과목명	자료구조
담당교수	우진운 교수님
학과	소프트웨어학과
학번	32153180
이름	이상민
제출일자	2018.11.19

소스 코드

```
#include <iostream>
using namespace std;

class BstNode;                                  // 전방선언
class Bst
{
private:
    BstNode *root;
public:
    Bst() {                                     // 생성자 함수
        root = 0;
    }
    BstNode *IterSearch(const int &x);          // 탐색 함수
    void inorder();                             // 중위우선순회 함수
    void inorder(BstNode *CurrentNode);         // 중위우선순회 함수
    bool Insert(const int &x);                  // 삽입 함수
    bool Delete(const int &x);                  // 삭제 함수
};

class BstNode
{
    friend Bst;                                 // friend 선언
private:
    BstNode *LeftChild;
    int data;
    BstNode *RightChild;
public:
    BstNode(int element = 0, BstNode *left = 0, BstNode *right = 0) {
        data = element;                         // 생성자 함수
        LeftChild = left;
        RightChild = right;
    }
};

BstNode* Bst::IterSearch(const int &x)          // 탐색 함수
{
    for (BstNode *t = root; t; )                // t가 root부터 탐색 시작
    {
        if (x == t->data)
            return t;
        if (x < t->data)
            t = t->LeftChild;
        else
            t = t->RightChild;
    }
    return 0;
}

void Bst::inorder()                             // 중위우선순회 함수
{
    inorder(root);
}

void Bst::inorder(BstNode *CurrentNode)
{
    if (CurrentNode)
    {
        inorder(CurrentNode->LeftChild);        // 왼쪽 자식으로 이동
        cout << CurrentNode->data << ' ';       // data 출력
        inorder(CurrentNode->RightChild);       // 오른쪽 자식으로 이동
    }
}
```

```cpp
bool Bst::Insert(const int &x)                          // 삽입 함수
{
    BstNode *p = root;
    BstNode *q = 0;                                     // p를 뒤따라오는 노드

    while (p)
    {
        q = p;
        if (x == p->data)                               // 삽입하려는 x가 이미 존재할 경우
        {
            cout << x << "가 이미 존재합니다" << endl;
            return false;
        }
        if (x < p->data)
            p = p->LeftChild;
        else
            p = p->RightChild;
    }
    p = new BstNode;
    p->LeftChild = p->RightChild = 0;                   // p의 왼쪽 자식, 오른쪽 자식 NULL
    p->data = x;

    if (!root)                                          // 빈 리스트
        root = p;
    else if (x < q->data)
        q->LeftChild = p;
    else
        q->RightChild = p;
    return true;
}

bool Bst::Delete(const int &x)                          // 삭제 함수
{
    BstNode *p = root;
    BstNode *q = 0;                                     // p를 뒤따라오는 노드

    while (p)                                           // 빈 리스트가 아닌 경우
    {
        if (x < p->data)                                // x가 p의 data보다 작은 경우
        {
            q = p;
            p = p->LeftChild;                           // 왼쪽 자식으로 이동
        }
        else if (x > p->data)                           // x가 p의 data보다 큰 경우
        {
            q = p;
            p = p->RightChild;                          // 오른쪽 자식으로 이동
        }
        else                                          // x가 p의 data와 같은 경우
            break;
    }

    if (!p)                                             // 일치하는 값이 없는 경우
    {
        cout << "삭제할 노드가 없습니다." << endl;
        return false;
    }

    if (p->LeftChild == 0 && p->RightChild == 0)        // 삭제할 노드의 자식이 없는 경우(단말노드)
    {
        if (x < q->data)                                // x가 q의 data보다 작은 경우
        {
            q->LeftChild = 0;                           // q의 왼쪽 자식 NULL
            delete p;                                   // p 삭제
        }
        else                                            // x가 q의 data보다 큰 경우
        {
            q->RightChild = 0;                          // q의 오른쪽 자식 NULL
            delete p;                                   // p 삭제
        }
    }
    else if (p->LeftChild == 0 || p->RightChild == 0)   // 삭제할 노드의 자식이 하나인 경우
    {
        if (p->LeftChild == 0)                          // 오른쪽 자식이 있는 경우
        {
            if (x < q->data)                            // p가 q의 왼쪽 자식인 경우
            {
                q->LeftChild = p->RightChild;           // p의 오른쪽 자식을 q의 왼쪽 자식으로
