

苦しんで覚える C 言語・目次

苦 C の読み方	3
各章の解説	6
コンパイラ紹介	20
0 章 コンピュータとは何か?	23
0.1 コンピュータとは何か?	24
0.1.1 現代人とコンピュータ	24
0.1.2 コンピュータとは	25
0.1.3 CPU とは?	27
0.1.4 メモリとは?	27
0.2 プログラムとは何か?	29
0.2.1 プログラムとは	29
0.2.2 単純な文法	30
0.2.3 明確な意味	32
0.2.4 まだまだ曖昧	35
1 章 世界最小のプログラム	39
1.1 何もしないプログラム	40
1.1.1 C 言語の構造	40
1.1.2 関数の作り方	41
1.1.3 main 関数は特別	42
1.1.4 プログラムを動作させる	43
1.2 コンパイラは翻訳ソフト	44
1.2.1 すべては機械語	44
1.2.2 プログラミング言語の登場	45
1.2.3 C 言語翻訳ソフト	46

2 章 プログラムの書き方	51
2.1 書き方のルール	52
2.1.1 トークン	52
2.1.2 フリーフォーマット	53
2.1.3 そのほかのルール	54
2.2 書き方の慣習	55
2.2.1 関数の書き方	55
2.2.2 インデント	56
2.2.3 コメント	57
3 章 画面への表示	61
3.1 文字列の表示	62
3.1.1 どうしても必要	62
3.1.2 printf 関数	62
3.1.3 どこに書くのか?	63
3.1.4 説明書の取り込み	64
3.1.5 お待たせしました	66
3.2 改行文字	67
3.2.1 改行問題	67
3.2.2 エスケープシーケンス	68
4 章 数値の表示と計算	75
4.1 数値の表示	76
4.1.1 文字列と数値	76
4.1.2 数値を表示する	77
4.1.3 文字列との組み合わせ表示	78
4.1.4 複数の数値の表示	80
4.2 基本的な計算	82
4.2.1 計算とその結果の表示	82
4.2.2 四則演算子	84
4.2.3 複雑な式	86

4.3	数値の種類	87
4.3.1	さまざまな数値	87
4.3.2	実数の計算	88

5章	数値の記憶と計算	95
-----------	-----------------	-----------

5.1	数値を記憶する	96
5.1.1	記憶の必要性	96
5.1.2	変数というメモリ	96
5.1.3	変数の宣言	97
5.1.4	変数への値の代入	99
5.1.5	変数を数値の代わりに使う	100
5.1.6	代入と演算を同時に	102
5.2	変数の種類	105
5.2.1	データ型	105
5.2.2	実数を記憶する変数	106
5.3	型の変換	108
5.3.1	整数と実数の混合計算	108
5.3.2	強制的に変換	109
5.4	数値の桁そろえ	111
5.4.1	整数の桁そろえ	111
5.4.2	コンピュータ的な表示	113
5.4.3	実数の桁そろえ	114

6章	キーボードからの入力	119
-----------	-------------------	------------

6.1	入力用の関数	120
6.1.1	入力の必要性	120
6.1.2	scanf 関数	120
6.1.3	数値の入力	121
6.1.4	複数の入力	123
6.1.5	簡易シグマプログラム	124
6.2	入力の恐怖	127
6.2.1	恐怖の入力ミス	127

6.2.2	区切り記号のミス	127
6.2.3	大きすぎる数値	128
6.2.4	文字列の恐怖	128
6.2.5	解決法について	129

7章	比較と判断	133
-----------	--------------	------------

7.1	比較を行う文	134
7.1.1	条件判断	134
7.1.2	条件判断をする文	134
7.1.3	比較のための演算子	136
7.2	比較のための演算子	139
7.2.1	等値演算子	139
7.2.2	関係演算子	140
7.2.3	論理演算子	142
7.3	複数の処理の実行	144
7.3.1	複数の処理の必要性	144
7.3.2	ブロック文	145

