ステップ30

COBOL
ワークブック

CUTT
カットシステム

もくじ

謝 辞

COBOL言語仕様は，CODASYL（the COnference on DAta SYstems Languages：データシステムズ言語協議会，コダシル）によって開発されました。CODASYLの要求にしたがって，「CODASYL COBOL JOURNAL OF DEVELOPMENT 1984」の謝辞の一部を以下に掲げます。

COBOLは産業界の言語であり，特定の団体や組織の所有物ではない。CODASYL COBOL委員会又は仕様変更の提案者は，このプログラミングシステムと言語の正確性や機能について，いかなる保証も与えない。さらに，それに関する責任も負わない。
次に示す著作権表示付き資料の著作者及び著作権者は，これら全体又は一部分をCOBOLの原仕様書中に利用することを許可した。この許可は，COBOL原仕様書をプログラミングマニュアルや類似の刊行物に複製したり，利用したりする場合にまで拡張される。
FLOW-MATIC(Sperry Rand Corporationの商標),Programming for the Univac(R) I and II,Data Automation Systems，Sperry Rand Corporation著作権表示1958年,1959年
IBM Commercial Translator Form No.F 28-8013,IBM著作権表示1959年
FACT,DSI 27A5260-2760,Minneapolis-Honeywell 著作権表示1960年

Step 01

COBOLとは

COBOLは、現在でも事務処理用の言語として広く使われています。
アメリカ国防省において事務データ処理のための共通言語を作成することが必要であるか、また必要とした場合にそれを実現することが可能であるかどうかを検討するため、企業および政府のユーザ、電子計算機メーカ、その他これに関心をもつ団体などのそれぞれの代表者によってCODASYL（COnference on Data System Languages）が組織され、この組織で研究開発されたものがCOBOLです。この会議でこのような言語を開発することが必要であり、また実現可能であることが確認され、その実際の開発に当たるものとして、短期作業委員会、中期作業委員会、長期作業委員会を設立しました。
短期作業委員会は1959年に大綱をまとめ、CODASYLはこれをCOBOLと命名しました（グレース・M・ホッパー女史が開発した事務計算用コンパイラFLOW-MATICで、自然言語（英語）の一部を原始コードとして採用していて、これが後にCOBOLの原型となりました）。

1.1 COBOLの特徴

COBOLはたいへんわかりやすく、コンピュータを知らない人でも短期間の勉強ですぐにプログラムが書けるようになります。

COBOLは、プログラムの文書化ということに注意をはらって開発された最初の言語で、人間の言葉にもっとも近くわかりやすくできています。すなわち、プログラムをそのままの形で文書として保存できるのです。

COBOL自身が、単純な英語に似た構文をもっているので、読むだけで内容を理解でき、メンテナンスも楽です。

1.2 COBOLプログラムとCOBOLコンパイラ

普通、COBOLとは、COBOL言語とCOBOLコンパイラのことを指します。

プログラマは、COBOL言語を使ってプログラム（COBOLプログラム）を書きます。

コンピュータは、機械語しか理解できないので、COBOLプログラムを、コンピュータが直接理解できる機械語に翻訳編集するプログラムが必要となります。これが、COBOLコンパイラです。

以下の順番でCOBOLは動きます。

①ソース（原始）プログラム、つまりコンパイラにかける前のプログラムを書く。
②コンパイラにかけ機械語に変換する（コンパイルするといいます）。
③プログラムを動かすのに必要なファイルなどの環境をつなげる（リンクする）。
④プログラムを動かす（実行する）。

1.3 言語の構造

文字列　プログラムの中で、つなっがている文字の列をいいます。文字列には、以下の種類があります。

COBOLの語

A、B、C、……、Z、0、1、2、……、9、-の30文字以下の組み合わせの文字列です。最初の一文字目はA、B、C、……、Zで、必ず一意（UNIQUE）でなければいけません。

