やさしい C言語／C++ 入門

日向俊二◉著

CUTT カットシステム

■**サンプルファイルのダウンロードについて**

本書掲載のサンプルファイルは、下記 URL からダウンロードできます。

https://----------

・本書の内容についてのご意見、ご質問は、お名前、ご連絡先を明記のうえ、小社出版部宛文書（郵送または E-mail）でお送りください。
・電話によるお問い合わせはお受けできません。
・本書の解説範囲を越える内容のご質問や、本書の内容と無関係なご質問にはお答えできません。
・匿名のフリーメールアドレスからのお問い合わせには返信しかねます。

はじめに

プログラミング言語である C 言語と C++ 言語は、主要なプログラミング言語の中でも昔から特に重要な言語です。そして、現在でも進化している、技術者にとって必要不可欠な言語です。

C 言語は、言語としては比較的シンプルです。そのため、習得が比較的容易で、プログラミングの基礎教育にも良く使われますが、概念の習得がやや難しいポインタや再帰のような要素もあります。

C++ は、オブジェクトを扱うプログラミングに対応できるように C 言語を拡張してクラスを導入したプログラミング言語です。C++ はもともとは C 言語を拡張して作られた言語であるため、C 言語の知識に C++ 特有の事項を追加するだけでマスターすることができます。

本書では、C 言語と C++ との関係に着目して、C 言語と C++ を並行してやさしく解説します。

本書は、C 言語と C++ 言語を同時にマスターしたい読者や、C 言語について多少知識があって C 言語についての知識を確実にしながらさらに C++ を習得したい読者のための、C 言語と C++ のやさしい入門書です。

C 言語と C++ を並行して学ぶことで、C 言語 /C++ を効率よく学べるだけでなく、それぞれの言語の特性についての理解をより深めることができます。

2023 年春 著者しるす

■ 本書の表記

C/C++　C 言語と C++ 言語を表します。

【C++】　C++ 言語固有の事柄を表します。

()　ひとまとまりの実行可能なコードブロックである関数であることを示します。たとえば、main という関数を表すときに、「main という名前の関数」や「関数 main()」と表記しないで、単に「main()」と表記することがあります。

abc123　斜体で示す語は、そこに具体的な文字や数値、式などが入ることを表します。たとえば「if (*expr1* < *expr2*)」は、「if (x < 0)」や「if (a + b < 100)」などとなることを表します。

0x*n*　0x で始まる表記は 16 進数表現の整数であることを表します。たとえば、0x41 は 10 進数で 65 であることを表します。

0*n*　0 で始まる数値の表記は 8 進数表現の整数であることを表します。たとえば、041 は 10 進数で 33 であることを表します。

...　書式の説明において「...」は、任意の個数を記述できることを示します。

[ ]　書式の説明において [と] で囲んだものは、省略可能であることを示します。

［Key］　キーボードのキーを押すことを示します。たとえば、［F5］は F5 キーを押すことを意味します。

［A］+［B］　キーボードの A キーを押したまま B キー押すことを示します。たとえば、［Ctrl］+［F5］は Ctrl キーを押したまま F5 キーを押すことを意味します。

>　Windows のコマンドプロンプトを表します。

$　Linux や WSL など UNIX 系 OS のコマンドプロンプトを表します。

本文を補足するような説明や、知っておくとよい話題です。

処理系　C/C++ のコンパイラの具体的構成やコンパイルの方法、それが生成するファイルなどは、OS やコンパイラの種類とバージョンによって異なります。ある OS の特定の種類の特定のバージョン全体を表すときに、処理系という言葉を使います。たとえば、Linux 系の GCC と Windows で Microsoft のコンパイラを使うときはそれぞれ別の処理系であるとみなします。

標準出力　C/C++ のプログラムでは、出力にウィンドウを使うことがよくありますが、パイプやリダイレクトという OS の機能を使って他のファイルやデバイスへと出力することもできます。C/C++ ではこれを標準出力と呼びます。

