

# 22336216-陶宇卓-Project7-实验报告

---

程序功能简要说明

程序运行截图，包括计算功能演示、部分实际运行结果展示、命令行或交互式界面效果

部分关键代码及其说明

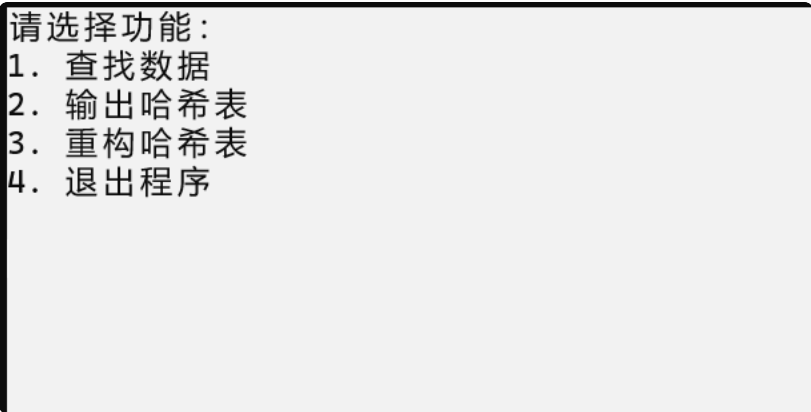
程序运行方式简要说明

## 程序功能简要说明

针对某个集体（例如是所在的班级）中的“姓名”设计一个哈希表，使得平均查找长度不超过 $L$ ，完成相应的建表和查表程序。

假设姓名是汉语拼音的形式，需填入哈希表的姓名共有30个，取平均查找长度的上限为2。哈希函数采用除留余数法的方式进行构造，并用伪随机探测再散列法处理冲突。

## 程序运行截图，包括计算功能演示、部分实际运行结果展示、命令行或交互式界面效果



```
请选择功能：
1. 查找数据
2. 输出哈希表
3. 重构哈希表
4. 退出程序
```

```

1
请输入要查找的姓名：Sun ZeKun
姓名：Sun ZeKun，在哈希表中的索引：35
查找次数：1
请选择功能：
1. 查找数据
2. 输出哈希表
3. 重构哈希表
4. 退出程序

```

伪随机：

哈希表内容：			
索引	姓名	键值	查找次数
0	Chu ShiSi	800	1
1	Li Si	401	1
2	Zhou Zheng	962	1
3	Qin ErShi	803	1
4		0	0
5		0	0
6	Han ShiBa	766	1
7		0	0
8	Wu Shi	528	1
9		0	0
10	You Er ShiYi	1050	1
11	Feng ShiEr	891	1
12		0	0
13	Zheng ShiYi	1026	2
14	Su Zhuo	654	1
15	Lu ShiLiu	815	1
16	Tang Er ShiBa	1096	1
17	Liu SanShi	912	2
18	Cai ShenLin	990	2
19	Yang ShiJiu	1019	1
20	Zhu ShiSi	823	8
21	Wei ShiWu	821	1
22	Xu ShiWu	733	4
23	Zhu SiShi	823	1
24		0	0
25		0	0
26	Zhang San	826	1
27		0	0
28	Shen ShiQi	908	1
29		0	0
30	Zhou Jiu	750	1
31	Jiang ShiLiu	1111	1
32	Tao YuZhuo	952	1
33	Wang Wu	633	1
34	Wei Er ShiQi	1018	2
35	Sun ZeKun	835	1
36	Chen ShiSan	996	1
37		0	0
38	Chen ZhenHuang	1318	1
39	Chen ZhengXi	1115	4
平均查找次数：1.56667			

