|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра прикладной математики | | | |
| Практическое задание № 4 | | | |
| по дисциплине «Структуры данных и алгоритмы» | | | |
| **«Управление таблицами»** | | | |
|  | | | |
|  | Группа | ПМ-21 | |
| Бригада 7 | Егупов иван | |
|  | Щербакова Мария | |
| Сагыдыев Данияр | |
|  | |
|  | |
| Преподаватель | ХИЦЕНКО ВАЛЕНТИНА ПАВЛОВНА | |
| Дата | 02.05. | |
| Новосибирск,2023 | | | |

1. **Условие задачи**

В файле WORK содержатся результаты работы цеха за день. Элемент файла включает: шифр изделия (8-символьный код), наименование изделия, количество (штук). Построить таблицу, содержащую результаты работы за день, считая ключом шифр изделия. Элемент таблицы имеет ту же структуру, что и элемент файла. Содержащаяся в файле информация с равными ключами должна быть помещена в таблицу один раз с общим количеством штук изделия. Организовать таблицу как таблицу с вычисляемым входом (таблицу с перемешиванием).

**2. Анализ задачи**

Входные данные:

P = {elemi = (keyi, namei, producedi)| keyi ∈ G, namei ∈ M, , producedi ∈ N,i>=1}

G = {gi|g i∈ {‘a’,’b’… ‘z’,’1’,’2’…’0’}, i = 1,8}

M = {mi|m i∈ {‘a’,’b’… ‘z’}, i = 1,10}

Выходные данные:

K ∈ {0;1}

Table = {eli = (keyi, namei, producedi)| keyi ∈ G, namei ∈ M, , producedi ∈ N,i<=10}

Решение:

v(key) =

f(ключ) = [N\*(v(key)\*А mod 1)], где A = ,N = 10, квадартные скобки – взятие целой части числа

**при** k = 1, Table = ∅

**повторять**   
elem = очередной элемент из P, hash = f (keyelem)

*если* Table[elhash] = ∅, то добавить elem в эту позицию

*иначе если* keyelem=keyel(hash)(шифры равны), то producedel(hash) = producedel(hash)+producedelem

*иначе*  **при** i = hash+1,flag = 1

**повторять** если eli = ∅, то добавить elem в эту позицию, flag = 0

иначе i = i+1, если i>N, то i = 0, если i = hash, то k = 0

**пока** k = 1 и flag = 1

**пока** k = 1 и ∃elem

Подзадачи:

*hashfunc*– вычисление хэш-значения для размещаемого элемента

Входные данные: шифр (ключ) размещаемого элемента

Выходные данные: целочисленное значение местоположения элемента в таблице

*add\_in\_table–* размещение очередного элемента в таблице

Входные данные: Таблица, очередной рассмтариваемый на размещение элемент

Выходные данные: целоче число – удалось ли вместить в таблицу очередной элемент

*Print\_table –* печать элементов таблицы в файл

Входные данные: Table, информация о файле, в который будет записан результат

**3.Структуры данных**

Входные данные

Внешнее представление:

В файле WORK содержатся элементы, состоящие из шифра изделия (8-символьный код), наименование изделия, количество. Каждая состовляющая элемента написана через пробел. Элемент – новая строка.

Внутреннее представление:

Элемент Последовательности P представлен следующей структурой

struct Elem

{

char key[K];

char name[B];

int produced;

};

const char K = 9; const char B = 11;

ВЫХОДНЫЕ ДАННЫЕ

*Внешнее представление:*

В первой строке файла число, означающее выполнение преобразования последовательности в таблицу (0 или 1);

С новой строки последовательность, состоящая из непустых элементов таблицы: позиции элемента, хеш-значения, шифра изделия (8-символьный код), наименования изделия, количества. Каждая составляющая элемента написана через пробел, каждый элемент – с новой строки.

