|  |  |  |
| --- | --- | --- |
| **数据结构上机实习报告** | | |
| 电话号码查询系统 | | |
|  | 班级： | 191174班 |
|  | 学号： | 20161001764 |
|  | 姓名： | 牟鑫一 |
|  | 日期： | 2018.11.23 |

# 一、课程设计题目与要求

## 1.1 题目：电话号码查询系统

## 1.2 系统功能要求

【基本要求】

1. 设每个记录有下列数据项：电话号码、用户名、地址；
2. 从键盘输入各记录，分别以电话号码和用户名为关键字建立散列表；
3. 采用合适的方法解决冲突；
4. 查找并显示给定电话号码的记录；
5. 查找并显示给定用户名的记录。

【实现提示】

构造哈希函数时，要注意使其分布均匀。用户名的长度均不超过20个字符。可先对过长的用户名作折叠处理。

【选作内容】

1. 从几种哈希函数构造方法中选出适用者并设计几个不同的哈希函数，比较它们的地址冲突率。
2. 在哈希函数确定的前提下，尝试各种不同的处理冲突的方法，并比较平均查找长度的变化。

# 二、需求分析

## 2.1 问题描述

设计散列表实现电话号码查询系统

## 2.2 系统环境

Microsoft Visual Studio 2017

## 2.3 运行要求

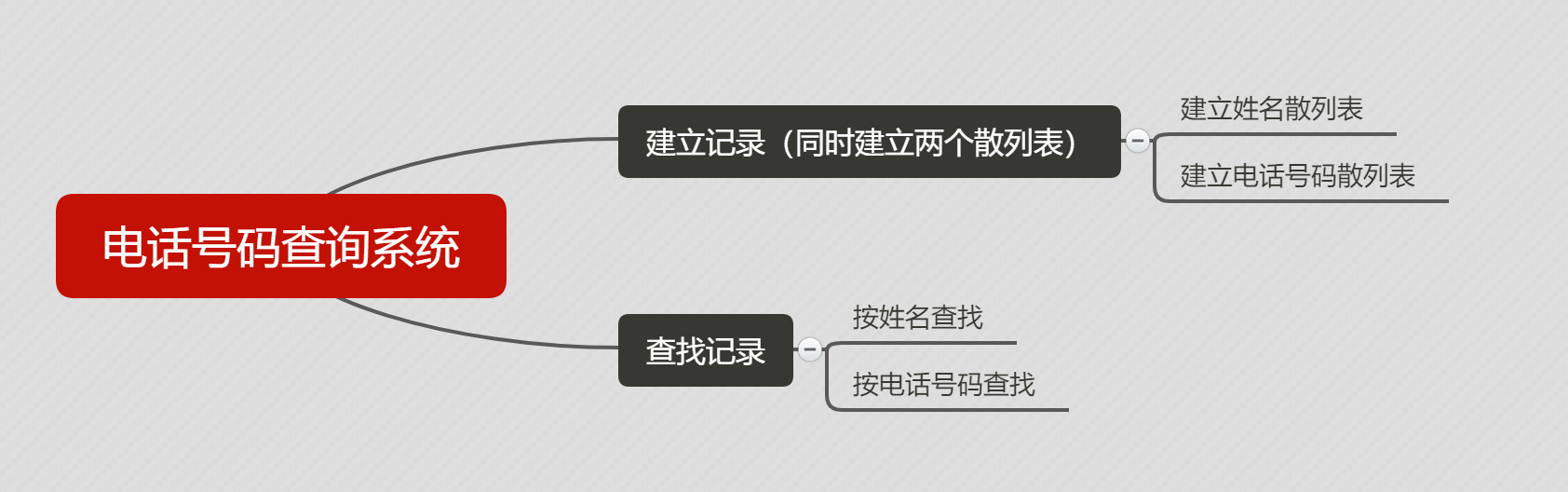
Windows环境下运行

# 三、概要设计

## 3.1 数据结构和储存结构的设计

散列表保存位置信息，开散列法解决冲突

## 3.2 逻辑结构设计



建立两个散列表，分别保存姓名（NameTable[]）和电话号码（NumTable[]）的位置信息。定义散列表大小为20。

根据姓名建立散列表，将姓名各个字符的ASCII码相加得sum，再用sum对散列表大小20求余（哈希函数：hash = sum % lenHash）得到对应散列表中的位置。用开散列法解决冲突，NameTable[hash]如果为空则未发生冲突，直接插入数据，如果非空则发生冲突，将新的数据插入到hash位置的链表尾。

