이진탐색트리.c

```
#include "stdafx.h"
#include <iostream>
using namespace std;

class BST;

class Element
{//key 클래스 선언
public:
        int key;
};

class BstNode
{//이진탐색트리노드 클래스 선언
        friend class BST;//이진탐색트리 클래스 프랜드 선언
private:
        Element data;//노드의 data 변수
public:
        BstNode * left, *right;//부모의 자식(leftchild, right child 선언)
        BstNode() {}//생성자
};

class BST
{//이진탐색트리 클래스 선언
private:
        BstNode * root;//이진탐색트리노드형 변수 root 선언
public:
        BST() { root = 0; }//이진탐색트리 생성자 root=0으로 초기화
        bool Insert(const Element &x);//삽입함수
        bool Delete(const Element &x);//삭제함수
        BstNode* Search(const Element &x);//탐색함수
        void Print();//출력함수
        void InorderPrint(const BstNode *p);//중위우선순회함수
};

bool BST::Insert(const Element &x)
{
        BstNode *p = root;
        BstNode *q = 0;
        while (p)
        {//p가 있을 때까지 실행
                q = p;
                if (x.key == p->data.key)//삽입할 값이 존재하면
                        return false;
                if (x.key < p->data.key)//삽입할 값이 p값보다 작을 때
                        p = p->left;
                else
                        p = p->right;
        }
        p = new BstNode;//노드 생성
        p->left = p->right = 0;// 생성후 자식들 초기화
```

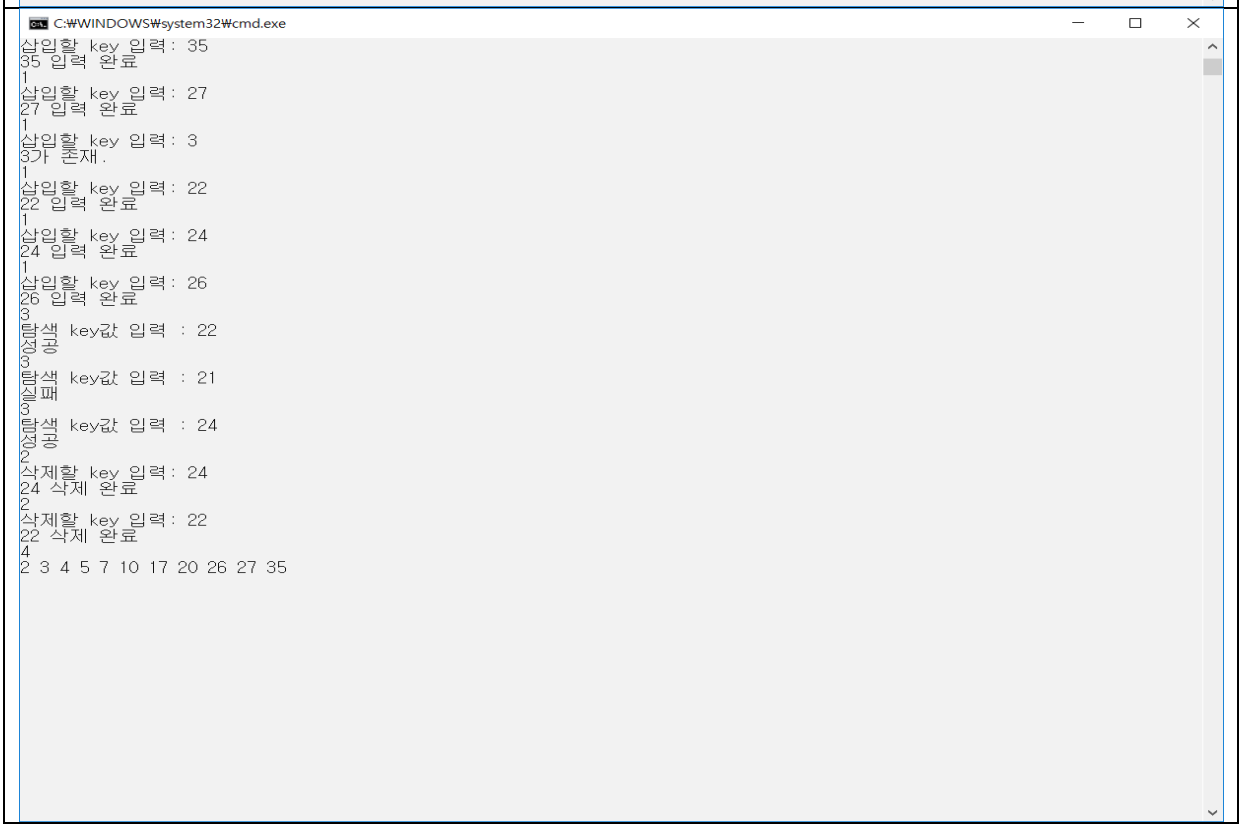
```
        p->data = x;//x값을 p->data에 저장
        if (!root)//root가 없으면 p값을 root에 저장
                root = p;
        else if (x.key < q->data.key)//값보다 작으면 p값을 *q의 left에저장
                q->left = p;
        else
                q->right = p;
        return true;
}

BstNode* BST::Search(const Element &x)
{
        for (BstNode *t = root; t;)//탐색할 Bst노드 t 선언, t가 null이 아닐때 까지 실행
        {
                if (x.key == t->data.key)//탐색 값과 *t 값이 일치하면 t를 반환
                        return t;
                else if (x.key < t->data.key)//탐색 값이 *t보다 작으면 *t의 left로 이동
                        t = t->left;
                else
                        t = t->right;
        }
        return 0;
}

bool BST::Delete(const Element &x)
{
        BstNode *p = root;
        BstNode *q = 0;
        while (p) {
                if (p->data.key == x.key)//p값이 삭제할 값과 같을 경우
                        break;
                q = p;
                if (x.key < p->data.key)//삭제할 값이 p값보다 작을경우
                        p = p->left;//
                else
                        p = p->right;
        }
        if (!p)//빈 경우
                return false;

        if (p->left == 0 && p->right == 0)
        { //자식이 하나도 없을 때
                if (!q)
                        root = 0;
                else if (q->left == p)
                        q->left = 0;
                else
                        q->right = 0;

                delete p;
        }
        else if (p->left == 0 || p->right == 0)
        {//자식이 하나 있을 때
                BstNode *tNode;
                if (p->left == 0)//*p의 left가 없을때 *p의 right가 tNode가 됨
```

```
                        tNode = p->right;
