課題文；

コマンドラインパラメータ(大小文字変換)

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変換しない。　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"）

プログラムには以下の関数を定義して作成すること。

```
replace( char *dest, char *str)
```

サンプルプログラム（3つ）

ナンバー１

```
 1 /*
 2   Program    : argcv.c
 3   Student-ID :075739A
 4   Author     :TSUHAKO Masaki
 5   UpDate     :2007/07/08(Sun)
 6   Comment    :Command-Line-Parameter1
 7 */
 8
 9   #include <stdio.h>
10
11   int  get_n(char *);
12   void print_data(char *,int);
13
14   int main(int argc, char **argv){
15   int   i,n;
16
17   printf("argc = %d\n\n",argc);
18   for (i=1,argv++; *argv != NULL; i++,argv++) {
19   printf("parameter(%2d)\t%s\n",i,*argv);
20   n = get_n(*argv);
21   print_data(*argv,n);
```

```
 22   }
 23   return(0);
 24   }
 25
 26   int get_n(char *pa){
 27   int i;

 28   for (i=0; *pa != NULL; i++,pa++);
 29   return(i);
 30   }
 31
 32   void print_data(char *pa,int n){
 33   printf("n=%2d  %s\n",n,pa);
 34   }
```

ナンバー１の出力結果

```
[Masaki:~/PRGRMNG] e075739% ./a.out a bb ccc dddd eeeee
argc = 6


parameter( 1)   a
n= 1  a
parameter( 2)   bb
n= 2  bb
parameter( 3)   ccc
n= 3  ccc
parameter( 4)   dddd
n= 4  dddd
parameter( 5)   eeeee
n= 5  eeeee
```

ナンバー２

```
  1  /*
  2    Program    : argcv.c
```

```
 3    Comment    :Command-Line-Parameter2
 4  */
 5
 6  #include <stdio.h>
 7
 8  int  get_n(char *);
 9  void print_data(char *,int);
10
11
12   int main(int argc, char **argv){
13     int   i, n;
14
15     printf(" argc = %d\n",argc);
16     printf("&argc = %08x\n",&argc);
17
18
19     for (i=0; *argv != NULL; i++,argv++) {
20       printf("parameter(%2d)\t%s\n",i,*argv);
21       n = get_n(*argv);
22       print_data(*argv,n);
23       printf("**argv = %c, ",**argv);
24       printf(" *argv = %08x, ",*argv);
25       printf("  argv = %08x, ",argv);
26       printf(" (sub) = %08x\n",(long)argv-(long)*argv);
27     }
28     return(0);
29   }
30
31   int get_n(char *pa){
32     int i;
33
34     for (i=0; *pa != '\0'; i++,pa++);
```

```
 35     return(i);
 36  }
 37
 38   void print_data(char *pa,int n){
 39     printf("n=%2d  %s\n",n,pa);
 40   }
```

ナンバー２の出力結果

```
argc = 1
&argc = bffffab0
parameter( 0)   ./a.out
n= 7  ./a.out
**argv = .,  *argv = bffffb9c,   argv = bffffb14,  (sub) = ffffff78
```

ナンバー３

```
  1  /*
  2    Command-Line-Parameter3
  3  */
  4  #include <stdio.h>
  5
  6  int main(){
  7    int i;
  8    char  m[24]     ={'a','b','c','d','e','f','g','h','i','j','k','l',
  9                       'm','n','o','p','q','r','s','t','u','v','w','x' };
 10    char mm[6][4]   ={'a','b','c','d','e','f','g','h','i','j','k','l',
 11                       'm','n','o','p','q','r','s','t','u','v','w','x' };
 12
 13   puts("2 次元配列 mm[]の先頭アドレス");
 14    printf("&mm[0][0] => %08x\n",&mm[0][0]);
 15    printf("    mm[0] => %08x\n",mm[0]);
 16    printf("   &mm[0] => %08x\n",&mm[0]);
 17    printf("   *mm    => %08x\n",*mm);
 18    printf("    mm    => %08x\n",mm);
```

```
 19    puts("2 次元配列 mm[]の先頭の値");
 20    printf(" mm[0][0] => %c\n",mm[0][0]);
 21    printf("   *mm[0] => %c\n",*mm[0]);
 22    printf("  **mm    => %c\n",**mm);
 23
 24    puts("2 次元配列 mm[6][4]の 5 番目の値");
 25    printf(" mm[1][0] => %c\n",mm[1][0]);
 26    printf(" *(*mm+4) => %c\n",*(*mm+4));
 27    printf(" *(*mm+4) => %c\n",**(mm+1));
 28
 30
 30    return(0);
 31  }
```

