プログラミング 1

Report#6

提出日　：2013 年 6 月 27 日
所　属　：工学部情報工学科
学籍番号：e135732J
氏　名　：前城　健太郎

Report#6

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。また、文字列を反転して表示するプログラムも作成せよ。

１．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の作成

1.1 プログラムのソースコード全体(psample-2.c)

```
1   #include <stdio.h>
2
3   int  get_n(char *);
4   void replace(char *,char *);
5
6   int main(int argc, char **argv){
7     int   i,n,m;
8     printf("argc = %d\n",argc);
9
10    puts("-----");
11    i = 1,m = 1,argv++;
12    while (m != argc){
13      printf("parameter(%2d)\t%s\n",i,*argv);
14      n = get_n(*argv);
15      char dest[n+1];
16      replace(dest,*argv);
17      printf("%s ---> %s\n",*argv,dest);
18      i++,m++;argv++;
19    }
20    puts("-----");
21    return(0);
22  }
23
24
25  int get_n(char *pa){
26    int i;
27    for (i=0; *pa != NULL; i++,pa++);
28    return(i);
29  }
30
31  void replace(char *dest, char *str){
32    while(*str != NULL){
33      if('A' <= *str && *str <= 'Z')
34        *dest = *str + 32;
35      else if('a' <= *str && *str <= 'z')
36        *dest = *str - 32;
37      else
38        *dest = *str;
39      str++,dest++;
40    }
41    *dest = NULL;
42  }
```

1.2 コマンドラインでのコンパイルと実行時のコマンド

```
$ gcc -o psample2 psample-2.c
$ ./psample2 abc DEF gieYFsorfgwiHV foG72TQ fs
```

1.3 実行結果

```
argc = 6
-----
parameter( 1)     abc
abc ---> ABC
parameter( 2)     DEF
DEF ---> def
parameter( 3)     gieYFsorfgwiHV
gieYFsorfgwiHV ---> GIEyfSORFGWIhv
parameter( 4)     foG72TQ
foG72TQ ---> FOg72tq
parameter( 5)     fs
fs ---> FS
-----
```

1.4 考察

a) 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した.

b) 7 行目の main()関数の引数が(int argc, char **argv)となっており,argc に引数の個数が,*argv に引数それぞれの文字列が格納される.

c) 16 行目において,文字列を変換後の文字列を格納するための変数をその引数の数(最後の NULL を含めて)だけ宣言している.

d) 26-30 行の get_n()関数では,コマンドラインから受け取った引数の文字数をカウントし,int 型で返すような動作をする.

e) 33-42 行の replace()関数では引数の文字の大文字と小文字を if 文により判断し,用意された変数へ格納している.

2.　文字列を反転して表示するプログラムの作成

2.1 ソースコード全体(psample-3.c)

```
1   #include <stdio.h>
2
3   int  get_n(char *);
4   void replace(char *,char *);
5
6   int main(int argc, char **argv){
7     int   i,n,m;
8     printf("argc = %d\n",argc);
9
10    puts("-----");
11    i = 1,m = 1,argv++;
12    while (m != argc){
13      printf("parameter(%2d)\t%s\n",i,*argv);
14      n = get_n(*argv);
15      char dest[n+1];
16      replace(dest,*argv);
17      printf("%s ---> %s\n",*argv,dest);
18      i++,m++,argv++;
19    }
20    puts("-----");
21    return(0);
22  }
23
24
25  int get_n(char *pa){
26    int i;
27    for (i=0; *pa != NULL; i++,pa++);
28    return(i);
29  }
30
31  void replace(char *dest, char *str){
32    int i = 0;
33    int n = get_n(str) - 1;
34    char *tmp;
35    while(i <= n){
36      tmp = str + (n-i);
37      *dest = *tmp;
38      dest = dest +1;
39      i++;
40    }
41    *dest = NULL;
42  }
```

2.2 コマンドラインでのコンパイルと実行時のコマンド

```
$ gcc -o psample3 psample-3.c
$ ./psample3 abc DEF gieYFsorfgwiHV foG72TQ fs
```