標準入力　C/C++ のプログラムでは、入力にキーボードを使うことがよくありますが、パイプやリダイレクトという OS の機能を使って他のファイルやデバイスから入力することもできます。C/C++ ではこれを標準入力と呼びます。

ターミナル　コマンドラインを入力してプログラムを起動したり、入力や表示を行う場所（典型的にはウィンドウ）を指します。Linux など UNIX 系システムでは一般にターミナル（Terminal）と呼ばれますが、Windows 10 の場合は「コマンド プロンプト」か「PowerShell」（切り替え可能）で、Windows 11 の場合は既定のターミナルは「Windows Terminal」または「Windows ターミナル」という名前で用意されています。ターミナルという言葉を初心者に理解しやすいように標準出力の代わりに使うことがあります。

キーボード　標準入力からの入力を、初心者にわかりやすいようにキーボードからの入力と呼ぶ場合があります。

ご注意

- 本書の内容は本書執筆時の状態で記述しています。C/C++ の場合、コンパイラの種類やバージョンによって異なる点があり、一般的でない環境の場合は、本書の記述と実際とが異なる結果となる可能性があります。また、第 12 章の記述内容は処理系や実行環境に依存します。
- 本書は C/C++ のすべてのことについて完全に解説するものではありません。必要に応じて C/C++ のドキュメントなどを参照してください。
- 本書のサンプルは、プログラミングを理解するために掲載するものです。実用的な

アプリとして提供するものではありませんので、ユーザーのエラーへの対処やセキュリティー、その他の面で省略してあるところがあります。

■ 本書に関するお問い合わせについて

本書の内容に関するご質問については、sales@cutt.co.jp にメールでお問い合わせください。

なお、本書の記述内容から外れるご質問にはお答えできません。特に、特定の環境における特定のコンパイラや開発ツールのインストールや設定、使い方、読者固有の環境におけるエラーなどについてご質問いただいてもお答えできませんので、あらかじめご了承ください。

お問い合わせの際には下記事項を明記してください。

- 氏名
- 連絡先メールアドレス
- 書名
- 記載ページ
- お問い合わせ内容
- 実行環境

もくじ

第1章

C言語とC++の基礎知識

ここではC言語とC++についての基本的なことがらを紹介します。

1.1 C 言語と C++

C 言語は、現在、最も重要で広く使われているプログラミング言語の一つです。

C++ 言語は C 言語を拡張して作成されたオブジェクト指向プログラミング言語です。言い換えると、C++ は C 言語から生まれました。そのため、C 言語と C++ のシンタックスの大半は同じです。実際、C++ のプログラムの中では C 言語のコードをほとんどそのまま使うことができます。そのため、C++ 固有のいくつかの点を除くと、C 言語と C++ はほとんど同じものと考えることも可能です。しかし、それぞれの言語を適切に利用するためには、それぞれの言語の特性をきちんと把握しておく必要があります。

ここでは、C 言語と C++ のプログラムについて最も基本的なことを説明します。

◆C 言語 ◆

プログラミング言語 C は、さまざまな目的に使われている、最も重要なプログラミング言語の一つです。また、C 言語は、現在、いろいろな分野で広く使われている多様なプログラミング言語の中では、歴史が比較的長く、標準化が進んでいるため、さまざまなシステムでコンパイルして実行できるプログラミング言語でもあります。

C 言語の特徴は次のとおりです。

- プログラミング言語として標準化されており、そのライブラリもまた標準化されています。システムコールやウィンドウシステムのようなプラットフォーム固有の機能に依存する部分を除いて、プログラムの移植が比較的容易です。
- 言語仕様が比較的単純でありながら、機能を実現するさまざまなライブラリを活用することで複雑高度なことも実現できます。
- システムのリソース（メモリや CPU など）を直接利用する、OS やデバイスドライバのような低レベルのプログラムの記述にも適しています。組み込み機器のソフトウェアのプログラミングにも良く使われます。
- コンパクトで高速なプログラムを記述したり生成したりすることが可能です。
- C 言語で記述したモジュールは、他のプログラム言語とのリンクの標準的な方法として使われることが多く、さまざまなプログラム言語のプログラムとのリンクが比

