DD  SPACE=(CYL,(10,10)),UNIT=WORK
//SORTWK03  DD  SPACE=(CYL,(10,10)),UNIT=WORK
//SYSOUT    DD  SYSOUT=A
//SYSIN     DD  *
    SORT  FIELDS=(31,10,CH,A)
/*
```

ＳＯＲＴユーティリティのワークファイル（例ではSORTWKnn）とＣＯＢＯＬのＳＤの整列ファイルの定義を混同する場合があるので注意を要する。

◎ＭＥＲＧＥ機能

ＳＯＲＴ機能同様、複数の入力ファイルを併合して出力ファイルに書き出す場合は、プログラミングすることなく汎用ユーティリティを使用してできることを説明する。

【ＣＯＢＯＬの場合】

```
IDENTIFICATION  DIVISION.
ENVIRONMENT  DIVISION.
INPUT-OUTPUT  SECTION.
FILE-CONTROL.
    SELECT  入力ﾌｧｲﾙ1 ASSIGN  TO  作成者語1.
    SELECT  入力ﾌｧｲﾙ2 ASSIGN  TO  作成者語1.
    SELECT  入力ﾌｧｲﾙ3 ASSIGN  TO  作成者語1.
    SELECT  併合ﾌｧｲﾙ  ASSIGN  TO  作成者語1.
    SELECT  出力ﾌｧｲﾙ  ASSIGN  TO  作成者語1.
DATA  DIVISION.
FILE  SECTION.
FD  入力ﾌｧｲﾙ1.
01  入力ﾚｺｰﾄﾞ1       PICTURE  X(100).
FD  入力ﾌｧｲﾙ2.
01  入力ﾚｺｰﾄﾞ2       PICTURE  X(100).
FD  入力ﾌｧｲﾙ3.
01  入力ﾚｺｰﾄﾞ3       PICTURE  X(100).
SD  併合ﾌｧｲﾙ.
01  併合ﾚｺｰﾄﾞ.
    03  施設名       PICTURE  X(15).
    03  電話番号     PICTURE  X(12).
    03  FILLER       PICTURE  X(03).
    03  開始年月日   PICTURE  X(10).
    03  FILLER       PICTURE  X(60).
FD  出力ﾌｧｲﾙ.
01  出力ﾚｺｰﾄﾞ        PICTURE  X(100).
PROCEDURE  DIVISION.
HAJIME.
    MERGE 併合ﾌｧｲﾙ  ON  ASCENDING  KEY  電話番号
          USING   入力ﾌｧｲﾙ1  入力ﾌｧｲﾙ2  入力ﾌｧｲﾙ3
          GIVING  出力ﾌｧｲﾙ
    STOP  RUN.
```

【汎用ユーティリティの場合】

```
//ｼﾞｮﾌﾞ名  JOB  ･･･
//STEP      EXEC  PGM=SORT,REGION=4096K
//STEPLIB   DD  DSN=linklib,DISP=SHR
//SORTLIB   DD  DSN=sortlib,DISP=SHR
//SORTIN01  DD  DSN=入力ﾃﾞｰﾀﾌｧｲﾙ1,DISP=SHR
//SORTIN02  DD  DSN=入力ﾃﾞｰﾀﾌｧｲﾙ2,DISP=SHR
//SORTIN03  DD  DSN=入力ﾃﾞｰﾀﾌｧｲﾙ3,DISP=SHR
//SORTOUT   DD  DSN=出力ﾃﾞｰﾀﾌｧｲﾙ,DISP=SHR
//SYSOUT    DD  SYSOUT=A
//SYSIN     DD  *
    MERGE FIELDS=(16,12,CH,A)
/*
```

ＭＥＲＧＥ機能はＳＯＲＴユーティリティの一部の機能として実現されている場合が多いので、これ例も呼び出すプログラムはＳＯＲＴとしている。

◎コード体系
文字データを並べかえる場合に、昇順、降順の意味とコード体系の関係を理解させる。

ENVIRONMENT DIVISIONのSPECIAL-NAMESで規定されている符号系名の「STANDARD-1」と「STANDARD-2」は次の通り。

STANDARD－1

列 / 行	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	C	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	'	<	L	＼	l	|
D	CR	GS	－	=	M	]	m	}
E	SO	RS	.	>	N	^	n	￣
F	SI	US	／	?	O	_	o	DEL

STANDARD－2

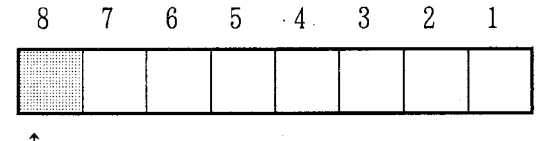
0	1	2	3	4	5	6	7
NUL	DLE	SP	0	@	P	`	p
SOH	DC1	!	1	A	Q	a	q
STX	DC2	"	2	B	R	b	r
ETX	DC3	#	3	C	S	C	s
EOT	DC4	ロ	4	D	T	d	t
ENQ	NAK	%	5	E	U	e	u
ACK	SYN	&	6	F	V	f	v
BEL	ETB	'	7	G	W	g	w
BS	CAN	(	8	H	X	h	x
HT	EM	)	9	I	Y	i	y
