МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 6.1

по дисциплине

«Структуры и алгоритмы обработки данных»

Тема: Быстрый доступ к данным с помощью хеш-таблиц.

Выполнил студент группы ИКБО-41-23 Гольд Д.В.

Принял старший преподаватель Рысин.М.Л.

СОДЕРЖАНИЕ

ЦЕЛЬ

Освоить приёмы хеширования и эффективного поиска элементов множества.

ХОД РАБОТЫ

ИНДИВИДУАЛЬНЫЙ ВАРИАНТ:

Метод хеширования: Открытая адресация (двойное хеширование)

Структура элемента множества: Читательский абонемент: номер читательского - целое пятизначное число, ФИО, адрес

КОД ПРОГРАММЫ:

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

const int DELETED = -1;//маркер для удаления

//структура - хранения данных читательского абонемента
struct Abonement {
    int number;
    string name;
    string address;
};

//класс для работы с хеш-таблицей (открытая адресация - двойное хеширование)
class HashTable {
private:
    vector<Abonement*> table;//хеш-таблица
    int size;
    int count;
    vector<int> collisionCount;//массив для хранения колва коллизий

    int hash1(int key) {//начальный иднекс
        return key % size;
    }

    int hash2(int key) {//вычисляем сдвиг
        return 1 + (key % (size - 1));
    }

    //увеличение размера таблицы и пересчет хешей
    void rehash() {
        int oldSize = size;
        size *= 2;//увеличиваем размер таблицы в 2 раза
        vector<Abonement*> oldTable = table;

        vector<int> oldCollisions = collisionCount;//сохраняем старый массив коллизий

        table.clear();
        table.resize(size, nullptr);
        collisionCount.clear();
        collisionCount.resize(size, 0);
        count = 0;

        for (int i = 0; i < oldSize; i++) {
            if (oldTable[i] != nullptr && oldTable[i]->number != DELETED) {
                insert(*oldTable[i]);//перезаполняем таблицу
            }
        }
```

```
50          }
51      }
52
53  public:
54      //конструктор
55      HashTable(int initialSize) : size(initialSize), count(0) {
56          table.resize(size, nullptr);
57          collisionCount.resize(size, 0);//иИнициализируем массив коллизий
58      }
59
60      //вставка нового абонемента в таблицу
61      void insert(const Abonement& abonement) {
62          if (count >= size / 2) {
63              rehash();//если таблица заполнена более чем на 50% - увеличиваем размер
64          }
65
66          int index = hash1(abonement.number);
67          int step = hash2(abonement.number);//вычисляем сдвиг
68          bool collided = false;
69
70          //пробирование пока место не найдем
71          while (table[index] != nullptr && table[index]->number != DELETED) {
72              collided = true;
73              collisionCount[index]++; //увеличиваем количество коллизий для текущего индекса
74              index = (index + step) % size;
75          }
76
77          if (table[index] == nullptr || table[index]->number == DELETED) {
78              table[index] = new Abonement(abonement);
79              count++;
80              if (collided) {
81                  collisionCount[index]++;//фиксируем коллизию на конечной позиции
82              }
83          }
84      }
85
86      //поиск абонемента по номеру
87      Abonement* search(int number) {
88          int index = hash1(number);
89          int step = hash2(number);
90
91          while (table[index] != nullptr) {
92              if (table[index]->number == number) {
93                  return table[index];
94              }
95              index = (index + step) % size;
96          }
97          return nullptr;//если не найден
98      }
99
```

```
    //удаление абонемента по номеру (реальное удаление без использования маркера)
    void remove(int number) {
        int index = hash1(number);
        int step = hash2(number);

        while (table[index] != nullptr) {
            if (table[index]->number == number) {
                delete table[index];//удаляем сам элемент
                table[index] = nullptr;//очищаем указатель - чтобы ячейка стала пустой
                count--;//уменьшаем количество элементов
                cout << "abonement successfully deleted\n";
                return;
            }
            index = (index + step) % size;
        }
        cout << "abonement not found\n";
    }

    //вывод всех абонементов
    void display() {
        for (int i = 0; i < size; i++) {
            if (table[i] != nullptr && table[i]->number != DELETED) {
                cout << "abonement: " << table[i]->number
                    << ", name: " << table[i]->name
                    << ", address: " << table[i]->address
                    << ", collisions: " << collisionCount[i] << endl;//колво коллизий для ячейк
            }
        }
    }

    void displayCollisions() {
        cout << "\ncollision information:\n";
        for (int i = 0; i < size; i++) {
            if (collisionCount[i] > 0) {
                cout << "index " << i << ": " << collisionCount[i] << " collisions\n";
            }
        }
    }

    ~HashTable() {
        for (int i = 0; i < size; i++) {
            if (table[i] != nullptr) {
                delete table[i];
            }
        }
    }
};
```

```
143     void displayCollisions() {
144         cout << "\ncollision information:\n";
145         for (int i = 0; i < size; i++) {
146             if (collisionCount[i] > 0) {
147                 cout << "index " << i << ": " << collisionCount[i] << " collisions\n";
148             }
149         }
150     }
151
152     ~HashTable() {
153         for (int i = 0; i < size; i++) {
154             if (table[i] != nullptr) {
155                 delete table[i];
156             }
157         }
158     }
159 };
160
161 void autoFill(HashTable& table) {
162     Abonement abonements[] = {
163         {11111, "name a first", "city A"},
164         {22222, "name b second", "city B"},
165         {33333, "name c three", "city C"},
166         {44444, "name d four", "city D"},
167         {55555, "name e five", "city E"}
168     };
```

