第1章 プログラム作成の手順

1　プログラムの作成

コンピュータを利用して，いろいろな仕事を行う場合，仕事の手順をコンピュータに指示する必要がある。コンピュータに与える仕事の手順を**プログラム**といい，さまざまな**プログラム言語**で作成される。プログラムを作成する作業を**プログラミング**という。

プログラムは次の手順で作成する。

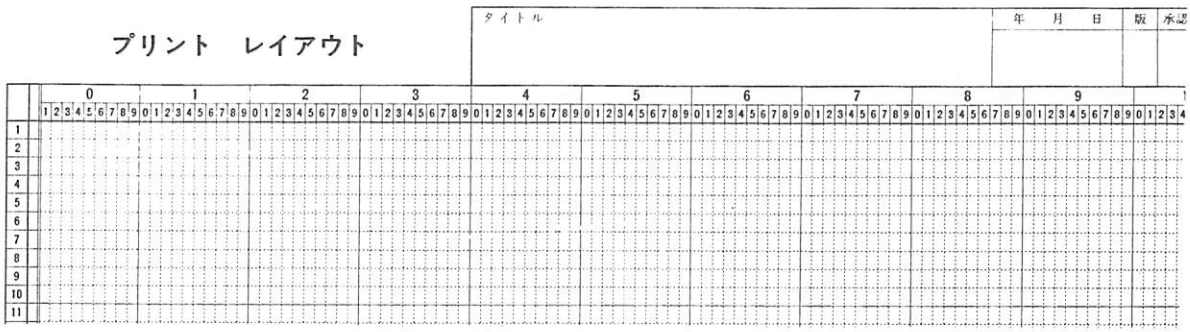
① 入力・出力の検討 ⇨ ② 流れ図の作成 ⇨ ③ コーディング ⇨ ④ 入力媒体への記録

入力・出力の検討

どのようなデータを入力し，どのような形式で処理結果を出力するかは，プログラム作成の最初に決定する。

通常，入力データは**データシート**に記述し，プリンタの出力形式は**スペーシングチャート**に，ディスプレイ画面の出力形式は**CRT*レイアウトシート**に記述する。

＊Cathode Ray Tubeの略で，ブラウン管のことである。

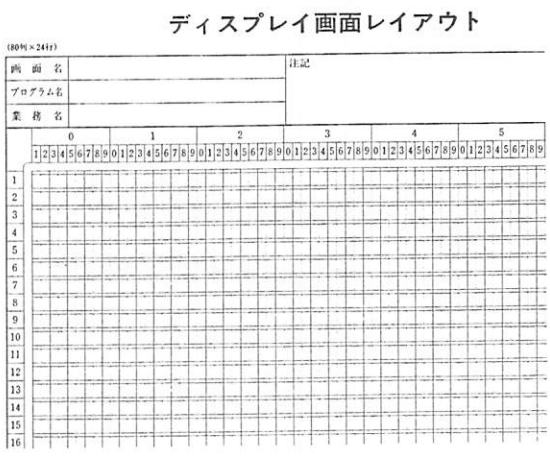


▲スペーシングチャート

流れ図の作成

処理手順の流れを図式化したものを，**流れ図**（**フローチャート**）といい，**テンプレート**と呼ばれる専用の定規でかく。

流れ図記号はJIS（日本工業規格）で決められている。



▲CRTレイアウトシート

コーディング

流れ図で示した処理手順にしたがってプログラムをかくことを**コーディング**という。

プログラム言語ごとに専用の**コーディングシート**（**プログラムシート**）が用意されている。

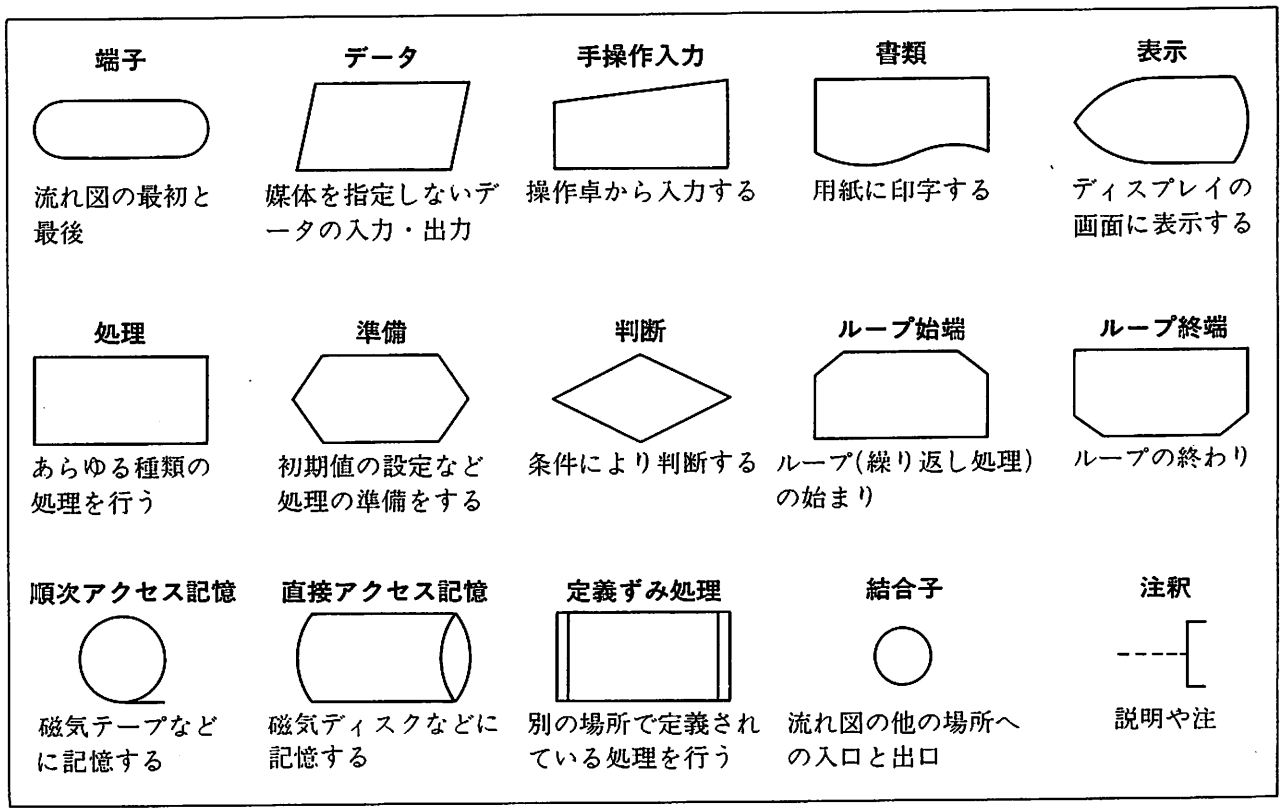
入出力媒体への記録

コーディングされたプログラムは，コンピュータが解読できる形にしなければならない。以前はパンチカードに変換されていたが，現在では，直接キーボードから磁気ディスクやフロッピーディスクに入力する場合が多い。

2　流れ図記号

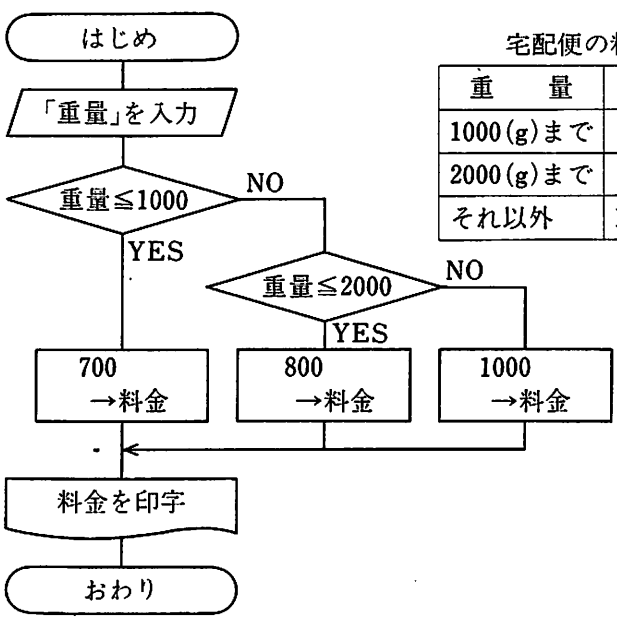
ある処理をプログラミングする場合，その処理手順をアルゴリズムといい，アルゴリズムを図式化したものが流れ図である。

