ま　え　が　き

本書は，C 言語によるディジタル無線通信技術のシミュレーションを行うための基礎的な考え方をまとめたものである。今日，C 言語はプログラミング言語として新しいものではないが，特に信号処理分野では依然として重要な役割を果たしている。しかし，その習得は非常に難しいと思われることが多いようである。特に，ポインタや構造体などの概念が厄介な印象を与えている。筆者が在籍する大学においても，学生は C 言語と聞いただけでうんざりする様子である。C 言語を用いてなにかの数値計算を行うだけなら，ポインタも構造体も関数も関係なく，main 関数内にやりたいことをずらりと書き並べれば，一応，事足りる。しかし，例えばディジタルシグナルプロセッサ（DSP）にアルゴリズムを実装するとき，また，大規模なソフトウェアの開発にあたって大勢で作業を分担するときなどは，構造化されたプログラミングを行わなければ対処できない。そこで関数，ポインタや構造体の概念が必要となってくるのであるが，重要なことは，これらがあるとやはり便利であるという事実である。

C 言語でこれらの概念とその文法の理解が難しいとされるのは，“C 言語の勉強”をすると，それらの便利さや必然性があまり感じられないまま，知識として押し込まれることが原因であるように思われる。実際，学生もたいへんであろうが，教える側も，“C 言語を教える”というのはとても苦しい。やはり，そこに必然性がないからである。

本書は C 言語の教科書ではない。近年のディジタル無線通信技術で重要となっている諸技術およびその理解に必要な諸現象を説明し，その実現を C 言語で行う方法を示したものである。C 言語とディジタル無線通信を一緒に説明することはとても有意義である。MATLAB をはじめとする近年のさまざまな数値計算ソフトウェアは，多くの便利な関数を提供しており，われわれはそれ

らを使うことによってあまり細かいことを考えなくても簡単に数値計算を行える。しかしそれが落とし穴である。その関数の中がブラックボックスとなって，その上に実現したアルゴリズムについても，そのレベルより深い本質的理解はできない。一方，C 言語では“なにからなにまで”自分で作っていかなくてはならないし，コンピュータのメモリについても意識しなくてはならない。このため，実現しようとするディジタル無線通信技術についても同様に，“なにからなにまで”わかっていなければ作れないのである。したがって，もし C 言語で実現できたら，それは，その技術について完全に理解したということであるといっても過言ではないと筆者は考えている。

C 言語の勉強としてではなく，目的はあくまでディジタル無線通信技術のシミュレーションであるという見地から C 言語の諸概念を見ていくと，関数や構造体があると確かに便利であると感じることができる。この感じを得て初めて，それらを使いこなせるようになる。世の中に，すでに多くの C 言語の本もあるし，無線通信技術に関する本もあるが，本書がそれらとまったく異なるのはこの点である。つまり，ディジタル無線通信技術の大きな流れからその本質までを C 言語を通して理解するのと同時に，C 言語自体を“体得”するという一石二鳥を目指したのである。

本書で示したプログラムの一部は，コロナ社の Web サイトに掲載されている（http://www.coronasha.co.jp にアクセスし，キーワード検索に本書名を入力して本書紹介ページへ移動すると，関連情報よりダウンロードが可能）。

読者におかれては，ぜひコンピュータ上でプログラムを動かし，それぞれの変数の値をご自分で確認されることで理解を完全なものとされたい。

本書の完成のため多大なる支援をいただいたコロナ社の方々に厚くお礼申し上げる。

2010 年 9 月

神谷 幸宏

目　　次

1. 本書で必要とするC言語の文法

2. ディジタル通信技術の概観

3. 信号と雑音の基礎理論

4. 変復調技術

5. フェージング

6. スペクトル拡散通信と RAKE 受信機

7. 直交周波数分割多重（OFDM）

付　録

プログラム例一覧

1 本書で必要とするC言語の文法

本章では，本書で必要とするC言語の知識についてまとめた。ただし，ここではC言語の文法の原理および流れの説明に重点を置くものとし，言語としての詳細な仕様については，別途参考書[1)～3)†]を参照されたい。関数化，ポインタ，構造体およびメモリについての理解は重要である。本章では特に，本書で説明するプログラム例を明確に理解するための情報をまとめた。なお，本書で使用する関数は，一覧として巻末の付録6にまとめている。

1.1 基礎的事項

1.1.1 main 関数

C言語のプログラムは複数の**関数**の集合からなる。main関数はそれらの関数の一つではあるが，他のすべての関数をどのような順番で実行していくかを

リスト1.1 main関数の例

```
 1  #include<stdio.h>
 2  #include<stdlib.h>
 3  #define N 100
 4  int main(void){
 5    int i,s[N];
 6    for(i=0; i<N; i++){
 7      if(rand()%2==0)  s[i] = 1;
 8      else  s[i] = -1;
 9    }
10    return 0;
11  }
```

† 肩付き番号は巻末の引用・参考文献の番号を示す。

記述している最も基本的な関数である。また，main関数のみのプログラムも可能である。**リスト1.1**に，簡単なmain関数の例を示す。この例において，main関数は4行目から始まる。このリストを用いて以下に基礎的事項を示す。

