プログラミング　第 3 回　レポート

202212022 田島瑞起

2023/06/20

1　はじめに

今回の課題では, 配列のマッチング (設問 1),strcmp の実装 (設問 2),strcat の実装 (設問 3) について latex を用いてレポート作成する。

2　配列のマッチング (設問 1)

2.1　課題内容の説明

標準入力を複数回受け付け, その結果をもとに入力内容をリストを作成し, 再度標準入力を受け付けた際, その内容がリストに含まれているか判定できるプログラムを作成する。

2.2　課題への取り組み方針

まず, 入力されたテキストから改行コードを削除する外部関数 chomp() を実装する。main 関数内では, 複数回の標準入力をリスト化可能である事、検索マッチング可能である事, 以上 2 条件を満たすよう実装する。

2.3　解答結果

図 1　s2212022-1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ASIZE 10

char* chomp(char* s){
    int i;
    for(i=0;s[i] != '\0';i++){
        if(s[i] == '\n'){
            s[i] = '\0';
        }
    }
}


int main(int ac,char* av[]){
    char* s[ASIZE];
    char search[ASIZE];
    int i;

    for(i=0;i<ASIZE;i++){
        char buf[ASIZE];
        char* p;
        printf("input␣string:");
        chomp(fgets(buf,ASIZE,stdin));
        p = (char*)malloc(sizeof(
          strlen(buf) + 1));

        if(p == NULL){
            printf("Memory␣allocation␣
              error!\n");
            exit(EXIT_FAILURE);
        }
        strcpy(p,buf);
        s[i] = p;
    }
    printf("input␣string␣list\n");
    printf("-----------------\n");
    for(i=0;i<ASIZE;i++){
        printf("No.%d␣:%s␣\n",i+1,s[i
          ]);
    }
    printf("-----------------\n");

    printf("search␣string:");
    chomp(fgets(search,ASIZE,stdin));
```

```
45
46          for(i=0;i<ASIZE;i++){
47              if(strcmp(s[i],search) == 0){
48                  printf("found␣at␣index␣%d␣
                      in␣the␣array␣:␣%s",i+1,s
                      [i]);
49                  break;
50              }
51              if(i == ASIZE -1){
52                  printf("not␣found␣:%s\n",
                      search);
53              }
54          }
55
56      }
```

図 1 にて確認できるように, 改行処理は chomp 関数で実装した。配列を受け取り, ループ中に改行コードを発見したら配列の末尾文字で置き換えてリターンするという処理内容になっている。標準入力を複数回受け付け, リストを作成するため,char* s[ASIZE] を用意し, この文字列のポインタを格納する配列に, 標準入力で受け付けた文字列のポインタを格納する仕組みをループ文で実装した。入力を受け付け,buf[ASIZE] に文字列を格納し, 動的にメモリを作成して, その箇所をポインタ p で指定し,p に buf をコピーし, 最後にポインタの配列である s に代入するという流れだ。そして最後にマッチングを実装するために文字列比較可能な strcmp を利用した。

2.4 確認

入力に対して, マッチング結果が条件を満たしているか確認できれば良い。

図 2 test1

```
1   input string:adsfae
2   input string:aga
3   input string:gaer
4   input string:sdfa
5   input string:sda
6   input string:adgar
7   input string:fdgar
8   input string:afae
9   input string:dasfea
10  input string:daf
11  input string list
12  -----------------
13  No.1 :adsfae
14  No.2 :aga
15  No.3 :gaer
16  No.4 :sdfa
17  No.5 :sda
18  No.6 :adgar
19  No.7 :fdgar
20  No.8 :afae
21  No.9 :dasfea
22  No.10 :daf
23  -----------------
24  search string:sda
25  found at index 5 in the array : sda
```

図 3 test2

