魔法の
Cプログラミング演習書

―入門から実践まで―

博士(理学)　倉光　君郎　著

コロナ社

ま　え　が　き

本書は，これ一冊解けば，プログラミング入門から実践的なプログラミングまでひととおり習得できる魔法の演習書を目指して書きました。

本書は，まず教える立場から，徹底的に楽できることを目標としています。「はい」と一冊渡して，あとは Web 上にプログラミング情報はあふれているのだから自ら調べながら勉強を進めてほしい，願わくは本当に実用的なプログラムが書けるようになってほしい，という願望を実現できるように編集してあります。そのため，問題を解く上で検索すべき用語を キーワード のように枠で囲って，どういうところに興味をもち関心を広げていくべきか示してあります。どんどん調べて，プログラミングの深淵に迫ってもらいたいと思います。

学ぶ者の立場からすると，丁寧に体系だって教えてくれる教科書がほしいなと思うかもしれません。しかし，そのような教科書では，プログラミングの本質を身に付けるのは難しいです。プログラミングは，習った知識を活用する能力だけでは不十分で，未知の問題にアプローチする「問題解決能力」が必要だからです。ちょっと難しい問題にも挑戦することで，問題解決の姿勢を身に付けてください。本演習書は，理解して進めれば，Web で検索しても，友達に聞いても構いません。立ち止まらずに，問題を解決し，前に進むことが大切です。

本書の執筆にあたり，つぎのような点に配慮しながら，演習問題を集めてみました。

- 基礎から実践力まで，無理なくステップアップできる
- （問題数を厳選し）一題一題，新しい知見が得られる内容
- プログラミングへの興味が高まるように幅広いテーマ

本書の問題は，著者の勤務校である横浜国立大学の「プログラミング入門」や「プログラミング演習」で出題してきた過去問をベースに出題しております。解説は，学生たちが間違いやすかった点を踏まえながら，講義の語り口で解説してみました。コーディングスタイルは，著者のオープンソース開発の経験を生かし，読みやすくバグの少ないコードを書く習慣が身に付くように心掛けました。また，先に解説を読んでもあまり実害がないように，中盤以降はできるだけソースコード全文の掲載は避けてあります。

もしソースコード全文を参考にしたい場合は，著者の GitHub レポジトリを探してください。

https://github.com/kkuramitsu/book/

この一冊を仕上げて，プログラミングの楽しさを学んでもらいたいと思います。

本書の執筆にあたっては多くの方にお世話になりました。その中でも，特に菅谷みどり氏，そして学生目線で草稿を読んでくれた千田忠賢氏，大川貴一氏に感謝します。またコロナ社には企画段階からたいへんにお世話になりました。本書で利用しているソフトウェアや技術的な内容は日ごろの研究活動のたまものです。研究室のメンバーに感謝します。最後に，こうして本を書いていられるのも妻雅子の協力があってこそです。ありがとう。

2016 年 12 月

倉光　君郎

問題の難易度

問題の難易度を表すレベルとして，各問題には★が付いております。難易度の参考にしながら，★を集める要領で問題を解いていってください。

【★】	言語文法に関する基礎問題
【★★】	初心者レベル
【★★★】	初心者卒業レベル
【★★★★】	情報系学科で標準的に期待されるレベル
【★★★★★】	挑戦者向け。解けなくても心配しなくてもよい問題

まずは目安として，★を 50 個以上集めることを目指してください。「プログラミング入門」は完了です。★を 100 個以上集めたら，どこに行っても恥ずかしくない基礎力が身に付いているでしょう。

目　　次

1. 最初の一歩

2. ウォーミングアップ

3. 配 列

4. ポ イ ン タ

5. C を超える

6. システムプログラミング

7. データ構造とアルゴリズム

8. 仕上げの問題

1 最初の一歩

C Programming

この章は，プログラミング初学者向けの基本文法を確認しながらのスタートアップです。最初のうちは，他人の書いたコードを参考にする機会も多いと思います。プログラムが動いたら満足ではなく，好奇心をもってここを変更したらどうなるだろうと試してみてください。

1.1 Hello, world

演習室で課題を解くときは，教員があらかじめプログラミング環境を用意してくれています。しかし，演習室を一歩出たら，誰かがプログラミング環境を整備してくれることはありえません。ということは，自分自身で，自分のパソコンに環境設定できなければ，頑張って勉強した知識やスキルはどこで使うのでしょうか？まずは，Web の情報を頼りに是非 C プログラミング環境設定 を試してみてください。

