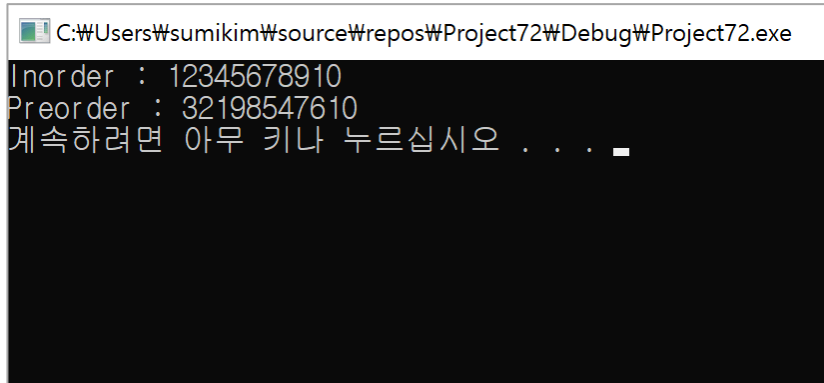


오픈랩 과제 #6 : Binary Search Tree

2020. 11. 24 (화) / 20181202 김수미

1. 코드 실행 결과



```
C:\Users\sumikim\source\repos\Project72\Debug\Project72.exe
Inorder : 12345678910
Preorder : 32198547610
계속하려면 아무 키나 누르십시오 . . . .
```

2. 코드 및 알고리즘 설명

Binary Search Tree(이진 탐색 트리)는 데이터의 탐색, 삽입, 삭제, 연산에 있어서 매우 성능이 좋은 자료구조로, 키 값 혹은 순위(rank)에 따라 작업을 수행한다는 특징이 있다.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define MAX_TERMS 100
4
5  typedef struct treeNode* treePtr;
6  typedef struct treeNode {
7      treePtr leftChild;
8      int data;
9      treePtr rightChild;
10 } treeNode;
11
12 void inorder(treePtr);
13 void preorder(treePtr);
14 void insert_node(treePtr*, int);
15 treePtr modifiedSearch(treePtr, int);
```

이번 코드는 상당히 간단하다. 우선 코드 작성에 필요한 헤더파일들을 불러온 뒤, 상수 MAX_TERMS 값을 100으로 설정해 준다. 다음 트리의 각 노드값이 될 treeNode 구조체를 정의해 준다. 각 노드는 leftChild 값과 rightChild 값, data값을 가진다.

해당 코드에서는 4개의 함수가 사용될 것이다. 4개의 함수를 먼저 선언해준 뒤 차례대로 정의해주자.

```

17 void inorder(treePtr ptr) {
18     if (ptr) {
19         inorder(ptr->leftChild);
20         printf("%d", ptr->data);
21         inorder(ptr->rightChild);
22     }
23 }
24
25 void preorder(treePtr ptr) {
26     if (ptr) {
27         printf("%d", ptr->data);
28         preorder(ptr->leftChild);
29         preorder(ptr->rightChild);
30     }
31 }

```

먼저 inorder 함수의 정의이다. inorder 함수는 매개변수 tree를 inorder traversal 하는 함수로, 더이상 값이 없을 때 까지 왼쪽 방향으로 타고 내려간 다음, 다시 거슬러 올라가며 차례대로 값을 반환하고, 거슬러 올라가는 과정에서 오른쪽 자식 노드가 있는 경우 다시 왼쪽 방향으로 이동하여 내려간 후 다음의 방식을 반복하는 식으로 값을 반환한다. 결과적으로 데이터 값을 오름차순으로 traversal한다.

다음으로 preoder 함수의 정의이다. preorder 함수는 매개변수 tree를 preorder traversal하는 함수로, 더이상 값이 없을 때 까지 왼쪽방향으로 타고 내려가는 것은 inorder traversal 과 동일하나 타고 내려감과 동시에 값을 반환한다. 더 이상 값이 없는 경우 거슬러 올라가면서(값을 반환하지는 않는다) 오른쪽 자식 노드가 있는지 확인한다. 있는 경우 해당 노드의 왼쪽방향으로 타고 내려가며 같은 과정을 반복한다.

