1.ライブラリ関数 islower(), toupper()を使い、下記の trlowup プログラムを書き換えて、新規に trupper プログラムを作成せよ。

＜ソースコード＞ファイル名 trupper001.c

```
 1  #include <stdio.h>
 2  #include <ctype.h>
 3
 4  /*関数 trupper のプロトタイプ宣言*/
 5  char trupper( char c );
 6
 7  int main(void)
 8  {
 9   /*入力した文字を格納する変数の宣言*/
10   int g;
11
12   /*EOF でループを抜け出す*/
13   while( (g=getchar()) != EOF )
14     putchar( trupper(g) );
15   return(0);
16  }
17
18  /*関数 trupper*/
19  char trupper( char c )
20  {
21   if ( islower(c) )
22     return( toupper(c) );
23   else
24     return( c );
25  }
```

＜出力結果＞

```
[nw0945:~/prog1/kadai4] e095745% ./trupper001
abcdefghijklmnopqrstuvwsyz
ABCDEFGHIJKLMNOPQRSTUVWSYZ


ABCdefgHIJKlmnOPQRstuVWXYz
ABCDEFGHIJKLMNOPQRSTUVWXYZ


Jun NISHISHI (^*^)
JUN NISHISHI (^*^)
```

＜解説＞
ライブラリ関数 islower(), toupper()を使うために、ヘッダファイル ctype.h を読み込んでいる。メイン関数の前に関数 trupper を宣言し、ユーザ関数を作る準備をしている。７～１６行目にメイン関数の処理を行っている。注目する点は１４行目の putchar(trupper(g))で１９～２５行目の関数 trupper に処理が飛び、その結果を出力するようにしている。関数 trupper では関数 islower、関数 toupper、return を用いている。
関数 islower は英小文字に対してだけ真を返す。
関数 toupper は英大文字を対応する英小文字に変換する機能を持っている。つまりサンプルプログラムの(c-'a'+'A')と同じ機能を持っている。
その後 return で出た値を戻している。

＜考察＞
まず関数 islower(), toupper()を調べて、サンプルプログラムと見合わせ同じになる部分を探した。サンプルがとても参考になったのですぐに同じプログラムを題意に沿ったプログラムが組めた。
ユーザ関数の意味が分からなくてプログラムの流れがよくわからず苦労した。サンプルではプロトタイプ宣言の変数とメイン関数内の変数の名前が同じで、違いがわからなくて混乱していることに気づき、変数の名前を変えて分かりやすく工夫したした。

2.trupper プログラムを書き換えて、rot13 暗号化プログラム、rot13 復号化プログラムを作成せよ。rot13 とは、次のようにアルファベットを 13 文字ずらす暗号化の方法である。

＜ソースコード＞rot13_001.c

```
 1  #include <stdio.h>
 2
 3  /*関数 rot13 のプロトタイプ宣言*/
 4  char rot13( char c );
 5
 6  int main(void)
 7  {
 8   /*入力した文字を格納する変数の宣言*/
 9   char g;
10
11   /*EOF でループを抜け出す*/
12   while( (g=getchar()) != EOF )
13    {
14    putchar( rot13(g) );
15    }
```

```
16    return(0);
17  }
18
19  /*関数 rot13*/
20  char rot13( char c )
21  {
22    if ('A' <= c && c <= 'M' || 'a' <= c && c <= 'm')
23      {
24      return  ( c+13 );
25      }
26    if ('N' <= c && c <= 'Z' || 'n' <= c && c <= 'z')
27      {
28      return  ( c-13 );
29      }
30    return c;
31  }
```

＜出力結果＞

```
[nw0945:~/prog1/kadai4] e095745% ./rot13_001
abcdefghijklmnopqrstuvwsyz
nopqrstuvwxyzabcdefghijflm


nopqrstuvwxyzabcdefghijflm
abcdefghijklmnopqrstuvwsyz


ABCDEFGHIJKLMNOPQRSTUVWSYZ
NOPQRSTUVWXYZABCDEFGHIJFLM


NOPQRSTUVWXYZABCDEFGHIJFLM
ABCDEFGHIJKLMNOPQRSTUVWSYZ


Jun NISHISHI (;.;)
Wha AVFUVFUV (;.;)


Wha AVFUVFUV (;.;)
Jun NISHISHI (;.;)
```

＜解説＞
１～１８行目までは、ヘッダファイルctype.hを消去しただけで全問のファイルtoupper001.cと同じである。
１９～３１行目までがアルファベットを13文字ずらす処理のなので説明する。if文を用いて大文字と小文字のA~Mまでが１３足し、N~Zまでが１３引き、値を返している。それ以外はそのまま値を返している。
暗号化した文章をまた変換すると、元の文章に戻る。

＜考察＞
単純に１３を足せばできると思い作ってみたが、n~zとN~Zまでが意味のないことに出力してみるまで気づかなかった。その後、今の形で題意を満たせるようにした。工夫した点は演算子"||"を用いて簡略にまとめたことである。

