目 次

C language Programming

第1章 C言語プログラムの実行

C言語はさまざまなシステム開発に使われており，現代の言語の基礎となる言語です。そのC言語を学習するには，自分でプログラムを作成して動かしてみるのが一番です。

この章では，C言語によるプログラムのコーディングから動かし方に至るまでの手順を説明します。

実際のプログラム例として，文字列を出力するプログラムと変数に値を入力してその変数の値を表示するプログラムを示しますので，そこからC言語プログラムの構成をつかんでください。

1.1 C 言語プログラムの実行の仕組み

コンパイラ型言語
ソースプログラム
インタプリタ型言語
テキストエディタ
オブジェクトプログラム
ライブラリ
実行可能プログラム

C 言語はコンパイラ型言語です。C 言語のソースプログラム（人間に読める形で記述されたプログラム）は，コンパイラというソフトウェアでコンピュータに分かる機械語に翻訳されます。この操作をコンパイルといい，C 言語のプログラムをコンピュータ上で実行するために必要な処理手順です。

コンパイラ型に対して，インタプリタ型言語というのがあります。ソースプログラムを変換することなく，そのままの形で解釈しながらコンピュータが実行します。この解釈・実行するソフトウェアをインタプリタと呼びます。

コンパイラ型言語とインタプリタ型言語の比較を表に示します。

表　コンパイラ型言語とインタプリタ型言語の比較

	コンパイラ型言語	インタプリタ型言語
実行の仕方	あらかじめ機械語に変換してから実行	ソースプログラムをそのまま実行
実行速度	速い	遅い
プログラムのサイズ	（機械語として）大きい	（機械語として）小さい
デバッグのしやすさ	面倒（エラーメッセージが 1 度にたくさん出てくる）	簡単（実行途中でその都度エラーが出る）
利用方法	一括実行	対話型実行

コンパイラ型言語としての C 言語のプログラムの作成から実行までの手順は次のとおりです。

①　ソースプログラムを入力する

ソースプログラムはテキストファイルです。Windows 上のメモ帳やプログラム作成専用のテキストエディタなどを使って C 言語のソースプログラムを入力します。C 言語のソースプログラムのファイル名は〰〰.c という名前にします（〰〰の部分は任意の名前でかまいません）。

②　C 言語のコンパイラでソースプログラムをコンパイルする

次に①でできたソースプログラムを Borland C++などのコンパイラでコンパイルして機械語のプログラムを生成します。この機械語の形のプログラムをオブジェクトプログラムといいます。Windows 上では〰〰.obj というファイル名のオブジェクトプログラムができます。〰〰の部分はソースプログラム〰〰.c の〰〰と同じ名前になります。

③　ライブラリと結合して実行可能プログラムを生成する

②で生成したオブジェクトプログラムは，そのままではコンピュータ上で実行できません。標準ライブラリなどのライブラリと結合（リンク）することによって，コンピュータ上で実行できる形式に変換されます。ライブラリもオブジェクトプログラムの集まったものです。この結果できた実行可能プ

コンパイルエラー
実行時エラー
デバッグ

ログラムをロードモジュールといいます。Windows 上では〰〰.exe という名前のファイルになります。

④　実行可能プログラムを実行する

Windows 上ではコマンドプロンプトというウィンドウを立ち上げ，その中で〰〰.exe の〰〰の部分をコマンドとして入力することによって，プログラムが実行されます（.exe は，入力の必要はありません）。

