プログラミング Ⅰ

Report #4(再)

提出日　：2013年8月1日
所　属　：工学部情報工学科
学籍番号：e135732J
氏　名　：前城　健太郎

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。

1.1 trlowup プログラムの解析

1.1.1 trlowup プログラムのソースコード全体

```
1   #include <stdio.h>
2   #include <ctype.h>
3
4   char trlowup(char);
5
6   int main(){
7     char c;
8
9     while( (c=getchar()) != EOF )
10      putchar(trlowup(c) );
11    return(0);
12  }
13
14  char trlowup( char c ){
15    if ( 'a' <= c && c <= 'z' )
16      return( c-'a'+'A' );
17    else
18      return( c );
19  }
```

1.1.2 出力結果　※ "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
```

1.1.3 プログラムの解析

a) ４行目で新たな関数 trlowup()のプロトタイプを宣言し、14-19 行目においてその関数の定義をしている.
b) 新たな関数の戻り値は char 型、引数も char 型となっている.
c) 19 行目によって、入力された文字の 1 文字を判断し、出力が EOF、つまり "Control+C"が入力されるまでプログラムが繰り返し実行されるようになっている.
d) 14-19 行目における定義から、trlowup()関数による効果は「入力された値が小文字なら大文字、大文字ならそのまま出力する」とわかる.
e) islower()関数、toupper()関数を使用するために、２行目で新たに ctype.h をインクルードしている.

1.2 islower()関数を使った trupper プログラム。

1.2.1 ソースコード全体

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

int main(){
  char c;

  while( (c = getchar()) != EOF ){ /*変数 c に入力された 1 文字を格納、EOF でなければ繰り返す*/
    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*/
      putchar( c -('a'-'A') ); /*真であれば大文字に変換して出力*/
    else
      putchar(c); /*偽であればそのまま出力*/
  }
  return(0);
}
```

1.2.2 出力結果　※ "abcDEF"および "xyzXyYyZz","ldoeUGpigoIA”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
ldoeUGpigoIA
LDOEUGPIGOIA
^C
```

1.2.3 考察

a) islower()関数によって入力された文字列に存在する小文字を大文字へ変換するプログラムを制作した.
b) islower()関数の機能は「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」となっている.
c) これは trlowup プログラムにおける trlowup()関数の " 'a' <= c && c <= 'z' "に当たる機能である.
d) よって trlowup()関数ではなく islower()関数を用いて if 文を構成しプログラムを組んだ.

1.3 toupper()関数を使ったtrupperプログラム。

1.3.1 ソースコード全体

```
1   #include <stdio.h>
2   #include <ctype.h>
3
4   int main(){
5     char c;
6
7     while( (c = getchar()) != EOF ){ /*変数cに入力された1文字を格納、EOFでなければ繰り返す*/
8       putchar(toupper(c));
9     return(0);
10  }
```

1.3.2 出力結果　※ "abcDEF"および "xyzXyYyZz","ldoeUGpigolA”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
laghrlshoJGUG
LAGHRLSHOJGUG
^C
```

1.3.3 考察

a) toupper()関数によって、入力された文字列に存在する小文字を大文字へ変換するプログラムを制作した.

b) toupper()関数の機能は「入力された英小文字を英大文字に変換して返し、それ以外の文字はそのままで値を返す」となっている.

c) これは、trlowupプログラムにおけるtrlowup()関数と同じ機能である.

d) よって、trlowup()関数ではなくtoupper()関数を用いてプログラムを組んだ.

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

2.1 rot13 暗号化プログラム

2.1.1 ソースコード全体

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

char encode(char);

int main(){

  char c;
  printf("--encryption?--\n");
  while((c = getchar())!= EOF)
    putchar(encode(c));
  return(0);
}


char encode(char rot){

  if ('A' <= rot && rot <= 'M' || 'a' <= rot && rot <= 'm')
    return(rot + 13);
  else if('N' <= rot && rot <= 'Z' || 'n' <= rot && rot <= 'z')
    return(rot - 13);
  else
    return( rot );
}
```

2.1.2 出力結果　※ abcdefghijklmnopqrstuvwxyz および ABCDEFGHIJKLMNOPQRSTUVWXYZ、In fact I'm a girl 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。