較的容易です。C++ のプログラムも C 言語のインターフェイスを介して他の言語のモジュールとアクセスできるようにすることができます。

C 言語が理解しやすいといえる例をいくつか示します。

```
x = y * 2;
```

これは、y という変数に入っている値を 2 倍して、結果を変数 x に保存するコードの例です。直感的にとてもわかりやすいです。

```
printf("Hello, C¥n");
```

これは、printf という名前の関数と呼ばれるものを使って、「Hello, C」と出力（表示）して改行するコードです。printf() は文字列を出力するもので、¥n は改行のためのシンボルを示すことがわかれば、その動作は容易に理解できるでしょう。

なお、これはコード断片なので、これだけでは実行できません。実行できるプログラムはあとで示します。

◆ C++ ◆

プログラミング言語 C++ は、**オブジェクト指向プログラミング**のために生まれたプログラミング言語です。オブジェクト指向では、クラスというものを定義してオブジェクトというもの作って利用します。そして、クラスにはデータと操作や処理を行うためのコードが含まれますが、これについては第 9 章「構造体とクラス」で説明します。

C++ の特徴は次のとおりです。

- 比較的複雑で大規模なプログラムの、オブジェクト指向プログラミングに適しています。
- さまざまなクラスライブラリが提供されていて、これを活用することで複雑高度な機能を実現でき、また、文字列操作のような単純なことも C 言語より容易にできます。

- C++ のプログラムの中に C 言語の関数を記述したり、C 言語で記述したモジュールと容易にリンクすることができます。
- C 言語のプログラムと比較すると実行可能コードのサイズが大きくなり、実行時のパフォーマンスも劣る傾向があります。しかし、一般にインタープリタ言語に比べるとかなり高速です。

C++ のプログラムコードの例を次に示します。

```
std::cout << "Hello, C++" << std:endl;
```

これは、演算子 << を使って、「Hello, C++」と出力（表示）して改行するコードです。標準出力（std::cout）に向けて << を使えば文字列を出力できること、std:endl は改行のためのシンボルを示すことがわかれば、その動作は容易に理解できるでしょう。

なお、これはコード断片なので、これだけでは実行できません。実行できるプログラムはあとで示します。

C++ のプログラムコードの例をもう一つ見てみましょう。

```
x = y * 2;
```

これは、y という変数に入っている値を 2 倍して、結果を変数 x に保存する C++ のコードの例です。このコードは C 言語のコードとまったく同じです。つまり、C++ と C 言語はまったく同じ部分があるということです。

◆ C 言語と C++ の関係 ◆

C++ が誕生したころは、C++ は C 言語の一種の拡張と考えられていました。C++ が生まれ育った経緯と、C++ と C 言語との関係を理解しておくと、C++ のプログラミングで役に立ちます。ここでは、C++ と C 言語の関係について概説します。

C++ は、現在のプログラミングにおいて最も重要で強力な言語の一つです。しかし、C++ はオブジェクト指向のプログラミング言語としてゼロから作成されたプログラミン

グ言語ではありません（最初からオブジェクト指向言語として作られたプログラミング言語には、たとえば C# や Java、Smalltalk などがあります）。

初期の C++ は、C 言語の**プリプロセッサ**（前処理プログラム）を使って C++ に導入された機能を処理する言語として作られました。

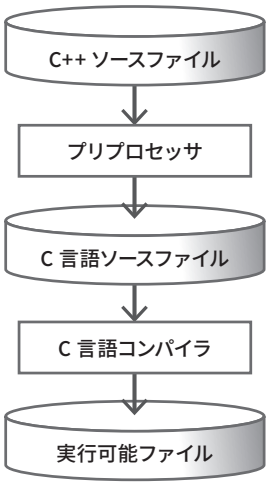

図1.1●初期のC++処理系

そして、現在でもこの考え方は基本的な部分で維持されています。そのため、C++ では次のように C 言語の要素を利用することができます。

- C 言語のキーワード（文やディレクティブなど）は、C++ でもほとんど同じように使うことができます（C++ には C 言語のキーワードに C++ 固有のキーワードが追加されています）。
- 基本的に、C++ のコンパイラは C 言語のプログラムをそのままコンパイルできます。
- クラスに属さないメンバー（変数、定数、関数など）を定義して使うことができます（Java や C# のような完全なオブジェクト指向プログラミング言語では、すべてがいずれかのクラスに属します）。
- 普通、C++ のコンパイラは C 言語のプログラムもコンパイルできます。しかし、逆に、C++ の固有の要素は、通常、C 言語では直接利用できません。つまり、C++ のプログラムの中に C 言語のコードを挿入して C++ のプログラムとしてコンパイルすること

