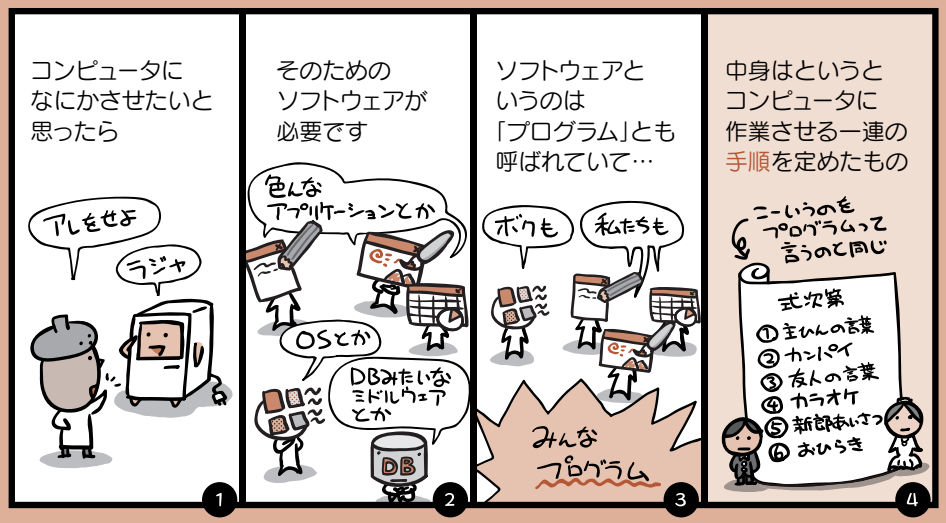
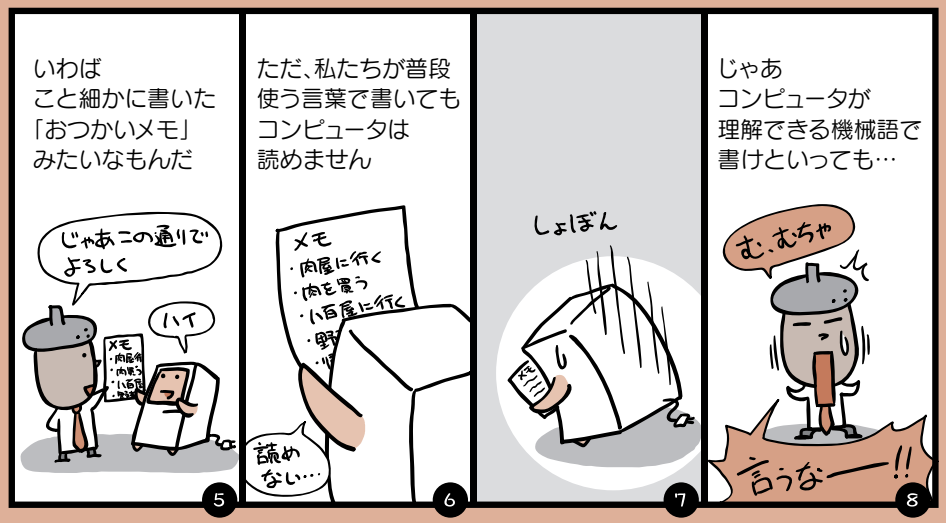
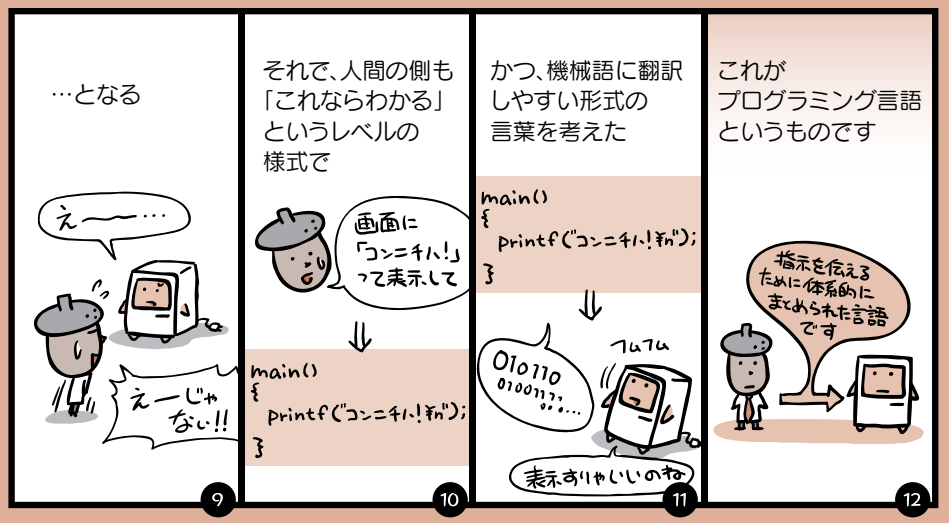
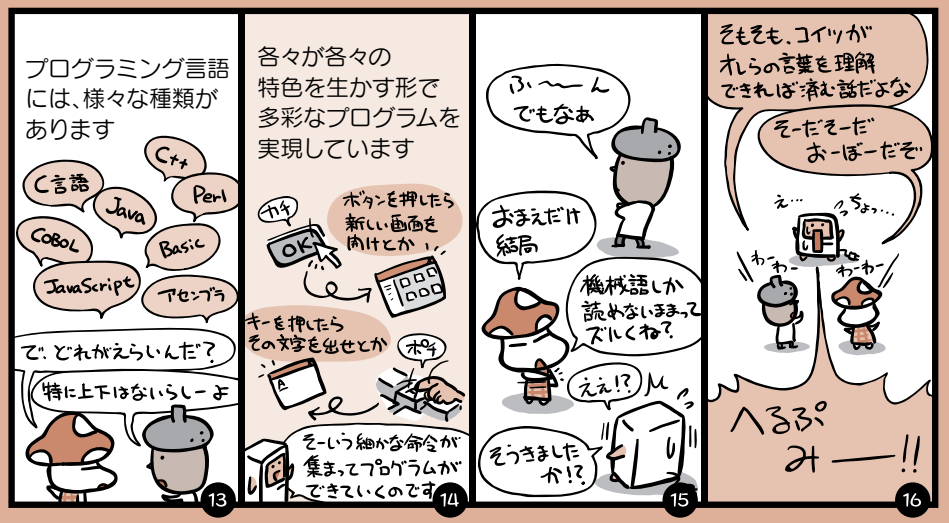
Chapter 16

