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 №5
по дисциплине
«Структуры и алгоритмы обработки данных»

Тема. Алгоритмы поиска

Выполнил студент группы ИКБО-43-23 Кощеев М. И.

Принял старший преподаватель Рысин М.Л.

Москва 2024

СОДЕРЖАНИЕ

1 ЦЕЛЬ

Освоить приёмы хеширования и эффективного поиска элементов множества, освоить приёмы реализации алгоритмов поиска образца в тексте.

2 ХОД РАБОТЫ (6.1)

Разработайте приложения в соответствии с заданиями в индивидуальном варианте (Открытая адресация (линейное пробирование)).

Алгоритм решения (main):

1. Создание объекта ht класса HashTable
2. Вызов функции main, пока программа работает

Код (main):

```
156 int main() {
157     HashTable ht;
158     while (true) {
159         ui([&]ht);
160     }
161     return 0;
162 }
```

Код (класс HashTable):

```
18  class HashTable {
19      private:
20          vector<Product> table;
21          int size;
22          int capacity;
23
24          int hashFunction(int key) {
25              return key % capacity;
26          }
27
28          void rehash() {
29              if (static_cast<double>(size) / capacity >= LOAD_FACTOR_THRESHOLD) {
30                  int oldCapacity = capacity;
31                  capacity *= 2;
32                  vector<Product> newTable(capacity);
33
34                  for (const auto& product : table) {
35                      if (product.code != -1) {
36                          int newIndex = hashFunction(product.code);
37                          while (newTable[newIndex].code != -1) {
38                              newIndex = (newIndex + 1) % capacity;
39                          }
40                          newTable[newIndex] = product;
41                      }
42                  }
43                  table = newTable;
44                  cout << "The table has been enlarged and rehashed. New size: " << capacity << endl;
45              }
46          }
47
```

```
    public:
        HashTable(int cap = INITIAL_SIZE) : size(0), capacity(cap) {
            table.resize(capacity);
        }

        void insertProduct(int code, const string& name, double price) {
            if (size >= capacity) {
                rehash();
            }
            int index = hashFunction(code);
            while (table[index].code != -1) {
                index = (index + 1) % capacity;
            }
            table[index] = Product(code, name, price);
            size++;
            cout << "New product has been added: code " << code << ", name: " << name << ", price: " << price << endl;
            rehash();
        }

        void searchProduct(int code) {
            int index = hashFunction(code);
            int startIndex = index;
            while (table[index].code != code) {
                index = (index + 1) % capacity;
                if (index == startIndex || table[index].code == -1) {
                    cout << "Product with code " << code << " has not been found." << endl;
                    return;
                }
            }
            cout << "Product has been found: code " << code << ", name: " << table[index].name << ", price: " << table[index].price << endl;
        }

        void deleteProduct(int code) {
            int index = hashFunction(code);
            int startIndex = index;
            while (table[index].code != code) {
                index = (index + 1) % capacity;
                if (index == startIndex || table[index].code == -1) {
                    cout << "Product with code " << code << " has not been found." << endl;
                    return;
                }
            }
            table[index] = Product();
            size--;
            cout << "Product with code " << code << " has been deleted." << endl;
        }

        void displayProducts() {
            cout << "Products:" << endl;
            for (const auto& product : table) {
                if (product.code != -1) {
                    cout << "Code: " << product.code << ", name: " << product.name << ", price: " << product.price << endl;
                }
            }
        }
};
```

Код (структура Product)

```
struct Product {
    int code;
    string name;
    double price;

    Product(int c = -1, string n = "", double p = 0.0) : code(c), name(n), price(p) {}
};
```

Алгоритм решения (ui):

3. Инициализация переменной choice типа int
4. Вывод возможных команд
5. Ввод значения choice (выбор команды)
    a. 1: вывод товаров
    b. 2: ввод кода товара, поиск товара по коду
    c. 3: ввод данных о товаре, запись товара
    d. 4: ввод кода товара, удаление товара по коду
    e. 5: выход из программы
    f. Остальное: вывод об ошибке