1.1.2　式（文）とセミコロン

C言語において一つの式の最後はセミコロン（;）により明示される。

1.1.3　ヘッダファイルの読込みとマクロ

標準入出力を定義するstdio.hおよび標準ライブラリstdlib.hの読込みが，それぞれ1行目および2行目で定義される。これら“.h”で終わるファイルは，C言語において**ヘッダファイル**と呼ばれ，コンパイル時にこれらのヘッダファイル内に定義された変数や関数が参照される。3行目は`N`という文字を100に置換することを定義している。これをマクロと呼ぶ。

1.1.4　変　数　宣　言

5行目で整数（`int`）型の変数`i`と配列`s[N]`が定義される。ここで，3行目で`N`は100と読み代えることが定義されているため，`s[100]`という配列のためのメモリが，コンピュータが有するすべてのメモリ領域のどこかに確保される。使用されるすべての変数は，その変数の型を明らかにして宣言される。これは，型によって必要なメモリの量が異なるためである。

1.1.5　繰　返　し　演　算

6行目の`for(i=0; i<N; i++)`は，`i`の初期値を0として`(i=0)`，`i<N`である限り1ずつ増やし`(i++)`，それに続く{　}で囲まれた部分（7～8行目）までを繰り返す。{　}がない場合には，`for`文のつぎの1文しか繰り返されないので注意が必要である。

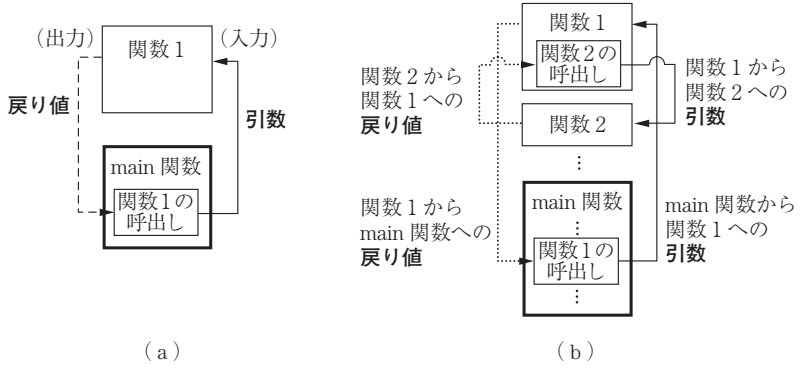
1.1.6 条 件 分 岐

7行目でif文は，括弧の中の条件が真となる場合のみ，その後のプログラムを実行する。この例で，`rand()%2` は一様乱数を発生する関数 `rand()` の値を2で割った余りを意味する。これが0に等しければ，続く `s[i]=-1;` を実行する。if文のカッコの中で“==”と等号が2重となっているのは，“その両辺が等しい”ことを意味する。等号が一つである場合は“左辺に右辺の値を代入する”という意味となることに注意を要する。また，8行目では，if文の条件が真でないとき実行する内容を `else` 以降に記述する。ifもelseも，実行する内容が複数行にわたる場合は，その範囲を { } で囲う必要がある。また，関数 `rand()` を使うためには，ヘッダファイル stdlib.h をインクルードする必要があることに注意する。

1.2 関　　数

1.2.1 関 数 と は

定型的に何度も繰り返される処理は，関数としてmain関数とは別に記述するのが便利である。その処理が必要となったとき，main関数は別途定義された関数を呼び出す。また，main関数以外の関数が，さらに別の関数を呼び出すこともできる。このイメージを**図1.1**（a）および（b）に示す。



（a）　　（b）

図1.1　関数のイメージ

1.2.2 引数と戻り値

図1.1（a）に示すように，例えばmain関数から関数1へ“引きわたす”値のことを**引数**と呼ぶ。つまり，引数は関数の入力となる。反対に，関数1で計算され，main関数に戻される値のことを**戻り値**と呼ぶ。つまり，戻り値は関数の出力となる。また，関数はmain関数とこれらの値をやりとりするだけでなく，ある関数1が別の関数2を呼び出して，引数・戻り値のやりとりをすることができる。このイメージを図1.1（b）に示す。

1.2.3 関数の書式

C言語では，main関数も含めすべての関数がつぎのような書式で表される。

```
戻り値の型　関数名(引数1の型,引数1,引数2の型,引数2…)
{
処理の内容
}
```

関数の説明として**リスト1.2**を示す。

コラム1　main関数が先か関数が先か

図1.1（a）および（b）では，main関数はいくつかの関数の一番下に置かれている。これは，プログラムをコンパイルする際に，コンパイラがプログラムを上から読んで機械語に翻訳していくことに起因する。もし図1.1（a）でmain関数と関数1を逆転させ，main関数が先で関数1がその後に配置されているようにするとなにが起こるだろうか。このときコンパイラは，先にmain関数を翻訳することになる。そしてその中で関数1を発見することになるが，その時点でコンパイラは関数1の定義を知らない。このため，それを機械語に翻訳することができず，それ以降のコンパイルを続行できないのである。main関数を最後に配置すれば，このような問題は生じない。どうしてもmain関数を先に記述したい場合は，**プロトタイプ宣言**[2)]を行うことでこの問題を解決することができる。しかし，本書ではより簡潔なプログラムを目指してmain関数を最後に置き，プロトタイプ宣言は行わない。