はできますが、C 言語のソースプログラムの中に C++ のコードを混ぜると、C 言語としてコンパイルできません。

C 言語と C++ の関係を図で表すと次のようになります。

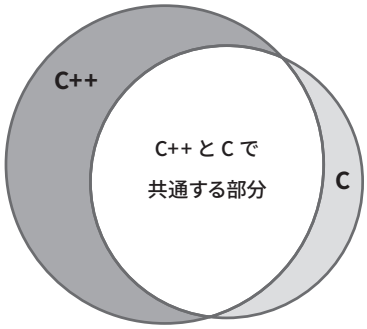

図1.2●C言語とC++の関係

このような C++ の性質は、オブジェクト指向プログラミング言語としては特別な性質といえます。つまり、非オブジェクト指向の C 言語の関数を呼び出すことができたり、C++ のソースの一部に C 言語の形式でプログラムを記述することが可能であるだけでなく、C++ でオブジェクトをまったく使わない手続き型のプログラミングさえ可能です。

◆C 言語と C++ のプログラミングスタイル◆

C++ と C 言語の言語キーワードはほとんど同じです。また、C++ のプログラムの中では C 言語のコードをほとんどそのまま使うことができます。しかし、C++ のプログラミングの方法は、本来、C 言語のプログラミングの方法とは異なります。

端的に言えば、**C 言語のプログラミングとは関数を作ること**です。

C 言語のプログラミングでは、目的を達成するための機能を分割して、それぞれ独立した関数として記述します。たとえば、単純な住所録アプリケーションを作る場合、ユーザーからの入力を受け取るための関数、入力データに矛盾がないかどうか調べる関数、データをファイルに保存する関数、ファイルからデータを探して取り出す関数、特定の住所録レコードを表示する関数、住所録のリストを表示する関数など、さまざまな関数

を記述し、それぞれの関数を適切な機会に呼び出すようにプログラムします。

一方、**C++ のプログラミングとは、端的に言えば、クラスを定義して使うこと**です。

C++ は、オブジェクト指向プログラミングのためのプログラミング言語です。オブジェクト指向のアプローチでは、一連のものの共通する特性を突き止めて分類し（抽象化）、クラスを定義します。たとえば、住所録アプリケーションを作る場合、住所と氏名などからなる個人の情報のクラスと、そのクラスのインスタンス（オブジェクト）に氏名や住所などを保存するための C++ のクラスのメンバー関数（メソッド）を記述します。

C++ ではものごとを抽象化してオブジェクトとしてとらえますが、このような作業は特別なことではありません。日常生活の中でも、我々は意識せずにものの特性を見極めて分類し、クラスとしてモデル化します。たとえば、近所の野良犬から山田さんの家のハチ公まで、犬とみなせる動物をすべて犬として扱って、ほかの動物と区別します。そして、犬は哺乳動物に属するということを知ると、犬を、他の哺乳動物が持つ特性と同じ特性を持つものとして認識します。