*Внутреннее представление:*

Таблица представлена из массива структур размерности N=10

struct Elem table[N];

**4.Алгоритм решения задачи**



*4.1 Алгоритм вычисления хэш-значения для элемента последовательности P*

str = шифр текущего элемента последовательности P

int h = 0;

for (int i=1; \*str != 0;i++, str++)

h = (очередной символ шифра)\*i + h;

h = int(N \* Остаток(h \* , 1));

return h;

*4.2 Алгоритм добавления элемента последовательности P в таблицу Table*

int k = 1,flag = 1, place = 0;

int hash = Хэш-функция(elem.key);

while (k и существует elem) {

if (в позиции hash место свободны или ключи от elem и elemhash различны)

{

place = hash;

flag = 0;

if (strlen(table[place].key) == 0)

table[place] = elem;

else

table[place].produced += elem.produced;

}

else

{

int i = hash + 1; flag = 1;

while (k && не разрешили коллизию)

{

if (strlen(table[i].key) == 0)

{

if (strlen(table[i].key) == 0)

table[i] = elem;

else

table[i].produced += elem.produced;

flag = 0;

}

else i++;

if (i > N) i = 0;

if (i == hash) k = 0;

}

}

}

return k;

5.Структура программы

int hashfunc(char\* str) Назначение: вычисление хэш значения для рассмтариваемого элемента

char \*str – указатель на строку, содержащую шифр элемента

Выходные данные: вычисленное хэш-значение для рассмтариваемого элемента

int add\_in\_Table(Elem table[], Elem elem) Назначение: добавление текущего элемента последовательности P в таблицу

Elem table[] – таблица, в которую добавляем элемент

Elem elem – текущий рассматриваемый элемент

Выходные данные: результат 1 или 0, в зависимости, удалось ли уместить текущий элемент последовательности P в таблицу

void Print\_table(Elem table[], FILE\* output) Назначение: печать элементов таблицы в файл

Elem table[] – таблица, содержащая итоговый ответ

FILE \*output – указатель на файл, куда будет произведена печать

6.Текст программы

Заголовочный файл “Table.h”

#pragma once

#include <math.h>

#include <string.h>

const int K = 9;

const int B = 11;

const int N = 10;

struct Elem

{

char key[K];

char name[B];

int produced = 0;

};

int hashfunc(char\* str)

{

int h = 0;

for (int i=1; \*str != 0;i++, str++)

h = (\*str)\*i + h;

h = int(N \* fmod(h \* sqrt(5), 1));

return h;

}

int add\_in\_Table(Elem table[], Elem elem)

{

int k = 1,flag = 1, place = 0;

int hash = hashfunc(elem.key);

while (k && flag) {

if (strlen(table[hash].key) == 0 || !strcmp(table[hash].key, elem.key))

{

place = hash;

flag = 0;

if (strlen(table[place].key) == 0)

table[place] = elem;

else

table[place].produced += elem.produced;

}

else

{

int i = hash + 1;

while (k && flag)

{

if (strlen(table[i].key) == 0)

{

table[i] = elem; flag = 0;

}

else i++;

if (i > N) i = 0;

if (i == hash) k = 0;

}

}

}

return k;

}

void Print\_table(Elem table[], FILE\* output)

{

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

if (strlen(table[i].key) != 0) {

fprintf\_s(output, "%d %d %s %s %d \n", i, hashfunc(table[i].key), table[i].key, table[i].name, table[i].produced);

}

}

Основная программа Source.cpp

#include <stdio.h>

#include "Table.h"

int main()

{

Elem table[N]{ };

FILE\* work = NULL;

FILE\* output = NULL;

int flag = 1;

if (fopen\_s(&work, "WORK.txt", "r") == NULL) {

struct Elem el;

fscanf\_s(work, "%s", &el.key, K);

fscanf\_s(work, "%s", &el.name, B);

fscanf\_s(work, "%d", &el.produced);

while (flag && !feof(work))

{

flag = add\_in\_Table(table, el);

fscanf\_s(work, "%s", &el.key, K);

fscanf\_s(work, "%s", &el.name, B);

fscanf\_s(work, "%d", &el.produced);

}

fclose(work);

}

else printf\_s("Failed to open file WORK");

if (!flag) {

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

table[i].produced = NULL;

for (int j = 0; j < K; j++)

table[i].key[j] = NULL;

for (int j = 0; j < B; j++)

table[i].name[j] = NULL;

}

}

fopen\_s(&output, "Table.txt", "w");

fprintf\_s(output, "%d\n", flag);

Print\_table(table, output);

fclose(output);

return 1;