线性：

哈希表内容：			
索引	姓名	键值	查找次数
0	Chu ShiSi	800	1
1	Li Si	401	1
2	Zhou Zheng	962	1
3	Qin ErShi	803	1
4		0	0
5		0	0
6	Han ShiBa	766	1
7		0	0
8	Wu Shi	528	1
9		0	0
10	You Er ShiYi	1050	1
11	Feng ShiEr	891	1
12		0	0
13	Xu ShiWu	733	1
14	Su Zhuo	654	1
15	Lu ShiLiu	815	1
16	Tang Er ShiBa	1096	1
17		0	0
18	Wei Er ShiQi	1018	1
19	Yang ShiJiu	1019	1
20		0	0
21	Wei ShiWu	821	1
22		0	0
23	Zhu SiShi	823	1
24	Zhu ShiSi	823	2
25		0	0
26	Zhang San	826	1
27	Zheng ShiYi	1026	2
28	Shen ShiQi	908	1
29		0	0
30	Zhou Jiu	750	1
31	Jiang ShiLiu	1111	1
32	Tao YuZhuo	952	1
33	Wang Wu	633	1
34	Cai ShenLin	990	5
35	Sun ZeKun	835	1
36	Chen ShiSan	996	1
37	Chen ZhengXi	1115	3
38	Chen ZhenHuang	1318	1
39	Liu SanShi	912	8
平均查找次数：1.5			

请选择功能：

1. 查找数据
2. 输出哈希表
3. 重构哈希表
4. 退出程序

3

请输入新的哈希表大小：50

哈希表已重构

哈希表内容：

索引	姓名	键值	查找次数
0	Zhou Jiu	750	1
1	Li Si	401	1
2	Tao YuZhuo	952	1
3	Chu ShiSi	800	4
4	Qin ErShi	803	2
5	You Er ShiYi	1050	6
6	Su Zhuo	654	3
7		0	0
8	Shen ShiQi	908	1
9		0	0
10		0	0
11	Jiang ShiLiu	1111	1
12	Zhou Zheng	962	1
13	Liu SanShi	912	2
14		0	0
15	Lu ShiLiu	815	1
16	Han ShiBa	766	1
17	Chen ZhengXi	1115	3
18	Chen ZhenHuang	1318	1
19	Yang ShiJiu	1019	1
20	Wei Er ShiQi	1018	3
21	Wei ShiWu	821	1
22		0	0
23	Zhu SiShi	823	1
24	Zhu ShiSi	823	2
25		0	0
26	Zhang San	826	1
27	Zheng ShiYi	1026	2
28	Wu Shi	528	1
29		0	0
30		0	0
31		0	0
32		0	0
33	Wang Wu	633	1
34	Xu ShiWu	733	2
35	Sun ZeKun	835	1
36		0	0
37		0	0
38		0	0
39		0	0
40	Cai ShenLin	990	1
41	Feng ShiEr	891	1

# 部分关键代码及其说明

定义了字典

```
1  mt19937 rng;
2
3  vector<string> names = { "Zhang San", "Li Si", "Wang Wu", "Sun ZeKun", "Ta
4  o YuZhuo", "Zhou Zheng",
5  "Zhou Jiu", "Wu Shi", "Zheng ShiYi", "Feng ShiEr", "Chen ShiSan",
6  "Chu ShiSi", "Wei ShiWu",
7  "Jiang ShiLiu", "Shen ShiQi", "Han ShiBa", "Yang ShiJiu", "Zhu SiSh
8  i", "Qin ErShi", "You Er ShiYi",
   "Cai ShenLin", "Chen ZhenHuang", "Zhu ShiSi", "Xu ShiWu", "Lu ShiLi
   u", "Wei Er ShiQi", "Tang Er ShiBa",
   "Chen ZhengXi", "Liu SanShi", "Su Zhuo" };
```

定义HashTable类

```

1
2 class HashTable {
3 public:
4     HashTable(int size) : size(size), table(size, ""), attempts(size, 0)/
        *, randomlist(30, 0)*/ {
5         /*Inirandom(randomlist);*/
6     }
7
8
9
10    int hashFunction(const string& name) {
11        // 简单的除留余数法哈希函数
12        int total = 0;
13        for (char c : name) {
14            total += static_cast<int>(c);
15        }
16        return (total+size) % size;
17    }
18    int pseudoRandomFunction(int key, int attempt) {
19        // 线性探测再散列法
20        return (key + attempt) % size;
21    }
22    //int pseudoRandomFunction(int key, int attempt) {
23    //    // 使用伪随机数生成器
24    //    uniform_int_distribution<int> distribution(1, size - 1);
25    //    return (key + distribution(rng)) % size;
26    //}
27
28    void insert(const string& name) {
29        int attempt = 0;
30        int index = hashFunction(name);
31
32        while (!table[index].empty()) {
33            // 冲突处理
34            attempt++;
35            index = pseudoRandomFunction(hashFunction(name), attempt);
36        }
37
38        table[index] = name;
39        attempts[index] = attempt + 1;
40    }
41
42    int search(const string& name, int& numAttempts) {
43        int attempt = 0;
44        int index = hashFunction(name);