                else
                        tNode = p->left;

                if (q == 0)//q가 비면
                        root = tNode;//tNode가 root
                else if (q->left == p)//같으면 tNode가 *q의 left에 저장
                        q->left = tNode;
                else
                        q->right = tNode;

                delete p;
        }
        else
        {
                BstNode *dpNode = p->right;//*p 기준 오른쪽
                BstNode *dqNode = p;
                while (dpNode->left)
                {//*p 기준으로 오른쪽에서 가장 작은수 찾기
                        dqNode = dpNode;
                        dpNode = dpNode->left;
                }

                p->data.key = dpNode->data.key;//가장 작은수 값을 p값에 저장

                if (dqNode->right == dpNode)
                        dqNode->right = 0;
                else
                        dqNode->left = 0;
                delete dpNode;
        }
        return true;
}

void BST::Print()
{
        InorderPrint(root);
}

void BST::InorderPrint(const BstNode *p)
{
        if (!p)
                return;
        InorderPrint(p->left);//재귀함수 호출(p->left)
        cout << p->data.key << " ";//*p값 출력
        InorderPrint(p->right);//재귀함수 호출(p->right)
}

int main(void)
{
        BST bst;
        Element x;
        int n, key;
        cout << "1.삽입  2.삭제  3.탐색  4.중위우선순회 출력 5.종료" << endl;
```

```
	while (1)
	{
		cin >> n;
		if (n == 1)
		{
			cout << "삽입할 key 입력: ";
			cin >> key;
			x.key = key;
			if (!bst.Insert(x))
				cout << key << "가 존재." << endl;
			else
				cout << key << " 입력 완료" << endl;
		}
		else if (n == 2)
		{
			cout << "삭제할 key 입력: ";
			cin >> key;
			x.key = key;
			if (!bst.Delete(x))
				cout << key << "가 존재하지 않습니다." << endl;
			else
				cout << key << " 삭제 완료" << endl;
		}
		else if (n == 3)
		{
			cout << "탐색 key값 입력 : ";
			cin >> key;
			x.key = key;
			if (!bst.Search(x))
				cout << "실패" << endl;
			else
				cout << "성공" << endl;
		}
		else if (n == 4)
		{
			bst.Print();
			cout << endl;
		}
		else if (n == 5)
			break;
		else
			cout << "메뉴번호 재입력" << endl;
	}

	return 0;
}
```

실행화면

```
C:₩WINDOWS₩system32₩cmd.exe
1.삽입  2.삭제  3.탐색  4.중위우선순회 출력 5.종료
1
삽입할 key 입력 : 4
4 입력 완료
1
삽입할 key 입력 : 5
5 입력 완료
1
삽입할 key 입력 : 3
3 입력 완료
1
삽입할 key 입력 : 2
2 입력 완료
1
삽입할 key 입력 : 7
7 입력 완료
1
삽입할 key 입력 : 10
10 입력 완료
1
삽입할 key 입력 : 20
20 입력 완료
1
삽입할 key 입력 : 17
17 입력 완료
1
삽입할 key 입력 : 9
9 입력 완료
1
삽입할 key 입력 : 1
1 입력 완료
4
1 2 3 4 5 7 9 10 17 20
3
탐색 key값 입력 : 3
성공
3
탐색 key값 입력 : 21
실패
2
삭제할 key 입력 : 9
9 삭제 완료
2
삭제할 key 입력 : 1
1 삭제 완료
4
2 3 4 5 7 10 17 20
1
삽입할 key 입력 : 35
```



```
C:₩WINDOWS₩system32₩cmd.exe
삽입할 key 입력 : 35
35 입력 완료
1
삽입할 key 입력 : 27
27 입력 완료
1
삽입할 key 입력 : 3
3가 존재.
1
삽입할 key 입력 : 22
22 입력 완료
1
삽입할 key 입력 : 24
24 입력 완료
1
삽입할 key 입력 : 26
26 입력 완료
3
탐색 key값 입력 : 22
성공
3
탐색 key값 입력 : 21
실패
3
탐색 key값 입력 : 24
성공
2
삭제할 key 입력 : 24
24 삭제 완료
2
삭제할 key 입력 : 22
22 삭제 완료
4
2 3 4 5 7 10 17 20 26 27 35
```