```
1   input string:asdf
2   input string:gafd
3   input string:agadsa
4   input string:garedfg
5   input string:a
6   input string:df
7   input string:a
8   sinput string:ef
9   input string:a
10  input string:dfsgdv
11  input string list
12  -----------------
13  No.1 :asdf
14  No.2 :gafd
15  No.3 :agadsa
16  No.4 :garedfg
17  No.5 :a
18  No.6 :df
19  No.7 :a
20  No.8 :sef
21  No.9 :a
22  No.10 :dfsgdv
23  -----------------
24  search string:are
25  not found :are
```

上記二つの条件を満たしていることがわかる。

3 strcmp の実装 (設問 2)

3.1 strcmp を用いた文字列マッチング (設問 2-1)

strcmp を用いて標準入力から受け付けた文字列のマッチングを行う。

3.1.1 課題への取り組み方針

設問 1 で作成した chomp 関数で入力した文字列の改行コードをそぎ落とし, その後 strcmp で文字列のマッチングを行う。

3.1.2 解答結果

図 4 s2212022-2-1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 100

char* chomp(char* s){
    int i;
    for(i=0;s[i] != '\0';i++){
        if(s[i] == '\n'){
            s[i] = '\0';
        }
    }
}

int main(int ac, char* av[]){
    char s1[SIZE];
    char s2[SIZE];

    printf("input␣␣first␣string:\n");
    chomp(fgets(s1,SIZE,stdin));
    printf("input␣␣second␣string:\n");
    chomp(fgets(s2,SIZE,stdin));

    printf("the␣result␣of␣comparing(%s
      ,%s)␣=␣%d\n",s1,s2,strcmp(s1,s2
      ));

}
```

条件を満たすように実装した結果、図 4 となった。

3.1.3 確認

数回呼び出して strcmp 関数が正常に動作しているか確認すると、

図 5 test3

```
input  first string:
abc
input  second string:
abcd
the result of comparing(abc,abcd) =
  -100
```

図 6 test4

```
input  first string:
asdfg
input  second string:
asdfg
the result of comparing(asdfg,asdfg) =
   0
```

図 7 test5

```
input  first string:
asdfgh
input  second string:
asdfg
the result of comparing(asdfgh,asdfg)
  = 104
```

正の値, 負の値, 零が期待通りに帰っているので正常に動作していることがわかる。

3.2 mystrcmp の実装 (設問 2-2)

strcmp を用いて標準入力から受け付けた文字列のマッチングを行う。

3.2.1 課題への取り組み方針

mystrcmp は引数に配列 2 つ, 返り値に状況に応じた整数値が対応するように実装する。受け取った配列のサイズを strlen() で測り, その長さに応じて処理を変化させる。

3.2.2 解答結果

図 8　s2212022-2-2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 100

char* chomp(char* s){
    int i;
    for(i=0;s[i] != '\0';i++){
        if(s[i] == '\n'){
            s[i] = '\0';
        }
    }
}

int mystrcmp(const char *s1,const char *s2){
    int l1;
    int l2;
    int i;

    l1 = strlen(s1);
    l2 = strlen(s2);
    if(l1>=l2){
        for(i=0;i<l2;i++){
            if(s1[i] != s2[i]){
                return i;
            }
        }
        if(l1 != l2){
            return l2;
        }else{
            return -1;
        }
    }

    if(l1<l2){
        for(i=0;i<l1;i++){
            if(s1[i] != s2[i]){
                return i;
            }
        }
        return l1;
    }
}

int main(int ac, char* av[]){
    char s1[SIZE];
    char s2[SIZE];

    printf("input␣␣first␣string:\n");
    chomp(fgets(s1,SIZE,stdin));

    printf("input␣␣second␣string:\n");
    chomp(fgets(s2,SIZE,stdin));

    printf("the␣result␣of␣comparing(%s,%s)␣=␣%d\n",s1,s2,mystrcmp(s1,s2));

}
```

図 8 では mystrcmp の条件を満たすために、l1 と l2 の大小関係によってリターンする値を分岐させた。

3.2.3　確認

l1>l2,l1=l2,l1<l2 の三通りの結果が期待通りになるか確認すると、

図 9　test6

