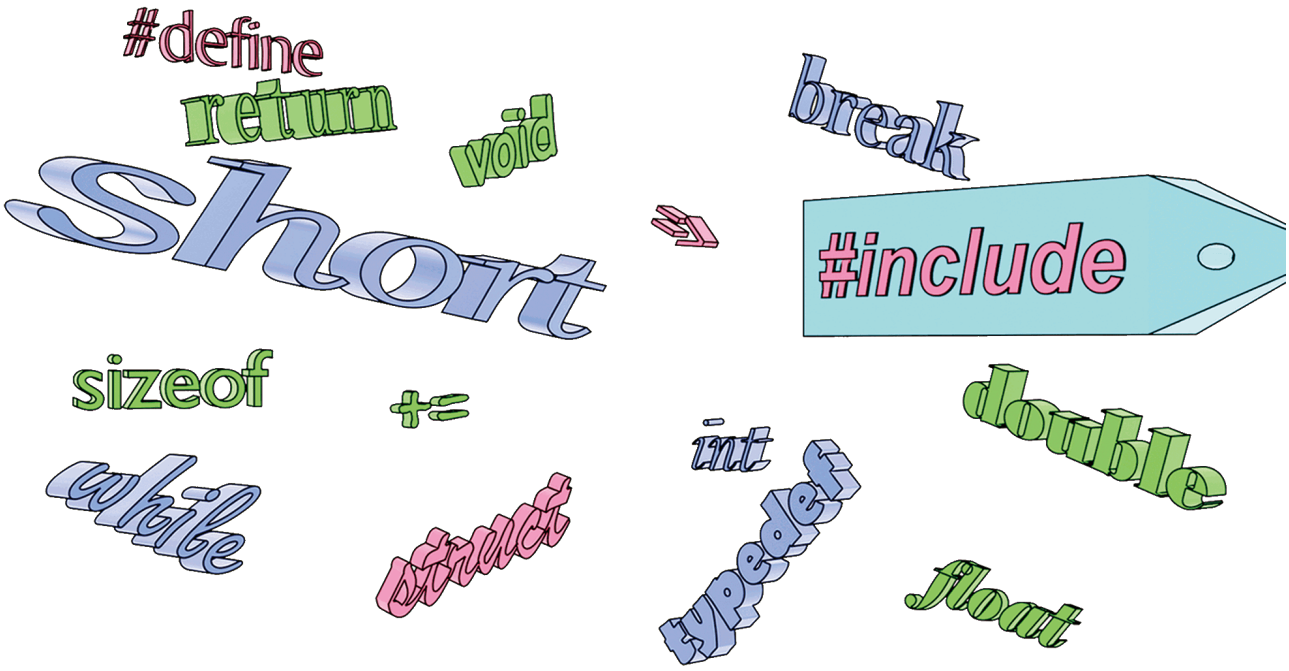
キックオフ C 言語



関西学院大学 情報・機械系 実験準備室

2024-09-03 10:06:43

まえがき

本書は、10 年以上にわたるプログラミング実習の授業支援の経験から生まれました。通常の入門書には書かれないようなことまで書いています。

通常の C 言語の入門書であれば、文法の説明から始まって、正しいプログラムの書き方を説明してあります。しかし、コンパイルエラーの読み方や、「最初のエラーが重要」との教えまで言及したものはなかなか見つかりません。

授業でつまずいている人に出会うと、コンパイルエラーの意味がわからなかったり、大量のエラーメッセージのどこから対処してよいかわからなかったり、あるいは止めたつもりのプロセスが動き続けてファイルを書き込みロックしてコンパイルができなくなっている、などということがよくあります。本書ではそのような人にも役立つよう、実習授業のレジメに書いてありそうな、現実問題への対処方法も記載しました。

またサンプルコードには、役に立たない動作ではなく、単体として見ても意味のある動作をするように心がけました。1 つの動作を実現するのにも実装方法は何通りもありますから、段階を踏んで改良していくスタイルを積極的に採用しました。

執筆陣は、著名なオープンソースソフトウェアの開発に携わったり、世界規模のプログラミングコンテストで好成績を修めた人ばかりです。このような経験も生かして、命名の習慣や実際のコーディング現場での現実的な対処方法などにも踏み込んで記述しました。現実のコーディング現場では絶対に使われない、関数ブロック内のプロトタイプ宣言や構造体宣言などは、サンプルコードからも排除しました。

