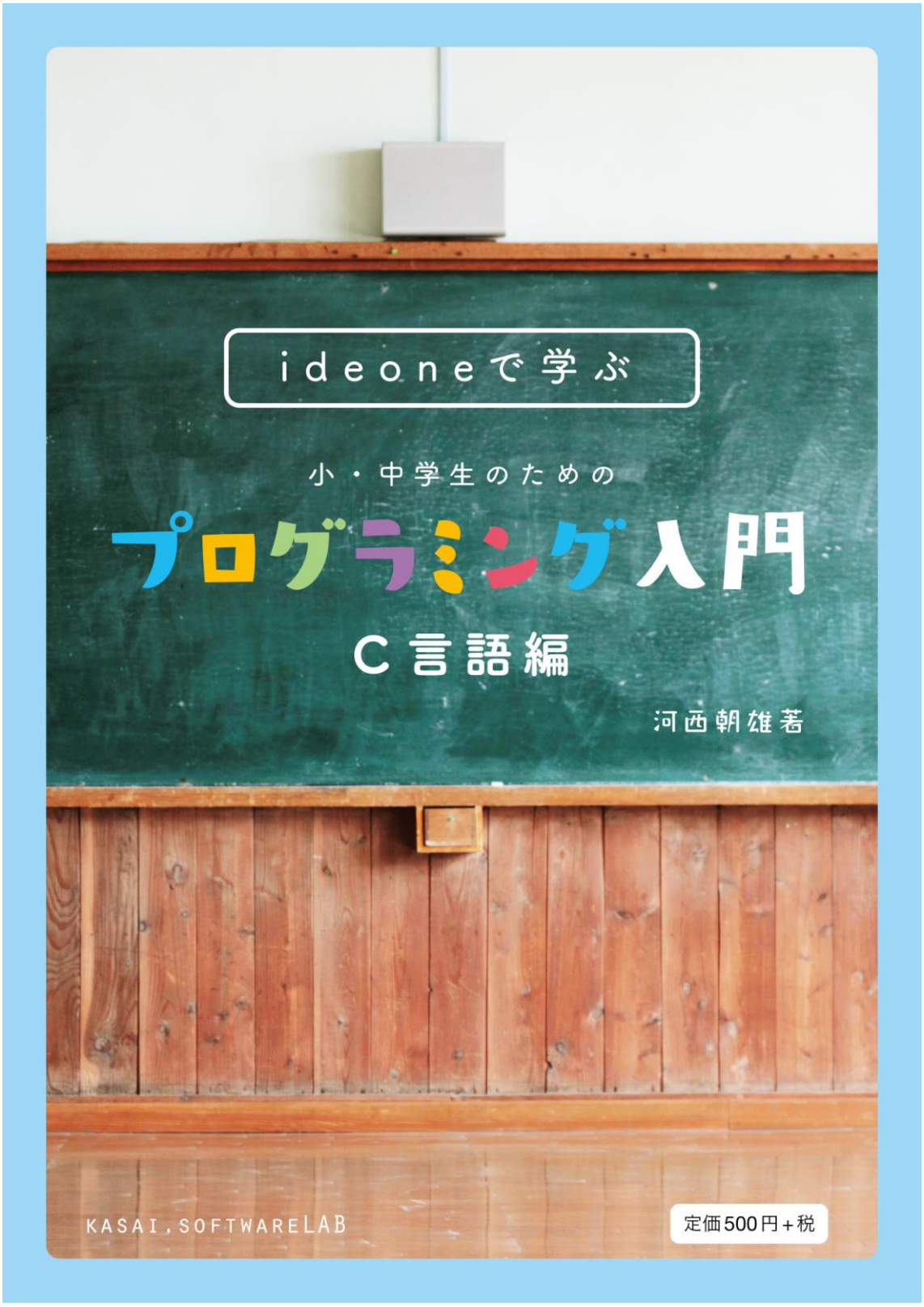

ideoneで学ぶ
小・中学生のための
プログラミング入門
C言語編
河西朝雄著
KASAI,SOFTWARELAB
定価500円+税

ideone で学ぶ

小・中学生のためのプログラミング入門

C 言語編

河西　朝雄著

カサイ．ソフトウエアラボ

はじめに

■ideone(アイディーイー・ワン)とは

・今までのプログラミング学習は Visual Studio や Eclipse などの本格的なプログラミング開発環境をインストールして行っていました。

・この方法ではインストールが煩雑であったり、実務向きの環境なので取扱が難しく、初心者がプログラミングを学ぶには敷居が高いものでした。

・ideone は ideone の Web サイト（https://ideone.com/）にブラウザから入るだけでプログラミングができます。

■Scratch か ideone か

・米国のマサチューセッツ工科大学で子ども向けに開発されたプログラム開発環境として Scratch があります。

・Scratch は操作が視覚的で分かり易いので、小・中学生にプログラミングの基礎教えるのに最適です。

・Scratch では「スクリプト」というブロックをつなぎ合わせることでプログラムを書きます。ブロックをつなぎ合わせるというのは視覚的で分かり易いですが、生産性が悪く、汎用性に欠けます。

・本格的にプログラミングを行うには、多くの「命令」を覚えて、コードを打ち込み、プログラムを書かなければなりません。こうした目的で、小・中学生にプログラミング学習を行うのに ideone は最適です。