2.3 実行結果

```
argc = 6
-----
parameter( 1)     abc
abc ---> cba
parameter( 2)     DEF
DEF ---> FED
parameter( 3)     gieYFsorfgwiHV
gieYFsorfgwiHV ---> VHiwgfrosFYeig
parameter( 4)     foG72TQ
foG72TQ ---> QT27Gof
parameter( 5)     fs
fs ---> sf
-----
```

2.4 考察

a) コマンドラインから受け取った文字列を反転して表示するプログラムを作成した.
b) 1.1 項のソースコード(psample-2.c)をそのまま使用し，replace()関数の部分だけを書き換えた.
c) 新しい replace()関数では,ポインタを逆から辿りそのポインタ内に格納されているアドレスを確保するポインタを一時的に用意している.
d) 一時的なポインタが指す変数を予め用意された変数*dest に格納し,dest をインクリメントすることで次の変換された文字を格納する準備をしている.

3. コマンドラインパラメータにおける二重ポインタの考察

3.1 方針

以下のプログラム(argcv1.c)を用いてポインタ内に格納されているアドレスとポインタ自体のアドレスを表示し,コマンドラインパラメータにおける二重ポインタの挙動を観察し考察する.また,アドレスを直接入力し変数を呼び出すプログラム(argcv2.c)を用いてより考察を深める.

3.2 ソースコード全体(argcv1.c)

```
1   #include <stdio.h>
2
3   int  get_n(char *);
4   void print_data(char *,int);
5
6   int main(int argc, char **argv){
7     int  i, n ,m;
8     char *str;
9
10    printf(" argc = %d\n",argc);
11    printf("&argc = %08x\n",&argc);
12
13    puts("-----");
14    i=0,m=0;
15    while (m != argc){
16      printf("parameter(%2d)\t%s\n",i,*argv);
17      n = get_n(*argv);
18      print_data(*argv,n);
19      for (str = *argv; *str != NULL; str++){
20        printf("Character ==> %8c ( *str =%08x)\n",*str,&*str);
21      }
22      printf(" *argv = %08x, ",*argv);
23      printf("&*argv = %08x\n",&*argv);
24      printf("  argv = %08x, ",argv);
25      printf("&argv = %08x\n",&argv);
26      printf(" (sub) = %08x\n",(long)argv-(long)*argv);
27      puts("-----");
28      i++,m++,argv++;
29    }
30    return(0);
31  }
32
33  int get_n(char *pa){
34    int i;
35    for (i=0; *pa != NULL; i++,pa++);
36    return(i);
37  }
38
39  void print_data(char *pa,int n){
40    printf("n=%2d  %s\n",n,pa);
41  }
```

3.3 コマンドラインパラメータを変えた実行その 1

3.3.1 コマンドラインでのコンパイルと実行時のコマンド

```
$ cc -o argcv1 argcv1.c
$ ./argcv1 abc DEF igHQ549 m ufoanwufovnhfh
```