昨今のスマートフォンも含めたコンピュータ環境を鑑みると、いま C 言語を学んだからといって、将来も C 言語でソフトウェア開発をするとは、必ずしも言えません。したがって C 言語に深く精通することを目標とはせず、**コンピュータ言語一般に通用する知識**を身につけることを目標とし、同じような事象にも、言語が違うとどのように取り扱われ方が変わるのか、注釈を加えました。

プログラミング学習の指導法として、「文法を学んだら、後は自分で考えなさい」というのもありますが、本書を執筆しているうちに、それでは上達しない、少なくともこの方針で育てられた人材が社会で活動するには悪影響があると、一層強く感じるようになりま

した。一人で仙人のように、他人のサンプルも読まずにプログラムを作るのならともかく、チームでの共同作業を想定すると、語彙はまわりの人と合わせる必要があります。つまりソースコードは、プログラマにとって会話の媒体です。上達の近道は「**いろいろな人の（できれば上質の）プログラムをたくさん読む**」ことです。多くの人が同じような処理を何度も書くのには、それ相応の理由があるので、理由を完全に理解せずとも、まずはその**慣用句**を覚えるのが、会話を成立させる近道です。独創的な、他人に理解されないすばらしい処理を思いつくよりも重要なことです。これは英語のような、外国語の習得にも似ているように思います。いくら文法上は正しくても、ネイティブならその言い回しはしない、となればそれまでです。考えてわかることではありません。

この考えに至ったのは、執筆中の経験のおかげです。自分ならこう書くけれども、初学者なら違う書き方をするかもしれない、しかし世の中では見たことがない、という処理がいくつもありました。調べてみると、初学者の書きそうなものには問題があって、自分はそこまで理解せずに、**慣用句**を使っていたために問題を避けていたのだとわかりました。つまり、文法だけから正しい書き方にたどりつくには、とても初学者には説明しきれない、膨大な知識が必要だったのです。この時になって、ようやく慣用句のありがたみが理解できました。

本書が、初学者の人にとって、よい慣用句に出会う機会となり、またプログラマとしてのよい心構えを身につけてもらうきっかけとなれば、望外の喜びです。そして本書以外の書物にも触れて、よりよいプログラミング技術を身につけていただければと思います。

本書を執筆するにあたっては、多彩な関係者のご協力をいただきました。学生実験のスタッフ、特に黒澤 隆之さんには、まったく頭が上がりません。イラスト作成には、理工学部 情報科学科 巳波研究室の高木美紀さん、スライド作成には、同研究室の犬童梨子さん、矢野皓己さん、大島由裕さんにご協力いただきました。そして巳波研究室の卒業生の高野歩路さん、大学院生の三柳海渡さん、大崎研究室の南口宙太さん、授業科目（プログラミング実習 I）のティーチングアシスタントの酒井一徳さん、杉本祐貴さん、梅林立さん、数理科学科助教の陰山真矢さんには、数々の助言をいただきました。三田市の飲食店「煮こみや りん。」と「酒楽スタンド にこいち」では、構想を練らせていただいただけでなく、東野弘志さん、東野朝海さん、高田学さんをはじめとする 常連客の方にも話題にしていただき、また雑談の中からよい例題を思いついたこともありました。この場を借りて厚く御礼申し上げます。

2025 年ドラフト春
情報・機械系 学生実験準備室

本書の位置づけ

本書の読者は、理系の大学在籍者で、初めて C 言語を学ぶ人を想定しています。大学の初等数学の知識を使って説明することがあります。

C 言語の機能を網羅的に紹介するわけではありません。プログラミングというものの考え方を伝えるために題材を絞ります。例えば以下のものは取り上げません。

- 正しく使うことの難しい goto 文，代替できる場面の多い switch 文
- ビット演算，ビットフィールド，共用体，列挙型
- 再帰呼び出し，分割コンパイル，メモリの動的確保

習得目標は、駐車場の自動精算機を模倣することです。そのために必要な機能を述べます。特に**問題分割の習慣**を身につけてもらうため、関数を早い段階で説明しました。

入門書では触れられることの少ない、（int 型を流用した）**論理型**の関数や変数にも言及しました。これは**教えられなければ使い方を誤る**代表格ですので、使用例を例題や練習問題にも取り入れました。ほかにも各場面で、初心者にありがちな間違いを指摘しました。