Код:

```
105  void ui(HashTable& ht) {
106      int choice;
107      cout << endl;
108      cout << "Enter the command:" << endl;
109      cout << "1. Show the list of products" << endl;
110      cout << "2. Find a product by code" << endl;
111      cout << "3. Add a product" << endl;
112      cout << "4. Delete a product" << endl;
113      cout << "5. Exit" << endl;
114      cin >> choice;
115      cout << endl;
116
117      switch (choice) {
118          case 1:
119              ht.displayProducts();
120              break;
121          case 2: {
122              int code;
123              cout << "Enter the code of the product: ";
124              cin >> code;
125              ht.searchProduct(code);
126              break;
127          }
128          case 3: {
129              int code;
130              string name;
131              double price;
132              cout << "Enter the product code: ";
133              cin >> code;
134              cout << "Enter the product name: ";
135              cin.ignore();
136              getline([&]cin, [&]name);
137              cout << "Enter the product price: ";
138              cin >> price;
139              ht.insertProduct(code, name, price);
140              break;
141          }
142          case 4: {
143              int code;
144              cout << "Enter the code for deleting: ";
145              cin >> code;
146              ht.deleteProduct(code);
147              break;
148          }
149          case 5:
150              exit(Code: 0);
151          default:
152              cout << "Error!" << endl;
153      }
154  }
155
```

Алгоритм решения (displayProducts):

1. Перебор всех товаров векторе table
2. Если код товара не равен -1, вывод информации о товаре (код, название, цена)

Код:

```
95          void displayProducts() {
96              cout << "Products:" << endl;
97              for (const auto& product : table) {
98                  if (product.code != -1) {
99                      cout << "Code: " << product.code << ", name: " << product.name << ", price: " << product.price << endl;
100                 }
101             }
102         }
103 };
```

Демонстрация работы программы:

```
Enter the command:
1. Show the list of products
2. Find a product by code
3. Add a product
4. Delete a product
5. Exit
1

Products:
Code: 123456, name: Tea, price: 120
Code: 123457, name: Meat, price: 1200
```

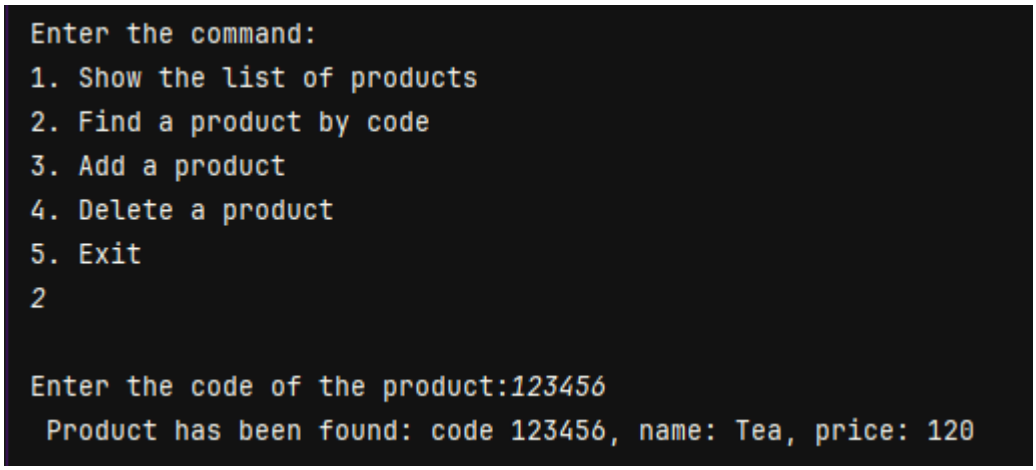
Алгоритм решения (searchProduct):

