プログラミング言語実験・C言語 第2回課題レポート

1510151 柳 裕太

2017年4月20日

1 課題3

1.1 ソースコード

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <ctype.h>
5
6  #define TRUE 0
7  #define FALSE 1
8
9  typedef char data_type; /* データの型 */
10 typedef struct node_tag {
11     data_type data; /* データ */
12     struct node_tag *next; /* 後続ノードへのポインタ */
13 } node_type; /* ノードの型 */
14
15 void error(char *msg) {
16     fprintf(stderr, "error: %s\n", msg);
17     exit(1);
18 }
19
20 void initialize(node_type **pp) /* スタックの初期化 */
21 {
22     *pp = NULL; /* スタックは空（先頭ノードなし） */
23 }
24
25 node_type *new_node(data_type x, node_type *p)
26 {
27     node_type *temp;
```

```
28      temp = (node_type *)malloc(sizeof(node_type));
29      /* メモリの割当て */
30      if (temp == NULL) return NULL; /* メモリ割当て失敗 */
31      else { /* ノードの各メンバーの値の設定 */
32      temp->data = x;
33      temp->next = p;
34      return temp;
35      }
36  }
37
38  int is_empty(node_type *p) /* 空スタックのとき真、 */
39  { /* そうでないならば偽を返す */
40      if (p == NULL) return TRUE; /* 空スタックのとき */
41      else return FALSE; /* 空スタックでないとき */
42  }
43
44  data_type top(node_type *p)
45  {
46      if (p == NULL) /* 空スタックのとき */
47        return '\0'; /* ナル文字を返す */
48      else /* 空スタックでないとき */
49        return p->data; /* スタックの先頭のデータを返す */
50  }
51
52  int push(node_type **pp, data_type x)
53  {
54      node_type *temp;
55      temp = new_node(x, *pp); /* 関数の呼出し new_node */
56      if (temp == NULL) return 1;
57      *pp = temp;
58      return 0;
59  }
60
61  int pop(node_type **pp) {
62      node_type *temp;
63      if (*pp != NULL) {
64          temp = (*pp)->next;
65          free(*pp); /* メモリの解放 */
66          *pp = temp;
67          return 0;
68      }
```

```
69      else return 1;
70  }
71
72
73  // 引用元: http://home.a00.itscom.net/hatada/c-tips/rpn/rpn02.html
74  // 演算子の優先順位を返す
75  int rank(char *op) {
76      if (*op == '*' || *op == '/' ) return 3;
77      if (*op == '+' || *op == '-') return 2;
78      if (*op == '(') return 4;
79      if (*op == ')') return 1;
80      if (*op == '=') return 0;
81      return 5;
82  }
83
84
85  void convert2(char *token[], int length) {
86      node_type *stack;
87      initialize(&stack);
88      int n;
89      data_type head_stack[1];
90      for (n = 0; n < length; n++) {
91          head_stack[0] = top(stack);
92          while(is_empty(stack) != TRUE &&
93                '(' != head_stack[0] &&
94                rank(token[n]) <= rank(head_stack)){
95              printf("%c ", top(stack));
96              pop(&stack);
97              head_stack[0] = top(stack);
98          }
99          if(*token[n] != ')'){
100             push(&stack, *token[n]);
101         }else{
102             pop(&stack);
103         }
104     }
105     while(is_empty(stack) != TRUE){
106         printf("%c ", top(stack));
107         pop(&stack);
108     }
109     printf("\n");
```