ナンバー 3 の出力結果

```
2 次元配列 mm[]の先頭アドレス
&mm[0][0] => bffffa6c
    mm[0] => bffffa6c
   &mm[0] => bffffa6c
   *mm    => bffffa6c
    mm    => bffffa6c
2 次元配列 mm[]の先頭の値
 mm[0][0] => a
   *mm[0] => a
  **mm    => a
2 次元配列 mm[6][4]の 5 番目の値
 mm[1][0] => e
 *(*mm+4) => e
 *(*mm+4) => e
```

前回やった『大文字は小文字へ、小文字は大文字に変換するプログラム』

```
 1  /*ヘッダー*/
```

```
2    #include <stdio.h>
3    #include <ctype.h>
4
5  /*文字型の宣言*/
6  char trupper(char);
7
8  /*変数宣言*/
9  int main(){
10    char c;
11
12    printf("大文字は小文字,小文字は大文字にする\n");
13    /*while 文  EOF(End Of File)まできたら終了*/
14    while( (c=getchar()) != EOF )
15      putchar( trupper(c) );
16    return(0);
17  }
18
19  /*c が小文字ならば大文字に変換*/
20  char trupper( char c ){
21
22
23    if (islower(c))
24      return(toupper(c)) ;
25    /*c が大文字ならばそのまま変換*/
26    else if (isupper(c))
27      return(tolower(c));
28    return(c);
29  }
```

出力結果：

```
大文字は小文字,小文字は大文字にする
abcdefg ABCDEFG
ABCDEFG abcdefg
```

まずは、課題文を分かりやすくする。といっても今回の課題文は簡単に理解できる。しかし、今までの課題とは、比べ物にならない程難しそうだ。

サンプルプログラムで疑問に思ったことを調べる。

【ポインタ】

*（アスタリスク）がついているものがあるが、これはいったいなんなのか。

これは、ポインタ演算子といわれるもの。プログラミングの中でも難しい〝ポインタ〟の概念である。いままでは、関数を宣言すれば、その関数の値を決めることができたが、この、ポインタを使用すれば、値と、その値が保存されている場所まで決定することができるとういなんともお得な機能である。*（アスタリスク）を複数書くことで、2 重ポインタ、3 重ポインタと、多段のポインタにすることができる。

ポインタ変数を使うためには、〝ポインタ変数〟と〝実際に値を入れて置く場所〟が必要。

【関数のプロトタイプ宣言】

コンパイラに、関数名、return 型、引数の型をしらせる。

【文字列と配列】

文字列というのは、文字型の配列である。各文字を配列の各要素として順番に格納している。

最後の文字の要素は文字列の終了を表す'\0'（NULL）が入る。

例

Nice! Record

'N'	'i'	'c'	'e'	'!'	','	'R'	'e'	'c'	'o'	'r'	'd'	'¥n'	'¥0'

という配列である。

2 次元配列の場合

char str [3][7]={“ABC”,”DEFGHI”,”JK”};

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
1000 番地	A	B	C	\0			
1007 番地	D	E	F	G	H	I	\0
1014 番地	J	K	\0				

ポインタの主な用途

１,呼び出された関数の中から呼び出し元の変数の値を変更するとき

２,配列や文字列の高速な読み書き

３,変数データどうしの関連性の表現

知っていなければならない知識：

*p++は *(p++)（ポインタ参照のあとでポインタを 1 増やす)という意味であり (*p)++(ポインタの指す先を 1 増やす)という意味ではない点に注意

x = *p; と書く代わりに x = p[0]; と書くことができる。

a[1] = x; と書く代わりに *(a + 1) = x; と書くことが できる。

さて、以上のことと、サンプルプログラムを含め、文字変換プログラムを作る。

考察：

今回の課題での特徴は、ポインタについてである。ただの文字変換ならば、前回の課題のプログラムを使用すればいい。なので、ポインタを使った方法を考える。

方法：

『入力されたコマンドラインパラメータ argc が argv という箱に入る。アスタリスクを使い（=*argv）、argv の中に入っている値を参照する。参照した値を別の箱（char str）にいれ、*str を使って参照した値を変換。変換した値*str をまた別の箱の char dest に入れる。そして dest の値（変換された*str＝*dest）を line という箱に入れ、出力』。

