(課題 3) 誤り制御技術

—水平垂直パリティ検査符号とハミング符号—

目的

- 基本的な誤り訂正符号である水平垂直パリティ検査符号を通じて誤り訂正の手法を理解する

関連授業：確率論と情報理論，情報ネットワーク

1　導入

通信・記録において，ディジタル情報は何らかの物理的性質を利用して 0,1 を表現している[*1]．この物理的性質が何らかの外乱によって変化してしまうと情報に誤りが生じる．すなわち，0 が 1 に変化したり，逆に 0 が 1 に変化したりする．デジタル情報に生じてしまった誤りを検出し，訂正する技術が**誤り訂正符号**である．誤り訂正符号は通信・記録システムで広く用いられている実用的な技術であるとともに，代数学を基礎においた数学的にも興味深い分野である．本実験では基礎的な誤り訂正符号について仕組みを調べ，その特性を調査する．

1.1　誤り訂正符号

ここでは，誤り訂正符号の大まかな仕組みを説明する．具体的な構成法については以降の節を参照せよ．

誤り訂正符号では，情報に生じる誤りを発見するための下準備として，何らかの規則にしたがって，情報に冗長性を付加する．この規則のことを**符号**と呼び，冗長性を付加する手続きを**符号化**と呼ぶ．符号化がなされた後に得られる系列は**符号語**と呼ばれ，元の情報に対応している**情報部**と冗長性に対応している**検査部**を連接させた形式で書くことができる．誤り訂正符号では，情報をそのまま伝送する代わりに，符号語を伝送することで誤りに対する耐性をもたせる．

情報の受け手は，受け取った情報系列 (**受信語**) に誤りがないか判定し，誤りがある場合はどのビットに誤りが生じているかを調べ，訂正する必要がある．この手続きのことを**復号**と呼ぶ．言い換えると，復号では受信語をもとに送られた符号語を推定する手続きである．復号で得られる系列は**推定語**と呼ばれる．生じた誤りの個数が少ない場合は正しく元の符号語を推定できるが，生じた誤りの個数が多い場合は送られた符号語とは別のものを推定してしまったり，誤りが発生したという事実しかわからない場合がある．推定語が別の符号語になってしまう事象は**誤訂正**と呼ばれ，誤りが発生したことしかわからない事象は**誤り検出**と呼ばれる．

以上をまとめと，誤り訂正符号の仕組みは図 1 のように与えられる．なお，図中の**通信路**とは情報の送り手 (送信者) と受け手 (受信者) の間に存在する情報の通り道であり，誤りの発生源となっている．

誤り訂正符号の性能は，(i) 訂正できる誤りの個数 (あるいは，検出できる誤りの個数) (ii) 符号語のうち情報部が占める割合 (**符号化率**)[*2]によって決まり，どちらも大きいほうが望ましい．

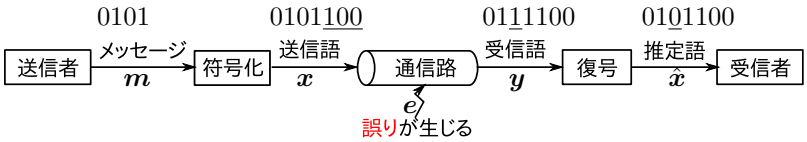

図 1　誤り訂正符号を用いた通信システム

[*1] QR コードの場合を例に取ると，白い四角が 0 を表現し，黒い四角が 1 を表現している．

[*2] より正確には，符号語の長さ (符号長) が N bit 情報部の長さ (情報点数) が K bit のとき，符号化率は K/N で与えられる．

1.2 二元体 (ビット同士の演算法)

以降の説明を数学的に記述するために二元体[*3](ビット同士の演算法) を導入する．集合 $\{0,1\}$ に対して加算 $+$ と乗算 $\cdot$ を以下の表のように定義する．

