プログラミングI

REPORT#4

提出日	:	2012年 6月14日　金曜日
所属	:	琉球大学工学部情報工学科
学籍番号	:	125722G
氏名	:	知念 辰明

1.標準ライブラリ関数 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。

a-0:trlowupプログラムの解析。

高水準コードの全体▼

```
1   /*
2     program  :trlowup.c
3     studentID:125722G
4     author   :Tatsunori Chinen
5     date     :2012/06/10/SUN
6     contents :translate! low > up
7    */
8   #include <stdio.h>
9   #include <ctype.h>
10
11  char trlowup(char);                 /* trlowup()という自作関数を呼び出す予告 */
12
13  int main(){
14
15      char c;                         /* char型変数cを宣言 */
16      while( (c = getchar()) != EOF ) /* getchar()関数により変数cに文字を入力し,EOFの値を取らなければ繰り返す */
17          putchar( trlowup(c) );      /* ユーザ指定のtrlowup()関数によって返された値を出力します */
18      return(0);
19
20  }
21
22  char trlowup( char c ){             /* ユーザ指定のtrlowup()関数を開始します */
23
24      if ( 'a' <= c && c <= 'z' )     /* 条件:与えられた値が'a'から'z'のAsciiコードの範囲内である */
25          return( c - 'a'+'A' );      /* cを大文字にした値を返す */
26      else
27          return(c);                  /* そのままの値を返す */
28
29  }
```

実行結果▼　※"abcDEF"および"xyzXyYyZz"が,getchar()関数で入力した値である。
また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
abcDEF
ABCDEF
xyzXyYyZz
XYZXYYYZZ
^C
```

a-1:islower()関数を使ったtrupperプログラム。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  int main(){
12
13      char c;
14      while( (c = getchar()) != EOF ){ /* getchar()関数により変数cに文字を入力し,EOFの値を取らなければ繰り返す */
15          if ( islower(c) )            /* cの値が小文字であるかどうかを条件に分岐します */
16              putchar( c -('a'-'A') ); /* 真であれば大文字のASCIIコードに変換して出力します */
17          else
18              putchar(c);              /* 偽であればそのまま出力します */
19      }
20      return(0);
21
22  }
```

実行結果▼　※"abcDEF"および"xyzXyYyZz"が,getchar()関数で入力した値である。
また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
abcDEF
ABCDEF
xyzXyYyZz
XYZXYYYZZ
^C
```

考察▼

・islower()関数を用いて,入力された文字列に存在する英小文字を英大文字に変換するプログラムを作成した。
・シェルにおいて"man islower"と打ち込むことで以降に記述するようなislower()関数のマニュアルを読むことができる。
・これによれば,islower関数は入力された値が英小文字であったとき真（0以外）の値を返し,英小文字でなければ偽（0）の値を返す。
・これを条件としてif文を構成し,真であればREPORT#3で触れた文字の演算を用いて大文字のASCIIコードへと値を変えている。
・これらのことから,islower(c)関数はtrlowupプログラム内における" 'a' <= c && c <= 'z' "に相当する役割を果たすことが分かる。
・また,islower()関数による文字判定と分岐後の処理は各文字に対して行われ,値が偽となる他の文字列には

値が真であるときの処理は適用されない。

islower()関数のマニュアルの一部▼

```
DESCRIPTION
     The islower() function tests for any lower-case letters.  The value of
     the argument must be representable as an unsigned char or the value of
     EOF.

     In the ASCII character set, this includes the following characters (with
     their numeric values shown in octal):

     141 ``a''     142 ``b''     143 ``c''     144 ``d''     145 ``e''
     146 ``f''     147 ``g''     150 ``h''     151 ``i''     152 ``j''
     153 ``k''     154 ``l''     155 ``m''     156 ``n''     157 ``o''
     160 ``p''     161 ``q''     162 ``r''     163 ``s''     164 ``t''
     165 ``u''     166 ``v''     167 ``w''     170 ``x''     171 ``y''
     172 ``z''

RETURN VALUES
     The islower() function returns zero if the character tests false and
     returns non-zero if the character tests true.
```

a-2:toupper()関数を使ったtrupperプログラム。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  int main(){
12
13      char c;                          /* 変数を宣言します */
14      while( (c = getchar()) != EOF )  /* 変数cに文字列を取り込み,その値が"EOF"を示さなければ繰り返します */
15          putchar( toupper(c) );       /* toupper()関数の返した文字列を出力します */
16      return(0);
17
18  }
```