JIS で決められている流れ図記号のうち，おもなものを次に示す。



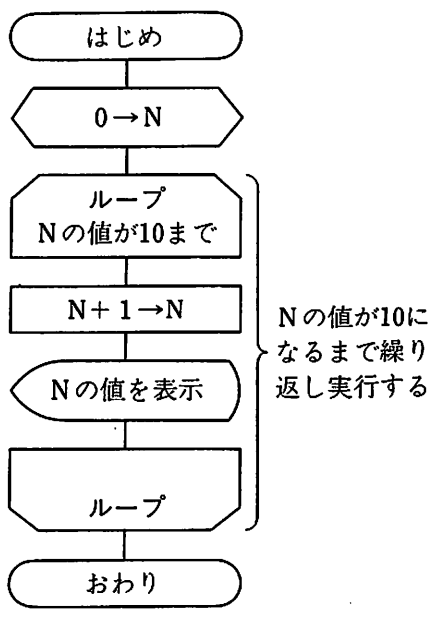
▲流れ図記号

例



宅配便の料金

重　量	料　金
1000(g)まで	700(円)
2000(g)まで	800(円)
それ以外	1000(円)



3　プログラムの実行

わたしたちが作成したプログラムは**原始プログラム***といい，そのままではコンピュータは解読できない。コンピュータが解読できるのは，数字の0と1の組み合わせで命令やデータを表現する**機械語**である。プログラムを機械語に変換することを**翻訳（コンパイル）****といい，翻訳されたプログラムを**目的プログラム*****という。

翻訳するさいに，原始プログラムの書き方に誤りがあると，**文法エラー**としてエラーメッセージが出力される。このエラーメッセージを見て，原始プログラムを修正する。

目的プログラムが正しく作られたかどうかを調べるには，テストデータを作成して，プログラムが目的どおりに動くかどうかを確認する。このことを**テストラン**といい，プログラムの処理手順や考え方が誤っている場合，**論理エラー**がみつかる。

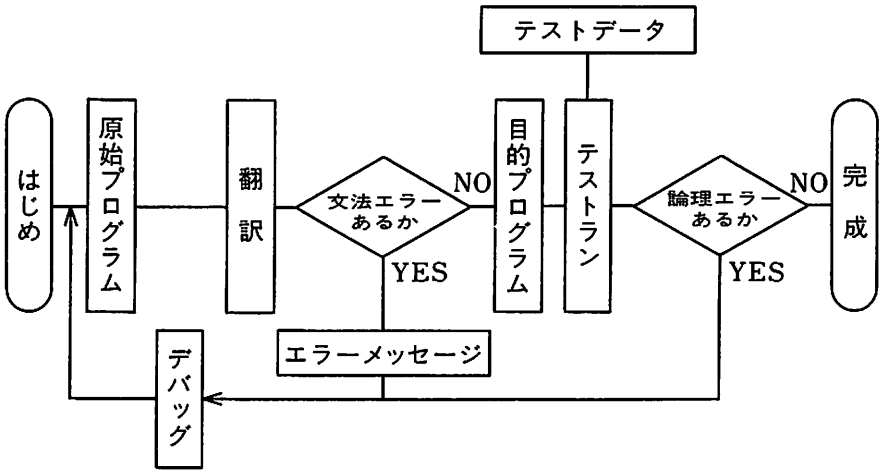
論理エラーの場所を探すには，簡単なプログラムであれば，**トレース**を行う。トレースとは，流れ図やプログラムの処理手順をひとつひとつたどり，各変数の値がどのように変化するかを調べることである。

文法エラーや論理エラーのことを**バグ**といい，エラーを修正し，正しいプログラムに直す作業のことを**デバッグ**という。

*ソースプログラムともいう。

**翻訳を実行するプログラムをコンパイラという。

***オブジェクトプログラムともいう。



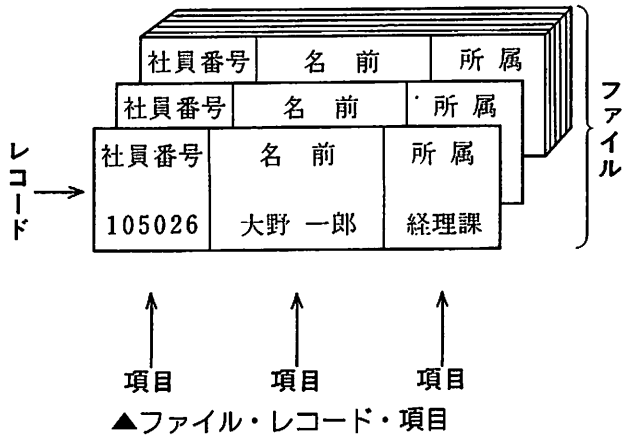
▲プログラムの作成と実行

4　データ

コンピュータで処理をする右図のようなデータの集まりがある場合，社員番号・名前・所属をまとめた1件分のデータを**レコード**という。

社員番号・名前・所属のようなレコードを構成する要素を**項目**という。

また，ひとまとまりの処理を実行するためのレコードの集まりを**ファイル**という。



▲ファイル・レコード・項目

●練習問題●

1　次の記述中の（　　）にあてはまる語句を解答群から選びなさい。

わたしたちが最初に書いたプログラムは，（　a　）といい，コンピュータは直接解読することができない。コンピュータが解読できるのは，数字の１と０の組み合わせからなる（　b　）である。

（　a　）を（　b　）に変換することを（　c　）といい，（　c　）されたプログラムを（　d　）という。通常，（　c　）と同時にプログラムの文法をチェックする（　e　）が出力される。

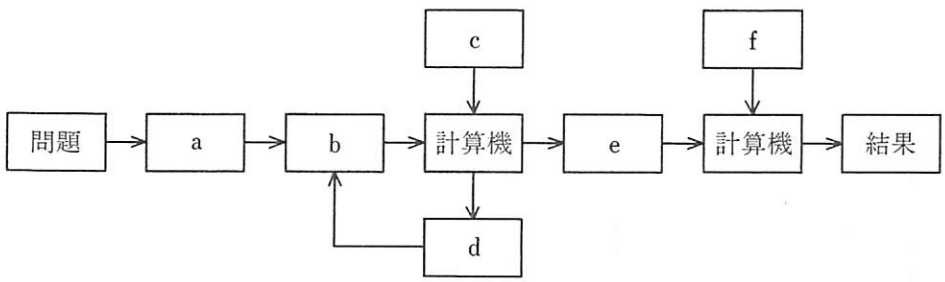
(解答群)

ア．自然語	イ．目的プログラム	ウ．原始プログラム
エ．翻訳	オ．エラーメッセージ	カ．デバッグ
キ．機械語	ク．実行	

2　次のA群に関係の深いものをB群の中から選びなさい。

A群	B群
(1)　文法エラー	ア．正しい結果が得られるかどうか調べるために，テストデータを用意して試してみること。
(2)　論理エラー	
(3)　レコード	イ．データとして意味を持つ最小の単位。
(4)　テンプレート	ウ．プログラムの書き方によるまちがい。
(5)　テストラン	エ．プログラムの考え方によるまちがい。
(6)　プログラム	オ．コンピュータに与える仕事の手順。
(7)　項目	カ．流れ図を書くための専用の定規。
	キ．データ処理１件分の単位。

3　次の図は，プログラムを作成するときの一般的な手順を示したものである。（　　）に入れるべき適当な語句を解答群の中から選びなさい。



(解答群)

ア．目的プログラム	イ．原始プログラム	ウ．コンパイラ
エ．データ	オ．流れ図	カ．エラーメッセージ

第2章 基本的なプログラム

1 COBOLとは

COBOLは，COmmon Business Oriented Languageの略で，事務処理用に開発された言語である。言語表現が日常会話に近い形で表現でき，見やすく，プログラムの修正が簡単なために，事務処理用として用いられているプログラム言語は，COBOLがほとんどである。

COBOLは，1959年にアメリカ国防総省によってCODASYL（COnference on DAta SYstems Languages：データシステムズ言語協議会）が組織され，コンピュータメーカやユーザなどの代表たちと一緒に，事務処理用の統一言語を提案したことに始まる。そしてCODASYLは，その後次々と機能を追加して現在に至っている。

COBOL－60	1960年　CODASYLの最初の提案
COBOL－61	1963年
拡張COBOL－61	1963年　整列，報告書作成機能
COBOL 1965	磁気ディスクファイル操作機能，表操作命令
COBOL 1968	プログラム間連絡機能
COBOL 1969	通信機能
COBOL 1970	デバッグ機能
COBOL 1973	ファイルの各種機能の改訂
COBOL 1976	データベース機能
COBOL 1978	構造化プログラミングのための機能
COBOL 1981	
COBOL 1985	構造化プログラミングの機能を大幅に強化

