

# Associative Containers and Custom Types

# Book Class

```
class book {  
private:  
    string title;  
    string publisher;  
public:  
    book(string title, string publisher) : title(title), publisher(publisher) {}  
    ...  
};  
  
multimap<string, book> library;
```

- Add some books to the library

```
book prog_princs("Programming Principles and Practice", "Addison-Wesley");  
library.insert( {"Stroustrup, Bjarne", prog_princs} );
```

```
book cpp_primer("C++ Primer", "Addison-Wesley");  
library.insert( {"Lippman, Stanley B.", cpp_primer} );
```

```
book cpp_prog("The C++ Programming Language", "Addison-Wesley");  
library.insert( {"Stroustrup, Bjarne", cpp_prog} );
```

```
// for (auto b: library)
```

```
//   cout << b.first << ", " << b.second << endl;
```

- The new class will contain the fields we want to sort on

```
class book_idx {  
private:  
    string author;  
    string title;  
public:  
    ...  
};
```

- We will use this class as the key in our multimap

```
multimap<book_idx, book> library;
```

## book\_idx < Operator

```
bool book_idx::operator < (const book_idx& other) const {  
    // Check if the authors are different  
    if (author != other.author) {  
        // The authors are different, so we can sort by them  
        return author < other.author;  
    }  
  
    // If we get here, the authors are the same  
    // We sort by title  
    return title < other.title;  
}
```