БУ ВО Ханты-Мансийского автономного округа – Югры

«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

ОТЧЁТ

по лабораторной работе №7

по дисциплине: «Алгоритмы и структуры данных»

Выполнил: студент(ка) группы №609-31,

Гаврилов Е.Е.

Принял: старший преподаватель кафедры АиКС

Назаров Е.В.

Сургут

2025г.

**Цель работы:** изучить принципов построения хеш-функций, обладающих равномерным распределением, исследовать статистические свойства хеш-функций, закрепить навыки структурного программирования.

**Задание:**

1. Разработать и реализовать функцию, осуществляющую хеширование данных (тип данных определяется вариантом).
2. Разработать и реализовать функцию-генератор, осуществляющую формирование значений ключей в соответствии с заданным типом данных. Генерируемые ключи должны быть уникальны.
3. Исследовать статистические свойства разработанной хеш-функции при заданных размерах хеш-таблицы и количестве ключей.
4. Составить отчет, в котором привести листинг хеш-функции, гистограммы распределений индексов, формируемых хеш-функцией (для двух значений размера хеш-таблицы) и выводы по работе (дать оценку зависимости от размера таблицы и от природы исходных данных – если таковые имеются; оценить качество разработанной хеш-функции).

**Вариант 4:**

Таблица 1. Индивидуальное задание.

|  |  |  |
| --- | --- | --- |
| Тип данных | Размеры хэш-таблицы и количество ключей | Примечание |
| struct Date{ int Month; int Year;}; | M1 = 64 M2 = 67 K = 200 | Диапазон изменения Year равен [1972; 1992]. |

Рис 1. Гистограмма распределений индексов, сформированных функцией хэширования для таблицы размера 64.

Рис 2. Гистограмма распределений индексов, сформированных функцией хэширования для таблицы размера 67.

Таблица 2. Характеристики распределения в зависимости от размера таблицы.

|  |  |  |  |
| --- | --- | --- | --- |
| Размер таблицы | Среднее количество индексов | Медиана распределения индексов | Статистическая оценка χ2 |
| 64 | 3.125 | 3 | 2.24 |
| 67 | 2.985 | 3 | 0.33 |

**Анализ:**

Индексы распределены равномерно. Отклонения от среднего не превышают. Статистическая оценка показывает хороший результат. Это достигается в основном за счет природы хешируемых данных: всего 240 (12 \* 20) возможных различных значений ключей и всего 200 различных значений для анализа.

**Вывод:** изучены принципы построения хеш-функций, обладающих равномерным распределением, исследованы статистические свойства хеш-функции.

Листинг 1.

#include <iostream>

#include <fstream>

#include <math.h>

#include <map>

#include <unordered\_map>

#include <functional>

using namespace std;

struct Date

{

int Year;

int Month;

Date(int a, int b)

{

Year = a;

Month = b;

}

Date()

{

Month = 0;

Year = 0;

}

bool operator==(const Date &d) const

{

return Year == d.Year && Month == d.Month;

}

};

namespace std

{

template <>

struct hash<Date>

{

std::size\_t operator()(const Date &obj) const

{

return obj.Year \* 12 + obj.Month;

}

};

}

void DateGen(Date \*array, int n)

{

int i, j, c;

c = 0;

for (i = 1972; i <= 1992 && c < n; i++)

for (j = 1; j <= 12 && c < n; j++)

{

array[c] = Date{i, j};

c++;

}

}

void HashMapping(int \*map, Date \*dates, int m, int k)

{

hash<Date> haser;

for (int i = 0; i < k; i++)

map[haser(dates[i]) % m]++;

}

double xiFun(int \*map, int n, int m)

{

double result = 0;

for (int i = 0; i < m; i++)

result += ((double)map[i] - (double)n / m) \* ((double)map[i] - (double)n / m);

result \*= (double)m / n;

return result;

}

int main()

{

ofstream file1("m1.txt");

ofstream file2("m2.txt");

int k = 200;

Date dates[k];

DateGen(dates, k);

int m1 = 64, m2 = 67;

int map1[m1] = {0}, map2[m2] = {0};

HashMapping(map1, dates, m1, k);

HashMapping(map2, dates, m2, k);

for (auto i : map1)

file1 << i << endl;

for (auto i : map2)

file2 << i << endl;

file1 << "xi:" << xiFun(map1, k, m1);

file2 << "xi:" << xiFun(map2, k, m2);

int c = 0;

for (auto i : map1)

{

c += i;

}

cout << "m1:" << c << "/" << k << endl;

c = 0;

for (auto i : map1)

{

c += i;

}

cout << "m2:" << c << "/" << k << endl;

file1.close();

file2.close();

}