1. Вычисления индекса в таблице с помощью хеш-функции
2. Вектор таблицы обходит по кругу, начиная с вычисленного индекса, до тех пор, пока не найдется товар с нужным кодом или не будет возвращено к началу
3. Если товар с заданным кодом существует, выводятся его данные (код, название и цена). Если товар с данным кодом не найден, выводится сообщение об отсутствии товара

Код:

```
67      void searchProduct(int code) {
68          int index = hashFunction(code);
69          int startIndex = index;
70          while (table[index].code != code) {
71              index = (index + 1) % capacity;
72              if (index == startIndex || table[index].code == -1) {
73                  cout << "Product with code " << code << " has not been found." << endl;
74                  return;
75              }
76          }
77          cout << "Product has been found: code " << code << ", name: " << table[index].name << ", price: " << table[index].price << endl;
78      }
79
```

Демонстрация работы программы:



Алгоритм решения (insertProduct)

1. Если таблица заполнена, выполняется операция рехеширования для увеличения ее емкости
2. Для нового товара вычисляется индекс с помощью хеш-функции
3. Если место уже занято другим товаром, выполняется поиск следующего свободного индекса
4. Новый товар добавляется на найденное свободное место, обновляется счетчик
5. После добавления товара выполняется проверка, не нужно ли увеличить таблицу снова

Алгоритм решения (rehash)

1. Если отношение текущего количества элементов к емкости таблицы превышает определенный порог начинается процесс рехеширования
2. Емкость таблицы удваивается для предотвращения дальнейшего переполнения.
3. Создается новая таблица с обновленной емкостью
4. Все элементы из старой таблицы переносятся в новую. Для каждого элемента повторно вычисляется новый индекс с помощью хеш-функции. Если индекс уже занят, используется линейное пробирование (проверка последующих индексов), чтобы найти пустое место.
5. После переноса всех элементов новая таблица заменяет старую.
6. Программа сообщает, что таблица была увеличена и повторно хеширована.

Код:

```
28          void rehash() {
29              if (static_cast<double>(size) / capacity >= LOAD_FACTOR_THRESHOLD) {
30                  int oldCapacity = capacity;
31                  capacity *= 2;
32                  vector<Product> newTable(capacity);
33
34                  for (const auto& product : table) {
35                      if (product.code != -1) {
36                          int newIndex = hashFunction(product.code);
37                          while (newTable[newIndex].code != -1) {
38                              newIndex = (newIndex + 1) % capacity;
39                          }
40                          newTable[newIndex] = product;
41                      }
42                  }
43                  table = newTable;
44                  cout << "The table has been enlarged and rehashed. New size: " << capacity << endl;
45              }
46          }
47
```

```
53          void insertProduct(int code, const string& name, double price) {
54              if (size >= capacity) {
55                  rehash();
56              }
57              int index = hashFunction(code);
58              while (table[index].code != -1) {
59                  index = (index + 1) % capacity;
60              }
61              table[index] = Product(code, name, price);
62              size++;
63              cout << "New product has been added: code " << code << ", name: " << name << ", price: " << price << endl;
64              rehash();
65          }
66
```

Демонстрация работы программы:

```
Enter the command:
1. Show the list of products
2. Find a product by code
3. Add a product
4. Delete a product
5. Exit
3

Enter the product code:123456
 Enter the product name:Tea
 Enter the product price:120
 New product has been added: code 123456, name:
Tea, price: 120
```

Алгоритм решения (deleteProduct)

1. С помощью хеш-функции вычисляется индекс, где может находиться товар
2. Если на вычисленном индексе не найден товар с заданным кодом, выполняется линейное пробирование до нахождения нужного товара или возвращения к начальному индексу
3. Если товар не найден после полного обхода, выводится сообщение об отсутствии товара
4. Если товар найден, он удаляется из таблицы (код устанавливается в -1)
5. После удаления размер таблицы уменьшается на 1

