Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

**ЛАБОРАТОРНАЯ РАБОТА №4**

**Тема: Обработка данных на основе метода закрытого хеширования**

по учебной дисциплине «Алгоритмы и структуры данных»

Выполнил:

студент гр. 381574

Жгуновский О.Б.

Минск — 2024

**Интерпретация задачи**

Необходимо вывести массив из 7 целых чисел, из диапазона 47000−89000. После чего осуществить поиск элемента в хеш-таблице. На экран вывести исходный массив, хеш-таблицу, и результат поиска. Ввод должен производиться как в ручном режиме, так и автоматически.

**Код программы**

**using System;**

**using System.Collections.Generic;**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int n = 7;**

**int lowerBound = 47000;**

**int upperBound = 89000;**

**int M = 10;**

**// Создание хеш-таблицы**

**List<LinkedList<int>> hashTable = new List<LinkedList<int>>(M);**

**for (int i = 0; i < M; i++)**

**{**

**hashTable.Add(new LinkedList<int>());**

**}**

**Console.WriteLine("Выберите способ ввода данных:");**

**Console.WriteLine("1. Ввод вручную");**

**Console.WriteLine("2. Автоматический ввод");**

**string choice = Console.ReadLine();**

**if (choice == "1")**

**{**

**List<int> arr = new List<int>();**

**// Ввод вручную**

**for (int i = 0; i < n; i++)**

**{**

**Console.WriteLine($"Введите