`KVTree` WITH A `list` ITERATOR

The final chapter

Delroy A. Brinkerhoff

KVTree DATA MEMBERS

```
#include <list>

template <class K, class V>
class KVTree
{
    private:
       K              key;
       V              value;
       KVTree<K, V>*  left = nullptr;
       KVTree<K, V>*  right = nullptr;
    public:
       list<K> keys;
```

KVTree OPERATIONS

```
public:
    ~~iterator get_keys() { iterator i(this); return i; }~~
    auto get_keys() { fill_list(); return keys.begin(); }
    auto get_end() { return keys.end(); }

private:
    void fill_list();
    void add_keys(KVTree<K,V>* tree);
```

BUILDING THE KVTree ITERATOR

```
template<class K, class V>
void KVTree<K,V>::fill_list()
{
    keys.clear();

    if (right != nullptr)
        add_keys(right);
}
```

```
template <class K, class V>
void KVTree<K, V>::
    add_keys(KVTree<K,V>* tree)
{
    if (tree->left != nullptr)
        add_keys(tree->left);
    keys.push_back(tree->key);
    if (tree->right != nullptr)
        add_keys(tree->right);
}
```

USING THE KVTree:
THE CLIENT CODE

```
list<string>::iterator keys = tree.get_keys();
list<string>::iterator end = tree.get_end();

while (keys != end)
{
    string  word = *keys++;
    int     count = *tree.search(word);
    cout << left << setw(20) << word <<
        right << setw(3) << count << endl;
}
```

BUILDING AND USING THE ITERATOR: A SUB-OPTIMAL SOLUTION

```
template <class K, class V>
void KVTree<K, V>::
    add_keys(KVTree<K,V>* tree)
{
    if (tree->left != nullptr)
        add_keys(tree->left);
    keys.push_back(tree->key);
    if (tree->right != nullptr)
        add_keys(tree->right);
}
```

```
while (keys != end)
{
    string  word = *keys++;
    int count = *tree.search(word);
    ...
}
```

OPTIMIZING THE KVTree WITH THE STL pair

- `list<pair<K,V>> keys;`
- `keys.push_back(make_pair(tree->key, tree->value));`
-
```
list<pair<string, int>>::iterator keys = tree.get_keys();
list<pair<string, int>>::iterator end = tree.get_end();

while (keys != end)
{
    cout << left << setw(20) << keys->first << right <<
        setw(3) << keys->second << endl;
    keys++;
}
```