이진 검색 트리의 구축과 활용

1. 트리 구성을 위한 기본 노드의 정의

   Self-referential Structures(자기참조구조체)

   이진 트리의 경우 자신의 노드와 같은 구조로의 포인터를 2개 가지고 있음

2. 이진 검색 트리의 구축

   - 공백 트리인 경우

   - 이미 노드가 구축되어 있는 경우에 삽입

3. 구축된 이진검색트리에서 검색 알고리즘

4. 트리의 운행/순회(Tree Traversal)

5. 이진검색트리를 활용한 응용 예 프로그래밍 실습

이진 검색트리의 정의

1. 정의 : 이진검색 트리는 다음의 성질을 만족하는 이진 트리 이다.

   (1) 모든 원소는 키 값을 갖는다.

   (2) 왼쪽 서브 트리의 키들은 그의 루트의 키보다 작아야 한다.

   (3) 오른쪽 서브 트리의 키들은 그 루트의 키보다 커야 한다.

   (4) 왼쪽과 오른쪽 서브 트리도 또한 이진검색 트리이다.(재귀적 성질)

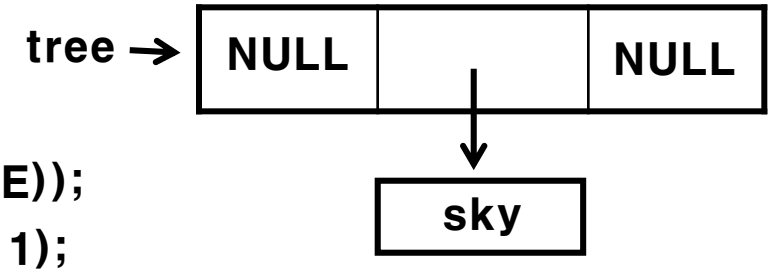
2. 이진트리의 노드 구조 정의를 위한 선언

```
typedef  struct  node {
        struct node *left;
        char  *word;
        struct node *right;
} NODETYPE;
```

이진 검색 트리의 구축(1)

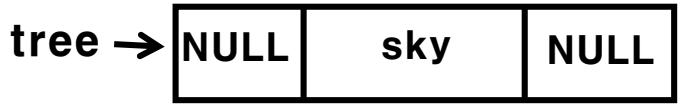
sky, blue, tiger, happy, child, yellow, aha…의 순으로 문자열 데이터가 입력된다고 가정하며 구축 과정을 이해해 보자.

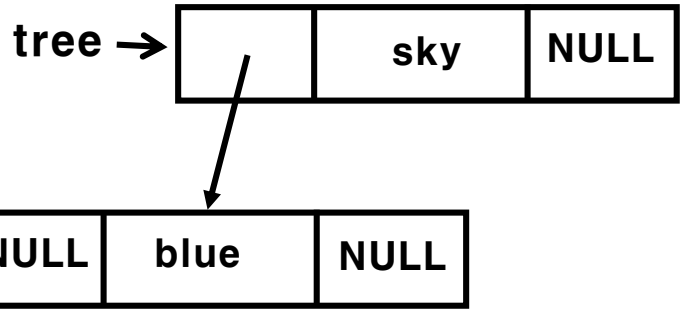
```
// 트리가 비었을 때, tree는 root를 가리키는 포인터로 NULL로 초기화
// gets(wbuf)를 통하여 입력 문자열을 가지고 있다고 가정
if (!tree)
{
        tree = (NODETYPE *)malloc(sizeof(NODETYPE));
        tree -> word = (char *) malloc(strlen(wbuf) + 1);
        strcpy(tree->word, wbuf);
        tree->left = tree->right = NULL;
}
```

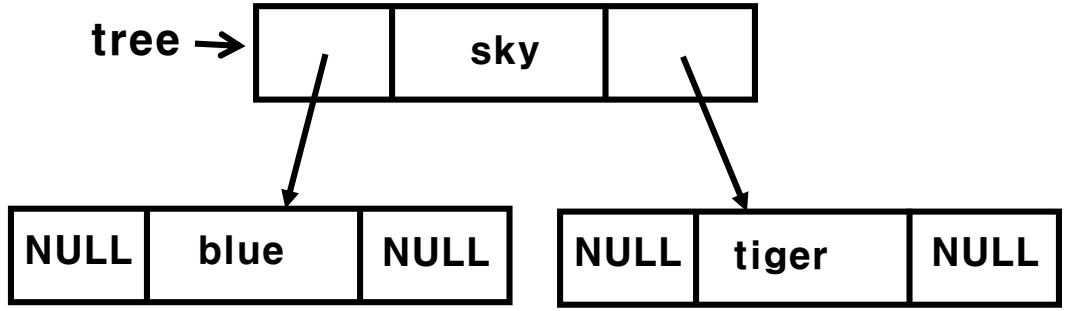


이진 검색 트리의 구축(2)

sky, blue, tiger, happy, child, yellow, aha…의 순으로 문자열 데이터가 입력된다고 가정하며 구축 과정을 이해해 보자.