実行結果▼　※"abcDEF"および"xyzXyYyZz"が,getchar()関数で入力した値である。
また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
abcDEF
ABCDEF
xyzXyYyZz
XYZXYYYZZ
^C
```

考察▼

・toupper()関数を用いて,入力された文字列に存在する英小文字を英大文字に変換するプログラムを作成した。
・シェルにおいて"man toupper"と打ち込むことで以降に記述するようなtoupper()関数のマニュアルを読むことができる。
これによれば,toupper()関数は入力された英小文字を英大文字に変換して,それ以外の文字はそのままで値を返す関数である。
・よって,ユーザ指定のtrlowup()関数をtoupper()関数に置き換えることで同じ動作を実現することができる。

toupper()関数のマニュアルの一部▼

```
DESCRIPTION
     The toupper() function converts a lower-case letter to the corresponding
     upper-case letter.  The argument must be representable as an unsigned
     char or the value of EOF.

     Although the toupper() function uses the current locale, the toupper_l()
     function may be passed a locale directly. See xlocale(3) for more infor-
     mation.

RETURN VALUES
     If the argument is a lower-case letter, the toupper() function returns
     the corresponding upper-case letter if there is one; otherwise, the argu-
     ment is returned unchanged.
```

a-3:islower()関数,toupper()関数の両方を使ったtrupperプログラム。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  int main(){
12
13      char c;                         /* char型変数cを宣言します */
14      while((c = getchar()) != EOF )  /* getchar()関数によって変数cに文字列を取り込みます */
15          if ( islower(c) )           /* islower()関数の返す値を条件に分岐します */
16              putchar( toupper(c) );  /* 値が真であったとき, toupper()関数により英小文字を英大文字に変換します */
17          else
18              putchar(c);             /* 値が偽であったとき,そのまま文字列を出力します */
19      return(0);
20
21  }
```

実行結果▼　※"abcDEF"および"xyzXyYyZz"が,getchar()関数で入力した値である。
　　また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
abcDEF
ABCDEF
xyzXyYyZz
XYZXYYYZZ
^C
```

考察▼
・islower()関数,およびtoupper()関数を用いて入力された文字に存在する英小文字を
英大文字に変換するプログラムを作成した。
・様々な文字列を入力したがa-1項やa-2項のislower()関数,toupper()関数それぞれを単独で用いた場合と
得られる結果が変わらなかった為,このコード構成にする意味は特に無い。

a-4:islower()関数,toupper()関数の両方を使ったtrupperプログラム。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  int main(){
12            char c;
13            while(    (c = getchar()) != EOF )
14          if ( islower(c) != 0 ){
15              printf( "\n line \n" );
16              putchar( c -('a'-'A') );
17          } else
18              putchar(c);
19            return(0);
20
21  }
```

実行結果▼　※"test"が,getchar()関数で入力した値である。
　　また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
test

 line
T
 line
E
 line
S
 line
T
^C
```

考察▼
・getchar()関数の挙動に関して観察を行った。
・Wikiサイト『Open Lecture』内,C言語の入出力関数が示された資料によれば,
getchar()関数は一文字を入力する関数であるという。
・得られた結果より,入力する際"test"と繋げて入力しても,プログラムによって1文字ずつ処理されていることが
確認できた。

a-5:サブルーチンの挙動の観察。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  char trupper(char);                         /* サブルーチンtrupper()を宣言 */
12
13  int main(){
14
15            char c;                           /* char型変数cを宣言 */
16            while( (c = getchar()) != EOF )   /* getchar()関数により変数cに文字を入力し,繰り返す */
17                      putchar( trupper(c) );  /* サブルーチンtrupper()が返した値を文字として出力 */
18            return(0);
19
20  }
21
22  char trupper( char c ){                     /* サブルーチンtrupper()を開始 */
23
24            if ( islower(c) )                 /* islower()関数が返す値によって分岐 */
25                      return( toupper(c) );   /* 値が真であればtoupper()関数によって大文字に変換 */
26            else
27                      return( c );            /* 値が偽であれば値を変換せずに返す */
28
29  }
```