элемент {i + 1} (в диапазоне {lowerBound}-{upperBound}):");**

**int inputValue;**

**while (!int.TryParse(Console.ReadLine(), out inputValue) || inputValue < lowerBound || inputValue > upperBound)**

**{**

**Console.WriteLine($"Ошибка: введите целое число в диапазоне {lowerBound}-{upperBound}.");**

**}**

**arr.Add(inputValue);**

**int hashIndex = HashFunction(inputValue, M);**

**hashTable[hashIndex].AddLast(inputValue);**

**}**

**//выводим массив**

**Console.Write("Заполняем следующими значениями: ");**

**foreach (var number in arr)**

**{**

**Console.Write(number + " ");**

**}**

**Console.WriteLine();**

**}**

**else if (choice == "2")**

**{**

**// Автоматический ввод**

**Console.Write("Заполняем следующими значениями: ");**

**for (int i = 0; i < n; i++)**

**{**

**int randomValue = GenerateRandomNumber(lowerBound, upperBound);**

**//Для вывода массива**

**Console.Write(randomValue + " ");**

**int hashIndex = HashFunction(randomValue, M);**

**hashTable[hashIndex].AddLast(randomValue);**

**}**

**Console.WriteLine();**

**}**

**else**

**{**

**Console.WriteLine("Ошибка: неверный выбор.");**

**return;**

**}**

**// Вывод хеш-таблицы**

**Console.WriteLine("Хеш-таблица:");**

**for (int i = 0; i < M; i++)**

**{**

**Console.Write($"[{i}] -> ");**

**foreach (var item in hashTable[i])**

**{**

**Console.Write($"{item} -> ");**

**}**

**Console.WriteLine("null");**

**}**

**// Поиск элемента в хеш-таблице**

**Console.WriteLine("Введите элемент для поиска в хеш-таблице:");**

**int searchElement;**

**while (!int.TryParse(Console.ReadLine(), out searchElement))**

**{**

**Console.WriteLine("Ошибка: введите целое число для поиска.");**

**}**

**int searchIndex = HashFunction(searchElement, M);**

**bool found = hashTable[searchIndex].Contains(searchElement);**

**if (found)**

**{**

**Console.ForegroundColor = ConsoleColor.Green;**

**Console.WriteLine($"Элемент {searchElement} найден в хеш-таблице.");**

**}**

**else**

**{**

**Console.ForegroundColor = ConsoleColor.Red;**

**Console.WriteLine($"Элемент {searchElement} не найден в хеш-таблице.");**

**}**

**Console.ResetColor();**

**}**

**// Хеш-функция**

**static int HashFunction(int key, int M)**

**{**

**return key % M;**

**}**

**// Функция генерации случайного числа в заданном диапазоне**

**static int GenerateRandomNumber(int lowerBound, int upperBound)**

**{**

**Random random = new Random();**

**return random.Next(lowerBound, upperBound + 1);**

**}**

**}**

**Демонстрация работы программы**

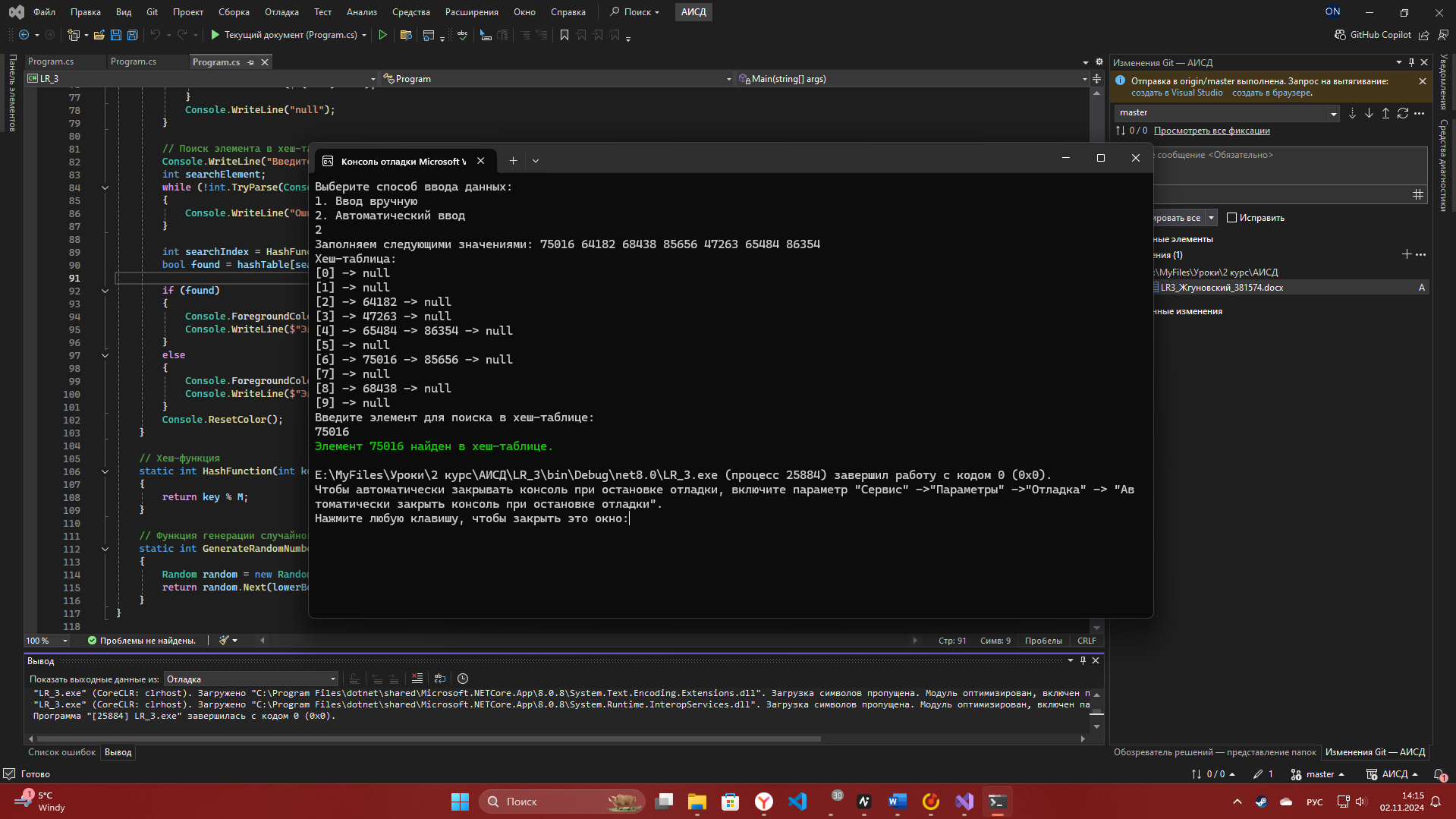


