Дисциплина: «Теория алгоритмов»

**Лабораторная работа № 3**

# Хеш-функциии и хеш-таблицы. Применение хеширования.

### Цель работы

Изучение методов построения таблиц с постоянным временем доступа к элементам. Освоение технологии реализации хеш-таблиц и их практического применения.

**Краткие теоретические сведения**

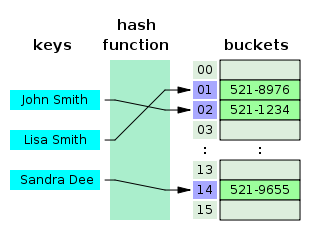
**Хеш-таблицей** называется структура данных, предназначенная для реализации ассоциативного массива, такого в котором адресация реализуется посредством хеш-функции. **Хеш-функция** – это функция, преобразующая ключ **key** в некоторый индекс **i** равный **h(key)**, где h(key) – **хеш-код** (хеш-сумма, хеш) key. Весь процесс получения индексов хеш-таблицы называется **хешированием**.

Хеш-таблицы представляют собой очень практичный способ реализации словарей.

**Принципы хеширования:**

1) Сначала хеш-функция соотносит каждый ключ с большим целым числом. В результате получаются уникальные идентификационные числа, но они будут настолько большими, что очень скоро превысят количество ячеек в хеш-таблице (m) (Таблица 1).

2) Полученное большое целое число k уменьшают до целого числа в диапазоне от 0 до m-1, для чего выполняется операция получения остатка от деления на m. **h(k) = k mod m**. В идеале размер хеш-таблицы должен быть большим простым числом не слишком близким к 2i-1. В результате получается простое целое число, которое служит индексом,идентифицирующим номер блока в хэш-таблице (bucket) (Рисунок 1).

Рисунок 1 — «внутреннее устройство» хэш-таблицы, хранящей пары «ключ, значение»

Пример реализации хэш-функции:

**int hashfunc(int integer\_key) {**

**return integer\_key % HASHTABLESIZE;**

**}**

Хеш-функции могут быть различны, но должны соблюдаться следующие свойства хэш-функции:

1) она должна быть последовательной (т. е. для одних и тех же входных данных давать одно и то же значение)

2)разным ключам должны соответствовать разные числа (правда, существуют коллизии, но об этом ниже).

Таблица 1 - 1й этап: приведение ключа к форме большого целого числа перед хешированием