採用する C 言語規格

C 言語の**言語規格** (programming language standard) は、（初期を除くと）米国国家規格協会 (ANSI) や国際標準化機構 (ISO) によって、約 12 年毎に改訂され、以下の略称で呼ばれています。日本産業規格 (JIS X 3010) にも同等のものが取り込まれています。

K&R（1972 年頃）旧スタイルの関数定義，関数の引数の型チェックなし

C89/C90 (**ANSI C**)（ANSI 1989 年，ISO 1990 年）新スタイルの関数定義，プロトタイプ宣言による引数の型チェック，void 型や列挙型 (enum) の導入

C99（1999 年 [6][*1]）変数定義がブロックの途中でもよい，1 行コメント (//)，可変長配列，論理型 (bool) や複素数型，インライン関数指示子の導入

C11（2011 年 [7]）gets() の廃止などによる脆弱性対処，型による分岐 (Generic) の導入，可変長配列と複素数をオプションに格下げ

C23（2024 年出版予定 [8]）旧スタイルの関数定義の廃止，2 の補数表現が必須，2 進数リテラル, 10 進浮動小数点，POSIX 関数 (str(n)dup, memccpy) の導入

本書では、他の言語との親和性も考慮して、C99 を採用します[*2]。

[*1] ISO 規格書の正式版は有償ですが、（ほとんど差のない）ドラフト版は無償で公開されています。

[*2] コンパイラの準拠する言語規格は、バージョンによっても異なりますが、オプションで指定できる場合もあります。例えば GCC なら `"gcc␣-std=gnu99␣source.c"` のように `-std=gnu99` で C99 準拠になります。（☞**??**節）

想定する開発環境

本書では、テキストエディタとコマンドラインのコンパイラを前提に解説します[*3]。想定する**コンパイラ**は次のようなものですが、これ以外のものでも大丈夫です。

- GNU Compiler Collection (GCC)：
  Linux を含む Unix 系と、Windows の Cygwin，MinGW-w64 (MSYS2)，WSL2
- Clang：Linux を含む Unix 系と、Mac の Xcode
- Microsoft Visual C++：Windows

仮想ターミナルは、Windows 10 までの「Windows コンソールホスト」はコピー＆ペーストの操作が煩雑なので、MSYS2 では mintty を、WSL2 では wsltty を追加導入するのがお薦めでした。Windows 11 からの「Windows ターミナル」は良くなりました。

シェルは、Windows では PowerShell は癖が強いので、コマンドプロンプト（cmd.exe）か、できれば bash がいいでしょう。その他の環境では、標準のもので十分です。

テキストエディタは、どれでも構いませんが、さすがに Windows のメモ帳では機能不足、予約語強調表示や自動整形くらいの機能は必要です。Microsoft 社の Visual Studio Code (VSCode) は、OS に依存せず、支援機能が充実しているので、よい選択肢です。

Cygwin のインストール

Cygwin は Windows 上に Linux 環境を実現することを目指したソフトウェアです。GCC をはじめ、make, diff, git など様々なツールが用意されています。

自作プログラム（*.exe）が動作するのに Cygwin 環境（Cygwin DLL）が必要なため、配布目的の開発には工夫が必要です。

- https://www.cygwin.com/ からインストーラ "setup-x86_64.exe" をダウンロードして、実行します。
- gcc はディフォルトではインストールされないので、インストール中にパッケージを追加します。“Devel” カテゴリの “gcc-core” パッケージです。
- その時点での最新バージョンがインストールされるので、作業時期によって環境が微妙に異なります。
- Cygwin64 Terminal の初期フォルダは、Windows の個人ファイル置き場（ドキュメントフォルダ）とは異なり、C:\cygwin64\home*userid* のような場所になっています。環境変数の HOME でカスタマイズ可能です。

[*3] 統合開発環境の利便性は高いですが、初学者には、その裏で動いているプリミティブな機構を理解してもらうことも重要だと考えています。

MinGW-w64 (MSYS2) のインストール

MinGW-w64 は Windows 環境で GCC を動かすためのプロジェクトです。配布形態がいくつもありますが、MSYS2 という、ソフトウェアの集合体から入手するのがよいでしょう。

