

**Российский университет транспорта (МИИТ)**  
**Институт транспортной техники и систем управления**  
**Кафедра «Управление и защита информации»**

**Отчет**  
**по практическому заданию**  
**по теме «Структуры данных»**  
**по дисциплине «Системы управления базами данных»**

Выполнили:

Студенты группы ТКИ-441

Чекан Ф.С.

Комаричев Г.Ю.

Проверил:

Доцент кафедры УиЗи, к.т.н., с.н.с.

Васильева М.А.

Москва 2023

Оглавление	
Задание.....	3
1. UML диаграмма .....	4
2. Текст программы на языке C++ .....	4
2.1 Код файла HashMap.h.....	4
2.2 Код файла HashMap.cpp .....	6
2.3 Код файла main.cpp .....	8
2.4 Код файла HashMapTest.cpp .....	9
3. Результат работы программы .....	10
Заключение.....	12

## **Задание**

Разработать структуру данных на языке программирования C++ в ООП парадигме. В нашем случае структура данных – это словарь (map), основные операции которого будут добавление элемента, обновления значения по ключу, получение значения по ключу и удаление пары (ключ, значения). Структура способна обрабатывать любые типы данных (template).

## 1. UML диаграмма

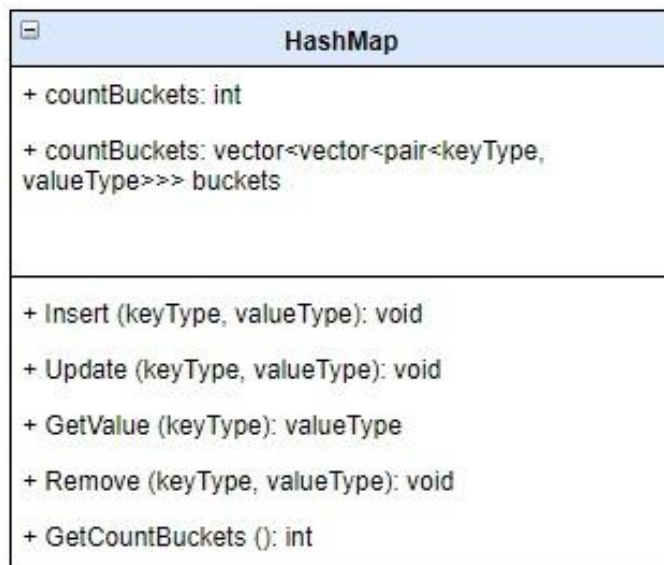


Рисунок 1 – UML диаграмма классов HashMap

## 2. Текст программы на языке C++

### 2.1 Код файла HashMap.h

```
#include <vector>
#include <iostream>

using namespace std;
/**
 * @brief Класс HashMap.
 * @tparam keyType тип данных ключа.
 * @tparam valueType тип данных значения.
 */
template <typename keyType, typename valueType>
class HashMap {
public:
    /**
     * @brief Конструктор по умолчанию.
     */
    HashMap();

    /**
     * @brief Метод вставки пары (Ключ, значение).
     * @param key Ключ.
```

```

* @param value значение.
*/
void Insert(const keyType &key, const valueType &value);
/**
* @brief Метод обновление значения по ключу.
* @param key Ключ.
* @param value Новое значение.
*/
    void Update(const keyType &key, const valueType &value);
/**
* @brief Метод получения значения по ключу.
* @param key Ключ.
* @return значение.
*/
    valueType GetValue(const keyType &key);
/**
* @brief Метод удаления пары (ключ, значения) по ключу.
* @param key Ключ.
*/
    void Remove(const keyType &key);
/**
* @brief Метод получения размера HashMap.
* @return размер HashMap.
*/
    int GetCountBuckets();
/**
* @brief размер HashMap.
*/
    int countBuckets;
/**
* @brief Массив пар (ключ, значение).
*/
    vector<vector<pair<keyType, valueType>>> buckets;
};

