JavaScriptによる
はじめての
アルゴリズム入門

河西　朝雄 著

KASAI.SOFTWARELAB

定価 540円（税込）

JavaScript によるはじめてのアルゴリズム入門の特徴

１．HTML5+JavaScript で標準的なグラフィックス処理が可能

C のグラフィックスは各処理系に依存しますが、HTML5+JavaScript では標準のグラフィックスメソッドを使用できます。これにより「はじめてのアルゴリズム入門」シリーズでは主に C、Java、VisualBasic などでしかプログラムを書けませんでしたが JavaScript でも可能になりました。

・HTML5 の canvas タグに対し JavaScript からグラフィックスメソッドを使用して簡単にグラフィックス描画ができるようになりました。プログラムの先頭に、以下を定番で置いておけば、「context」に対し、context.fillText(テキスト,x,y);でテキストが context.moveTo(x1,y1);context.lineTo(x2,y2);で直線が描けます。この他にも各種グラフィックスメソッドが用意されています。これにより表示位置を制御したいテキスト表示や８章のグラフィックスが簡単に表現できるようになりました。

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";
```

・以下のようなライブラリ「turtle.js」を作っておくことで、簡単にタートルグラフィックスを行うことができます。

```
// ---------------------------------
// *     タートル・グラフィックス     *
// ---------------------------------

    var LPX,LPY,          // 現在位置
        Angle;            // 現在角
    var WX1,WY1,WX2,WY2,  // ワールド座標
```

```
    VX1,VY1,VX2,VY2, // ビュー座標
    FACTX,FACTY;     // 倍率
var context;

function View(x1,y1,x2,y2)
{
    context.beginPath();   // クリップ処理が必要なかったり、
    context.moveTo(x1,y1); // クリップがうまくいかない場合は
    context.lineTo(x1,y2); // context.メソッドはコメントアウト
    context.lineTo(x2,y2);
    context.lineTo(x2,y1);
    context.lineTo(x1,y1);
    context.save();
    context.clip();
    context.restore();
    context.beginPath();

    VX1=x1;VY1=y1;VX2=x2;VY2=y2;
    FACTX=(VX2-VX1)/(WX2-WX1);
    FACTY=(VY2-VY1)/(WY2-WY1);
}
function Window(x1,y1,x2,y2)
{
    WX1=x1;WY1=y1;WX2=x2;WY2=y2;
    FACTX=(VX2-VX1)/(WX2-WX1);
    FACTY=(VY2-VY1)/(WY2-WY1);
}

function move(leng)
{
    var x,y,x1,y1,x2,y2;
    x=LPX+leng*Math.cos(Math.PI*Angle/180);
    y=LPY+leng*Math.sin(Math.PI*Angle/180);
    x1=(LPX-WX1)*FACTX+VX1;
    y1=(WY2-LPY)*FACTY+VY1;
    x2=(x-WX1)*FACTX+VX1;
```

```
    y2=(WY2-y)*FACTY+VY1;
    context.moveTo(x1,y1);
    context.lineTo(x2,y2);
    context.stroke();
    LPX=x;LPY=y;
}

function moveto(x,y)
{
    var x1=(LPX-WX1)*FACTX+VX1;
    var y1=(WY2-LPY)*FACTY+VY1;
    var x2=(x-WX1)*FACTX+VX1;
    var y2=(WY2-y)*FACTY+VY1;
    context.moveTo(x1,y1);
    context.lineTo(x2,y2);
    context.stroke();
    LPX=x;LPY=y;
}

function pset(x,y)
{
    var x1=(x-WX1)*FACTX+VX1;
    var y1=(WY2-y)*FACTY+VY1;
    context.fillRect(x1,y1,1,1);
    LPX=x1;LPY=y1;
}

function turn(angle)
{
    Angle+=angle;
    Angle=Angle%360;
}

function setangle(angle)
{
    Angle=angle;
```

```
}

function setpoint(x,y)
{
    LPX=x;LPY=y;
}

var canvas = document.getElementById("canvas");
if(canvas.getContext){
    context = canvas.getContext("2d");
    context.strokeStyle ="blue";
    context.fillStyle ="blue";
    Window(0,0,600,600);
    View(0,0,600,600);
}
```

２．簡易的なクラス表現ができます

以下のようなクラスを定義しておけば、a[i].name や a[i].age のような表現ができますので、オブジェクト指向的な表現ができ、アルゴリズム学習に適しています。