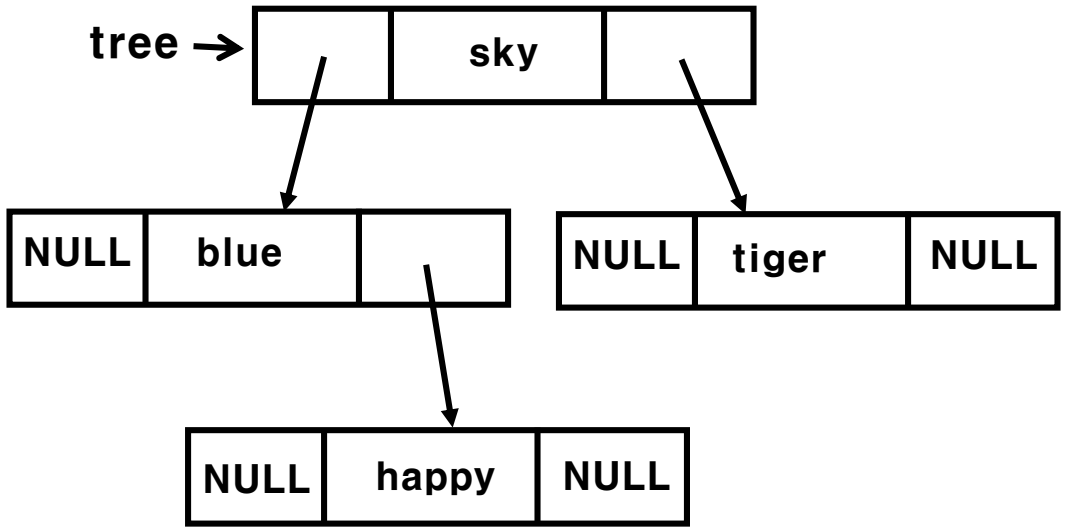






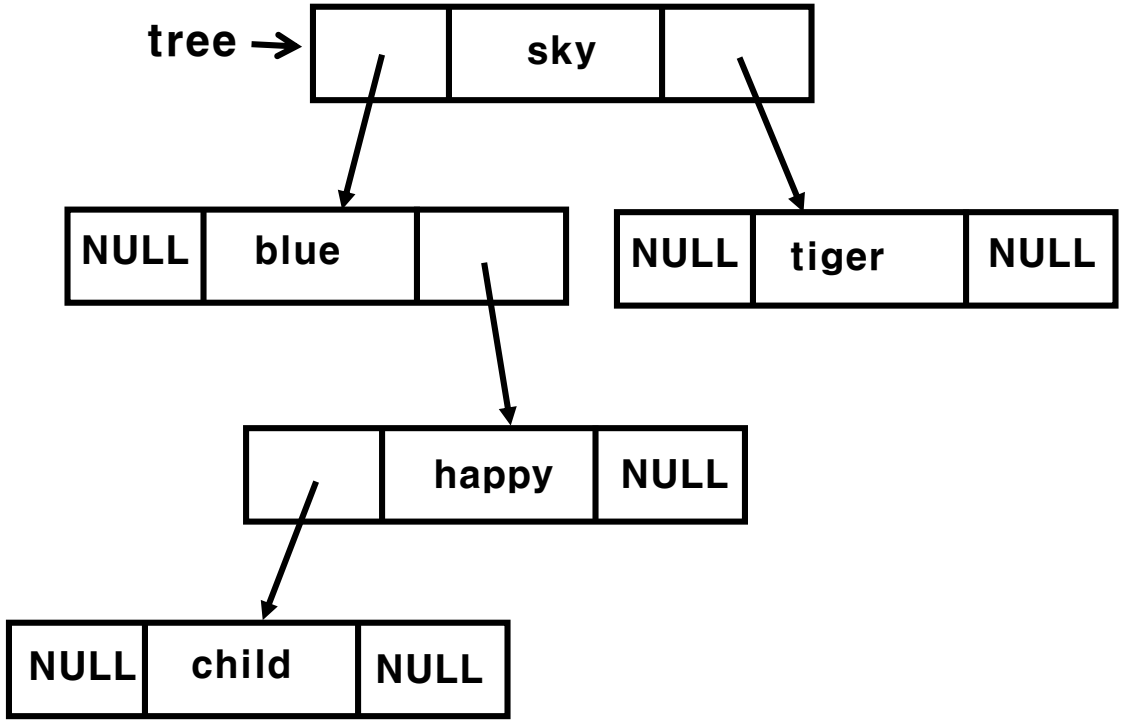
이진 검색 트리의 구축(2)

sky, blue, tiger, happy, child, yellow, aha…의 순으로 문자열 데이터가 입력된다고 가정하며 구축 과정을 이해해 보자.