この問題はプログラミングを始める前段階に難しさがあるので，いきなり涙の★★★です。

★★★

問題 1.1: （自分のパソコンにプログラミング環境を設定し）C プログラムを書いて，`hello,world` と表示せよ。

【ヒント】 まずは内容を気にせず，つぎのコードを一字一句正確に入力してみよう

```
#include<stdio.h>
int main() {
   printf("hello,world\n");
   return 0;
}
```

【難度 Up! (★★★★)】 コンパイラは Clang がオススメです。

解　　説

Hello, world プログラムは，C 言語の開発者 K & R（Kernighan and Ritchie）による『プログラミング言語 C』（原題：The C Programming Language）に登場する最初の C プ

ログラムです。あまりに有名なプログラムなので，プログラミング環境（C コンパイラやエディタ）の動作確認をするテストプログラムの代名詞になっています。

ここでは，標準的な Unix 開発環境が設定できたという想定で進めます。コンパイラは Clang を想定しています。

まず，vim などのエディタを使って，ヒントのコードを入力し，`hello.c` というファイル名に保存してください。

Hello, world プログラムでは，あまりコードの内容を考えずに動かしてみることが重要です。ただし，内容が気になる読者のために簡単に解説を付けると：

- 関数 `main`() は，プログラムが起動したとき，最初に実行される関数です。最後の `return 0` は，プログラムが正常に完了したことを表す状態です（この辺りは，C 言語というより，オペレーティングシステムの決まり事です）
- 関数 `printf`() は，書式付きで文字列表示をするライブラリ関数です。この関数を利用するため，標準入出力（standard input/output）ライブラリ `stdio.h` を事前に include しています

C 言語は，コンパイラ型のプログラミング言語です。コンパイル＆ビルドという手続きを踏んで，実行可能なプログラムを得ることができます。

標準的な Unix の開発環境であれば，`hello.c` は，つぎのとおり `make` コマンド一発でコンパイル＆ビルドできます。ターミナルのプロンプトから入力してみてください。

```
$ make hello
cc     hello.c   -o hello
```

コンパイル＆ビルドに成功すると，新しく `hello` コマンドが作成されます。同じく，Unix ターミナルのプロンプト`$` から実行し結果が得られます。

```
$ ./hello
hello,world
```

構 文 エ ラ ー

もしコンパイル＆ビルドに失敗すると，（それはソースコードが文法的に正しくなかった場合ですが，）**エラーメッセージ**（error）が表示されます。`error` というキーワードには注意してください。つぎは，エラーメッセージの例です。

```
$ make hello
cc     hello.c   -o hello
hello.c:3:25: error: expected ';' after expression
        printf("hello,world\n")
```

```
                                    ^
                                    ;
1 error generated.
make: *** [hello] Error 1
```

エラーメッセージは，コードの上の問題箇所を行番号，例えば `hello.c:3` のように指摘してくれます。英語でメッセージが表示されたとしても，無視しないでください。もしエラーメッセージの意味が見当も付かない場合は，error: expected ';' after expression のように，そのまま検索すると，ヒントが得られることも多いです（ちなみに，上のエラーは文末の ; の打ち忘れによるものです）。エラーを修正したら，再度，コンパイル&ビルドし，エラーメッセージが出力されなくなるまで繰り返します。

もちろん，「vim や make を使うのはしんどい」という人もいるでしょう。そういう場合は，統合開発環境 IDE を用いて演習を行っても構いません。 よりプログラミングしやすい環境を整えるのも重要なプログラミング能力の一つです。

1.2 表 示 と 書 式

C プログラミングの学習において，まず初めに慣れておきたいのは，`printf` 関数と書式です。これらは，プログラムの実行結果を表示するだけでなく，プログラムが期待どおり動作しない場合，原因を探すのに使うからです。

★

問題 1.2: つぎは，変数 `x` に対して，キーボードから入力した整数値を読み込み，その整数値を表示するコードである。

```
#include<stdio.h>
int main() {
   int x = 0;
   printf("x=");
   scanf("%d", &x);
   printf("x=%d\n", x);
   return 0;
}
```