```
function girl(name,age) // girl クラス
{
    this.name=name; // フィールド
    this.age=age;
}
```

３．ブラウザとテキストエディタさえあれば簡単にプログラムできます

JavaScript はブラウザに内蔵されているので、特別なソフトが必要でなくインストールなどの作業が必要ないので、初心者には敷居が低いと思います。

４．C 言語で学ぶより学習効果が得られます

学校の授業などでプログラムの入門として C 言語を学ぶ例が多いでようです。C の言語仕様にはグラフィックス処理は含まれておらず､各C処理系でのサポートとなりますので、統一性がありません。

JavaScript を使えば、HTML5 から使用できるようになった<canvas>要素（タグ）へのグラフィックス処理を簡単に行うことができます。

JavaScript の基本言語仕様は C 言語をベースにしているので、C 言語の文法を学んだのと同等の効果が得られます。

５．パソコン、スマートフォン、タブレットで動作します

本書のプログラムはパソコン上で作成していますが、スマートフォンやタブレット上でも動作します。スマートフォンやタブレットで動作できればより身近なものに感じられはずです。

本書は以下の書籍を JavaScript 版にしたもので、JavaScript のソースコードのみを提供しています。各例題、各練習問題の解説は以下の書籍を参考にしてください。

「C 言語によるはじめてのアルゴリズム入門」技術評論社、河西朝雄著、1992 年
「改訂第 3 版 C 言語によるはじめてのアルゴリズム入門」技術評論社、河西朝雄著、2008 年
「Java によるはじめてのアルゴリズム入門」技術評論社、河西朝雄著、2001 年
「Visual Basic によるはじめてのアルゴリズム入門」技術評論社、河西朝雄著、1999 年
「N_{88}-BASIC によるはじめてのアルゴリズム入門」技術評論社、河西朝雄著、1993 年

目次

第１章　ウォーミング・アップ

プログラミング技術に深みを持たせるためには、異なる視点でのアルゴリズム（algorithms）をできるだけ多く学ぶことが大切である。

本書は第２章以後に各分野別に、その分野での典型的なアルゴリズムを説明している。

この章では、そういった分野とは離れた比較的簡単なアルゴリズムを学び、基礎的な力をつける。

1-1　漸化式

例題１　${}_nC_r$を求める

n 個の中から r 個を選ぶ組み合わせの数 ${}_nC_r$を求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ　Ｐゴシック'";

// --------------------------------
// *     漸化式（nCr の計算）      *
// --------------------------------

    var n,r;
    for (n=0;n<=5;n++) {
        for (r=0;r<=n;r++) {
            context.fillText(n+"Ｃ"+r+"="+combi(n,r),r*60,n*20+20);
        }
    }

    function combi(n,r)
    {
        var i,p=1;
        for (i=1;i<=r;i++) {
            p=p*(n-i+1)/i;
        }
        return p;
    }
```

```
</script>
</body>
</html>
```

■実行結果

```
0C0=1
1C0=1  1C1=1
2C0=1  2C1=2  2C2=1
3C0=1  3C1=3  3C2=3  3C3=1
4C0=1  4C1=4  4C2=6  4C3=4  4C4=1
5C0=1  5C1=5  5C2=10 5C3=10 5C4=5  5C5=1
```

練習問題１　Horner（ホーナー）の方法

多項式 $f(x)=a_nx^n+a_{n-1}x^{n-1}+・・・+a_1x+a_0$ の値を Horner の方法を用いて計算する。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ　Ｐゴシック'";

// ----------------------
// *     Horner の方法     *
// ----------------------

    var x;
    var a=new Array(1,2,3,4,5); // 係数
    for (x=1;x<=5;x++) {
```

```
        context.fillText("fn("+x+")="+fn(x,a,4),0,x*20);
    }

    function fn(x,a,n)
    {
        var i,p;
        p=a[n]
        for (i=n-1;i>=0;i--) {
            p=p*x+a[i];
        }
        return p;
    }