                delete p;                               // p 삭제
            }
            else                                        // p가 q의 오른쪽 자식인 경우
            {
                q->RightChild = p->RightChild;          // p의 오른쪽 자식을 q의 오른쪽 자식으로
                delete p;                               // p 삭제
            }
        }
```

```
        else                                        // 왼쪽 자식이 있는 경우
        {
            if (x < q->data)                        // p가 q의 왼쪽 자식인 경우
            {
                q->LeftChild = p->LeftChild;        // p의 왼쪽 자식을 q의 왼쪽 자식으로
                delete p;                           // p 삭제
            }
            else                                    // p가 q의 오른쪽 자식인 경우
            {
                q->RightChild = p->LeftChild;       // p의 왼쪽 자식을 q의 오른쪽 자식으로
                delete p;                           // p 삭제
            }
        }
    }
    else                                            // 삭제할 노드의 자식이 둘인 경우
    {
        BstNode *r = p;                             // p를 r에 저장
        q = p;                                      // q는 p를 뒤따라오는 노드
        p = p->RightChild;                          // p는 오른쪽 자식으로 이동

        while (p->LeftChild)
        {
            q = p;
            p = p->LeftChild;
        }
        r->data = p->data;                          // p의 data를 r에 저장

        if (r == q)
        {
            r->RightChild = p->RightChild;          // p의 오른쪽 자식을 r의 오른쪽 자식으로
            delete p;                               // p 삭제
        }
        else if (p->RightChild)                     // p의 오른쪽 자식이 있는 경우
        {
            q->LeftChild = p->RightChild;           // p의 오른쪽 자식을 q의 왼쪽 자식으로
            delete p;                               // p 삭제
        }
        else                                        // p가 단말노드인 경우
        {
            q->LeftChild = 0;                       // q의 왼쪽자식 NULL
            delete p;                               // p 삭제
        }
    }

    return true;
}

int main()
{
    Bst bst;
    int menu, x, val;

    cout << "<<<<<<<<<<<<<<<<<<<< 이진탐색트리 구현 >>>>>>>>>>>>>>>>>>>>" << endl;
    cout << "(1) 삽입  (2) 삭제  (3) 탐색  (4) 중위우선순회  (0) 종료" << endl;

    while (1)
    {
        cout << "[메뉴 입력] ";
        cin >> menu;

        switch (menu)
        {
        case 1:                                     // 1 입력 시 삽입 함수 호출
            cout << "입력할 키 개수 : ";
            cin >> x;
            cout << "삽입할 키 입력 : ";
            for (int i = 0; i < x; i++)
            {
                cin >> val;
                bst.Insert(val);
            }
            break;
        case 2:                                     // 2 입력 시 삭제 함수 호출
            cout << "삭제할 값 입력 : ";
            cin >> val;
            bst.Delete(val);
            break;
```

```
        case 3:                                         // 3 입력 시 탐색 함수 호출
            cout << "탐색할 값 입력 : ";
            cin >> val;
            if (bst.IterSearch(val) == 0)
                cout << val << " => 탐색 실패" << endl;
            else
                cout << val << " => 탐색 성공" << endl;
            break;
        case 4:                                         // 4 입력 시 중위우선순회 함수 호출
            cout << "중위우선순회 : ";
            bst.inorder();
            cout << endl;
            break;
        case 0:                                         // 0 입력 시 종료
            break;
        }

        if (menu == 0)
            break;
    }
    return 0;
}
```

실행 파일

```
C:₩Windows₩system32₩cmd.exe
<<<<<<<<<<<<<<<<<<<< 이진탐색트리 구현 >>>>>>>>>>>>>>>>>>>>
(1) 삽입  (2) 삭제  (3) 탐색  (4) 중위우선순회  (0) 종료
[메뉴 입력] 1
입력할 키 개수 : 10
삽입할 키 입력 : 3 2 1 5 8 0 9 6 11 22
[메뉴 입력] 4
중위우선순회 : 0 1 2 3 5 6 8 9 11 22
[메뉴 입력] 3
탐색할 값 입력 : 6
6 => 탐색 성공
[메뉴 입력] 3
탐색할 값 입력 : 7
7 => 탐색 실패
[메뉴 입력] 2
삭제할 값 입력 : 3
[메뉴 입력] 2
삭제할 값 입력 : 22
[메뉴 입력] 4
중위우선순회 : 0 1 2 5 6 8 9 11
[메뉴 입력] 1
입력할 키 개수 : 5
삽입할 키 입력 : 17 14 23 84 29
[메뉴 입력] 3
탐색할 값 입력 : 14
14 => 탐색 성공
[메뉴 입력] 3
탐색할 값 입력 : 9
9 => 탐색 성공
[메뉴 입력] 2
삭제할 값 입력 : 14
[메뉴 입력] 2
삭제할 값 입력 : 5
[메뉴 입력] 4
중위우선순회 : 0 1 2 6 8 9 11 17 23 29 84
[메뉴 입력] 0
계속하려면 아무 키나 누르십시오 . . .
```