|  |  |
| --- | --- |
| Пример ключа | Метод приведения к большому простому числу |
| Натуральный многоразрядный ключ | уменьшить разрядность ключей до интервала размера хеш-таблицы.  Пример: **Метод выбора цифр** формирует новое значение как комбинацию отдельных цифр ключа k.  Например, из 12-значного ключа можно выбрать первую, шестую и одиннадцатую цифру. Запись этих цифр в позиционной десятичной системе и будет давать трёхзначное натуральное число, которое будет превращено в индекс с помощью хеш-функции.  Для уменьшения вероятности коллизий значения выбираемых цифр можно модифицировать суммой пропущенных цифр.  Например:  для k 1 = 1**2**3456**7**8912**3** k = (1 + **2**) mod 10, (3 + 4 + 5 + 6 + **7**) mod 10,(8 + 9 + 1 + 2 + **3**) mod 10 = 353,  для k 2 = 4**2**5426**7**5432**3** k = (4 + **2**) mod 10, (5 + 4 + 2 + 6 + **7**) mod  10, (5 + 4 + 3 + 2 + **3**) mod 10 = 647 |
| Натуральный многоразрядный ключ | **Метод «середины квадрата».** Значение ключа возводится в квадрат и из середины полученного числа выбираются нужное количество цифр.  Например, пусть k = 134295445 и квадрат k 2 = 18035266**574**60711401.  Можно выбрать три цифры из середины k 2 и сформировать число  k = 574 для функции хеширования. |
| Натуральный многоразрядный ключ | **Метод свёртки** выделяет в многоразрядном натуральном ключе k  группы по n цифр и складывает группы цифр по модулю 10 n . Например,для группы размером n = 3 ключ k 1 = 123342954451 можно преобразовать по схеме:  k  = (123 + 342 + 954 + 451) mod 10 3 = 870.  Если другой ключ k 2 = 123342451954 имеет те же группы цифр в  ином порядке, то после свёртки получится такое же число:  k  = (123 + 342 + 451+ 954) mod 10 3 = 870.  Чтобы избежать коллизий ключей при свёртке, можно модифициро-  вать значения групп с учётом их порядкового номера. Например, с помощью операции «исключающее ИЛИ» – xor:  для k 1 = 123342954451 k = (123 xor 1 + 342 xor 2 + 954 xor 3 + 451  xor 4) mod 10 3 = (122 + 340 + 953 + 455) mod 10 3 = 870,  для k 2 = 123342451954 k  = (123 xor 1 + 342 xor 2 +451xor 3 + 954  xor 4) mod 10 3 = (122 + 340 + 448 + 958) mod 10 3 = 868. |
| Строка символов произвольной длины | **Конкатенация битовых образов чисел.**  Например, если строка включает только заглавные буквы латинского алфавита, то каждой букве от A до Z можно поставить  в соответствие числа от 1 до 26. Строка символов заменяется конкатенацией битовых образов этих чисел. Десятичное значение получившейся битовой последовательности используется в качестве k для хеширования.  Например, строковый ключ “ALPHA” можно преобразовать этим способом:  буква А кодируется числом 1, или 00001 2 ,  буква L кодируется числом 12, или 01100 2 ,  буква P кодируется числом 16, или 10000 2 ,  буква H кодируется числом 8, или 01000 2 .  Собирая эти битовые образы в целочисленной переменной, полу-  чаем двоичное значение 0000101100100000100000001, которое интерпретируется как десятичное значение k = 1458433. |
| Строка символов произвольной длины | **Степенной полином по правилу Горнера**  Вычисление k как числа в позиционной системе счисления по правилу Горнера. Например, если ключ-строка состоит только из заглавных букв латинского алфавита, то ASCII-код каждой буквы преобразуется в её порядковый номер в алфавите. Этот номер используется в качестве значения цифры в позиционной системе счисления с основанием 32, наиболее соответствующим мощности латинского алфавита (26 букв) и кириллицы  (33 буквы). Пример: k = "ALPHA" ,  k  = 1 Х32 4 + 12 Х32 3 + 16Х32 2 + 8Х32 1 + 1Х32 0 = 1458433.  Такие степенные полиномы эффективнее вычисляются с помощью  схемы Горнера:  k  = ((( 1Х32 + 12)×32 + 16)×32 +8)×32 + 1 = 1458433. |

**Коллизии**

Независимо от того, насколько хороша хэш-функция, время от времени она будет отображать два разных ключа в одно хэш-значение. Такой случай называется коллизией или столкновением. Эта ситуация неизбежна, поскольку количество возможных значений ключей значительно превышает размер хеш-таблицы.

Существуют два подхода к разрешению коллизий, а именно к реализации хэш-таблиц с учетом отработки коллизий:

1) **Применение цепочек:** Хеш-таблица с цепочками коллизий представляет собой массив списков коллизий. В позиции i массива хранится указатель на голову списка тех элементов, у которых хеш-значение ключа равно i. Если таких элементов в таблице нет, то позиция i содержит нулевой адрес null. Такой способ разрешения коллизий снимает ограничения на количество элементов, включённых в таблицу, но требует дополнительной памяти вне хэш-таблицы.

На рисунке 2 рассмотрен пример заполнения хэш-таблицы, используя простую хэш-функцию «key mod 7» для последовательности ключей

50, 700, 76, 85, 92, 73, 101.

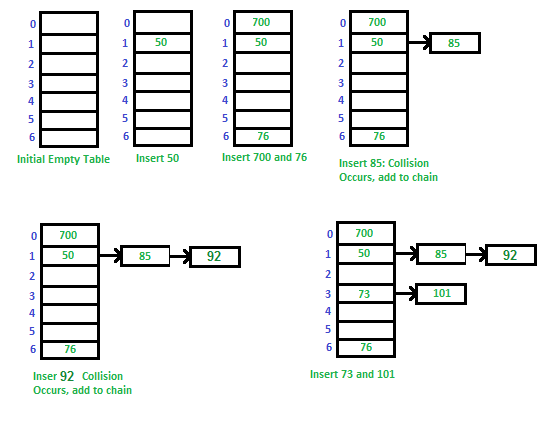


Рисунок 2 — пример заполнения хэш-таблицы с цепочками коллизий

**Достоинства:**  
1) Простота реализации.   
2) Хэш-таблица никогда не переполняется, мы всегда можем добавить больше элементов в цепочку.   
3) Менее чувствительна к выбору хэш-функции или коэффициенту заполненности.   
4) Обычно используется тогда, когда неизвестно ни количество ключей, ни частота их вставки и удаления.