```
146  void autoFill(HashTable& table) {
147      Abonement abonements[] = {
148          {11111, "name a first", "city A"},
149          {22222, "name b second", "city B"},
150          {33333, "name c three", "city C"},
151          {44444, "name d four", "city D"},
152          {55555, "name e five", "city E"}
153      };
154
155      for (const auto& abonement : abonements) {
156          table.insert(abonement);
157      }
158  }
159
160  void menu(HashTable& table) {
161      int choice;
162      do {
163          cout << "\n1 - insert abonement\n2 - search abonement\n3 - remove abonement";
164          cout << "\n4 - display all abonements\n5 - display collision info\n6 - exit\n";
165          cout << "choose number: ";
166          cin >> choice;
167
168          if (choice == 1) {
169              Abonement abonement;
170              cout << "enter abonement number: ";
171              cin >> abonement.number;
172              cout << "enter name: ";
173              cin.ignore();
174              getline(cin, abonement.name);
175              cout << "enter address: ";
176              getline(cin, abonement.address);
177              table.insert(abonement);
178          }
179          else if (choice == 2) {
180              int number;
181              cout << "enter abonement number to search: ";
182              cin >> number;
183              Abonement* result = table.search(number);
184              if (result) {
185                  cout << "abonement found: " << result->number << ", name: " << result->name
186                      << ", address: " << result->address << endl;
187              }
188              else {
189                  cout << "abonement not found.\n";
190              }
191          }
192          else if (choice == 3) {
193              int number;
194              cout << "enter abonement number to remove: ";
195              cin >> number;
```

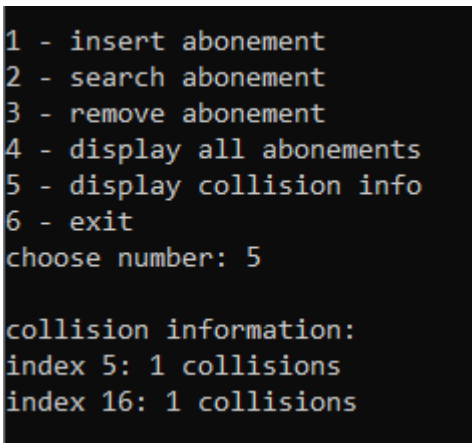
```
192         else if (choice == 3) {
193             int number;
194             cout << "enter abonement number to remove: ";
195             cin >> number;
196             table.remove(number);
197         }
198         else if (choice == 4) {
199             table.display();
200         }
201         else if (choice == 5) {
202             table.displayCollisions();
203         }
204     } while (choice != 6);
205 }
206
207 int main() {
208     HashTable table(11);
209     autoFill(table);
210     menu(table);
211     return 0;
212 }
213
```

ТЕСТИРОВАНИЕ

```
1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 1
enter abonement number: 88888
enter name: EIGHTEIGHT
enter address: 8address address8

1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 1
enter abonement number: 99999
enter name: NINENINE
enter address: 9NINE nine9

1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 1
enter abonement number: 23744
enter name: dshfjsdfk
enter address: ksdjfldsjflksdjflkd dlfjsdkfjksdlf

1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 4
abonement: 11111, name: name a first, address: city A, collisions: 0
abonement: 22222, name: name b second, address: city B, collisions: 0
abonement: 33333, name: name c three, address: city C, collisions: 0
abonement: 44444, name: name d four, address: city D, collisions: 0
abonement: 55555, name: name e five, address: city E, collisions: 0
abonement: 23744, name: dshfjsdfk, address: ksdjfldsjflksdjflkd dlfjsdkfjksdlf, collisions: 0
abonement: 88888, name: EIGHTEIGHT, address: 8address address8, collisions: 0
abonement: 99999, name: NINENINE, address: 9NINE nine9, collisions: 0
```

```
1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 2
enter abonement number to search: 11111
abonement found: 11111, name: name a first, address: city A

1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 1
enter abonement number: 55555
enter name: s
enter address: s

1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 2
enter abonement number to search: 99999
abonement found: 99999, name: NINENINE, address: 9NINE nine9
```

```
1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 3
enter abonement number to remove: 88888
abonement marked as deleted

1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 3
enter abonement number to remove: 11111
abonement marked as deleted

1 - insert abonement
2 - search abonement
3 - remove abonement
4 - display all abonements
5 - display collision info
6 - exit
choose number: 4
abonement: 22222, name: name b second, address: city B, collisions: 0
abonement: 33333, name: name c three, address: city C, collisions: 0
abonement: 44444, name: name d four, address: city D, collisions: 0
abonement: 55555, name: name e five, address: city E, collisions: 1
abonement: 23744, name: dshfjsdfk, address: ksdjfldsjflksdjflkd dlfjsdkfjksdlf, collisions: 0
abonement: 99999, name: NINENINE, address: 9NINE nine9, collisions: 0
abonement: 55555, name: s, address: s, collisions: 1
```



ВЫВОД

Были освоены приёмы хеширования и эффективного поиска элементов в хеш-таблице.