

# ま え が き

本書は、C言語によるデ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つて、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なされたい。

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ます。

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

1.1 基礎的事項	1
1.1.1 main 関数	1
1.1.2 式(文)とセミコロン	2
1.1.3 ヘッダファイルの読み込みとマクロ	2
1.1.4 変数宣言	2
1.1.5 繰り返し演算	2
1.1.6 条件分岐	3
1.2 関数	3
1.2.1 関数とは	3
1.2.2 引数と戻り値	4
1.2.3 関数の書式	4
1.3 ポインタ	9
1.3.1 ポインタとは	9
1.3.2 ポインタを用いた関数: IV型 — III型への適用	12
1.3.3 ポインタとメモリ割当て	14
1.4 配列	16
1.4.1 配列とその書式	16
1.4.2 配列とポインタ	16
1.4.3 ポインタを使って配列を表現するときのメモリ割当て	17
1.4.4 配列を使う関数: V型	18
1.5 構造体	19
1.5.1 構造体とは	19

1.5.2	構造体の便利さ	21
1.5.3	構造体と関数	23
1.5.4	構造体のポインタ	24
1.5.5	構造体の配列	25
1.6	ファイル入出力	26

## 2. デジタル通信技術の概観

2.1	デジタル通信とアナログ通信	28
2.2	本書が対象とする範囲	32

## 3. 信号と雑音の基礎理論

3.1	信号の数学的表現	33
3.1.1	実信号と解析信号	33
3.1.2	実信号と解析信号の関係	35
3.2	雑音の数学的表現	39
3.2.1	付加的白色ガウス雑音	39
3.2.2	SN 比	42
3.3	SN 比を最大にするフィルタ —— 整合フィルタ	44
3.3.1	トランスバーサルフィルタと畳込み	45
3.3.2	相 関	46
3.3.3	整合フィルタの導出	47
3.3.4	整合フィルタによる SN 比改善	48

## 4. 変復調技術

4.1	BPSK と QPSK	53
4.1.1	BPSK の 原 理	53

4.1.2	ビット誤り率	55
4.2	BPSK のシミュレーション	58
4.2.1	配列を用いたプログラム	58
4.2.2	データ型の定義とポインタによる信号の表現	64
4.2.3	変数の構造化	72
4.3	QPSK	77
4.3.1	QPSK の原理	77
4.3.2	QPSK のシミュレーション	80

## 5. フェージング

5.1	フェージングの影響	84
5.2	信号の減衰——周波数フラットフェージング	85
5.2.1	原理	85
5.2.2	計算機シミュレーション	88
5.3	波形ひずみ——周波数選択性フェージング	89
5.4	フェージング通信路のモデル化と計算機シミュレーション	92
5.4.1	遅延プロファイル	92
5.4.2	計算機シミュレーション	93

## 6. スペクトル拡散通信と RAKE 受信機

6.1	原理	108
6.2	プログラム上で実現するための考え方	110
6.2.1	送信機の実現	111
6.2.2	受信機の実現	112
6.2.3	拡散系列	113
6.2.4	整合フィルタを用いた SS 信号の復調	117
6.3	RAKE 合成	118

6.3.1 遅延波の分離 ..... 118  
 6.3.2 RAKE 合成 ..... 122  
 6.3.3 RAKE 合成による SN 比の改善 ..... 126  
 6.4 プログラム例 ..... 127

## 7. 直交周波数分割多重 (OFDM)

7.1 原 理 ..... 141  
 7.2 OFDM 送信機・受信機の原理と構成 ..... 142  
     7.2.1 原 理 ..... 142  
     7.2.2 FFT を用いた OFDM 送信機・受信機の構成 ..... 147  
     7.2.3 ガードインターバルの必要性 ..... 149  
     7.2.4 スクランプリングと誤り訂正符号の必要性 ..... 151  
 7.3 プログラム例 ..... 154

## 付 録

1. MATLAB にテキストファイルを読み込む方法 ..... 169  
 2. デシベルの値 ..... 169  
 3. 相関係数を用いた SN 比の計算法 ..... 171  
 4. スペクトル拡散信号計算の効率化 ..... 172  
 5. RAKE 合成計算の効率化 ..... 172  
 6. 本書で使用する関数 ..... 176  
 引用・参考文献 ..... 177  
 索 引 ..... 178

## プログラム例一覧

リスト 1.1	main 関数の例	1
リスト 1.2	関数の例	5
リスト 1.3	ポインタの例	11
リスト 1.4	IV 型関数 (ポインタを用いた関数の例)	13
リスト 1.5	ポインタとメモリ割当ての問題	14
リスト 1.6	malloc 関数によるメモリ確保	15
リスト 1.7	配列とポインタを用いた表現	17
リスト 1.8	malloc によるメモリ確保	17
リスト 1.9	V 型関数の例: 配列を使った場合	18
リスト 1.10	_funcV() のポインタによる表現	19
リスト 1.11	最も簡単な複素数の定義	20
リスト 1.12	構造体の例	20
リスト 1.13	typedef を利用した構造体の宣言	21
リスト 1.14	構造体を引数・戻り値とする関数の例	23
リスト 1.15	構造体のポインタの例	24
リスト 1.16	配列とポインタを用いた表現	25
リスト 1.17	malloc によるメモリ確保	26
リスト 1.18	ファイルへの書出し	27
リスト 4.1	配列を用いて作成した BPSK のプログラム例	58
リスト 4.2	構造体を用いた SEQ_DAT 型と SEQ_SIG 型の定義	65
リスト 4.3	データ型の定義とポインタによる信号の表現を行った BPSK の m-file 例	68
リスト 4.4	変数を構造化した BPSK の m-file 例	73
リスト 4.5	30 台の送信機を作成する例	76
リスト 4.6	QPSK のプログラム例	80

