

```
1 #include "stdafx.h"
2 /* Hash.cpp
3 *
4 * Hash table implementation from:
5 * Kernighan & Ritchie, The C Programming Language,
6 * Second Edition, Prentice-Hall, 1988.
7 */
8
9 #include <iostream>
10 #include <iomanip>
11 #include <stdlib>
12 #include <cstring>
13 #include <string>
14
15 using namespace std;
16
17 #include "hash.h"
18
19
20
21 static NListPtr hashTable[HASH_TABLE_SIZE];
22 static int bucketSize[HASH_TABLE_SIZE];
23
24
25 /* Hash
26 * Generate hash value for string s
27 */
28
29 unsigned Hash(string s)
30 {
31     unsigned hashVal = 0;
32
33     for (int i = 0; i < s.length(); i++)
34         hashVal = s.at(i) + 31 * hashVal;
35
36     return hashVal % HASH_TABLE_SIZE;
37 }
38
39
40 /* Lookup
41 * Look for s in hashTable
42 */
43
44 NListPtr Lookup(string s)
45 {
46     NListPtr np;
47     int count = 0;
48
49     for (np = hashTable[Hash(s)]; np != NULL; np = np->next)
```

```
50     {
51         count++;
52         //if (strcmp(s, np->name) == 0)
53         if (s.compare(np->name) == 0) {
54             //cout << s << ": " << count << endl;
55             return np;    // found
56         }
57     }
58
59     return NULL;        // not found
60 }
61
62 NListPtr LookupPrintCount(string s)
63 {
64     NListPtr np;
65     int count = 0;
66
67     for (np = hashTable[Hash(s)]; np != NULL; np = np->next)
68     {
69         count++;
70         //if (strcmp(s, np->name) == 0)
71         if (s.compare(np->name) == 0) {
72             cout << s << ": " << count << endl;
73             return np;    // found
74         }
75     }
76
77     return NULL;        // not found
78 }
79
80 /* Insert
81 * Put (name) in hash table
82 */
83
84 NListPtr Insert(string name)
85 {
86     unsigned hashVal;
87     NListPtr np;
88
89     if ((np = Lookup(name)) == NULL) // not found
90     {
91         //np = (NListPtr)malloc(sizeof(*np));
92         np = new nList;
93         if (np == NULL )
94             return NULL;
95
96         np->name = name;
97         hashVal = Hash(name);
98         np->next = hashTable[hashVal];
```

```
199         hashTable[hashVal] = np;
200         bucketSize[hashVal]++;
201     }
202
203     return np;
204 }
205
206
207 /* PrintHashTable
208 * Print the hash table contents
209 */
210
211 void PrintHashTable()
212 {
213     NListPtr np;
214
215     cout << "Hash table contents:" << endl;
216     cout << "-----\n" << endl;
217
218     for (int i = 0; i < HASH_TABLE_SIZE; i++)
219     {
220         np = hashTable[i];
221         while (np != NULL)
222         {
223             cout << setw(3) << i << ":   ";
224             //cout << np->name;
225             cout << endl;
226             np = np->next;
227         }
228     }
229 }
230
231 void PrintHashTableBuckets()
232 {
233     NListPtr np;
234
235     cout << "Hash table bucket amounts:" << endl;
236     cout << "-----\n" << endl;
237
238     for (int i = 0; i < HASH_TABLE_SIZE; i++)
239     {
240         cout << i << ": " << bucketSize[i] << endl;
241     }
242 }
243
244 void PrintHashTableBucketsMinMax()
245 {
246     NListPtr np;
247
```

```
148     cout << "Hash table bucket min-max:" << endl;
149     cout << "-----\n" << endl;
150     int max = INT_MIN;
151     int min = INT_MAX;
152     int empty = 0;
153
154     for (int i = 0; i < HASH_TABLE_SIZE; i++)
155     {
156         //cout << i << ": " << bucketSize[i] << endl;
157         if (bucketSize[i] > max) {
158             max = bucketSize[i];
159         }
160         if (bucketSize[i] < min) {
161             min = bucketSize[i];
162         }
163         if (bucketSize[i] == 0) {
164             empty++;
165         }
166     }
167
168     cout << "MAX BUCKET SIZE: " << max << endl;
169     cout << "MIN BUCKET SIZE: " << min << endl;
170     cout << "EMPTY BUCKETS : " << empty << endl;
171 }
```