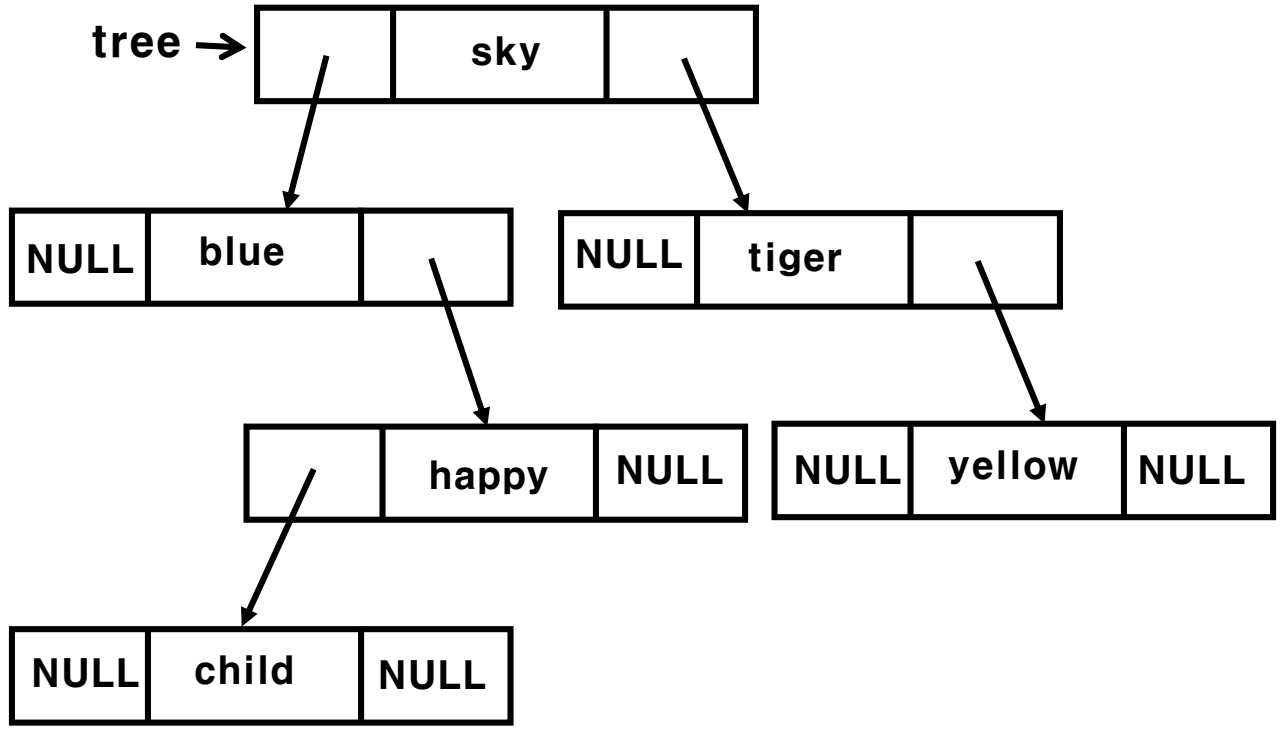
이진 검색 트리의 구축(2)

sky, blue, tiger, happy, child, yellow, aha…의 순으로 문자열 데이터가 입력된다고 가정하며 구축 과정을 이해해 보자.