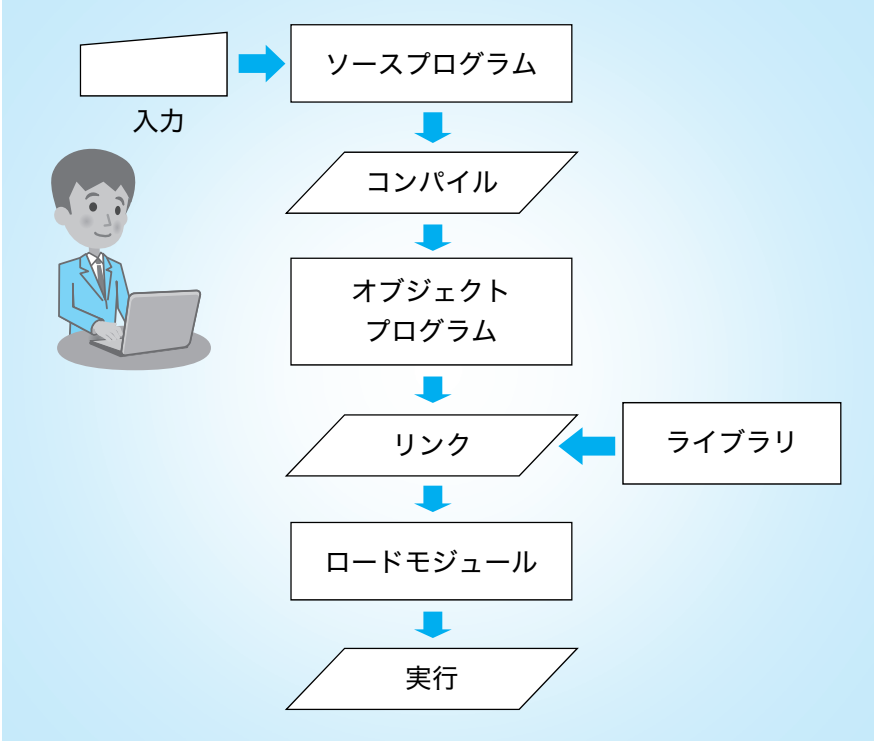


図　プログラムの作成から実行まで

アドバイス

標準ライブラリ：キーボード入力やディスプレイ表示などを行うためのライブラリです。標準的なプログラムで必要となる機能をまとめているため，標準ライブラリと呼ばれます。

このようにコンパイラ型言語では，ソースプログラムを作成・修正するたびに，この一連の操作を繰り返さなければなりません。通常は 1 回で正しい結果が得られることは少なく，コンパイル段階でエラーが出たり，実行段階でエラーになったりします。**コンパイルエラー**は，コンパイラがエラーメッセージとして出力しますので，ある程度見つけやすく，修正も比較的簡単なのですが，**実行時エラー**はなかなか見つけにくいです。主に論理的に間違ったことをコンピュータに指示するエラーのため，実行結果が予想したものと違ったり，実行が終わらなかったりします。このようなコンパイル時エラーや実行時エラーを取り除く作業を**デバッグ**といいます。

ポイントは頭に入っているかな。理解できているか Check してみましょう。

check

- [x] プログラムのコンパイル・実行のさせ方
- [x] 文字列出力プログラム
- [x] 変数値の入力 …… scanf

  変数値の出力 …… printf
- [x] 整数の変換指定子 …… %d

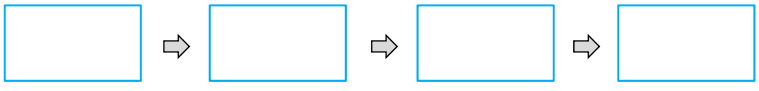
章末問題

1章の仕上げです。問題を解いて力を試してみましょう。

問1　次の解答群をC言語のプログラムの作成手順として正しい順に並び換えなさい。

解答群

ア　ライブラリと結合して実行可能プログラムを作成する。
イ　ソースプログラムを入力する。
ウ　実行可能プログラムを実行する。
エ　コンパイラでコンパイルする。



問2　次のプログラムはディスプレイに“konnichiha”と表示するプログラムです。［　a　］，［　b　］，には何が入りますか。適切なものを解答群の中から選びなさい。

```
#include <stdio.h>
int [ a ](void)
{
    [ b ]
    return 0;
}
```

aの解答群

ア　MAIN　　イ　main
ウ　mein　　エ　Main

答え

bの解答群

ア　printf('konnichiha');　　イ　printf("konnichiha");
ウ　printf 'konnichiha';　　エ　printf "konnichiha";

答え

11.1 午後の言語問題の解法


アルゴリズム

(1) 解き方のテクニック

午後の言語問題でC言語を選択する人は，すでに教育機関や企業でC言語を学んできた方が多いと思います。また，企業にこれから就職する人で，企業でC言語を使うのでこれから学習するという学生の方も多いかと思います。

過去に出題されたC言語の問題を見ると，9章までに説明した項目のうち，配列，ポインタ，構造体を使った問題がほとんどです。なお，配列問題では単純な配列だけでなく，構造体やポインタと組み合わせた複雑なデータ構造を使った問題も出題されています。

今までに出題された問題の一部を演習問題として11.2以降に掲載していますが，いずれも一筋縄ではいかない難易度の高い問題といえます。これらはどれも，文法を学習しただけの知識では簡単には解けません。それではどうしたら合格ラインに達することができるでしょうか。