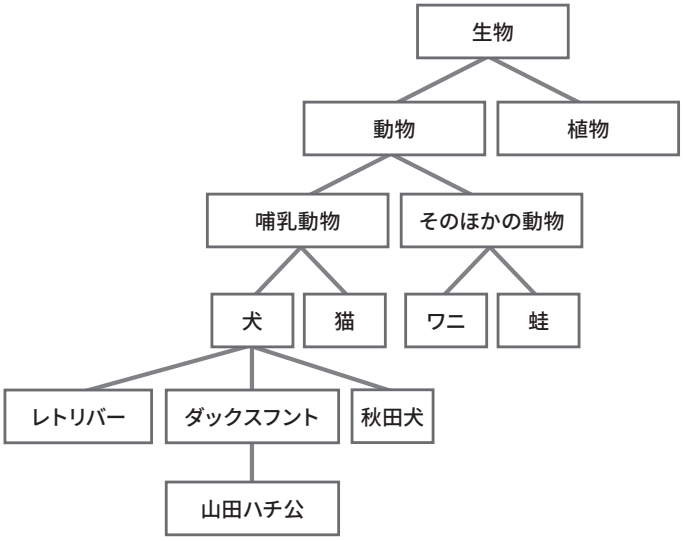

図1.3●オブジェクト指向のアプローチ

プログラミングにおけるオブジェクト指向のアプローチは、このような考え方をプログラミングに導入したものととらえることができます。

オブジェクト指向のプログラミングでは、あらゆることを想定し、よく分析し検討し

て可能な限り現実の事象を論理的に矛盾なく扱いやすいように抽象化して**モデル**を考えます。オブジェクトのモデル化の方法や結果は、目的や対象によって異なります。また、複雑な問題を扱う場合には最初の段階ですべてを完全にとらえることが事実上不可能なため、試行錯誤を伴う作業になることがあるのがオブジェクト指向プログラミングの特徴です。たとえば、図 1.3 では動物を哺乳動物とそれ以外の動物に分けました。実生活においてはこの程度の分類で十分な場合が多いでしょうが、動物を学術的に扱うときには、哺乳類、甲殻類、両生類などのように類で分ける必要があるでしょう。また、目的によっては、「ワンと吠える」、「ニャオと鳴く」、「吠えも鳴きもしない」という基準で分類するほうが合理的である場合もあります。分類のしかたは目的によって異なり、いずれが正解でどれが間違いということはありません。プログラミングでも同じことで、目的に合わせて最適なモデル作りを目指します。

◆ C 言語と C++ の主な違い ◆

C 言語と C++ の主な違いを、以下に示します。これらの意味は本書を読み進めるうちにわかるようになるので、この段階では C 言語と C++ にはいつくかの違いがあるという点に注目してください。

- C++ には new と delete や using のような C 言語にはないキーワードがあります。コンパイラによっては C 言語でも C++ のキーワードが予約されている場合があります（C 言語のプログラムの中で C++ のキーワードを変数名などの識別子として使えない場合があります）。
- C++ では、関数の引数にデフォルト値を指定できます。
- C++ ではシグネチャ（引数の数や型、戻り値の型）が異なる複数の同じ名前の関数を宣言して使うこと（オーバーロード）が可能です。C 言語では同じ名前の関数を作ることはできません。
- C++ では、インライン関数や、仮想関数がサポートされています。
- C++ では、テンプレートがサポートされています。
- C++ では、例外が言語でサポートされています。C 言語ではコンパイラがサポートしている場合があります。
- C++ では typedef 宣言しないで構造体を型として宣言できます。C 言語では typedef

を使って構造体にエイリアスを定義する必要があります。

- C++ では、文ブロックの中でローカルなオブジェクト宣言を行うことができます。たとえば、`for (int i=0;;)` のように for 文の中でもローカルな変数を宣言できます。
- ANSI C++ では、関数をネストできます。
- C++ では、すべての関数は正式にプロトタイプを宣言する必要があります。また、C++ では、型チェックは C 言語より厳しく行われます。

本来、C 言語ではサポートされていない C++ の機能でも、コンパイラによっては C++ の機能の一部が C 言語でサポートされていることがあります。しかし、初歩の段階では、C 言語ではサポートされていないとされている C++ の機能は、C 言語では使えないと考えておくほうが良いでしょう。

◆ いろいろな C 言語と C++ ◆

C 言語は 1972 年に誕生し、それ以来、いろいろな人々がかかわってそれぞれ改良し開発して来た歴史があります。また、C 言語の拡張として誕生した C++ も当初はさまざまな種類がありました。現在は言語の仕様（バージョンと考えても良い）は ANSI、ISO、JIS で標準規格化されていますが、それでも経年とともに細かい部分が変化しています。主要なものを次に示します。