■C か Java か？

・ideone では C、C++、Java、PHP などの多くのプログラミング言語が使用できますが、小・中学生にプログラミング学習を教えるには C か Java が適しています。

・C は 1972 年頃 AT&T ベル研究所の D.M.リッチーにより UNIX システムの記述用言語として開発されました。現在のプログラミング言語の基礎となる構造化プログラミングという概念を導入しました。C にオブジェクト指向を導入したものが C++です。

・Java は C++をベースにして 1995 年に Sun Microsystems 社が開発したオブジェクト指向言語です。

・実際のプログラミング開発現場では C は主に制御系に使われ、Java は Web アプリ系に使われています。

・C と Java は if や for などの基本的な命令は同じです。初心者向けには C がシンプルで最も適していますが、オブジェクト指向の基礎まで学ぶには Java が良いでしょう。

■プログラミングを学ぶことによって何が得られるのか？

・プログラミングという作業を通じて問題解決能力、創造力、論理的思考が高まります。

・そうした力を養うことにより学習に対する意欲の向上や、自ら進んで粘り強く学ぼうとする学習態度が養われます。

・これはアクティブラーニングという考え方にも通じます。
・プログラミング能力は、これからの情報化社会を生きて行くための「教養」です。読み書きの能力を身に付けること同様に重要です。

■教育現場でも注目が集まっています

・プログラミングは、中学校の「技術・家庭」や高校の教科「情報」で教えられていますがあまり普及していません。
・アメリカをはじめ世界各国ではプログラミング教育の機運が高まっており、日本もその流れに乗ろうとしています。
・そこで、次期学習指導要領が始まる２０２０年度からコンピューターのプログラミング教育を小・中学校で必修にする計画です。

■2021 年度からの大学入試に対応

・2021 年度から大学入試制度が大きく変わります。
・習得した「知識・技能」の確認に留まらず、「知識・技能を活用する力」、つまり総合的な「思考力・表現力」を問う検査が重要になります。
・こうした思考力・表現力を身につけるにはプログラミングは最適な教材と言えます。

本書は小・中学生が ideone を使って C プログラミングを学ぶためのもので、以下のような８回の構成になっています。

第１回　プログラムを作る上での基本事項
第２回　プログラムを制御する流れ制御文Ⅰ
第３回　データをまとめて管理する配列
第４回　プログラムを制御する流れ制御文Ⅱ
第５回　同じ処理をひとつにまとめる関数
第６回　math.h を使っていろいろな計算
第７回　ソートに挑戦
第８回　データ構造とアルゴリズム

小・中学生にとってはかなり高度な内容になっていますが、分かり易く説明しているので理解できるはずです。
学校での学習に使う場合は総合学習の時間やクラブの時間を使って４５分×８回で学習できるようになっています。パソコン教室などのテキストとしても使えます。
本書を通して多くの小・中学生が、これからの情報化社会を生きて行くための「教養」としてのプログラミング能力を身につけてもらえたら幸いです。

2016 年 7 月　河西　朝雄

目次

第１回

プログラムを作る上での基本事項

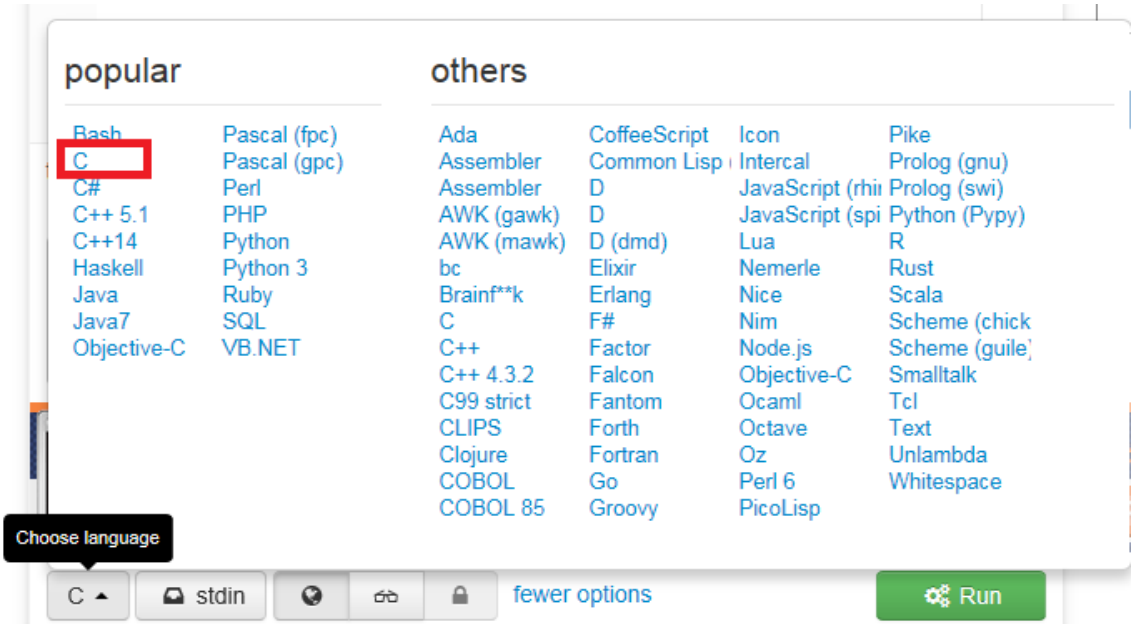
C プログラムを作る上で必要な基礎事項を説明します。
プログラム作成でまず必要なのはプログラムの処理結果を表示する方法です。
C ではこれを「printf」という命令で行います。
次にデータを格納するための容器として変数という概念があり、
データを計算するためには加減乗除を行う演算子を使います。
また、データには整数型、実数型、文字型などのデータ型があります。