```

```

45
46     while (!table[index].empty()) {
47         if (table[index] == name) {
48             numAttempts = attempt + 1; // 找到了
49             return static_cast<int>(index);
50         }
51     }
52     else {
53         // 冲突处理
54         attempt++;
55         index = pseudoRandomFunction(hashFunction(name), attempt)
56     };
57 }
58
59 numAttempts = attempt + 1; // 未找到
60 return -1;
61 }
62
63 void displayTable() {
64     cout << endl << "                哈希表内容: " << endl;
65     int totalAttempts = 0;
66
67     cout << left << setw(15) << "索引" << setw(15) << "姓名" << setw(1
5) << "键值" << "查找次数" << endl;
68
69     for (int i = 0; i < size; ++i) {
70         int total = 0;
71         for (char c : table[i]) {
72             total += static_cast<int>(c);
73         }
74
75         cout << left << setw(15) << i << setw(15) << table[i] << setw
(15) << total << attempts[i] << endl;
76         totalAttempts += attempts[i];
77     }
78
79     double averageAttempts = static_cast<double>(totalAttempts) / 30;
80     cout << "平均查找次数: " << averageAttempts << endl << endl;
81 }
82
83 void rebuild(int newSize) {
84     size = newSize;
85     table.clear();
86     table.resize(size);
87     attempts.clear();
88     attempts.resize(size);
89
90     for (const string& name : names) {

```

```

90         insert(name);
91     }
92 }
93
94 private:
95     int size;
96     vector<string> table;
97     vector<int> attempts;
98     /* vector<int> randomlist; */
99 };
100

```

private: table代表哈希表，attempt代表查找次数

public: hashFunction()是简单的除留余数法取下标；pseudoRandomFunction () 代表用线性/伪随机探测再散列法来处理冲突；insert ()用来插入元素；displaytable()用来可视化哈希表；rebuild () 用来重构哈希表大小。

## 程序运行方式简要说明

main.cpp代码如下：

首先创建大小为initialSize的哈希表，接着向里面插入30个名字。在接下来进入死循环，选择四个功能：查找元素、打印哈希表、重构哈希表、退出程序。接着调用相应的函数即可。



```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <windows.h>
5  #include "Hash.h"
6
7  using namespace std;
8
9  int main() {
10     system("color F0");
11     int initialSize = 40;
12
13     HashTable hashTable(initialSize);
14
15     // 插入测试数据
16     for (const string& name : names) {
17         hashTable.insert(name);
18     }
19
20     int choice;
21     while (true) {
22         cout << "请选择功能: " << endl;
23         cout << "1. 查找数据\n2. 输出哈希表\n3. 重构哈希表\n4. 退出程序\n" << endl;
24         cin >> choice;
25
26         switch (choice) {
27             case 1: {
28                 string name;
29                 cout << "请输入要查找的姓名: ";
30                 cin.ignore();
31                 getline(cin, name);
32
33                 int numAttempts;
34                 int index = hashTable.search(name, numAttempts);
35
36                 if (index != -1) {
37                     cout << "姓名: " << name << ", 在哈希表中的索引: " << index << endl;
38                     cout << "查找次数: " << numAttempts << endl;
39                 }
40                 else {
41                     cout << "姓名: " << name << " 未找到" << endl;
42                 }
43             }
```

```

44         break;
45     }
46     case 2:
47         hashTable.displayTable();
48         break;
49     case 3: {
50         int newSize;
51         cout << "请输入新的哈希表大小: ";
52         cin >> newSize;
53         hashTable.rebuild(newSize);
54         cout << "哈希表已重构" << endl;
55         break;
56     }
57     case 4:
58         cout << "程序退出\n";
59         return 0;
60     default:
61         cout << "无效选择, 请重新输入\n";
62     }
63 }
64 }

```