Код программы:

```
80          void deleteProduct(int code) {
81              int index = hashFunction(code);
82              int startIndex = index;
83              while (table[index].code != code) {
84                  index = (index + 1) % capacity;
85                  if (index == startIndex || table[index].code == -1) {
86                      cout << "Product with code " << code << " has not been found." << endl;
87                      return;
88                  }
89              }
90              table[index] = Product();
91              size--;
92              cout << "Product with code " << code << " has been deleted." << endl;
93          }
94
```

Демонстрация работы программы:

```
Enter the command:
1. Show the list of products
2. Find a product by code
3. Add a product
4. Delete a product
5. Exit
4

Enter the code for deleting:123456
 Product with code 123456 has been deleted.

Enter the command:
1. Show the list of products
2. Find a product by code
3. Add a product
4. Delete a product
5. Exit
1

Products:
Code: 123457, name: Meat, price: 1200

Enter the command:
1. Show the list of products
2. Find a product by code
3. Add a product
4. Delete a product
5. Exit
```

Вывод по заданию:

В ходе разработки и реализации приложения для управления записями с использованием хеш-таблицы была успешно создана система, обеспечивающая эффективный доступ к данным

3 ХОД РАБОТЫ (6.2)

3.1 Задание 1

Дан текст, состоящий из слов, разделенных знаками препинания. Сформировать массив из слов, в которых заданная подстрока размещается в конце слова.

Алгоритм решения:

1. Удаление всех знаков пунктуации из исходного текста для корректного разбиения на слова
2. Преобразование очищенного текста в вектор строк, где каждая строка — это отдельное слово
3. Оставление только тех слов, которые оканчиваются на заданную подстроку
4. Вывод отфильтрованных слов на экран

Код:

```
44  void task1() {
45      cout<<"task 1"<<endl;
46      cin.get();
47      cout<<endl;
48      string text;
49      string substring;
50      cout<<"Enter text: "<<endl;
51      getline([&]cin,[&]text);
52      cout<<"Your text: "<<text<<endl;
53      cout<<"Enter substring:"<<endl;
54      cin>>substring;
55      cout<<"Your substring: "<<substring<<endl;
56      find_substring_in_words(text,substring);
57  }
```

```cpp
void find_substring_in_words(string input_text, string input_substring) {
    string clean_text;
    vector<string> words;
    string word;
    for (char symbol1 : input_text) {
        if (!ispunct(symbol1)) {
            clean_text += symbol1;
        }
    }
    for (char symbol2 : clean_text) {
        if (symbol2 != ' ') {
            word += symbol2;
        } else {
            if (!word.empty()) {
                words.push_back(word);
                word.clear();
            }
        }
    }
    if (!word.empty()) {
        words.push_back(word);
    }
    for (auto i = words.begin(); i != words.end();) {
        if (i->size() >= input_substring.size() && i->compare(pos: i->size() - input_substring.size(), n: input_substring.size(), input_substring) == 0) {
            ++i;
        }
        else {
            i = words.erase(i);
        }
    }
    for (const auto& i(const string&) : words) {
        cout << i << endl;
    }
}
```

Демонстрация работы программы:

```
Enter text:
Born and raised in Boston, Massachusetts, USA, the Scout is a fast-running scrapper with a baseball bat and snarky "in-your-face" attitude
Your text: Born and raised in Boston, Massachusetts, USA, the Scout is a fast-running scrapper with a baseball bat and s
narky "in-your-face" attitude
Enter substring:
ed
Your substring: ed
raised
```

3.2 Задание 2

В текстовом файле хранятся входные данные: на первой сроке – подстрока (образец) длиной не более 17 символов для поиска в тексте; со второй строки – текст (строка), в котором осуществляется поиск образца. Строка, в которой надо искать, неограниченна по длине. Применяя алгоритм Рабина-Карпа определить количество вхождений в текст заданного образца

Алгоритм решения:

1. Вычисление длины шаблона и текста, а также задание базовых констант (основание системы счисления base и простое число для модуляции mod)
2. Рассчитывается коэффициент h, который используется для сдвига при обновлении хеша строки
3. Для первых m символов шаблона и текста вычисляется хеш с использованием базовой системы счисления

