[1] 課題内容

コマンドラインから受け取った文字列の大文字と小文字を変換するプログラムを作成せよ。入力は1バイトの表示文字とし、アルファベット文字以外は使用しない。
また、文字列を反転して表示するプログラムも作成せよ。（例 "abcd" => "dcba"）

[2] 大小英文字変換プログラム

a. プログラム

```
------------------------------------------------------------------------------program1----
/***********************************************
  Program    : replace.c
  Student-ID : 045713C
  Author     : OSHIRO,Kazuya
  Date       : 04/07/06
  Comment    : コマンドラインパラメータ(大小文字変換)
***********************************************/

#include <stdio.h>
#include <ctype.h>              /* troupper,islower,isupper,tolowerを使うため宣言 */

void replace (char *,char *);
int get_n (char *);

int main(int argc, char *argv[])
{
  int i;                 /* 変数iを宣言、iはパラメータの番号を表す */
  char dest[30][30];             /* char型の二次元配列を宣言 */
  printf("argc = %d¥n",argc);    /* コマンドにいくつのパラメータがあるかを表す */
  for (i=1,argv++; *argv != NULL; i++,argv++){
  /* iを1に初期化、argvのアドレスを一つずらし、argvの値がNULLでないならiに1をたし、
     argvのアドレスを1つずらす */

    printf ("¥nparameter(%2d)¥n",i);
    printf ("  n = %d ",get_n(*argv)); /* 関数get_nを呼び出し、その値を出力する */
    printf ("¥t%s -> ",*argv);          /* replace処理前の*argvを出力 */
    replace(*dest, *argv);              /* 関数replaceの呼び出し */
    printf (" %s¥n",dest);            /* replace処理後の*argvを出力 */
  }
}

int get_n(char *pa)             /* 整数型の関数get_nの定義 */
{
  int i;

  for (i=0; *pa != NULL; i++, pa++);
  /* iを0に初期化、*paがNULLになるまでポインタpaとiに1を足す */
  return i;
}

void replace (char *dest, char *str) /* void型の関数replaceの定義 */
{
```

```
  while(*str != NULL){                  /* *strがNULLになるまで繰り返す */

     if      (islower(*str) != 0)       /* 英小文字なら真(0以外の数)を返す。 */
         *dest = toupper(*str);         /* 小文字を大文字に変えて代入 */
     else if (isupper(*str) != 0)       /* 英大文字なら真を返す */
         *dest = tolower(*str);         /* 大文字を小文字に変えて代入 */
     else
         *dest = *str;           /* 上記に当てはまらないときそのまま代入 */

     dest++;                    /* ポインタdestのアドレスを一つずらす */
     str++;                     /* ポインタstrのアドレスを一つずらす */
   }
  *dest = *str;
  /* destの値にstrの値(このときはNULLということ)を代入する */
}
```

---

b. 実行結果

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++result1++++
  [Kazuya-OSHIRO:~/prog1/report/#6] j04013% ./replace oshiro kAZUYA 234
  argc = 4

  parameter( 1)
    n = 6         oshiro -> OSHIRO

  parameter( 2)
    n = 6         kAZUYA -> Kazuya

  parameter( 3)
    n = 3         234 -> 234
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

c. 考察

＊＊＊関数の説明＊＊＊

今回は前レポートで使用した関数ですので簡単に説明します。

islower()・・・文字が英小文字なら真、それ以外なら偽を返す。
isupper()・・・文字が英大文字なら真、それ以外なら偽を返す。
toupper()・・・文字が小文字なら大文字に変換した値を返す。
tolower()・・・文字が大文字なら小文字に変換した値を返す。

＊＊＊replace関数の説明＊＊＊

これは与えられたアドレスの値が英文字が大文字なら小文字に小文字なら大文字に変えるための関数である。ifの判断で使われているislower,isupperは入れなくても動くがわかりやすくするため書いた。

この関数のwhile文をぬけた後にある*destへの代入は文字列の終わりを表すためNULLを入れる処理である。この代入がないと繰り返し関数が呼ばれて出力したとき次のような結果がでたりします。

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
[Kazuya-OSHIRO:~/prog1/report/#6] j04013% ./sample2 abcde FGH
argc = 3
```

```
parameter( 1)
  n = 5         abcde ->  ABCDE

parameter( 2)
  n = 3         FGH ->  fghDE
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```
NULLがでるまでが文字列なので前回、保持されていた文字が出力されるとおもわれます。

＊＊＊プログラムの説明＊＊＊

このプログラムはコマンドラインから受け取った引数の個数をargcで出力してargvに格納された文字列を英大文字なら英小文字に英小文字なら英大文字に変換して出力するプログラムである。結果を見るとparameterは3つしかないのにargcが4とでているのはプログラムを起動するためのコマンド./replaceも引数としてとられているためである。