このコードを修正することで，変数 `x` の値を 10 進数表記と 16 進数表記で 1 行に表示するようにせよ。

【ヒント】 printf 書式

【例】 `x=255` なら `x=255, FF`

【難度 Up! (★★)】 整数の代わりに，実数 0.12345 を入力したら，`0.1235 1.23e-01 12.35%`と表示する

解　　説

C 言語の初学者には，多少，意味が理解できない記法があっても「決まり文句」だと思って，前に進まないといけないところがあります。ポインタを理解するまでは，`scanf("%d", &x)` は，ユーザの数値入力を変数 `x` に読み込むときの「決まり文句」としてください。

関数 `printf()` は，Hello, world プログラムでも使ったとおり，`" ... "`で囲まれた文字列[†] を 標準出力 (`stdout`) に出力するライブラリ関数です。文字列の中で`%`で始まる書式 (format) を書くと，書式に従って数値をフォーマットして表示してくれます。なぜ書式機能があるのかといえば，整数値はコンピュータ内部ではビット列 (2 の補数表現) で表現されており，人間が読むためには 10 進数や 16 進数の表現に変換する必要があるからです。書式は，その変換の役割を担当しています。

整数の 10 進数表記は`%d`，16 進数表記は`%x` となります。問題は，「10 進数表記と 16 進数表記で 1 行に出力する」ということなので，改行 (`\n`) の前に 2 種類の書式を書いて，変数 `x` を 2 回，それぞれの書式で表示できるようにします。

```
printf("x=%d %x\n", x, x);
```

改行 (`\n`) は文字列の中で特別な文字コードを表す エスケープシーケンス と呼ばれる記号です。改行以外に，ダブルクオート（`\"`），バッククオート（`\\`）など， 文字列中にそのまま表現できない文字を表すのに使います。書式と併せて，エスケープシーケンスも調べておきましょう。

浮動小数点数の出力

課題★★は，実数の入力と出力です。ただし，C 言語では，厳密な意味で，実数を扱うことはできません。実数を近似したデータ型として，単精度浮動小数点数 `float` 型 と倍精度浮動小数点数 `double` 型 が用意されています。特に理由がないかぎり，精度の優れた `double` 型 を使います。`double` 型 の変数にユーザ入力を読むときは，`scanf` の書式は`%d` の代わりに`%lf` を使います。

```
double x = 0.0;
scanf("%lf", &x);
```

[†] 文字列は，文字が並んだデータで，ポインタ操作を覚えると，整数値と同じくプログラムから操作できます。

これも，標準入力から double 型の変数を読み込む「決まり文句」としてください（混乱しやすい話ですが，あとから述べる関数 printf() の書式とは別物です。%f ではありません）。

double 型の printf() の書式は，%f と%e の 2 種あります。それぞれ小数点表記，指数表記を意味します。また，オプションで桁数を指定することができます。例えば，つぎのようにすると，それぞれ小数点以下 4 桁，2 桁の指定となります。

```
%.4f    %.2e
```

百分率表記は，専用の書式がありません。表示する前に，(x * 100) のように 100 倍して，百分率に変換します。最後の単位の%は，検索力を試すような意地悪な問題でしたが，%%という特殊な書式を使います。結構，\%と間違えてしまう箇所です。

さて，まだ★★問題なので，ソースコード全文を掲載しておきましょう。

```
#include<stdio.h>
int main() {
   double x = 0.0;
   printf("x=");
   scanf("%lf", &x);
   printf("x=%.4f %.2e %3.2f%%\n", x, x, (100*x));
   return 0;
}
```

print() の書式は，C 言語の主要な型ごとに用意されています。新しいデータ型を学んだら，書式を確認して正しく表示できるようにしましょう。

1.3 変 数 と 代 入

数学は得意だけど，どうもプログラミングは苦手という学生は少なくありません。C 言語は，数式の記法は数学を基盤にしていますが，数学とは本質的に異なる世界です。両者の違いを理解し，発想を切り替えることが必要です。

★

問題 1.3: $n = n + 1$ は，数式としては（矛盾しているので）成り立たないが，プログラムとしてはある正しい操作であることをプログラム実行例と共に示せ。

解　　説

C 言語では，演算子=は「代入（assignment）」と呼ばれる操作をする演算子です。つぎの式は，「変数 n に対し，変数 n+1 の結果を代入する」という意味です。

```
   n = n + 1
```

索　　引