根据电话号码建立散列表以及处理冲突的方法与根据姓名完全一致。

建立记录时用输入的信息同时建立姓名和电话号码两个散列表，方便查找记录时，录入的信息不管按姓名还是按电话号码都可以找到。

查找记录分按姓名查找和按电话号码查找。

按姓名查找，输入要查找的姓名，用建立姓名散列表时相同的哈希函数计算得到对应的hash值，若NameTable[hash]为空直接输出找不到相关数据的提示信息，否则按链表顺序查找，找到相同name的数据则输出找到相关数据的提示信息，并跳出循环结束查找，若直到链表尾也没有找到同样输出找不到相关数据的提示信息。

按电话号码查找的方法与按姓名查找也完全一样。

# 四、详细设计

## 4.1 结构

**Treenode：**

string NAME; //姓名

string NUMBER; //电话号码

string ADDRESS; //地址

Node \* next; //开散列法处理冲突

## 4.2 函数

//根据姓名建立记录

void addNameberHashList(string name, string num, string addr)

//根据电话号码建立记录

void addNumberHashList(string name, string num, string addr)

//按姓名查找

void findNameHsah(string name)

//按电话号码查找

void findNumHsah(string num)

# 测试

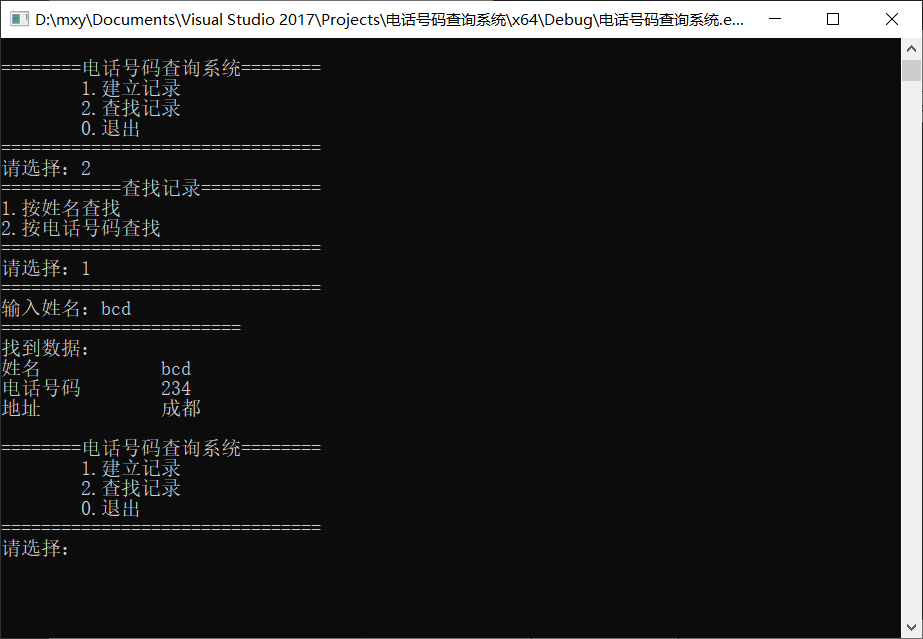
1.主界面



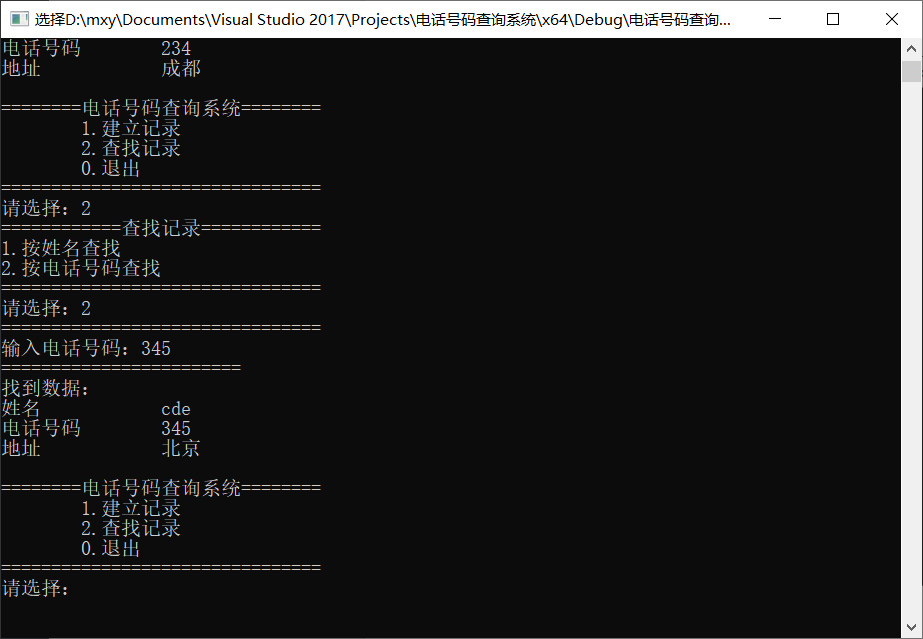
2.建立记录



3.查找记录（按姓名查找）



4.查找记录（按电话号码查找）



# 六、源程序代码

1. // 电话号码查询系统.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。
3. #include"pch.h"
4. #include<iostream>
5. #include<string>
6. **using** **namespace** std;
8. **const** **int** Max = 100;
9. **const** **int** lenHash = 20;   //散列表大小
11. **struct** Node
12. {
13. string NAME;          //姓名
14. string NUMBER;        //电话号码
15. string ADDRESS;       //地址
16. Node \* next;          //开散列法处理冲突
17. Node()
18. {
19. next = NULL;
20. };
21. Node(string name, string num, string addr)
22. {
23. NAME = name;
24. NUMBER = num;
25. ADDRESS = addr;
26. next = NULL;
27. }
28. };
30. Node \*NameTable[Max];
31. Node \*NumTable[Max];
33. **void** addNameberHashList(string name, string num, string addr)   //根据姓名建立记录
34. {
35. **int** sum\_name = 0;
36. **for** (**int** i = 0; i < name.length(); i++)    //把姓名的各个字符转换为ASCII码并求和
37. {
38. sum\_name += **int**(name[i]);
39. }
40. **int** hash\_name = sum\_name % lenHash; //哈希函数，计算要插入的数据在NameTable[]中的位置
41. Node \*temp\_name = NameTable[hash\_name];
42. **if** (temp\_name == NULL)
43. {
44. NameTable[hash\_name] = **new** Node(name, num, addr);
45. }
46. **else** {
47. **while** (temp\_name->next != NULL)
48. {
49. temp\_name = temp\_name->next;
50. }
51. temp\_name->next = **new** Node(name, num, addr);
52. }
53. }
55. **void** addNumberHashList(string name, string num, string addr)   //根据电话号码建立记录
56. {
57. //所有操作与根据姓名建立记录类似