▲COBOLの歴史

日本においては，CODASYLの提案を受けて，1972年に最初のJIS（日本工業規格）COBOLが制定された。その後CODASYLの改正に沿って，1980年と1988年に改正が行われた。

最新のCOBOLは，CODASYLが改正されたのをうけてJISも改正された。最新のCOBOLは，プログラムの表記をわかりやすくする，**構造化プログラミング**の考え方を大幅に導入し，その機能が強化された。

このテキストは最新のCOBOLの水準で説明してある。

2　COBOLで使用できる文字と語

COBOLで使用できる文字は次のとおりである。*

数字	0	1	2	3	4	……	9
英大文字	A	B	C	D	E	……	Z
英小文字	a	b	c	d	e	……	z
空白							
特殊記号	+	−	*	/	=	¥	,	;
	.	"	(	)	>	<	:	

＊現在ではカタカナや漢字が使える計算機が多くなっている。

COBOLで用いられる語には予約語と利用者語がある。**

＊＊厳密にいうと，p.13でふれる「装置名」のようなシステム名もある。

予約語

COBOLには，READ（読め），WRITE（書け）などのように，あらかじめその用途が決まっている語があり，これを**予約語**という。

予約語の一覧表を巻末に示した。

利用者語

ファイル名やデータ名のように，プログラムを作成する人が自由に決める語を**利用者語**という。

利用者語を決めるさいには，次の点に注意する。

① 英字（A～Z），数字（0～9），ハイフン（−）の組み合わせで，30字以内であること。***

② ハイフンを最初と最後に使ってはいけない。

③ 予約語を使ってはいけない。

④ 語の途中に空白があってはいけない。

＊＊＊ 英小文字を用いてもよいが，英大文字と同じとみなされる。
利用者語の中には数字だけでよいものもあるが，通常は少なくとも1文字の英字を含まなければならない。

例	（正しい例）	（誤りの例）	
	M−KENCODE	M　KENCODE	（途中に空白がある）
	HIZUKE	DATE	（予約語である）
	BANGO−01	BANGO：01	（：は使えない）

3　コーディング

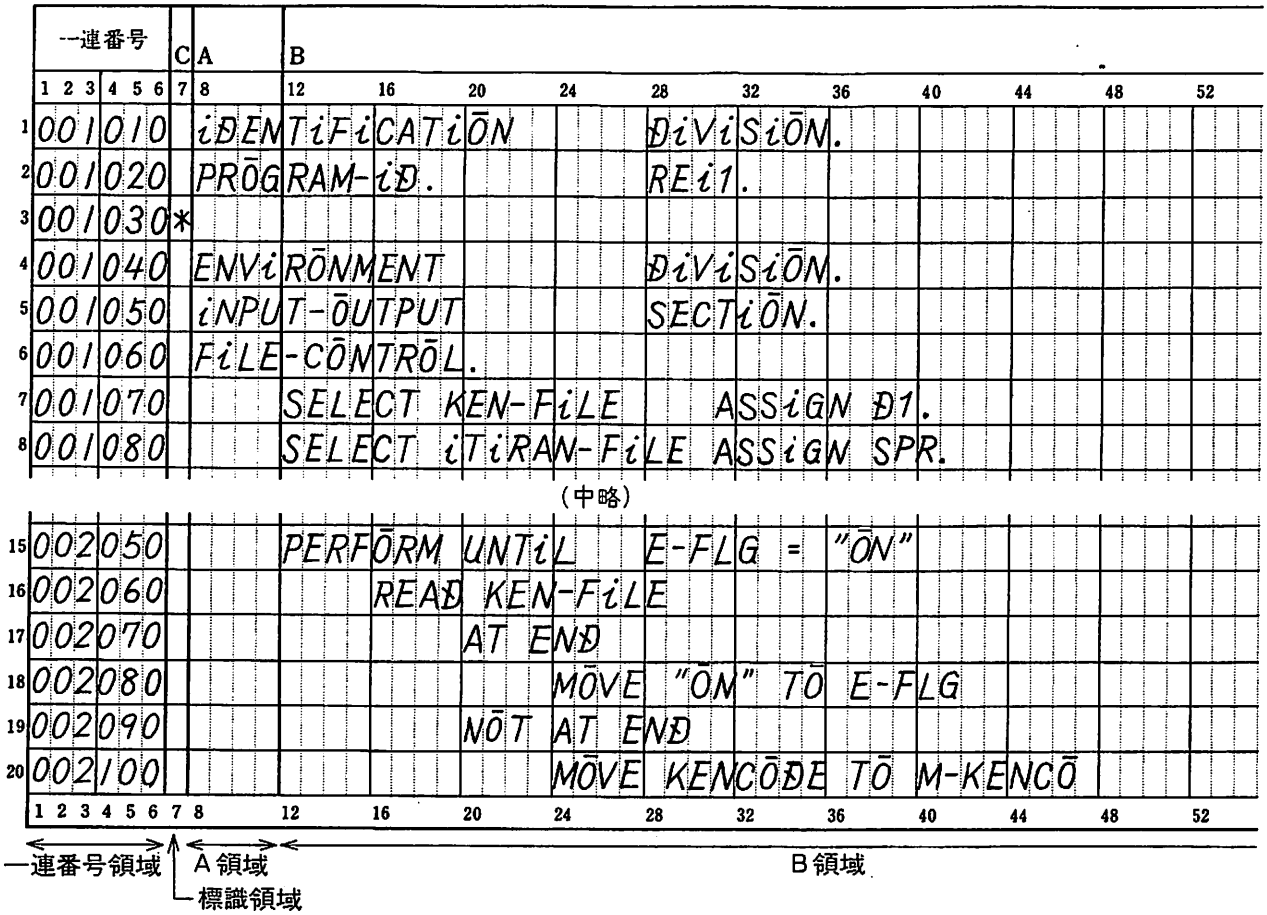
コーディングは手書きされるために，よく似た文字を区別するために特別な書き方をする場合がある。

英字のO→Ō または Ø（数字の0と区別する）
英字のD→Đ　（　〃　）
英字のI→$\dot{I}$ または Ī（数字の1と区別する）
英字のZ→Ƶ　（数字の2と区別する）
英字のU→Ⱦ　（英字のVやWと区別する）
英字のC→Ć　（特殊文字の（と区別する）

また，空白を明確にする場合は，△または␣を使う場合もある。

コーディングシートの書き方

COBOL用のコーディングシートは下記のようであり，それぞれの欄には書く内容が決まっている。



1～6桁（一連番号領域）　プログラムの行番号を書く桁である。前半の3桁をコーディングシートのページ番号，後半の3桁を行番号に使用する。行番号はあとからの挿入に便利なように10刻みに付けられる場合が多い。

7桁（標識領域）　この欄に「＊」「／」「－」を用いると，次のような役割を果たす。

＊　プログラムの実行には影響を与えない注記行となる。プログラムの説明書きやプログラムを見やすくするために用いる。

／　注記行であると同時に，翻訳時のプログラムリストをページをかえて出力する。

－　1つ前の行からプログラムの内容が継続していることを示す。

8～72桁（A領域・B領域）　プログラムを記述する所で，8から11桁をA領域，12から72桁をB領域と呼ぶ。部・節の見出しや段落名などはA領域から書き，手続きの部の命令などはB領域から書く。*

73～80桁（識別領域）　プログラムを識別するための欄であるが，現在では省略する場合が多い。

空白行　1行すべて空白の行をいい，プログラムをみやすくするために用いられる。空白行はどこにあってもよい。

＊部，節，段落などについてはp.12参照。

4　COBOL のプログラム

例題１　都道府県別一覧表の出力

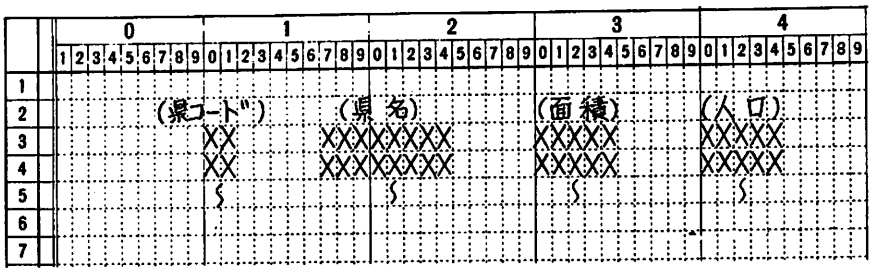
入力形式のようなデータ（県ファイル）を読んで，出力形式にしたがって都道府県別の面積・人口の一覧表を作成しなさい。