プログラムの作り方



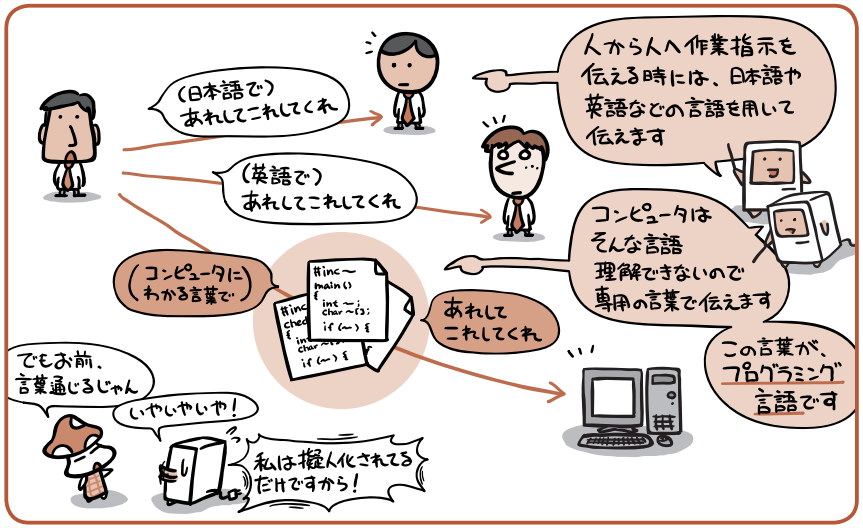






Chapter 16-1

プログラミング言語とは



コンピュータに作業指示を伝えるための言葉、
それがプログラミング言語です。

「コンピュータは機械語しかわかりませんよ」というのは前にも述べた通りです。しかしだからといって私たちが機械語を話すというのも難しい話。英語や中国語ならちょっとがんばってみようかなと思わなくもないですが、機械語は…ねぇ。