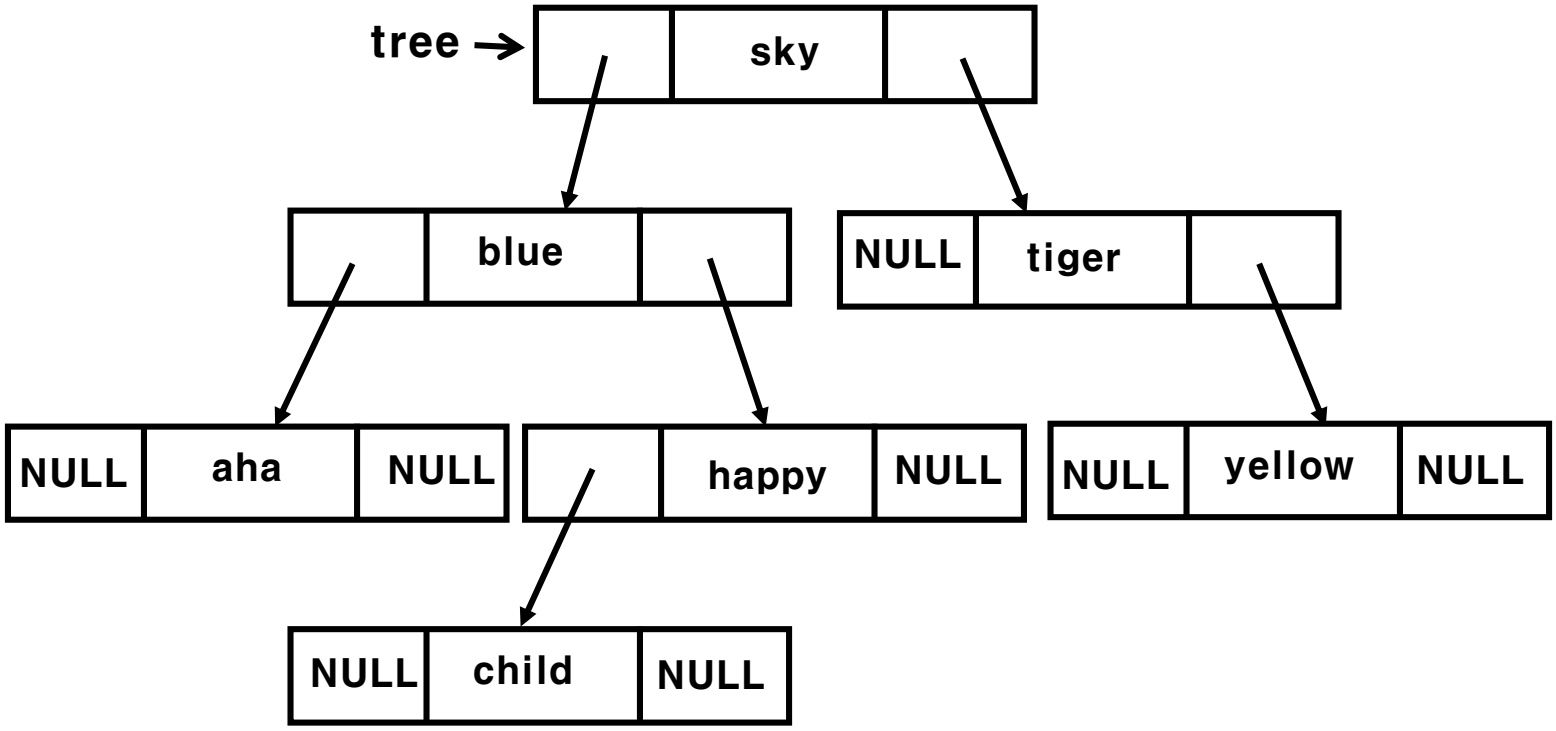
이진 검색 트리의 구축(2)

sky, blue, tiger, happy, child, yellow, aha…의 순으로 문자열 데이터가 입력된다고 가정하며 구축 과정을 이해해 보자.



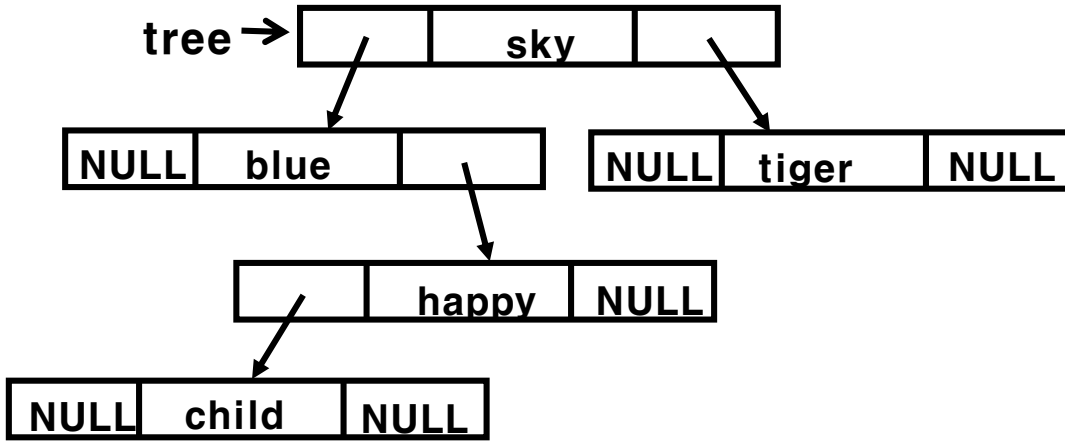
이진 검색 트리의 구축(2)

sky, blue, tiger, happy, child, yellow, aha…의 순으로 문자열 데이터가 입력된다고 가정하며 구축 과정을 이해해 보자.



