Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

Хэш-таблицы

Вариант № 7

Преподаватель \_\_\_\_\_\_\_\_\_\_ И. В. Матковский

подпись, дата

Студент КИ20-08Б 032049025 \_\_\_\_\_\_\_\_\_\_ А. Н. Головань

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2021

**1 Цель работы**

Изучить алгоритмы хеширования и методы разрешения коллизий.

**2 Порядок выполнения работы**

разработать программу, демонстрирующую основные этапы работы с

хэш-таблицами – формирование хэша, получение данных, разрешение

коллизий.

**3 Задание**

|  |  |  |  |
| --- | --- | --- | --- |
| *№* | *Метод хэширования* | *Тип данных* | *Метод разрешения коллизий* |
| 7 | Деление | числа | Метод цепочек |

**4 Код программы**

Program.cpp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120 | a=23  key = 10  count = 34  ####################  print("\n\n\n\n")    def hash(value,key):  result=value%key  return result  def getValue(hashTable,targetHash,key):  listValue=[]  for hashDict in hashTable:  if targetHash in hashDict:  listValue = hashDict[targetHash]    if listValue:  return listValue    return -1  def delValue(hashTable,hash):  for i in range(len(hashTable)):  if hashTable[i]["hash"] == hash:  del hashTable[i]  return 1  return -1  def putValue(hashTable,value,key):  targetHash = hash(value,key)  for hashDict in hashTable:  if targetHash in hashDict:  listValue = hashDict[targetHash]  if value not in listValue:  listValue.reverse()  listValue.append(value)  listValue.reverse()  return hashTable  def newPutValue(hashTable,value,key):  targetHash = hash(value,key)  here = False  for hashDict in hashTable:  if targetHash in hashDict:  listValue = hashDict[targetHash]  if not listValue or here:  listValue.append(value)  else:  here = True    for i in hashDict:  listValue = hashDict[i]    if not listValue and here:  here = False  listValue.append(value)  return hashTable  def newGetValue(hashTable,value,key):  targetHash = hash(value,key)  here = False  listValue=[]  for hashDict in hashTable:  if targetHash in hashDict or here:  here = True    for i in hashDict:  if hashDict[i] == [value]:  return value  return -1  def newDelValue(hashTable,value,key):  targetHash = hash(value,key)  here = False  listValue=[]  for hashDict in hashTable:  if targetHash in hashDict or here:  here = True    for i in hashDict:  if hashDict[i] == [value]:  hashDict[i] = []  return -1  def delValue(hashTable,value,key):  targetHash = hash(value,key)  for hashDict in hashTable:  if targetHash in hashDict:  listValue = hashDict[targetHash]  if value in listValue:  del listValue[listValue.index(value)]  return hashTable  hashTable = []  for i in range(key):  table = {}  table[hash(i,key)] = []  hashTable.append(table)  print("hashTable was generated")  newPutValue(hashTable,2,key)  newPutValue(hashTable,12,key)  newPutValue(hashTable,2,key)  newPutValue(hashTable,2,key)  newPutValue(hashTable,14,key)  newPutValue(hashTable,74,key)  newDelValue(hashTable,14,key)  # newDelValue(hashTable,5,key)  for i in hashTable:  print(i)  x = newGetValue(hashTable,14,key)  print(x)  # x = getValue(hashTable,0,key)  # print(x) |

**5 Теоретические оценки сложности алгоритмов**

В программе используются два алгоритма разрешения коллизий. Первый из них – метод цепочек, его сложность составляет . В качестве второго алгоритма используется Метод открытой адресации, так же имеет сложность .

**6 Экспериментальные оценки сложности алгоритмов**

Таблица 1 - зависимость времени от объема входных данных

|  |  |  |  |
| --- | --- | --- | --- |
| Количество значений в хэш таблице. | 150 | 500 | 1000 |
| Метод цепочек с | 0 с | 0.005 с | 0.009 с |
| Метод открытой адресации с | 0 с | 0 с | 0 с |