**Недостатки:**  
1) Производительность хуже, чем у таблиц с открытой адресацией, поскольку ключи сохраняются в связанных списках.   
2) Потеря полезных областей памяти (поскольку часть блоков таблицы никогда не используется)  
3) Если цепочка становится слишком длинной, то время поиска приобретает ассимптотическую сложность O(n) в худшем случае.   
4) Использует дополнительную память для хранения списков.

Временная сложность хэш-таблиц с использованием цепочек коллизий:

Если m = Количество блоков в хэш-таблице, n = количество ключей уже добавленных в таблицу, то:

Коэффициент заполненности α = n/m

Ожидаемое время поиска ключа= O(1 + α)

Ожидаемое время удаления = O(1 + α)

Время вставки = O(1)

Временная сложность поиска, вставки и удаления O(1), если α O(1)

#### Пример вставки элемента в хэш-таблицу с цепочками:

**class** Hash

{

**int** BUCKET;    // количество блоков в хэш-таблице

    // указатель на массив блоков

    list<**int**> \*table;

**public**:

    Hash(**int** V);  // Конструктор

    // вставка ключа key

**void** insertItem(**int** x);

   // хэш-функция

**int** hashFunction(**int** x) {

**return** (x % BUCKET);

    }

  };

Hash::Hash(**int** b)

{

**this**->BUCKET = b;

    table = **new** list<**int**>[BUCKET];

}

**void** Hash::insertItem(**int** key)

{

**int** index = hashFunction(key);

    table[index].push\_back(key);

}

2) **Открытая адресация:** Хеш-таблица с открытой адресацией представляет собой массив Т, в который помещаются данные по индексу, равному хеш-значению ключа. При возникновении коллизии выполняется поиск другой ячейки массива, в которой может разместиться элемент. Эта ячейка должна быть свободной (открытой). Для пометки состояния в ячейки таблицы вводится специальный признак состояния ячейки **state**. Ячейка хеш-таблицы может быть в одном из трех состояний: свободна (**free**), занята (**busy**), свободна после удаления из неё элемента (**deleted**). Открытой считается ячейка, находящаяся в состоянии free или deleted. При поиске свободной ячейки вычисляется и просматривается последовательность индексов ячеек массива. Последовательность зависит от ключа k и номера попытки зондирования i. Для вычисления последовательности к хеш-функции добавляется второй аргумент – номер попытки (нумерация начинается с 0). Последовательность ячеек для некоторого ключа k имеет вид <h (k, 0), h (k, 1), ..., h (k, m – 1)>.

При открытой адресации все элементы хранятся в самой хэш-таблице. Поэтому размер таблицы должен быть больше или равен общему количеству ключей.

Insert(k): Продолжаем поиск пока не будет найден пустой слот, как только он будет найден — добавляем ключ.

Search(k): Продолжаем поиск пока не найдем слот с ключом равным k или не достигнем пустого слота.

Delete(k): Операция удаления ключа довольно сложная процедура. Если мы просто удалим ключ, то поиск может перестать работать. Поэтому слоты удаляемых ключей помечаются как «удаленные». Вставка может вставить ключ в удаленный слот, но поиск не останавливается на такой ячейке.  
Обычно используются три способа поиска свободных ячеек: последовательное исследование (зондирование) (sequential probing), квадратичное зондирование и зондирование с двойным хешированием.

При **линейном зондировании** хеш-функция при каждой попытке поиска вычисляет новую ячейку по формуле h (k, i) = (h(k) + i) mod m, где h(k) – обычная хеш-функция; i – номер попытки поиска свободной ячейки.