▼入力形式

県コード	県　　名	面　　積	人　　口
０１	ﾎｯｶｲﾄﾞｳ	８３５２０	０５６４６
０２	ｱｵﾓﾘ	０９６１９	０１５２９
０３	ｲﾜﾃ	１５２７７	０１４３３
０４	ﾐﾔｷﾞ	０７２９２	０２２１０
０５	ｱｷﾀ	１１６１３	０１２４５
⌇		⌇	⌇

(注) 面積の単位は㎢，人口の単位は千人である。

▼出力形式



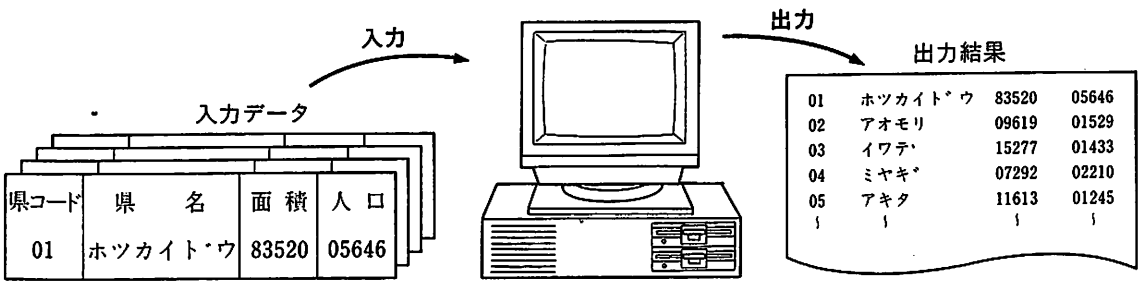
▼出力結果

01	ﾎｯｶｲﾄﾞｳ	83520	05646
02	ｱｵﾓﾘ	09619	01529
03	ｲﾜﾃ	15277	01433
04	ﾐﾔｷﾞ	07292	02210
05	ｱｷﾀ	11613	01245
⌇	⌇	⌇	⌇

●問題の分析と考え方●

この例題では，県ファイルの１件分のデータ（県レコード）を読み，その内容を印字する処理を県ファイルのデータが終了するまで繰り返す。

この処理を実行するプログラムは次ページのようになるが，最初のプログラムであるので，ひとつひとつ段階を追って説明しよう。



●プログラム●

```
 IDENTIFICATION       DIVISION.
 PROGRAM-ID.          REI0201.
*
 ENVIRONMENT          DIVISION.
 INPUT-OUTPUT         SECTION.
 FILE-CONTROL.
     SELECT KEN-FILE      ASSIGN D1.
     SELECT ITIRAN-FILE   ASSIGN SPR.
*
 DATA                 DIVISION.
 FILE                 SECTION.
 FD   KEN-FILE.
 01   KEN-REC.
      05 KENCODE       PICTURE 9(02).
      05 KENMEI        PICTURE X(08).
      05 MENSEKI       PICTURE 9(05).
      05 JINKO         PICTURE 9(05).
 FD   ITIRAN-FILE.
 01   ITIRAN-REC      PICTURE X(132).
 WORKING-STORAGE SECTION.
 01   E-FLG           PICTURE X(03).
 01   MEISAI-GYO.
      05 FILLER        PICTURE X(09)   VALUE  SPACE.
      05 M-KENCODE     PICTURE 9(02).
      05 FILLER        PICTURE X(05)   VALUE  SPACE.
      05 M-KENMEI      PICTURE X(08).
      05 FILLER        PICTURE X(05)   VALUE  SPACE.
      05 M-MENSEKI     PICTURE 9(05).
      05 FILLER        PICTURE X(05)   VALUE  SPACE.
      05 M-JINKO       PICTURE 9(05).
*
 PROCEDURE     DIVISION.
 SYORI.
      OPEN      INPUT KEN-FILE       OUTPUT ITIRAN-FILE
      MOVE      "OFF"          TO    E-FLG
      PERFORM       UNTIL  E-FLG  =  "ON"
          READ KEN-FILE
               AT  END
                   MOVE  "ON"  TO  E-FLG
               NOT AT END
                   MOVE   KENCODE      TO    M-KENCODE
                   MOVE   KENMEI       TO    M-KENMEI
                   MOVE   MENSEKI      TO    M-MENSEKI
                   MOVE   JINKO        TO    M-JINKO
                   MOVE   MEISAI-GYO   TO    ITIRAN-REC
                   WRITE  ITIRAN-REC   AFTER 1
          END-READ
      END-PERFORM
      CLOSE     KEN-FILE    ITIRAN-FILE
      STOP      RUN.
```

見出し部
- プログラム名段落

環境部
- 入出力節
  - ファイル管理段落

データ部
- ファイル節
- 作業場所節

手続き部
- SHORI段落

㊟ SHORI段落で，"ON␣"（␣は空白）としなければならない機種もある。

COBOLの構造

COBOLは次に示す4つの部（DIVISION）から構成されている。

```
IDENTIFICATION DIVISION.

ENVIRONMENT    DIVISION.

DATA           DIVISION.

PROCEDURE      DIVISION.
```

見出し部
プログラム名を記述する部分

環境部
使用するファイル名や装置名を記述する部分

データ部
プログラムで処理するデータの性質を記述する部分

手続き部
処理の手順を記述する部分

見出し部・環境部

```
 IDENTIFICATION       DIVISION.
 PROGRAM-ID.          REI0201.
*
 ENVIRONMENT          DIVISION.
 INPUT-OUTPUT         SECTION.
 FILE-CONTROL.
     SELECT KEN-FILE      ASSIGN D1.
     SELECT ITIRAN-FILE   ASSIGN SPR.
```

「REI0201」というプログラム名を定義している。

入出力節の始まりを示す。

ファイル管理段落の始まりを示す。

「KEN-FILE」という入力ファイルを定義している。

「ITIRAN-FILE」という出力ファイルを定義している。

D1やSPRをシステム名といい，使用するコンピュータによって決められている。この名前が，プログラムを実行するときに，実際のファイル名と対応づけられる。また，プログラム名，ファイル名は利用者語であるので，利用者語の規則にしたがって，作成者が任意に決める。

IDENTIFICATION DIVISION

```
IDENTIFICATION   DIVISION.
PROGRAM-ID.      プログラム名.
```

ENVIRONMENT DIVISION

```
ENVIRONMENT   DIVISION.
INPUT-OUTPUT   SECTION.
FILE-CONTROL.
      SELECT   入力ファイル名   ASSIGN TO 入力装置名.
      SELECT   出力ファイル名   ASSIGN TO 出力装置名.
```

注　下線のある語は必須語といい，必ず記述する。TOのように，下線のない語は補助語といい，省略できる。

データ部

データ部では，プログラムで処理するさまざまなデータの性質を記述するが，大きく分けると「入力」「出力」「作業」で構成される。

```
DATA    DIVISION.
FILE    SECTION.                          ファイル節の始まりを示す
FD      入力ファイル名.                   入力ファイルの記述
01      入力レコード名.
        02  項目の記述
        .....
FD      出力ファイル名.                   出力ファイルの記述
01    出力レコードの記述
WORKING-STORAGE    SECTION.               作業場所節の始まりを示す
01    レコード名.                         作業領域の内容の記述
      02  項目の記述
.....
```