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

例えば，$0\cdot1+1\cdot1=0+1=1$ であり，$1+0+1+1=1$ となる．加算の表に注目すると排他的論理和 (Exclusive OR: `XOR`, $\oplus$) に等しいことがわかる．同様に，乗算の表は論理積 (`AND`, $\wedge$) に等しい．したがって，ビット同士の演算は論理回路で容易に実装できる．

二元体上のベクトルの加算は，ベクトルの要素毎に加算を行うことで実行できる．すなわち，$\boldsymbol{v}=(v_1,v_2,\ldots,v_n)$ と $\boldsymbol{u}=(u_1,u_2,\ldots,u_n)$ の加算は $\boldsymbol{v}+\boldsymbol{u}=(v_1+u_1,v_2+v_2,\ldots,v_n+u_n)$ となる．例えば，(1,0,0,1,1) と (0,1,0,1,0) の加算は $(1,0,0,1,1)+(0,1,0,1,0)=(1+0,0+1,0+0,1+1,1+0)=(1,1,0,0,1)$ となる．

問題 1 次の計算をせよ．

(a) $1\cdot1+0\cdot0+1\cdot1+0\cdot1$

(b) $(1,0,0,1,0,0)+(0,1,0,1,0,1)$

問題 2 $x_1,x_2,x_3\in\{0,1\}$ とする．$x_1+x_2+x_3=0$ となる (x_1,x_2,x_3) の組を列挙し，その性質を述べよ．同様に，$x_1+x_2+x_3=1$ となる (x_1,x_2,x_3) の組を列挙し，その性質を述べよ．(Hint: 1 の個数で分類)

1.3 水平垂直パリティ検査符号

水平垂直パリティ検査符号は最も理解がしやすい誤り訂正符号である．この節では，符号化・復号の方法を簡単に説明した後に数式による表現法を与え，どのようにプログラムをすれば良いかを示していく．

1.3.1 表を用いた水平垂直パリティ検査符号の符号化・復号

水平垂直パリティ検査符号は表を利用した誤り訂正符号である．表の大きさは，予め受信者と送信者の間で決めておけば，どのような大きさでも構わない．この節では表 1 のような 3 行 4 列の表を利用し，符号化・復号を説明する．

符号化は以下の手順で行われる．

1. **(情報部の埋め込み)** 表 2 の順に情報部を白マスに 1 bit ずつ埋め込む．例えば，010110 という情報の場合は表 3 のようになる．
2. **(検査部の決定 1)** 表の右側のあみかけ部分を各行の 1 の個数が偶数個となるように決める．先ほどの例だと，表 4 のようになる．
3. **(検査部の決定 2)** 表の下側のあみかけ部分を各列の 1 の個数が偶数個となるように決める．先ほどの例だと，表 5 のようになる．
4. **(符号語の作成)** 情報部に Step 2, Step 3 で決定した bit を連接し，符号語にする．先ほどの例では，010110 10 1001 となる．

3×4 の表を使った場合は，表の左上の 2×3 の部分が情報部に対応するので，情報点数 K は 6 になる．また，表全体の大きさが符号長 N に対応するので，$N=12$ になる．したがって，この符号の符号化率は $6/12=1/2$ である．

[*3] 体とは (ざっくり説明すると) 四則演算と分配法則が成立する数の集合である．例えば，有理数・実数は体である．整数は除算の結果が必ずしも整数にならないので，体ではない．詳しくは「代数学」の本を参照すると良い

一般に，$n_1 \times n_2$ の表を使った場合は，$K = (n_1 - 1)(n_2 - 1)$ で $N = n_1 n_2$ であるので，符号化率は $\frac{(n_1-1)(n_2-1)}{n_1 n_2}$ となる．

ここから先は誤りの訂正法について論じていく．誤りが発生しなかった場合，誤りが 1 つだけの場合，誤りが 2 つの場合について順に説明していく．以降の例では表 5 で得られた符号語を送信したと仮定する．

まずは誤りが生じなかった場合について考えよう．誤りが生じなかった場合は，受信語が送信語と全く同じになるため，受信語を表で表すと全ての行と列で 1 の個数が偶数個のままである．したがって，全ての行と全ての列で 1 の個数が偶数個であれば誤りが発生しなかったと判定すれば良い．