</script>
</body>
</html>
```

■実行結果

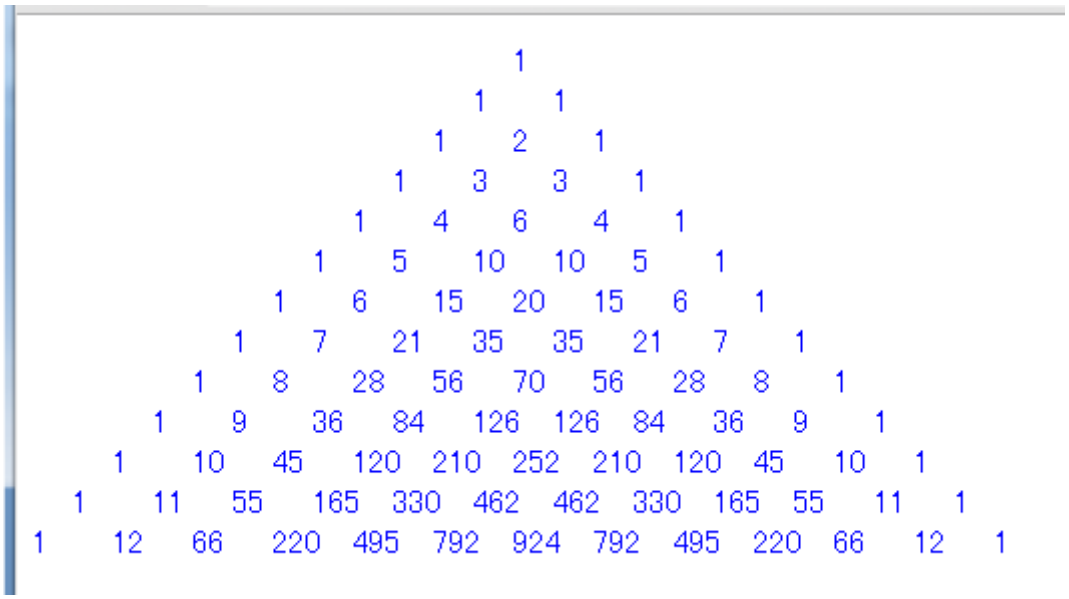
```
fn(1)=15
fn(2)=129
fn(3)=547
fn(4)=1593
fn(5)=3711
```

参考　Pascal の三角形

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";

// -------------------------
// *     Pascal の三角形     *
// -------------------------

    var n,r,t,N=12;
    for (n=0;n<=N;n++){
        for (r=0;r<=n;r++) {
            context.fillText(combi(n,r),(N-n)*20+r*40,n*20+20);
        }
    }

    function combi(n,r)
    {
         var i,p=1;
         for (i=1;i<=r;i++) {
             p=p*(n-i+1)/i;
         }
         return p;
    }

</script>
</body>
</html>
```

■実行結果


1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
1 11 55 165 330 462 462 330 165 55 11 1
1 12 66 220 495 792 924 792 495 220 66 12 1

1-2　写像

例題 2　ヒストグラム

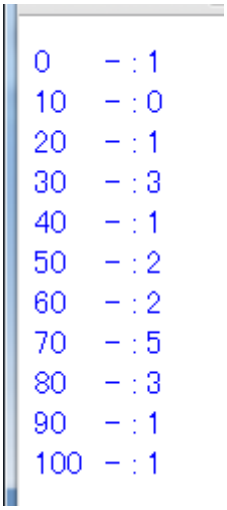
0～100 点までの得点を 10 点幅で区切って（0～9、10～19、・・・、90～99、100 の 11 ランク)、各ランクの度数分布（ヒストグラム）を求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";

// ----------------------------------
// *     度数分布（ヒストグラム）      *
// ----------------------------------

    var a=new Array(35,25,56,78,43,66,71,73,80,90,
                    0,73,35,65,100,78,80,85,35,50);
    var histo=new Array(11);
    var i,rank,N=20;

    for (i=0;i<=10;i++) {
        histo[i]=0;
    }
    for (i=0;i<N;i++){
        rank=parseInt(a[i]/10);              // 写像
        if (0<=rank && rank<=10) {
            histo[rank]++;
        }
    }
```