```
--encryption?--
abcdefghijklmnopqrstuvwxyz
nopqrstuvwxyzabcdefghijklm
ABCDEFGHIJKLMNOPQRSTUVWXYZ
NOPQRSTUVWXYZABCDEFGHIJKLM
In fact I'm a girl
Va snpg V'z n tvey
^C
```

2.1.3 考察

a) 1.1 項を参考にして、trupper プログラムを書き換えて rot13 暗号化プログラムを制作した.

b) 入力されたアルファベットを昇順で 13 文字ずらすような出力を得ることができる.

c) 'a'と 'z'、'A'と 'Z'は続いてるとみなしている.

2.2 rot13 複合プログラム

2.2.1 ソースコード全体

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

char decode(char);

int main(){

  char c;
  printf("--decryption?--\n");
  while((c = getchar())!= EOF)
    putchar(decode(c));
  return(0);
}


char decode(char rot){

  if ('A' <= rot && rot <= 'M' || 'a' <= rot && rot <= 'm')
    return(rot + 13);
  else if('N' <= rot && rot <= 'Z' || 'n' <= rot && rot <= 'z')
    return(rot - 13);
  else
    return( rot );
}
```

2.2.2 出力結果　※ abcdefghijklmnopqrstuvwxyz および ABCDEFGHIJKLMNOPQRSTUVWXYZ、Va snpg V'z n tvey 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。

```
--encryption?--
abcdefghijklmnopqrstuvwxyz
nopqrstuvwxyzabcdefghijklm
ABCDEFGHIJKLMNOPQRSTUVWXYZ
NOPQRSTUVWXYZABCDEFGHIJKLM
Va snpg V'z n tvey
In fact I'm a girl
^C
```

2.2.3 考察

a) 1.1 項、rot13 暗号化プログラムを参考にして、rot13 複合プログラムを制作した.
b) 入力されたアルファベットを降順で 13 文字ずらすような出力を得ることができる.
c) 'a'と 'z'、'A'と 'Z'は続いてるとみなしている.
d) rot13 の場合では昇順、降順どちらでずらそうとも取る値は同じであるので、rot13 暗号化プログラム(encode()関数)とまったく同じプログラムを使って複合することができる.

3. オリジナルの暗号化・復号プログラムを作成せよ。

3.1 rot47 暗号化、複合プログラム

3.1.1 ソースコード全体

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

char encode(char);

int main(){

  char c;
  printf("--encryption?rot47--\n");
  while((c = getchar())!= EOF)
    putchar(encode(c));
  return(0);
}


char encode(char rot){

  if ('!' <= rot && rot <= 'P')
    return(rot + 47);
  else if('Q' <= rot && rot <= '~')
    return(rot - 47);
  else
    return( rot );
}
```

3.1.2 出力結果(1)　※ the Answer to the Ultimate Question of Life, the Universe, and Everything = 42 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。

```
--encryption?rot47--
the Answer to the Ultimate Question of Life, the Universe, and Everything = 42
E96 p?DH6C E@ E96 &=E:>2E6 "F6DE:@? @7 {:76[ E96 &?:G6CD6[ 2?5 tG6CJE9:?8 l ca
^C
```

3.1.3 出力結果(2)　※ E96 p?DH6C E@ E96 &=E:>2E6 "F6DE:@? @7 {:76[E96 &?:G6CD6[2?5 tG6CJE9:?8 l ca 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。

```
--encryption?rot47--
E96 p?DH6C E@ E96 &=E:>2E6 "F6DE:@? @7 {:76[ E96 &?:G6CD6[ 2?5 tG6CJE9:?8 l ca
the Answer to the Ultimate Question of Life, the Universe, and Everything = 42
^C
```

3.1.4 考察

a) rot13 プログラム(2.1 項、2.2 項)を参考にして、rot47 プログラムを作成した.
b) アルファベットだけではなく、ASCII コードで表すことのできる図形文字（%x:21-[!]）から（%x:7e-[~]）の 94 個まで範囲を拡張した.
c) 丁度半分の 47 ずらすことで暗号化、複合を同じプログラムで実行することができる.

3.2 オリジナルの暗号化、復号プログラム

3.2.1 オリジナル暗号化プログラム

3.2.1.1 ソースコード全体

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

char encode(char);

int main(){

  char c;
  printf("--encryption?original--\n");
  while((c = getchar())!= EOF)
    putchar(encode(c));
  return(0);
}


char encode(char rot){

  if (0 <= rot && rot <= 31)
    return(rot);
  else if (32 <= rot && rot <= 50)
    return(rot + 76);
  else
    return(rot - 19);
}
```