次に誤りが 1 つだけ生じた場合を考える．例えば，表 6 (2 行 2 列に誤り) や表 7 (2 行 4 列に誤り) のように誤りが生じたと仮定しよう．このとき誤りを含んだ行と列の 1 の個数が奇数個になっていることがわかる．したがって，水平垂直パリティ検査符号で誤りを訂正するためには，**1 の個数が奇数個になる行と列を見つけ出し，交点が誤りと判定する．**

それでは誤りが 2 つ生じた場合はどのようになるだろうか? 例えば，表 8 のように誤りが生じた場合，2 つの行と 2 つの列で 1 の個数が奇数個になる．全ての行と列で 1 の個数が偶数個になるように 2 箇所誤りを直そうとすると，表 9 と表 10 のように 2 通りの候補が存在する．したがって，誤りの位置を一意に特定できないので，誤りの訂正はできなくなる．但し，1 の個数が奇数個の行または列が存在するため，誤りが生じたという事実だけはわかる．このような状況は**誤り検出**と呼ばれる．

問題 3 表 1 を利用して，101001 を符号化しなさい．
問題 4 表 11, 表 12 のような受信語を得た．誤りがある場合は誤りの場所を指摘しなさい．

1.3.2 数式による表現法

この節では，二元体を利用して前節で紹介した水平垂直パリティ検査符号を数式で表現していく．この節では議論を簡単にするために，3×3 の表を利用した水平垂直パリティ検査符号に限って話を進めていく．

3×3 の表を利用した水平垂直パリティ検査符号の場合, 情報点数は $2\times2 = 4$ となり, 符号長は $3\times3 = 9$ となる．したがって, 検査部の長さは $9-4 = 5$ である．いま, メッセージに対応する 4 bit を (m_1, m_2, m_3, m_4) と書き, 検査部に対応する 5 bit を $(p_1, p_2, p_3, p_4, p_5)$ と書くことにする．すると，符号語 $\boldsymbol{x}$ は $\boldsymbol{x} = (m_1, m_2, m_3, m_4, p_1, p_2, p_3, p_4, p_5)$ と書けることがわかり，表を使うと表 13 の通りになる．

表 1　表の例

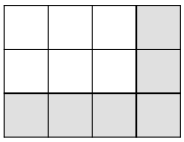

表 2　情報部

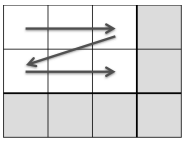

表 3　符号化 (手順 1)

0	1	0	
1	1	0	

表 4　符号化 (手順 2)

0	1	0	1
1	1	0	0

表 5　符号化 (手順 3)

0	1	0	1
1	1	0	0
1	0	0	1

表 6　誤りの例 1

0	1	0	1
1	$\not{1}_0$	0	0
1	0	0	1

表 7　誤りの例 2

0	1	0	1
1	1	0	$\not{0}_1$
1	0	0	1

表 8　誤りの例 3

0	1	$\not{0}_1$	1
1	$\not{1}_0$	0	0
1	0	0	1

表 9　訂正の候補 1

0	1	0	1
1	1	0	0
1	0	0	1

表 10　訂正の候補 2

0	0	1	1
1	0	1	0
1	0	0	1

表 11　問題 4(1)

1	1	1	1
1	1	1	1
0	0	0	0

表 12　問題 4(2)

0	0	1	0
1	1	0	0
0	1	1	0

水平垂直パリティ検査符号の各行，各列の 1 の個数は偶数個であった．問題 2 の結果と照らし合わせると，第一行目の制約は次の式で表すことが出来る．

$$m_1 + m_2 + p_1 = 0 \tag{1}$$

同様にして，全ての行と列の制約を式で表すと以下のとおりになる．

$$\begin{cases} m_1 + m_2 + p_1 = 0 \\ m_3 + m_4 + p_2 = 0 \\ m_1 + m_3 + p_3 = 0 \\ m_2 + m_4 + p_4 = 0 \\ p_1 + p_2 + p_5 = 0 \end{cases} \tag{2}$$