이진 검색 트리의 구축(2)

```
int insert_node(NODETYPE *root, char *word)
{
        NODETYPE *tptr = root, *before;
        int cmp;
        while (tptr)  {
           cmp = strcmp(word, tptr->word);
           if (cmp < 0) {
              before = tptr;
              tptr = tptr -> left;
           }
           else if (cmp > 0) {
              before = tptr;
              tptr = tptr -> right;
           }
           else       // found
              return 0;
        }
```




```
        tptr = (NODETYPE *)malloc(sizeof(NODETYPE));
        tptr -> word = (char *) malloc(strlen(word) + 1);
        strcpy(tptr->word, word);
        tptr->left = tptr->right = NULL;
        if (cmp < 0) before -> left = tptr;
        else  before -> right = tptr;
        return 1;

} //end of insert_node
```

이진 검색 트리의 검색

검색은 루트로부터 시작한다.

이때 루트가 NULL이면 이 트리는 원소를 갖지 않으므로 검색은 실패한다.

그렇지 않으면 루트의 키 값과 비교하면 세 가지 경우가 된다.

(1) key가 루트의 key값과 같다면 검색 성공

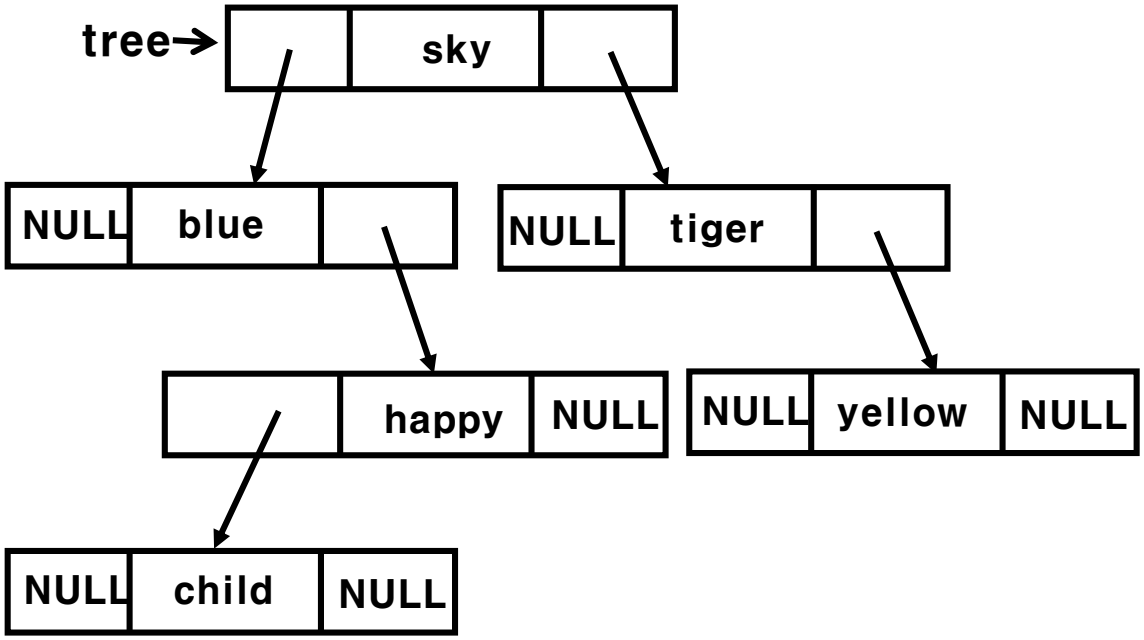
(2) key가 루트의 key보다 작다면 왼쪽 서브 트리를 검색한다.

(3) key가 루트의 key보다 크다면 오른쪽 서브트리를 검색한다.

(4) 서브 트리들은 재귀적으로 검색이 이루어진다.

