プログラミング　第 4 回　レポート

202212022 田島瑞起

2023/07/01

1　はじめに

今回の課題では, 文字列を格納した一方向リストの作成及び, 表示 (設問 1), 一方向リスト内の文字列検索 (設問 2), 双方向リストを用いた文字列の削除 (設問 3) について latex を用いてレポート作成する。

2　文字列を格納した一方向リストの作成及び, 表示 (設問 1)

2.1　課題内容の説明

講義ノート (14) の図 11 を改造し, リストが文字列を保持するようにし, 標準入力で受け付けた文字列から改行コードを削除しリストに追加していく。その際, 何も入力されていない場合入力終了とし,Ctrl+d が押された場合はプログラムを終了する。

2.2　課題への取り組み方針

まず, リストが文字列を格納出来るよう構造体 Element のメンバを char*型に変更する。また getElements に関しても仮引数を char*型に変更し, 関数内部では変更した構造体のポインタを返すようにする。また構造体を作成する際に使用する, 標準入力から受け付けた文字列を, 動的なメモリに保存し, そのポインタを返す getstring() を作成する。getstring() の挙動は課題内容の説明にある要件を満たすよう条件分岐を用いて実装する。

2.3　解答結果

図 1　s2212022-1.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4
5   #define BUFSIZE 100
6
7   char* chomp(char* s){
8       int i;
9       for(i=0;s[i] != '\0';i++){
10          if(s[i] == '\n'){
11              s[i] = '\0';
12          }
13      }
14  }
15
16  struct Element{
17      char* val;
18      struct Element *next;
19  };
20
21  struct LIST{
22      struct Element *h;
23      struct Element *t;
24  };
25
26  struct Element *getElement(char* s){
27      struct Element *P;
28
29      P = (struct Element*)malloc(sizeof
        (struct Element));
30
31      if(P == NULL){
```

```
32              printf("Memory allocation
                  error\n");
33              exit(EXIT_FAILURE);
34          }
35          P->val = s;
36          P->next = NULL;
37          return P;
38      }
39
40      struct LIST *intitList(){
41          struct LIST *l;
42
43          l = (struct LIST*)malloc(sizeof(
                struct LIST));
44          if(l == NULL){
45              printf("MEmory allocation
                  error\n");
46              exit(EXIT_FAILURE);
47          }
48          l->h = getElement("");
49          l->t = l->h;
50          return l;
51      }
52
53      void appendElement(struct LIST *l,
          char* s){
54          struct Element *e;
55          int i;
56
57          e = getElement(s);
58          l->t->next = e;
59          l->t = e;
60      }
61
62      void printALLElements(struct LIST *l){
63          struct Element *e;
64
65          for(e=l->h->next; e != NULL; e =e
                ->next){
66              printf("val = %s\n", e->val);
67          }
68      }
69
70      char* getstring(){
71          char buf[BUFSIZE];
72          char* p;
73
74          if(fgets(buf,BUFSIZE,stdin) ==
                NULL){
75              exit(EXIT_SUCCESS);
76          }
77
78          if(buf[0] == '\n'){
79              return NULL;
80          }
81
82          chomp(buf);
83          p = (char*)malloc(sizeof(char)*(
                strlen(buf)+1));
84          strcpy(p,buf);
85          return p;
86      }
87
88      int main(int ac,char* av[]){
89          struct LIST *l1, *l2;
90          char* s;
91
92          l1 = intitList();
93          while(1){
94              printf("input a string (quit
                    whem Ctrl + D):");
95              s = getstring();
96
97              if (s == NULL){
98                  break;
99              }
100             appendElement(l1, s);
101         }
102
103         printf("l1\n");
104         printALLElements(l1);
105     }
```

元あるコードからの大きな変更点は getstring() で,buffa に存在する配列を動的なメモリに strcpy によりコピーした。また, 無入力の際に終了できるよう,buf のインデックス 0 が改行文字である場合, 標準入力の受付を終了するようにした。また,Ctrl+D で強制終了を実現できるように,

2.4 確認

入力した文字列がすべて表示される事, 空行入力によって受付終了される事,ctrl+D にて強制終了される事を確認する。

図 2 test1

```
1   input a string (quit whem Ctrl + D):
      apple
2   input a string (quit whem Ctrl + D):
      orange
3   input a string (quit whem Ctrl + D):
```

```
    banana
input a string (quit whem Ctrl + D):
    tomato
input a string (quit whem Ctrl + D):
    Ctrl+D detected!
```

図 3　test2