１．ideone で C プログラムを作るには

ideone(アイディーイー・ワン)は、ブラウザでプログラムを入力すると実行した結果を返してくれるサイトです。C や Java などの多くのプログラミング言語が使用できます。

https://ideone.com/

■言語として「C」を選択します

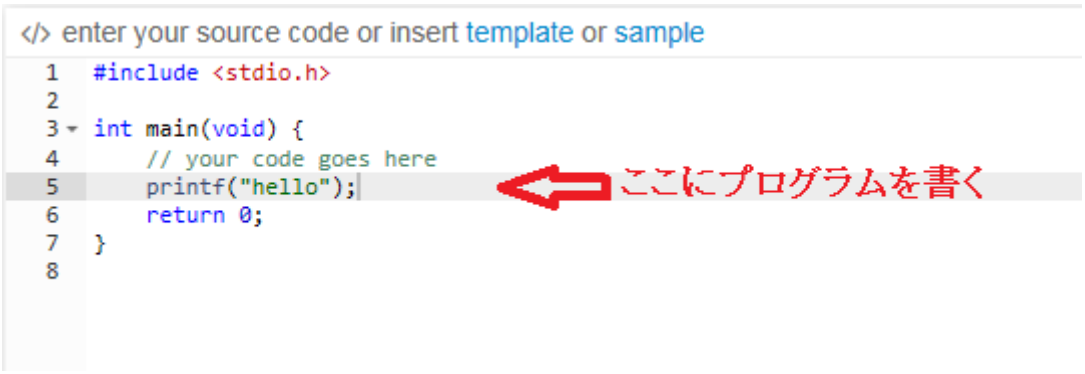


■C プログラムのスケルトン（骨格）が表示されます

画面に「hello」と表示するプログラムとして

printf("hello");

を追加します。



・プログラムを実行するには「Run」をクリックします

以下のように結果が表示されます。

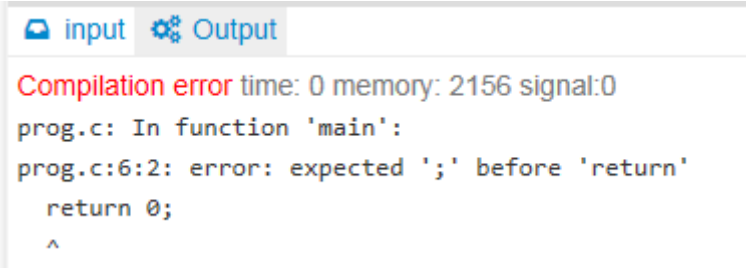
・プログラムを修正するには「edit」を選択します

■２回目以後は「ideone it!」で実行します

結果は「Output」画面に表示されます。

■エラーがあると以下のようにメッセージが表示されます

このエラーは「セミコロン（；）がない」というエラーです。



第２回

プログラムを制御する流れ制御文Ｉ

プログラムは通常、記述してある順序に上から下に向かって実行されていきます。
こうしたプログラムの流れを制御する文を流れ制御文と言います。
流れ制御文には同じ処理ををくり返すくり返し文（反復文）や
条件により処理内容を選択する条件判断文（選択文）などがあります。
今回はくり返しを行う for 文と条件判断を行う if else 文について説明します。
また、for 文の中に for 文が二重にはいったくり返し構造についても説明します。
プログラミングの基本はこの流れ制御文をよく理解することから始まります。

１．くり返しを行う for 文

for 文はくり返しを行う制御文で、くり返し回数があらかじめ決まっているくり返し（所定回反復）に使います。くり返す文が１つのときは｛｝を省略することができますが、安全のために省略しない方が良いでしょう。

```
        初期値
        ↓   ↓終了条件
for (i=0;i<10;i++) {
   文           ↑増減式
}
```

■インクリメント演算子

上の文は i を 0 から始めて＋１しながら 10 未満の間、{と} で囲まれたブロックをくり返します。従って i の値は 0,1,2,...8,9 と変化します。++はインクリメント演算子で、i++で i の値を＋１します。i の値を－１するにはデクリメント演算子を使って i- -とします。

■ループ変数

for 文で使用している変数は一般のものと同じですが、ループ変数と呼ぶことがあります。ループ変数の値は for 文が管理しているため、for ループの中でユーザーが勝手にその値を変えてはいけません。ループ変数は int 型以外にも、char 型や double 型を使用できます。また、変化させる値も+1 だけでなくあらゆる値を指定できます。

「文例」

```
for (i=5;i<10;i++)
  i を 5 から始め、i を+1 しながら 10 未満（9 まで）の間くり返す
for (i=10;i>0;i--)
  i を 10 から始め、i を-1 しながら 0 より大きい間くり返す
for (x=0.0;x<=360.0;x=x+20.0)
  x を 0.0 から始め、+20 しながら 360.0 以下の間くり返す
```