MSYS2 は Cygwin からの派生物です。Cygwin と異なって、自作プログラム（*.exe）が単独で動作します。

- https://www.msys2.org/ からインストーラ "msys2-x86_64-*yyyymmdd*.exe" をダウンロードして、実行します。（これには gcc は含まれません。）
- インストール後に、UCRT64 のアイコンでターミナルを起動して、次を実行します。 `pacman -S mingw-w64-ucrt-x86_64-gcc` これで追加されるコンパイラが MinGW-w64 によるものです。
- 最新バージョンがインストールされることや、ターミナルの初期フォルダが Windows の個人ファイル置き場と異なることは、Cygwin と共通です。

Linux (Ubuntu) 環境の構築

USB メモリから起動する Linux があります。Linux にはいくつもの銘柄がありますが、例えば Ubuntu のインストーラの Live 機能を使えば、インストールせずに（つまり Windows には影響を与えずに）、そのまま Linux 環境が動きます。

- https://ubuntu.com/download/desktop から Ubuntu Desktop (LTS) インストーラの ISO イメージをダウンロードして、Rufus（https://rufus.ie/）で、「保存領域」を数 GB 確保して[*4]、USB メモリに書き込みます。
- USB メモリを挿した状態で PC を起動し、F12 などのキー（PC のメーカーによって異なります）を押して、一時的に起動ドライブを USB メモリに変更します。
- Ubuntu のインストーラが起動したら「インストール」ではなく「試す」を選択します。そしてターミナルを開き、次を実行します。 `sudo apt-get install gcc` でコンパイラの追加、 `sudo snap install --classic code` [*5]で VSCode のインストールです。

USB メモリの空き領域に、追加インストールしたソフトや、自作プログラムが保存されるので、USB メモリ 1 本ですべての環境を持ち歩けます。

[*4] これで Ubuntu でのファイル操作が USB メモリに保存されるようになります。USB メモリは 16GB 以上で、書き込み速度の早いもののほうが快適です。

[*5] `--classic` のオプションは将来不要になると思われます。

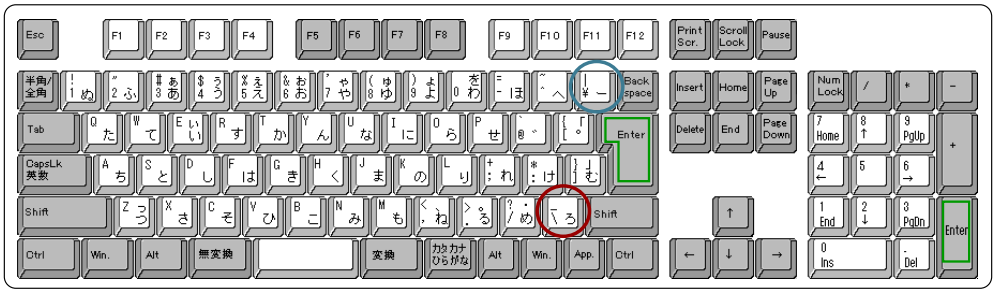
コラム： Windows でのプログラミング環境構築

1. Windows の初回セットアップ時にユーザ名の入力が求められ、**氏名を漢字で**入力しがちです。このユーザ名が、個人ファイル置き場の「フォルダ名」に流用される[a]場合があります。フォルダ名に**漢字**や**スペース文字**が含まれていると、動作のおかしくなるソフトがあるので、避けたいところです。ただし、このフォルダ名やユーザ名を**後から変更**すると、様々な機能に影響があって、これも動作不良を引き起こします。アルファベットのみの**ユーザを新規に作る**のが安全です。
2. 一般に、インストール作業には管理者権限が必要です。一般ユーザであれば、管理者ユーザでログオンしなおして作業します。さらに、右クリックメニューの「管理者として実行」まで必要な場合もあります。
3. **ウイルス対策ソフトの誤判定**で、インストールに失敗する事例を数多く目にしています。ソフトによっては、報告もせずに動作に必要な **DLL を削除**します。信頼できるインストーラを使っているのなら、「リアルタイム検出」機能は一時的に停止するのが自衛手段です。インストール時間の短縮（数分の 1）にもなります。
4. 拡張子の表示機能が、初期設定では OFF になっています。プログラミングでは何度もファイル形式を変換して、拡張子違いのファイルを何通りも扱いますので、必ず**拡張子表示を ON** にしておきます。
5. プログラミングと直接の関係はありませんが、BitLocker という暗号化の機能が、意図せず起動ドライブで有効になっているのを散見します。この機能は、パソコン本体を紛失した際には情報漏洩の危険を低減しますが、ちょっとしたことで Windows 自体が起動しなくなったり、ハードウェア故障時のデータ救出が困難になる[b]副作用もあります。取り扱うデータの機密性が低いなら、無効化を検討すべきでしょう。