ファイル節　　データ部　　作業場所節

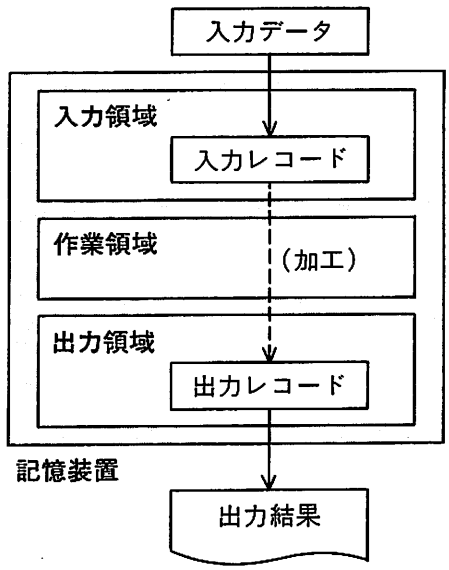
ファイル節

ファイル節では，入力ファイル・出力ファイルの定義をする。

ファイル名は，環境部で定義した入力用・出力用のファイル名を書く。

FDはFile Description（ファイルの記述）の略である。

入力レコードは，次ページに示したように入力形式をみて，入力項目の性質や桁数を書く。この書き方は，p.16で詳しく説明する。



作業場所節

ふつう入力したデータをそのまま出力することはなく，出力結果の形式を整えたり，計算した結果を出力したりする。

このため，出力するときの1行分にあたる明細行の形式を記述したり，計算の結果を一時的に記憶するためのデータ項目の定義をしたりする必要がある。

これらは，作業場所節で記述する。

▼入力形式

県コード	県　名	面 積	人 口
01	ホツカイト゛ウ	83520	05646
02	アオモリ	09619	01529
03	イワテ	15277	01433
≀	≀	≀	≀

入力レコード名 → KEN-REC

KENCODE	KENMEI	MENSEKI	JINKO
数字で2桁	文字で8桁	数字で5桁	数字で5桁

← 項目名
← 項目の性質と桁数

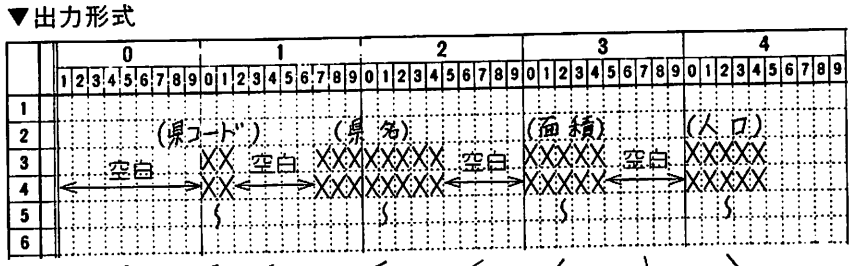
入力ファイルの記述

```
DATA DIVISION.
FILE SECTION.
FD   KEN-FILE.                          ―― 入力ファイル名
01   KEN-REC.                           ―― 入力レコード名
     05 KENCODE    PICTURE 9(02).
     05 KENMEI     PICTURE X(08).
     05 MENSEKI    PICTURE 9(05).
     05 JINKO      PICTURE 9(05).
FD   ITIRAN-FILE.                       ―― 出力ファイル名
01   ITIRAN-REC    PICTURE X(132).     ―― 出力レコード名とその性質と桁数。
                                            通常，プリンタの最大桁数を定義しておく。
```

出力ファイルの記述

```
WORKING-STORAGE SECTION.
01   E-FLG          PICTURE X(03).     ―― 終了フラグ（p.22参照）の定義。
```

▼出力形式



明細行のレコード名 → MEISAI-GYO

空白	M-KENCODE	空白	M-KENMEI	空白	M-MENSEKI	空白	M-JINKO
9桁	数字で2桁	5桁	文字で8桁	5桁	数字で5桁	5桁	数字で5桁

明細行の記述

```
01   MEISAI-GYO.
     05 FILLER       PICTURE X(09)   VALUE  SPACE.
     05 M-KENCODE    PICTURE 9(02).
     05 FILLER       PICTURE X(05)   VALUE  SPACE.
     05 M-KENMEI     PICTURE X(08).
     05 FILLER       PICTURE X(05)   VALUE  SPACE.
     05 M-MENSEKI    PICTURE 9(05).
     05 FILLER       PICTURE X(05)   VALUE  SPACE.
     05 M-JINKO      PICTURE 9(05).
```

レベル番号

```
01    KEN-REC.
      05  KENCODE    PICTURE 9 (02).
```

において，０１，０５などの数字を**レベル番号**といい，レコードや項目間の階層関係を表す。すなわち，レベル番号の小さいものが，レベル番号の大きいものを含んでいることを示している。

下位に所属する項目をもたない項目を**基本項目**といい，基本項目の集まりを**集団項目**という。

レベル番号は，次の規則にしたがってつける。

① ０１から４９まで使用できる。

② レコードのレベル番号は，必ず０１とする。また，階層構造をもたない独立した項目も０１をつける。

③ 集団項目が下位の基本項目を含むようにレベル番号をつける。
（０１，０２，０３と連続してつける必要はない）

④ 同じ階層の項目には，同じレベル番号をつける。

例 下記の２つのレベル番号の使い方は同じ意味である。

従業員レコード						
所属		社員番号	名前	生年月日		
部	課			年	月	日

```
01  JUGYOUIN-REC.          01  JUGYOUIN-REC.          集団項目
    02  SYOZOKU.                  10  SYOZOKU.              集団項目
        03  BU  ……                    20  BU  ……            基本項目
        03  KA  ……                    20  KA  ……            基本項目
    02  SYAINBANGO  ……            10  SYAINBANGO  ……       基本項目
    02  NAMAE       ……            10  NAMAE       ……       基本項目
    02  SEINENGAPPI.               10  SEINENGAPPI.          集団項目
        03  YY  ……                    20  YY  ……            基本項目
        03  MM  ……                    20  MM  ……            基本項目
        03  DD  ……                    20  DD  ……            基本項目
```

レコード全体につけた名前をレコード名，項目につけた名前を項目名，この２つをあわせて**データ名**という。なお，レベル番号０１をもつ項目はＡ領域から書き，それより下位の項目はＢ領域に書く。

PICTURE句

PICTURE句に「9」「X」などを用いて，項目の性質と桁数を指定する。PICTUREは PIC と省略できる。

PICTURE句は，基本項目に対してだけ指定できる。

データが数字の項目を**数字項目**といい，

```
BANGO    PICTURE 9 (02)
```

のように，項目の性質を「9」で表し，桁数を（ ）内に書く。

データが文字*の項目を**英数字項目**といい，

```
NAMAE    PICTURE  X (10)
```

のように，項目の性質を「X」で表し，桁数を（ ）内に書く。

桁数は，9やXを桁数分並べてもよい。

例 ① 9(01)，9(1)，9 は同じ。X(01)，X(1)，X は同じ。

② X(05)，X(5)，XXXXX は同じ。

*使用するコンピュータで使えるすべての文字のことで，数字や特殊記号でもよい。

FILLER句

```
05  FILLER     PICTURE  X (09)  VALUE  SPACE.
```

のように，FILLERと定義した項目を**無名項目**といい，項目名をつける必要がない場合に用いる。基本項目に対してだけ指定できる。

FILLERは省略できるので，次のように書いてもよい。

```
05                PICTURE  X (09)  VALUE  SPACE.
```

VALUE句

作業場所節においては，項目にあらかじめ特定の内容を記憶させておくことができる。これを**初期値**といい，VALUE句を用いて定義する。

初期値には**定数**と**表意定数**がある。

定数　数値項目の場合は，そのまま数字を記述する。**数値定数**という。最大は18桁である。

例　PIC　9(02)　VALUE　10　　数値の10を初期値とする

英数字項目の場合は，2つの"(引用符)で囲んで記述する。**文字定数**という。最大は256桁である。

例　PIC　X(02)　VALUE　"AA"　　文字「AA」を初期値とする

表意定数　特別の数値または文字列をあらわす。単数形も複数形も同じ意味である。

SPACE (SPACES)	1桁または何桁かの空白
ZERO (ZEROS, ZEROES)	1桁または何桁かの数値の0または文字の0
QUOTE (QUOTES)	1桁または何桁かの引用符「"」
ALL　文字定数	同じ文字列の何桁かの繰り返し
HIGH-VALUE	計算機の文字の大小順序で最大のもの
LOW-VALUE	計算機の文字の大小順序で最小のもの