```
    for (i=0;i<=10;i++) {
        context.fillText(i*10,0,i*20+20);
        context.fillText(" - : "+histo[i],30,i*20+20);
    }

</script>
</body>
</html>
```

■実行結果



練習問題２　暗号

暗号文字"KSOIDHEPZ"を解読する。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ　Ｐゴシック'";
```

```
// ----------------------
// *     暗号の解読     *
// ----------------------

    var table="QWERTYUIOPASDFGHJKLZXCVBNM?";
    var ango="KSOIDHEPZ";
    var i,index,result="";
    for (i=0;i<ango.length;i++){
        if ("A"<=ango.charAt(i) && ango.charAt(i)<="Z")
            index=ango.charCodeAt(i)-"A".charCodeAt(0);
        else
            index=26;
        result+=table.charAt(index);
    }
    context.fillText(result,0,20);

</script>
</body>
</html>
```

■実行結果

参考　暗号化の方法の例

・イクスクルーシブオア（排他的論理和）により暗号

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
```

```
    context.font = "16px 'ＭＳ Ｐゴシック'";

// ----------------------------------------
// *      イクスクルーシブ OR による暗号      *
// ----------------------------------------

    var ango="FK@HUNSOJ";
    var i,result="",code,KCode=0x7;
    for (i=0;i<ango.length;i++){
        code=ango.charCodeAt(i)^KCode;
        result+=String.fromCharCode(code);
    }
    context.fillText(result,0,20);

</script>
</body>
</html>
```

■実行結果

1-3　順位付け

例題３　単純な方法

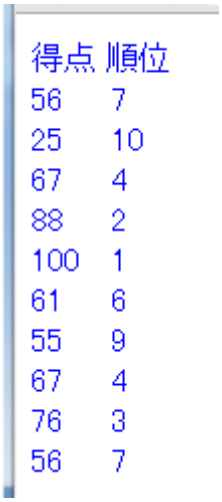
たとえば、テストの得点などのデータがあったとき、その得点の順位を求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";

// --------------------
// *     順位付け      *
// --------------------

    var i,j,N=10;
    var a=new Array(56,25,67,88,100,61,55,67,76,56);
    var juni=new Array(N);

    for (i=0;i<N;i++){
        juni[i]=1;
        for (j=0;j<N;j++){
            if (a[j]>a[i])
                juni[i]++;
        }
    }

    context.fillText("得点　順位",0,20);
    for (i=0;i<N;i++){
        context.fillText(a[i],0,i*20+40);
```

```
        context.fillText(juni[i],40,i*20+40);
    }

</script>
</body>
</html>
```

■実行結果



練習問題 3-1　例題 3 の改良版

例題 3 の順位は求めるアルゴリズムではデータが n 個の場合、繰り返し回数は n^2 となるため、データ数が増えると処理に時間がかかってしまう。そこで、繰り返し回数を減らすようにした順位付けアルゴリズムを考える。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ　Ｐゴシック'";
```

```
// --------------------------------
// *     順位付け  (改良版)      *
// --------------------------------

    var i,N=10,Max=100,Min=0;
    var a=new Array(56,25,67,88,100,61,55,67,76,56);
    var juni=new Array(Max+2);

    for (i=Min;i<=Max;i++) {
        juni[i]=0;
    }
    for (i=0;i<N;i++) {
        juni[a[i]]++;
    }
    juni[Max+1]=1;
    for (i=Max;i>=Min;i--) {
        juni[i]=juni[i]+juni[i+1];
    }

    context.fillText("得点  順位",0,20);
    for (i=0;i<N;i++){
        context.fillText(a[i],0,i*20+40);
        context.fillText(juni[a[i]+1],40,i*20+40);
    }