LF	SUB	*	:	J	Z	j	z
VT	ESC	+	;	K	[	k	{
FF	IS4	'	<	L	＼	l	|
CR	IS3	－	=	M	]	m	}
SO	IS2	.	>	N	^	n	￣
SI	IS1	／	?	O	_	o	DEL

ＥＢＣＤＩＣコード（Extended Binary Coded Decimal Interchange Code）
８ビットで１文字を表すコードである。
汎用機の世界では標準的なコード体系である。

列／行	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DEL	DS		SP	&	＿			ソ			{	}	＼	0
1	SOH	DC1	SOS		。	ｪ	／		ア	タ			A	J		1
2	STX	DC2	FS	SYN	「	ｫ			イ	チ	ヘ		B	K	S	2
3	ETX	TM			」	ｬ			ウ	ツ	ホ		C	L	T	3
4	PF	RES	BYP	PN	、	ｭ			エ	テ	マ		D	M	U	4
5	HT	NL	LF	RS	・	ｮ			オ	ト	ミ		E	N	V	5
6	LC	BS	ETB	UC	ヲ	ｯ			カ	ナ	ム		F	O	W	6
7	DEL	IL	ESC	EOT	ｧ				キ	ニ	メ		G	P	X	7
8	GE	CAN			ｨ	ー			ク	ヌ	モ		H	Q	Y	8
9	RLF	EM			ｩ				ケ	ネ	ヤ		I	R	Z	9
A	SMM	CC	SM		¢	!			コ	ノ	ユ	レ				
B	VT	CU1	CU1	CU3	.	¥		#				ロ				
C	FF	IFS		DC4	<	*	%	@	サ		ヨ	ワ				
D	CR	IGS	EMQ	NAK	(	)	－	｜	シ	ハ	ラ	ン				
E	SO	IRS	ACK		＋	:	>	＝	ス	ヒ	リ	゛				
F	SI	IUS	BEL	SUB	｜		?	”	セ	フ	ル	°				EO

ＡＳＣＩＩコード（American Standard Code for Information Interchange）

ＡＳＡ（アメリカ規格協会）よりＩＳＯ（国際標準化機構）に提出され、アメリカで最初に定められた標準コードである。ＩＳＯ６４６コード（７ビットコード）にパリティビットを付加したコードである。ＪＩＳコードの母体となっている。



↑
パリティビット
（奇数パリティ）
（偶数パリティ）

列／行	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DE		0	@	P	'	p								
1	SH	D1	!	1	A	Q	a	q								
2	SX	D2	"	2	B	R	b	r								
3	EX	D3	#	3	C	S	c	s								
4	ET	D4	$	4	D	T	d	t								
5	EQ	NK	%	5	E	U	e	u								
6	AK	SN	&	6	F	V	f	v								
7	BL	EB	'	7	G	W	g	w								
8	BS	CN	(	8	H	X	h	x								
9	HT	EM	)	9	I	Y	i	y								
A	LF	SB	*	:	J	Z	j	z								
B	HM	EC	+	;	K	[	k	{								
C	CL	→	,	<	L	＼	l	|								
D	CR	←	-	=	M	]	m	}								
E	SO	↑	.	>	N	^	n	~								
F	SI	↓	/	?	O	_	o									

ＪＩＳ　Ｘ０２０１　コード

７ビット、８ビット、の２種類があり、ＩＳＯの標準規格（ＩＳＯ　６４６）コードにカナコードを付け加えたもの。わが国の標準コードとなっている。

列 / 行	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DEL	SP	0	@	P	'	p				ー	タ	ミ		
1	SOH	DC1	!	1	A	Q	a	q			。	ア	チ	ム		
2	STX	DC2	"	2	B	R	b	r			「	イ	ツ	メ		
3	ETX	DC3	#	3	C	S	c	s			」	ウ	テ	モ		
4	EOT	DC4	$	4	D	T	d	t			、	エ	ト	ヤ		
5	END	NAK	%	5	E	U	e	u			・	オ	ナ	ユ		
6	ACK	SYN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7	BEL	ETB	、	7	G	W	g	w			ァ	キ	ヌ	ラ		
8	BS	CAN	(	8	H	X	h	x			ィ	ク	ネ	リ		
9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A	LF	SUB	*	:	J	Z	j	z			ェ	コ	ハ	レ		
B	VT	ESC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C	FF	FS	'	<	L	¥	l	|			ャ	シ	フ	ワ		
D	CR	GS	-	=	M	]	m	}			ュ	ス	ヘ	ン		
E	SO	RS	•	>	N	^	n	-			ョ	セ	ホ	゛		
F	SI	US	/	?	O	_	o	DEL			ッ	ソ	マ	゜		