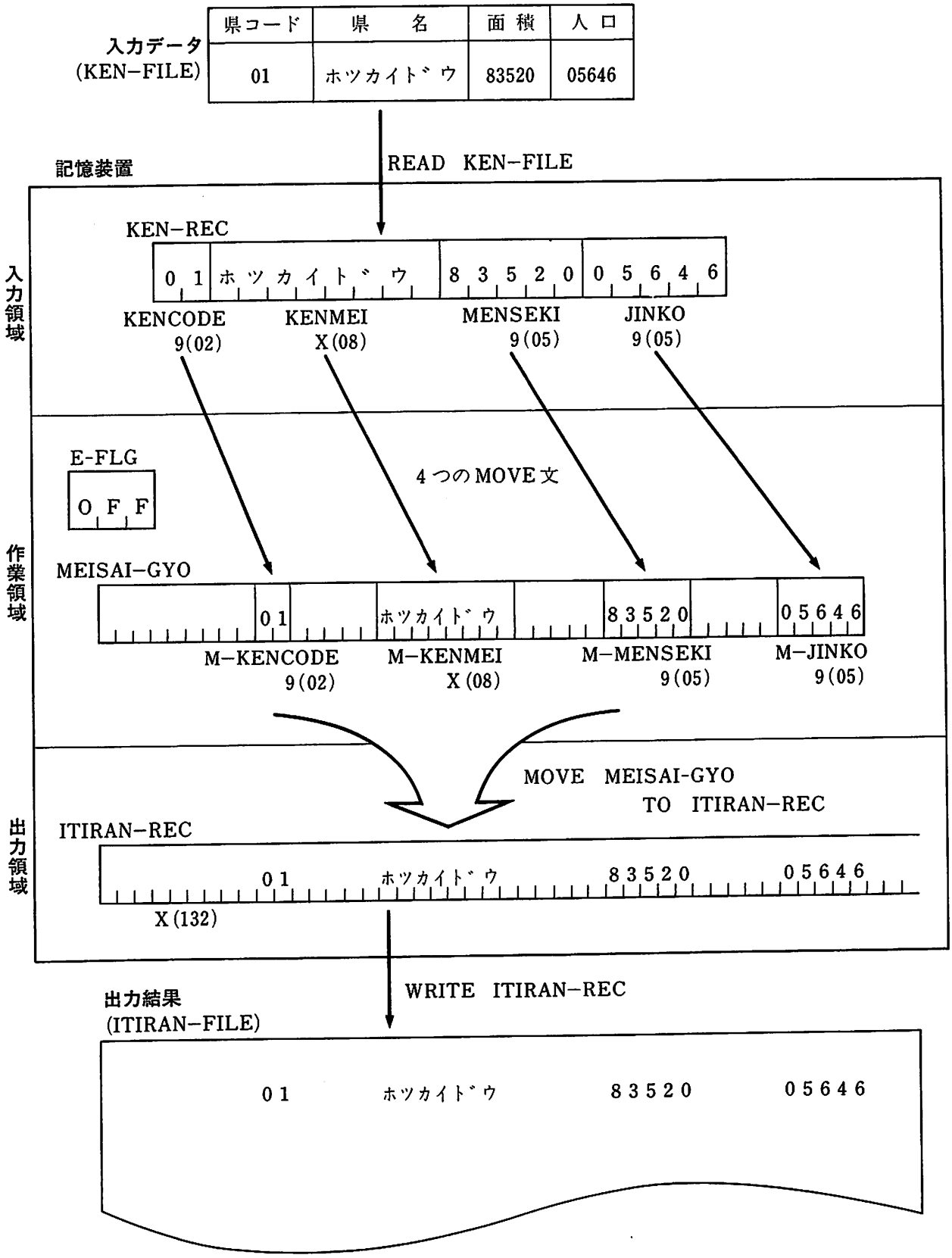
例 PIC X(05) VALUE SPACE	5桁の英数字項目にスペースを入れる**
PIC 9(03) VALUE ZEROS	3桁の数字項目に0を入れる***
PIC X(02) VALUE QUOTES	2桁の英数字項目に2桁の引用符をいれる
PIC X(09) VALUE ALL "Z"	9桁の英数字項目すべてにZという文字をいれる
PIC X(01) VALUE ZERO	1桁の英数字項目に文字の0をいれる

** このことをスペースクリアという。

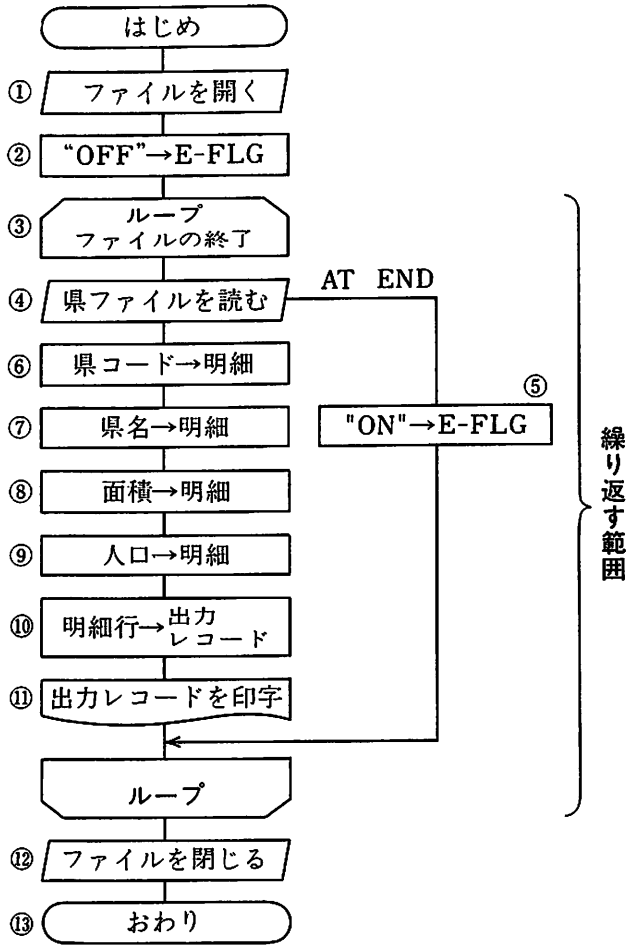
*** このことをゼロクリアという。

◉プログラムマップ◉

問題の流れをつかむために作成される記憶領域図をプログラムマップという。
例題１のマップを作成すると下記のようになる。



●流れ図と処理手順●



〔処理手順〕

① ファイルを開く。

② 終了フラグ（E−FLG）に"OFF"を転記

③ 終了フラグが"ON"になるまで④〜⑪を繰り返す。

④ 県ファイルを読む。

[データが終了したら]

⑤ 終了フラグに"ON"を転記。

[データが終了しない場合は]

⑥ 県コードを明細（明細行）に転記。

⑦ 県名を明細に転記。

⑧ 面積を明細に転記。

⑨ 人口を明細に転記。

⑩ 明細行を出力レコードに転記。

⑪ 出力レコードを印字する。

⑫ ファイルを閉じる。

⑬ 終了

（流れ図とプログラム中の番号①〜⑬は処理手順に対応したものである）

●手続き部●

```
PROCEDURE     DIVISION.
SYORI.
    OPEN      INPUT KEN-FILE       OUTPUT ITIRAN-FILE      ……①
    MOVE      "OFF"          TO    E-FLG                   ……②
    PERFORM       UNTIL  E-FLG  =  "ON"                    ……③
        READ KEN-FILE                                      ……④
             AT  END
                 MOVE  "ON"  TO  E-FLG                     ……⑤
             NOT AT END
                 MOVE   KENCODE      TO    M-KENCODE       ……⑥
                 MOVE   KENMEI       TO    M-KENMEI        ……⑦
                 MOVE   MENSEKI      TO    M-MENSEKI       ……⑧
                 MOVE   JINKO        TO    M-JINKO         ……⑨
                 MOVE   MEISAI-GYO   TO    ITIRAN-REC      ……⑩
                 WRITE  ITIRAN-REC   AFTER 1               ……⑪
        END-READ
    END-PERFORM
    CLOSE     KEN-FILE    ITIRAN-FILE                      ……⑫
    STOP      RUN.                                         ……⑬
```

繰り返す範囲

手続き名

前ページのプログラムの冒頭にある「SYORI」は段落名であるが，これをふつう**手続き名**という。*

このプログラムは簡単な処理なので手続き名は1つであるが，複雑なプログラムになると，ひとまとまりの処理単位ごとに手続き名をつけたほうがよい。

手続き名は，A領域から書き始め，必ずピリオドを打つ。

OPEN，MOVEなどの文**は，B領域に書く。手続き名の示す範囲，すなわち段落の最後（このプログラムではSTOP　RUN）には，必ずピリオドを打つ。

＊厳密には，段落名と節名をあわせて手続き名という。

＊＊COBOLの手続き部における一固まりの内容を文といい，文の説明をするものを句という。文は命令という場合もある。

OPEN文

OPEN文

OPEN　INPUT　入力ファイル名　OUTPUT　出力ファイル名

プログラムで使用する，入力ファイル，出力ファイルを開き，処理の準備をする文。入力ファイル名，出力ファイル名は，環境部で定義したファイル名を記述する。流れ図記号はデータを用いる。