というわけで、「じゃあウチらの作業指示を、機械語に翻訳して伝えればいいんじゃね」というアイデアが生まれることになるわけです。本当なら、そこで日本語がそのまま通じてくれれば話が早いのですが、残念ながら翻訳機もさすがにそこまでは賢くない。

それなら…と、「機械語に翻訳しやすくて、かつ人間にもわかりやすい中間の言語」が作られました。

もうおわかりですよね。それがプログラミング言語というわけです。

私たちの使う言葉に日本語や英語や中国語やギャル語などの様々な言語があるように、プログラミング言語も用途に応じて様々な言語が存在します。代表的なのはC言語やJavaなど。それでは各々の特徴からまずは見ていくといたしましょう。

代表的な言語とその特徴

代表的なプログラミング言語には下記のようなものがあります。

C言語（シー）

OSやアプリケーションなど、広範囲で用いられている言語です。

もともとはUNIXというOSの移植性を高める目的で作られた言語なので、かなりハードウェアに近いレベルの記述まで出来てしまう、何でもアリの柔軟性を誇ります。

BASIC（ベーシック）

初心者向けとして古くから使われている言語です。

簡便な記述方法である他に、書いたその場ですぐ実行して確かめることができるインタプリタ方式（これについては次ページで）が主流という特徴を持ちます。そのため未完成のコードでも、途中まで実行して動作を確認したりしながら開発を進めることができます。

COBOL（コボル）

事務処理用に古くから使われていた言語です。

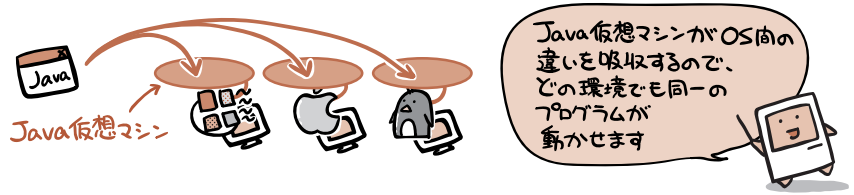
現在では、新規のシステム開発でこの言語を使うというのはまずなくなりました。ただし、大型の汎用コンピュータなどで古くから使われているシステムでは、過去に作ったCOBOLのシステムが今でも多く稼働しています。そのため、システムの改修などではまだまだ出番の多い言語です。

Java（ジャバ）

インターネットのWebサイトや、ネットワークを利用した大規模システムなどで使われることの多い言語です。

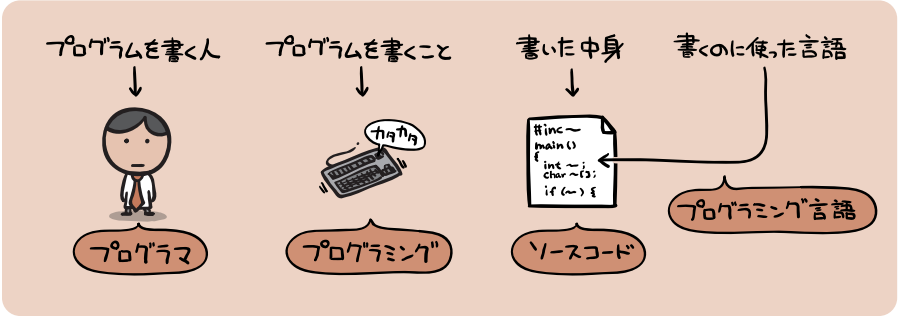
C言語に似た部分を多く持ちますが、設計初期からオブジェクト指向やネットワーク機能が想定されていたという特徴を持ちます。