Если слот hash(x) % S занят, вычислить (hash(x) + 1) % S

Если (hash(x) + 1) % S также занят, вычислить (hash(x) + 2) % S

Если (hash(x) + 2) % S также занят, вычислить (hash(x) + 3) % S и т.д.

Недостаток: возникновение длинных последовательностей (кластеров), удлиняющих поиск свободной ячейки.

При **квадратичном зондировании** хеш-функция задаёт квадратичную последовательность ячеек по формуле h (k, i) = (h (k) + c 1 х i + c 2 хi 2 ) mod m, где c 1 ≠ 0 и c 2 ≠ 0 – некоторые константы. Пробы начинаются с ячейки T [h(k)], но дальше ячейки просматриваются не подряд. Номер пробуемой ячейки квадратично зависит от номера попытки. Для того чтобы

использовались все ячейки таблицы, c 1 и c 2 нужно тщательно подбирать.

При квадратичном зондировании тенденция к образованию кластеров частично устраняется. Хотя и остается эффект кластеризации в более мягкой форме – образование вторичных кластеров.

Если слот hash(x) % S занят, вычислить (hash(x) + 1\*1)%S

Если (hash(x) + 1\*1) % S также занят, вычислить (hash(x) + 2\*2)%S

Если (hash(x) + 2\*2) % S также занят, вычислить (hash(x) + 3\*3)%S и т.д.

**Двойное хеширование** – один из лучших методов зондирования в хеш-таблице с открытой адресацией. Последовательности индексов, возникающие при зондировании с двойным хешированием, близки к равномерному хешированию. При двойном хешировании функция h

имеет вид h (k, i) = (h(k) + iх h 1 (k)) mod m, где h(k) и h 1 (k) – обычные хеш-функции. То есть последовательность проб при зондировании является арифметической прогрессией по модулю m с первым членом h(k) и шагом h 1 (k).Чтобы зондирование ячеек охватывало все индексы таблицы, значение h 1 (k) должно быть взаимно простым с размером хеш-таблицы m. Так для мультипликативной хеш-функции h(k) размер хеш-таблицы m задаётся как степень двойки, а функция h 1 (k) – любое нечётное натуральное число. Если h(k) использует модульное хеширование и m – простое число, то значения h 1 (k) – любое натуральное число, меньшее m: h(k) = k mod m, h 1 (k) = 1 + (k mod m 1 ), где m 1 чуть меньше, чем m ( m 1 = m – 1 или m – 2).

Если слот hash(x) % S занят, вычислить (hash(x) + 1\*hash2(x)) % S

Если (hash(x) + 1\*hash2(x)) % S также занят, вычислить (hash(x) + 2\*hash2(x)) % S

Если (hash(x) + 2\*hash2(x)) % S занят, то вычислить (hash(x) + 3\*hash2(x)) % S и т.д.

Пример определения хэш-таблицы с открытой адресацией:

#define HASHTABLESIZE 51

typedef struct {

int key[HASHTABLESIZE];

int state[HASHTABLESIZE];

/\* -1=освобожден, 0=пусто, 1=занят \*/

} hashtable;

Пример вставки элемента в хэш-таблицу с открытой адресацией:

void open\_addressing\_insert(int item, hashtable \* ht )

{

int hash\_value;

hash\_value = hash(item);

i = hash\_value;

while (ht->state[i]== 1)

{

if (ht->key[i] == item) {

fprintf(stderr,”Duplicate entry\n”);

exit(1);

}

i = (i + F(i)) % HASHTABLESIZE;

if (i == hash\_value) {

fprintf(stderr, “The table is full\n”);

exit(1);

} }

ht->key[i] = item;

ht->state[i] = 1;

}

**unordered\_map и unordered\_set в библиотеке C++ STL**

Для реализации хеш-таблиц в стандартной библиотеке STL C++ предусмотрены контейнеры unordered\_map и unordered\_set. Контейнер unordered\_map сохраняет пары <ключ,значение> . Ключ используется как уникальный идентификатор, а значение — как содержимое, ассоциированное с ключом. И ключ и значение могут быть как базового типа, как и пользовательского. Данный контейнер реализован как хэш-таблица, поэтому среднее время поиска, вставки и удаления происходят за время O(1).