完成した変換プログラム；

```
1 /*str を変換し、dest に変換結果代入するプログラム*/
2 /*必要なヘッダー*/
3   #include <stdio.h>
4   #include <ctype.h>
5
6 /*関数のプロトタイプ */
7 void replace(char *dest, char *str);
8
```

```
9 /*main 関数*/
10 /*仮変数*/
11 /*int argc はコマンド行のパラメータ(このプログラムでは実行コマンドは含まない)*/
12 /*int argv は配列*/
13 int main(int argc, char *argv[]){
14
15   /*配列の宣言*/
16   char line[100];
17
18  /*文字列の認識*/
19 printf("argc（文字列の数）=%d  (./a.out も含める)\n",argc);
20
21  /*for を使った作業*/              /*読み込むのは，./a.out の次から  argv++       */
22  for(argv++;*argv !=NULL;argv++) /*文字列の最後まできたら終了       *argv !=NULL*/
23                                  /*それまで argv は 1 ずつ増やしていく  argv++    */
24 {
25   /*  関数 replace */
26    replace(line,*argv);
27     printf("大文字は小文字,小文字は大文字にする\n");
28     printf("%s→\t",*argv);
29     printf("%s\n",line);
30   }
31   return(0);
32 }
33
34      /*関数 replace */                  /*       関数 replace の流れ     */
35 void replace(char *dest, char *str ){   /* *argv=*str→  *str=*dest     */
36                                         /* *dest=line → line=%s で出力*/
37
38         /*for 文を使った作業*/           /* 文字列の最後まできたら終了 */
39   for(; *str !=NULL ; str++,dest++){    /*それまで str と dest を一緒に 1 ずつ増やす
*/
```

```
40
41      /* *str を変換(大文字を小文字に)*/
42      if (isupper(*str) )
43
44      /* 変換した*str を*dest に代入*/
45      *dest=tolower(*str);
46
47      /* *str を変換(小文字を大文字に)*/
48      else if (islower(*str) )
49
50      /*変換した*str を*dest に代入*/
51      *dest=toupper(*str);
52
53      /*大文字、小文字以外の時はそのまま代入*/
54      else
55        *dest=*str;
56    }
57    *dest=NULL;
58    return;
59 }
```

出力結果；

```
[Masaki:~/PRGRMNG] e075739% ./a.out 777 is lucky number.
argc（文字列の数）=5  (./a.out も含める)
大文字は小文字,小文字は大文字にする
777→   777
大文字は小文字,小文字は大文字にする
is→    IS
大文字は小文字,小文字は大文字にする
lucky→ LUCKY
大文字は小文字,小文字は大文字にする
number.→       NUMBER.
```

説明はプログラム内でも説明しているが、もう一度簡単に説明する。

7 行目 void replace(char *dest, char *str);

関数のプロトタイプ。

19 行目 printf("argc（文字列の数）=%d （./a.out も含める)\n",argc);

文字列の個数を調べる。

22 行目 for(argv++;*argv !=NULL;argv++)

初期が argv++,ということは、プログラム名（./a.out）から読み込む。

条件が*argv !=NULL。NULL というのは文字列の最後ということ。（NULL を'\0'に書き換え可能）

反復文は argv++,ということは、条件が偽になるまで続く

39 行目 for(; *str !=NULL ; str++,dest++)

22 行目の説明とほぼ同じ。初期値が書かれてないので、始めから読み込む

36 行目と 46 行目の replace について

【関数 replace は何か？】

ポインタと同じ位、難しかった（私事だが）のがこの replace という関数。

意味は簡単。繋がっているのだ。行数 L33 を境目に 2 つのプログラムがある。そして、この 2 つのプログラムを繋げる橋として関数 replace が機能している。要するにこうだ。

行数 L1～26 までいった後に、行数 L35 にワープする。そして行数 L59 までいき、行数 L27 に戻る、といった感じだ。（L34 のコメント「関数 replace の流れ」でも述べている。）

53～62 行目　　大文字、小文字の変換

65～66 行目の*dest=*str　大文字、小文字以外の文字の場合はそのまま。

68 行目の*dest=NULL　dest の最後に'\0'が入ってないのでいれている。ちなみに'\0'=NULL である。

文字列反転のプログラムを作る。

方法：

基本的には、文字変換プログラムと一緒なはずだ。

違ってくる所は、行数 L33 行目以降。L33 行目以降は、文字変換プログラムになっているので、この部分を、反転のプログラムに変えればいいはずである。

この反転のプログラムについて考える。

まず、入力された文字列の文字数を数える必要がある。これは、文字列の長さを取得することができる strlen でできる。

それからどうするか。

分からなくなったので例を考え、法則性を見つける。

例:ABC

```
str[0]=A    →    dest[0]='\0'
   [1]=B    →         [1]=C
   [2]=C    →         [2]=B
   [3]='\0' →         [3]=A                    注意：ABC のあとにある、'\0'を忘れないこと
```