特定機種に依存しないことを目標とした言語でもあるため、Java仮想マシンという実行環境を用いることで、OSやコンピュータの種類といった環境に依存することなく、作成したプログラムを動かすことができます。

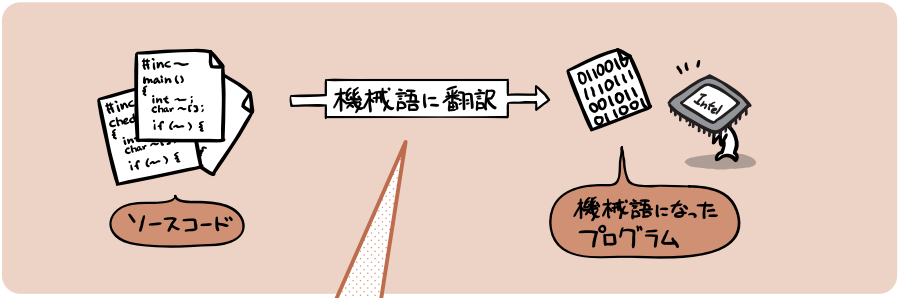


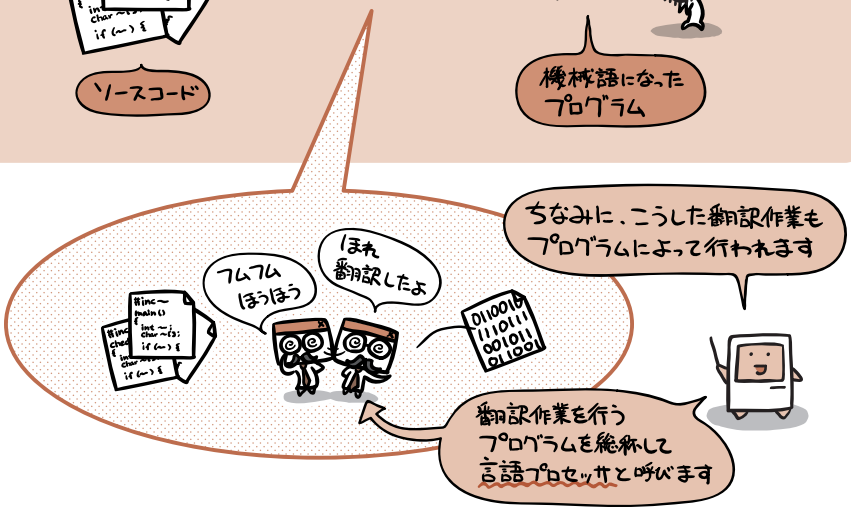
インタプリタとコンパイラ

さて、プログラムというのは、このプログラミング言語を使って命令をひとつひとつ記述していくことで作られます。ここでちょっと「あれ? どこかで見たような」という図を引っぱり出して復習してみましょう。用語の理解はバッチリですか?