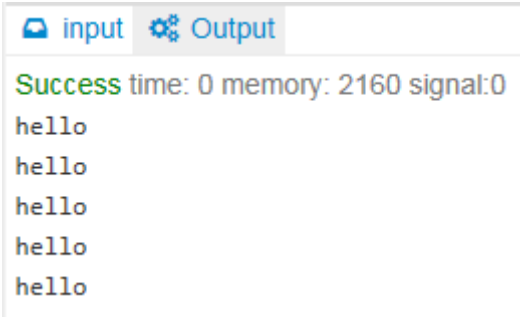
例題 2-1　くり返し表示

「hello」を５回表示します。

・プログラム

```
#include <stdio.h>

int main(void) {
	int i;
	for (i=1;i<=5;i++){
		printf("hello¥n");
	}
	return 0;
}
```

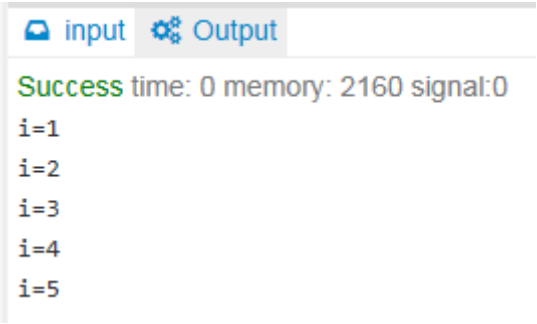
・実行結果



練習問題 2-1　ループ変数の値の変化

ループ変数 i の値がどのように変化するのか画面に表示しなさい。

・実行結果



・プログラム

```
#include <stdio.h>

int main(void) {
	int i;
	for (i=1;i<=5;i++){
		printf("i=%d¥n",i);
	}
	return 0;
}
```

第３回
データをまとめて管理する配列

多量のデータは一般変数では管理できないので、
配列というデータ構造を用います。
配列は同じ型のデータを集め、配列データ全体を指す総称名（配列名）と
何番目のデータかを示す要素番号（添字：そえじ）で管理できるようにしたものです。
添字に変数を指定できるので、
for 文のループ変数と組み合わせると効率的な処理が行えます。
配列の添字が１つのものを１次元配列、
添字が２つのものを２次元配列といいます。

１．１次元配列の基礎

たとえば 10 個のデータがあったとき、これを入れるための変数として、a0,a1,a2,...といった単純変数を用いて、

a0=0;

a1=1;

などとしていたのでは大変なことは明らかです。このような場合に配列を使います

■配列の宣言

int 型の配列は以下のように宣言します。

```
    配列名
   ↓
int a[10];
↑      ↑確保する要素の個数
型名
```

これにより a[0],a[1],...、a[9]という 10 個の要素からなる配列が用意されます。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

配列の開始要素は 0 番から始まることに注意して下さい。このため 10 個の要素を宣言した場合 a[0]～a[9]の 10 個が用意され、a[10]という要素は確保されません。

■配列要素の参照

配列は配列全体を示す配列名と、配列の要素番号を示す添字で管理されます。添字は[]で囲みます。a[0]は 0 番目の要素、a[1]は 1 番目の要素を示します。i 番目の要素は a[i]と書けます。このように[]内に i などの変数を書けることが配列の最大の特徴です。

a[0]～a[9]のすべての要素を 0 にするプログラムは for 文を使って以下のように書けます。

```
for (i=0;i<10;i++) {
    a[i]=0;
}
```

i の値が 0～19 まで変化する間に、a[i]=0 は a[0]=0,a[1]=0,a[2]=0,...a[9]=0 が順次おこなわれることになります。

■配列の宣言時の初期化

配列の各要素に初期値を与えるには、配列の宣言時に｛｝内に初期化するデータをカンマ（,）で区切って指定します。初期化データがある場合はnew演算子は使わなくて良いです。

int a[]={0,1,2,・・・ 9};

これで、a[0]に 0、a[1]に 1,...a[9]に 9 が初期化されます。

■配列の要素数

配列の要素数は sizeof 演算子を使って調べることができます。たとえば int a[10];と宣言された配列に対し sizeof(a)/sizeof(a[0])は 10 を返します。従って、

```
for (i=0;i<10;i++) {
    a[i]=0;
}
```

というプログラムは

```
int N= sizeof(a)/sizeof(a[0]);
for (i=0;i<N;i++) {
    a[i]=0;
}
```

のように、プログラム中に 10 のような特定な値を埋め込まずに記述できます。プログラムは多少煩雑になりますが、配列の要素数が変わってもプログラムを変更する必要がありません。sizeof(a)は配列全体のサイズ、sizeof(a[0])は配列要素 1 つのサイズを返すので、sizeof(a)/sizeof(a[0])は配列の要素数となります。

例題 3-1　配列要素の表示

配列 a[]の各要素の値を表示します。

・プログラム

```
#include <stdio.h>

int main(void) {
	int a[]={10,5,13,20,35};
	int i,N=sizeof(a)/sizeof(a[0]);
	for (i=0;i<N;i++){
		printf("a[%d]=%d¥n",i,a[i]);
	}
	return 0;
}
```