表1.1●C/C++の主な言語仕様

C	C++
ISO/IEC 9899:2018（C18）	ISO/IEC 14882:2020（C++20）
ISO/IEC 9899:2011（C11）	ISO/IEC 14882:2017（C++17）
ISO/IEC 9899:1999（C99）	ISO/IEC 14882:2014（C++14）
	ISO/IEC 14882:2011（C++11）

また、C 言語と C++ の処理系（後述するコンパイラを中心としたプログラムを実行できるようにするソフトウェア）にもさまざまな種類とバージョンがあって、プログラムの書き方も厳密にいえば異なる部分が少しあります。

たとえば、プロセッサ（CPU）には、現在、PC や多くのシステムで良く使われる 64 ビットや 32 ビットのほかに、4 ビット、8 ビット、16 ビット、そして 128 ビットなど、さまざまなものがあります（リモコンや機能を限定した組み込み機器などでは少ないビット数の CPU が採用される傾向があります）。それらの多くは、C 言語（そして C++）でプログラミングできます。これは C 言語 /C++ では特定のハードウェアを想定していないからです。そのため、C 言語（そして C++）コンパイラといっても多種多様なものがあります。たとえば、整数値のサイズは CPU のビット数によって異なり、実数のサイズも異なる上に実数計算が高速でできない CPU さえあるので、特定のコンパイラが扱うことができる数の種類や範囲も異なります。

とはいえ、現在実際に使われている C 言語や C++ 言語の本質的な部分はほとんど同じなので、本書の範囲では違いを意識する必要はありません。いいかえると、本書で示すプログラムはプログラミングを学ぶために使われる一般的な環境で問題なくコンパイルして実行できます（第 12 章の内容は除きます）。

1.2 プログラムの作成・実行手順

ここでは、C 言語と C++ 言語の単純なプログラムを作成して実行する手順を解説します。

◆ プログラム開発の流れ ◆

プログラマーが作成したり編集する C 言語や C++ 言語の**ソースプログラムファイル**は、テキストファイルです。

プログラマーは、最初にエディターや開発ツールを使ってソースファイルを作成し編集します。そして、それを**コンパイル**して**実行可能なプログラムファイル**を作成し、実行します。

ここでは、テキストエディターと C/C++ コンパイラを使って C/C++ プログラムを開発するときの作業の流れとコンパイラの処理の過程を概説します。統合開発環境（IDE）や高機能エディターを使って開発する場合でも、開発やデバッグを効率良く行うために、

この基本的な操作の流れとコンパイルの各過程について知っておく必要があります。

Note　コンパイラと IDE については付録 A「開発環境」で簡単に紹介しています。

ソースプログラムを作成して実行するまでの手順は次のとおりです。

(1) テキストエディターを使って C/C++ のプログラムのソースファイルを作成したり編集する

編集したファイルは、C 言語のプログラムの場合、通常、拡張子が **.c** であるファイル名を付けて保存します。C++ のプログラムは、拡張子を **.cpp** か .cc あるいは .cxx などにして保存します。

(2) ソースファイルをコンパイラでコンパイルする

通常使うコンパイラのコマンドは **gcc**、**g++**（C++）、**cc**（C 言語）、**cl**（Microsoft）などです。

単純なプログラムをコンパイルして実行する場合は、いずれを使っても操作方法や生成されるみかけの結果はほぼ同じです。

IDE を使う場合は、たとえばメニューから「ビルド」というコマンドを選択してコンパイルします。なお、プログラムを正しくコンパイルできない場合は、付録 C「トラブル対策」を参照してください。