リスト 1.2　関数の例

```
1  #include<stdio.h>
2  #include<math.h>
3  void _funcI(void){ // I型関数：エラーメッセージの表示
4      printf("エラー：正の値を入力してください．¥n");
5  }
6  float _funcII_1(void){// II-1型関数：キーボードから値の入力
7      float v;
8      printf("真数の値を入力してください:");
9      scanf("%f",&v);
10     return v;
11 }
12 void _funcII_2(float b){ // II-2型関数：値の表示
13     printf("入力された値は%fです．¥n",b);
14 }
15 float _funcIII(float b){ // III型関数：真数からdBへの変換
16     float c;
17     c = 10*log10(b);
18     return c;
19 }
20 int main(void){
21     float a, a_dB;
22     a = _funcII_1(); // キーボードから値を受け取り変数aへ
23     if (a<=0.0)
24        _funcI(); // もしaの値が0より小さければエラーメッセージ表示
25     else {
26         _funcII_2(a); // そうでなければ，aの値を表示
27         a_dB = _funcIII(a); // aの値に対応するdBの値を計算しa_dBへ
28         printf("それは%f[dB]です．¥n",a_dB);
29     }
30     return 0;
31 }
```

この例は，ある値をキーボードから入力すると，そのデシベル（dB）の値を計算して表示する（付録 2 参照）プログラムである。0 以下の値を入力すると対数を計算できないため，エラーメッセージが表示される。

このプログラムの考え方を以下にまとめる。C 言語では，プロトタイプ宣言をしなければ通常 main 関数は最後に置かれる（コラム 1 参照）。プログラムの処理の全体像は main 関数に書かれているため，最初に見るのは最後に置かれている main 関数である。

（21 行目）　変数宣言

（22 行目）　6 行目の関数 `_funcII_1( )` に飛び，キーボードから入力され

た値を，変数aに取り込む。引数はなく，_funcII_1()内でキーボードから入力された値が戻り値となる。

（23～24行目）　もし入力値が0以下であれば，3行目の関数_funcI()に飛び，エラーメッセージを表示する。この関数は，あらかじめ決められたメッセージを表示するだけなので，引数も戻り値もない。

（25行目）　そうでなければ（入力値が0より大きければ），

（26行目）　12行目_funcII_2()に飛び，入力された値をディスプレイに表示する。引数はあるが戻り値はない。

（27～28行目）　15行目_funcIII()に飛び，dBの値を計算し，a_dBに戻された計算結果を画面に表示する。この関数には引数も戻り値も存在する。

いま，便宜上，ここで使われた関数を引数・戻り値の組合せによって**表1.1**のように3種類に分類し，それぞれⅠ型，Ⅱ-1およびⅡ-2型，さらにⅢ型と名付ける。以下にそれぞれについて説明する。

表1.1　関数の分類

	引　数	戻り値	関数名
Ⅰ　型	×	×	_funcI()
Ⅱ-1型	×	○	_funcII_1()
Ⅱ-2型	○	×	_funcII_2()
Ⅲ　型	○	○	_funcIII()

Ⅰ型……引数・戻り値ともになし　　リスト1.2の24行目で，エラーメッセージを表示する関数_funcI()はこの型である。決められたエラーメッセージを画面に表示するだけであるから，引数も戻り値も不要である。このような関数をⅠ型と分類する。リスト1.2の3行目だけを取り出して**図1.2**に示す。

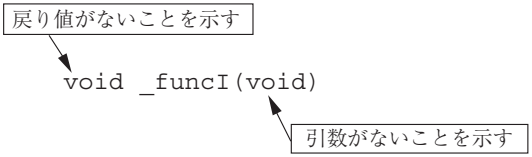


図1.2　リスト1.2の3行目：Ⅰ型関数

索　　引

―― 著 者 略 歴 ――

2000 年　名古屋大学大学院工学研究科博士後期課程修了（電子情報学専攻）
　　博士（工学）
2003 年　東京農工大学助教授
2007 年　東京農工大学大学院准教授
　　現在に至る

C 言語によるディジタル無線通信技術

Digital Wireless Communication Technologies in C

2010 年 11 月 22 日　初版第 1 刷発行　★

検印省略

著　者　神谷幸宏（かみや ゆきひろ）
発行者　株式会社　コロナ社
　　代表者　牛来真也
印刷所　萩原印刷株式会社

112-0011　東京都文京区千石 4-46-10

発行所　株式会社　**コロナ社**

CORONA PUBLISHING CO., LTD.

Tokyo Japan

振替 00140-8-14844・電話（03）3941-3131（代）

ホームページ　http://www.coronasha.co.jp

ISBN 978-4-339-00817-3　（安達）　（製本：愛千製本所）

Printed in Japan