以下のいずれかに分類されます。

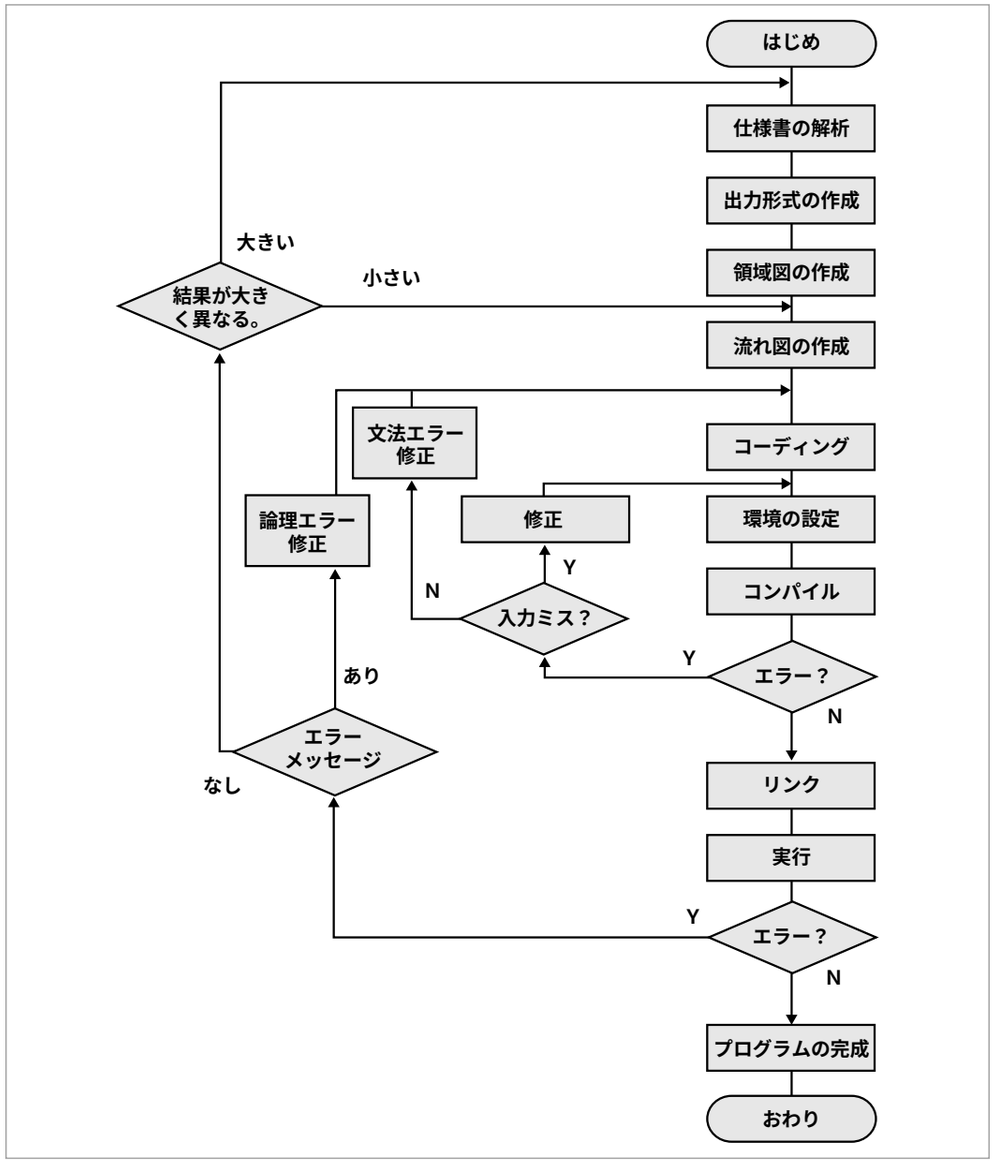
- **利用者語**　プログラマが作るデータ名や手続き名など。30文字以下の文字列で、1文字以上の英字が文字列の中になければいけません。
- **システム名**　計算機名、作成者名、言語名のいずれか。
- **予約語**　特に注意するべきもので、あらかじめ意味が定義されていて、定められた書き方でしか使ってはいけない語のこと。例えば、COBOL、DATA、DATEなど。
- **定数**　以下のものがあります。
  文字定数　コンピュータで使用できる文字集合から任意に作る文字列で、両端を引用符（"）でくくったもの。その値は、両端の引用符をはずした文字列。1文字以上160文字以下でなければいけません。"EIGO"、"SUGAKU"、"KOKUGO"など。
  数字定数　数値そのものを表すデータ。符号（ないときは正）、小数点（ないときは整数）、1つ以上の数字の組み合わせで作られています。1桁以上18桁以下の数字でなければいけません。123、-25.0、7、.5（＝0.5のこと）など。
  表意定数　特別の数値または文字列を表す予約語。ZERO（数字ゼロまたは何桁かの文字ゼロ）、SPACE（何桁かの空白）、QUOTE、HIGH-VALUE、LOW-VALUE、ALL。
- **PICTURE句**　項目名の形式を決める句で、基本項目にだけ書きます。

分離符（区切り文字）

COBOL語では、COBOLの文字列ごとに間隔を置いて、語と語をはっきり区切って書きます。この間隔を分離符といいます。また、文字列の終わりを識別し、後続の文字列と分離するための文字または連続する文字の並びです。区切り文字自身は、文字列に属しません。空白は分離符。分離符はいくつ書いても1つと同じとみなされます。