[a] ローカルアカウントの場合です。Microsoft アカウントならメールアドレスから生成されるようです。

[b] 回復キーを保存していなければ、データ救出は原理的に不可能です。

図 1　日本語キーボード（ODGA109A）のキー配列



バックスラッシュ・円記号

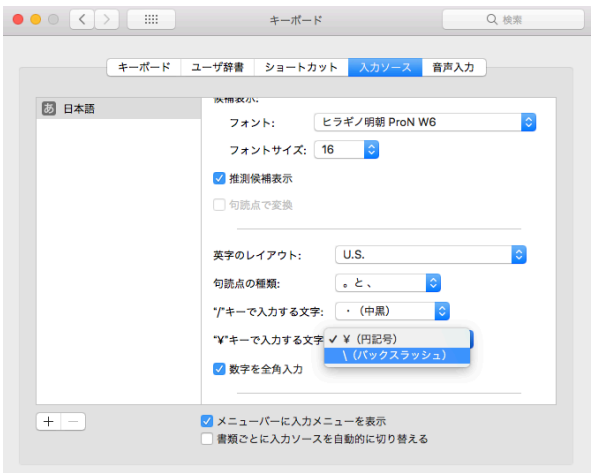
本書では、頻繁に \（バックスラッシュ）を記載しています。この文字は、日本ではこれまで ¥（円記号）に置き換えて使われてきました。そのため図 1 のように、日本語キーボードには \ と ¥ の両方のキーがあるのに、日本語 Windows 環境では区別されず、どちらを入力しても円記号が表示されてきました[*6]。

ところが Unicode が普及して、この 2 文字が区別できるようになりました。したがって、どちらで入力するかは、環境（OS だけでなく、アプリや設定）によって異なります。

区別できる場合　バックスラッシュを入力してください。

区別できない場合　どちらで入力しても構いません。円記号が表示されても、バックスラッシュに読み替えてください。

Mac の日本語環境では、2 文字を区別するので、バックスラッシュを入力する必要があります。1 文字だけなら option と同時に ¥ を押せばよいのですが、¥ で常にバックスラッシュにするには設定変更が必要です。「システム環境設定」→「キーボード」→「入力ソース」→「日本語」と進んで、「"￥"キーで入力する文字」をバックスラッシュにしてください。



[*6] 英語キーボード（101 キーボード/US 配列）には円記号がありませんが、エンターキーの上にあるバックスラッシュを入力しても円記号が表示されたことでしょう。

エンターキー・リターンキー

キーボードの刻印に関して、もうひとつ話題があります。エンターキーとリターンキーの違いです。こちらは単純で、キーボードによって「Enter」と「↵」の 2 通りの刻印があるというだけで、区別の必要はありません。

本書では Enter と表記します。よく使われるキーなので、フルキー部分とテンキー部分の 2 ヶ所に用意されています。形状や位置も特徴的ですから、多くの人が刻印を気にしていないのでしょう。

文字の表記

キーボードから入力する文字は、細かな違いが問題になりがちです。本文中では、下記のように、黄色地のタイプライター書体にします。

```
gcc␣-Wall␣-O2␣source.c
```

- 通常のスペース文字は（前後の文字の間隔が広がるだけで）表記されませんが、上記のように、入力することが重要な場面では、「␣」と下線がついたように表記します。
- 数字の 0（ゼロ）は、タイプライター書体では「0」と、斜線が入ります。アルファベット大文字の O（オー）の「O」と区別するためです。上の例の -O2 は「マイナス・オー・に」です。

ソースコード（プログラム）には、以下のような装飾をつけます。この中に説明を書き込みますが、OK と NG は文法上の間違いかどうかを、○と×は習慣上の良し悪しを表します。

```
  int a[3] = {1,2,3};  // OK
//int b[3] = a;        // NG 文法上の間違い
  for (int i=0; i<10; i++) { /* 10回ループ */ }; // ○ 良い
  for (int i=0; i<=9; i++) { /* 10回ループ */ }; // × 悪い
```

