01 プログラムの基本

Keyword ☑ プログラム ☑ プログラミング言語

プログラムとは

プログラムとは、コンピューターへの命令の集まりです。

学校の先生が「プリントを持ってきて」と生徒に指示した場合を考えてみましょう。先生をプログラマー（プログラムの作成者）、生徒をコンピューターとしたとき、「プリントを持ってきて」という指示がプログラムです。

人間とは違い、コンピューターは曖昧な指示を理解できません。「どのプリントか」「プリントの置き場所はどこか」など、具体的な指示が必要です。

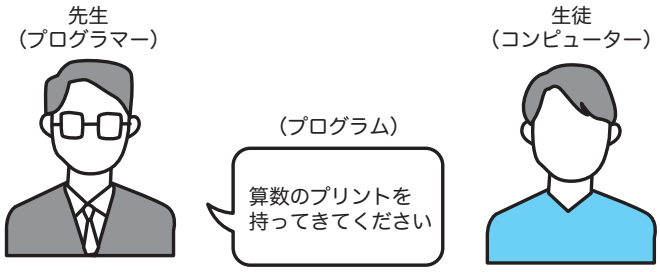
ポイント

コンピューターは、曖昧な指示を自分で判断できない。

私たちが普段使っている日本語などの自然言語は曖昧な表現が多く、コンピューターに的確に指示できません。一方、コンピューターは数字の 0 と 1 だけで表現される**マシン語**という言葉を通じて指示を理解することができますが、人間はマシン語を理解することができません。

C# を始めとした多くのプログラミング言語は、人間が理解しやすい表現で、コンピューターに具体的な指示を与えることができます。そして、コンピューターが唯一理解できるマシン語に変換することもできます（**図 1-1**）。

▼図 1-1　プログラミング言語の役割



ポイント

プログラミング言語は、人間がコンピューターに指示を与えるためのもの。

プログラミング言語で書かれたプログラムは、**コード・ソース**、あるいは**ソースコード**と呼ばれますが、本書では**コード**という表現で統一します。

●簡単なプログラムの例

次のプログラムを目にするまで、「コンピュータープログラミングは非常に難解で、専門家だけのもの」という印象を持っていました。

```
PRINT "新しい言語へようこそ！"
```

これは、私が 14 歳のとき初めて目にしたプログラムで、友人が BASIC というプログラミング言語で記述したものです。

当時プログラミングに関する知識のない私でも、プログラムの実行結果を容易に想像することができました。この経験は、私がコンピュータープログラムに興味を持つ大きなきっかけになりました。

それでは、次の節からプログラミングについての学習を進めていきます。

02 C#の基本

Keyword ☑ C# ☑ 基本ルール

C# とは

本書で解説する C# プログラミング言語は、マイクロソフトが発表した、オブジェクト指向プログラミングを始めとするさまざまな概念を用いて開発することができる「マルチパラダイムプログラミング言語」です。皆さんも C# を習得することで、さまざまな概念を用いて開発できる楽しさを知ることができるでしょう。

プラットフォーム

C# は、.NET Framework および .NET Core というプラットフォームで動作するソフトウェアを開発できます。

.NET Framework は、Windows にインストールすることができるため、Windows で動作するソフトウェアを開発することができます。

.NET Core は、Windows 以外の OS へ移植できるように設計されており、mac OS や Linux などでも利用可能になる予定です。

本書では、.NET Framework を使用して、Windows で学習を進めていきます。

C# の基本的なルール

C# を記述する際の基本的なルールをまとめます。詳細については次の章から少しずつ学習していくので、現時点で理解できなくても、そのまま読み進めてかまいません。まずは、C# プログラムの雰囲気をつかんでください。「←①」や「←②」は入力しません。

```
static void Main(string[] args) ←②
{
   Console.WriteLine("こんにちは C#"); ←①
}
```


●使用する文字の種類

C# で使用する文字は、上記のコードの通り英小文字が基本になります。単語の先頭に英大文字を使うこともあります。

●命令の末尾はセミコロン

命令は、プログラムを実行する最小単位です。①のように、各命令の末尾には「;（セミコロン）」を付けます。

●命令の実行

命令を実行するには、命令を意味するキーワードの後ろに「( )（丸括弧）」を付けます。また、命令に対してデータを渡したい場合には、①のように「( )（丸括弧）」の中にデータを記述します。