```
input a string (quit whem Ctrl + D):
    apple
input a string (quit whem Ctrl + D):
    orange
input a string (quit whem Ctrl + D):
    tomat
input a string (quit whem Ctrl + D):
    end
input a string (quit whem Ctrl + D):
l1
val = apple
val = orange
val = tomat
val = end
```

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　課題内容

設問 1 で作成したコードに改良を加えて, 検索機能を付与する。

3.2　課題への取り組み方針

int search(struct LIST *l, char* s) を定義し, 検索に引っかかれば 1 を, リストの末尾まで検索にかからなければ 0 を返すよう関数内部を記述する。

3.3　解答結果

図 4　s2212022-2.c

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

#define BUFSIZE 100

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

struct Element{
    char* val;
    struct Element *next;
};

struct LIST{
    struct Element *h;
    struct Element *t;
};

struct Element *getElement(char* s){
    struct Element *P;

    P = (struct Element*)malloc(sizeof
      (struct Element));

    if(P == NULL){
        printf("Memory␣allocation␣
          error\n");
        exit(EXIT_FAILURE);
    }
    P->val = s;
    P->next = NULL;
    return P;
}

struct LIST *intitList(){
    struct LIST *l;

    l = (struct LIST*)malloc(sizeof(
      struct LIST));
    if(l == NULL){
        printf("MEmory␣allocation␣
          error\n");
        exit(EXIT_FAILURE);
    }
    l->h = getElement("");
    l->t = l->h;
    return l;
}

```

```
53      void appendElement(struct LIST *l,
          char* s){
54          struct Element *e;
55          int i;
56
57          e = getElement(s);
58          l->t->next = e;
59          l->t = e;
60      }
61
62      void printALLElements(struct LIST *l){
63          struct Element *e;
64
65          for(e=l->h->next; e != NULL; e =e
              ->next){
66              printf("val = %s\n", e->val);
67          }
68      }
69
70      int search(struct LIST *l, char* s){
71          struct Element* e;
72          for(e=l->h->next; e != NULL; e =e
              ->next){
73              if(strcmp(e->val,s) == 0){
74                  return 1;
75              }
76          }
77          return 0;
78      }
79
80      char* getstring(){
81          char buf[BUFSIZE];
82          char* p;
83
84          if(fgets(buf,BUFSIZE,stdin) ==
              NULL){
85              exit(EXIT_SUCCESS);
86          }
87
88          if(buf[0] == '\n'){
89              return NULL;
90          }
91
92          chomp(buf);
93          p = (char*)malloc(sizeof(char)*(
              strlen(buf)+1));
94          strcpy(p,buf);
95          return p;
96      }
97
98      int main(int ac,char* av[]){
99          struct LIST *l1;
100         char* s1;
101         char* s2;
```

```
102
103         l1 = intitList();
104         while(1){
105             printf("input a string (quit
                  whem Ctrl + D): ");
106             s1 = getstring();
107
108             if (s1 == NULL){
109                 break;
110             }
111             appendElement(l1, s1);
112         }
113
114         printf("l1\n");
115         printALLElements(l1);
116
117         printf("input a search string (
              quit whem Ctrl + D): ");
118         s2 = getstring();
119         if(search(l1,s2) == 1){
120             printf("found\n");
121         }else{
122             printf("not found\n");
123         }
124     }
```

条件を満たすように実装した結果、図??となった。

3.3.1 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

```
1   input a string (quit whem Ctrl + D):
      apple
2   input a string (quit whem Ctrl + D):
      orange
3   input a string (quit whem Ctrl + D):
      tomato
4   input a string (quit whem Ctrl + D):
5   l1
6   val = apple
7   val = orange
8   val = tomato
9   input a search string (quit whem Ctrl
      + D): apple
10  found
```

図 6　test4

```
    input a string (quit whem Ctrl + D):
      apple
    input a string (quit whem Ctrl + D):
      orange
    input a string (quit whem Ctrl + D):
      tomato
    input a string (quit whem Ctrl + D):
      cucumber
    input a string (quit whem Ctrl + D):
    l1
    val = apple
    val = orange
    val = tomato
    val = cucumber
    input a search string (quit whem Ctrl
      + D): lettace
    not found
```

検索にかかる際, かからない際, 共に作動していることが確認できた。

4 mystrcat の実装 (設問 3)

4.1 課題内容の説明

作成したリストから入力した文字列を削除し, 新しいリストを作成, および表示できるよう設問 2 で作成したコードを改良する。

4.2 課題への取り組み方針

削除機能を持たせるため, まずは構造体 Element のメンバに一つ手前の構造体ポインタを格納する before を追加する。そして双方向リストを作成するために appendElements を next の登録だけではなく,before の登録も同時に出来るよう変更する。また, 実際に削除を実行する deleteElements の実装をする。

4.3 解答結果

図 7 s2212022-3.c

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

#define BUFSIZE 100

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

struct Element{
    char* val;
    struct Element *next;
    struct Element *before;
};

struct LIST{
    struct Element *h;
    struct Element *t;
};

struct Element *getElement(char* s){
    struct Element *P;