この式を行列を用いて表現すると，

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} (m_1, m_2, m_3, m_4, p_1, p_2, p_3, p_4, p_5)^T = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3}$$

になる (第 i 行目が i 番目の式に対応)．ただし，^{T} は転置を表す記号である．ここで，行列を $\mathbf{H}$ と定義し，**パリティ検査行列**と呼ぶことにする，すなわち，

$$\mathbf{H} := \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

である．すると，式 (3) は $\mathbf{H}\boldsymbol{x}^T = \mathbf{0}$ と表すことができる．

水平垂直パリティ検査符号の各行・各列の制約がどのように式で与えられるかがわかったところで，次に符号化の方法を説明していく．二元体において，$0 + 0 = 0$ と $1 + 1 = 0$ が成立することに注意すると，$p_1 + p_1 = 0$ が成り立つことがわかる．したがって，式 (2) の 1 番上の式は次のように変形できる．

$$\begin{aligned} & m_1 + m_2 + p_1 = 0 \\ \iff & m_1 + m_2 + p_1 + p_1 = p_1 \\ \iff & m_1 + m_2 = p_1 \end{aligned}$$

この式から，検査部の 1 bit 目である p_1 は $m_1 + m_2$ を計算することで得られることがわかる．同様にして，式 (2) の他の式も変形すると，以下のとおりになる

$$\begin{cases} m_1 + m_2 = p_1 \\ m_3 + m_4 = p_2 \\ m_1 + m_3 = p_3 \\ m_2 + m_4 = p_4 \\ p_1 + p_2 = p_5 \end{cases} \tag{5}$$

この式を利用すれば $p_1, p_2, \ldots, p_5$ を計算できるので，符号化ができる．

次に復号の説明をしていく．受信語を $\boldsymbol{y} = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9)$ と表現することにする．

まずは誤りが生じなかった場合を考える．この場合は表 14 の各行・各列で 1 の個数が偶数個になるので，式に表すと次のようになる．

$$\begin{cases} y_1 + y_2 + y_5 = 0 \\ y_3 + y_4 + y_6 = 0 \\ y_1 + y_3 + y_7 = 0 \\ y_2 + y_4 + y_8 = 0 \\ y_5 + y_6 + y_9 = 0 \end{cases} \tag{6}$$

表 13　3 × 3 の水平垂直パリティ検査符号

m_1	m_2	p_1
m_3	m_4	p_2
p_3	p_4	p_5

表 14　3 × 3 の水平垂直パリティ検査符号 (受信語)

y_1	y_2	y_5
y_3	y_4	y_6
y_7	y_8	y_9

この式は式 (4) で与えられているパリティ検査行列 $\mathbf{H}$ によって次のように書くことができる．

$$\mathbf{H}\boldsymbol{y}^T = \mathbf{0} \tag{7}$$

になる．(なお，この式については $\mathbf{H}\boldsymbol{x}^T = \mathbf{0}$ であることと，受信語に誤りがないので $\boldsymbol{x} = \boldsymbol{y}$ が成立することを利用しても導出できる．) 以上から，$\mathbf{H}\boldsymbol{y}^T = \mathbf{0}$ であれば，誤りがなかったと判定する．

次に誤りが含まれている場合について考える．符号語 $\boldsymbol{x}$ が送られたことを仮定し，受信語が $\boldsymbol{y}$ であったとしよう．例えば，符号語 $\boldsymbol{x} = (0,1,0,1,1,1,0,0,0)$ の 1 bit 目に誤りが生じた場合は受信語は $\boldsymbol{y} = (1,1,0,1,1,1,0,0,0)$ となる．このとき，$\boldsymbol{x}$ と $\boldsymbol{y}$ の関係式は $\boldsymbol{y} = \boldsymbol{x} + (1,0,0,0,0,0,0,0,0)$ で与えられる．ベクトル $(1,0,0,0,0,0,0,0,0)$ の 1 の部分は誤りが生じた場所を表しているため，**誤りベクトル**と呼ばれ，$\boldsymbol{e} = (e_1, e_2, \ldots, e_9)$ と書かれる．誤りベクトル $\boldsymbol{e}$ を用いると，受信語 $\boldsymbol{y}$ は次式の通りに与えられる