実行結果▼　※"abcDEF"および"xyzXyYyZz"が,getchar()関数で入力した値である。
また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
abcDEF
ABCDEF
xyzXyYyZz
XYZXYYYZZ
^C
```

考察▼

・サンプルプログラムのようなサブルーチン構成を用いて,trupperプログラムを作成した。
・サブルーチン"char trupper(char c)"以降の変数名"c"を変数名として適当な他の変数名に置き換えても
プログラムは同様の結果を出力した。
・また,11行目および22行目"char trupper"における"char"を"float"や"int"に置き換えても同じ結果が得られた。
ただし,変数型として存在しない文字列,"function"などに置き換えるとコンパイルの際に以下のようなエラーが
表示された。

コンパイルの際のエラー▼

```
yp1f.c:11: error: expected ‘=’ , ‘,’ , ‘;’ , ‘asm’ or ‘__attribute__’ before ‘trupper’
yp1f.c:22: error: expected ‘=’ , ‘,’ , ‘;’ , ‘asm’ or ‘__attribute__’ before ‘trupper’
```

エラーの意味▼

・"=",",",";", "asm" もしくは"__attribute__"が"trupper"の前にありません。

2.trupperプログラムを書き換えて,rot13暗号化・復号化プログラムを作成せよ。

a-1:暗号化プログラムの作成。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  char encode(char);                      /* サブルーチンencode()を宣言します */
12
13  int main(){
14
15      char lock;                          /* char型変数lockを宣言します */
16      puts( " ====encoding" );
17      while( (lock = getchar())!= EOF )   /* 変数lockに文字を取り込み,その値がEOFをとらなければ繰り返します */
18          putchar( encode(lock) );        /* ユーザ定義のencode()ルーチンが返した値を出力します */
19      return(0);
20
21  }
22
23  char encode( char rot ){
24      if      ( 'A' <= rot && rot <= 'M' || 'a' <= rot && rot <= 'm' ) /* if文で分岐します */
25          return( rot + 13 );
26      else if ( 'N' <= rot && rot <= 'Z' || 'n' <= rot && rot <= 'z' ) /* else if文で分岐します */
27          return( rot - 13 );
28      else                                                            /* そのままの値を返します */
29          return( rot );
30  }
```

実行結果▼　※"ABCDEFGHIJKLMnopqrstuvwxyzおよび"abcdefghijklmNOPQRSTUVWXYZ"が,getchar()関数で入力した値である。また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
 ====encoding
ABCDEFGHIJKLMnopqrstuvwxyz
NOPQRSTUVWXYZabcdefghijklm
abcdefghijklmNOPQRSTUVWXYZ
nopqrstuvwxyzABCDEFGHIJKLM
^C
```

考察▼

・1-a-4項における,サブルーチン構成を用いたtrupperプログラムを書き換えてrot13暗号化プログラムを作成した。
・入力された文字を,アルファベット昇順に13文字ずらす結果が得られる。
・ASCIIコードとして見た場合,昇順に13文字ずらした先がアルファベットでない場合（"n"から"z","N"から"Z"）"z","Z"の次の文字をそれぞれ"a","A"として扱っている。

a-2:複合化プログラムの作成。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  char decode(char);                      /* サブルーチンdecode(char)を宣言します */
12
13  int main(){
14
15      char unlock;                          /* char型変数unlockを宣言します */
16      puts( " ====decoding" );
17      while( (unlock = getchar())!= EOF )  /* 変数unlockに文字を取り込み,その値がEOFをとらなければ繰り返します */
18          putchar( decode(unlock) );       /* ユーザ定義のdecode()ルーチンが返した値を出力します */
19      return(0);
20
21  }
22
23  char decode( char rot ){
24      if      ( 'A' <= rot && rot <= 'M' || 'a' <= rot && rot <= 'm' ) /* if文で分岐します */
25          return( rot + 13 );
26      else if ( 'N' <= rot && rot <= 'Z' || 'n' <= rot && rot <= 'z' ) /* else if文で分岐します */
27          return( rot - 13 );
28      else                                                            /* そのままの値を返します */
29          return( rot );
30  }
```