    P = (struct Element*)malloc(sizeof
      (struct Element));

    if(P == NULL){
        printf("Memory␣allocation␣
          error\n");
        exit(EXIT_FAILURE);
    }
    P->val = s;
    P->next = NULL;
    P->before = NULL;
    return P;
}

struct LIST *intitList(){
    struct LIST *l;

    l = (struct LIST*)malloc(sizeof(
      struct LIST));
    if(l == NULL){
        printf("MEmory␣allocation␣
          error\n");
        exit(EXIT_FAILURE);
    }
    l->h = getElement("");
    l->t = l->h;
    return l;
}
```

```
54
55      void appendElement(struct LIST *l,
          char* s){
56          struct Element *e;
57          int i;
58
59          e = getElement(s);
60          e->before = l->t;
61          l->t->next = e;
62          l->t = e;
63      }
64
65      void printALLElements(struct LIST *l){
66          struct Element *e;
67
68          for(e=l->h->next; e != NULL; e =e
              ->next){
69              printf("val = %s\n", e->val);
70          }
71      }
72
73      void deleteElements(struct LIST *l,
          char* s){
74          struct Element* e;
75          for(e=l->h->next; e != NULL; e =e
              ->next){
76              if((strcmp(e->val,s) == 0) &&
                  e->next != NULL){
77                  e->before->next = e->next;
78                  e->next->before = e->
                      before;
79              }
80
81              if((strcmp(e->val,s) == 0) &&
                  e->next == NULL){
82                  l->t = e->before;
83                  l->t->next = NULL;
84
85              }
86
87
88          }
89      }
90
91      char* getstring(){
92          char buf[BUFSIZE];
93          char* p;
94
95          if(fgets(buf,BUFSIZE,stdin) ==
              NULL){
96              exit(EXIT_SUCCESS);
97          }
98
99          if(buf[0] == '\n'){
100             return NULL;
101         }
102
103         chomp(buf);
104         p = (char*)malloc(sizeof(char)*(
              strlen(buf)+1));
105         strcpy(p,buf);
106         return p;
107     }
108
109     int main(int ac,char* av[]){
110         struct LIST *l1;
111         char* s1;
112         char* s2;
113
114         l1 = intitList();
115         while(1){
116             printf("input a string (quit
                  whem Ctrl + D): ");
117             s1 = getstring();
118
119             if (s1 == NULL){
120                 break;
121             }
122             appendElement(l1, s1);
123         }
124
125         printf("l1\n");
126         printALLElements(l1);
127
128         printf("input a search string (
              quit whem Ctrl + D): ");
129         s2 = getstring();
130         deleteElements(l1,s2);
131
132         printf("l1\n");
133         printALLElements(l1);
134
135     }
```

課題への取り組み方針にて示した要件を満たすようコードを改良した結果，図??となる。

4.4 確認

無入力時の挙動，入力時の挙動について，それぞれ検索にかかる場合と検索にかからない場合を確認すると

図 8 test9

```
1   input a string (quit whem Ctrl + D):
2   l1
```

```
3     input a search string (quit whem Ctrl
        + D): apple
4     l1
```

図 9　test10

```
1     input a string (quit whem Ctrl + D):
2     l1
3     input a search string (quit whem Ctrl
        + D):
4     l1
```

図 10　test10

```
1     l1
2     val = orange
3     val = apple
4     val = pudding
5     input a search string (quit whem Ctrl
        + D): banana
6     l1
7     val = orange
8     val = apple
9     val = pudding
```

図 11　test10

```
1     l1
2     val = orange
3     val = apple
4     val = banana
5     val = suica
6     input a search string (quit whem Ctrl
        + D): suica
7     l1
8     val = orange
9     val = apple
10    val = banana
```

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,getstring の分岐が個人的な難所であった。fget が EOF に達した際, 分岐条件で最初に==NULL ではなく==EOF としてしまったため、分岐が正しく行われず、バグが発生してしまった。そこで stdio.h にて EOF の定義を確認したところ,int 型の-1 であるということを知った。fget の返り値は char*型の為, 型が一致せずエラーが発生していることから, 今回は==NULL で対応した。設問 2 ではリストを巡回すれば良いとすぐ気が付いたので時間はかからなかった。設問 3 では、リストを巡回して検索にマッチすれば削除を行い, 削除には next のみでは, 巡回して手前の対象に戻ることが出来ないと考え, 構造体に before を加えればよいと数分考えて思いつくことが出来た。今回は文献を参照せずに自力でコードを作成することが出来た。web 上で簡単に検索できる時代ではあるが, 知識を定着して直ぐに出せる事を目標に様々な学習を進めたい。