8章	場合分け処理を行う	151
-----------	------------------	------------

8.1	2つの場合分け	152
8.1.1	偽の場合の処理	152
8.1.2	使い方は同じ	152
8.1.3	読みやすくする	154
8.2	3つ以上の場合分け	155
8.2.1	条件が複数の場合の処理	155
8.2.2	見やすい書き方	157
8.3	番号による場合分け	159
8.3.1	番号と対応させる処理	159
8.3.2	当てはまらない場合の処理	161
8.3.3	同様の処理をまとめる	163
8.3.4	判断力の弱さ	165

9章	回数が決まっている繰り返し	171
9.1	繰り返しを行う文	172
9.1.1	繰り返し動作	172
9.1.2	回数の表示	173
9.2	ループ動作の仕組み	175
9.2.1	初期化と条件	175
9.2.2	いつまでも…	177
9.2.3	強制脱出	178
10章	回数がわからない繰り返し	183
10.1	回数不明ループ	184
10.1.1	回数を求めるループ	184
10.1.2	for文との交換性	187
10.2	入力チェック	188
10.2.1	後判定と先判定	188
10.2.2	入力チェック	189
11章	関数の作り方	197
11.1	自作関数を作る	198
11.1.1	プログラムの部品化	198
11.1.2	自作関数を作る	199
11.1.3	プロトタイプ宣言	200
11.1.4	自作関数を呼び出す	202
11.2	関数に数値を渡す	204
11.2.1	引数を持つ関数	204
11.2.2	関数に数値を渡す	205
11.2.3	複数の引数	207
11.3	関数から数値を返す	209
11.3.1	戻り値を返す関数	209
11.3.2	戻り値の制限	211

12章	変数の寿命	217
12.1	関数内で寿命が尽きる変数	218
12.1.1	ローカル変数の寿命	218
12.1.2	同じ名前でも別の変数	221
12.1.3	関数の独立性	222
12.1.4	正確にはブロック内	223
12.2	最後まで生き残る変数	225
12.2.1	グローバル変数の寿命	225
12.2.2	すべての関数で共有される	226
12.2.3	ローカル変数は独立する	228
12.3	関数内で生き残る変数	230
12.3.1	静的なローカル変数の寿命	230
13章	複数の変数を一括して扱う	235
13.1	複数の変数をまとめて扱う	236
13.1.1	配列の概念	236
13.1.2	配列の宣言	236
13.1.3	配列の取り扱い	237
13.2	配列の使い方	239
13.2.1	初期値の代入	239
13.2.2	全要素の表示	241
13.2.3	要素数を求める	242
13.2.4	配列のコピー	244
14章	文字列を扱う方法	251
14.1	文字の扱い方	252
14.1.1	文字列を扱う変数	252
14.1.2	文字を扱うには	253
14.1.3	文字コード	254
14.1.4	文字に対する計算	255
14.2	文字列を扱う方法	259
14.2.1	配列にしてしまおう	259

14.2.2	文字列の初期化	261
14.3	文字列処理関数	263
14.3.1	数値への変換	263
14.3.2	文字列のコピー	264
14.3.3	文字列の連結	266
14.3.4	究極の文字列合成関数	268
14.3.5	文字列の入力	269
14.3.6	文字数を数える	271
14.3.7	文字列の比較	273

15章 | ポインタ変数の仕組み **279**

15.1	メモリの仕組み	280
15.1.1	メモリ?	280
15.1.2	超巨大な1列ロッカー	280
15.1.3	CPUのビット数	281
15.1.4	32ビットのロッカー	282
15.2	変数とメモリの関係	283
15.2.1	変数はメモリ上に存在する	283
15.2.2	メモリ上の番号を表示する	284
15.2.3	複数の変数の番号	286
15.2.4	配列の番号	287
15.3	&付けが必要な変数の正体	289
15.3.1	&付き変数の正体	289
15.3.2	すべては値渡しである	289
15.3.3	scanf関数で&をつける理由	290
15.4	アドレスを記憶する変数	293
15.4.1	ポインタという単語	293
15.4.2	ポインタ型	294
15.4.3	ポインタ値	295
15.4.4	ポインタ変数	296
15.5	ポインタ変数を使ってみる	298
15.5.1	ポインタ変数の宣言	298