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e} \tag{8}$$

誤りベクトルが推定できれば誤りの位置が推定できるため，復号においては受信語 $\boldsymbol{y}$ をもとに誤りベクトル $\boldsymbol{e}$ を推定することが重要である．

誤りが生じなかった場合と同様に $\mathbf{H}\boldsymbol{y}^T$ の計算をしてみよう．$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$ であることと $\mathbf{H}\boldsymbol{x}^T = 0$ であることに注意すると，次式の通りに変形できる．

$$\mathbf{H}\boldsymbol{y}^T = \mathbf{H}(\boldsymbol{x}^T + \boldsymbol{e}^T) = \mathbf{H}\boldsymbol{x}^T + \mathbf{H}\boldsymbol{e}^T = \mathbf{H}\boldsymbol{e}^T \tag{9}$$

この式から，$\boldsymbol{H}\boldsymbol{y}^T$ の計算結果は，受信語 $\boldsymbol{x}$ によらず，誤りベクトル $\boldsymbol{e}$ のみに依存することがわかる．$\boldsymbol{H}\boldsymbol{y}^T$ の計算結果は**シンドローム**と呼ばれており，$\boldsymbol{s} = (s_1, s_2, \ldots, s_5)^T$ で表現される，すなわち

$$\boldsymbol{s} := \mathbf{H}\boldsymbol{y}^T = \mathbf{H}\boldsymbol{e}^T$$

である．式 (4) で与えられている $\mathbf{H}$ をもとに計算をすると

$$\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{pmatrix} = \begin{pmatrix} y_1 + y_2 + y_5 \\ y_3 + y_4 + y_6 \\ y_1 + y_3 + y_7 \\ y_2 + y_4 + y_8 \\ y_5 + y_6 + y_9 \end{pmatrix} \tag{10}$$

である．

誤り訂正符号ではシンドロームを利用することで復号がなされる．まず，1 bit 目が誤っている場合から考えよう．このとき，誤りベクトル $\boldsymbol{e}$ は $(1,0,0,0,0,0,0,0,0)$ で与えられる．シンドローム $\boldsymbol{s}$ は $\mathbf{H}\boldsymbol{e}^T$ から計算ができ，$(1,0,1,0,0)^T$ になる．2 bit 目から 9 bit 目に誤りが生じた場合は，同様の計算をすることによって，表 15 のようにシンドロームを得ることが出来る．表 15 より，誤り位置が異なれば，シンドロームも異なることがわかる．したがって，受信語 $\boldsymbol{y}$ からシンドローム $\boldsymbol{s}$ を計算することによって，誤り位置が特定できることがわかる．受信語の誤り位置を反転させれば，推定語を得ることが出来る．

問題 5　式 (5) を利用してメッセージ (1,0,1,0) を符号化し，検査部 $(p_1, p_2, p_3, p_4, p_5)$ を決定し，符号語を与えなさい．

問題 6　問題 5 で得られた符号語が誤りなく受信できたと仮定する．このときシンドロームを計算し，$(0,0,0,0,0)^T$ となることを確認しなさい．

表 15　水平垂直パリティ検査符号のシンドローム

誤り位置	シンドローム	誤り位置	シンドローム	誤り位置	シンドローム
1 bit 目	$(1,0,1,0,0)^T$	4 bit 目	$(0,1,0,1,0)^T$	7 bit 目	$(0,0,1,0,0)^T$
2 bit 目	$(1,0,0,1,0)^T$	5 bit 目	$(1,0,0,0,1)^T$	8 bit 目	$(0,0,0,1,0)^T$
3 bit 目	$(0,1,1,0,0)^T$	6 bit 目	$(0,1,0,0,1)^T$	9 bit 目	$(0,0,0,0,1)^T$

