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

```
1.  #include <stdio.h>
2.  #include <ctype.h> /*troupper,islower,isupper,tolowerを使うため宣言*/
3.
4.  int get_n(char *);
5.  void replace(char *,char *);
6.  void print_data(char *,char *,int,int);
7.
8.  int main(int argc,char **argv,char **str){
9.      int i,n; /*変数iを宣言、iはパラメータの番号を表す。nはget_n関数からの値を入れるためのもの
    */
10.     char dest[10][10]; /*char型の2次元配列を宣言*/
11.     str = argv;/*strとargvの先頭アドレスを同じにしている/*
12.     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*/
13.     for(i=1,argv++,str++;*argv !=NULL;i++,argv++,str++){
    /*iの初期値を1に、argvとstrのアドレスを1つずらし,argvの値がNULLでないならiに1をたし、argv
    とstrのアドレスを1つずらす*/
14.             n = get_n(*argv);/*関数get_nを呼び出し、その値を出力する*/
15.             replace(*dest,*argv);/*関数replaceの呼び出し*/
16.             print_data(*dest,*str,n,i);
17.             }
18.     }
19.
20. int get_n(char *pa){ /*整数型の関数get_nの定義*/
21.     int i;
22.     for(i=0;*pa !=NULL;i++,pa++);
    /*iをに初期化、*paがNULLになるまでポインタpaとiにを足す*/
23.     return(i);
24.     }
25.
26. void replace(char *dest,char *str){/*void型の関数replaceの定義*/
27.     while(*str !=NULL){ /**strがNULLになるまで繰り返す*/
28.             if(islower(*str) !=0) /*英小文字なら真(0以外の数)を返す。*/
29.                     *dest = toupper(*str);/*小文字を大文字に変えて代入*/
```

```
30.                     else if(isupper(*str) != 0) /*英大文字なら真を返す*/
31.                             *dest = tolower(*str); /*大文字を小文字に変えて代入*/
32.                     else
33.                             *dest = *str; /*上記に当てはまらないときそのまま代入*/
34.                     dest++; /*ポインタdestのアドレスを1つずらす*/
35.                     str++; /*ポインタstrのアドレスを1つずらす*/
36.                     }
37.         *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*/
38.         }
39.   void print_data(char *dest,char *str,int n,int i){/*void型の関数print_dataの定義
40.         printf("¥nparameter(%02d)¥n",i);
41.         printf("文字数= %2d文字",n);
42.         printf("¥t(変換前)%s => (変換後)%s¥n",str,dest);
43.   }
```

実行結果

```
C:¥Program Files¥Microsoft Visual Studio 9.0¥VC¥report6>report0 123 ASDF zxcvb QwErTy
argc = 5

parameter(01)
文字数 =  3 文字 (変換前)123 => (変換後)123

parameter(02)
文字数 =  4 文字 (変換前)ASDF => (変換後)asdf

parameter(03)
文字数 =  5 文字 (変換前)zxcvb => (変換後)ZXCVB

parameter(04)
文字数 =  6 文字 (変換前)QwErTy => (変換後)qWeRtY
```

解説

2 行目: troupper,islower,isupper,tolower を使うために必要なヘッダー

4～6 行目:

main 関数内で関数を使うためのプロトタイプ宣言

関数 replace 関数,print_data 関数は void 型、get_n 関数は int 型で宣言している。

8 行目～main 関数

main 関数にも引数を渡すことができる。この役割をコマンドライン引数という。

渡せる引数は、引数の総個数、引数の文字列を指すポインタの配列の 2 つである

9 行目:変数 i ,n の宣言。i はパラメータの番号を表すために、n は get_n 関数からの値を入れるためのもの。

10 行目 char 型で 2 次元配列 dest を宣言

11 行目:str と argv のアドレスの先頭アドレスを一致させている。

12 行目:main 関数で引き渡された引数の総個数を表示させている。ここで実行結果をみると引数の総個数が 5 となっているのは report0(.exe)が引数の 1 番最初に入っているからである。

13 行目～17 行目 for 文によるループ文。

13行目:iの初期値を1に、argvとstrのアドレスのアドレスを1つずらし,argvの値がNULLでないならiに1をたし、argvとstrのアドレスを1つずらす。iの初期値を1にargvとstrのアドレスのアドレスを1つずらしているのは、report0の変換結果を表示させないためである。ちなみにiの初期値を0にアドレスのアドレスをずらさないと下記ような実行結果が追加される

parameter(00)