3.3.2 実行結果(1)

```
 argc = 6
&argc = 5c5c89f8
-----
parameter( 0)     ./argcv1
n= 8  ./argcv1
Character ==>        . ( *str =5c5c8b88)
Character ==>        / ( *str =5c5c8b89)
Character ==>        a ( *str =5c5c8b8a)
Character ==>        r ( *str =5c5c8b8b)
Character ==>        g ( *str =5c5c8b8c)
Character ==>        c ( *str =5c5c8b8d)
Character ==>        v ( *str =5c5c8b8e)
Character ==>        1 ( *str =5c5c8b8f)
 *argv = 5c5c8b88, &*argv = 5c5c8a20
  argv = 5c5c8a20, &argv = 5c5c89f0
 (sub) = fffffe98
-----
parameter( 1)     abc
n= 3  abc
Character ==>        a ( *str =5c5c8b91)
Character ==>        b ( *str =5c5c8b92)
Character ==>        c ( *str =5c5c8b93)
 *argv = 5c5c8b91, &*argv = 5c5c8a28
  argv = 5c5c8a28, &argv = 5c5c89f0
 (sub) = fffffe97
-----
parameter( 2)     DEF
n= 3  DEF
Character ==>        D ( *str =5c5c8b95)
Character ==>        E ( *str =5c5c8b96)
Character ==>        F ( *str =5c5c8b97)
 *argv = 5c5c8b95, &*argv = 5c5c8a30
  argv = 5c5c8a30, &argv = 5c5c89f0
 (sub) = fffffe9b
-----
parameter( 3)     igHQ549
n= 7  igHQ549
Character ==>        i ( *str =5c5c8b99)
Character ==>        g ( *str =5c5c8b9a)
Character ==>        H ( *str =5c5c8b9b)
Character ==>        Q ( *str =5c5c8b9c)
Character ==>        5 ( *str =5c5c8b9d)
Character ==>        4 ( *str =5c5c8b9e)
Character ==>        9 ( *str =5c5c8b9f)
 *argv = 5c5c8b99, &*argv = 5c5c8a38
  argv = 5c5c8a38, &argv = 5c5c89f0
 (sub) = fffffe9f
-----
parameter( 4)     m
n= 1  m
Character ==>        m ( *str =5c5c8ba1)
 *argv = 5c5c8ba1, &*argv = 5c5c8a40
  argv = 5c5c8a40, &argv = 5c5c89f0
 (sub) = fffffe9f
-----
parameter( 5)     ufoanwufovnhfh
n=14  ufoanwufovnhfh
Character ==>        u ( *str =5c5c8ba3)
Character ==>        f ( *str =5c5c8ba4)
Character ==>        o ( *str =5c5c8ba5)
Character ==>        a ( *str =5c5c8ba6)
Character ==>        n ( *str =5c5c8ba7)
Character ==>        w ( *str =5c5c8ba8)
Character ==>        u ( *str =5c5c8ba9)
Character ==>        f ( *str =5c5c8baa)
Character ==>        o ( *str =5c5c8bab)
Character ==>        v ( *str =5c5c8bac)
Character ==>        n ( *str =5c5c8bad)
Character ==>        h ( *str =5c5c8bae)
Character ==>        f ( *str =5c5c8baf)
Character ==>        h ( *str =5c5c8bb0)
 *argv = 5c5c8ba3, &*argv = 5c5c8a48
  argv = 5c5c8a48, &argv = 5c5c89f0
 (sub) = fffffea5
-----
```

3.3.3 実行結果(2)

```
 argc = 6
&argc = 505ca9f8
-----
parameter( 0)     ./argcv1
n= 8  ./argcv1
Character ==>        . ( *str =505cab88)
Character ==>        / ( *str =505cab89)
Character ==>        a ( *str =505cab8a)
Character ==>        r ( *str =505cab8b)
Character ==>        g ( *str =505cab8c)
Character ==>        c ( *str =505cab8d)
Character ==>        v ( *str =505cab8e)
Character ==>        1 ( *str =505cab8f)
 *argv = 505cab88, &*argv = 505caa20
  argv = 505caa20, &argv = 505ca9f0
 (sub) = fffffe98
-----
parameter( 1)     abc
n= 3  abc
Character ==>        a ( *str =505cab91)
Character ==>        b ( *str =505cab92)
Character ==>        c ( *str =505cab93)
 *argv = 505cab91, &*argv = 505caa28
  argv = 505caa28, &argv = 505ca9f0
 (sub) = fffffe97
-----
parameter( 2)     DEF
n= 3  DEF
Character ==>        D ( *str =505cab95)
Character ==>        E ( *str =505cab96)
Character ==>        F ( *str =505cab97)
 *argv = 505cab95, &*argv = 505caa30
  argv = 505caa30, &argv = 505ca9f0
 (sub) = fffffe9b
-----
parameter( 3)     igHQ549
n= 7  igHQ549
Character ==>        i ( *str =505cab99)
Character ==>        g ( *str =505cab9a)
Character ==>        H ( *str =505cab9b)
Character ==>        Q ( *str =505cab9c)
Character ==>        5 ( *str =505cab9d)
Character ==>        4 ( *str =505cab9e)
Character ==>        9 ( *str =505cab9f)
 *argv = 505cab99, &*argv = 505caa38
  argv = 505caa38, &argv = 505ca9f0
 (sub) = fffffe9f
-----
parameter( 4)     m
n= 1  m
Character ==>        m ( *str =505caba1)
 *argv = 505caba1, &*argv = 505caa40
  argv = 505caa40, &argv = 505ca9f0
 (sub) = fffffe9f
-----
parameter( 5)     ufoanwufovnhfh
n=14  ufoanwufovnhfh
Character ==>        u ( *str =505caba3)
Character ==>        f ( *str =505caba4)
Character ==>        o ( *str =505caba5)
Character ==>        a ( *str =505caba6)
Character ==>        n ( *str =505caba7)
Character ==>        w ( *str =505caba8)
Character ==>        u ( *str =505caba9)
Character ==>        f ( *str =505cabaa)
Character ==>        o ( *str =505cabab)
Character ==>        v ( *str =505cabac)
Character ==>        n ( *str =505cabad)
Character ==>        h ( *str =505cabae)
Character ==>        f ( *str =505cabaf)
Character ==>        h ( *str =505cabb0)
 *argv = 505caba3, &*argv = 505caa48
  argv = 505caa48, &argv = 505ca9f0
 (sub) = fffffea5
-----
```