이진 검색 트리의 검색 함수

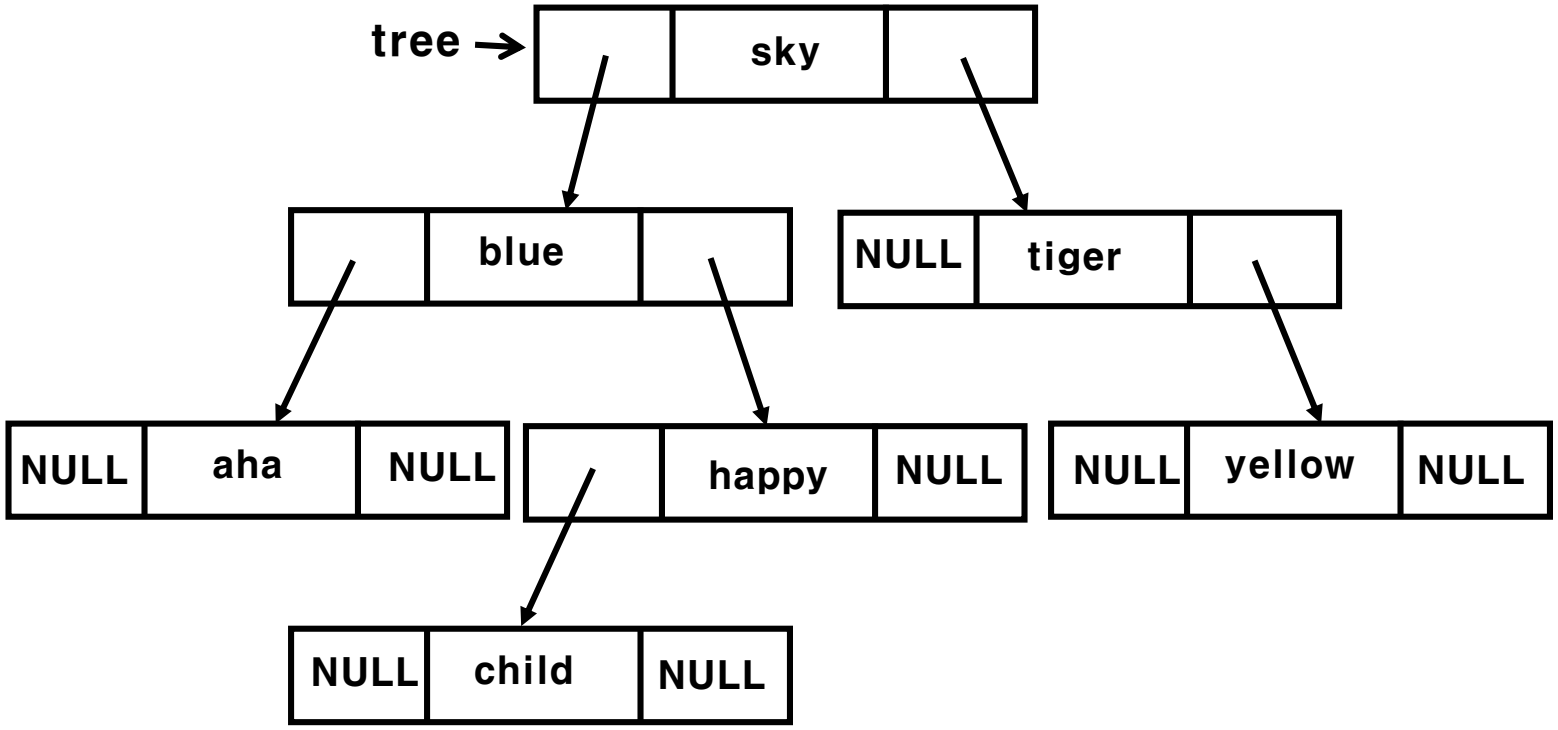
```
NODETYPE *search(NODETYPE *root, char *key)
 {
     NODETYPE *tptr = root;
     int cmp;
     while (tptr) {
        cmp = strcmp(key, tptr->word);
        if (cmp <0)
            tptr = tptr -> left;
        else if (cmp > 0)
            tptr = tptr -> right;
        else       // found
          return tptr;
     }
     return NULL;   // not found
 }
```



이진 검색 트리 운행

이진 검색트리를 중위운행(inorder traversal)하면

키값에 순서에 의하여 방문할 수 있다. (다음의 예제로 확인)



aha blue child happy sky tiger yellow

이진 검색 트리의 운행 함수

1. 이진 검색트리를 중위운행(inorder traversal)하는 함수를
   재귀용법에 의해 간단히 작성할 수 있다.
2. 재귀적으로 call하면서 구축된 이진트리를 방문하면서 여러 기능(출력/조건체크 등) 수행 가능

```
void inorder(NODETYPE *ptr) {
        if (ptr) {
            inorder(ptr->left);
            printf("%s\n", ptr->word);
            inorder(ptr->right);
        }
}
```

이진 검색 트리 구축 예제

▶ 정수형 key 값이 다음과 같이 차례로 주어지는 경우 만들어지는 이진검색트리를 그리고 중위운행(inorder traversal)한 결과를 쓰시오.

key 값 : 200, 350, 150, 111, 500, 333, 100, 700

프로그래밍 실전(1)