COBOL語の世界では、「空白」と「間隔」は同じです。

1.4　プログラミングの手順



1.5　正書法

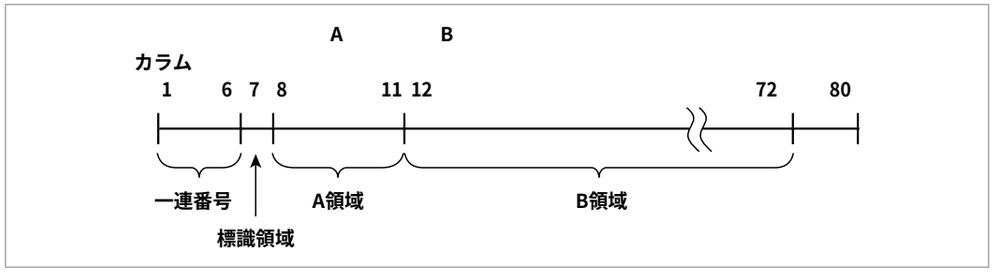
フローチャート（流れ図）に従って、プログラムを書くことをコーディングといいます。

最近は、コーディングせずに、直接プログラムを入力する人も多いですが、論理的にまとめるためにも、エラーをなくすためにも、きちんとコーディングすることをお薦めします。特に、勉強している過程では必ずコーディングをしてください。

コーディングの書き方（もちろん、入力も含みます）には、守るべきいくつかの約束があります。

記述用紙は、コーディングシートといいます。

コーディングシート



・**一連番号**：1～6カラム目
　プログラムの先頭から6桁の数字で行単位に割り振られる番号です。利用者が特定の順序を任意に付けることができます。

・**標識領域**：7カラム目
　"*"で、その行が、注釈行であることを示します。

・**A領域**：8～11カラム目
　部、節の見出し部、段落名、レベル番号01または77、見出し部・環境部の内容、データ部の一部をA領域より書き始めます。

・**B領域**：12～72カラム目
　特に指定のない記述は、B領域から書きます。

※パソコンCOBOLでは、何カラムまで書いてもエラーになることはありませんが、メインフレームでは、エラーになります。どちらで動かしても正しく実行することができるように、73カラム目以降は何も書かないでください。

昔は、今のようにキーボードから直接パソコンに入力するのではなく、カードパンチャーからカードに、プログラムやデータを打ち込み、カードリーダーでコンピュータに読み込んでいました。このカードが、1枚80桁の形式でできていたので、コーディングシートがこの形式でできています。同様に入力データも80バイト（桁）の大きさで作成されることが多いです。

Step 02

COBOLプログラムの構造

ここでは、COBOLプログラムを構成する最も基本的な要素を説明します。

2.1 プログラムの構造

COBOLプログラムは以下のDIVISIONで構成されます。

IDENTIFICATION DIVISION. **（見出し部）**

プログラム全体の見出しで、プログラム名などを書きます。

PROGRAM-ID. プログラム名.

プログラムに名前を付けます。

AUTHOR. 作成者名.

プログラム作成者の名前を書きます（自分の書いたプログラムに責任をもつためにも、書きましょう）。

ENVIRONMENT DIVISION. **（環境部）**

使用するコンピュータの機器構成や、特別の技法など機械に関することを書きます。プログラムが、どのような環境で実行されるかを指定します。

CONFIGURATION SECTION. **（構成節）**

SOURCE-COMPUTER. 翻訳用計算機名.

ソース（原始）プログラムをコンパイルする計算機名を書きます。

OBJECT-COMPUTER. 実行用計算機名.

実行用プログラムを実行する計算機名を書きます。

INPUT-OUTPUT SECTION. **（入出力節）**

FILE-CONTROL. **（ファイル管理段落）**

```
SELECT ファイル名1 ASSIGN TO 装置名1(アクセス名1).
SELECT ファイル名2 ASSIGN TO 装置名2(アクセス名2).
SELECT ファイル名3 ASSIGN TO PRINTER.
```

入出力ファイルを選択（SELECT）し、装置名を割り当てます（ASSIGN）。

DATA DIVISION. **（データ部）**

入力領域と出力領域をどのように用意するかを指定します。また、作業領域や中間領域を指定します。

PROCEDURE DIVISION. **（手続き部）**

仕事をどのような手順で行うか指定します。

2.2 データ部の構造

DATA DIVISION.