実際に試験を受けて合格に結び付けるための対策の要点は次の2点です。

> ① 基本的なアルゴリズムを学習することによって，プログラムの流れを理解すること
> ② 実際の問題に数多く当たってプログラムの解読の仕方を学ぶこと

基本的なアルゴリズムのパターンを覚えておくことは非常に重要です。第10章では主に①に沿った対策として，具体的なアルゴリズムをどう記述するのか，いくつかの例を挙げて解説しました。ここに掲載されたプログラムを実際に自分で入力し，コンピュータ上で動作させて確認することで，アルゴリズムへの理解が深まり，試験に出題されたときに問題がどのパターンに該当するのかがつかめるようになります。

次に②の問題解読の仕方を説明します。

実際に試験場に行って問題を読んだときには，時間の制限もあり，十分に問題を理解することはできません。特に何ページにも渡る長い問題では読むだけで時間がかかってしまいます。その場合はまず設問を先に読み，設問で問われていることは何なのかを早く理解するようにします。穴埋め問題ならその空欄に相当する〔プログラムの説明〕の部分があるはずです。空欄の前後をよく読み，〔プログラムの説明〕の該当箇所にアンダーラインを引いておきましょう。

選択肢を一つ一つ検討し，何が当てはまるかを調べます。選択肢の中には紛らわしいものも含まれますから，問題で示されたプログラムとよく対比して選択を間違わないことが大切です。

選択肢の中に添字を含む問題では，添字をｉとすると，選択肢の中にｉかi+1かi-1が入っていることがあります。このような問題で正解を導くためには，添字の値をトレースすることも必要です。

最近の問題では，設問の中に「プログラムの～部分を何回通るか」というように実行回数を要求するものもあります。この場合もプログラムを正しく

トレースできるようになっていないと正解が導けません。

また設問が 1，2，3 のように分かれている場合，その中にはプログラムを読まなくても解けるようなサービス問題が含まれていることがあります。その場合には〔プログラムの説明〕に書かれている内容をよく読んで解答してください。これができるだけでも点数が稼げます。中には設問 2 以降にそのようなサービス問題が含まれていることがありますから，設問 1 が分からなかったからといってあきらめず，ほかの設問を解いてみることをお勧めします。

それでは空欄の穴埋め問題の例を(2)に，設問 2 にプログラムの変更を含む問題の例を(3)に示し，解き方のポイントを解説しましょう。

(2) 実際の試験問題を解く様子

ここまで説明した試験テクニックの紹介に続き，ここでは，実際に出題された情報処理技術者試験の問題を例にして，どのような順序で，論理を組み立てて問題を解いていけばよいか，解説します。

例題

次の C プログラムの説明及びプログラムを読んで，設問に答えよ。

(H16 秋-FE 午後問 6)

〔プログラムの説明〕

CGI などへのリクエストの際に文字列パラメタを URL に含める場合には，その文字列パラメタを決められた規則に従って変換し送出する必要がある。このプログラムは，それに用いる変換プログラム URLEncode である。

(1) 変換対象の文字列パラメタは，JIS X 0201（7 ビット及び 8 ビットの情報交換用符号化文字集合）の文字で構成される。また，文字列パラメタの途中にナル文字（文字符号 0x00）は含まれないものとする。

(2) 変換規則は次のとおりである。

① 英数文字(0x30～0x39, 0x41～0x5A, 0x61～0x7A)及び“@”(0x40)，“*”（0x2A），“-”（0x2D），“.”（0x2E），“_”（0x5F）は変換しない（これらの文字を以降，無変換文字と呼ぶ)。

② ①に含まれない文字は，“%”の後に文字符号の 2 けたの 16 進表記を続けた 3 文字に変換する。例えば，文字符号 0x5E の文字は“%5E”と変換する。

