МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Індивідуальне завдання № 3

з курсу «Алгоритми і структури даних»

на тему: «Абстрактний тип даних»

Виконала: студентка 2 курсу групи МФ-22

напряму підготовки (спеціальності)

Комп’ютерні науки

122 Комп’ютерні науки

Мухачова В.В.

Харків – 2024

Завдання:

Розробіть систему для реєстрації гостей на подію, Реалізувавши АТД Відображення (Map) на базі хеш-таблиці у вигляді масиву списків, елементи яких є парами. Про кожного гостя зберігається інформація: унікальний ID, місто з якого прибув, телефон.

Аналіз коду:

```
typedef struct Guest
{
    int id;
    char first_name[name_length];
    char last_name[name_length];
    char city[city_length];
    char phone[phone_length];
    struct Guest *next;
}Guest;
```

Структура елемента списка з гостями: зберігає id, ім’я, місто та номер телефону.

```
typedef struct HashTable
{
    Guest *table[table_size];
}HashTable;
```

Структура хеш-таблиці: зберігає масив вказівників на Guest.

```
int Menu()
{
    int i = 0;
    printf("Welcome to the menu!\n");
    printf("What whould you like to do?\n");
    printf("1. Output the whole table.\n");
    printf("2. Find a guest by ID.\n");
    printf("3. Delete guest by ID.\n");

    i = get_number_in_range (1, 3);
    return i;
}
```

Функція Menu виводить можливі функції для користувача.

```c
void Work(int i, HashTable *ht)
{
    int id = 0;
    switch (i)
    {
        case 1:
            printf("Table:\n");
            print_table(ht);
            break;
        case 2:
            printf("Input the ID:\n");
            id = get_number_in_range(0, __INT_MAX__);
            Guest *guest = find_guest(ht, id);
            if (guest)
            {
                printf("Found - ID: %d, Name: %s %s, City: %s, Phone: %s\n",
                    guest->id, guest->first_name, guest->last_name, guest->city, guest->phone);
            }
            else
            {
                printf("Guest not found.\n");
            }
            break;
        case 3:
            printf("Input the ID:\n");
            id = get_number_in_range(0, __INT_MAX__);
            printf("Deleting Guest with ID %d...\n", id);
            delete_guest(ht, id);
            break;
        default:
            printf("Sorry, that option is not found\n");
```

Фунція Work обробляє можливі функції для користувача.

```c
//хеш-функція
unsigned int hash(int id)
{
    return id % table_size;
}
```

Функція hash отримує ключ і повертає індекс для кожного елемента.

```c
//ініціалізація хеш-таблиці
HashTable* create_table()
{
    HashTable *ht = (HashTable*)malloc(sizeof(HashTable));
    for (int i = 0; i < table_size; i++)
    {
        ht->table[i] = NULL;
    }
    return ht;
}
```

Функція create_table ініціалізує таблицю.

```c
//додавання гостя
void add_guest(HashTable *ht, int id, const char *first_name, const char *last_name, const char *city, const char *phone)
{
    int index = hash(id);
    Guest *new_guest = (Guest*)malloc(sizeof(Guest));
    new_guest->id = id;
    strncpy(new_guest->first_name, first_name, name_length);
    strncpy(new_guest->last_name, last_name, name_length);
    strncpy(new_guest->city, city, city_length);
    strncpy(new_guest->phone, phone, phone_length);
    new_guest->next = ht->table[index];
    ht->table[index] = new_guest;
}
```

Функція add_guest приймає на вхід дані гостя, на основі ID знаходить індекс нового елемента і додає його до хеш-таблиці.

```c
// Функція для читання гостей з файлу
void read_guests_from_file(const char *filename, HashTable *ht)
{
    FILE *file = fopen(filename, "r");
    if (file==NULL)
    {
        perror("Can't open the file.\n");
        return;
    }

    int id = 0;
    char city[city_length];
    char phone[phone_length];
    char first_name[name_length];
    char last_name[name_length];
    char line[city_length];
    while (fgets(line, sizeof(line), file)!=NULL)
    {
        if (sscanf(line,  "%d %49s %49s %49s %15s", &id, first_name, last_name, city, phone) == 5)
        {
            add_guest(ht, id, first_name, last_name, city, phone);
        }
        else
        {
            printf("Error during manipulating the line: %s\n", line);
        }
    }

    fclose(file);
}
```

