

Implementing Binary Search

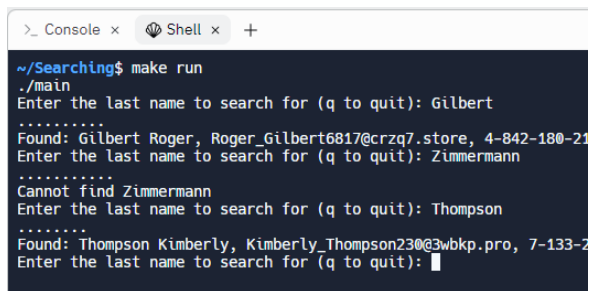
Here's an implementation of binary search, named `bfind()`:

```
1  int bfind(const Person data[], int first, int last,
2          const string& key)
3  {
4      cout << ".";           // display comparisons
5      if (last < first) {return -1;} // not found
6      int mid = (first + last) / 2; // mid point
7
8      if (key == data[mid].name.substr(0, key.size())) return mid;
9      if (key < data[mid].name.substr(0, key.size()))
10         return bfind(a, first, mid - 1, key); // Look in Left
11     else
12         return bfind(a, mid + 1, last, key); // Look in right
13 }
```

Add this code **before** your `find()` function, and then comment out the body of `find()`, and add a call to `bfind()` in its place, like this:

```
return bfind(contacts, 0, size - 1, key);
```

Now, when you run the program, you'll see that even the worst case will take only about 10 or 11 comparisons, instead of 500 or 1000.



```
>_ Console x Shell x +
~/Searching$ make run
./main
Enter the last name to search for (q to quit): Gilbert
.....
Found: Gilbert Roger, Roger_Gilbert6817@crzq7.store, 4-842-180-21
Enter the last name to search for (q to quit): Zimmermann
.....
Cannot find Zimmermann
Enter the last name to search for (q to quit): Thompson
.....
Found: Thompson Kimberly, Kimberly_Thompson230@3wbkp.pro, 7-133-2
Enter the last name to search for (q to quit):
```



This course content is offered under a [CC Attribution Non-Commercial](#) license. Content in this course can be considered under this license unless otherwise noted.