このソースコードを機械語に翻訳することで、プログラムはコンピュータが実行できる形式となるわけです。

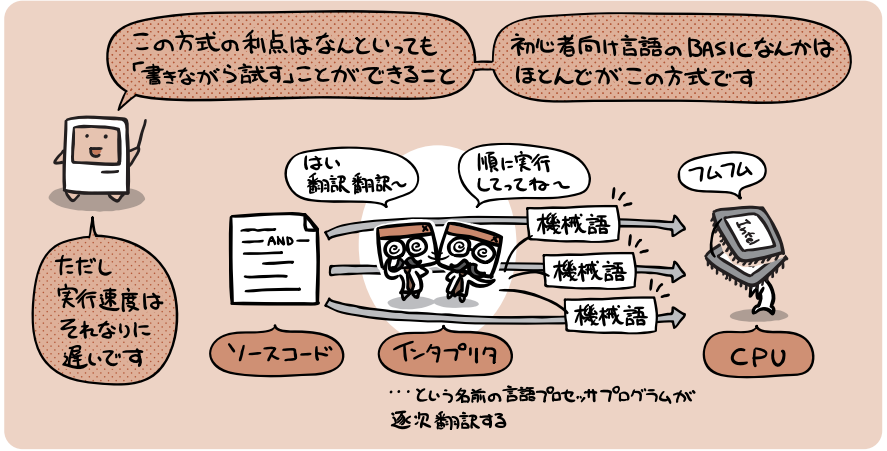




この翻訳には、2種類の方法があります。そう、これまでチラリチラリと登場していたインタプリタ方式やコンパイラ方式というのがそれなのです。

インタプリタ方式

この方式では、ソースコードに書かれた命令を、1つずつ機械語に翻訳しながら実行します。逐次翻訳していく形であるため、作成途中のプログラムもその箇所まで実行させることができるなど、「動作を確認しながら作っていく」といったことが容易に行えます。



コンパイラ方式

この方式では、ソースコードの内容を最初にすべて翻訳して、機械語のプログラムを作成します。ソースコード全体を解釈して機械語化するため、効率の良い翻訳結果を得ることができますが、「作成途中で確認のために動かしてみる」といった手法は使えません。

このように出題されています

過去問題練習と解説

問1 (FE-H30-A-08)

Javaの特徴はどれか。

ア　オブジェクト指向言語であり，複数のスーパクラスを指定する多重継承が可能である。
イ　整数や文字は常にクラスとして扱われる。
ウ　ポインタ型があるので，メモリ上のアドレスを直接参照できる。
エ　メモリ管理のためのガーベジコレクションの機能がある。

解説

ア　Javaはオブジェクト指向言語ですが、複数のスーパクラスを指定する多重継承はできません（＝1つのスーパクラスを指定する単一継承しかできません）。
イ　整数や文字は、通常、変数に格納されて使用されます。
ウ　Javaには、C言語などで使われるポインタ型変数はありません。
エ　そのとおりです。なお、Javaのガーベジコレクション機能とは、参照されなくなったオブジェクトのメモリ領域を、主記憶装置上から開放する機能のことです。

正解：エ

問2 (FE-H16-S-44)

次の文はある二つの言語処理系について記述したものである。Bと比べたAの利点を記述しているものはどれか。

A：高水準言語で作成されたプログラムを、中間言語、アセンブラ言語又は機械語で記述されたプログラムに翻訳する。
B：原始プログラム中の命令文を一文ずつ解釈し、実行する。

ア　処理の最適化が図れる。
イ　対話的な実行環境が構築できる。
ウ　デバッグ機能を組み込みやすい。
エ　プログラム作成とテストを並行してできる。

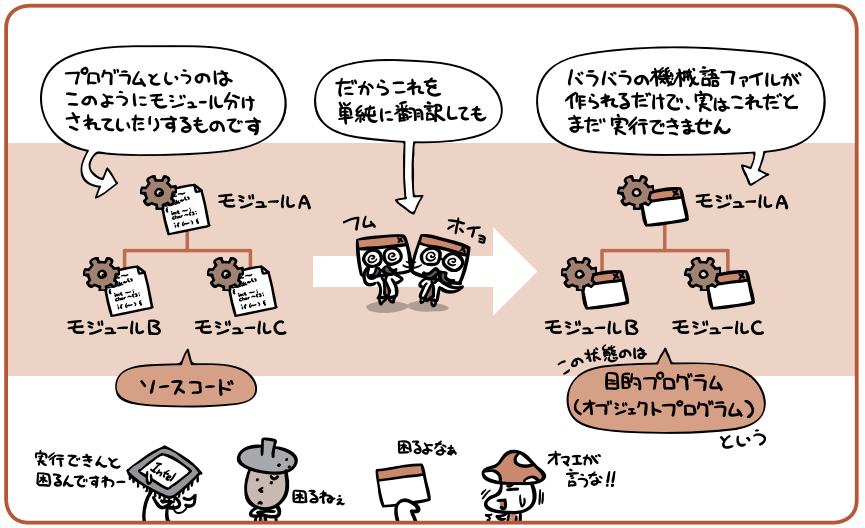
解説

Aはコンパイラ方式、Bはインタプリタ方式に関する記述です。
ア　コンパイラ方式は、インタプリタ方式と比較して処理の最適化が図れます。例えば、「定数の畳込み」です。これは、①行目：a ＝ 5 ＋ 8　②行目：b ＝ a ＊ 3のプログラムを、①行目：a ＝ 13　②行目：b ＝ 39　に置き換えてコンパイルすることです。これによって演算が代入になり、処理が高速化されます。
イとウとエ　これらは、Aと比べたBの利点の記述です。