実行結果▼　※"ABCDEFGHIJKLMnopqrstuvwxyzおよび"abcdefghijklmNOPQRSTUVWXYZ"が,getchar()関数で入力した値である。また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
 ====encoding
ABCDEFGHIJKLMnopqrstuvwxyz
NOPQRSTUVWXYZabcdefghijklm
abcdefghijklmNOPQRSTUVWXYZ
nopqrstuvwxyzABCDEFGHIJKLM
^C
```

考察▼

・a-1項,暗号化プログラムを書き換えて,復号化プログラムを作成した。
・rot13暗号化の場合,アルファベットを昇順,降順いずれの順にずらそうとも取る値は変わらない。
よって,復号化プログラムはa-1項の暗号化プログラムで代用できてしまう。
・復号化の場合,"a"から"m","A"から"M"にかけて降順に13文字ずらした先がアルファベットでないため,"a","A"の前の値をそれぞれ"z","Z"として扱っている。

3.オリジナルの暗号化・複合プログラムを作成せよ。

a-1:暗号化プログラムの作成。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  char encode(char);
12
13  int main(){
14
15      char lock;                          /* char型変数lockを宣言 */
16      puts( " ====encoding" );
17      while( (lock = getchar())!= EOF )   /* 入力された文字を1つずつlockに代入してループを開始 */
18          putchar( encode(lock) );        /* サブルーチンencode()が返した値を文字として出力 */
19      return(0);
20
21  }
22
23  char encode( char rot ){                /* サブルーチンencode()を開始 */
24
25      if      ( 0x00 <= rot && rot <= 0x1f || (rot % 5) == 0 ) /* 制御文字及び5で割り切れる値について分岐 */
26          return( rot );                  /* 値を変えずに返します */
27      else if ( rot == 32 )               /* 値が32（空白" "）であった場合の例外処理に分岐 */
28          return( 126 );
29      else if ( rot == 126 )              /* 値が126（"~"）であった場合の例外処理に分岐 */
30          return( 34 );
31      else if ( (rot % 5) == 1 )          /* 値を5で割った余りが1であった場合の変則処理に分岐 */
32          return( rot + 3 );
33      else                                /* それ以外の値に対する通常処理に分岐 */
34          return( rot - 1 );
35
36  }
```

実行結果▼　※"ABCDEFGHIJKLMnopqrstuvwxyzおよび"nopqrstyvwxyzABCDEFGHIJKLMが,getchar()関数で入力した値である。また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
 ====encoding
ABCDEFGHIJKLMnopqrstuvwxyz
AEBCDFJGHIKOLnropqswtuvx|y
abcdefghijklmNOPQRSTUVWXYZ
`abdhefgimjklMNPTQRSUYVWXZ
^C
```

考察▼

・オリジナルの暗号化プログラムを作成した。

a-2:暗号化プログラムの作成。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #include <ctype.h>
10
11  char decode(char);
12
13  int main(){
14
15          char unlock;                    /* char型変数unlockを宣言 */
16      puts( " ====decoding" );
17      while( (unlock = getchar())!= EOF )     /* 入力された文字を1つずつlockに代入してループを開始 */
18          putchar( decode(unlock) );          /* サブルーチンdecode()が返した値を文字として出力 */
19      return(0);
20
21  }
22
23  char decode( char rot ){                /* サブルーチンencode()を開始 */
24
25      if      ( 0x00 <= rot && rot <= 0x1f || (rot % 5) ==0) /* 制御文字及び5で割り切れる値について分岐 */
26          return( rot );
27      else if ( rot == 126 )              /* 値が126（"~"）であった場合の例外処理に分岐 */
28          return( 32 );
29      else if ( rot == 32 )               /* 値が32（空白" "）であった場合の例外処理に分岐 */
30          return( 126 );
31      else if ( rot % 5 == 4 )            /* 値を5で割った余りが4であった場合の変則処理に分岐 */
32          return( rot - 3 );
33      else                                /* それ以外の値に対する通常処理に分岐 */
34          return( rot + 1 );
35
36  }
```

実行結果▼　※"ABCDEFGHIJKLMnopqrstuvwxyzおよび"nopqrstyvwxyzABCDEFGHIJKLMが,getchar()関数で入力した値である。また,"^C"はプログラムを終了するための"Control+C"の入力を示している。

```
 ====encoding
ABCDEFGHIJKLMnopqrstuvwxyz
AEBCDFJGHIKOLnropqswtuvx|y
abcdefghijklmNOPQRSTUVWXYZ
`abdhefgimjklMNPTQRSUYVWXZ
^C
```