Рисунок 1 – автоматическое заполнение и успешный поиск

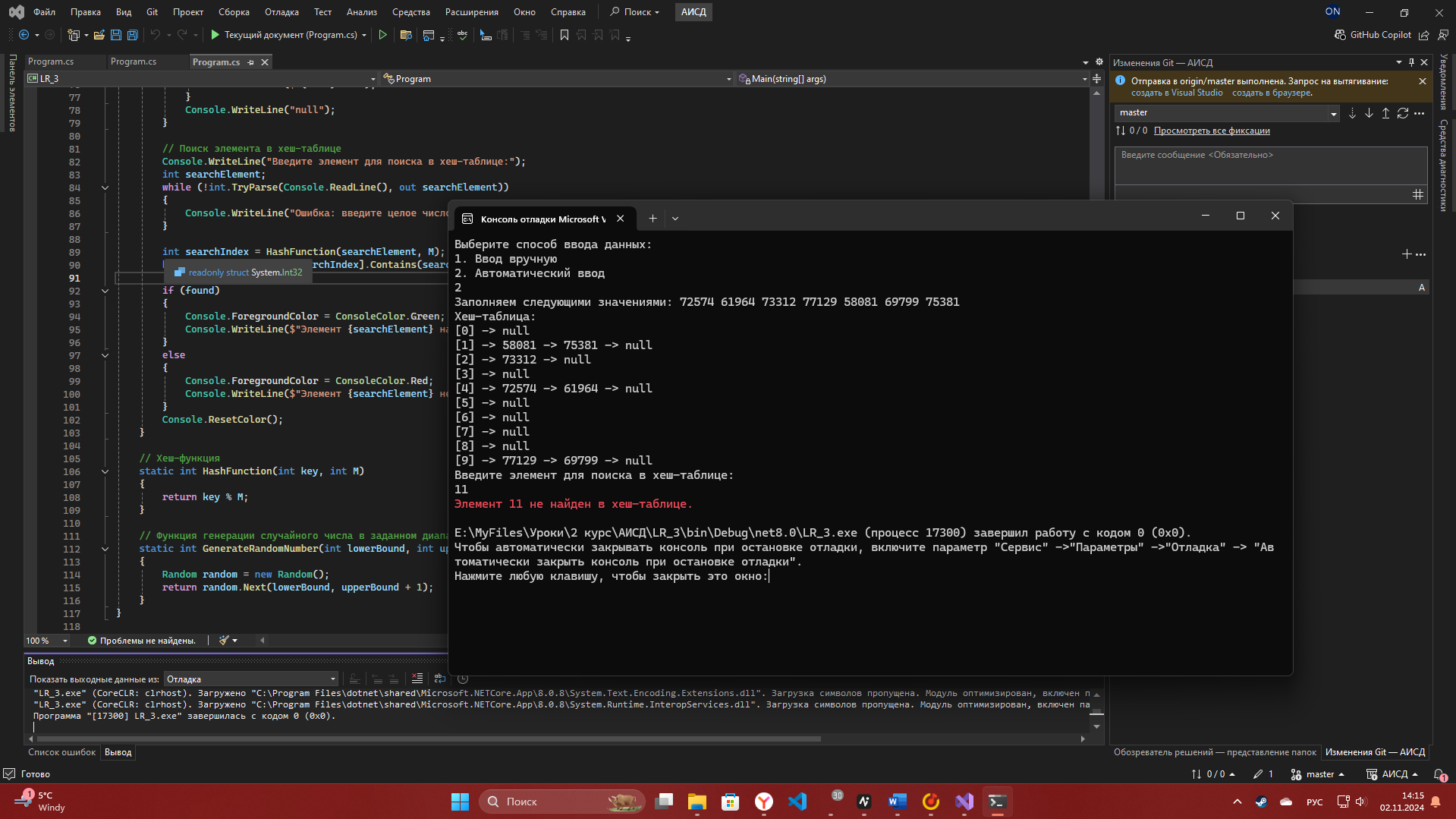


Рисунок 2 – автоматическое заполнение и неудачный поиск

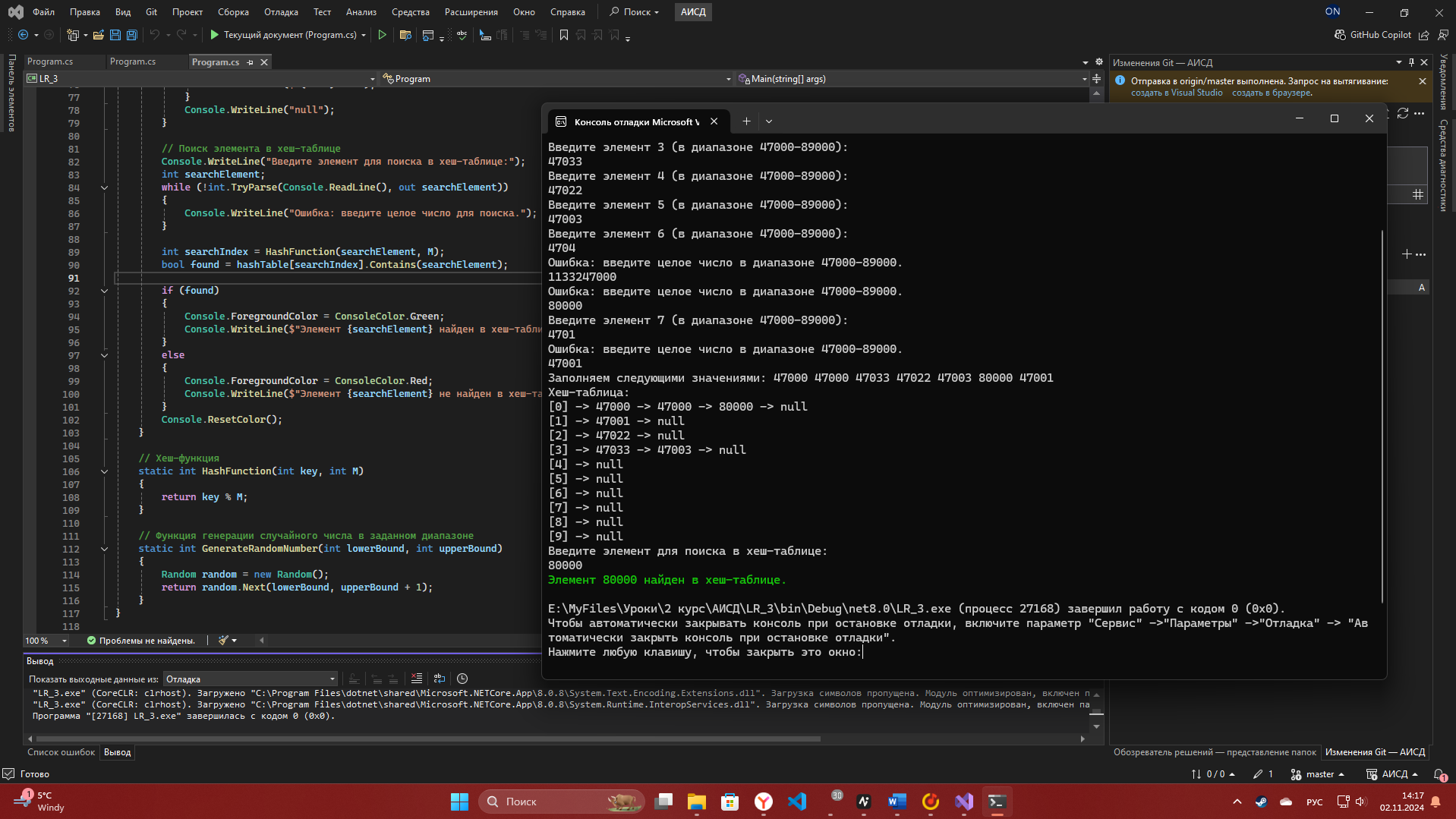


Рисунок 3 – ручное заполнение и успешный поиск

**Контрольные вопросы**

*В чем отличия между открытым и закрытым хешированием данных?*

В открытом хешировании, каждая ячейка хеш-таблицы хранит указатель на список/цепочку, в которым размещаются все элементы с одинаковым хешем. Если происходит коллизия – элемент кидается в этот список следующим

При закрытом хешировании, ячейка хранит не список, а элемент. И при возникновении коллизии, происходит поиск другой ячейки по установленному алгоритму (например, закидывается в следующую ячейку)

*Объясните суть коллизий при закрытом хешировании данных.*

Хеш-функция при прогоне через неё двух разных значений, выдаёт один и тот же хеш. В результате нужно пытаться решить, как положить в хеш-таблицу новое значение.

*Перечислите способы борьбы с коллизиями.*

1. Линейное пробивание – алгоритм последовательно проверяет каждую следующую ячейку таблицы (с шагом 1) до тех пор, пока не найдёт пустую.
2. Квадратичное пробивание – система ищет новую ячейку, но делает это с возрастающим шагом.
3. Двойное хеширование – используется вторая хеш-функция, которая определяет размер шага.