正解：ア

Chapter 16-2

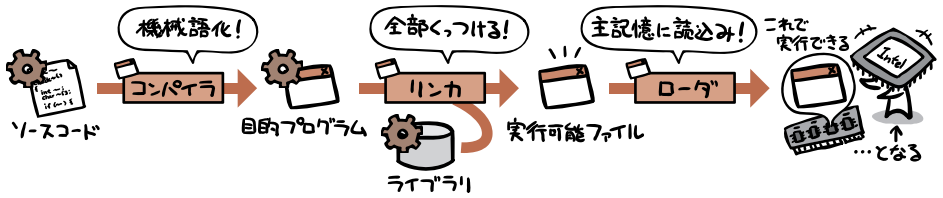
コンパイラ方式でのプログラム実行手順



「リンカ」というプログラムが、実行に必要なファイルをすべてくっつけることで、実行可能ファイルは生成されます。

コンパイラ方式のプログラムの場合、その実行に至るまでの過程では、コンパイラ以外に、2種類のプログラムが登場します。それがリンカとローダです。

ここでちょっと、この方式のプログラムが実行に至る流れを図にしてみましょう。



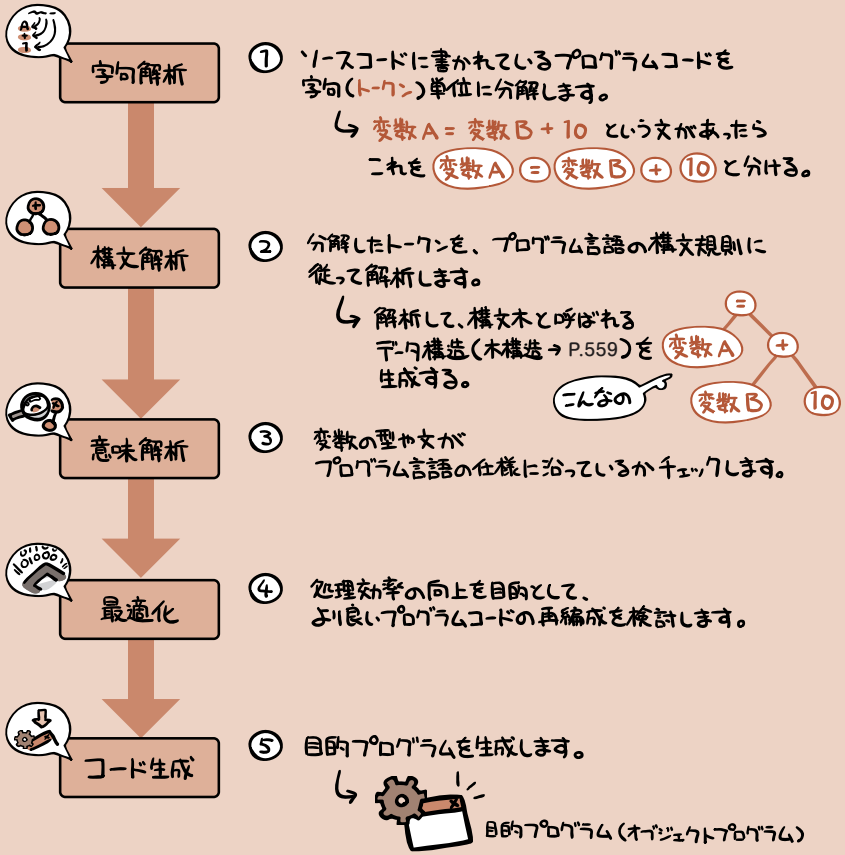
もう見たまんまでありますが一応ざっくり説明すると、「ソースコードを機械語化して、それを全部くっつけて、実行時にはこれを主記憶上に読み込む」というのが実行までの流れになるわけですね、うん。…え？あまりにざっくり過ぎる？

