1. ポインタ

1.1 変数とアドレス

int a = 256;
「メモリ上のある番地（例 : 1000番地）に、int型変数aの領域を確保し、その領域に値256を代入する」という意味。

int型変数aのアドレス … 1000番地 [　　] ← 整数値「256」を格納

1.2 ポインタとは？

ポインタ = アドレス変数。「変数のアドレス」を記憶する変数のこと。

aのアドレス値 1000 を格納 → [1000] …… ポインタ

1.3 ポインタ演算子

ポインタ演算子	使用例	意味
&	&a	変数aのアドレス(ポインタ値)を取り出す。
*	*p	アドレスpに格納されている値を取り出す。

1.4 配列とアドレス

char m[4] = "ABC"; とした場合それぞれ次のように格納される。

1000番地	'A'
1001番地	'B'
1002番地	'C'
1003番地	'\0'

m	配列mの先頭アドレス（1000）
m[0] m[1] m[2] m[3]	m[0]の値（'A'） m[1]の値（'B'） m[2]の値（'C'） m[3]の値（'\0'）
&m[0] &m[1] &m[2] &m[3]	&m[0]のアドレス（1000） &m[1]のアドレス（1001） &m[2]のアドレス（1002） &m[3]のアドレス（1003）

1.5 ポインタの代入式

代入式	意味
p = str	配列str[]の先頭アドレスの代入
p = &var	変数valの先頭アドレスを代入
p = "ABC"	文字列"ABC"を確保(領域の大きさは"ABC"+1)し、その先頭アドレスを代入
p = pp	ポインタppの値(アドレス)を、ポインタpに代入
var = *p	ポインタpの指すアドレスの値を、変数valに代入
pp = p	ポインタpの値(アドレス)を、ポインタppに代入

2. 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。

2.1 ソースコード : replace.c

```
01  /*
02     Program : replace.c
03     Comment : small -> large & large -> small
04  */
05
06  #include<stdio.h>
07  #include<ctype.h>
08
09  replace(char *dest, char *str);
10
11  int main(int argc, char **argv){
12      char dest[256];
13
14      printf("---------------- result ----------------\n");
15      for(argv++; *argv != NULL; argv++){
16         replase(dest,*argv);
17         printf("before : %s\t\t",*argv);
18         printf("after  : %s\n",dest);
19      }
20      printf("----------------------------------------\n");
21
22      return(0);
23  }
24
25  void replase(char *dest, char *str){
26       for(str; *str != NULL; str++,dest++)
27            {
28             if(islower(*str))
29               *dest = toupper(*str);
30             else if(isupper(*str))
31               *dest = tolower(*str);
32             else
33               *dest = *str;
34            }
35
36     *dest = 0;
37     return;
38  }
```

2.2 解　析 : replase.c

7行目 : 文字処理関数を使用するため。ctype.hを読み込む。

9行目 : 関数replaseを宣言する。引数はchar型アドレス変数 *dest と *str。

11行目 : main関数。引数は、int型変数 argc とchar型アドレス変数 **argv。

12行目 : char型配列 destを宣言。変換後の文字列の保存先。

15行目 : 初期値はargv++ 。実行可能ファイルの ./replace を変換させない為。

16行目 : 関数replaseに、配列 destの先頭アドレスと *argv のアドレスを渡す。

17行目 ： 変換前の文字列を出力する。エスケープシーケンス \t はタブを表す。

18行目 ： 変換後の文字列を出力する。

25行目 ： 16行目の replase(dest,*argv); から値を受け取る。dest を *dest に *argv は *str に代入する。

26行目 ：*str をインクリメントしていき、文字を1文字ずつ取り出す。

28-29行目 ： if 文。取り出した文字が英小文字なら(islower 関数)、大文字に変換(toupper 関数)して、*dest に格納する。

30-31行目 ： else-if 文。取り出した文字が英大文字なら(isupper 関数)、小文字に変換(tolower 関数)して、*dest に格納する。

32-33行目 ： else 文。取り出した文字がアルファベットでないならば、取り出した文字をそのまま *dest に格納する。

36行目 ： *dest = 0; は、文字列の最後に'\0'(NULL 値)を格納する。

2.3 実行結果 ： replase.c

```
new-host:Report6 kouta$ ./replace ThiS IS a pEn.
---------------- result ----------------
before : ThiS       after  : tHIs
before : IS         after  : is
before : a          after  : A
before : pEn.       after  : PeN.
----------------------------------------
```

2.4 考　察 ： replase.c

コマンドラインから受け取った文字列は以下の表の様に格納されていると考えられる。

*argv	.	/	r	e	p	l	a	c	e	\0
*argv+1	T	h	i	S	\0					
*argv+2	I	S	\0							
*argv+3	a	\0								
*argv+4	p	E	n	\0						