В отличие от unordered\_map в unordered\_set хранится только ключ (без ассоциированного значения), чаще всего этот контейнер используется для ответа на вопрос присутствует/отсутствует ключ в наборе, на практике этот контейнер, например, используется для поиска дубликатов.   
*Примечание: Помимо unordered\_map и unordered\_set в библиотеке C++ STL реализованы контейнеры map и set - в отличие от контейнеров описанных выше в этих контейнерах ключи хранятся упорядоченно, это связано с тем, что map реализован как BST-дерево. Ассимптотическая сложность операций над map O(Log n).*

Некоторые методы **unordered\_map** :

* at(): возвращает ссылку на значение по ключу k.
* begin(): возвращает итератор на первый элемент контейнера
* [end()](https://www.geeksforgeeks.org/unordered_map-end-function-in-c-stl/): возращает итератор на позицию после последнего элемента в контейнере
* bucket(): возвращает номер блока, в котором хранится ключ k.
* [bucket\_count:](https://www.geeksforgeeks.org/stdbucket_count-stdbucket_size-unordered_map-cpp/) bucket\_count используется для подсчета общего количества блоков в контейнере. Вызывается без параметров.
* bucket\_size: возвращает количество элементов в каждом блоке контейнера
* [count()](https://www.geeksforgeeks.org/unordered_map-count-in-c/): количество элементов, присутствующих в контейнере с соответствующим ключом.

Основные методы для работы с unordered\_map: оператор =, оператор [], empty и size для расчета текущей заполненности, begin и end для перемещения итераторов, find и count для поиска, insert и erase для вставки/удаления элементов.   
В библиотеке C++11 также предусмотрены функции задания хэш-функции и различных политик формирования хэша.

Пример использования и работы с unordered\_map:

**#include** <iostream>

**#include** <unordered\_map>

**using** **namespace** std;

**int** **main**(){

//создание хэш-таблицы

unordered\_map <string, string> hashSetEx;

hashSetEx.insert(std::make\_pair("Turanga", "Leela"));

hashSetEx.insert(std::make\_pair("Bender", "Rodriguez"));

hashSetEx.insert(std::make\_pair("Philip", "Fry"));

//1 вариант вывода на экран хэш-таблицы

**for** (unordered\_map <string, string>::iterator p = hashSetEx.begin(); p != hashSetEx.end(); p++)

cout << p->first << " " << p->second<<"(bucket:"<<hashSetEx.bucket(p->first) << ")\n";

//2 вариант вывода на экран хеш-таблицы

**for** (**auto** x : hashSetEx)

cout << x.first << " " << x.second << **endl**;

cout<<hashSetEx.find("Philip")->second<<**endl**;

**return** 0;

}

**Порядок выполнения лабораторной работы:**

1. Спроектируйте и реализуйте абстрактный тип данных «хеш-таблица» для данных в соответствии с вариантом задания.
2. Проведите оценку средней трудоёмкости операций вставки, удаления и поиска элементов в хеш-таблице в зависимости от коэффициента заполнения: 0,5; 0,7; 0,9 .
3. Оформите отчет по лабораторной работе.
4. Защитите работу.

**Требования к оформлению лабораторной работы и отчета:**

1. Работа выполняется в форме консольного приложения
2. **Модуль программы** должен предваряться комментарием с текстом исходной задачи, имена переменных и компонентов должны быть понятны стороннему пользователю или сопровождаться комментарием, объясняющим их смысл. Имена проекта и модуля должны иметь осмысленные названия.
3. **Отчет по лабораторной работе** должен содержать: титульный лист, текст задания, формат разработанной «хеш-таблицы», оценку трудоёмкости операций вставки, удаления и поиска элементов в хеш-таблице, листинг самой программы с комментариями.

**Контрольные вопросы:**

1) Какой из методов разрешения коллизий в наименьшей степени зависит от коэффициента заполненности хэш-таблицы, поясните почему?

2) Поясните, что такое «коллизии» в хешировании.

3) Поясните, почему при удалении элемента из хеш-таблицы элемент на самом деле не удаляется, а помечается как «свободная» ячейка?