「Console.WriteLine」は、コンピューターの画面にデータを表示するための命令です。①では、画面に表示させたい文字列「こんにちは C#」を、「( )（丸括弧）」の中に指定しているため、実行すると「こんにちは C#」というメッセージが表示されます。

●ブロック

命令をグループ化するときは、**ブロック**を使用します。「{」から「}」の範

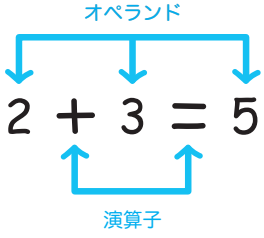
01 演算子の基本

Keyword ☑ オペランド ☑ 優先順位

演算子とは

演算子（**オペレーター**）は、コンピューターに演算を指示するための記号やキーワードです。たとえば、私たちが普段計算するときに使用している「+（**プラス**）」や「-（**マイナス**）」などの記号が演算子です。一方、演算対象となる変数や定数などの値は、**被演算子**（**オペランド**）といいます（**図 4-1**）。

▼図 4-1　演算子とオペランド



演算子を使う場面

演算子は、変数や定数の値を演算するときに使用します。演算という言葉を聞くと、金額や数量の合計を求めたり、平均を求めたりといった、数字の計算を想像するでしょう。しかし、プログラミングにおける演算子は、条件

に応じて実行される処理を変更したり、同じ処理を繰り返し実行させたりなど、プログラムの実行を制御する目的でも使用されます。

本章では、「どのようなときに使用する演算子か」についても学習しますが、まだ学習していない実行制御に関する演算子も多く含まれます。したがって、本章では、演算子の動きについて理解するようにしてください。

メ モ

プログラムの実行制御については、「5 章　条件分岐」や「6 章　繰り返し処理」で詳しく学習します。

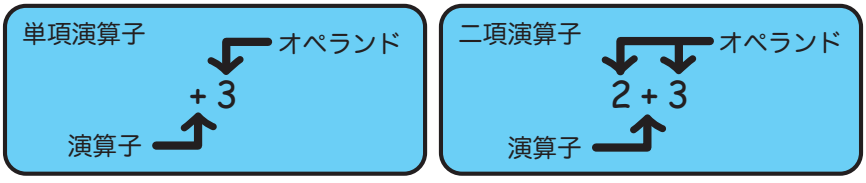
単項演算子・二項演算子・三項演算子

単項演算子は、1 つのオペランドに対して演算を実行します。たとえば、+3 の「+」や -5 の「-」も単項演算子です。

二項演算子は、左右 2 つのオペランドを使用する演算子です。たとえば、「1 + 1」や「2 - 1」のように、一般的な計算の多くは 2 つのオペランドを使用した二項演算子です。**図 4-2** は、単項演算子と二項演算子を表したものです。

この他に、三項演算子という 3 つのオペランドを使用する演算子もあります。三項演算子については、「4-07 その他の演算子」で詳しく学習します。

▼図 4-2　単項演算子と二項演算子のイメージ



この章のまとめ

- 演算子は、コンピューターに演算を指示するための記号やキーワードである
- 代入演算子は、変数に値を代入するときに使用する演算子である
- 算術演算子とは、数値を計算する際に使用する演算子である
- 連結演算子は、文字列を連結するときに使用する演算子である
- 比較演算子は、2 つのオペランドを比較して、true か false を返す演算子である
- 論理演算子は、2 つのブール値を評価した結果をブール値で返す演算子である

章末復習問題

「CS4_A」というプロジェクトを新規に作成し、以下の問題を解いてください。

練習問題 4-1

初期値を 1 とした int 型の変数「a」を宣言し、1 を加算するコードを「単項演算子」「二項演算子」「複合代入演算子」それぞれを使って作成してください。

練習問題 4-2

初期値を 1 とした int 型の変数「b・c」を宣言して、2 つの変数を比較した結果を初期値とする bool 型の変数「d」を宣言してください。

練習問題 4-3

bool 型の変数「e」を宣言してください。初期値として、定数の true と false を使用した、論理 AND 演算子を、ショートサーキット評価で演算した結果で代入してください。

解答は P.580