</script>
</body>
</html>
```

■実行結果

```
得点 順位
56   7
25   10
67   4
88   2
100  1
61   6
55   9
67   4
76   3
56   7
```

練習問題 3-2　負のデータ版

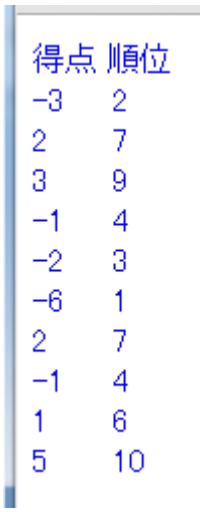
ゴルフ(Golf)のスコアのように小さい値の方が順位が高い場合の順位付けについて考える。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ　Ｐゴシック'";

// ------------------------------------
// *      順位付け　(負のデータ版)       *
// ------------------------------------

    var i,N=10,Max=36,Min=-20,
        Bias=1-Min; // 最小値を配列要素の 1 に対応させる
    var a=new Array(-3,2,3,-1,-2,-6,2,-1,1,5);
    var juni=new Array(Max+Bias+1);
```

```
    for (i=Min+Bias;i<=Max+Bias;i++) {
        juni[i]=0;
    }
    for (i=0;i<N;i++) {
        juni[a[i]+Bias]++;
    }
    juni[0]=1;
    for (i=Min+Bias;i<Max+Bias;i++) {
        juni[i]=juni[i]+juni[i-1];
    }

    context.fillText("得点　順位",0,20);
    for (i=0;i<N;i++){
        context.fillText(a[i],0,i*20+40);
        context.fillText(juni[a[i]+Bias-1],40,i*20+40);
    }

</script>
</body>
</html>
```

■実行結果



1-4　ランダムな順列

例題４　ランダムな順列（効率の悪い方法）
1～N の値を１回使ってできるランダムな順列をつくる。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";

// ------------------------------------------
// *     ランダムな順列（効率の悪い方法）      *
// ------------------------------------------

    var N=20,i,j,flag,s="";
    var a=new Array(N+1);

    a[1]=irnd(N);
    for (i=2;i<=N;i++){
        do {
            a[i]=irnd(N);
            flag=0;
            for (j=1;j<i;j++){
                if (a[i]==a[j]){
                    flag=1;
                    break;
                }
            }
        }while (flag==1);
```

```
    }
    for (i=1;i<=N;i++) {
        s+=a[i]+" , "; // 結果を文字列に連結していく
    }
    context.fillText(s,0,20);

    function irnd(n){ // １～ｎの乱数
        return parseInt(Math.random()*n+1);
    }

</script>
</body>
</html>
```

■実行結果

```
20,2,7,10,13,18,19,4,11,16,3,1,8,9,12,14,6,5,17,15,
```

練習問題４　ランダムな順列（改良版）

例題４のアルゴリズムを改良した効率のよいアルゴリズムを考える。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";

// ------------------------------------
// *      ランダムな順列（改良版）       *
// ------------------------------------
```

```
    var N=20,i,j,d,s="";
    var a=new Array(N+1);

    for (i=1;i<=N;i++) {
        a[i]=i;
    }

    for (i=N;i>1;i--){
        j=irnd(i-1);
        d=a[i];a[i]=a[j];a[j]=d;
    }

    for (i=1;i<=N;i++) {
        s+=a[i]+" , ";
    }
    context.fillText(s,10,20);

    function irnd(n){ //  1～nの乱数
        return parseInt(Math.random()*n+1);
    }

</script>
</body>
</html>
```

■実行結果

```
9 , 20 , 12 , 15 , 14 , 16 , 6 , 11 , 5 , 19 , 7 , 18 , 8 , 17 , 1 , 2 , 13 , 10 , 4 , 3 ,
```

1-5　モンテカルロ法

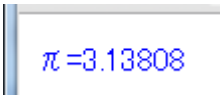
例題５　π を求める

モンテカルロ法を用いて π の値を求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";

// ----------------------------------------
// *     モンテカルロ法によるπの計算      *
// ----------------------------------------

    var i,x,y,sum=0,pai,N=50000;
    for (i=1;i<N;i++) {
        x=Math.random(); // ０～１の乱数
        y=Math.random();
        if (x*x+y*y<1) {
            sum++;
        }
    }
    pai=4*sum/N;
    context.fillText("π="+pai,0,20);

