Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

ХЕШИРОВАНИЕ

Отчет по лабораторной работе № 5 по дисциплине «Структуры и алгоритмы обработки данных в ЭВМ»

Выполнил: Обучающийся гр. 439-3
(группа)

_____ А. С. Мазовец
(подпись) *(И. О. Фамилия)*

« _____ » _____ 2021г.
(дата)

Проверил: ассистент кафедры АСУ
(должность, ученая степень, звание)

_____ _____ Я. В. Яблонский
(оценка) *(подпись)* *(И. О. Фамилия)*

« _____ » _____ 2021г.
(дата)

Томск 2021

1 Задание на лабораторную работу

1.1 Вариант 17

Написать программу, которая реализует метод открытого хеширования и хеш-функцией, основанной на методе деления с остатком. Данные, хранящиеся в файле занести в хеш-таблицу. Файл должен содержать не менее 15 целых чисел. Вывести построенную хеш-таблицу на экран (вместе с количеством выполненных проб). Организовать поиск данных в хеш-таблице. Результаты поиска данных вывести на экран. Также вывести количество проб, которые были затрачены при поиске.

2 Листинг программы

2.1 Заголовочный файл

```
 1| #ifndef HASH_HPP_
 2| #define HASH_HPP_
 3| #include <iostream>
 4| #include "list.hpp"
 5|
 6| namespace ads {
 7|     class entry {
 8|             friend std::ostream& operator<< (std::ostream& o,
    const entry &e);
 9|
10|         friend bool operator== (entry A, int B);
11|         friend bool operator== (int A, entry B);
12|
13|         friend bool operator== (entry A, entry B);
14|
15|         public:
16|             int key;
17|             int value;
18|
19|             entry (int);
20|             entry (int, int);
21|     };
22|
23|     class hash {
24|             friend std::ostream& operator<< (std::ostream& o,
    const hash &t);
25|
26|         private:
27|             int length;
```

```
28|             list<entry>* keys;
29|
30|         public:
31|             hash(int length);
32|             ~hash();
33|
34|             int hashfunction (int key) const;
35|             entry *find (int key, int *iter) const;
36|             hash& insert (entry data);
37|             hash& remove (int key, int *iter);
38|     };
39| }
40|
41| #endif /* HASH_HPP_ */
```

2.2 Заголовочный файл

```
 1| #ifndef HASH_CPP_
 2| #define HASH_CPP_
 3|
 4| #include <iostream>
 5| #include <stdexcept>
 6| #include "HashTable.hpp"
 7| #include "list.hpp"
 8|
 9| namespace ads {
10|     entry::entry (int) {}
11|
12|     entry::entry (int k, int v)
13|     : key (k), value (v) {}
14|
15|     bool operator== (entry A, int B) {
16|         return A.key == B;
17|     }
18|
19|     bool operator== (int A, entry B) {
20|         return A == B.key;
21|     }
22|
23|     bool operator== (entry A, entry B) {
24|         return A.key == B.key;
25|     }
26|
27|     hash::hash (int length)
28|     : length (length) {
29|         if (length < 1)
30|             throw std::invalid_argument ("Invalid length");
31|         keys = new list<entry>[length];
32|     }
33|
34|     hash::~hash () {
35|         delete[] keys;
36|     }
```

```
37|
38|     int hash::hashfunction (int key) const {
39|         return abs (key % length);
40|     }
41|
42|     entry* hash::find (int key, int *iter) const {
43|                                     _node<entry>    *find    =
    keys[hashfunction(key)].find(entry (key, 0), iter);
44|         if (find && find->next)
45|             return &find->next->field;
46|         return nullptr;
47|     }
48|
49|     hash& hash::insert (entry data) {
50|         keys[hashfunction(data.key)].pushStart(data);
51|         return *this;
52|     }
53|
54|     hash& hash::remove (int key, int *iter) {
55|                                     _node<entry>    *find    =
    keys[hashfunction(key)].find(entry (key, 0), iter);
56|         std::cout << "Remove " << key << " ";
57|         if (find) {
58|             keys[hashfunction(key)].remove_next(find);
59|             std::cout << "Success deletion  ";
60|         } else {
61|             std::cout << "There is no key  ";
62|         }
63|         std::cout << "Iterations " << *iter << std::endl;
64|         return *this;
65|     }
66|
67|       std::ostream& operator<< (std::ostream& o, const entry
    &e) {
68|         o << "(" << e.key << " : " << e.value << ")";
69|         return o;
70|     }
71|
72|     std::ostream& operator<< (std::ostream& o, const hash &t)
    {
73|         for (int i = 0; i < t.length; i++)
74|                 o << "[" << i << "] -> " <<   t.keys[i] <<
    std::endl;
75|         return o;
76|     }
77| }
78|
79| #endif /* HASH_CPP_ */
```

3 Пример решения задачи

Пример выполнения программы. Вывод таблицы заполненной входными

данными из файла “I” (См. п. 4.1). Для каждой операции вставки и поиска выводится кол-во совершенных проб (См. п. 4.2).

3.1 Входной файл

```
 1| asmazovec@mobilehost insert ~/dev/sem4.СиАОДвЭВМ.lab5 > cat I
 2| 123 1111
 3| 112 2222
 4| 332 3333
 5| 411 4444
 6| 451 5555
 7| 435 6666
 8| 513 7777
 9| 886 8888
10| 113 9999
11| 551 0000
12| 885 2121
13| 558 3232
14| 567 4343
15| 788 5454
16| 671 6565
17| 777 7676
```

3.2 Выходной файл

```
 1| asmazovec@mobilehost  insert  ~/dev/sem4.СиАОДвЭВМ.lab5  >  ./
    a.out
 2| [0] ->  ####
 3| [1] -> (671 : 6565) -> (551 : 0) -> (451 : 5555) -> (411 :
    4444) ->  ####
 4| [2] -> (332 : 3333) -> (112 : 2222) ->  ####
 5| [3] -> (113 : 9999) -> (513 : 7777) -> (123 : 1111) ->  ####
 6| [4] ->  ####
 7| [5] -> (885 : 2121) -> (435 : 6666) ->  ####
 8| [6] -> (886 : 8888) ->  ####
 9| [7] -> (777 : 7676) -> (567 : 4343) ->  ####
10| [8] -> (788 : 5454) -> (558 : 3232) ->  ####
11| [9] ->  ####
12|
13| 411
14| Find   411 Just (411 : 4444)  Iterations 4
15| Remove 411 Success deletion  Iterations 4
```

4 Вывод

В результате выполнения лабораторной работой мной были изучены принципы работы со структурой данных хеш-таблица.