3.2.1.2 出力結果　※ the Answer to the Ultimate Question of Life, the Universe, and Everything = 42 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。

```
--encryption?original--
the Answer to the Ultimate Question of Life, the Universe, and Everything = 42
aURl.[`dR_la\laURlBYaVZNaRl>bR`aV\[l\Sl9VSRxlaURlB[VcR_`RxlN[Ql2cR_faUV[Tl*l!~
^C
```

3.2.1.3 考察

a) オリジナルの暗号化プログラムを作成した.

b) rot47 暗号化とは違い、図形文字に空白スペースを含めた 95 個を 19 個の区切りでずらすことで、英単語の区切りを悟られないように工夫した.

3.2.2 オリジナル暗号化プログラムの復号プログラム

3.2.2.1 ソースコード全体

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

char decode(char);

int main(){

  char c;
  printf("--decryption?original--\n");
  while((c = getchar())!= EOF)
    putchar(decode(c));
  return(0);
}


char decode(char rot){

  if (0 <= rot && rot <= 31)
    return(rot);
  else if (108 <= rot && rot <= 126)
    return(rot - 76);
  else
    return(rot + 19);
}
```

3.2.2.2 出力結果　※ aURl.[`dR_la\laURlBYaVZNaRl>bR`aV\[l\Sl9VSRxlaURlB[VcR_`RxlN[Ql2cR_faUV[Tl*l!~ 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。

```
--decryption?original--
aURl.[`dR_la\laURlBYaVZNaRl>bR`aV\[l\Sl9VSRxlaURlB[VcR_`RxlN[Ql2cR_faUV[Tl*l!~
the Answer to the Ultimate Question of Life, the Universe, and Everything = 42
^C
```

3.2.2.3 考察

a)　3.2.1 項のオリジナル暗号化プログラムを複合するプログラムを作成した.

3.3 rot47 プログラムとオリジナルプログラムの統合

3.3.1 暗号化プログラム

3.3.1.1 ソースコード全体

---プログラムファイル本体(report@4-3-3.c)

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

char rot47(char);
char encode(char);

int main(){

  char c;
  printf("--encryption?rot47+original--\n");
  while((c = getchar())!= EOF)
    putchar(encode(rot47(c)));
  return(0);
}
```

---rot47 プログラムファイル(rot47.c)

```
char rot47(char rot){

if (33 <= rot && rot <= 80)
  return(rot + 47);
 else if(81 <= rot && rot <= 126)
   return(rot - 47);
 else
   return( rot );
}
```

---オリジナル暗号化プログラムファイル(ori-encry.c)

```
char encode(char c){

  if (0 <= c && c <= 31)
    return(c);
  else if (32 <= c && c <= 50)
    return(c + 76);
  else
    return(c - 19);
}
```

---Makefile ファイル(Makefile-encry)

```
#
#Report@4
#ⅲ 項のコンパイル、実行のための Makefile
#暗号化プログラム
#

ccencry:report@4-3-3.o rot47.o ori-encry.o
        gcc -o encry report@4-3-3.o rot47.o ori-encry.o

encry:report@4-3-3.c
        gcc -c report@4-3-3.c

rot47.o:rot47.c
        gcc -c rot47.c

oriencry:ori-encry.c
        gcc -c ori-encry.c
```

3.3.1.2 出力結果　※ the Answer to the Ultimate Question of Life, the Universe, and Everything = 42 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。

```
--encryption?rot47+original--
the Answer to the Ultimate Question of Life, the Universe, and Everything = 42
2&#l],15#0l2-l2&#lr*2'+~2#ln3#12'-,l-$lh'$#Hl2&#lr,'4#01#Hl~,"la4#072&',%lYlPN
^C
```

3.3.1.3 考察

・今までに制作した rot47 暗号化プログラム、オリジナル暗号化プログラムを同時に行うプログラムを作成した.

・暗号化プログラムは変更が容易になるようにプログラムファイルを分割し、Makefile によってコンパイル・実行できるようにした.