3.オリジナルの暗号化・復号化プログラムを作成せよ

＜暗号化ソースコード＞**ori_cryp_001.c**

```
 1  #include <stdio.h>
 2
 3  /*プロトタイプ宣言*/
 4  char rot( char c );
 5
 6  int main(void)
 7  {
 8   /*入力した文字を格納する変数の宣言*/
 9   char g;
10
11   /*EOFでループを抜け出す*/
12   while( (g=getchar()) != EOF )
13     {
14       putchar( rot(g) );
15     }
16   return(0);
17  }
18
19  /*関数 rot*/
20  char rot( char c )
21  {
22   /*アルファベット子音*/
23   if (('B'<= c && c <='D')||('F'<= c && c <='H')||
24       ('J'<= c && c <='N')||('P'<= c && c <='T')||
```

```
25      ('V'<= c && c <='Z'))
26    {
27      return  ( c-33 );
28    }
29  if (('b'<= c && c <='d')||('f'<= c && c <='h')||
30      ('j'<= c && c <='n')||('p'<= c && c <='t')||
31      ('v'<= c && c <='z'))
32    {
33      return  ( c-33 );
34    }
35
36  /*アルファベット母音*/
37  if((c == 'a')||(c == 'A'))
38    {
39      return(122);
40    }
41  if((c == 'i')||(c == 'I'))
42    {
43      return(123);
44    }
45  if((c == 'u')||(c == 'U'))
46    {
47      return(124);
48    }
49  if((c == 'e')||(c == 'E'))
50    {
51      return(125);
52    }
53  if((c == 'o')||(c == 'O'))
54    {
55      return(126);
56    }
57
58  /*数字*/
59  if('0'<= c && c <='9')
60    {
61      return(c+43);
62    }
63      return c;
64    }
```

＜出力結果＞

```
[nw0945:~/prog1/kadai4] e095745% ./ori_cryp_001
abcdefghijklmnopqrstuvwsyz
zABC}EFG{IJKLM~OPQRS|UVRXY


ABCDEFGHIJKLMNOPQRSTUVWSYZ
z!"#}%&'{)*+,-~/0123|56289


0123456789
[\]^_`abcd


watasinonamaeha
VzSzR{M~MzLz}Gz


Jun NISHIdesu
)|M -{2'{C}R|
```

＜解説＞

英小文字 a ~ z と英大文字 A ~ Z と数字 0 ~ 9 をアスキーコード表の出力可能文字に変換する暗号化プログラムである。１９~６４行でアルファベットの子音は３３字づらし、母音はそれぞれ文字を対応させた。数字は４３字づらしている。

＜復号化ソースコード＞ ori_cryp_002.c

```
 1  #include <stdio.h>
 2
 3  /*プロトタイプ宣言*/
 4  char rot( char c );
 5
 6  int main(void)
 7  {
 8   /*入力した文字を格納する変数の宣言*/
 9   char g;
10
11   /*EOF でループを抜け出す*/
12   while( (g=getchar()) != EOF )
13    {
14      putchar( rot(g) );
15    }
16   return(0);
17  }
```

```
18
19  /*復号化*/
20  char rot( char c )
21  {
22    if (('!'<= c && c <='#')||('%'<= c && c <=39)||
23        (')'<= c && c <='-')||('/'<= c && c <='3')||
24        ('5'<= c && c <='9'))
25      {
26        return  ( c+33 );
27      }
28    if (('A'<= c && c <='C')||('E'<= c && c <='G')||
29        ('I'<= c && c <='M')||('O'<= c && c <='S')||
30        ('U'<= c && c <='Y'))
31      {
32        return  ( c+33 );
33      }
34    if(c == 'z')
35      {
36        return(97);
37      }
38    if(c == '{')
39      {
40        return(105);
41      }
42    if(c == '|')
43      {
44        return(117);
45      }
46    if(c == '}')
47      {
48        return(101);
49      }
50    if(c == '~')
51      {
52        return(111);
53      }
54
55    if(91<= c && c <=100)
56      {
57        return(c-43);
```

```
    58      }
    59        return c;
    60      }
```

＜出力結果＞

```
[nw0945:~/prog1/kadai4] e095745% ./ori_cryp_002
zABC}EFG{IJKLM~OPQRS|UVRXY
abcdefghijklmnopqrstuvwsyz


z!"#}%&'{)*+,-~/0123|56289
aBCDeFGHiJKLMNoPQRSTuVWSYZ


[\]^_`abcd
0123456789


VzSzR{M~MzLz}Gz
watasinonamaeha


)|M -{2'{C}R|
Jun NiSHidesu
```

＜解説＞

暗号化ソースコードとつくりは同じである。暗号化でづれた文字をもとに戻す作業をしている。

＜考察＞

前問の rot13 暗号化プログラムを参考にして、入力された文字を別の文字に変換するプログラムを作成した。暗号文は一つの法則性だけではすぐに見破られてしまうと思ったことと、先生が授業中に「日本語で暗号を書いた場合、母音が何度も出るので法則がすぐ見破られてしまう」ということをヒントに、アルファベットの子音と母音と数字を別のづらしかたで変換する方法を考えた。しかし、全く思った通りにはできていなかった。
作成した後に出力して、アルファベット小文字と大文字の差が３２であり、３３づらしたこのプログラムは小文字で入力するとすぐに見破られてしまうという欠点も見つけてしまった。

<感想>

暗号を簡単なものと言えど作るというのはすごい技術が必要だと思っていたが、以外とできたので驚いたし、プログラミングの奥深さを感じた。難しい課題だったが、楽しくできた。もっと高度な暗号も作ってみたい。

＜参考文献＞

柴田望洋後援会オフィシャルホームページ
http://www.bohyoh.com/index.html
初心者のためのポイント学習C言語
http://www9.plala.or.jp/sgwr-t/index.html