```

33 void insert_node(treePtr *node, int num) {
34     treePtr ptr, temp = modifiedSearch(*node, num);
35     if (temp || !(*node)) {
36         ptr = (treePtr)malloc(sizeof(*ptr));
37         ptr->data = num;
38         ptr->leftChild = ptr->rightChild = NULL;
39         if (*node) {
40             if (num < temp->data) temp->leftChild = ptr;
41             else temp->rightChild = ptr;
42         }
43         else *node = ptr;
44     }
45 }

```

다음은 insert_node 함수이다. 새로운 data값이 될 int num을 삽입하기 위해서는 먼저 기존의 원소들이 가지고 있는 data값과 새로운 값이 다른지를 확인해야 하므로 탐색이 수행된다. 탐색 결과 이상이 없으면 탐색이 종료된 지점에 새로운 data값을 삽입한다. 이 때의 규칙은 여전히 root를 제외한 node들은 자신보다 왼쪽의 노드보다 값이 크고, 오른쪽 노드보다는 값이 작다는 것이다.

```

47 treePtr modifiedSearch(treePtr node, int k) {
48     treePtr temp = node;
49     while (node) {
50         temp = node;
51         if (k < node->data) //삽입값이 data보다 작은경우
52             node = node->leftChild; //왼쪽 tree 탐색
53         else if (k > node->data) //삽입값이 data보다 큰 경우
54             node = node->rightChild; // 오른쪽 tree 탐색
55         else return NULL;
56     }
57     return temp;
58 }

```

위 코드는 modifiedSearch 함수로, insert_node 함수에서 호출되는 함수이다.

이 함수는 binary search tree 의 *node에서 data값 중에서 새롭게 삽입되는 data 값과 같은 것이 있는지를 탐색하는데, 만약 트리가 공백 이거나 새롭게 삽입되는 data와 같은 값의 data가 존재하는 경우 NULL 값을 반환하고, 그렇지 않으면 가장 마지막으로 검사한 노드에 대한 포인터를 반환한다. 그럼으로 인해 insert_node 함수에서 가장 마지막 노드의 자식으로 새로운 data값을 삽입할 수 있게 해준다.

```

60 void main(void) {
61     int i, n, A[MAX_TERMS];
62     treePtr tree = NULL;
63
64     // 파일 오픈한다.
65     FILE* fp;
66     fp = fopen("input.txt", "r");
67     if (fp == NULL) {
68         fprintf(stderr, "file open error \n");
69         exit(1);
70     }
71
72     // 파일을 읽어 정수의 개수는 n에, 삽입되어야 할 정수들은 A에 저장
73     fscanf(fp, "%d", &n);
74     for (i = 0; !feof(fp); i++)
75         fscanf(fp, "%d", &A[i]);
76     fclose(fp);
77
78     // 순서대로 binary search tree에 삽입
79     for (i = 0; i < n; i++)
80         insert_node(&tree, A[i]);
81
82     // 구성된 binary search tree를 inorder와 preorder로 출력
83     printf("Inorder : ");
84     inorder(tree); printf("\n");
85
86     printf("Preorder : ");
87     preorder(tree); printf("\n");
88
89     system("pause");
90 }

```

메인 함수이다. 메인함수에서는 txt파일을 오픈해 파일 안에 저장되어있는 값으로 tree를 구성하게 된다. 파일을 오픈한 후 내부에 저장되어 있는 정수값들을 배열에 따로 저장한 다음, 배열과 insert_node 함수 를 이용해 차례대로 binary search tree 에 저장한 다음, 완성된 tree 를 inorder 함수와 preorder 함수를 이용해 출력하여 결과를 확인할 수 있다.