図書の推薦

学習前や学習中に並列して読むと役立ちそうなものを挙げておきます。

- 「**C プログラミング入門以前**」[14] は、当たり前としてなかなか説明されないことを、言語化してあります。ざっと目を通すだけでも価値があります。
- 「**C の絵本**」[9] は、イラストが多く、特に初学者には読みやすいと思います。
- 「**やさしい C**」[12] は簡潔な説明で、網羅的に内容を取り上げています。
- 「**C 言語 [完全] 入門**」[13] は、ページ数は多いですが、説明が丁寧で、紙面も読みやすいです。内容も新しく、C99 はもちろん、C23 まで言及があります。

また、本書を一通り読み終えたら、次のような書物も参考になるでしょう。

- 「**C 言語による標準アルゴリズム事典**」[11] は、文法の次に学ぶべき、アルゴリズムの数々が簡潔にまとめられています。
- 「**リーダブルコード**」[1] と「**プログラミング作法**」[4] は、いずれもどのコンピュータ言語にも共通する「良い習慣」を説明した名著です。
- 「**組込みソフトウェア開発向け コーディング作法ガイド：C 言語版**」[10] は、PDF でも無償で公開されていて、組み込みソフトウェア分野に限らず、参考になる作法が紹介されています。

言語規格を知るには、次の書物が（難解ですが）有用です。

- 最初の C 言語の本である「**プログラミング言語 C**」[2]（通称 K&R）は、C 言語の作者であるカーニハンとリッチーによる C 言語の解説書で、通称は二人のイニシャルから来ています。文法からライブラリとして提供される機能の実装例まで、簡潔に説明されているバイブルです。同書の日本語訳 [3] とともにベストセラーですが、残念ながら内容は少々古く、C89 で止まっています。
- 「**C リファレンスマニュアル**」[5] は、C99 に対応した、言語規格の解説書です。

参考文献

[1] D. Boswell (原著), T. Foucher (原著), 角 征典 (翻訳). **リーダブルコード**. オライリージャパン, 2012.

[2] B. W. Kernighan and D. M. Ritchie. *The C Programming Language, 2nd edition*. Prentice-Hall, 1988.

[3] B. W. Kernighan (原著), D. M. Ritchie (原著), 石田 晴久 (翻訳). **プログラミング言語 C ANSI 規格準拠 第 2 版（訳書訂正）**. 共立出版, 1994.

[4] B. W. Kernighan (原著), R. Pike (原著), 福崎 俊博 (翻訳). **プログラミング作法**. KADOKAWA, 2017.

[5] Samuel P. Harbison, 3 (原著), Jr. Steele,Guy L. (原著), 玉井 浩 (翻訳). **Ｓ・Ｐ・ハービソン３世とＧ・Ｌ・スティール・ジュニアの *C* リファレンスマニュアル**. エスアイビーアクセス, 2008.

[6] WG14/N1256. *Programming languages — C*. https://www.open-std.org/JTC1/SC22/WG14/www/docs/n1256.pdf, September 2007.

[7] WG14/N1570. *Programming languages — C*. https://www.open-std.org/JTC1/SC22/WG14/www/docs/n1570.pdf, April 2011.

[8] WG14/N3047. *Programming languages — C*. https://www.open-std.org/JTC1/SC22/WG14/www/docs/n3096.pdf, April 2023.

[9] 株式会社アンク. **C の絵本 第 2 版 C 言語が好きになる新しい 9 つの扉**. 翔泳社, 2016.

[10] 情報処理推進機構ソフトウェア高信頼化センター. **組込みソフトウェア開発向けコーディング作法ガイド: C 言語版 : ESCR ver. 3.0**. SEC books. 情報処理推進機構 https://www.ipa.go.jp/sec/publish/tn18-004.html, 2018.

[11] 奥村 晴彦. **[改訂新版]C 言語による標準アルゴリズム事典**. 技術評論社, 2018.

[12] 高橋 麻奈. **やさしい C 第 5 版**. SB クリエイティブ, 2017.

[13] 松浦 健一郎, 司 ゆき. **C 言語 [完全] 入門**. SB クリエイティブ, 2022.

[14] 村山 公保. **C プログラミング入門以前 [第 2 版]**. マイナビ出版, 2019.

索引