|  |  |  |
| --- | --- | --- |
|  | | |
| 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** | | |
|  | | |
|  | Бригада 7 | Дроздов Даниил |
| Группа ПМ-15 | Чепурная Алина |
| Вариант 2 б | Палий Никита |
|  |  |
|  |  |
|  |  |
| Преподаватели | Тракимус Юрий Викторович |
|  |  |
| Новосибирск, 2022 | | |

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 таблице информация с равными ключами должна быть помещена в таблицу один раз с общим количеством штук изделия. Организовать таблицу как упорядоченную.

1. **Анализ задачи**

Входные данные:

Шифр изделия – последовательность из 8 символов;

Наименование изделия – последовательность латинских букв.

Количество изделий – целое положительное число.

Выходные данные – упорядоченная таблица вида:

Шифр изделия – последовательность из 8 символов;

Наименование изделия – последовательность латинских букв;

Количество изделий – целое положительное число.

Решение: составить таблицу из содержимого файла WORK, при этом при обнаружении нескольких элементов с одинаковым ключом, занести такой элемент в таблицу один раз с количеством изделия равным сумме количества изделий одинаковых элементов.

Внутреннее представление входных данных: два одномерных массива символьного типа key – шифр изделия и name – наименование изделия, и переменная quantity целого типа.

Внутреннее представление выходных данных: статический массив, где каждый элемент – структура, состоящая из двух одномерных массивов символьного типа key – шифр изделия и name – наименование изделия, и переменной quantity целого типа.

Внешнее представление входных данных: последовательность, элемент которой состоит из 8-символьной строки и строки, состоящей из не более чем 15 символов и целого числа. Части элемента и сами элементы разделены пробелами.

Внешнее представление выходных данных: таблица, каждый элемент которой состоит из 8-символьной строки и строки, состоящей из не более чем 15 символов и целого числа.

1. **Текст программы**

#include <stdio.h>

#include <windows.h>

const UCHAR Emax = 100, Kmax = 9, Nmax = 15;

struct field

{

UINT quantity = 0;

char key[Kmax]{ }, name[Nmax]{ };

};

struct table

{

UINT size = 0;

field elem[Emax]{ };

bool empty()

{

return size == 0;

}

int binary\_search(field f)

{

UINT l = 0, r = size, m = 0;

for (; l < r; m = (l + r) / 2)

if (strcmp(f.key, elem[m].key) > 0)

l = m + 1;

else

r = m;

return l;

}

void shift(int pos)

{

size++;

for (int i = size - 1; i >= pos; i--)

elem[i + 1] = elem[i];

}

void insert(field f)

{

if (size > Emax)

{

printf\_s("Таблица полна.");

return;

}

int pos = binary\_search(f);

if (strcmp(f.key, elem[pos].key) == 0)

elem[pos].quantity += f.quantity;

else

{

shift(pos);

elem[pos] = f;

}

}

int find(field f)

{

if (empty()) printf\_s("Таблица пуста.");

int pos = binary\_search(f);

if (strcmp(f.key, elem[pos].key) != 0)

{

printf\_s("Элемента нет в таблице. ");

return -1;

}

printf\_s("Индекс элемента: %u ", pos);

return pos;

}

void input()

{

FILE \*fp = NULL;

fopen\_s(&fp, "WORK.txt", "r");

if (!fp)

{

printf\_s("Файл WORK.txt не найден.");

return;

} else

{

field f;

for (; fscanf\_s(fp, "%s", f.key, \_countof(f.key)) != EOF

&& size < Emax; insert(f))

fscanf\_s(fp, "%s %u", f.name, \_countof(f.name), &f.quantity);

fclose(fp);

}

}

void output()

{

FILE \*fp = NULL;

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

if (!fp)

{

printf\_s("Не удалось открыть файл output.txt.");

return;

} else

{

if (empty()) printf\_s("Таблица пуста.");

else

for (UCHAR i = 0; i < size; i++)

fprintf\_s(fp, "%9s %15s %10u \n", elem[i].key, elem[i].name, elem[i].quantity);

fclose(fp);

}

}

};

int main()

{

table t;

t.input();

t.output();

return 0;

}

1. **Набор тестов**

|  |  |  |
| --- | --- | --- |
| **№** | **Входные данные** | **Назначение** |
| **1** |  | Пустой файл. |
| **2** | 00000002 Bun 35  00000003 Donut 10  00000001 Bread 2  00000005 Bagel 12  00000007 Cake 25  00000001 Bread 3  00000001 Bread 5  00000006 Pie 25 | Есть повторяющиеся ключи, ключи расположены беспорядочно. |
| **3** | 00000001 Bread 30  00000003 Donut 15  00000008 Cookie 30  00000002 Bun 35  00000007 Cake 20  00000006 Pie 20 | Нет повторяющихся ключей, ключи расположены беспорядочно. |
| **4** | 00000008 Cookie 5  00000007 Cake 10  00000006 Pie 10  00000005 Bagel 15  00000004 Pretzel 20  00000003 Donut 20  00000002 Bun 15  00000001 Bread 10 | Ключи расположены в убывающем порядке, нет повторяющихся ключей. |
| **5** | 00000001 Bread 10  00000002 Bun 15  00000003 Donut 20  00000004 Pretzel 20  00000005 Bagel 15  00000006 Pie 10  00000007 Cake 10  00000008 Cookie 5 | Ключи расположены в возрастающем порядке, нет повторяющихся ключей. |

**5 Результаты работы программы**

|  |  |
| --- | --- |
| **№** | **Ввод/Вывод программы** |
| **1** | Таблица пуста. |
| **2** | 00000001 Bread 10  00000002 Bun 35  00000003 Donut 10  00000005 Bagel 12  00000006 Pie 25  00000007 Cake 25 |
| **3** | 00000001 Bread 30  00000002 Bun 35  00000003 Donut 15  00000006 Pie 20  00000007 Cake 20  00000008 Cookie 30 |
| **4** | 00000001 Bread 10  00000002 Bun 15  00000003 Donut 20  00000004 Pretzel 20  00000005 Bagel 15  00000006 Pie 10  00000007 Cake 10  00000008 Cookie 5 |
| **5** | 00000001 Bread 10  00000002 Bun 15  00000003 Donut 20  00000004 Pretzel 20  00000005 Bagel 15  00000006 Pie 10  00000007 Cake 10  00000008 Cookie 5 |