考察▼

・a-1項で作られた暗号を複合するプログラムを作成した。

a-3:暗号化の対応表を作成。

高水準コードの一部▼

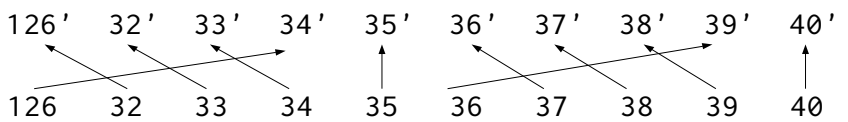
```
8   #include <stdio.h>
9   #define FALSE 0       /* 以降のFALSEを0に置き換える */
10  #define TRUE !FALSE   /* 以降のTRUEを!FALSE (=!0) に置き換える */
11
12  char encode(char);
13
14  int main(){
15
16      int number = 0x20 ,count = 0;
17
18      while(TRUE){
19          printf( "  C / %%d/%%x    aft/ %%d/%%x ||" );
20          count++;
21          if ( count == 4 ) {
22              printf( "\n" ); break;
23          }
24      }
25      while(TRUE){
26          if( number == 0x7f ){
27              printf( "\n" );
28              break;
29          }
30          printf( " \"%c\"=%3d=%x => \"%c\"=%3d=%x ||"
    ,number,number,number,encode(number),encode(number),encode(number));
31          if( ((number - 0x1f) % 4) == 0 ) printf( "\n" );
32          number+=0x01;
33      }
34      puts( " ----Encode List(0x20~0x7e) outputs finished." );
35      return(0);
36
37  }
38
39  char encode( char rot ){
40      if      ( 0x00 <= rot && rot <= 0x1f || (rot % 5) == 0 )
41          return( rot );
42      else if ( rot == 32 )
43          return( 126 );
44      else if ( rot == 126 )
45          return( 34 );
46      else if ( (rot % 5) == 1 )
47          return( rot + 3 );
48      else
49          return( rot - 1 );
50
51  }
```