・実行結果

練習問題 3-1　char*型配列

文字列型配列 girl に名前データを格納し、各要素の値を表示しなさい。

■char*型

「char*」は文字列を扱う型です。文字列定数は”（二重引用符）で囲みます。

・実行結果

・プログラム

```
#include <stdio.h>

int main(void) {
	char* girl[]={"Ann","Lisa","Amica","Elice","Machilda"};
	int i,N=sizeof(girl)/sizeof(girl[0]);
	for (i=0;i<N;i++){
		printf("girl[%d]=%s¥n",i,girl[i]);
	}
	return 0;
}
```

第４回
プログラムを制御する流れ制御文Ⅱ

C は構造化プログラミングを行う上で必要な以下の
近代的流れ制御構造を完備しています。
・選択（if else 文）
・複数方向選択（switch case 文、else if 文）
・所定回反復（for 文）
・前判定反復（while 文）
・後判定反復（do while 文）
・分岐（break 文）
すでに２章で if else 文と for 文は解説しました。
今回は switch case 文、else if 文、while 文、break 文について解説します。
do while 文は使用頻度が少ないので本書では扱いません。
また、goto 文、continue 文などもありますがこれらはさらに使用頻度は少ないです。

１．while 文

for 文はくり返し回数が決まっているくり返しに使いますが、while 文はくり返し回数が決まっていないくり返しに使います。

```
while (式) {
    文
}
```

while 文では、まず（）内の式が評価され、それが真なら、{} で囲まれた文を実行し、再び式を評価して同じことをくり返します。つまり（）内の式が真の間 {} 内の文をくり返し、式が偽になると while 文から抜けます。

「文例」

```
i=0;
while (a[i]!=-999) {   ←配列 a[i]の値が-999 でない間くり返す

    i++;
}
```

例題 4-1　夢の積立方式

今日は１円預金し、明日は倍の２円、その翌日はさらに倍の４円というように、前日の倍の金額を預金していったら何日で 100 万円を越すかを調べます。日を day、その日の預金額を money、預金総額を sum で管理します。

・プログラム

```
#include <stdio.h>

int main(void) {
    int day=1,money=1,sum=0;
    while (sum<1000000) {
        sum+=money;
        printf("%d 日目:%d¥n",day,sum);
        money*=2;
        day++;
    }
```

```
    return 0;
}
```

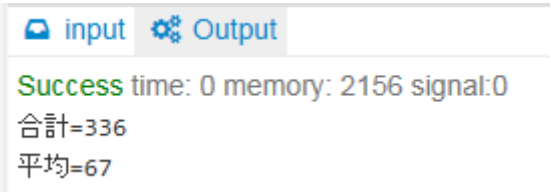
・実行結果

```
stdout
1日目:1円
2日目:3円
3日目:7円
4日目:15円
5日目:31円
6日目:63円
7日目:127円
8日目:255円
9日目:511円
10日目:1023円
11日目:2047円
12日目:4095円
13日目:8191円
14日目:16383円
15日目:32767円
16日目:65535円
17日目:131071円
18日目:262143円
19日目:524287円
20日目:1048575円
```

練習問題 4-1　配列の合計と平均

配列 a[]の合計と平均を求めなさい。配列の終わりの印に「-1」が置いてあるものとします。

・実行結果



・プログラム

```
#include <stdio.h>

int main(void) {
	int a[]={56,78,30,90,82,-1};
	int i=0,sum=0;
	while (a[i]!=-1) {
		sum+=a[i];
		i++;
	}
	printf("合計=%d¥n",sum);
	printf("平均=%d¥n",sum/i);
	return 0;
}
```

第５回

同じ処理をひとつにまとめる関数

関数はある処理を行う機能単位で、
ユーザーは引数というデータをこの関数に渡して目的の処理を行います。
すでに printf 関数などを使用しましたが、
こうした関数は、ライブラリで定義されていて stdio.h をインクルード
することで使用できるようになります。
今回はユーザーが独自の関数を定義し使用する方法を説明します。

１．ユーザー定義関数

長いプログラムを良く見ると、それはいろいろな仕事の寄せ集めでできています。そのような個々の仕事に分割した単位を機能単位と呼びます。この機能単位をメインプログラム（メインルーチン）と別な部分に置き、必要に応じて呼び出すようにしたものを一般にサブルーチンといいます。C ではこのサブルーチンを関数というもので構成します。
printf 関数はライブラリで定義されていて stdio.h をインクルードすることで使用できるようになります。
このようにライブラリとして提供される関数の他にユーザーは独自の関数を定義し、使用することができます。

■関数の定義と引数

ユーザー定義関数の定義と呼び出しは次のように行います。一般の C 処理系では関数定義部を関数の呼び出しより後に置く場合は関数プロトタイプ宣言を置かなければなりません。そこで本書では main 関数より先にユーザー定義関数を置くことにします。ただし、ideone 環境では関数定義部が関数呼び出しより後に定義されていても関数プロトタイプ宣言を置かなくてもよいです。

```
型 関数名（型 仮引数,型 仮引数,・・・）　←関数の定義
{

    return 式;
}

int main(void) {
    関数名（実引数,実引数,・・・）；　←関数の呼び出し
}
```

関数は処理結果を呼び出し元に返すという機能を持っているので、その関数がどのような型のデータを返すものなのかを関数の型として明確に宣言しておかなければなりません。「return 式;」の式の値が関数の結果として呼び出し元に返されます。この値を戻り値と呼びます。従って式の値と関数の型は一致していなければなりません。
呼び出し元と関数との間でデータのやり取りを行うには引数（ひきすう）というものを用います。呼び出し側で指定する引数を実引数（じつひきすう）、関数の定義側で指定する引数を仮引数（かりひきすう）と呼びます。仮引数は関数定義部においてその型を宣言しておく必要があります。実引数には、定数、変数、式（定数や変数を演算子で結んだもの）