**Задание**

|  |  |
| --- | --- |
| **Вариант 1** | |
| **Форма хеш-таблицы** | С цепочками коллизий |
| **Исходные данные (ключ)** | Строка текста произвольной длины (символы – заглавные буквы латинского алфавита) |
| **Метод преобразования ключа в натуральное число** | Преобразование методом конкатенации битовых образов символов. |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск подстроки в строке с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 2** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Двойное хеширование |
| **Исходные данные (ключ)** | Строка текста произвольной длины (символы – заглавные буквы латинского алфавита) |
| **Метод преобразования ключа в натуральное число** | Преобразование методом конкатенации битовых образов символов. |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск подстроки в строке с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free/busy/deleted).

|  |  |
| --- | --- |
| **Вариант 3** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | квадратичный |
| **Исходные данные (ключ)** | вещественное число от 0 до 10 000 |
| **Метод преобразования ключа в натуральное число** | Преобразование k к натуральному значению с точностью 10 3 и выбор цифр из середины квадрата k |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск первого неповторяющегося элемента в массиве вещественных чисел с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 4** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | целое число на интервале [0, +1 000 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод выбора цифр |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск является ли массив подмассивом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 5** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | строка текста произвольной длины (символы – заглавные буквы латинского алфавита). |
| **Метод преобразования ключа в натуральное число** | по схеме Горнера |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск трех наиболее повторяющихся слов в тексте с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 6** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Квадратичное зондирование |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Преобразование k к натуральному значению с точностью 104 и свёртка к значению k на интервале [10 000, +20 000]. |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 7** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Квадратичное зондирование |
| **Исходные данные (ключ)** | строка текста произвольной длины (символы – заглав-  ные буквы латинского алфавита) |
| **Метод преобразования ключа в натуральное число** | методом конкатенации битовых образов символов. |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск потенциального плагиата из фрагмента текста с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

**Примечание 2:** Для реализации функции варианта поиска плагиата можно создать хэш-таблицу всех перекрывающихся окон (подстрок) длиной w символов для документов. Любое совпадение хэш-кодов означает, что оба фрагмента текста, вероятно, содержат одинаковую подстроку длиной w символов, что и будет индикатором потенциального плагиата. Значение w нужно выбрать достаточно длинным, чтобы минимизировать вероятность случайного совпадения хэш-кодов. Чтобы уменьшить размеры получаемой таблицы можно оставить только некоторое подмножество хэш-кодов

|  |  |
| --- | --- |
| **Вариант 8** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Квадратичное зондирование |
| **Исходные данные (ключ)** | натуральное число на интервале [100 000 000,  300 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод свертки |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 9** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | строка текста произвольной длины (символы – заглав-  ные буквы кириллицы) |
| **Метод преобразования ключа в натуральное число** | по схеме Горнера |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск дубликата строки в хэш-таблице |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).



