|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра прикладной математики | | | |
|  | | | |
| Практическое задание № 4 | | | |
| по дисциплине «Структуры данных и алгоритмы» | | | |
| **Управление таблицами** | | | |
|  | | | |
|  | Группа | ПМ-95 |
| Бригада | 8 |
| Алгоритм | Лаворов Дмитрий |
| Реализация | Костыря Сергей |
| Тестирование | Краус Александр |
| Преподаватель | Тракимус Юрий Викторович |
| Дата | 17.06.2020 |
|  | | | |
| Новосибирск | | | |

**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 должны быть помещена в таблицу один раз с общим количеством штук изделия. Организовать таблицу как неупорядоченную.

**2. Анализ задачи**

Дано: A = {шифр изделия, название изделия, количество}

Шифр ∈ последовательность символов;

Название ∈ последовательность латинских букв;

Количество ∈ Z+.

Результат: Таблица с вычисляемым входом вида: шифр изделия – последовательность букв и цифр, наименование изделия – последовательность латинских букв, количество изделий – целое положительное число.

Решение: Если просматривая файл work, найдутся несколько элементов с одинаковым ключом, в таблицу с вычисляемым входом заносится такой элемент один раз с общим количеством штук изделия. Если шифр одного изделия не равен шифру другого изделия, а наименование изделия равно наименованию второго изделия, то возникает коллизия и такая проблема решается методом линейного рехеширования: пусть ключ – сумма всех кодов символов входящих в имя переменной (шифр изделия). Для линейного рехеширования будем использовать следующую функцию: hash = (sym + i) % N (N – размер таблицы), i = 1, 2, … используя это рехеширование, при поиске свободного элемента таблицы последовательно рассматриваются все элементы таблицы, начиная с первоначального индекса, полученного в результате хеширования.

**3. Структура входных и выходных данных**

1) Внешнее представление входных данных:

Элементы F = {B, C, D|B – шифр изделия, C – название изделия, D – количество изделий}

2) Внешнее представление выходных данных:

Элементы F = {B, C, D|B – шифр изделия, C – название изделия, D – количество изделий}, изменённые в результате линейного рехеширования и сложения количества элементов с одинаковым ключом.

3) Внутреннее представление входных данных:

code – шифр изделия,

name – наименование изделия,

numb – количество изделий.

4) Внутреннее представление выходных данных:

Таблица с вычисляемым входом – динамический массив с ключом.

**4. Текст программы**

#include <stdio.h>

#include <string.h>

const int N = 10000;

int num = 0;

struct table

{

char code[9];

char name[255];

int numb;

};

void table\_add(table\* t, char\* code, char\* name, int numb)

{

int flag = 0;

for (int i = 0; i < num && !flag; i++)

if (!strcmp(code, t[i].code))

{

flag++;

t[i].numb += numb;

}

if (!flag)

{

strcpy\_s(t[num].code, code);

strcpy\_s(t[num].name, name);

t[num].numb = numb;

num++;

}

}

void hash\_add(table\* t, char\* code, char\* name, int numb)

{

int sym = 0, hash;

for (int i = 0; i < 9; i++)

sym += name[i];

hash = sym % N;

for (int i = 1; i < N && strlen(t[hash].code); i++)

hash = (sym + i) % N;

if (!strlen(t[hash].code))

{

strcpy\_s(t[hash].code, code);

strcpy\_s(t[hash].name, name);

t[hash].numb = numb;

}

}

int main()

{

table\* t = new table[N];

FILE\* f;

fopen\_s(&f, "work.txt", "r");

int numb;

char name[255], code[9];

while (fscanf\_s(f, "%s %s %d", code, sizeof(code), name, sizeof(name), &numb) != EOF && num < N)

table\_add(t, code, name, numb);

fclose(f);

table\* t2 = new table[N];

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

{

t2[i].code[0] = '\0';

t2[i].name[0] = '\0';

t2[i].numb = 0;

}

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

hash\_add(t2, t[i].code, t[i].name, t[i].numb);

fopen\_s(&f, "table.txt", "w");

fprintf\_s(f, "|Шифр |Наименование |Количество |");

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

if (t2[i].code[0] != '\0')

fprintf(f, "\n|%8s| %12s| %10d|", t2[i].code, t2[i].name, t2[i].numb);

delete[] t;

delete[] t2;

return 0;

}

**5. Набор тестов**

|  |  |  |
| --- | --- | --- |
| **№** | **Входные данные** | **Назначение** |
| 1 | po4852l4 table 67  78ok0009 chair 67  00001940 desk 56  00100907 screw 899  190000p0 nut 98  679odr54 nail 90  77445150 sofa 76  74913005 drill 71  po4852l4 table 67  190000p0 nut 98 | Проверка правильности решения коллизии и правильности сложения количества элементов с одинаковыми ключами. |

**6. Результаты отладки**

|  |  |  |
| --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** |
| 1 | po4852l4 table 67  78ok0009 chair 67  00001940 desk 56  00100907 screw 899  190000p0 nut 98  679odr54 nail 90  77445150 sofa 76  74913005 drill 71  po4852l4 table 67  190000p0 nut 98 | |Шифр |Наименование |Количество |  |190000p0| nut| 196|  |679odr54| nail| 90|  |00001940| desk| 56|  |77445150| sofa| 76|  |78ok0009| chair| 67|  |po4852l4 | table| 134|  |74913005| drill| 71|  |00100907| screw| 899| |