3.4.コマンドラインパラメータを変えた実行その２

3.4.1 コマンドラインでのコンパイルと実行時のコマンド

```
$ cc -o argcv1 argcv1.c
$ ./argcv1 dr sifgh sia
```

3.4.2 出力結果(1)

```
 argc = 4
&argc = 5b0a1a18
-----
parameter( 0)     ./argcv1
n= 8  ./argcv1
Character ==>        . ( *str =5b0a1ba0)
Character ==>        / ( *str =5b0a1ba1)
Character ==>        a ( *str =5b0a1ba2)
Character ==>        r ( *str =5b0a1ba3)
Character ==>        g ( *str =5b0a1ba4)
Character ==>        c ( *str =5b0a1ba5)
Character ==>        v ( *str =5b0a1ba6)
Character ==>        1 ( *str =5b0a1ba7)
 *argv = 5b0a1ba0, &*argv = 5b0a1a48
  argv = 5b0a1a48, &argv = 5b0a1a10
 (sub) = fffffea8
-----
parameter( 1)     dr
n= 2  dr
Character ==>        d ( *str =5b0a1ba9)
Character ==>        r ( *str =5b0a1baa)
 *argv = 5b0a1ba9, &*argv = 5b0a1a50
  argv = 5b0a1a50, &argv = 5b0a1a10
 (sub) = fffffea7
-----
parameter( 2)     sifgh
n= 5  sifgh
Character ==>        s ( *str =5b0a1bac)
Character ==>        i ( *str =5b0a1bad)
Character ==>        f ( *str =5b0a1bae)
Character ==>        g ( *str =5b0a1baf)
Character ==>        h ( *str =5b0a1bb0)
 *argv = 5b0a1bac, &*argv = 5b0a1a58
  argv = 5b0a1a58, &argv = 5b0a1a10
 (sub) = fffffeac
-----
parameter( 3)     sia
n= 3  sia
Character ==>        s ( *str =5b0a1bb2)
Character ==>        i ( *str =5b0a1bb3)
Character ==>        a ( *str =5b0a1bb4)
 *argv = 5b0a1bb2, &*argv = 5b0a1a60
  argv = 5b0a1a60, &argv = 5b0a1a10
 (sub) = fffffeae
-----
```