</script>
</body>
</html>
```

■実行結果



練習問題５　面積を求める

モンテカルロ法を用いて、楕円の面積を求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ　Ｐゴシック'";

// ----------------------------------------
// *      モンテカルロ法による面積の計算      *
// ----------------------------------------

    var i,x,y,sum=0,s,N=50000;
    for (i=1;i<N;i++) {
        x=2*Math.random();
        y=Math.random();
        if (x*x/4+y*y<=1) {
            sum++;
        }
    }
    s=4*(2.0*sum/N);
    context.fillText("楕円の面積="+s,0,20);

</script>
</body>
</html>
```

■実行結果

```
楕円の面積=6.276
```

1-6　ユークリッドの互除法

例題 6　ユークリッドの互助法（その 1）

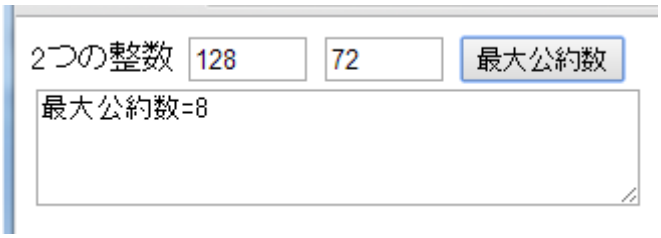
2 つの整数 m,n の最大公約数をユークリッド（Euclid）の互助法を用いて求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
2 つの整数
<input id="t1" type=text" size="4" />
<input id="t2" type=text" size="4" />
<input type="button" value="最大公約数" onClick="calc()"><br />
<textarea id="result" rows="4" cols="40"></textarea>
<script type="text/javascript">

// ----------------------------------------
// *     ユークリッドの互除法（減算）      *
// ----------------------------------------

    function calc()
    {
        var m=parseInt(document.getElementById("t1").value); // m!=n の比較用に
        var n=parseInt(document.getElementById("t2").value); // 整数化する必要が
ある
        while (m!=n){
            if (m>n)
                m=m-n;
            else
                n=n-m;
        }
        var result=document.getElementById("result");
        result.value="最大公約数="+m;
    }
```

```
</script>
</body>
</html>
```

■実行結果



練習問題 6　ユークリッドの互助法（その 2）

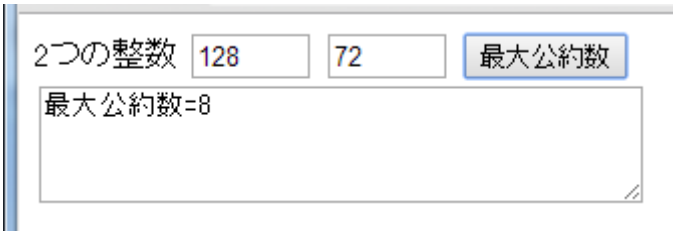
m と n の差が大きいときは減算（m-n）の代わりに剰余（m%n）を用いた方が効率がよい。この方法で m と n の最大公約数を求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
2 つの整数
<input id="t1" type=text" size="4" />
<input id="t2" type=text" size="4" />
<input type="button" value="最大公約数" onClick="calc()"><br />
<textarea id="result" rows="4" cols="40"></textarea>
<script type="text/javascript">

// ----------------------------------------
// *      ユークリッドの互除法（剰余）       *
// ----------------------------------------

    function calc()
    {
        var m=parseInt(document.getElementById("t1").value); // m!=n の比較用に
        var n=parseInt(document.getElementById("t2").value); // 整数化する必要が
ある
        var k;
```

```
        do {
            k=m % n;
            m=n;
            n=k;
        } while(k!=0);
        var result=document.getElementById("result");
        result.value="最大公約数="+m;
    }

</script>
</body>
</html>
```

■実行結果



1-7　エラトステネスのふるい

例題 7　素数の判定

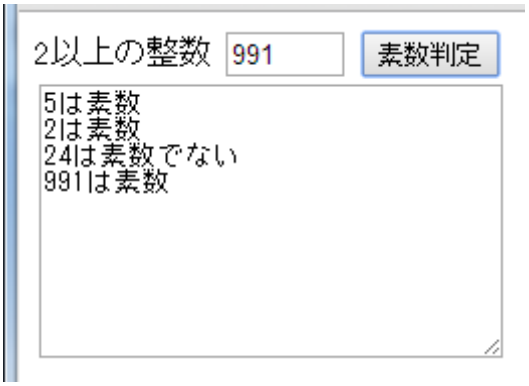
n が素数か否か判定する。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
2 以上の整数
<input id="t1" type=text" size="4" />
<input type="button" value="素数判定" onClick="calc()"><br />
<textarea id="result" rows="10" cols="30"></textarea>
<script type="text/javascript">