問題 7 受信語 $\boldsymbol{y}$ が $(1,1,0,1,1,1,0,0,0)$ であった．シンドローム $\mathbf{H}\boldsymbol{y}^T$ を計算しなさい．さらにシンドロームを元に誤り位置を推定しなさい．

1.3.3　プログラムによる実装法

前節で与えた 3×3 の水平垂直パリティ検査符号の符号化と復号をするプログラムは Listing 1 に与えているとおりである．

プログラムの 4-5 行目では符号の情報点数と符号語長を与えている．6-15 行目は表 15 に対応しており，7 行目は 1 bit 目が誤った場合のシンドローム，8 行目は 2 bit 目が誤った場合のシンドロームといった具合に対応している．

27-46 行目では符号化を行う関数 `Encoder` を与えている．この関数では 33-37 行目で式 (5) に対応した計算をしている．ここで用いられている演算子 `^` はビット同士の排他的論理和を計算している．

48-55 行目ではシンドロームの計算を行う関数 `Syndrome` を与えている．50-54 行目は式 (10) の計算を行っている．

84-111 行目では復号を行う関数 `Decoder` を与えている．7-15 行目で与えているシンドロームと誤り位置の対応を利用して，シンドロームから誤り位置を推定し，誤り位置に対応する部分を反転させ推定語を出力する．もしシンドロームがゼロベクトルであれば，誤りがなかったと推定する．また，7-15 行目に対応するシンドロームがなければ誤り検出を出力する．

問題 8 Listing 1 のプログラムを入力し，コンパイルしなさい．

問題 9 プログラムを実行して，メッセージとして (1,0,1,0) を入力し，問題 5 の結果と一致することを確認しなさい．

問題 10 問題 9 の結果に誤りベクトル (0,0,0,1,0,0,0,0,0) を入力した際の受信語・シンドローム・推定語を示しなさい．

2　事前課題

前節までの文章を読み，問題 1 から問題 10 に対する答えを与えなさい．

3　実験課題

1. 3×3 の水平垂直パリティ検査符号の場合，情報点数 K が 4 なので，全部で 16 種類のメッセージが存在する．全てのメッセージに対して符号語を与え，表にしなさい．(Hint: プログラムを適宜改良すると楽にできます．もちろん手計算でも良いです．)
2. 誤りが 0 個または 1 個の時には，推定語が元の符号語と一致することを確認しなさい．
3. プログラムを改良して，誤りが 2 個の場合は必ず誤り検出になることを確認してなさい．また，2 つの異なる誤りベクトルの組の内，シンドロームが一致するものが存在することを確認し，その例を挙げなさい．
4. 以上の問いに対して考察を与えるか，より発展的な問いを考えてそれに対する解答を与えなさい．例えば，以下のような問に対する解答を与えなさい．
   - 1. で導出した符号語はどのような性質があるか?
   - 誤りが 3 個以上の場合はどんなことが起きるのか?

- 3 の水平垂直パリティ検査符号のパリティ検査行列・プログラムはどうなるか? 誤りはいくつまで訂正できるだろうか?
- パリティ検査行列 $\mathbf{H}$ と情報点数・符号語長にはどのような関係があるのか?
- パリティ検査行列 $\mathbf{H}$ を別のものに変えたらどうなるのか? 例えば，次の行列 ((7,4) ハミング符号) の場合はどのような符号語になり，いくつまで誤りを訂正できるか?