3.4.3 出力結果(2)

```
 argc = 4
&argc = 58895a18
-----
parameter( 0)     ./argcv1
n= 8  ./argcv1
Character ==>        . ( *str =58895ba0)
Character ==>        / ( *str =58895ba1)
Character ==>        a ( *str =58895ba2)
Character ==>        r ( *str =58895ba3)
Character ==>        g ( *str =58895ba4)
Character ==>        c ( *str =58895ba5)
Character ==>        v ( *str =58895ba6)
Character ==>        1 ( *str =58895ba7)
 *argv = 58895ba0, &*argv = 58895a48
  argv = 58895a48, &argv = 58895a10
 (sub) = fffffea8
-----
parameter( 1)     dr
n= 2  dr
Character ==>        d ( *str =58895ba9)
Character ==>        r ( *str =58895baa)
 *argv = 58895ba9, &*argv = 58895a50
  argv = 58895a50, &argv = 58895a10
 (sub) = fffffea7
-----
parameter( 2)     sifgh
n= 5  sifgh
Character ==>        s ( *str =58895bac)
Character ==>        i ( *str =58895bad)
Character ==>        f ( *str =58895bae)
Character ==>        g ( *str =58895baf)
Character ==>        h ( *str =58895bb0)
 *argv = 58895bac, &*argv = 58895a58
  argv = 58895a58, &argv = 58895a10
 (sub) = fffffeac
-----
parameter( 3)     sia
n= 3  sia
Character ==>        s ( *str =58895bb2)
Character ==>        i ( *str =58895bb3)
Character ==>        a ( *str =58895bb4)
 *argv = 58895bb2, &*argv = 58895a60
  argv = 58895a60, &argv = 58895a10
 (sub) = fffffeae
-----
```

3.6 考察

a) argv をインクリメントすると,argv に格納されているアドレスの値が増加する.
b) インクリメントによるアドレスの増加量は状況に応じて変化している.
c) ポインタ argv に格納されてるアドレスが変化することで*argv として扱われるメモリのアドレスが変更され,それに伴って**argv の値も変化する.
d) *argv が指すアドレスと,その引数の先頭文字のアドレスが一致している.
e) よって,ポインタが複数重なっている場合,*argv をインクリメントすると示す値は次の文字ではなく,次の文字列の先頭文字である.
f) *argv の値をインクリメントではなく別の二重以上でないポインタにアドレスを渡すことでエラーを避け,同じ文字列内で次の文字を参照することが出来る.
g) 以上より,２重ポインタを用いることで複数の文字をあたかも文字列であるかのように扱うことができる.
h) ある配列に格納されている文字を文字列として処理したいとき,文字列としてサブルーチンに渡したいときなど大きな効果を発揮する.

i) また,コマンドラインにおいて実行するコマンドを２回繰り返すと,ポインタ内や argv のアドレスが変化している(3.4,3.5 項の実行結果(2)).３回目以降も同じく変化した.
j) しかし,argv と*argv のアドレスの差には変化は見られなかった(実行結果における”(sub)=”の部分).
k) よって,実行するたびにアドレスは変化するが,アドレスの差,つまり距離は変化しないことがわかる.

l) ポインタ argv について.どの実行結果においてもポインタ内のアドレスの末尾は０か８であり,インクリメントするごとにアドレスは８づつ増加している.
m) ポインタ*argv について.格納されているアドレスは変数**argv の先頭を表している.
n) また,ある argv が指すポインタ*argv と argv をインクリメントした際に指すポインタ*argv のアドレスは,最初の*argv が指している変数**argv の数+１の数の開きがある.
o) +1 だけ多いのは,配列と同様に最後には NULL が格納されている変数が存在しているためである.

3.7 アドレスの入力

3.7.1 アドレスを直接入力し任意の変数を得る実験

3.7.2 ソースコード全体(argcv2.c)

```
#include <stdio.h>

int  get_n(char *);
void print_data(char *,int);
int in_add();

int main(int argc, char **argv){
  int   i, n ,m,e;
  char *str;

  printf(" argc = %d\n",argc);
  printf("&argc = %08x\n",&argc);

  puts("-----");
  i=1,m=1;
  argv++;
  while (m != argc){
    printf("parameter(%2d)\t%s\n",i,*argv);
    n = get_n(*argv);
    print_data(*argv,n);
    printf(" *argv = %08x, ",*argv);
    printf("&*argv = %08x\n",&*argv);
    printf("  argv = %08x, ",argv);
    printf("&argv = %08x\n",&argv);
    printf(" (sub) = %08x\n",(long)argv-(long)*argv);
    str = *argv,e=0;
    while (e != n){
      printf("Address? ==> ");
      scanf("%x",&str);
      printf("Character ==> %8c\n",*str);
      str++,e++;
    }
    puts("-----");
    i++,m++,argv++;
  }
  return(0);
}


int get_n(char *pa){
  int i;

  for (i=0; *pa != NULL; i++,pa++);
  return(i);
}

void print_data(char *pa,int n){
  printf("n=%2d  %s\n",n,pa);
}
```