```
110 }
111
112
113 // 引用終了
114
115 int main(void){
116     printf("B=2,␣C=3,␣D=4,␣E=5,␣Fとして演算=6\n");
117     printf("----------\n");
118     data_type *frml1[] = {
119         "A", "=", "(", "2", "-", "3", ")",
120         "/", "4", "+", "5", "*", "6"
121     };
122     convert2(frml1, sizeof(frml1)/sizeof(char*));
123     printf("A=%f\n", (2.0-3.0)/4.0+5.0*6.0);
124     printf("----------\n");
125     data_type *frml2[] = {
126         "A", "=", "(", "2", "-", "3", "/",
127         "4", "+", "5", ")", "*", "6"
128     };
129     convert2(frml2, sizeof(frml2)/sizeof(char*));
130     printf("A=%f\n", (2.0-3.0/4.0+5.0)*6.0);
131     printf("----------\n");
132     data_type *frml3[] = {
133         "A", "=", "2", "-", "3", "/", "(",
134         "4", "+", "5", "*", "6" ")",
135     };
136     convert2(frml3, sizeof(frml3)/sizeof(char*));
137     printf("A=%f\n", 2.0-3.0/(4.0+5.0*6.0));
138 }
```

1.2 実行結果

```
1 % ./ex3_rev_pol_not.c
2   B=2, C=3, D=4, E=5, Fとして演算=6
3   ----------
4   A 2 3 - 4 / 5 6 * + =
5   A=29.750000
6   ----------
7   A 2 3 4 / - 5 + 6 * =
8   A=37.500000
```

```
  ----------------------
  A 2 3 4 5 6 * + ( / - =
  A=1.911765
```

2 課題 4

2.1 ソースコード

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define SUCCESS 0
#define FAILURE 0

typedef double data_type;
typedef struct node_tag {
    data_type data;
    struct node_tag *left;
    struct node_tag *right;
} node_type; /* ノードの型 */


void initialize(node_type **pp){
    *pp = NULL;
}


node_type *new_node(data_type x) {
    node_type *temp;
    temp = (node_type *)malloc(sizeof(node_type));
    if (temp == NULL) return NULL;
    else {
        temp->data = x;
        temp->left = NULL;
        temp->right = NULL;
        return temp;
    }
}

```

```
33
34 int insert(node_type **pp, data_type x) {
35     node_type *temp;
36     if (*pp == NULL) {
37         temp = new_node(x);
38         if (temp == NULL) return FAILURE;
39         *pp = temp;
40         return SUCCESS;
41     }else {
42         /* 絶対値比較のためfabs使用() */
43         if (fabs(x) <= fabs((*pp)->data))
44             insert(&((*pp)->left), x);
45         else if (fabs(x) > fabs((*pp)->data))
46             insert(&((*pp)->right), x);
47     }
48     return SUCCESS;
49 }
50
51
52 double tree_size(node_type *p) {
53     double sump=0, left=0, right=0;
54     /* 子ノードが存在すれば、再帰呼び出しで子ノードまでの合計を呼び出す */
55     if(p->left != NULL){
56         left=tree_size(p->left);
57     }
58     printf("%g\n", p->data);
59     if(p->right != NULL){
60         right=tree_size(p->right);
61     }
62     /* 合計は、左右の子ノードまでの合計自分の値 +*/
63     sump = left + right + p->data;
64     return sump;
65 }
66
67
68
69 int main(void){
70     node_type *root;
71     initialize(&root);
72     double num, sum_sorted = 0;
73     while(1){
```

```
74          scanf("%lf", &num);
75          if(num == 0){break;}
76          insert(&root, num);
77      }
78
79      printf("------------------sorted by fabs()-----------------\n");
80      sum_sorted = tree_size(root);
81      printf("\n");
82      printf("sorted sum: %g\n", sum_sorted);
83
84      // list_free(root);
85      return 0;
86  }
```

2.2 実行結果

```
1  % ./ex4_bin_ser_tree.out < ./numbers.txt
2    ------------------sorted by fabs()-----------------
3    -0.002
4    -0.01
5    0.012
6    0.3
7    3.1
8    -6.4
9    7
10   8
11   10
12   23
13   -30
14   46
15   -70
16   -100
17   360
18   -500
19   -1700
20   -3000
21   5000
22   -1e+16
23   1e+16
24   sorted sum: 52
```