▶ [연습1] 단어 이진검색트리구축에 대하여 수업한 내용을 바탕으로 다음 코드를 시작으로 insert_node, search, inorder함수 정의를 작성하여 완성하시오.

```
typedef struct node {
    struct node *left;
    char *word;
    struct node *right;
} NODETYPE;

int insert_node(NODETYPE *root, char *str);
NODETYPE *search(NODETYPE *root, char *key);
void inorder(NODETYPE *root);
```

```
main()
{
    NODETYPE *tree = NULL, *ptr;
    char wbuf[30];
    printf("  검색 트리에 저장할 단어를 입력하세요.\n
              입력의 끝에는 quit를 입력하세요.\n");
    while (strcmp(gets(wbuf), "quit")) {
        if (!tree) {  // 트리가 비었을때 tree == NULL
            tree = (NODETYPE *)malloc(sizeof(NODETYPE));
            tree->word = (char *)malloc(strlen(wbuf) + 1);
            strcpy(tree->word, wbuf);
            tree->left = tree->right = NULL;
        }
        else
            insert_node(tree, wbuf);
    }
    printf("\n\nEnter a key to search : ");  gets(wbuf);
    ptr = search(tree, wbuf);
    if (ptr)
        printf("%s is in this tree.\n\n", ptr->word);
    else
        printf("%s is not exist.\n\n", wbuf);
   printf("------- 트리안의 단어들 (사전식 순서) -------\n\n");
   inorder(tree);
}
```

```
  검색 트리에 저장할 단어를 입력하세요.
  입력의 끝에는 quit를 입력하세요.
spring
apple
winter
cat
go
stop
baby
happysong
dog
eag
quit


Enter a key to search : happy
happy is not exist.

------- 트리안의 단어들 (사전식 순서) ------

apple
baby
cat
dog
eag
go
happysong
spring
stop
winter
```

프로그래밍 실전(2)

▶ [연습2] 앞의 프로그램을 node 구조에 count를 첨가시켜 단어를 계속 읽으면서 단어의 출현빈도수를 구하기 위한 프로그램으로 수정하시오.

중위 운행한 결과 알파벳 순서로 단어와 그의 빈도수를 출력하도록 한다.

```
typedef struct node {
    struct node *left;
    char *word;
    int count;
    struct node *right;
} NODETYPE;
```

```
main()
{
    NODETYPE *tree = NULL, *ptr;
    char wbuf[30];
    printf("  검색 트리에 저장할 단어를 입력하세요.\n  입력의 끝에는 quit를 입력하세요.\n");
    while (strcmp(gets(wbuf), "quit")) {
        if (!tree) {  // 트리가 비었을때 tree == NULL
            tree = (NODETYPE *)malloc(sizeof(NODETYPE));
            tree->word = (char *)malloc(strlen(wbuf) + 1);
            strcpy(tree->word, wbuf);
            tree->left = tree->right = NULL;
        }
        else
            insert_node(tree, wbuf);
    }
    printf("\n\nEnter a key to search : ");  gets(wbuf);
    ptr = search(tree, wbuf);
    if (ptr)
        printf("%s is in this tree.\n\n", ptr->word);
    else
        printf("%s is not exist.\n\n", wbuf);
   printf("------- 트리안의 단어들 (사전식 순서) ---------------------------------\n\n");
   inorder(tree);
}
```

```
int insert_node(NODETYPE *root, char *word)
{
        NODETYPE *tptr = root, *before;
        int cmp;
        while (tptr)  {
           cmp = strcmp(word, tptr->word);
           if (cmp < 0) {
              before = tptr;
              tptr = tptr -> left;
           }
           else if (cmp > 0) {
              before = tptr;
              tptr = tptr -> right;
           }
           else       // found
              return 0;
        }
                                        tptr = (NODETYPE *)malloc(sizeof(NODETYPE));
                                        tptr -> word = (char *) malloc(strlen(word) + 1);
                                        strcpy(tptr->word, word);
                                        tptr->left = tptr->right = NULL;
                                        if (cmp < 0) before -> left = tptr;
                                        else  before -> right = tptr;
                                        return 1;
                                    } //end of insert_node
```

```
void inorder(NODETYPE *ptr) {
    if (ptr) {
        inorder(ptr->left);
        printf("%s\n", ptr->word);
        inorder(ptr->right);
    }
}
```

```
 검색 트리에 저장할 단어를 입력하세요.
 입력의 끝에는 quit를 입력하세요.
spring
go
cat
dog
with
me
you
go
go
go
yes
ok
dog
spring
go
me
me
you
go
with
ok
with
quit
Enter a key to search : cat
cat is in this tree.

----------------------------------------
cat      1
dog      2
go       6
me       3
ok       2
spring   2
with     3
yes      1
you      2
```

프로그래밍 실전(3)

[연습3] 사원번호(정수형), 입사년도(정수형), 인사등급(문자형)을 자료로 가지는
이진검색트리의 노드 구조(ENODE)를 정의하고 준비한 사원정보를 입력하고 입력의 끝에는 (0 0 0)을 입력한다.
이렇게 입력된 정보를 사원번호를 기준으로 이진 검색트리를 구축하고 그 정보를 출력해 본다.