3.3.2 復号プログラム

3.3.2.1 ソースコード全体

---プログラムファイル本体(report@4-3-3b.c)

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

char rot47(char);
char decode(char);

int main(){

  char c;
  printf("--decryption?rot47+original--\n");
  while((c = getchar())!= EOF)
    putchar(rot47(decode(c)));
  return(0);
}
```

---rot47 プログラムファイル(rot47.c)

```
char rot47(char rot){

if (33 <= rot && rot <= 80)
  return(rot + 47);
 else if(81 <= rot && rot <= 126)
   return(rot - 47);
 else
   return( rot );
}
```

---オリジナル復号プログラムファイル(ori-decry.c)

```
char decode(char c){

  if (0 <= c && c <= 31)
    return(c);
  else if (108 <= c && c <= 126)
    return(c - 76);
  else
    return(c + 19);
}
```

---Makefile ファイル(Makefile-decry)

```
#
#Report@4
#ⅲ 項のコンパイル、実行のための Makefile
#復号プログラム
#

ccencry:report@4-3-3b.o rot47.o ori-decry.o
        gcc -o decry report@4-3-3b.o rot47.o ori-decry.o

decry:report@4-3-3b.c
        gcc -c report@4-3-3b.c

rot47.o:rot47.c
        gcc -c rot47.c

oridecry:ori-decry.c
        gcc -c ori-decry.c
```

3.3.2.2 出力結果　※ 2&#l],15#0l2-l2&#lr*2'+~2#ln3#12'-,l-lh'#Hl2&#lr,'4#01#Hl~,"la4#072&',%lYlPN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.

```
--decryption?rot47+original--
2&#l],15#0l2-l2&#lr*2'+~2#ln3#12'-,l-$lh'$#Hl2&#lr,'4#01#Hl~,"la4#072&',%lYlPN
the Answer to the Ultimate Question of Life, the Universe, and Everything = 42
^C
```

3.3.2.3 考察

a) 3.3.1 項の暗号化プログラムの復号プログラムを作成した.

b) 同じように Makefile を用いてコンパイル・実行できるようにした.

3.3.3 暗号化、復号プログラムのモード切り替え

3.3.3.1 ソースコード全体　※ rot47.c、ori-encry.c、ori-decry.c については iii-1 項、 iii-2 項における同名ファイルと同一.

---プログラムファイル本体(report@4-3-3b.c)

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

char rot47(char);
char encode(char);
char decode(char);

int main(){

  char c;
  int i;
  printf("encryption = 0,decryption = 1\n");
  printf("===>");
    scanf("%d",&i);
    if(i==0){
      printf("\n--encryption?rot47+original--");
      while((c = getchar())!= EOF)
        putchar(encode(rot47(c)));
    }
    else if(i==1){
      printf("\n--decryption?rot47+original--");
      while((c = getchar())!= EOF)
        putchar(rot47(decode(c)));
    }
    else
      printf("ERROR!\n");
    return(0);
}
```

---Makefile ファイル(Makefile-de,en)

```
#
#Report@4
#iii 項のコンパイル、実行のための Makefile
#暗号化、複合プログラムのモード切り替え
#

ccencry:report@4-3-3c.o rot47.o ori-encry.o ori-decry.o
        gcc -o de-encry report@4-3-3c.o rot47.o ori-encry.o ori-decry.o

encry:report@4-3-3c.c
        gcc -c report@4-3-3c.c

rot47.o:rot47.c
        gcc -c rot47.c

oriencry:ori-encry.c
        gcc -c ori-encry.c

oridecry:ori-decry.c
        gcc -c ori-decry.c
```

3.3.3.2 出力結果(1)　※変数 i には 0 を格納、abcdefgEFGHIZk 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。

```
encryption = 0,decryption = 1
===>0

--encryption?rot47+original--
abcdefgEFGHIZk
~ !"#$%abcdew)
^C
```

3.3.3.3 出力結果(2)　※変数 i には 1 を格納、~ !"#$%abcdew)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。

```
encryption = 0,decryption = 1
===>1

--decryption?rot47+original--
~ !"#$%abcdew)
abcdefgEFGHIZk
^C
```

3.3.3.4 考察

a) rot47 プログラム、オリジナルプログラムの統合し、さらに暗号化、複合プログラムを統合し if 文により暗号化モードと複合モードを切り替えできるようにした.

--あとがき--

a)参考文献

・『C 実践プログラミング 第三版』