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

тема: «Хеширование»

Выполнила: студентка группы №609-11,

Пашина Илона Руслановна

Принял: доцент кафедры АиКС

Даниленко И.Н.

Сургут

2023г.

**Цель работы:**

Изучить принципы построения хеш-функций, обладающих равномерным распределением, исследовать статические свойства хеш-функций, закрепить навыки структурного программирования.

**Задание:**

1. Разработать и реализовать функцию, осуществляющую хеширование данных.
2. Разработать и реализовать функцию-генератор, осуществляющую формирование значений ключей в соответствии с заданным типом данных. Генерируемые ключи должны быть уникальны.
3. Исследовать статические свойства разработанной хеш-функции при заданных размерах хеш-таблицы и количестве ключей.
4. Составить отчёт, 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 8**

|  |  |  |
| --- | --- | --- |
| Тип данных | Размеры хеш-таблицы и количество ключей | Примечание |
| Struct Book{  int Year;  int Pages; }; | M1 = 1024  M2 = 1031  K = 5000 | Диапазоны изменения Year и Pages равны соответственно [2000; 2008] и [100; 600]. |

**Ход работы**

Для хеширования будем использовать значения «Year» и «Pages». Имеем (2008-2000+1) \* (600-100+1) = 4509 уникальных ключей, при необходимых 5000. Чтобы повысить количество уникальных ключей было решено увеличить диапазоны значений.

Новые диапазоны изменения Year и Pages равны соответственно [2000; 2009] и [100; 700].

Тогда количество уникальных ключей составит: (2009-2000+1) \* (700-100+1) = 6010.

Гистограмма 1. Распределение индексов, сформированных хеш-функцией

При размере хеш-таблицы 1024.

Значение Хи-квадрат: 2205,274

Гистограмма 1. Распределение индексов, сформированных хеш-функцией

При размере хеш-таблицы 1031.

Значение Хи-квадрат: 680,3976

В процессе написания программы были перепробованы различные варианты написания хеш-функции. В итоге самой лучшей (из того что получилось) оказалась формула:

((book.Pages - minPages) \* (1 + (book.Year%(maxYear - minYear)))) % TableSize,

**Вывод:**

В ходе лабораторной работы были изучены принципы построения хеш-функций, исследованы статические свойства хеш-функций, закреплены навыки структурного программирования.

**Листинг**

# include <iostream>

#include <ctime>

#define minPages 100

#define maxPages 700

#define minYear 2000

#define maxYear 2009

#define TableSize 1031

#define key 5000

using namespace std;

struct Book{

int Year;

int Pages;

};

struct Item{

Book book;

Item \* next;

};

void FillRand(Book \*book);

void PrintBook(struct Book \*book);

void CleanHashTable(Item \*\*HashTable);

int HashFunction(Book book);

void InsertItem(Book book);

void PrintHashTable(Item \*HashTable[]);

void FillRandHash(Item \*HashTable[], int quantity);

int main(){

srand(time(NULL));

struct Book \*a = new Book;

Item \* HashTable[TableSize];

CleanHashTable(HashTable);

FillRandHash(HashTable, key);

PrintHashTable(HashTable);

return 0;

}

void FillRand(Book \*book){

book->Pages = rand() % (maxPages - minPages) + minPages;

book->Year = rand() % (maxYear - minYear) + minYear;

}

void PrintBook(struct Book \*book){

cout << book->Pages << " " << book->Year << endl;

}

void CleanHashTable(Item \*HashTable[]){

for(int i = 0; i < TableSize; i++){

HashTable[i] = nullptr;

}

}

int HashFunction(Book book){

// return (book.Pages \* book.Year) % 1024;

// return (((book.Pages) \* 1024) / 500 + ((book.Pages) \* 1024) % 500)%TableSize;

return (int)((book.Pages - minPages) \* (1 + (book.Year%(maxYear - minYear)))) % TableSize; // лучший вариант

}

void InsertItem(Book book, Item \*HashTable[]){

int index = HashFunction(book);

Item \* temp = HashTable[index];

while(temp != nullptr){

if(book.Pages == temp->book.Pages && book.Year == temp->book.Year){

FillRand(&book);

index = HashFunction(book);

temp = HashTable[index];

continue;

}

temp = temp->next;

}

if (HashTable[index] == nullptr) {

HashTable[index] = new Item{book, nullptr};

}

else{

Item\* currentItem = HashTable[index];

while (currentItem->next != nullptr) {

currentItem = currentItem->next;

}

currentItem->next = new Item{book, nullptr};

}

}

void PrintHashTable(Item \*HashTable[]){

for(int i = 0; i < TableSize; i++){

// cout << i << ": ";

Item\* currentItem = HashTable[i];

int j = 0;

while (currentItem != nullptr) {

// cout << "(1); ";

j++;

currentItem = currentItem->next;

}

// if (j > 0) cout << j;

cout << j;

cout << endl;

}

}

void FillRandHash(Item \*HashTable[], int quantity){

for(int i = 0; i < quantity; i++){

struct Book \*a = new Book;

FillRand(a);

InsertItem(\*a, HashTable);

}

}