```
input  first string:
asdfgh
input  second string:
asdf
the result of comparing(asdfgh,asdf) =
   4
```

図 10　test7

```
input  first string:
asdfgh
input  second string:
asdfgh
the result of comparing(asdfgh,asdfgh)
   = -1
```

図 11　test8

```
input  first string:
asdfg
input  second string:
asxchsdf
the result of comparing(asdfg,asxchsdf
  ) = 2
```

3 通り確認し,mystrcmp() が正常に動作していることを確認できた。

4 mystrcat の実装 (設問 3)

4.1 課題内容の説明

標準入力を 2 回受け付けた後に、それらの文字列を結合した文字列を返却するような mystrcat() を実装する。

4.2 課題への取り組み方針

標準入力で受け付けた文字列のサイズに適したメモリを動的に生成して、その部分に結合した文字列を埋め込むことを考える。

4.3 解答結果

図 12 s2212022-3.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 100
char* chomp(char* s){
    int i;
    for(i=0;s[i] != '\0';i++){
        if(s[i] == '\n'){
            s[i] = '\0';
        }
    }
}
char* mystrcat(char* s1, char* s2){
    int l1;
    int l2;
    int size = l1 + l2 +1;
    int i;
    char *p;
    l1 = strlen(s1);
    l2 = strlen(s2);
    p = (char*)malloc(size);
    for(i=0;i<sizeof(s1);i++){
        if(s1[i] != '\0'){
            p[i] = s1[i];
        }
    }
    for(i=0;i<sizeof(s2);i++){
        p[l1+i] = s2[i];
    }

    return p;
}
int main(int ac, char* av[]){
    char s1[SIZE];
    char s2[SIZE];

    printf("input␣␣first␣string:\n");
    chomp(fgets(s1,SIZE,stdin));
    printf("input␣␣second␣string:\n");
    chomp(fgets(s2,SIZE,stdin));

    printf("the␣result␣of␣mystrcat(%s
      ,%s)␣=␣%s\n",s1,s2,mystrcat(s1,
      s2));

}
```

図 12 にて確認できるように、配列を二つ受け取り、それぞれの配列の長さを l1,l2 とした。そうすると mystrcat() の返り値となる配列のサイズは,l1+l2+1 となる。これは二つの配列の文字数+末尾文字が新たな結合配列のサイズになるからである。あとは for 文で動的に生成したメモリに一つずつ文字を埋め込んでいき、末尾に配列の末尾文字を埋め込んだ。

4.4 確認

入力に対して期待される文字列が帰ってくるか確認すると

図 13 test9

```
input  first string:
abcdefg
input  second string:
hijklmn
the result of mystrcat(abcdefg,hijklmn
  ) = abcdefghijklmn
```

図 14 test10

```
input  first string:
orange
input  second string:
apple
the result of mystrcat(orange,apple) =
   orangeapple
```

上記二つの条件を満たしていることがわかる。

5 感想

設問 1 では, 配列とポインタの差異について十分に理解していなかったため, 解答得るまでに時間を要した。ポインタが読み込み専用領域をさす場合と, 読み書き可能領域にある場合では取り扱いが全く違うということ, 加えて配列が読み書き可能領域にマッピングされていることを理解したことで [1], 解答に至ることが出来た。格納された変数とオブジェクトがそれぞれメモリ上のどの領域に書き込まれたものなのか理解し, デバッグをしながらメモリ上の動きを見ながらテストしたことも, エラーの原因を理解する助けになった。表層的な理解では応用が利かないと改めて理解できたので, 今後はエラーが出た際表面的な解決手段を探るのではなく, 時間がかかっても成るべく原理原則の理解に時間をかけるようにしたい。

参考文献

[1] rita. プログラムがメモリをどのように使うか理解する, 2022. https://zenn.dev/rita0222/articles/e6ff75245d79b5.