(3) 関数 URLEncode の仕様は，次のとおりである。

形式： `void URLEncode( unsigned char *input,`
`unsigned char *output )`

引数： `input` 変換前の文字列が格納されている文字型配列へのポインタ

`output` 変換後の文字列を格納する文字型配列へのポインタ

(4) output が指す配列は，変換後の文字列を格納するのに十分な領域が確保されているものとする。

(5) 例えば，“Hi!”という文字列パラメタを変換した結果は図のとおりとなる。

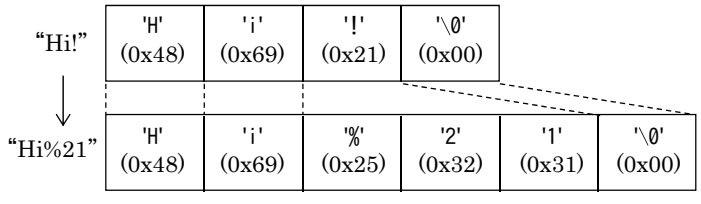


図　文字列パラメタの変換例

(6) 次の関数があらかじめ用意されている。

形式： `int replaceChar( unsigned char c )`

引数： c

引数： 文字 c が無変換文字の場合は 0 を，それ以外の場合は 1 を返す。

アドバイス

C 言語のプログラムでは，＼が登場しますが，これは¥と同じ文字を指しています。画面表示や印刷の際に使用されるフォントによって，＼になったり¥になったりしますが，本試験では，＼で表記されるため，本書でも＼と表記しています。

演習問題① 4けたの数字を当てるゲーム

次のCプログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

プログラムが生成した各けたの数字が異なる4けたの目標数を，なるべく少ない回数の推測で当てるゲーム GuessNumber である。回答者は，各けたが異なる4けたの推測数を入力し，当たらなかった場合，次の情報を得ることができる。

① 目標数に含まれていて，けたも一致している数字の個数（以下，Hit数という）

② 目標数に含まれているが，けたが一致していない数字の個数（以下，Blow数という）

例えば，目標数が“1632”で推測数が“7613”の場合，“6”はけたも一致しているのでHit数が1，“1”と“3”はけたが一致していないのでBlow数が2となる。

(1) 目標数はプログラムが自動的に生成する。生成した数字列は char 型の配列 target に格納される。推測数を文字列として標準入力から読み込み，char 型の配列 num に格納したうえで照合する。目標数と完全に一致するまで繰り返す。

(2) 次の関数が用意されている。

```
void createRandomNumber(char[] target)
```

機能：4けたの目標数を文字列として左のけたから順に配列 target に格納する。各けたの数字はすべて異なっている。

```
int isValidNumber(char[] num)
```

機能：文字列 num 中の各文字がすべて異なる数字のとき TRUE を，それ以外のとき FALSE を返す。

C 言語実行環境の構築の仕方

C 言語のソースプログラムを入力するためにはテキストエディタ（Windows ではメモ帳で代用）が必要です。また，コンパイルするためにはコンパイラが必要です。

(1) テキストエディタ

有名なフリーソフトに「terapad」や「サクラエディタ」があります。ここでは terapad を例にしてインストール方法を紹介します。

TeraPad は Windows 7／8 上で動作するシンプルなテキストエディタです。HTML，Perl，PHP，CSS，Ruby，C/C++，VB，Delphi などの各種言語の編集に対応しています。メモ帳とは違い，Tab，全角空白，改行，[EOF]マークなどの特殊文字が画面に表示されます。また，インデントを自動で付けてくれる便利な機能も備わっています。

TeraPad は，寺尾氏のサイト

http://www5f.biglobe.ne.jp/~t-susumu/library/tpad.html

から入手できます。同サイトの中ほどに，ダウンロード用のリンクがあり，ここから，インストーラのダウンロードができます。

ダウンロードしたファイルを実行し，画面の指示に従って 次へ をクリックしていきます。インストールが完了したら，デスクトップにできた「TeraPad」というショートカット，または，スタート画面の「TeraPad」からプログラムが起動できるようになります。

(2) コンパイラ

Windows 上の主なフリーソフトのコンパイラとして，Visual C++と Borland C++があります。Visual C++は統合開発環境で，エディタやデバッガの機能も備わっています。最新版の Visual C++ Express Edition が無料の体験版として配布されています。

ここではインストールと実行がやさしい Borland C++ Compiler 5.5 を例に挙げて，ダウンロードとインストール方法を述べます。

なお下記ホームページには，「Borland C++ Compiler 5.5 は，個人のお客様の使用を前提としております」と明記されていますので，注意を守ってください。