```

## 2.2 Код файла HashMap.cpp

```
#include <vector>
#include "HashMap.h"

template <typename keyType, typename valueType>
HashMap<keyType, valueType>::HashMap() {
    countBuckets = 32;
    buckets.resize(countBuckets);
}

template <typename keyType, typename valueType>
void HashMap<keyType, valueType>::Insert(const keyType &key, const valueType &value) {
    int hashIndex = hash<keyType>{}(key) % countBuckets;
    for (auto &pair: buckets[hashIndex]) {
        if (pair.first == key) {
            throw out_of_range("Pair already exists, choose update to set new value.
\n");
        }
    }
    buckets[hashIndex].push_back(make_pair(key, value));
}

template <typename keyType, typename valueType>
valueType HashMap<keyType, valueType>::GetValue(const keyType &key) {
    int hashIndex = hash<keyType>{}(key) % countBuckets;
    if (buckets[hashIndex].empty())
    {
        throw out_of_range("Pair is not found");
    }
    for (auto &pair : buckets[hashIndex]) {
        if (pair.first == key) {
            return pair.second;
        }
    }
}

template <typename keyType, typename valueType>
void HashMap<keyType, valueType>::Update(const keyType &key, const valueType &value) {
```

```

int hashIndex = hash<keyType>{}(key) % countBuckets;
if (buckets[hashIndex].empty())
{
    throw out_of_range("Pair is not found");
}
for (auto &pair: buckets[hashIndex]) {
    if (pair.first == key) {
        pair.second = value;
        return;
    }
}
}

```

```

template <typename keyType, typename valueType>
void HashMap<keyType, valueType>::Remove(const keyType &key) {
    int hashIndex = hash<keyType>{}(key) % countBuckets;
    for (auto it = buckets[hashIndex].begin(); it != buckets[hashIndex].end(); ++it){
        if (it->first == key) {
            buckets[hashIndex].erase(it);
            return;
        }
    }
}
}

```

```

template<typename keyType, typename valueType>
int HashMap<keyType, valueType>::GetCountBuckets() {
    return this->countBuckets;
}

```

## 2.3 Код файла main.cpp

```
#include <iostream>
#include <vector>
#include "HashMap.h"
#include "HashMap.cpp"

using namespace std;

int main() {
    HashMap<string, int> car;

    car.Insert("Kia", 123);
    car.Insert("BMW", 420);
    car.Insert("Tesla", 1020);

    cout << "Value of Kia: " << car.GetValue("Kia") << endl;
    cout << "Value of BMW: " << car.GetValue("BMW") << endl;
    cout << "Value of Tesla: " << car.GetValue("Tesla") << endl;

    car.Insert("Ford", 100);
    cout << "Value of Ford: " << car.GetValue("Ford") << endl;

    cout << "Remove value of BMW" << endl;

    car.Remove("BMW");

    cout << "Remove successful" << endl;

    cout << "Updating value of Kia: " << endl;

    car.Update("Kia", 130);

    cout << "Update successful" << endl;

    cout << "Value of Kia after update: " << car.GetValue("Kia") << endl;

    cout << "Value of BMW after remove" << car.GetValue("BMW") << endl;

    return 0;
}
```