上の図は下の図のような関係になっている。

```
*str[i]     →     *dest[i+3]
*str[i+1]   →     *dest[i+2]
*str[i+2]   →     *dest[i+1]
*str[i+3]   →     *dest[i]
```

ここで関係を見てみよう。

文字数は 4。

図の様に str→dest は分かりにくいので、反対の dest→str を考える。

すると、文字数 4 から、dest の[ ]の中身の i、i+1、i+2、i+3、を引くと 4-i、3-i、2-i、1-i となる。ここで i=0 とすると、4,3,2,1 となる。ここで更に引く 1 をすると、3、2、1、0 となる。つまり、

dest の箱番号		str の箱番号
0	→	3
1	→	2
2	→	1
3	→	0

とすることができた。

この関係を式で表すとこうなる。

公式

dest の箱番号[i]=str の箱番号[n-1-i]

(i は 0 から始まり、最終的に文字数と同じになったら終了)

よって、反転する仕組みを作ることができた。

文字列を列ごとに、反転するプログラム

```
1. /*必要なヘッダー*/
2.    #include <stdio.h>
3.    #include <string.h>
4.
5.  /*関数のプロトタイプ*/
6.  void rvs(char *dest, char *str, int n);
7.
8.  /*main 関数　仮変数*/
9.  int main(int argc, char *argv[]){
10.   /*変数宣言*/
11.   int n;
12.   /*文字列の数の表示*/
13.   printf("argc（文字列の数）=%d  (./a.out も含める)\n",argc);
14.
15.   /*文字の認識*/
16.   for(argv++;*argv !=NULL;argv++)
17.      /*認識した文字の出力*/
18.    {  printf("%s→\t",*argv);
```

```
19.       /*関数 strlen  文字列の長さを取得する関数*/
20.        n=strlen(*argv);
21.        char dest [n];
22.        rvs(dest,*argv,n);
23.      }
24.    return;
25.  }
26.
27.  /*関数 rvs*/
28.  void rvs(char *dest, char *str,int n){
29.    int i;
30.    /*反転の仕組み*/
31.    for(i=0;i<=n;i++)
32.      dest[i]=str[n-1-i];
33.    /*反転後の出力*/
34.    printf("%s\n",dest);
35.
36. }
```

出力結果：

```
[Masaki:~/PRGRMNG] e075739% ./a.out The world of speed per hour 100 mile
argc（文字列の数）=9  (./a.out も含める)
The→   ehT
world→ dlrow
of→    fo
speed→ deeps
per→   rep
hour→  ruoh
100→   001
mile→  elim
```

完成

行数 L31～32

反転の仕組みのプログラム

```
for(i=0;i<=n;i++)
    dest[i]=str[n-i-1];
```

今回苦労したのがこれである。説明は 13～14 ページでしている。

上の for 文で使われている『n』は文字列の長さである。

行数 L20 行目の n=strlen(*argv);で示している。

それ以外の仕組みは変わっていない。

感想：

「負けないで～もう少し～最後まで走り抜～けて～。（By　ザード）」が今回のテーマ曲です。

何回も投げ出しそうになり（というか投げ出しました）、ふてくされて寝たこともありました。

今回のプログラミングで神父さんが悪態をつくのも無理はないと実感しました。

ポインタは、分かったと思ったら、分からなくなるというむかつくヤツでした。なんか今回の課題はもう一度出されてもかなり時間がかかると思います。

今回の感想は、泣き言ばかり言っているようなきがします。

がんばって授業についていきたいです。

参考資料

言語ポインタの説明

http://homepage2.nifty.com/natupaji/DxLib/lecture/lectureOthers1.html

初心者の為のポイント C 言語

http://search.jword.jp/cns.dll?type=lk&fm=101&agent=1&partner=Excite&name=%BD%E9%BF%B4%BC%D4%A4%CE%A4%BF%A4%E1%A4%CE%A5%DD%A5%A4%A5%F3%A5%C8%B3%D8%BD%ACC%B8%C0%B8%EC&lang=euc&prop=500&bypass=0&dispconfig=

先輩ズ

http://www.ie.u-ryukyu.ac.jp/~e065745/

http://www.ie.u-ryukyu.ac.jp/~e065701/

http://www.ie.u-ryukyu.ac.jp/~e065702/

http://www.ie.u-ryukyu.ac.jp/~e065762/

もの凄い勢いで C 言語の課題を片付ける

http://c-kadai.sakura.ne.jp/

C 言語プログラミング教室

http://www.eonet.ne.jp/~nao2/c/