58. **int** sum\_num = 0;
59. **for** (**int** i = 0; i < num.length(); i++)
60. {
61. sum\_num += num[i];
62. }
63. **int** hash\_num = sum\_num % lenHash;
64. Node \*temp\_num = NumTable[hash\_num];
65. **if** (temp\_num == NULL)
66. {
67. NumTable[hash\_num] = **new** Node(name, num, addr);
68. }
69. **else** {
70. **while** (temp\_num->next != NULL)
71. {
72. temp\_num = temp\_num->next;
73. }
74. temp\_num->next = **new** Node(name, num, addr);
75. }
76. }
78. **void** findNameHsah(string name)    //按姓名查找
79. {
80. **int** sum = 0;
81. **for** (**int** i = 0; i < name.length(); i++)    //根据哈希函数计算name对应的位置
82. {
83. sum += name[i];
84. }
85. **int** hash = sum % lenHash;
86. Node \*temp = NameTable[hash];
87. **if** (temp == NULL)      //NameTable[hash]为空直接输出找不到的信息并结束查找
88. {
89. cout << "========================" << endl;
90. cout << "不存在名为" << name << "的数据" << endl;
91. }
92. **else**
93. {      //NameTable[hash]非空，按链表顺序继续查找
94. **while** (temp->NAME != name)
95. {
96. **if** (temp->next == NULL)
97. {
98. cout << "========================" << endl;
99. cout << "不存在名为" << name << "的数据" << endl;
100. **goto** END;      //直到链表尾也没有找到则跳转到函数尾结束查找
101. }
102. **else** temp = temp->next;
103. }
104. cout << "========================" << endl;
105. cout << "找到数据：" << endl;
106. cout << "姓名\t\t" << temp->NAME << endl;
107. cout << "电话号码\t" << temp->NUMBER << endl;
108. cout << "地址\t\t" << temp->ADDRESS << endl;
109. END:;
110. }
111. }
113. **void** findNumHsah(string num)     //按电话号码查找
114. {
115. **int** sum = 0;
116. **for** (**int** i = 0; i < num.length(); i++)    //根据哈希函数计算num对应的位置
117. {
118. sum += num[i];
119. }
120. **int** hash = sum % lenHash;
121. Node \*temp = NumTable[hash];
122. **if** (temp == NULL)       //NumTable[hash]为空直接输出找不到的信息并结束查找
123. {
124. cout << "========================" << endl;
125. cout << "不存在电话号码为" << num << "的数据" << endl;
126. }
127. **else**
128. {      //NumTable[hash]非空，按链表顺序继续查找
129. **while** (temp->NUMBER != num)
130. {
131. **if** (temp->next == NULL)
132. {
133. cout << "========================" << endl;
134. cout << "不存在电话号码为" << num << "的数据" << endl;
135. **goto** END;      //直到链表尾也没有找到则跳转到函数尾结束查找
136. }
137. **else** temp = temp->next;