3.7.3 コマンドラインでのコンパイルと実行時のコマンド

```
cc -o argcv2 argcv2.c
./argcv2 irnotpe
```

3.7.4 出力結果　※scanf()関数には順番に 506a5bae, 506a5bac, 506a5ba9, 506a5bab, 506a5bad, 506a5baf, 506a5baa を入力

```
 argc = 2
&argc = 506a5a28
-----
parameter( 1)     irnotpe
n= 7  irnotpe
 *argv = 506a5ba9, &*argv = 506a5a60
  argv = 506a5a60, &argv = 506a5a20
 (sub) = fffffeb7
Address? ==> 506a5bae
Character ==>        p
Address? ==> 506a5bac
Character ==>        o
Address? ==> 506a5ba9
Character ==>        i
Address? ==> 506a5bab
Character ==>        n
Address? ==> 506a5bad
Character ==>        t
Address? ==> 506a5baf
Character ==>        e
Address? ==> 506a5baa
Character ==>        r
-----
```

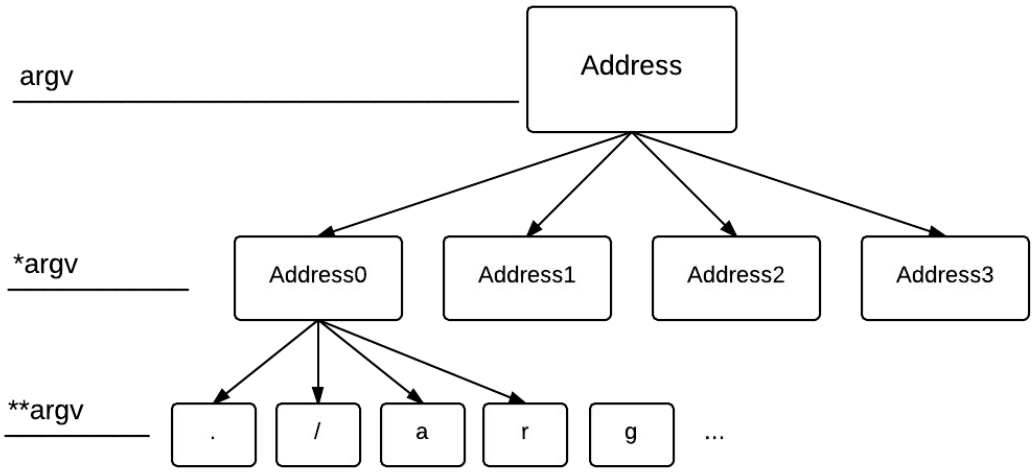
3.7.5 考察

a) コマンドラインパラメータによって文字が格納された変数のアドレスを予想し,scanf()関数によってポインタにアドレスを直接入力し任意の変数を呼び出した.

b) 今までの考察から,**argv の先頭のアドレスはポインタ*argv に格納されているアドレスと一致しており,**argv をインクリメントするとその隣の文字を表すことがわかる.

c) 狙い通りの出力を得ることができた.

3.8 図による二重ポインタの考察



a) 図によって二重ポインタの構造を表した.
b) 最上位のポインタ argv の初期状態では*argv の Address0 を表し,インクリメントする度に右へポインタが指すものが移っている.
c) 最下位の**argv も同様に,インクリメントすると右へ移っていく.

--あとがき--

a)参考文献

1.『C 実践プログラミング 第三版』

b)感想と反省

1.c 言語で最難関を言われるポインタ.死ぬかと思いました.