① Borland C++ Compiler 5.5 のダウンロード

次の URL から開始します。

http://www.embarcadero.com/jp/products/cbuilder/free-compiler

下へスクロールしていくと「C++コンパイラの入手方法」と書かれた箇所があります。そこに記載された文章の中の「こちら」と書かれているリンクをクリックしてください。

すると登録フォームが表示されます。必要事項を入力してから画面最下部にある 登録実行 と書かれたボタンを押してください。

ダウンロード画面が表示されるので，画面に表示された Download Now と書かれたボタンを押すとダウンロードが自動的に開始されます。このファイルは，任意の場所に保存してかまいません。ダウンロードは以上で終了です。

② C++ Compiler のインストール

ダウンロードしたファイル「freecommandlinetools2」はパスワード付きで圧縮されています。先ほどの登録フォームで入力したメールアドレス宛にパスワードが送付されますので，そのパスワードを使ってファイル

表中の使用例では宣言を省略しています。次のような宣言がされているとします。

```
int i, n;
long l;   (clock_t,time_t,size_t は long と同じ型)
double d,x,y,z;
int a[N],b[N];   (N は適当な大きさの整数)
char *s,*t, *buf;  または  char s[N], t[N], buf[N];
 (N は適当な大きさの整数)
int *p;  または  char *p, *q;   (ポインタ型変数)
FILE *fp;   (ファイルポインタ)
```

関数名	**abs**	ヘッダファイル	stdlib.h
プロトタイプ	int abs(int n);		
引数	n：正負の整数		
返却値	n の絶対値		
機能	整数の絶対値を返す。		
使用例	abs(-10) ⇒ 10		

関数名	**atof**	ヘッダファイル	stdlib.h
プロトタイプ	double atof(const char *s);		
引数	s：浮動小数点数の文字列表現		
返却値	変換した浮動小数点数		
機能	文字列 s を浮動小数点数に変換する。		
使用例	d = atof("12.345");		

関数名	**atoi**	ヘッダファイル	stdlib.h
プロトタイプ	int atoi(const char *s);		
引数	s：整数の文字列表現		
返却値	変換した整数		
機能	文字列 s を整数に変換する。		
使用例	i = atoi("123");		

関数名	**atol**	ヘッダファイル	stdlib.h
プロトタイプ	long atol(const char *s);		
引数	s：長整数の文字列表現		
返却値	変換した長整数		
機能	文字列 s を長整数に変換する。		
使用例	l = atol("100000");		

関数名	**bsearch**	ヘッダファイル	stdlib.h
プロトタイプ	void *bsearch(void *key, void *base, size_t n, size_t size, int (*cmp)(const void *arg1, const void *arg2));		
引数 1	key：探索するデータ		
引数 2	base：昇順に整列済みの配列の先頭アドレス		
引数 3	n：配列の要素数		
引数 4	size：配列の要素のバイト数		
引数 5	cmp：比較関数 （次の返却値をもつ関数でなければならない） 第 1 引数>第 2 引数：正 第 1 引数=第 2 引数：0 第 1 引数<第 2 引数：負		
返却値	見つかったとき：マッチした項目へのポインタ，見つからなかったとき：NULL		
機能	1 要素 size バイトの配列 base[0]～base[n-1]の中から key で示したデータを探索し結果を返す。		

使用例	int cmpfunc(const void *s1, const void *s2)
	{
	return strcmp((char *)s1, (char *)s2);
	}
	⋮
	char *data[10];
	char key[20];
	bsearchkey, data, 10, sizeof(char *), cmpfunc);		

関数名	**calloc**	ヘッダファイル	stdlib.h
プロトタイプ	void *calloc(size_t n, size_t size);		
引数 1	n：割り当てる変数の個数		
引数 2	size：変数に割り当てられる領域のバイト数		
返却値	割り当てたメモリの先頭アドレス		
機能	size バイトのメモリを n 個分割り当て，その先頭アドレスを返す。割り当てた領域は 0 でクリアされる。		
使用例	p = (int *) calloc(100, sizeof(int)); …… int 型変数 100 個分のメモリを割り当てる。		

関数名	**ceil**	ヘッダファイル	math.h
プロトタイプ	double ceil(double x);		
引数	x：実数値		
返却値	x の小数点以下を切り上げた結果の double の値		
機能	x の小数点以下を切り上げる。		
使用例	ceil(1.4) ⇒ 2.0		

関数名	**clearerr**	ヘッダファイル	stdio.h
プロトタイプ	void clearerr(FILE *fp);		
引数	fp：ファイルポインタ		
返却値	なし		
機能	ファイル fp のエラーフラグと EOF フラグをクリアする。		
使用例	clearerr(fp);		

索 引