138. }
139. cout << "========================" << endl;
140. cout << "找到数据：" << endl;
141. cout << "姓名\t\t" << temp->NAME << endl;
142. cout << "电话号码\t" << temp->NUMBER << endl;
143. cout << "地址\t\t" << temp->ADDRESS << endl;
144. END:;
145. }
146. }
148. **int** main()
149. {
150. **char** name[20];    //姓名
151. **char** num[20];     //电话号码
152. string addr;      //地址
154. **char** func = 1;
155. **while** (func != '0')
156. {
157. cout << endl;
158. cout << "========电话号码查询系统========" << endl;
159. cout << " 1.建立记录" << endl;
160. cout << " 2.查找记录" << endl;
161. cout << " 0.退出" << endl;
162. cout << "================================" << endl;
163. cout << "请选择：";
164. MENU:cin.clear();
165. cin >> func;
166. **switch** (func)
167. {
168. //建立记录
169. **case** '1':
170. {
171. cout << "================================" << endl;
172. cout << "请依次输入姓名、电话号码和地址：" << endl;
173. **while** (cin >> name >> num >> addr)
174. {
175. addNameberHashList(name, num, addr);     //分别通过姓名和电话号码建立两个hash表
176. addNumberHashList(name, num, addr);
177. }
178. cout << "================================" << endl;
179. cout << "数据已录入成功！" << endl;
180. }**break**;
182. //查找记录
183. **case** '2':
184. {
185. **char** func2;
186. cout << "============查找记录============" << endl;
187. cout << "1.按姓名查找" << endl;
188. cout << "2.按电话号码查找" << endl;
189. cout << "================================" << endl;
190. cout << "请选择：";
191. MENU2:cin >> func2;
192. cout << "================================" << endl;
193. **switch** (func2)
194. {
195. //按姓名查找
196. **case** '1':
197. {
198. string findname;
199. cout << "输入姓名：";
200. cin >> findname;
201. findNameHsah(findname);
202. }**break**;
203. //按电话号码查找
204. **case** '2':
205. {
206. string findnum;
207. cout << "输入电话号码：";
208. cin >> findnum;
209. findNumHsah(findnum);
210. }**break**;
211. //重新选择
212. **default**:
213. {
214. cout << "================================" << endl;
215. cout << "输入有误，请重新选择：";
216. **goto** MENU2;
217. }
218. }
219. }**break**;
221. //退出系统
222. **case** '0':
223. {
224. func = '0';
225. **break**;
226. }
228. //重新选择
229. **default**:
230. {
231. cout << "输入有误，请重新选择：";
232. **goto** MENU;
233. }
234. }
235. }
236. **return** 0;
237. }