それでは上記流れの中に登場している、コンパイラとリンカとローダ、それぞれの行う仕事について、もう少し詳しく見ていきましょう。

コンパイラの仕事

コンパイラの仕事は、これまでに何度も書いている通り、「人間の側にわかるレベルの様式」…つまりはプログラム言語を使って書いたソースコードを、翻訳して機械語のプログラムファイルにすることです。

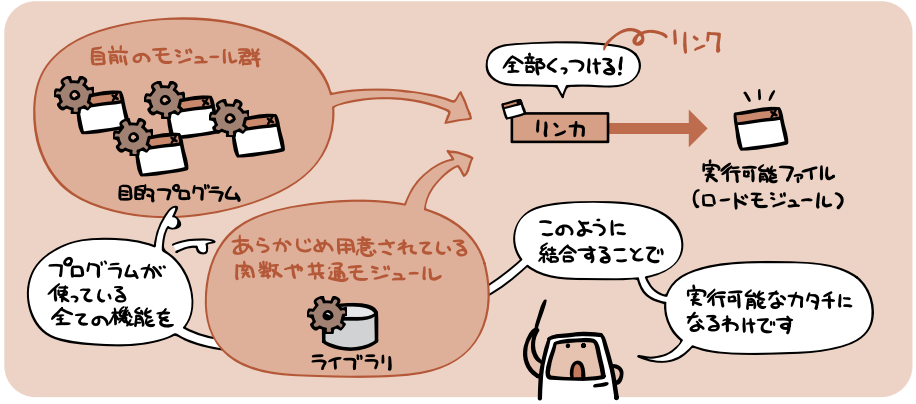
コンパイラの中では、ソースコードを次のように処理することで、目的プログラムを生成します。



リンカの仕事

プログラムは、自分で分割したモジュールはもちろん、ライブラリとしてあらかじめ提供されている関数や共通モジュールなどもすべてつなぎあわせることで、実行に必要な機能がそろったプログラムファイルになります。

この、「つなぎあわせる」作業をリンク（連係編集）と呼びます。つまりはこれが、リンカ（連係編集プログラム）の仕事というわけです。

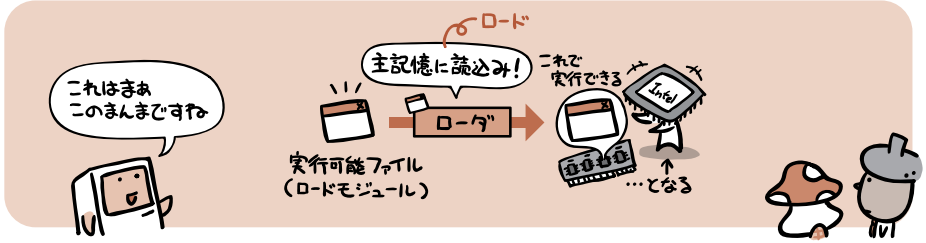


ちなみに、このような「あらかじめリンクしておく手法」を静的リンキングと呼びます。

一方、この時点ではまだリンクさせずにおいて、「プログラムの実行時に、共有ライブラリやシステムライブラリをロードしてリンクする手法」というのも存在します。こちらは動的リンキングと呼びます。

ローダの仕事

ロードモジュールを主記憶装置に読み込ませる作業をロードと呼びます。これを担当するプログラムがローダです。



このように出題されています

過去問題練習と解説

問1 (FE-H28-S-19)