実行結果▼

```
  C / %d/%x    aft/ %d/%x ||  C / %d/%x    aft/ %d/%x ||  C / %d/%x    aft/ %d/%x ||  C / %d/%x    aft/ %d/%x ||
 " "= 32=20 => "~"=126=7e || "!"= 33=21 => " "= 32=22 || """= 34=22 => "!"= 33=21 || "#"= 35=23 => "#"= 35=23 ||
 "$"= 36=24 => "'"= 39=27 || "%"= 37=25 => "$"= 36=24 || "&"= 38=26 => "%"= 37=25 || "'"= 39=27 => "&"= 38=26 ||
 "("= 40=28 => "("= 40=28 || ")"= 41=29 => ","= 44=2c || "*"= 42=2a => ")"= 41=29 || "+"= 43=2b => "*"= 42=2a ||
 ","= 44=2c => "+"= 43=2b || "-"= 45=2d => "-"= 45=2d || "."= 46=2e => "1"= 49=31 || "/"= 47=2f => "."= 46=2e ||
 "0"= 48=30 => "/"= 47=2f || "1"= 49=31 => "0"= 48=30 || "2"= 50=32 => "2"= 50=32 || "3"= 51=33 => "6"= 54=36 ||
 "4"= 52=34 => "3"= 51=33 || "5"= 53=35 => "4"= 52=34 || "6"= 54=36 => "5"= 53=35 || "7"= 55=37 => "7"= 55=37 ||
 "8"= 56=38 => ";"= 59=3b || "9"= 57=39 => "8"= 56=38 || ":"= 58=3a => "9"= 57=39 || ";"= 59=3b => ":"= 58=3a ||
 "<"= 60=3c => "<"= 60=3c || "="= 61=3d => "@"= 64=40 || ">"= 62=3e => "="= 61=3d || "?"= 63=3f => ">"= 62=3e ||
 "@"= 64=40 => "?"= 63=3f || "A"= 65=41 => "A"= 65=41 || "B"= 66=42 => "E"= 69=45 || "C"= 67=43 => "B"= 66=42 ||
 "D"= 68=44 => "C"= 67=43 || "E"= 69=45 => "D"= 68=44 || "F"= 70=46 => "F"= 70=46 || "G"= 71=47 => "J"= 74=4a ||
 "H"= 72=48 => "G"= 71=47 || "I"= 73=49 => "H"= 72=48 || "J"= 74=4a => "I"= 73=49 || "K"= 75=4b => "K"= 75=4b ||
 "L"= 76=4c => "O"= 79=4f || "M"= 77=4d => "L"= 76=4c || "N"= 78=4e => "M"= 77=4d || "O"= 79=4f => "N"= 78=4e ||
 "P"= 80=50 => "P"= 80=50 || "Q"= 81=51 => "T"= 84=54 || "R"= 82=52 => "Q"= 81=51 || "S"= 83=53 => "R"= 82=52 ||
 "T"= 84=54 => "S"= 83=53 || "U"= 85=55 => "U"= 85=55 || "V"= 86=56 => "Y"= 89=59 || "W"= 87=57 => "V"= 86=56 ||
 "X"= 88=58 => "W"= 87=57 || "Y"= 89=59 => "X"= 88=58 || "Z"= 90=5a => "Z"= 90=5a || "["= 91=5b => "^"= 94=5e ||
 "\"= 92=5c => "["= 91=5b || "]"= 93=5d => "\"= 92=5c || "^"= 94=5e => "]"= 93=5d || "_"= 95=5f => "_"= 95=5f ||
 "`"= 96=60 => "c"= 99=63 || "a"= 97=61 => "`"= 96=60 || "b"= 98=62 => "a"= 97=61 || "c"= 99=63 => "b"= 98=62 ||
 "d"=100=64 => "d"=100=64 || "e"=101=65 => "h"=104=68 || "f"=102=66 => "e"=101=65 || "g"=103=67 => "f"=102=66 ||
 "h"=104=68 => "g"=103=67 || "i"=105=69 => "i"=105=69 || "j"=106=6a => "m"=109=6d || "k"=107=6b => "j"=106=6a ||
 "l"=108=6c => "k"=107=6b || "m"=109=6d => "l"=108=6c || "n"=110=6e => "n"=110=6e || "o"=111=6f => "r"=114=72 ||
 "p"=112=70 => "o"=111=6f || "q"=113=71 => "p"=112=70 || "r"=114=72 => "q"=113=71 || "s"=115=73 => "s"=115=73 ||
 "t"=116=74 => "w"=119=77 || "u"=117=75 => "t"=116=74 || "v"=118=76 => "u"=117=75 || "w"=119=77 => "v"=118=76 ||
 "x"=120=78 => "x"=120=78 || "y"=121=79 => "|"=124=7c || "z"=122=7a => "y"=121=79 || "{"=123=7b => "z"=122=7a ||
 "|"=124=7c => "{"=123=7b || "}"=125=7d => "}"=125=7d || "~"=126=7e => """= 34=20 ||
 ----Encode List(0x20~0x7e) outputs finished.
```

考察▼

・REPORT#3で作成したASCIIコード表に,暗号化後の文字を対応させ出力させた。
・printf()関数内での変換指定子にもサブルーチン（あるいは関数）の返す値が使えることを用いた。



・ASCIIコード基数10における"126"および"32"のときを例外とし,上図のような暗号化を行っている。
・基数10におけるコードが5で割り切れる場合は変換せず,余りが1である場合は3つ先の文字へ,それ以外の場合は1つ戻った文字に変換される。

XXX.あとがき。（反省・感想・参考）

a-1.参考サイト・文献。

・『C実践プログラミング 第三版』（オーム社）
・『Open Lecture』
http://www.osn.u-ryukyu.ac.jp/lecture/wiki/index.php?Open%20Lecture

・シェルにてmanコマンドで得られる各種関数のマニュアル

a-2.反省・感想。

新しい変数が色々でてきて,どういう使い方をすればいいかてんやわんやです。

サブルーチンが出てくると,コード上を順次処理ではなく行ったり来たりするようになり,
目で追えなくなってきてます…。
コメントアウトがあると,その行で何をしているのか,どんな関数あるいはサブルーチンを呼び出しているのかが一目で分かるのでこれからも積極的に書いていきたいです。

これから新出する各関数の細かな動きについても,調べていけたらと思います。

最後までご覧いただきありがとうございました！