[3] 文字列反転プログラム

a. プログラム

```
-------------------------------------------------------------------------program2----
/***********************************************
  Program    : reverse.c
  Student-ID : 045713C
  Author     : OSHIRO,Kazuya
  Date       : 04/07/06
  Comment    : コマンドラインパラメータ(文字列反転)
***********************************************/

#include <stdio.h>

void reverse(char *);            /* 戻り値なしchar型でポインタの引数を持つ */

int main(int argc,char *argv[])
{
  printf(" 変換前¥t¥t¥t変換後¥n");

  for (argv++; *argv != NULL;argv++){
       printf(" [before] --> %s¥t[after] --> ",*argv);
       reverse(*argv);
       printf("¥n");
  }
}

void reverse(char *s)            /* 関数reverseの定義 */
{
  if (s[0] == NULL)              /* s[0]がNULLなら何もしない */
      return;
  else {
    reverse(&s[1]);              /* 再帰関数、引数にs[0]の次のアドレスを送る */
    putchar(s[0]);               /* s[0]を出力する */
   }
}

-------------------------------------------------------------------------------
```

b. 実行結果

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++result2++++
[Kazuya-OSHIRO:~/prog1/report/#6] j04013% ./reverse oshiro KAZUYA 1256 zyx
 変換前                  変換後
 [before] --> oshiro     [after] --> orihso
 [before] --> KAZUYA     [after] --> AYUZAK
 [before] --> 1256       [after] --> 6521
 [before] --> zyx        [after] --> xyz
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

c. 考察

＊＊＊関数reverse()について＊＊＊

この関数は再帰関数であり、それを使うことにより文字を反転して表示できるようにした。この関数は始め文字列のアドレスを引数としてとる。そこで先頭アドレスがNULLでなければ再帰がおこる、そのとき送られる引数が先頭アドレスの次のアドレス(&s[1])であり、それは再帰で呼び出されたreverse関数では先頭アドレスと考えられることになる。これをNULL文字まで繰り返すことによりputcharによる文字の出力が後回しになる。そして先頭アドレスの値がNULLになったとき再帰は終了し、後のほうの文字から出力が行われるため文字列が反転されることになる。

＊＊＊プログラムについて＊＊＊

このプログラムはコマンドラインから得られた文字列を反転して表示するプログラムである。実はこのプログラムは暗号化のときちょっと考えて断念した代物だったりします。あと2個ぐらい文字を反転させるプログラムを考えたが今回は一番時間がかかったこれを採用しました。

[4] 全体での考察

a. コマンドライン引数について

コマンドラインとはmain関数がコマンドラインからとる引数のことである。
1つ目の引数**argc**はargment countの略であり、そこにはコマンドラインでスペースで区切ったと見た場合の文字列の数が格納される。2つ目の引数**argv**はargument vectorの略である。これはポインタ配列であり、コマンドラインで指定された各文字列のポインタが格納されている。ちなみに、argc,argv共に変数であるためこれでなければ動かないないというわけではないが、一般的にはこれらの変数おくものであるようだ。
今回、結果には表示しないようにしているがreplaceプログラムのargv[0]には./replace、reverseプログラムのargv[0]には./reverseという文字列がそれぞれ格納されている。

b. プログラムにおける不具合

この2つのプログラム、英文字、数字はちゃんと表示できるのですがいくつかの特定の記号文字をコマンドラインに入力すると途中でプログラムが終わったり、プログラムが動かないなどの不具合が起こりました。

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++sample result++++
```

```
[Kazuya-OSHIRO:~/prog1/report/#6] j04013% ./replace oshiro kaz;uya
argc = 3

parameter( 1)
  n = 6         oshiro -> OSHIRO

parameter( 2)
  n = 3         kaz -> KAZ
tcsh: uya: Command not found.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++sample result2++++
[Kazuya-OSHIRO:~/prog1/report/#6] j04013% ./replace oshiro Kazuya k:i$gou
tcsh: gou: Undefined variable.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

これを見るとtcshと書かれている。これよりプログラムでの問題ではないことがわかった。これはコマンドが引数として渡される前にシェルのコマンドとして特定の記号文字は読まれ展開されるためこのような症状が起こるのだろうと思われる。
ちなみに$や;の他に',",*,@,(,),<,>,?,|,&,[,],&&,||,<<,>>などの文字ではそのまま打った場合表示することはできなかった。これらの文字はワイルドカード文字と言う。

[5] 感想・反省

今回のレポートで自分がポインタについてあまり理解してないことがよくわかりました。もっと勉強せねば・・・。それと不具合に関してははっきりとはしなかったのが残念。あと、reverseプログラムをもうちょっといじりたかったです。だけど、いままでのレポートに比べたら比較的順調に進んでいけたと思う。

前回の反省で見やすさを追求すると宣言したのですがどうでしょうか？自分ではいつもと変わらないような気が・・・。少しはいつもとは違う工夫をしたのですがね。まぁ、ともあれ今回はいつも通りぎりぎりとは言え結構楽しみながらできたのが嬉しかったです。以上！

参考文献

C実践プログラミング（Steve Oualline 著）
C言語によるプログラミング（スーパーリファレンス編）（北川 雅己）
初心者のためのポイント学習C言語
（http://www9.plala.or.jp/sgwr-t/）
信州大学 C言語 (応用編)
（http://cai.cs.shinshu-u.ac.jp/sugsi/Lecture/c2/e_top.html）