を指定できますが、仮引数はデータを受け取る器なので変数しか指定できません。また、対応する実引数と仮引数は同じ名前を付ける必要はありませんが、型は一致していなければなりません。

■void 型

関数に何かの処理を依頼するだけで、戻り値を必要としない関数もあります。このような戻り値を持たない関数は void 型として定義し、「return 式;」も置きません。

```
void disp(...)
{

}
```

例題 5-1　指定した大きさのの３角形を描く関数

指定した大きさのの３角形を描く関数 triangle を作ります。

・プログラム

```
#include <stdio.h>

void triangle(int n)
{
	int i,j;
	for (i=1;i<=n;i++) {
		for (j=0;j<i;j++) {
			printf("☆");
		}
		printf("¥n");
	}
}

int main(void) {
	triangle(3);
	triangle(4);
	return 0;
}
```

・実行結果

練習問題 5-1　指定した大きさ、指定した文字で３角形を描く関数

指定した大きさ、指定した文字で３角形を描く関数 triangle を作りなさい。

・実行結果

・プログラム

```
#include <stdio.h>

void triangle(int n,char *s)
{
	int i,j;
	for (i=1;i<=n;i++) {
		for (j=0;j<i;j++) {
			printf("%s",s);
		}
		printf("¥n");
	}
}

int main(void) {
	triangle(3,"☆");
	triangle(4,"★");
	return 0;
}
```

第６回
math.h を使っていろいろな計算

制御文、演算子、変数といった C の基本的な言語仕様だけでは
プログラムはできません。
そこで各種機能を実現するプログラムを、
標準ライブラリ関数として提供しています。
たとえば画面への出力は printf 関数を用います。
ANSI C では 146 種類の関数を用意しています。
これらの関数はカテゴリ別に分類され、
それぞれのヘッダー・ファイル（~.h）で宣言されています。
今回は標準ライブラリ関数の基本的な利用法として
math.h で定義されている数値計算関数と
stdlib.h で定義されている乱数を使って
いろいろな計算をしてみます。

１．math.h で定義されている関数

数値計算関数は math.h でプロトタイプ宣言されているので、使用する場合はこのファイルをインクルードしなければなりません。

```
#include <stdio.h>
#include <math.h>

int main(void) {
    pow(2,n);

}
```

math.h で宣言されている主な関数として以下のものがあります。関数の引数、戻り値はすべてのものが double 型です。

関数	機能
ceil(x)	x を下回らない最小整数
cos(x)	コサイン
exp(x)	指数 e^x
floor(x)	x を上回らない最大整数
log(x)	自然対数 $\log_e x$
pow(x, y)	べき乗 x^y
sin(x)	サイン
sqrt(x)	平方根
tan(x)	タンジェント

第 7 回
ソートに挑戦

配列 a[]の i 番目の要素を a[i]とすると、一つ前の要素は a[i-1]、
一つ後ろの要素は a[i+1]で表すことができます。
これらの隣合う項を隣接項と言います。
隣接項を操作することで、
配列の内容を小さい順または大きい順に並べ換えることができます。
これをソートと言います。

１．配列要素をローテイト

配列データを右にシフトし、最右端のデータを先頭に持ってくるような移動を右ローテイトと言います。

1	4	8	5	6	7	2

↓右ローテイト

2	1	4	8	5	6	7

最右端のデータを作業用変数tに保存しておき、右端から始めてa[i]にa[i-1]をコピーし、最後に先頭要素に変数 t の内容をコピーします。

例題 7-1　右ローテイト

配列 a[]の内容を右ローテイトします。

・プログラム

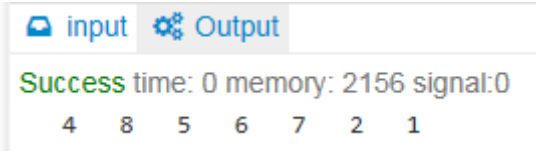
```
#include <stdio.h>

int main(void) {
	int a[]={1,4,8,5,6,7,2};
	int i,t,N=sizeof(a)/sizeof(a[0]);
	t=a[N-1];
	for (i=N-1;i>0;i--) {
		a[i]=a[i-1];
	}
	a[0]=t;
	for (i=0;i<N;i++) {
		printf("%4d",a[i]);
	}
	return 0;
}
```

・実行結果

練習問題 7-1　左ローテイト

配列 a[]の内容を左ローテイトしなさい。

・実行結果



・プログラム

```
#include <stdio.h>

int main(void) {
	int a[]={1,4,8,5,6,7,2};
	int i,t,N=sizeof(a)/sizeof(a[0]);
	t=a[0];
	for (i=0;i<N-1;i++) {
		a[i]=a[i+1];
	}
	a[N-1]=t;
	for (i=0;i<N;i++) {
		printf("%4d",a[i]);
	}
	return 0;
}
```