FILE SECTION. **（ファイル節）**

領域図(ファイル領域)に書いた、入力領域と出力領域を指定します。

```
ファイル記述項〔レコード記述項〕 ・・・
FD  ファイル名.
01  ファイルレコード名.
    05  ファイル項目名1    PIC  X(05).
    05  ファイル項目名2    PIC  9(02).
          ・
          ・
```

WORKING-STORAGE SECTION. **（作業場所節）**

領域図（作業領域）に書いた、作業領域を指定します。

```
〔レコード記述項〕 ・・・
01  作業レコード名.
    05  作業データ名1    PIC  X(03)  VALUE 定数1.
    05  作業データ名2    PIC  9(06)  VALUE 定数2.
          ・
          ・
```

VALUE句は、その領域に初期値を入れる場所。

【例1】

データ項目が英数字項目の場合は、定数を文字定数か表意定数で指定します（ここでのSPACEは文字として扱われる表意定数）。

```
01  FLG  PIC X(01)  VALUE SPACE.
```

英数字1桁の項目名FLGが確保され、初期値に空白が入ります。

【例2】

データ項目が数字項目の場合は、定数を数字か表意定数で指定します（ここでのZEROは数字として扱われる表意定数）。

```
01  P-CNT  PIC 9(02)  VALUE ZERO.
```

数字2桁の項目名P-CNTが確保され、初期値に0が入ります。

【例3】

VALUE句を書かないと、その領域には最初何が入っているかわかりません。

```
01  GOKEI   PIC 9(05).
```

数字5桁の項目名GOKEIだけが確保されます。初期値は不定です。

PICは、PICTURE句の省略形です。

※必要のないSECTIONは、省略できます。

2.3　手続き部

PROCEDURE DIVISION.

フローチャートの部分をコーディングします。

フローチャートには、いくつかの基本パターンがあります。例えば、改頁処理・グループトータル処理・（内部）ソート処理・マッチング処理などです。

これらのフローチャートはとても合理的で、わかりやすいものです。基本パターンを覚えて、有効に使いましょう。

主要な用語

・ファイル

同じ種類のデータが集まったもので、ある目的のために集められたレコードの集まりです。

・レコード

関連のある情報をもった項目の集まりで、1つの処理単位となるものです。

・項目

データが情報として意味をもつ最小の単位です。

基本項目　それ以上細分化されない項目

集団項目　基本項目の集まり

独立項目　その項目だけで独立していて、階層構造にならない項目

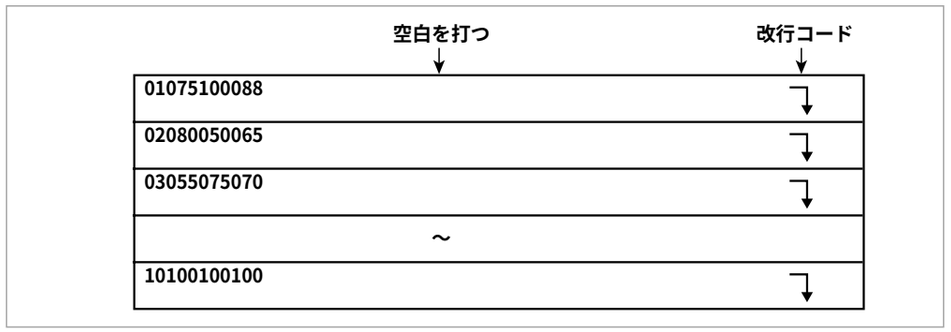
※最近は、ファイル名やデータ名などの利用者語を日本語で書くことができますが、世界中どこの機械でも動くプログラムを作成するためには、日本語の利用者語は使用しないほうがよいでしょう。

2.4　本書で使用するファイルの種類

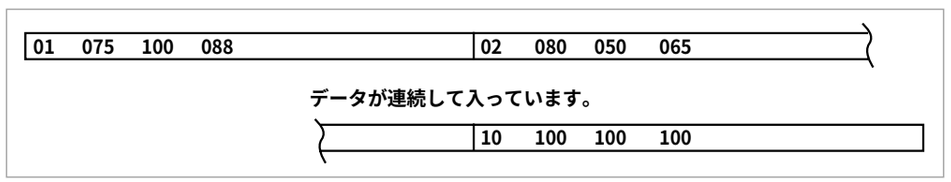
行順編成ファイル（固定長）

1レコードごとに、改行コードが入っています。一般的には、レコードが連続して入っている順編成ファイルを使いますが、行順編成ファイルは、1レコードごとに、改行コードが入っているテキストファイルです。メモ帳で作成したり、データの変更や確認も簡単にできます。

・行順編成ファイルの例



・順編成ファイルの例



索引順編成ファイル（固定長）

辞書のように、索引が付いているファイルと考えてください。指定したキーが、（物理的または、論理的に）昇順に並んでいます。

※固定長とは、全てのレコードが一定の長さ（固定されている）からなるファイルです。

では、以下のStepから、命令語を勉強しながら、例題プログラムを作成していきましょう（プログラムは、富士通株式会社のPowerCOBOL97を使って作成しています）。