|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Практическая работа № 3 | | |
| по дисциплине «Информационные технологии и основы программирования» | | |
| СТРУКТУРЫ ДАННЫХ: ДЕРЕВЬЯ | | |
|  | | |
|  | Бригада №3 | Ерощенко Артем, Кауфман Яна, Ковалева МАрия |
| Группа ПМИ-33 |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

**1.** **Цель работы:** получить практические навыки работы с таблицами и использования их при решении задач.

**2. Задание:** 2(в): Элемент таблицы характеризуется тремя признаками: один – вещественное значение, второй – целое положительное число и этот признак – ключ элемента, третий – это последовательность не более чем из 10 символов. Определить тип «таблица» и написать подпрограмму удаления элемента из упорядоченной таблицы;

**3. Анализ задачи:**

**Входные данные:** пользовательский ввод с клавиатуры для выбора действия (1, 2, 3, 4) и данных для добавления в таблицу.

**Выходные данные**: Таблица, с введёнными элементами.

**Метод решения:** Создание структуры «Таблица» и «Элемент Таблицы». Функция «hashFunc» - вычисляет хэш-значение ключа, «insert» - вставляет элемент в таблицу, «get» - взятие элемента из таблицы, «del» - удаление элемента из таблицы, «freeTable» - освобождение памяти таблицы. В main-функции происходит циклическая обработка пользовательского ввода.

**4. Текст программы:**

#include <iostream>

#include <cstring>

const unsigned int cap = 5000, str\_cap = 1000;

typedef struct TableItem {

unsigned int key;

float f\_data;

char str\_data[str\_cap];

TableItem \*next;

} TableItem;

typedef struct HashTable {

TableItem \*\*items;

unsigned int size;

static HashTable \*create(unsigned int size);

unsigned int HashFunc(unsigned int key);

char insert(unsigned int key, float f\_data, char \*str\_data);

TableItem \*get(unsigned int key);

char del(unsigned int key);

char freeTable();

} HashTable;

HashTable \*HashTable::create(unsigned int size){

HashTable \*table = new HashTable();

table->size = size;

table->items = (TableItem \*\*) malloc(sizeof(TableItem\*) \* size);

for(unsigned int i = 0; i < size; i++) table->items[i] = NULL;

return table;

}

unsigned int HashTable::HashFunc(unsigned int key){

return key % this->size;

}

char HashTable::insert(unsigned int key, float f\_data, char \*str\_data){

unsigned int hash = this->HashFunc(key);

TableItem\* item = new TableItem();

item->key = key;

item->f\_data = f\_data;

item->next = NULL;

strcpy(item->str\_data, str\_data);

if(this->items[hash] != NULL){ // Collision

TableItem \*buff = this->items[hash];

while(buff->next != NULL) buff = buff->next;

buff->next = item;

} else {

this->items[hash] = item;

}

return 0;

}

TableItem \*HashTable::get(unsigned int key){

unsigned int hash = this->HashFunc(key);

return (this->items[hash] != NULL) ? this->items[hash]: NULL;

}

char HashTable::del(unsigned int key){

unsigned int hash = this->HashFunc(key);

if(this->items[hash] != NULL){

TableItem \*buff = this->items[hash];

while(buff != NULL){

free(buff);

buff = buff->next;

}

this->items[hash] = NULL;

return 0;

}

return 1;

}

char HashTable::freeTable(){

for(unsigned int i = 0; i < this->size; i++){

free(this->items[i]);

}

free(this->items);

free(this);

return 0;

}

int main(){

HashTable \*table = HashTable::create(cap);

char inp\_buff, str\_buff[str\_cap];

float f\_buff;

unsigned int uint\_buff;

TableItem \*item;

while(1){

std::cout << "Enter 1 - to insert, 2 - to get, 3 - to delete, 4 - to exit\n";

std::cin >> inp\_buff;

if(inp\_buff == '1'){

std::cout << "Enter (key, float value and string) in format \"\%d \%f \%10s\":\n";

if(scanf("%u %f %s", &uint\_buff, &f\_buff, str\_buff) == 3 && strlen(str\_buff) <= 10){

table->insert(uint\_buff, f\_buff, str\_buff);

} else {

std::cout << "Wrong format\n";

}

} else if(inp\_buff == '2'){

std::cout << "Enter key:\n";

std::cin >> uint\_buff;

item = table->get(uint\_buff);

while (item != NULL){

printf("%u: %f | %s -> ", uint\_buff, item->f\_data, item->str\_data);

item = item->next;

}

std::cout << "NULL\n";

} else if(inp\_buff == '3'){

std::cout << "Enter key:\n";

std::cin >> uint\_buff;

table->del(uint\_buff);

} else if(inp\_buff == '4'){

break;

}

}

table->freeTable();

return 0;

}

**5. Тестирование программы:**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Выходные данные | Примечание |
| 1 | 1 1 4.5 abc | - | Добавление элемента |
| 2 | 1 2 9.3 fde | - | Добавление второго элемента |
| 3 | 1 2 4.6 dse | - | Намеренное создание коллизии |
| 4 | 1 5 9.5 abcdjfkglrehgtyjf | Wrong format | Длинна строки ≤ 10 |
| 5 | 2 1 | 1: 4.500000 | abc -> NULL | Получение элемента с ключем 1 |
| 6 | 2 2 | 2: 9.300000 | fde -> 2: 4.600000 | dse -> NULL | Получение элемента с ключем 2 |
| 7 | 3 1 | - | Удаление элемента с ключем 2 |
| 8 | 2 1 | NULL | Получение элемента с ключем 1 |

**6. Вывод**

Задача решена верно, что подтверждается результатами теста.