또한 이렇게 구축된 이진검색트리에서 다음의 두다지 정보를 출력한다

1) 입사년도를 입력하면 그 해에 입사한 사원의 정보(사원번호, 인사등급)를 출력한다
2) 사원번호를 입력하면 그 번호에 해당하는 사원정보(입사년도 인사등급)를 보여준다.

```
typedef struct node {
        struct node *left;
        int eid;
        int syear;
        char grade;
        struct node *right;
} ENODE;
```

```
int insert_node(ENODE *root, int num, int year, char score);
void inorder(ENODE *root);
void year_search(ENODE *root, int year);
ENODE *search(ENODE *root, int key);
```

```
main()
{
    ENODE *tree=NULL, *ptr;
    int id, year;
    char score;
    printf(" 검색 트리에 저장할 사원정보를 입력하세요.\n");
    printf(" 사원번호 입사년도 인사등급을 입력하고 \n");
    printf(" 입력의 끝에는 (0 0 0)를 입력하세요.\n");
    while (1) {
        scanf("%d %d %c", &id, &year, &score);
        if (id == 0) break;
        if (!tree) { // 트리가 비었을때
                tree = (ENODE *)malloc(sizeof(ENODE));
                tree -> eid = id;
                tree -> syear = year;
                tree -> grade = score;
                tree->left = tree->right = NULL;
         }
         else
                insert_node(tree, id, year, score);
}
```

```
printf("\n 트리에 구축된 사원정보 : \n");
inorder(tree);
printf("\n 입사년도 : ");
scanf("%d", &year);
year_search(tree, year);
getchar();
printf("\n 사원 번호 : ");
scanf("%d", &id);
ptr = search(tree, id);
if (ptr)
   printf("%d번 사원의 정보 : %d\t%c\n",
            ptr->eid, ptr->syear, ptr->grade);
else
   printf("%d번 사원에 대한 정보는 없습니다.\n", id);
}
```

```
int insert_node(ENODE *root, int id, int year, char score)
{
    ENODE *tptr = root, *before;

    while (tptr) {
        if (id < tptr->eid) {
            before = tptr;
            tptr = tptr -> left;
        }
        else if (id > tptr->eid) {
            before = tptr;
            tptr = tptr -> right;
        }
        else // found
             return 0;
    }
    tptr = (ENODE *)malloc(sizeof(ENODE));
    tptr -> eid = id;
    tptr -> syear = year;
    tptr -> grade = score;
    tptr->left = tptr->right = NULL;

    if (id < before->eid) before -> left = tptr;
    else before -> right = tptr;
    return 1;
}
```

```
void inorder(ENODE *ptr) {
    if (ptr) {
        inorder(ptr->left);
        printf("%d\t%d\t%c\n",
        ptr->eid, ptr->syear, ptr->grade);
        inorder(ptr->right);
    }
}
```

```
void year_search(ENODE *ptr, int year) {
    if (ptr) {
        year_search(ptr->left, year);
        if ((ptr->syear) == year)
            printf("%d\t%c\n", ptr->eid, ptr->grade);
        year_search(ptr->right, year);
    }
}
```

```
ENODE *search(ENODE *root, int key) {
    ENODE *tptr = root;
    while (tptr) {
        if (key < tptr->eid)
            tptr = tptr -> left;
        else if (key > tptr->eid)
            tptr = tptr -> right;
        else // found
            return tptr;
    }
    return NULL; // not found
}
```

```
 검색 트리에 저장할 사원정보를 입력하세요.
 사원번호 입사년도 인사등급을 입력하고
 입력의 끝에는 (0 0 0)를 입력하세요.
199 2015 B
230 2017 C
101 2014 B
707 2020 C
355 2019 A
717 2020 B
555 2019 B
150 2015 A
220 2016 A
111 2014 C
600 2019 C
299 2017 A
500 2019 B
414 2019 A
0 0 0

 트리에 구축된 사원정보 :
101     2014    B
111     2014    C
150     2015    A
199     2015    B
220     2016    A
230     2017    C
299     2017    A
355     2019    A
414     2019    A
500     2019    B
555     2019    B
600     2019    C
707     2020    C
717     2020    B

 입사년도 : 2019
355     A
414     A
500     B
555     B
600     C

 사원 번호 : 555
555번 사원의 정보 : 2019        B
```