|  |  |
| --- | --- |
| **Вариант 10** | |
| **Форма хеш-таблицы** | С цепочками коллизий |
| **Исходные данные (ключ)** | Строка текста произвольной длины (символы – заглавные буквы латинского алфавита) |
| **Метод преобразования ключа в натуральное число** | Степенной полином по правилу Горнера |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск подстроки в строке с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 11** | |
| **Форма хеш-таблицы** | С цепочками коллизий |
| **Исходные данные (ключ)** | натуральное число на интервале [100 000 000,  300 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод выбора цифр |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 12** | |
| **Форма хеш-таблицы** | С цепочками коллизий |
| **Исходные данные (ключ)** | натуральное число на интервале [100 000 000,  300 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод свертки |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 13** | |
| **Форма хеш-таблицы** | С цепочками коллизий |
| **Исходные данные (ключ)** | натуральное число на интервале [1 000 000,  3 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод «середины квадрата» |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск минимального количества операций, чтобы сделать все элементы равными, с помощью хэш-таблиц  Пример:  Вход : arr[] = {1, 2, 3, 4}  Выход : 3  Поскольку все элементы различны,необходимо произвести как минимум 3 операции, чтобы сделать их одинаковыми (например, равными 3 или 1)  Вход : arr[] = {1, 1, 1, 1}  Выход : 0 |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 14** | |
| **Форма хеш-таблицы** | С цепочками коллизий |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод «середины квадрата» |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 15** | |
| **Форма хеш-таблицы** | С цепочками коллизий |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод свертки |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 16** | |
| **Форма хеш-таблицы** | С цепочками коллизий |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод выбора цифр |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 17** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | строка текста произвольной длины (символы – заглав-  ные буквы кириллицы) |
| **Метод преобразования ключа в натуральное число** | Преобразование методом конкатенации битовых образов символов. |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск дубликата строки в хэш-таблице |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 18** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | строка текста произвольной длины (символы – заглав-  ные буквы кириллицы) |
| **Метод преобразования ключа в натуральное число** | Преобразование методом конкатенации битовых образов символов. |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск дубликата строки в хэш-таблице |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 19** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | натуральное число на интервале [1 000 000,  3 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод «середины квадрата» |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск дубликата массива в другом массиве |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 20** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | натуральное число на интервале [1 000 000,  3 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод свертки |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск минимального количества операций, чтобы сделать все элементы равными, с помощью хэш-таблиц  Пример:  Вход : arr[] = {1, 2, 3, 4}  Выход : 3  Поскольку все элементы различны,необходимо произвести как минимум 3 операции, чтобы сделать их одинаковыми (например, равными 3 или 1)  Вход : arr[] = {1, 1, 1, 1}  Выход : 0 |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 21** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод выбора цифр |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск первого неповторяющегося элемента в массиве вещественных чисел с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 22** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод свертки |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск первого неповторяющегося элемента в массиве вещественных чисел с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 23** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Линейное зондирование |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод «середины квадрата» |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск первого неповторяющегося элемента в массиве вещественных чисел с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 24** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Двойное хеширование |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод свертки |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск первого неповторяющегося элемента в массиве вещественных чисел с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 25** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Двойное хеширование |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод «середины квадрата» |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск дубликата массива в другом массиве с использованием хэш-таблиц |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 26** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Двойное хеширование |
| **Исходные данные (ключ)** | вещественное число на интервале [10000.0000, +15000.0000] |
| **Метод преобразования ключа в натуральное число** | Метод выбора цифр |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 27** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Двойное хеширование |
| **Исходные данные (ключ)** | строка текста произвольной длины (символы – заглав-  ные буквы кириллицы) |
| **Метод преобразования ключа в натуральное число** | по схеме Горнера |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск дубликата строки в хэш-таблице |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 28** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Двойное хеширование |
| **Исходные данные (ключ)** | натуральное число на интервале [1 000 000,  3 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод свертки |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск элементов, которые присутствуют в одном массиве и отсутствуют в другом с использованием хэш-таблицы |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 29** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Двойное хеширование |
| **Исходные данные (ключ)** | натуральное число на интервале [1 000 000,  3 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод «середины квадрата» |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск минимального количества операций, чтобы сделать все элементы равными, с помощью хэш-таблиц  Пример:  Вход : arr[] = {1, 2, 3, 4}  Выход : 3  Поскольку все элементы различны,необходимо произвести как минимум 3 операции, чтобы сделать их одинаковыми (например, равными 3 или 1)  Вход : arr[] = {1, 1, 1, 1}  Выход : 0 |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).

|  |  |
| --- | --- |
| **Вариант 30** | |
| **Форма хеш-таблицы** | с открытой адресацией |
| **Метод разрешения коллизий** | Двойное хеширование |
| **Исходные данные (ключ)** | натуральное число на интервале [1 000 000,  3 000 000] |
| **Метод преобразования ключа в натуральное число** | Метод выбора цифр |
| **Хеш-функция** | Модульная |
| **Операции интерфейса**  **общие\*** | конструктор таблицы,  поиск элемента по ключу k,  вставка элемента по ключу k,  удаление элемента по ключу k,  текущее количество элементов в таблице,  текущий размер таблицы  проверка на пустоту таблицы  вывод содержимого таблицы на экран  очистка таблицы  \*можете расширить перечень функциями по усмотрению |
| **Функция варианта** | Поиск вхождения подмассива в массив с использованием хэш-таблиц |

**Примечание:** При выводе структуры хеш-таблицы с цепочками коллизий в строках экрана должны отражаться индексы таблицы и значения ключей в цепочках. При выводе структуры хеш-таблицы с открытой адресацией в строках экрана должны отражаться индексы таблицы, значения ключей, состояния ячеек хеш-таблицы (free /busy /deleted).