▲データ

MOVE文

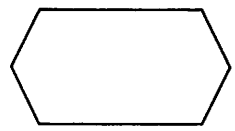
MOVE文

MOVE　{データ名1 / 定数}　TO　データ名2

注　{　}で囲まれた内容は，どれか1つを選んで記述することを示す。

左側のデータ名または定数を送出し側，右側のデータ名を受取り側といい，送出し側の内容を受取り側に移す文である。MOVE文の動作のことを転記という。流れ図記号は処理を用いる。ただし，事前に初期値を与える場合は準備を用いる場合もある。

▲処理



▲準備

```
          MOVE  A  TO  B
転記前    A 123        B 456
                ⇨          Aの内容をBに移す。
転記後    A 123        B 123   転記後は，Bの内容は変化するが
          送出し側     受取り側          Aの内容は変わらない。
```

▲転記の規則

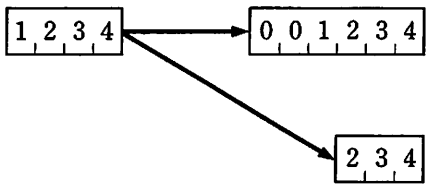
受取り側を複数指定した場合は，送出し側の内容がすべての受取り側に転記される。

例

```
MOVE   A   TO   B          Aの内容をBに転記する。
MOVE   A   TO   B  C  D    Aの内容をB，C，Dに転記する。
```

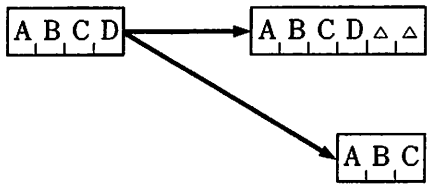
通常は送出し側，受取り側とも同じ桁数が取られるが，桁数が違う場合は次の規則にしたがう。

- 数字項目どうしの転記……**右側から順番に転記される。数字転記**という。



受取り側が大きい場合
　余った桁に0がはいる。
受取り側が小さい場合
　転記しきれない桁が切り捨てられる。

- 英数字項目どうしの転記……**左側から順番に転記される。文字転記**という。



受取り側が大きい場合
　余った桁に空白がはいる。
受取り側が小さい場合
　転記しきれない桁が切り捨てられる。

PERFORM文

PERFORM文

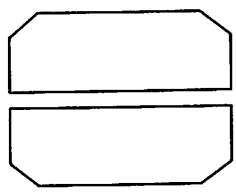
```
PERFORM      UNTIL    条件
          文
          ..............
END-PERFORM
```

PERFORMからEND−PERFORMまでの間を，条件が成立するまで繰り返す文。例題ではE−FLGが「ON」になるまで繰り返される。E−FLGについては終了フラグの説明(p.22)を参照すること。

繰り返し同じ処理を実行することを**ループ**という。

PERFORMとEND−PERFORMに対応して，流れ図記号は**ループ端**が用いられる。

「END-」からはじまる語を**明示範囲符**という。



▲ループ端

READ文

READ 文(1)

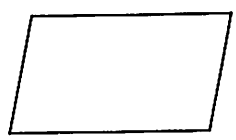
```
READ    入力ファイル名
        AT  END          文1
                         ..............

        NOT  AT  END     文2
                         ..............

END-READ
```

ファイルから1レコード分を読み取り，その内容を記憶装置の入力領域に記憶する文。そのさい，入力レコードが終了している場合は文1以下の内容を実行し，終了していない場合は文2以下を実行する。

流れ図記号は，**データ**を用いる。



▲データ

WRITE文

WRITE文(1)

```
WRITE  出力レコード名  { AFTER  }  数
                     { BEFORE }
```

出力領域に記憶されているデータを連続用紙などに出力する文。数には改行する行数を書く。AFTER句は印字の前に改行し，BEFORE句は印字後改行する。

流れ図記号は，データを用いてもよいが，通常はプリンタに出力されるので，書類を用いる場合が多い。

▲書類

```
WRITE ～
      AFTER 2  XXXXXX ←実行後の位置
```

改行後印字

```
WRITE ～
      BEFORE 2  XXXXXX ←実行後の位置
```

印字後改行

▲AFTERとBEFOREの違い

CLOSE文

CLOSE文

```
CLOSE  入力ファイル名  出力ファイル名
```

使用したファイルを閉じる文。入力ファイル名，出力ファイル名には，OPEN文で用いたファイル名を必ず指定する。流れ図記号はデータを用いる。

STOP文

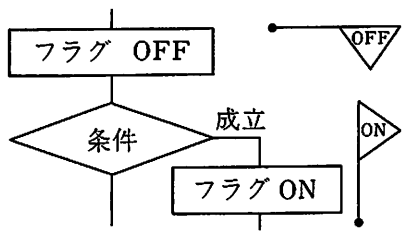
STOP文

```
STOP  RUN
```

プログラムの実行を停止する文である。

終了フラグ

プログラムの実行中に，ある条件が成立したかどうかを示す領域を**フラグ**という。この例題に用いているフラグは，入力データが終了したかどうかを判断するために用いるので，**終了フラグ**といわれる。



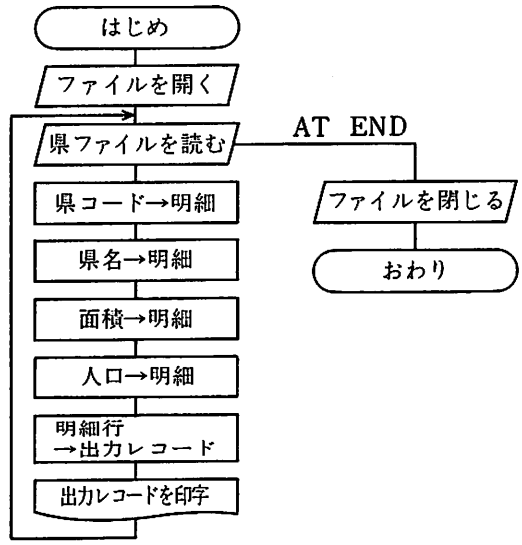
▲フラグ

PERFORM文で繰り返し処理を行う場合，繰り返しの終了をどこかで判断しなければならない。その判断に終了フラグを用いる。例題では，プログラムの実行に先立ち，終了フラグに「OFF」という文字定数が転記されており，入力ファイルの終了段階で「ON」という文字定数が転記される。入力ファイルが存在している間はPERFORM文を実行し，終了した時点で，終了条件を満足して，PERFORM文から抜け出る。

GO　TO 文を利用したプログラム

近年のプログラム開発では，構造化プログラミングの志向が浸透してきたため，処理の流れが複雑に入り組むGO　TO 文を多用したプログラムは書かれなくなってきた。しかし，従来型のプログラムを解読するためには，GO　TO 文を知っておく必要がある

例題１を GO　TO 文を用いた場合の流れ図とプログラムを示しておこう。



GO　TO文

GO　TO文

GO　TO　　手続き名.

処理の順序を無条件で変更する命令である。流れ図記号は GO　TO の行き先に応じて矢線が用いられる。

例題１のプログラムにおいては，段落「YOMU」を入力データが終わるまで GO　TO 文で繰り返している。

GO　TO 文の最後にはピリオドを打つ。

READ文

READ文(2)

READ　入力ファイル名　AT END 文.