15.5.2	アドレスを代入する	299
15.5.3	モードの切り替え	301
15.5.4	すなわちショートカット	303
15.6	引数による情報の受け渡し	305
15.6.1	ポインタ型の引数	305
15.6.2	配列型の引数	307
15.6.3	配列型引数の奇妙な性質	308
15.6.4	アドレスを渡している	310
15.7	配列とポインタの奇妙な関係	313
15.7.1	配列のような使い方	313
15.7.2	ポインタ専用の書き方	316
15.7.3	古き悪きポインタ演算	318
15.7.4	アドレスのことは忘れましょう	320

16章 | 複数の型をまとめる **325**

16.1	異なる型の変数をまとめる	326
16.1.1	まとめてデータを扱いたい場合	326
16.1.2	構造体の使い方	329
16.1.3	構造体変数自体の処理	331
16.1.4	構造体の簡潔な宣言	334
16.2	構造体の引数	336
16.2.1	構造体で情報を渡す	336
16.2.2	構造体でもポインタ変数	339
16.2.3	構造体でもポインタ引数	341
16.3	構造体の配列	343
16.3.1	構造体の配列	343
16.3.2	構造体配列の引数	344

17章 | ファイルの取り扱い **349**

17.1	テキストファイルの読み書き	350
17.1.1	ファイルの取り扱い	350
17.1.2	ファイルの開閉	350
17.1.3	ファイルへの書き込み	352
17.1.4	ファイルからの読み込み	354

17.2	バイナリファイルの読み書き	357
17.2.1	テキストとバイナリ	357
17.2.2	ファイルの開閉	357
17.2.3	ファイルへの書き込み	358
17.2.4	ファイルからの読み込み	361
17.3	ドラッグへの対応	363
17.3.1	ドラッグされたファイル名の取得	363
17.3.2	オプションの解析	365
18章 マクロ機能		371
18.1	不変の値の取り扱い	372
18.1.1	始めから終わりまで不変の値	372
18.1.2	数値に名前をつける	373
18.1.3	文字列に名前をつける	374
18.2	その他の方法による定数	376
18.2.1	const 定数	376
18.2.2	enum 定数	377
18.2.3	数値指定 enum 定数	379
18.3	簡易的な関数の実現	381
18.3.1	#define 疑似命令の高度な機能	381
18.3.2	マクロという簡易関数	382
18.3.3	副作用の恐怖	384
19章 動的配列		391
19.1	配列を自由自在に作る	392
19.1.1	配列の欠点	392
19.1.2	メモリの確保	393
19.1.3	動的配列の要素数を拡大する	396
20章 複数のソースファイル		403
20.1	最小限の分割	404
20.1.1	複数ファイルを使う理由	404
20.1.2	ソースとヘッダーファイル	404
20.1.3	最小限のヘッダーファイル	405

20.2	分割の定石	409
20.2.1	変数の共有	409
20.2.2	extern 宣言	410
20.2.3	ヘッダーファイルの重複防ぎ	413

21章 | キーボード入力 421

21.1	1行の文字列として入力する	422
21.1.1	gets 関数によるキーボード入力	422
21.1.2	バッファオーバーラン対策	423
21.1.3	文字列から数値などを取り出す	426

Addendum 429

A	単語と記号	430
A.1	予約語	430
A.2	出力変換指定子	431
A.3	入力変換指定子	432
A.4	演算子と優先順位	434
A.5	記憶クラス指定子	438
A.6	エスケープ文字	439
A.7	定義済み定数	440
B	標準ライブラリ関数一覧	442
B.1	入出力 <stdio.h>	442
B.2	汎用 <stdlib.h>	449
B.3	文字処理 <ctype.h>	454
B.4	文字列処理 <string.h>	457
B.5	数学関数 <math.h>	461
B.6	時間 <time.h>	466
B.7	制御	468

ASCII コード表 470

索引 471

用語・略語辞典 476

著者紹介・苦しんで覚える C 言語 Web サイト . . . 479