暗号円盤による暗号化

暗号化とは、通信文を見ても特別の知識なしには読めないようにして、第三者に通信内容を知られないようにする方法です。暗号化する前の文章を平文、暗号化して読めなくした文章を暗号文といいます。

シーザー暗号

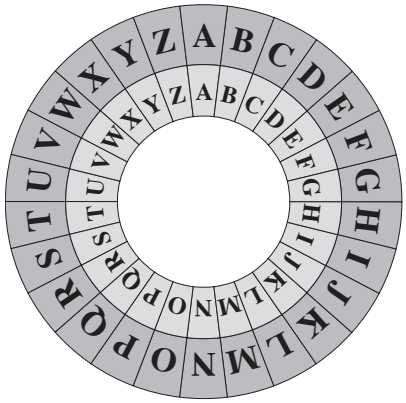
最もシンプルな暗号化の方法として、古代ローマ皇帝シーザーが用いたシーザ暗号があります。シーザー暗号は、換字式暗号で、平文の各文字を辞書順にn文字シフトして暗号文を作る方法です。但し、文字としてアルファベットのみを使う場合は、最後のzの後はa,b,cと巡回的に続いているとして文字をシフトします。例えば、シーザーが用いたn=3の場合、次のような換字のルールで平文の各文字を置き換えて暗号文を作成します。

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

シーザー暗号のような単純な換字式の暗号化は、一度暗号化の方法が知られてしまうと、二度と使うことはできません。

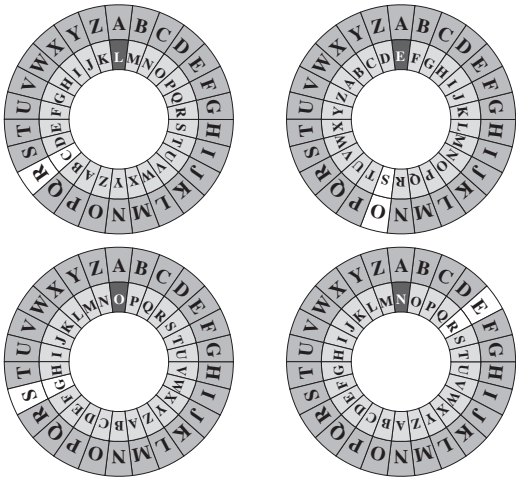
円盤暗号

暗号円盤は、１５世紀のイタリアの建築家レオン・アルベルティが発明したもので、アルファベットが順番に書いてある大小２つの円盤からできています。内側の円盤は回転することができ、これによって平文から暗号文への換字のルールを変えることができます。



◉ 円盤暗号機

円盤暗号で暗号化を行うには、キーワードを一つ決め、外側のAを内側のキーワードの文字に順に合わせることで換字のルールを変えながら、外側の平文の文字を内側の文字に置き換えて行います。例えば、キーワードを「LEON」としたとき、平文「ROSE」を暗号化すると、次の図のように暗号文は「CSGR」となります。



◉ 円盤暗号機の利用例

シーザー暗号は頻度分析などで容易に解読できますが、暗号円盤による暗号はキーワードがわからない限り解読するのは困難です。

暗号円盤による暗号化のプログラムの作成法

ここでは、簡単なオブジェクト指向の入門として、暗号円盤による暗号化のプログラムを作成します。オブジェクト指向プログラミングとは、プログラムを作成する一つの手法で、プログラムをいくつかのオブジェクトという部品で構成する方法です。オブジェクトとは、データとそれを処理する関数（メソッド）をひとまとまりにしたものです。

ここでは、円盤暗号の暗号生成機オブジェクトを生成し、これによって暗号化と複合化を行います。円盤暗号による暗号化と復号化に必要なデータと処理を一つのオブジェクトにまとめることで、アルゴリズムの見通しが良くなり、メンテナンス性も向上します。

オブジェクトを生成する方法として最もよく行われるのは、オブジェクトを定義するコンストラクタ関数を記述し、コンストラクタ関数をnew演算子と共に使ってオブジェクトを生成する方法です。この方法については、第９章で学びます。ここでは、コンストラクタを使わず、その代わり暗号生成機オブジェクトを戻り値として返す関数を定義します。このような関数を、一般に**オブジェクトファクトリ**と呼びます。ファクトリとは「工場」の意味で、ここではオブジェクトを生成する工場を意味します。そこで、暗号生成機オブジェクトファクトリとして、Encryptorという名前の関数を作成します。

Encryptorの作成

Encryptorは暗号生成機オブジェクトを返します。戻り値として返すオブジェクトをobjとします。

```
function Encryptor() {
    var obj = {};
    return obj;
}
```

以下、「var obj;」の後にコードを挿入して、objに必要なプロパティやメソッドを追加していきます。

a. 文字セットを定義する

暗号生成機オブジェクトを作るには、まず、どのような文字セットを扱うかを決めなければなりません。オリジナルの円盤暗号は大文字のアルファベット（A〜Z）のみを文字セットとしていますが、ここでは、小文