コンパイルが問題なく終了すると実行可能ファイルができます。

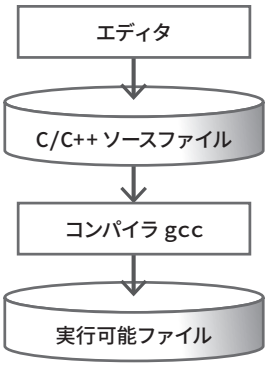

図1.4●C/C++プログラムのコンパイル手順の例

(3) 生成された実行可能ファイルを実行する

生成されたプログラムファイルを実行します。生成される実行可能なプログラムファイルは、Windows でファイル名を指定しない場合にはソースファイル名から拡張子を除いた名前に **.exe** を付けた名前です。UNIX 系 OS では **a.out** です。なお、コンパイルする際に生成する実行可能ファイル名を指定することもできます。

以上が基本的なコンパイルの手順です。このあと、プログラムをテストしてデバッグしますので、通常は手順（1）～（3）を繰り返す必要があります。

Note

make や nmake と Makefile を使って一連の作業を効率良く行うこともできます。

◆ コンパイルの過程 ◆

普通、単に**コンパイル**（または**ビルド**）と呼んでいる、ソースファイルから実行可能ファイルを生成する処理は、さらに細かく分けることができます。

実際に行われるコンパイルの過程は、図 1.5 に示すように、プリプロセス、コンパイル、リンクに分けることができます（図 1.5 は図 1.4 をさらに詳しく説明した図です）。

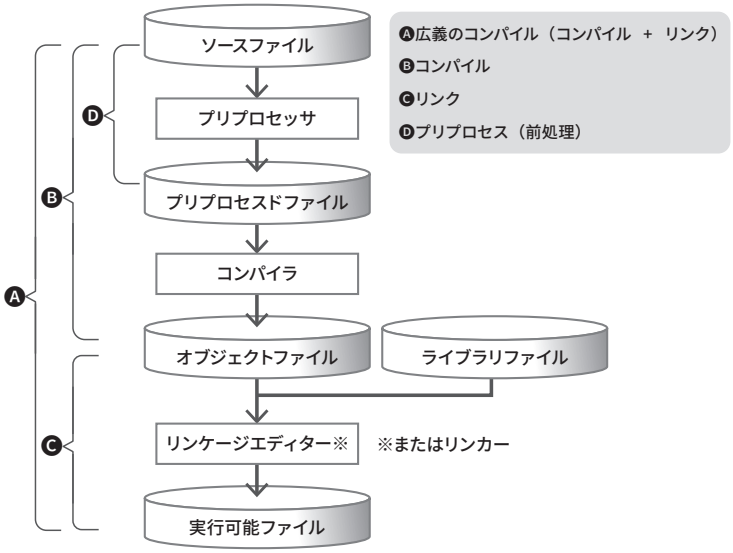

図1.5●C/C++プログラムのコンパイル手順

ソースファイルをコンパイルしてオブジェクトファイルを生成するプログラムを狭義の**コンパイラ**といいます。ソースファイルを前処理（プリコンパイル）する部分だけを、特に**プリプロセッサ**といいます。

オブジェクトファイルをほかのオブジェクトファイルやライブラリとリンクして実行可能ファイルを生成するプログラムを、**リンカー**またはリンケージエディターといいます。

「プログラム開発の流れ」で説明したコンパイルのコマンド（gcc や g++、cl など）は、正確にいえば、コンパイルとリンクを制御するコマンドです。プログラムを単にコンパイルして実行可能ファイルを作成するときには、コンパイルとリンクを制御するコマンドである gcc や g++ だけを使ってコンパイルできます。しかし、複数のソースファイルをコンパイルしてリンクするような場合や、デバッグや効率的なコンパイルオプションの指定などでこのようなコンパイルの詳しい過程を知っておくと役立ちます。

1.3 はじめてのC言語プログラム

ここではC言語の最も基本的なプログラムを作成して実行する方法を示します。プログラムの内容については気にしないで、プログラムを実際に作成してコンパイルし、実行してみましょう。

◆C言語のHelloプログラム◆

ここでは、「hello, C」という文字列を表示して終了する小さなプログラムを作成します。この最も基本的なC言語のプログラムのリストを次に示します。

リスト 1.1 ● hello.c

```
/*
 * hello.c
 */

#include <stdio.h>
```