第８回
データ構造とアルゴリズム

プログラムを使って問題を解くための論理または手順を
アルゴリズム（algorithms：算法）と言います。
プログラム処理では多量のデータを扱うことが多く、
この場合、取り扱うデータをどのようなデータ構造（data structure）にするかで、
問題解決のアルゴリズムが異なってきます。
データ構造とアルゴリズムは密接な関係にあり、
良いデータ構造を選ぶことが良いプログラムを作ることにつながります。
プログラムの世界に特有なアルゴリズムとして再帰という考え方があります。
再帰の代表的な例として階乗やハノイの塔を紹介します。
データ構造としては、決定木を用いた食文化判定や性格判定プログラムを紹介します。

１．再帰を用いたハノイの塔の解法

再帰という考え方は人間の一般的な感覚からは縁遠いものですがプログラムの世界では重要な考え方です。再帰の例として階乗とハノイの塔について説明します。

■再帰とは

再帰的（リカーシブ）な構造とは、自分自身（n 次）を定義するのに、自分自身より 1 次低い部分集合（n-1 次）を用い、さらにその部分集合は、より低次の部分集合を用いて定義するということをくり返す構造です。このような構造を一般に再帰と呼んでいます。
再帰を用いると、複雑なアルゴリズムを明解に記述することができます。
プログラムにおける再帰は次のような構造をしています。

```
void hanoi(n,…) // 再帰的関数の定義
{
    if (脱出条件)       // 再帰からの脱出口
        return;
     ・
     ・
    hanoi(n-1,…);     // 再帰呼び出し
}
・
・
    hanoi(3,…);       // 再帰関数の最初の呼び出し
```

ある関数の内部で、再び自分自身を呼び出すような構造の関数を再帰的関数と呼び、関数内部で再び 1 次低い自分自身を呼び出すことを再帰呼び出し（リカーシブ・コール）と呼びます。
再帰では一般に再帰からの脱出口を置かなければいけません。これがないと、再帰呼び出しが永遠に続いてしまうことになります。

■ハノイの塔

ハノイの塔とは次のようなパズルゲームです。

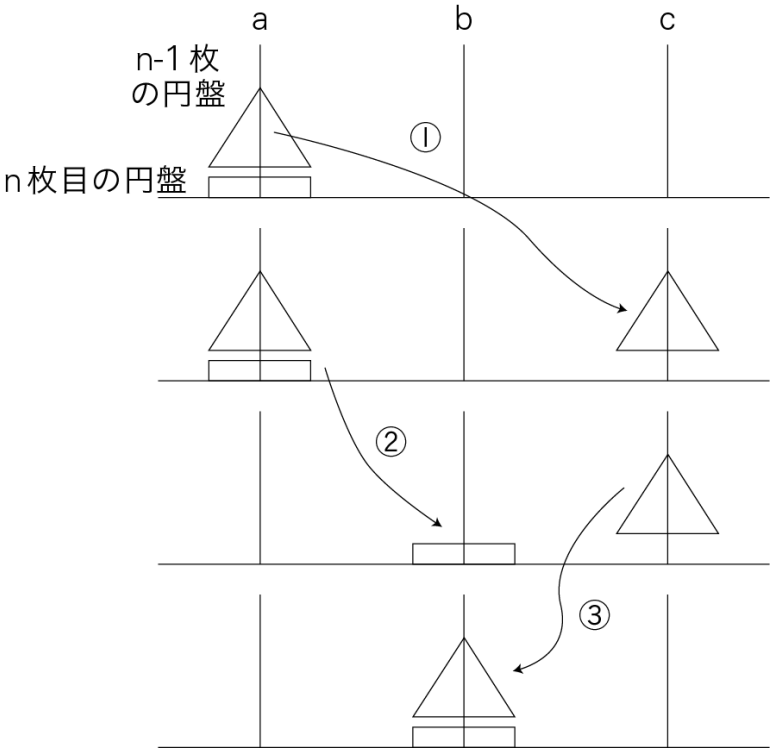
「３本の棒 a、b、c がある。棒 a に、中央に穴の空いた n 枚の円盤が大きい順に積まれている。これを 1 枚ずつ移動させて棒 b に移す。ただし、移動の途中で円盤の大小が逆に積まれてはならない。また、棒 c は作業用に使用するものとする。」

n 枚の円盤を a⇒b に移す作業は、次のような作業に分解できます。①と③の作業が再帰的

な作業となります。

①a の n-1 枚の円盤を a⇒ c に移す。
②n 枚目の円盤を a⇒b に移す。
③c の n-1 枚の円盤を c⇒b に移す。

ハノイの塔



例題 8-1　ハノイの塔

n 枚の円盤を a⇒b に移動する手順を画面に表示する関数 hanoi を作ります。プログラムにおける①~③の意味は以下です。

①n-1 枚の円盤を作業用棒 b を使って棒 a から棒 c に移動する再帰呼び出し。

②n 枚目の円盤を引数 a が示す棒から引数 b が示す棒に移す実際の処理。

③n-1 枚の円盤を作業用棒 a を使って棒 c から棒 b に移動する再帰呼び出し。

・プログラム

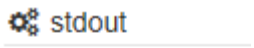
```
#include <stdio.h>

void hanoi(int n,char a,char b,char c)
{
    if (n>0) {
        hanoi(n-1,a,c,b);        // ①
        printf("%d 番の板%c->%c に移動¥n",n,a,b); // ②
        hanoi(n-1,c,b,a);        // ③
    }
}

int main(void) {
    hanoi(3,'a','b','c');
    return 0;
}
```