従来の GO　TO 文を利用したプログラムでは，AT　END 句だけを用いたREAD文が使われた。ファイルが終了した場合は，AT　END 句以降を実行し，終了しない場合は，READ文の次の文を実行する。AT　END 句以降には，条件によって処理が変わらない文(無条件文：MOVE，GO　TO など)を指定する。

```
PROCEDURE     DIVISION.
HAJIME.
    OPEN      INPUT KEN-FILE        OUTPUT ITIRAN-FILE.
YOMU.
    READ      KEN-FILE     AT END   GO TO  OWARI.
    MOVE      KENCODE      TO       M-KENCODE
    MOVE      KENMEI       TO       M-KENMEI
    MOVE      MENSEKI      TO       M-MENSEKI
    MOVE      JINKO        TO       M-JINKO
    MOVE      MEISAI-GYO   TO       ITIRAN-REC
    WRITE     ITIRAN-REC   AFTER    1
    GO TO     YOMU.
OWARI.
    CLOSE     KEN-FILE     ITIRAN-FILE
    STOP      RUN.
```

ファイルが終わったらOWARIにとぶ。

●練習問題●

1　次の条件でデータ部のファイル節の定義をしなさい。

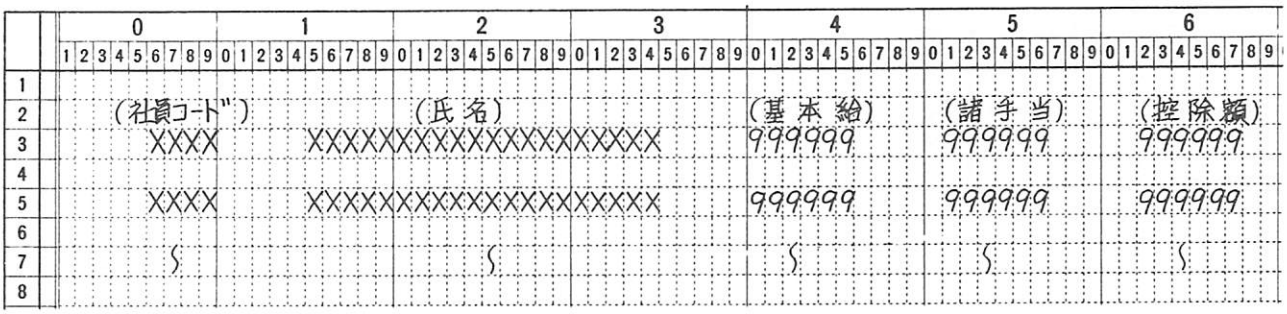
ファイル名　SYAIN−FILE　　レコード名　SYAIN−REC

社員コード	氏　名	基本給	諸手当	控除額
X (04)	X (20)	9 (06)	9 (06)	9 (06)

SCODE　　NAME　KIHON　TEATE　KOJO　　⇦項目名

2　次のスペーシングチャートより，作業場所節の明細行の定義をしなさい。

レコード名：MEISAI−GYO



（項目名）　M-SCODE　　M-NAME　　M-KIHON　　M-TEATE　　M-KOJO

3　次のA群は手続き部の文，B群はその文の機能である。関連するものを選び，記号で答えなさい。

A群	B群
(1)　OPEN文	ア．プログラムの実行を停止する。
(2)　PERFORM文	イ．プログラムで使用するファイルの準備をする。
(3)　READ文	ウ．処理の実行順序を変更する。
(4)　WRITE文	エ．データの内容を出力する。
(5)　MOVE文	オ．条件が成立するまで繰り返す。
(6)　STOP文	カ．ファイルからレコードを読み取る。
(7)　CLOSE文	キ．送出し側の内容を受取り側へ転記する。
(8)　GO　TO 文	ク．使用したファイルを閉じる。

4　次の MOVE 文実行後の受取り側の内容はどうなっているか答えなさい。

送出し側		受取り側	
PICTURE句	記憶内容	PICTURE句	転記後の内容
PIC　X(05)	VWXYZ	PIC　X(05)	ア
PIC　X(05)	VWXYZ	PIC　X(07)	イ
PIC　X(05)	VWXYZ	PIC　X(02)	ウ
PIC　9(04)	7568	PIC　9(04)	エ
PIC　9(04)	7568	PIC　9(07)	オ
PIC　9(04)	7568	PIC　9(01)	カ

◀実習問題▶

1　健康診断ファイルを読んで，健康診断一覧表を作成しなさい。

▼入力形式

社員コード	社員名	身　長	体　重
数字５桁	英数字10桁	数字３桁	数字３桁

▼出力形式

（社員コード）	（社員名）	（身長）	（体重）
×××××	××××××××××	×××	×××
×××××	××××××××××	×××	×××
×××××	××××××××××	×××	×××
∫	∫	∫	∫

2　在庫ファイルを読んで，在庫一覧表を作成しなさい。

▼入力形式

商品コード	商品名	最大在庫量	発 注 点	現在庫量
数字４桁	英数字10桁	数字５桁	数字５桁	数字５桁

▼出力形式

（商品コード）	（商品名）	（最大在庫量）	（発注点）	（現在庫量）
××××	××××××××××	×××××	×××××	×××××
××××	××××××××××	×××××	×××××	×××××
××××	××××××××××	×××××	×××××	×××××
∫	∫	∫	∫	∫

第1章　プログラム作成の手順

1　a-ウ　b-キ　c-エ　d-イ　e-オ

2　(1) ウ (2) エ (3) キ (4) カ (5) ア (6) オ (7) イ

3　a-オ　b-イ　c-ウ　d-カ　e-ア　f-エ

第2章　基本的なプログラム

1
```
FILE    SECTION.
FD  SYAIN-FILE.
01  SYAIN-REC.
    02  SCODE  PIC  X(04).
    02  NAME   PIC  X(20).
    02  KIHON  PIC  9(06).
    02  TEATE  PIC  9(06).
    02  KOJO   PIC  9(06).
```

2
```
WORKING-STORAGE  SECTION.
01   MEISAI-GYO.
    02    PIC  X(05)  VALUE  SPACE.
    02  M-SCODE  PIC  X(04).
    02    PIC  X(05)  VALUE  SPACE.
    02  M-NAME   PIC  X(20).
    02    PIC  X(05)  VALUE  SPACE.
    02  M-KIHON  PIC  9(06).
    02    PIC  X(05)  VALUE  SPACE.
    02  M-TEATE  PIC  9(06).
    02    PIC  X(05)  VALUE  SPACE.
    02  M-KOJO   PIC  9(06).
```

3　(1) イ (2) オ (3) カ (4) エ (5) キ (6) ア (7) ク (8) ウ

4　ア．VWXYZ　イ．VWXYZ△△　ウ．VW
エ．7568　オ．0007568　カ．8

第3章　演算と編集

1　ア．ABC　イ．ABCDE△△　ウ．00123△
エ．098　オ．0009876　カ．△9876
キ．876　ク．△9,876　ケ．△△△△△△△△0
コ．△△△△△△△△△

2
```
(1)  COMPUTE  A  =  B  *  C
(2)  COMPUTE  A  =  B  -  C  **  2
(3)  COMPUTE  A  =  (C  +  B)  /  D
(4)  COMPUTE  A  =  B  -  (C  +  D)
                    /  (E  *  F)
(5)  COMPUTE  A  =  (B  **  2  -  C
                **  2  )  /  D  **  3
(6)  COMPUTE  A  =  B  /  (C  /  D)
```

3　ア．￥98,765　イ．￥98,765　ウ．￥765
エ．△△△￥765　オ．￥△△△432　カ．△△△￥432
キ．￥△△△0　ク．△△△￥0

第4章　条件による判定

1
```
(1)  IF  X  NOT  =  100
     THEN  MOVE  Y  TO  Z
     ELSE  CONTINUE
     END-IF
(2)  IF  X  <=  50
     THEN  COMPUTE  Y  =  Y  +  1
     ELSE  COMPUTE  Z  =  Z  +  1
     END-IF
(3)  IF  X  <  15
     THEN  MOVE  Y  TO  Z
           COMPUTE  W  =  W  +  1
     ELSE  MOVE  Z  TO  Y
           COMPUTE  W  =  W  -  1
     END-IF
(4)  IF  X  >=  10
     THEN  COMPUTE  Y  =  Y  +  1
     END-IF
     IF  X  <=  50
     THEN  COMPUTE  Z  =  Z  +  1
     END-IF
```

2　(1) A＜B　(2) A＞B　(3) A＝B
(4) A＞B　(5) A＝B

3　ア．23,456　イ．121.2　ウ．△123.450
エ．12.340　オ．△△.987　カ．△123.4
キ．△△△98　ク．9876.5　ケ．−12345
コ．＋12345　サ．△△￥345　シ．−△￥345
ス．△△￥345　セ．＊＊＊345　ソ．△△△＋345
タ．△−1,345

4
```
(1)  01   YAKUSYOKU  PIC  X(01).
          88  BUCHO      VALUE  " 1 ".
          88  KACHO      VALUE  " 2 ".
          88  KAKARICHO  VALUE  " 3 ".
(2)  01   NENREI       PIC  9(02).
          88  SYONEN  VALUE 15  THRU  19.
          88  SEINEN  VALUE 20  THRU  39.
          88  CHUNEN  VALUE 40  THRU  49.
          88  JITUNEN VALUE 50  THRU  69.
          88  RONEN   VALUE 70  THRU  99.
```

5
```
(ア)  SYOKUSYU
(イ)  TEATE  =  JIKAN  *  700
(ウ)  COMPUTE  TEATE  =  JIKAN  *  1000
(エ)  OTHER  COMPUTE  TEATE  =  JIKAN
                              *  200
(オ)  END-EVALUATE
```