字のアルファベット（a〜z）、大文字のアルファベット（A〜Z）、数字（0〜9）、更に半角空白、ピリオド（.）やクエスチョンマーク（?）などのいくつかの記号を追加して文字セットとします。文字セットは、charsプロパティに配列として格納することにします。文字セットは配列リテラルで直接定義することはできますが、ここではfor文で各文字をchars配列に追加します。

```
var N_ALPHABET = 26;    // アルファベットの文字数
var extraCharactors = [" ",".","?","!","%","#","'","&","$","@",":","/"];
                        // アルファベット・数値以外の文字のリスト
obj.chars = [];
// charsリストにアルファベットを追加
for(var c="a".charCodeAt(0); c<="z".charCodeAt(0); c++) {
     obj.chars.push(String.fromCharCode(c));
}
for(var c="A".charCodeAt(0); c<="Z".charCodeAt(0); c++) {
     obj.chars.push(String.fromCharCode(c));
}
// charsリストに数記号を追加
for(var d=0); d<=9; d++) {
     obj.chars.push(d.toString());
}
// charsリストにアルファベット以外の文字を追加
for(var j=0; j<extraCharactors.length; j++) {
     obj.chars.push(extraCharactors[j]);
}
```

更に、配列charsに格納した文字の数をncharsプロパティに求めておきます。

```
obj.nchars = obj.chars.length;
```

b. メソッドの追加

暗号化は、平文の各文字をobj.chars配列の要素番号（0〜obj.nchars-1の整数値）に変換して、整数の演算として行います。そこで、文字を整数値に変換するメソッドが必要となります。numberOf(ch)は、文字chのあるobj.charsの要素番号を返します。見つからなかった場合は、nullを返します。

```
obj.numberOf = function(ch) {
     var code = ch.charCodeAt(0);
     if( code>="a".charCodeAt(0) && code<="z".charCodeAt(0)) {
          return code - "a".charCodeAt(0);
     } else if(code>="A".charCodeAt(0) && code<="Z".charCodeAt(0)) {
          return N_ALPHABET + code - "A".charCodeAt(0);
     } else if(code>="0".charCodeAt(0) && code<="9".charCodeAt(0)) {
          return 2*N_ALPHABET + code - "0".charCodeAt(0);
     } else {
          for(var k=0; k<extraCharactors.length; k++) {
               if( ch == extraCharactors[k] ) {
                    return 2*N_ALPHABET + 10 + k;
               }
          }
          return null;
     }
};
```

円盤暗号は、平文の各文字をキーワードで指定した数だけシフトして得られます。shift(ch,n)メソッドは、文字chを整数値nだけシフトした文字を返します。この際、シフトはnumberOfメソッドで整数値numに変換して行います。関数の中で、thisの値はobjであることに注意してください。

```
obj.shift = function(ch,n) {
     var num = this.numberOf(ch);
     if( num == null ) return ch;
     num = (num + n + this.nchars)%this.nchars;
     return this.chars[num];
};
```

ここで、４行目で整数値をnだけシフトする際、0〜obj.nchars-1の範囲内で巡回するようにthis.ncharsで割った余りを求めています。また、復号化を行うときにnは負となるため、負でも正しくシフトできるように余りを求める前にthis.ncharsを加えています。

以上で、円盤暗号で暗号化を行うデータとメソッドの準備ができました。最後に暗号化を行うメソッドencrypt(text,keyword,encryption)を定義します。encryptは、文字列textをキーワードkeywordで暗号化もしくは復号化した文字列を返します。encryptionがtrueの場合は暗号化を、falseの場合は復号化を行います。

```
obj.encrypt = function(text,keyword,encryption) {
     var cipherText = "";
     var nkey = keyword.length;
     for(var i=0, ikey=0; i<text.length; i++, ikey++) {
          ikey %= nkey;
          var nshift = this.numberOf(keyword[ikey]);
          if(!encryption) nshift *= -1;
          cipherText += this.shift(text[i],nshift);
     }
     return cipherText;
};
```

ここで、暗号化/復号化する際のシフト数nshiftをkeywordの文字から求めています。この際、keywordの何文字目かをあらわす番号ikeyをtextの１文字を処理する毎に1つ増やし、keywordの長さnkeyで割った余りを求めることで、0〜nkey-1の範囲を巡回させています。また、encryptだけで暗号化と複合化を行えるように、encryptionがtrueの場合はkeywordから求めたシフト数そのものを、falseの場合はシフト数に-1を乗じてマイナスの値にしたシフト数で、文字をシフトしています。

以上で、ファクトリ関数Encryptorの完成です。

フォームで平文/暗号文を入力する

最後に、フォームで平文と暗号文を入力できるようにします。更に、暗号化と復号化を行うボタンを用意し、クリックすると暗号化もしくは復号化を行うようにします。