Функція read_guests_from_file читає файл і з даних кожної строки створює нового гостя.

```c
//вивід усіх гостей
void print_table(HashTable *ht)
{
    for (int i = 0; i < table_size; i++)
    {
        Guest *current = ht->table[i];
        printf("Index %d:\n", i);
        while (current!=NULL)
        {
            printf("    ID: %d, Name: %s %s, City: %s, Phone: %s\n",
                current->id, current->first_name, current->last_name, current->city, current->phone);
            current = current->next;
        }
    }
}
```

Функція print_table виводить усіх гостей.

```c
//пошук гостя за ID
Guest* find_guest(HashTable *ht, int id)
{
    int index = hash(id);
    Guest *current = ht->table[index];
    while (current!=NULL)
    {
        if (current->id == id)
        {
            return current;
        }
        current = current->next;
    }
    return NULL;
}
```

Функція find_guest знаходить гостя за id, повертає вказівник на гостя.

```c
//видалення гостя
void delete_guest(HashTable *ht, int id)
{
    int index = hash(id);
    Guest *current = ht->table[index];
    Guest *prev = NULL;
    while (current)
    {
        if (current->id == id)
        {
            if (prev!=NULL)
            {
                prev->next = current->next;
            }
            else
            {
                ht->table[index] = current->next;
            }
            free(current);
            printf("Guest with ID %d deleted.\n", id);
            printf("Table After Deletion:\n");
            print_table(ht);
            return;
        }
        prev = current;
        current = current->next;
    }
    printf("Guest with ID %d not found.\n", id);
}
```

Функція delete_guest видаляє гостя та виводить таблицю після видалення.

```c
//звільнення пам'яті
void free_table(HashTable *ht)
{
    for (int i = 0; i < table_size; i++)
    {
        Guest *current = ht->table[i];
        while (current!=NULL)
        {
            Guest *temp = current;
            current = current->next;
            free(temp);
        }
    }
    free(ht);
}
```

Функція free_table звільнює пам’ять для таблиці.

Приклади роботи програми:

AiSD > indiv3 > output > guests.txt

```
106 Ivan Petrenko Kyiv +380671234567
57 Olena Shevchenko Lviv +380632345678
305 Andriy Ivanchuk Odessa +380931234567
42 Oksana Mykhailyuk Dnipro +380503456789
73 Taras Hrytsenko Chernihiv +380509876543
84 Anastasiya Koval Rivne +380671112223
97 Dmytro Sydorenko Kharkiv +380638765432
115 Kateryna Marchenko Zhytomyr +380931234555
123 Mykola Radchenko Poltava +380507654321
141 Svitlana Bondarenko Chernivtsi +380506789012
162 Pavlo Kryvoruchko Ternopil +380633213141
170 Viktoriya Lysenko Uzhhorod +380678901234
189 Oleksandr Vasylenko Kharkiv +380639876123
194 Inna Zhuk Kropyvnytskyi +380504321678
201 Bohdan Melnyk Lutsk +380502345678
```

Файл guests.txt

```
Welcome to the menu!
What whould you like to do?
1. Output the whole table.
2. Find a guest by ID.
3. Delete guest by ID.
1
Table:
Index 0:
    ID: 170, Name: Viktoriya Lysenko, City: Uzhhorod, Phone: +380678901234
Index 1:
    ID: 201, Name: Bohdan Melnyk, City: Lutsk, Phone: +380502345678
    ID: 141, Name: Svitlana Bondarenko, City: Chernivtsi, Phone: +380506789012
Index 2:
    ID: 162, Name: Pavlo Kryvoruchko, City: Ternopil, Phone: +380633213141
    ID: 42, Name: Oksana Mykhailyuk, City: Dnipro, Phone: +380503456789
Index 3:
    ID: 123, Name: Mykola Radchenko, City: Poltava, Phone: +380507654321
    ID: 73, Name: Taras Hrytsenko, City: Chernihiv, Phone: +380509876543
Index 4:
    ID: 194, Name: Inna Zhuk, City: Kropyvnytskyi, Phone: +380504321678
    ID: 84, Name: Anastasiya Koval, City: Rivne, Phone: +380671112223
Index 5:
```