}

7.Набор тестов

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Назначение |
| Oryt876w Sergey 50 | 1  6 6 Oryt876w Sergey 50 | Файл содержал один элемент. Простейший тест |
| 87436104 Kour 40  Vbhfyewp LOOOP 1  Mnpi652x qwert 100 | 1  5 5 Vbhfyewp LOOOP 1  6 6 Mnpi652x qwert 100  7 7 8743610b Kour 40 | Файл содержал три элемента с различными шифрами |
| bvry89YR Zavik 50  otr56BN3 Dima 40  bvry89YR Zavik 70  yrue46PO Vanya 100  уbtr8472 Sanina 73 | 1  0 0 уbtr8472 Sanina 73  1 1 otr56BN3 Dima 40  6 6 bvry89YR Zavik 120  8 8 yrue46PO Vanya 100 | Файл содержал пять элементов, один из шифров встречался дважды |
| bvry89YR Zavik 50  otr56BN3 Dima 40  bvry89YR Zavik 70  yrue46PO Vanya 100  netwyr90 Trakimus 10000  trytvvb2 Oropfvw 5222  teyyvve2 Porvbed 562  pte6736v Poland 89  bvrepw00 BVHewe 13 | 1  0 6 bvrepw00 BVHewe 13  1 1 otr56BN3 Dima 40  2 2 trytvvb2 Oropfvw 5222  3 3 netwyr90 Trakimus 10000  6 6 bvry89YR Zavik 120  7 7 teyyvve2 Porvbed 562  8 8 yrue46PO Vanya 100  9 9 pte6736v Poland 89 | Файл содержал девять элементов, встретилась коллизия |
| bvry89YR Zavik 50  otr56BN3 Dima 40  bvry89YR Zavik 70  yrue46PO Vanya 100  netwyr90 Trakimus 10000  trytvvb2 Oropfvw 5222  teyyvve2 Porvbed 562  pte6736v Poland 89  bvrepw90 BVHewe 13  pppppppp Bpppe2 78  tttttttt ororow 5 | 1  0 8 pppppppp Bpppe2 78  1 1 otr56BN3 Dima 40  2 2 trytvvb2 Oropfvw 5222  3 3 netwyr90 Trakimus 10000  4 8 tttttttt ororow 5  5 5 bvrepw90 BVHewe 13  6 6 bvry89YR Zavik 120  7 7 teyyvve2 Porvbed 562  8 8 yrue46PO Vanya 100  9 9 pte6736v Poland 89 | На вход поступило 11 элементов. Таблица полна, также присутсвовала коллизия |
| WsR43Kig sofa 52  Yyyy56UU dress 76  asDF0lkP stool 123  1as45hr8 booksh 45  ukII543L refrig 2  mkt78GY1 cabinet 55  234hyrt8 carpet 2  1Uki89fD hammock 9  XXX0krt3 mirror 187  Nun777FS wardro 11  678jkl11 cradle 567  yhbvGHas cot 444  rfgY67Ha lamp 12  xxx0KRt3 tablecl 87  98Mi34h4 stool 3 | 0 | Файл содержал 15 элементов, разместить все элементы в таблице не удалось |
| XXX0krt3 mirror 87  XXX0krt3 mirror 2  985HYasf carpet 447  xxx0KRt3 tablec 87  98Mi34h4 stool 3  mkt78GY1 cabin 55  985HYasf carpet 2  1YYY2345 dress 58  1Uki89fD hammo 9  XXX0krt3 mirror 187  Nun777FS wardr 11  mkt78GY1 cabin 501 | 1  0 9 1YYY2345 dress 58  1 7 Nun777FS wardr 11  4 4 98Mi34h4 stool 3  5 5 985HYasf carpet 449  6 6 mkt78GY1 cabin 556  7 7 XXX0krt3 mirror 276  8 4 1Uki89fD hammo 9  9 9 xxx0KRt3 tablec 87 | Файл содержал 12 элементов,три раза встретила коллизия, присутсвовали два элемента,встретившиеся несколько раз |

Количество коллизий, получаемых при работе программы на большом количестве входных данных при заданном размере таблицы:

|  |  |
| --- | --- |
| Размер таблицы | Количество коллизий |
| 12 | 3 |
| 15 | 1 |
| 23 | 1 |
| 30 | 0 |