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

プログラム例

Listing 1　水平垂直パリティ検査符号の符号化と復号

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

#define K 4 // 情報点数
#define N 9 // 符号語長
static int P[N][N-K] =
  {{1, 0, 1, 0, 0},
   {1, 0, 0, 1, 0},
   {0, 1, 1, 0, 0},
   {0, 1, 0, 1, 0},
   {1, 0, 0, 0, 1},
   {0, 1, 0, 0, 1},
   {0, 0, 1, 0, 0},
   {0, 0, 0, 1, 0},
   {0, 0, 0, 0, 1}};

// 長さ leng の配列 s[] を表示する
void PrintVec(int *s, int leng)
{
  int i;
  for(i=0; i<leng; i++){
    printf(" %d", s[i]);
  }
  puts("");
}

void Encoder(int *d, int *x)
{
  int i;
  int p[N-K]; // 検査部冗長性()

  // 検査部の計算
  p[0] = d[0] ^ d[1];
  p[1] = d[2] ^ d[3];
  p[2] = d[0] ^ d[2];
  p[3] = d[1] ^ d[3];
  p[4] = p[0] ^ p[1];

  // 符号語への変換
  for(i=0; i<K; i++){
    x[i] = d[i];
  }
  for(i=0; i<N-K; i++){
    x[i+K] = p[i];
  }
}

void Syndrome(int *y, int *s)
{
  s[0] = y[0] ^ y[1] ^ y[4];
  s[1] = y[2] ^ y[3] ^ y[5];
  s[2] = y[0] ^ y[2] ^ y[6];
  s[3] = y[1] ^ y[3] ^ y[7];
  s[4] = y[4] ^ y[5] ^ y[8];
}

/* 長さ leng の配列 s[], p[] を比較する一致していれば
     0 を返し，一致していなければ
     0 以外を返す
*/
int Diff(int *s, int *p, int leng)
{
  int i;
  int r = 0;
  for(i=0; i<leng; i++){
    r += s[i] ^ p[i];
  }
  return r;
}

/* 長さ leng の配列が全ゼロか判定
 * 全ゼロならば 0 を返し
 * 全ゼロでなければ 1 を返す
 */
int Zero(int *s, int leng)
{
  int i;
  for(i=0; i<leng; i++){
    if(s[i] != 0){  return 1; }
  }
  return 0;
}

void Decode(int *y, int *s, int *h)
{
```

```
86    int i;
87    int posi = N; // 推定した誤り位置
88
89    for(i=0; i<N; i++){ // 推定語の初期化
90      h[i] = y[i];
91    }
92
93    for(i=0; i<N; i++){
94      if(Diff(s,P[i], N-K) == 0){
95        posi = i;
96        break;
97      }
98    }
99    if( posi == N){
100     if(Zero(s, N-K) == 0){ // 誤りがない場合
101       return ;
102     }
103     else{ // 訂正できないが誤りを検出した場合
104       printf(" 誤り検出\n");
105       return ;
106     }
107   }
108
109   // 誤り位置を反転
110   h[posi] ^= 1;
111 }
112
113 int main(void)
114 {
115   int d[K]; // メッセージ
116   int x[N]; // 符号語
117   int e[N]; // 誤りベクトル
118   int y[N]; // 受信語
119   int s[N-K]; // シンドローム
120   int h[N]; // 推定語
121   int i;
122
123   // 元の情報の入力
124   printf("%d bit 分のメッセージを入力：", K);
125   for(i=0; i<K; i++){
126     scanf("%d", &d[i]);
127   }
128
129   // 符号語の作成
130   Encoder(d, x);
131
132   // 符号語の表示
133   printf(" 符号語:");
134   PrintVec(x, N);
135
136   // 誤りベクトルの入力
137   printf("%d bit 分の誤りベクトルを入力：",N);
138   for(i=0; i<N; i++){
139     scanf("%d", &e[i]);
140   }
141
142   // 受信語の作成
143   for(i=0; i<N; i++){
144     y[i] = x[i] ^ e[i];
145   }
146
147   // 受信語の表示
148   printf(" 受信語: ");
149   PrintVec(y, N);
150
151   // シンドロームの計算
152   Syndrome(y, s);
153
154   printf("Syndrome: ");
155   PrintVec(s, N-K);
156
157   // 復号
158   Decode(y, s, h);
159
160   // 推定語の表示
161   printf(" 推定語: ");
162   PrintVec(h, N);
163
164   return 0;
165 }
```