# コンパイル処理の流れ

宮田 賢一

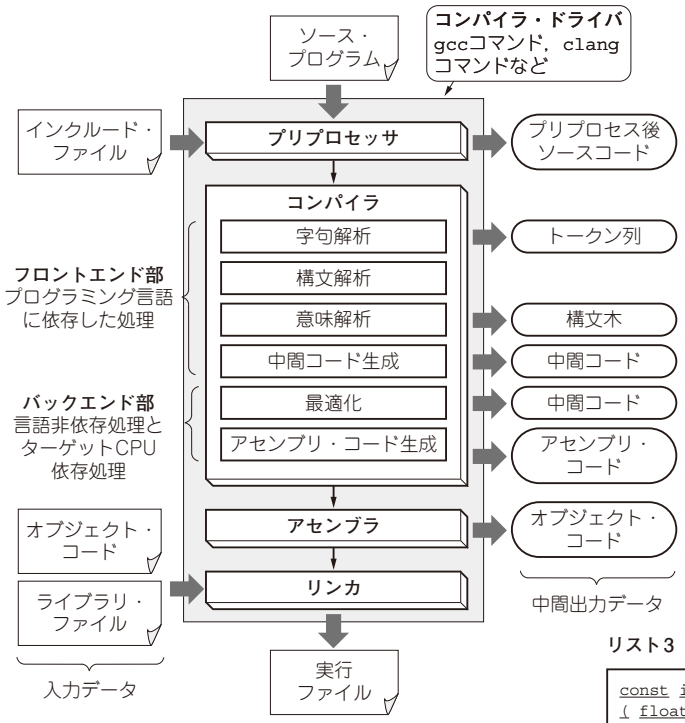


**図1　コンパイラ・ドライバの構成**（C言語の場合）

**リスト1　コンパイラ・ドライバへの入力コード**

```
01: // 型Tの引数を取り，型Tの値を返す関数定義を
02: // 生成するマクロ
03: #define DEFFUNC(T, a, b) \
04: T linear_##T(T x) { \
05:     T value; \
06:     value = a * x + b; \
07:     return value; \
08: }
09: const int N = 10;
10: DEFFUNC(float, 2, N)
```

**リスト2　プリプロセス後のプログラム**

```
01: const int N = 10;
02: // DEFFUNCを展開して3～7行目を得る
03: float linear_float(float x) {
04:     float value;
05:     value = 2 * x + N;
06:     return value;
07: }
```

**リスト3　字句解析で得られた字句の列**

```
const int <id, 1> = 10 ; float <id, 2>
( float <id, 3> ) { float <id, 4> ;
<id, 4> = 2 * <id, 3> + <id, 1> ; return <id, 4> ; }
```

あるプログラミング言語で書かれたソース・プログラムから実行ファイルを作成するツールをコンパイラと呼ぶことが多いのですが，この種のツールを正しくはコンパイラ・ドライバと言います．コンパイラ自体はコンパイラ・ドライバから呼び出されるツールの1つです．**図1**に一般的なコンパイラ・ドライバの構成を示します．本章では，コンパイラ・ドライバから呼び出されるツール群により，ソース・プログラムがどのように変換されていくかを追っていくことで，全体の処理を理解してみることにします．

サンプル・プログラムとしてC言語で書かれた**リスト1**をコンパイラ・ドライバへの入力とします．

## ● プリプロセッサ

プリプロセッサは，ソース・ファイルに他のファイル内容を挿入するインクルード機能や，同じパターンのコード片を1つにまとめて記述できるようにするマクロ機能などを持ちます．言い換えるとソース・プログラムを簡潔にし，一種の言語拡張を行うツールです．

**リスト1**をプリプロセッサに入力すると**リスト2**が得られます．DEFFUNC(float, 2, N)というマクロが展開され，float型の関数の定義に置き換わりました．

## ● コンパイラにおける処理の流れ

ここからコンパイラ本体の処理に入ります．コンパイラは本稿の主役ですので，コンパイラ内のステップを詳しく追っていくことにします．

### ▶ステップ1：字句解析

字句解析では，ソース・プログラムをプログラミン