コンパイラにおける最適化の説明として，適切なものはどれか。

ア　オブジェクトコードを生成する代わりに，インタプリタ用の中間コードを生成する。

イ　コンパイルを実施するコンピュータとは異なるアーキテクチャをもったコンピュータで動作するオブジェクトコードを生成する。

ウ　ソースコードを解析して，実行時の処理効率を高めたオブジェクトコードを生成する。

エ　プログラムの実行時に，呼び出されたサブプログラム名やある時点での変数の内容を表示させるようなオブジェクトコードを生成する。

解説

ア　コンパイラは、インタプリタ用の中間コードを生成しません。インタプリタの説明は、532ページを参照してください。

イ　クロスコンパイラの説明です。

ウ　コンパイラの最適化については、536ページを参照してください。

エ　プログラムの修正を容易するための「トレーサ」や「デバッガ」の一部の機能を説明しています。

正解：ウ

問2 (FE-H30-A-20)

リンカの機能として，適切なものはどれか。

ア　作成したプログラムをライブラリに登録する。

イ　実行に先立ってロードモジュールを主記憶にロードする。

ウ　相互参照の解決などを行い，複数の目的モジュールなどから一つのロードモジュールを生成する。

エ　プログラムの実行を監視し，ステップごとに実行結果を記録する。

解説

ア　<作成したソースプログラムを、ライブラリ（リポジトリ）に登録（チェックイン）する>と補って解釈すれば、ソフトウェア構成管理ツールやCASEツールの機能です。

イ　ローダの機能です。ローダの説明は、537ページを参照してください。

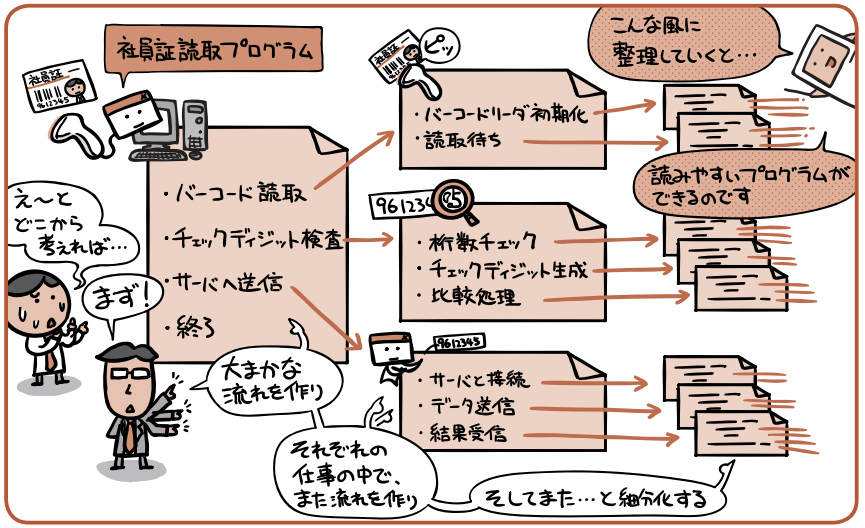
ウ　リンカの機能です。リンカの説明は、537ページを参照してください。

エ　トレーサやデバッガの機能です。

正解：ウ

Chapter 16-3

構造化プログラミング



構造化プログラミングは、プログラムを機能単位の部品に分けて、その組み合わせによって全体を形作る考え方です。

長い文章を、何の章立ても決めずにひと息で書こうとすると、往々にして「あれ? 何を書きたかったんだっけか」なんて迷走する結果になりがちです。

プログラミングもこれは同じ。ましてやプログラムの場合は「○○の場合は××をせよ」なんて条件分岐が色々出てきますから、アッチへ飛んだりコッチへ飛んだりと、後から読むのすら難しい…「そもそも本当に完成するのこれ?」といった、難物ソースコードいっちょあがりとなる可能性も否定できません。

それを避けようと生まれたのが**構造化プログラミング**。

この手法では、一番上位のメインプログラムには、大まかな流れだけが記述されることになります。当然それだけじゃ完成しませんから、大まかな流れのひとつひとつを、サブルーチンという形で別のモジュールに切り出してやる。このサブルーチンも、内部は大まかな流れを記述して、その詳細はサブルーチンで…と切り出していく。

このように少しずつ処理を細分化していくと、各階層ごとの流れがキチンと整理されることになります。結果、効率よく、ミスの少ないプログラムが出来上がるというわけです。