・実行結果

n=3 の場合

```
stdout
1番の板a->bに移動
2番の板a->cに移動
1番の板b->cに移動
3番の板a->bに移動
1番の板c->aに移動
2番の板c->bに移動
1番の板a->bに移動
```

n=4 の場合




```
1番の板a->cに移動
2番の板a->bに移動
1番の板c->bに移動
3番の板a->cに移動
1番の板b->aに移動
2番の板b->cに移動
1番の板a->cに移動
4番の板a->bに移動
1番の板c->bに移動
2番の板c->aに移動
1番の板b->aに移動
3番の板c->bに移動
1番の板a->cに移動
2番の板a->bに移動
1番の板c->bに移動
```

練習問題 8-1　階乗

n!（n の階乗）は次のように定義できます。

n!=n・(n-1)!　　n>0

0!=1

これは次のような意味に解釈できます。

・n!を求めるには 1 次低い(n-1)!を求めてそれに n を掛ける
・(n-1)!を求めるには 1 次低い(n-2)!を求めてそれに n-1 を掛ける
　・
　・
・1!を求めるには 0!を求め、それに 1 を掛ける
・0!は 1 である

階乗を求める再帰関数 kaijyou を作り、0!～12!までを求めて表示しなさい。

・実行結果


```
0!=1
1!=1
2!=2
3!=6
4!=24
5!=120
6!=720
7!=5040
8!=40320
9!=362880
10!=3628800
11!=39916800
12!=479001600
```

・プログラム

```
#include <stdio.h>

int kaijyou(int n)
{
	if (n==0)
		return 1;
	else
		return n*kaijyou(n-1);
}

int main(void) {
	int n;
	for (n=0;n<=12;n++) {
		printf("%d!=%d¥n",n,kaijyou(n));
	}
	return 0;
}
```

著者略歴

河西 朝雄（かさいあさお）

山梨大学工学部電子工学科卒（1974 年）。長野県岡谷工業高等学校情報技術科教諭、長野県松本工業高等学校電子工業科教諭を経て、現在は「カサイ．ソフトウエアラボ」代表。

「主な著書」

「入門ソフトウエアシリーズ C 言語」、「同シリーズ Java 言語」、「同シリーズ C++」、「入門新世代言語シリーズ VisualBasic4.0」、「同シリーズ Delphi2.0」、「やさしいホームページの作り方シリーズ HTML」、「同シリーズ JavaScript」、「同シリーズ HTML 機能引きテクニック編」、「同シリーズホームページのすべてが分かる事典」、「同シリーズ i モード対応 HTML と CGI」、「同シリーズ i モード対応 Java で作る i アプリ」、「同シリーズ VRML2.0」、「チュートリアル式言語入門 VisualBasic.NET」、「はじめての VisualC#. NET」、「C 言語用語辞典」ほか（以上ナツメ社）

「構造化 BASIC」、「Microsoft Language シリーズ Microsoft VISUAL C++初級プログラミング入門上、下」、「同シリーズ VisualBasic 初級プログラミング入門上、下」、「C 言語によるはじめてのアルゴリズム入門」、「Java によるはじめてのアルゴリズム入門」、「VisualBasic によるはじめてのアルゴリズム入門」、「VisualBasic6.0 入門編、中級テクニック編、上級編」、「Internet Language 改訂新版シリーズ ホームページの制作」、「同シリーズ JavaScript 入門」、「同シリーズ Java 入門」、「New Language シリーズ標準 VisualC++プログラミングブック」、「同シリーズ標準 Java プログラミングブック」、「VB.NET 基礎学習 Bible」、「原理がわかるプログラムの法則」、「プログラムの最初の壁」、「河西関数：C 言語プログラム学習の方程式」、「基礎から学べる VisualBasic2005 標準コースウエア」、「基礎から学べる JavaScript 標準コースウエア」、「基礎から学べる C 言語標準コースウエア」、「基礎から学べる PHP 標準コースウエア」、「なぞりがき C 言語学習ドリル」、「C 言語　標準ライブラリ関数ポケットリファレンス[ANSI C,ISO C99 対応]」、「 C 言語　標準文法ポケットリファレンス[ANSI C,ISOC99 対応]」、「[標準] C 言語重要用語解説 ANSI C / ISO C99 対応」ほか（以上技術評論社）

「電子書籍：カサイ．ソフトウエアラボ」

「Android プログラミング Bible 初級 基礎編」、「Android プログラミング Bible 中級 Android 的プログラミング法」、「Android プログラミング Bible 上級　各種処理」、「Android プログラミング完全入門」、「iPhone&iPad プログラミング Bible[上] 」、「iPhone&iPad プログラミング Bible[下] 」、「JavaScript によるはじめてのアルゴリズム入門 」、「Web アプリ入門（HTML5+JavaScript）」、「HTML5 を使った JavaScript 完全入門 」、「Scratch プログラミング入門」

ideone で学ぶ
小・中学生のためのプログラミング入門
Java 言語編

2016 年 7 月 1 日　初版　第 1 刷

著者＝河西 朝雄
発行者＝河西 朝雄
発行所＝カサイ．ソフトウエアラボ
長野県茅野市ちの 813　TEL.0266-72-4778

表紙デザイン＝河西 朝樹

定価＝500 円＋税