## 2.4 Код файла HashMapTest.cpp

```
#include <gtest/gtest.h>
#include "../Task2/Common/HashMap.h"
#include "../Task2/Common/HashMap.cpp"

TEST(HashMapTests, InsertTest) {
    HashMap<int, int> test;
    test.Insert(1, 2);
    EXPECT_EQ(test.GetValue(1), 2);
}

TEST(HashMapTests, GetValueTest) {
    HashMap<int, int> test;
    test.Insert(1, 2);
    int testValue = test.GetValue(1);
    EXPECT_EQ(testValue, 2);
}

TEST(HashMapTests, UpdateTest) {
    HashMap<int, int> test;
    test.Insert(1, 2);
    test.Update(1, 5);
    EXPECT_EQ(test.GetValue(1), 5);
}

TEST(HashMapTests, RemoveTest) {
    HashMap<int, int> test;
    test.Insert(1, 2);
    test.Insert(3, 4);
    test.Remove(1);
    EXPECT_THROW(test.GetValue(1), out_of_range);
}

TEST(HashMapTests, CreateTest) {
    HashMap<int, int> test;
    EXPECT_EQ(test.GetCountBuckets(), 32);
}

TEST(HashMapTests, CopyConstructorTest) {
    HashMap<int, int> test;
    test.Insert(1, 2);

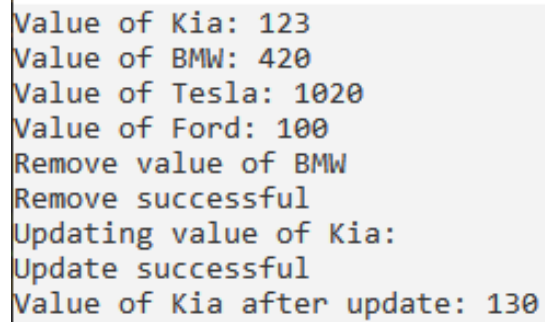
    HashMap<int, int> copy = HashMap<int, int>(test);

    EXPECT_EQ(copy.GetValue(1), 2);
}

TEST(HashMapTests, OperatorTest) {
    HashMap<int, int> test;
    test.Insert(1, 2);
}
```

```
HashMap<int, int> test2 = test;  
  
EXPECT_EQ(test2.GetValue(1), 2);  
}
```

### 3. Результат работы программы



```
Value of Kia: 123  
Value of BMW: 420  
Value of Tesla: 1020  
Value of Ford: 100  
Remove value of BMW  
Remove successful  
Updating value of Kia:  
Update successful  
Value of Kia after update: 130
```

Рисунок 2 – Результат отладки программы

```
CMakeLists.txt testlib\CMakeLists.txt HashMapTest.cpp x
1 #include <gtest/gtest.h>
2 #include "../Task2/Common/HashMap.h"
3 #include "../Task2/Common/HashMap.cpp"
4
5 TEST(HashMapTests, InsertTest) {
6     HashMap<int, int> test;
7     test.Insert( key: 1, value: 2);
8     EXPECT_EQ(test.GetValue( key: 1), 2);
9 }
10
11 TEST(HashMapTests, GetValueTest) {
12     HashMap<int, int> test;
13     test.Insert( key: 1, value: 2);
14     int testValue = test.GetValue( key: 1);
15     EXPECT_EQ(testValue, 2);
16 }
17
18 TEST(HashMapTests, UpdateTest) {
19     HashMap<int, int> test;
20     test.Insert( key: 1, value: 2);
21     test.Update( key: 1, value: 5);
22     EXPECT_EQ(test.GetValue( key: 1), 5);
23 }
24
25 TEST(HashMapTests, RemoveTest) {
26     HashMap<int, int> test;
27     test.Insert( key: 1, value: 2);
28     test.Insert( key: 3, value: 4);
29     test.Remove( key: 1);
30     EXPECT_THROW(test.GetValue( key: 1), out_of_range);
31 }
32
33 TEST(HashMapTests, CreateTest) {
34     HashMap<int, int> test;
35     EXPECT_EQ(test.GetCountBuckets(), 32);
36 }
37
38 ::TestBody
39
40 0ms ✓ Tests passed: 7 of 7 tests – 0 ms
```

Рисунок 3 – Результат выполнения тестов для HashMap.h

## **Заключение**

В результате выполнения практического задания была разработана структура данных - словарь (map). Для обеспечения обработки различных типов данных была использована техника шаблонов, что позволило использовать данную структуру с любым типом данных.