// ----------------------
// *     素数の判定     *
// ----------------------

    function calc()
    {

        var n=parseInt(document.getElementById("t1").value);
        var result=document.getElementById("result");
        var i,Limit;
        if (n>=2){
            Limit=parseInt(Math.sqrt(n));
            for (i=Limit;i>1;i--){
                if (n%i ==0)
                    break;
            }
            if (i==1)
                result.value+=n+"は素数¥n";
            else
                result.value+=n+"は素数でない¥n";
```

```
        }
    }

</script>
</body>
</html>
```

■実行結果



練習問題 7-1　2～N のすべての素数

2～N までの整数の中からすべての素数を求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";

// ------------------------------------
// *      2～N の中から素数を拾い出す      *
// ------------------------------------

    var i,n,m=0,y=1,Limit,N=1000;
```

```
    var prime=new Array(N/2+1);

    for (n=2;n<=N;n++){
        Limit=parseInt(Math.sqrt(n));
        for (i=Limit;i>1;i--){
            if (n%i==0)
                break;
        }
        if (i==1)
            prime[m++]=n;
    }

    context.fillText("求められた素数",0,20);
    for (i=0;i<m;i++){
        if (i%20==0) y++;
        context.fillText(prime[i],(i%20)*30,y*20);
    }

</script>
</body>
</html>
```

■実行結果

```
求められた素数
2   3   5   7   11  13  17  19  23  29  31  37  41  43  47  53  59  61  67  71
73  79  83  89  97  101 103 107 109 113 127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541
547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659
661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809
811 821 823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941
947 953 967 971 977 983 991 997
```

練習問題 7-2　エラトステネスのふるい

練習問題 7-1 のアルゴリズムでは、繰り返し回数が n√n/2（平均値）となる。もう少し効率的に素数を求める方法として「エラトステネスのふるい」がある。この方法を用いて 2～N の中から素数をすべて求める。

■プログラム

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="600" height="600"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "blue";
    context.font = "16px 'ＭＳ Ｐゴシック'";

// ---------------------------------
// *     エラトステネスのふるい     *
// ---------------------------------

    var i,j,x=0,y=1,Limit,N=1000;
    var prime=new Array(N+1);

    for (i=2;i<=N;i++) {
        prime[i]=1;
    }

    Limit=parseInt(Math.sqrt(N));
    for (i=2;i<=Limit;i++){
        if (prime[i]==1){
            for (j=2*i;j<=N;j++){
                if (j%i==0)
                    prime[j]=0;
            }
        }
    }
```

```
    context.fillText("求められた素数",0,20);
    for (i=2;i<N;i++){
        if (prime[i]==1){
            if (x%20==0) y++;
            context.fillText(i,(x%20)*30,y*20);
            x++;
        }
    }

</script>
</body>
</html>
```

■実行結果

```
求められた素数
2   3   5   7   11  13  17  19  23  29  31  37  41  43  47  53  59  61  67  71
73  79  83  89  97  101 103 107 109 113 127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541
547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659
661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809
811 821 823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941
947 953 967 971 977 983 991 997
```

参考　素因数分解

■プログラム

```
<!DOCTYPE html>
<html>
<body>
整数
<input id="t1" type=text" size="4" />
<input type="button" value="素因数分解" onClick="calc()"><br />
<textarea id="result" rows="10" cols="30"></textarea>
<script type="text/javascript">
```

```
// ----------------------
// *     素因数分解     *
// ----------------------

    function calc()
    {
        var n=parseInt(document.getElementById("t1").value);
        var a=2,s=n+"=";
        while (n>=a*a){
            if (n % a==0){
                s+=a+"*";
                n=n/a;
            }
            else
                a++;
        }
        s+=n+"¥n";
        var result=document.getElementById("result");
        result.value+=s;
    }

</script>
</body>
</html>
```

■実行結果