文字数 =　7文字 (変換前)report0 => (変換後)REPORT0

14行目:変数nにget_n関数で処理した値が入る。**argvのアドレス(*argv)が*paに引き渡される。

15行目:replace関数により変換をする。replace関数の内容はサブルーチンで説明する。*destは*destに*argvが*strに引き渡されてる。

16行目print_data関数により出力される。print_data関数の内容はサブルーチンで説明する。14行目で出された値が引き渡されて出力される。

20行目～24行目:文字数を求めるget_n関数

スペースにより区切られた物を1つの文字列として文字数として出力される。

Return(i)によりmain関数に引き渡している。

26行目～36行目: replace関数による変換

27行目while文により**strがNULLになるまで繰り返す

28行目islowerによる小文字の判定

29行目:小文字だったらtoupperにより大文字に変換する。

30行目isupperによる大文字の判定

31行目tolowerにより大文字を小文字に変換する。

32、33行目その他(アルファベット以外)の時は、そのまま*destに*strを出力する。

34、35行目:*str、*argvのアドレスに1を足す。

39行目: void型の関数print_dataの定義

40行目:parameterの番号の表示

41行目:各パラメータの文字数を表示

42行目:変換前の文字列と変化後の文字列の表示

考察

| argc | 5 | | | | |
|---|---|---|---|---|---|
| | | | | | |
| argv[0] | report0.exeのアドレス | | | | |
| argv[1] | 123のアドレス | | | | |
| argv[2] | ASDFのアドレス | | | | |
| argv[3] | zxcvbのアドレス | | | | |
| argv[4] | QwErTyのアドレス | | | | |
| | | | | | |
| アドレス | 配列 | 変換前 | アドレス | 配列 | 変換後 |
| 0x373494 | str[0] | A | 0x12ff00 | dest[0] | a |
| 0x373495 | str[1] | S | 0x12ff01 | dest[1] | s |
| 0x373496 | str[2] | D | 0x12ff02 | dest[2] | d |
| 0x373497 | str[3] | F | 0x12ff03 | dest[3] | f |
| | | | | | |
| 0x373499 | str[0] | z | 0x12ff00 | dest[0] | Z |
| 0x37349a | str[1] | x | 0x12ff01 | dest[1] | X |
| 0x37349b | str[2] | c | 0x12ff02 | dest[2] | C |
| | | | | | |
| | | | | | |
| | | | | | |

上記の表のように main 関数へ引数が渡されている。
そして変換前のアドレス、配列、値、変換後のアドレス、配列、値は図のようになっている。

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"）

```
1.  #include <stdio.h>
2.
3.  void reverse(char *,char *,char *);/*reverse関数のプロトタイプ宣言*/
4.
5.  int main(int argc,char **argv){
6.          int i,;
7.          char dest[10][10];
8.          for(i=1,argv++;*argv !=NULL;i++,argv++){
9.                      printf("[parameter][%02d]",i);
10.                     reverse(*dest,*argv,*argv);
11.                     }
```

```
12.         }
13.
14.   void reverse(char *dest,char *argv,char *str){
15.         int i,j;
16.         if(*argv == NULL)
17.                     return;
18.         else
19.                     for(i =0;str[i] != NULL;i++)/* 配列 str[] について null 文字が現れるまで */
20.                               argv[i] = str[i]; /* 配列 argv[] に配列 str[] の内容をコピー */
21.         for(j = i;i >=0;i--) /* 配列 str[] の［要素数］回ループ */
22.                     dest[i] = argv[j-i-1]; /* argv[] の内容を逆から dest[] にコピー */
23.         dest[j+1] = NULL; /* 配列 dest[] の一番うしろに null 文字を追加 */
24.         printf("[反転前] %s -> [反転後]   %s¥n",str,dest);
25.   }
```

実行結果

```
C:¥Program Files¥Microsoft Visual Studio 9.0¥VC¥report6>report2 what are you doing now.
[parameter][01][反転前] what -> [反転後]   tahw
[parameter][02][反転前] are -> [反転後]   era
[parameter][03][反転前] you -> [反転後]   uoy
[parameter][04][反転前] doing -> [反転後]   gniod
[parameter][05][反転前] now? -> [反転後]   ?won
```

解説

3 行目:Reverse 関数のプロトタイプ宣言

6 行目:変数 i の宣言。I はパラメータの番号を表示させるのに使う。

14 行目～24 行目:void 型の reverse 関数

15 行目:変数 i と j の宣言

16 行目:*str[0]の値が NULL だったら何もしない

19、20 行目:それ以外だった場合、配列 str を NULL が表れるまで配列 argv にコピー

21 行目:配列 str の要素数分だけ繰り返す。

22 行目: argv[] の内容を逆から dest[] にコピー

23 行目: 配列 dest[] の一番うしろに null 文字を追加

24 行目:反転前と反転後を表示させる。

**考察**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| argv[2] | areのアドレス | | | argv[1][1] | h | | |
| argv[3] | youのアドレス | | | argv[1][2] | a | | |
| argv[4] | doingのアドレス | | | argv[1][3] | t | | |
| | | | | | | | |
| | 変換前 | | | 変換後 | | | |
| argv[1][0] | w | i=1 | dest[0] | t | | | |
| argv[1][1] | h | | dest[1] | a | | | |
| argv[1][2] | a | | dest[2] | h | | | |
| argv[1][3] | t | | dest[3] | w | | | |
| argv[1][4] | NULL | | dest[4] | NULL | | | |
| : | : | | | | | | |
| argv[5][0] | n | i=5 | dest[0] | ? | | | |
| argv[5][1] | o | | dest[1] | w | | | |
| argv[5][2] | w | | dest[2] | o | | | |
| argv[5][3] | ? | | dest[3] | n | | | |
| argv[5][4] | NULL | | dest[4] | NULL | | | |
| | | | | | | | |
| | | | | | | | |

図のように変換されていると思われる。

エラー報告

コンパイルしたときに

report2.c(22) : warning C4047: '==' : 間接参照のレベルが 'int' と 'void *' で異なっています。

report2.c(25) : warning C4047: '!=' : 間接参照のレベルが 'int' と 'void *' で異なっています。

report2.c(29) : warning C4047: '=' : 間接参照のレベルが 'char' と 'void *' で異なっています。

という風にでるがこの warning を直せなかった。

二重ポインタで宣言してないのに二重配列の型を使ってしまうと下記のように怒られる

report2.c(31) : error C2109: 配列または、ポインタでない変数に添字が使われました。

感想

この課題をやって自分がポインタを理解していないことを痛感しました。

課題を通してポインタの理解が深まったと思います。

参考資料

「c 実践プログラミング」

「新 The UNIX Super Text(上)」

「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

「Makefile の書き方」

http://www.c.csce.kyushu-u.ac.jp/~seiichirou/wiki/index.php?Makefile%A4%CE%BD%F1%A4%AD%CA%FD