

# プログラミング言語実験・C 言語 第2回課題レポート

I 類 メディア情報学

氏名：LEORA DAVID

学籍番号：2210745

2024 年 04 月 24 日

## 課題3（逆ポーランド記法）

被演算数、演算子、および括弧のトークンで構成された中置記法の算術式を読み込み、スタックを用いて逆ポーランド記法の式を出力する C プログラムを作成した。それから、次の（1）～（3）の中置記法の算術式に対する実行結果を求めた。

(1)  $A = (B - C / D + E) * F$

(2)  $A = B - (C / D + E) * F$

(3)  $A = B - C / (D + E * F)$

また、プログラム中で使用する演算子は、二項演算しの加算「+」、減算「-」、乗算「\*」、除算「/」、および代入「=」だけに限定することにした。最後に、単項演算子の加減算を追加した。

ソースコード 1 reversePolishNotation.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define SIZE 100
6 #define TRUE 1
7 #define FALSE 0
8 #define SUCCESS 1
9 #define FAILURE 0
10
11 typedef char data_type;
12 typedef struct node_tag {
13     data_type data;
14     struct node_tag *next;
15 } node_type;
16
17 void initialize(node_type **pp){
18     *pp = NULL;
19 }
20
21 int is_empty(node_type *p){
22     return (p == NULL) ? TRUE : FALSE;
23 }
24
```

```

25 data_type top(node_type *p){
26     if (is_empty(p)){
27         printfスタックは空です ("\n");
28         return '\0';
29     }
30     return p->data;
31 }
32
33 node_type *new_node(data_type x, node_type *next) {
34     node_type *temp = (node_type *)malloc(sizeof(node_type));
35     if (temp != NULL) {
36         temp->data = x;
37         temp->next = next;
38     }
39     return temp;
40 }
41
42 int push(data_type x, node_type **pp){
43     node_type *temp = new_node(x, *pp);
44     if (temp == NULL) return FAILURE;
45     *pp = temp;
46     return SUCCESS;
47 }
48
49 int pop(node_type **pp)
50 {
51     node_type *temp;
52
53     if (!is_empty(*pp)){
54         temp = (*pp)->next;
55         free(*pp);
56         *pp = temp;
57         return SUCCESS;
58     }
59     return FAILURE;
60 }
61
62 int checkPriority(char a){
63     int priority;
64     switch (a) {
65         case '=':
66             return 0;
67         case ')' :
68             return 1;
69         case '+' :
70             return 2;

```

```

71     case '-' :
72         return 2;
73     case '*' :
74         return 3;
75     case '/' :
76         return 3;
77     case '(' :
78         return 4;
79     default:
80         return 5;
81     }
82 }
83
84 int main(){
85     node_type *stack;
86     initialize(&stack);
87     char input[SIZE];
88     int track = 0;
89     scanf("%s", input);
90
91     while (track < strlen(input) && input[track] != '\0'){
92         while(!is_empty(stack) && top(stack) != '(' && checkPriority(input[track]) <=
93             checkPriority(top(stack))) {
94             if(top(stack) != '\0')
95                 printf("%c", top(stack));
96             pop(&stack);
97         }
98         if (input[track] != ')') {
99             push(input[track], &stack);
100         } else {
101             while (!is_empty(stack) && top(stack) != '('){
102                 printf("%c", top(stack));
103                 pop(&stack);
104             }
105             if (!is_empty(stack) && top(stack) == '(') {
106                 pop(&stack);
107             }
108         }
109         track++;
110     }
111
112     while(!is_empty(stack)){
113         if(top(stack) != '\0') {
114             printf("%c", top(stack));
115         }
116         pop(&stack);

```

```
116     }
117     printf("\n");
118     return 0;
119 }
```

---

## 課題 4（二分探索木）

入力として整数（int 型）のデータが与えられ、これらのデータには同じ整数が重複して出現してもよいものとする。このとき、ポインタによる二分探索木を実現して、整数データを二分探索木に挿入した後、この二分探索木に挿入されたデータを取り出して、データを照準（厳密には非減少順）に出力する C のプログラムを作成した。ただし、同じ値の整数データが複数存在する場合には、入力時と同じ個数分出力することにした。

ここで、ポインタによる二分探索木の実現に際して、各ノードは、整数データを保持する int 型変数 data と、その整数が出現した頻度（度数）を保持する int 型変数 frequency をもつものとする。最後に、データの削除も行えるように改良した。

---

### ソースコード 2 binarySearchTree.c

---

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define TRUE 1
5 #define FALSE 0
6 #define SUCCESS 1
7 #define FAILURE 0
8
9 typedef int data_type;
10 typedef int freq_type;
11 typedef struct node_tag {
12     data_type data;
13     freq_type freq;
14     struct node_tag *left;
15     struct node_tag *right;
16 } node_type;
17
18 void initialize(node_type **pp) {
19     *pp = NULL;
20 }
21
22 int is_member(data_type x, node_type *p) {
23     if (p == NULL) return FALSE;
24     else {
25         if(x == p->data) return TRUE;
26         else {
```

```

27         if (x < p->data) return is_member(x, p->left);
28         else return is_member(x, p->right);
29     }
30 }
31 }
32
33 data_type min(node_type *p) {
34     while (p->left != NULL) {
35         p = p->left;
36     }
37     return p->data;
38 }
39
40 node_type *new_node(data_type x) {
41     node_type *temp = (node_type *)malloc(sizeof(node_type));
42     if (temp == NULL) return NULL;
43     else {
44         temp->data = x;
45         temp->freq = 1;
46         temp->left = NULL;
47         temp->right = NULL;
48         return temp;
49     }
50 }
51
52 int insert (data_type x, node_type **pp) {
53     if (*pp == NULL) {
54         node_type *temp = new_node(x);
55         if (temp == NULL) return FAILURE;
56         *pp = temp;
57         return SUCCESS;
58     } else {
59         if (x < (*pp)->data) return insert (x, &((*pp)->left));
60         else if (x > (*pp)->data) return insert(x, &((*pp)->right));
61         else {
62             (*pp)->freq++;
63             return SUCCESS;
64         }
65     }
66 }
67
68 /** 未実装：データの削除（予定）
69 int delete (data_type x, node_type **pp) {
70
71 }
72 **/

```

```

73
74 void print_in_order(node_type *p) {
75     if(p != NULL) {
76         print_in_order(p->left);
77         for (int i = 0; i < p->freq; i++){
78             printf("%d ", p->data);
79         }
80         print_in_order(p->right);
81     }
82 }
83
84 // プログラム終了時に、動的に割り当てたメモリを開放する処理
85 void free_tree(node_type *p) {
86     if (p != NULL) {
87         free_tree(p->left);
88         free_tree(p->right);
89         free(p);
90     }
91 }
92
93 int main(){
94     node_type *root;
95     initialize(&root);
96     int choice, x;
97
98     while(TRUE) {
99         printf("\n数値を入力してください（プログラム終了：-1）：n");
100         scanf("%d", &x);
101         if (x == -1) break;
102         if (insert(x, &root) == SUCCESS) {
103             printf挿入が成功しました ("\n");
104             printf木の中身：("");
105             print_in_order(root);
106             printf("\n");
107         } else {
108             printf挿入が失敗しました ("\n");
109         }
110     }
111     free_tree(root);
112     printfプログラムを終了します。("\n");
113     return 0;
114 }

```

---

## 参考文献

- [1] Name, URL