リスト 5.1	レイリー分布に従う振幅値の発生	88
リスト 5.2	複素数演算関数群	94
リスト 5.3	トランスバーサルフィルタ	98
リスト 5.4	フェージング通信路の出力を計算する例	101
リスト 6.1	SS 送信機および RAKE 合成を行う受信機の実現	128
リスト 7.1	OFDM 送信機・受信機のプログラム例	154
付リスト 1	テキストファイルを読み込む MATLAB の m-file 例	169
付リスト 2	効率化した <code>_ss()</code>	172
付リスト 3	SS 送信機および RAKE 合成を行う受信機の実現	173

# 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<sup>1)~3)</sup>†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。

## 2 1. 本書で必要とする C 言語の文法

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み込みとマクロ

標準入出力を定義する `stdio.h` および標準ライブラリ `stdlib.h` の読み込みが、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 繰り返し演算

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。

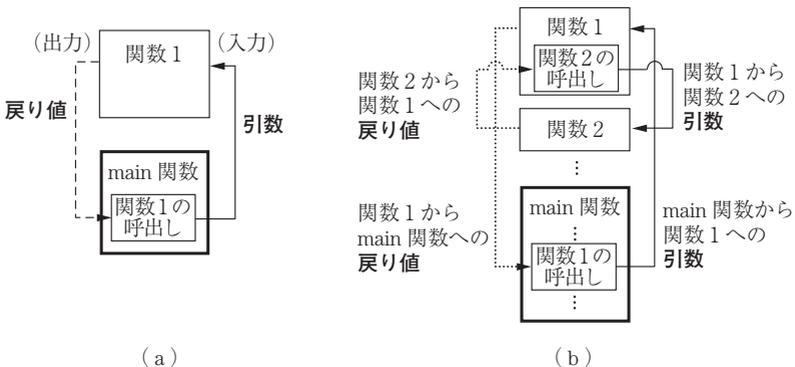


図 1.1 関数のイメージ

### 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**<sup>2)</sup>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

6 1. 本書で必要とする C 言語の文法

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。

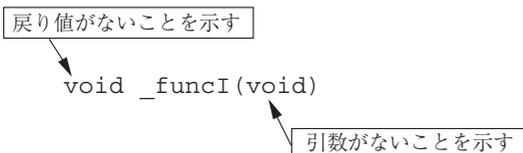


図 1.2 リスト 1.2 の 3 行目: I 型関数

# 索引

	<b>【あ】</b>				
アドレス	10	周波数フラットフェージング	85		
				<b>【と】</b>	
		情報源符号化	31	同期検波	38
		信号点	54	トランスバーサルフィルタ	45
エラー処理	16	信号点配置	54		
		シンボル	54	<b>【に】</b>	
		シンボル速度	77	2相PSK	53
		<b>【す】</b>		<b>【は】</b>	
解析信号	33	スペクトル拡散	108	波形ひずみ	89
ガウス分布	39			<b>【ひ】</b>	
拡散	110	<b>【せ】</b>		引数	4
拡散系列	109	整合フィルタ	44	ビット	54
関数	1			ビット誤り率	55
		<b>【そ】</b>		標準化定理	30
		相関	46		
		<b>【た】</b>		<b>【ふ】</b>	
		畳込み	45	フィンガー数	126
逆拡散	110	多値化	77	付加的白色ガウス雑音	39
				複素搬送波	34
		<b>【ち】</b>		復調	29, 78
		チップ	109	符号	30
		チャンネル係数	93	プリアンブル	152
		チャンネル推定	122		
		直交周波数分割多重方式	141	<b>【へ】</b>	
		直交成分	34	ヘッダファイル	2
				変調	28, 53
		<b>【つ】</b>		<b>【ほ】</b>	
		通信路符号化	31	ポインタ	9
		<b>【て】</b>		<b>【ま】</b>	
自己相関	46	データ速度	77	マルチパスフェージング	84
自己相関関数	115				
実信号	33				
周波数選択性フェージング					

	<b>【も】</b>		<b>【よ】</b>		<b>【れ】</b>
戻り値	4	4相PSK		77	レイリー分布 87
	<b>【や】</b>		<b>【り】</b>		
矢印演算子	24	量子化		31	

---

	<b>【A】</b>		<b>【M】</b>		<b>【R】</b>
AWGN	39	MRC M系列		48 115	RAKE合成 122
	<b>【B】</b>		<b>【O】</b>		<b>【S】</b>
BER	55			SN比	42
BPSK	53	OFDM		SS	108
	<b>【D】</b>		<b>【Q】</b>		
dB	169	QPSK		77	

— 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言語によるデジタル無線通信技術

Digital Wireless Communication Technologies in C

© Yukihiro Kamiya 2010

2010年11月22日 初版第1刷発行

★

検印省略

著者 かみ や ゆき ひろ  
神谷幸宏  
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



無断複写・転載を禁ずる

落丁・乱丁本はお取替えいたします