1回目のループでは、関数 replace に *argv+1 のアドレスを、2回目は、*argv+2 のアドレスを、……、n回目は、*argv+n のアドレスをという具合で、*argv の値が NULL 値になるまで繰り返す。今回の場合は、4回 replace 関数にアドレスを渡したことになる。

関数 replace では、main 関数から受け取ったアドレスをそれぞれ *dest,*str に格納している。

*str をインクリメントすることで、1文字ずつ取り出し変換処理を行っている。
小文字は大文字に、大文字は小文字に変換していて、その他の記号はそのまま格納。

配列 dest の最後に'\0'(NULL 値)を格納して、main 関数に戻る。

3. 文字列を反転して表示させるプログラムを作成せよ。

3.1 ソースコード : reverse.c

```
01  /*
02    Program : reverse.c
03    Comment : Reversal of the string
04  */
05
06  #include<stdio.h>
07
08  void reverse(char *dest, char *str);
09
10  int main(int argc, char **argv){
11      char dest[256];
12
13      printf("---------------- result ----------------\n");
14      for(argv++; *argv != NULL; argv++){
15         reverse(dest,*argv);
16         printf("before : %s\t\t",*argv);
17         printf("after  : %s\n",dest);
18      }
19      printf("----------------------------------------\n");
20
21      return(0);
22  }
23
24  void reverse(char *dest, char *str){
25       char *temp;
26
27       for(temp; *str != NULL; str++)
28          temp = str;
29
30       for(temp,dest; *temp != NULL; temp--,dest++)
31          *dest = *temp;
32
33       *dest = NULL;
34       return;
35  }
```

3.2 解　析 : reverse.c

8行目 : 関数reverseを宣言する。引数はchar型アドレス変数 *dest と *str。

10-22行目 : main関数。replace.cと同じ処理をしているので省略(p.2～3参照)。

24行目 : main関数から値を受け取り、それぞれのアドレスを *dest、*str に格納。

25行目 : char型アドレス変数 *tempを宣言する。

27-28行目 : *str の文字数をカウント。

30-31行目 : *temp をデクリメントすることで、文字列を後ろからコピーしている。

33行目 : 文字列の最後にNULL値をセットする。

3.3 実行結果 ： reverse.c

```
new-host:Report6 kouta$ ./reverse abcde This is a pen.
---------------- result ----------------
before : abcde      after  : edcba
before : This       after  : sihT
before : is         after  : si
before : a          after  : a
before : pen.       after  : .nep
----------------------------------------
```

3.4 考　察 ： reverse.c

① *str を NULL までインクリメントして、文字数をカウントする。

a	b	c	d	e	NULL	T	h	i	s	NULL	i	s	NULL	a	NULL	p	e	n	.	NULL

*str = 5 となり、*temp に5 を代入する。

② temp の値を初期値としているので、灰色の部分から処理が始まる。

a	b	c	d	e	NULL	T	h	i	s	NULL	i	s	NULL	a	NULL	p	e	n	.	NULL

③ temp

a	b	c	d	e	NULL	T	h	i	s	NULL	i	s	NULL	a	NULL	p	e	n	.	NULL

dest

e																				

1回目のループ終了後、temp と dest の内容は上のようになっている。

④ temp がデクリメントされ、③と同様の処理が行われる。

a	b	c	d	e	NULL	T	h	i	s	NULL	i	s	NULL	a	NULL	p	e	n	.	NULL

dest

e	d																			

2回目のループ終了後、temp と dest の内容は上のようになっている。

⑤ 2回目以降も同様の処理が行われ、全ての処理が終わると以下のような結果になる。

e	d	c	b	a	NULL	s	i	h	T	NULL	s	i	NULL	a	NULL	.	n	e	p	NULL

4. 感　想

文字列の大文字と小文字を変換するプログラムは、例題プログラムを参考にしながら割と楽に作成することが出来ました。

2つ目の文字列を反転させるプログラムは、上手く反転させることが出来ず3時間ほど悩みました。文字数をカウントという、簡単な処理を上手く組み合わせることで解決して、プログラムは、ヒラメキだなぁと今更実感しました。

今回の課題で、ある程度ポインタは理解したつもりなので、構造体や共用体も理解できるよう頑張りたいと思います。

5. 参考文献

- C実践プログラミング 第3版 谷口 功(訳) 望月 康司(監訳) Steve Oualline(著)
- 初心者のためのポイント学習C言語　http://www9.plala.or.jp/sgwr-t/
- ProgI/2009　http://www.osn.u-ryukyu.ac.jp/lecture/wiki/index.php