4. Для каждого окна длины m в тексте хеш текущего окна сравнивается с хешем шаблона. Если хеши совпадают, то выполняется дополнительная проверка на строковое равенство
5. После каждого сдвига окна хеш текста обновляется на основе предыдущего значения хеша, сдвигая буквы по одной с использованием коэффициента h
6. Вывод количества совпадений

Код:

```
int task2() {
    cout<<"task 2"<<endl;
    cin.get();
    string pattern;
    string text;
    cout << "Enter pattern: "<<endl;
    getline([&]cin, [&]pattern);

    if (pattern.length() > 17) {
        cout << "Error: The patern must contain no more than 17 characters" << endl;
        return 1;
    }
    cout << "Enter text: "<<endl;;
    getline([&]cin, [&]text);
    ofstream outputFile(s:"text.txt");
    if (outputFile.is_open()) {
        outputFile << pattern <<endl;
        outputFile << text <<endl;
        outputFile.close();
        cout << "The data has been successfully written to the file text.txt" << endl;
    }
    int occurrences = karp(pattern, text);
    cout << "Number of occurrences: " << occurrences << endl;
    return 0;
}

```

```cpp
int karp(const string &pattern, const string &text) {
    int m = pattern.length();
    int n = text.length();

    const int base = 256;
    const int mod = 101;

    int patternHash = 0;
    int textHash = 0;
    int h = 1;

    for (int i = 0; i < m - 1; i++) {
        h = (h * base) % mod;
    }
    for (int i = 0; i < m; i++) {
        patternHash = (base * patternHash + pattern[i]) % mod;
        cout<<patternHash<<endl;
        textHash = (base * textHash + text[i]) % mod;
    }
    int count = 0;

    for (int i = 0; i <= n - m; i++) {
        if (patternHash == textHash) {
            if (text.substr(pos: i, n: m) == pattern) {
                count++;
            }
        }
        if (i < n - m) {
            textHash = (base * (textHash - text[i] * h) + text[i + m]) % mod;
            if (textHash < 0) {
                textHash += mod;
            }
        }
    }
    return count;
}
```

Демонстрация работы программы:

```
Enter the task:
2
task 2
Enter pattern:
Mann
Enter text:
Mann Co., stylized as MANN CO., is a multinational division of TF Industries with headq
The data has been successfully written to the file text.txt
Number of occurrences: 12
```

Данный текст:

Mann Co., stylized as MANN CO., is a multinational division of TF Industries with headquarters in the Badlands, near Teufort, New Mexico. In the Badlands area alone, the company operates over 300 plants, office buildings, warehouses and outlet malls. The company was originally incorporated in the United States as Mann & Sons Munitions Concern by the arms-dealer Zepheniah Mann, who co-managed the firm with his brother, Silas Mann. Upon his death in 1850, Zepheniah Mann bequeathed ownership of Mann Co. to close friend and hunter Barnabus Hale. The company was subsequently owned and operated by three generations of Hales. The organization was recently re-branded as Gray Mann Co. by the most recent CEO, Gray Mann, previously the CEO of Gray Gravel. Mr. Mann was recently killed in his offices in Australia, believed murdered by a former employee of Builders League United. Corporate succession has not been resolved. Gray Mann is survived by his daughter and immediate predecessor at Mann Co., Miss Olivia Mann. Additionally, Saxton Hale has expressed interest in returning to the position.

4 ВЫВОДЫ

В ходе разработки и реализации приложения для управления записями с использованием хеш-таблицы была успешно создана система, обеспечивающая эффективный доступ к данным. Также был изучен алгоритм Рабина-Карпа и нахождение слова по подстроке

5 ИСПОЛЬЗУЕМЫЙ ИСТОЧНИК

1. М.Л. РЫСИН, М.В. САРТАКОВ, М.Б. ТУМАНОВА Учебно-методическое пособие СиАОД часть 2