Вивід усіх гостей.

```
Welcome to the menu!
What whould you like to do?
1. Output the whole table.
2. Find a guest by ID.
3. Delete guest by ID.
2
Input the ID:
189
Found - ID: 189, Name: Oleksandr Vasylenko, City: Kharkiv, Phone: +380639876123
Would you like to continue? Enter 'y' if yes, 'n' if no
```

Приклад пошуку гостя, ввід існуючого ID.

```
Welcome to the menu!
What whould you like to do?
1. Output the whole table.
2. Find a guest by ID.
3. Delete guest by ID.
2
Input the ID:
1111111
Guest not found.
```

Приклад пошуку гостя, ввід неіснуючого ID.

```
Welcome to the menu!
What whould you like to do?
1. Output the whole table.
2. Find a guest by ID.
3. Delete guest by ID.
3
Input the ID:
170
Deleting Guest with ID 170...
Guest with ID 170 deleted.
Table After Deletion:
Index 0:
Index 1:
```

Приклад видалення гостя, ввід існуючого ID.

```
What whould you like to do?
1. Output the whole table.
2. Find a guest by ID.
3. Delete guest by ID.
3
Input the ID:
11111
Deleting Guest with ID 11111...
Guest with ID 11111 not found.
```

Приклад видалення гостя, ввід неіснуючого ID.

Оригинальний код програми:

```c
// Розробіть систему для реєстрації гостей на подію,
// Реалізувавши АТД Відображення (Map) на базі
// хеш-таблиці у вигляді масиву списків, елементи яких
// є парами. Про кожного гостя зберігається інформація:
// унікальний ID, місто з якого прибув, телефон.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define table_size 10
#define name_length 50
#define city_length 50
#define phone_length 15
#define temp_length 50

typedef struct Guest
{
    int id;
    char first_name[name_length];
    char last_name[name_length];
    char city[city_length];
    char phone[phone_length];
    struct Guest *next;
}Guest;

typedef struct HashTable
{
    Guest *table[table_size];
```

```c
}HashTable;

int Menu();
void Work(int i, HashTable *ht);
unsigned int hash(int id);
HashTable* create_table();
void add_guest(HashTable *ht, int id,const char *first_name, const char
*last_name, const char *city, const char *phone);
void read_guests_from_file(const char *filename, HashTable *ht);
void print_table(HashTable *ht);
Guest* find_guest(HashTable *ht, int id);
void delete_guest(HashTable *ht, int id);
void free_table(HashTable *ht);
int get_number_in_range(int left, int right);


int main()
{
    HashTable *ht = create_table();

    read_guests_from_file("guests.txt", ht);

    int menu_switch;
    char ch = 'y';
    while (ch=='y'||ch=='Y')
    {
        menu_switch = Menu();
        Work(menu_switch, ht);
        printf("Would you like to continue? Enter 'y' if yes, 'n' if no \n");
        ch = getchar();
        getchar();
    }

    free_table(ht);
    return 0;
}

int Menu()
{
```

```
    int i = 0;
    printf("Welcome to the menu!\n");
    printf("What whould you like to do?\n");
    printf("1. Output the whole table.\n");
    printf("2. Find a guest by ID.\n");
    printf("3. Delete guest by ID.\n");

    i = get_number_in_range (1, 3);
    return i;
}

void Work(int i, HashTable *ht)
{
    int id = 0;
    switch (i)
    {
        case 1:
            printf("Table:\n");
            print_table(ht);
            break;
        case 2:
            printf("Input the ID:\n");
            id = get_number_in_range(0, __INT_MAX__);
            Guest *guest = find_guest(ht, id);
            if (guest)
            {
                printf("Found - ID: %d, Name: %s %s, City: %s, Phone: %s\n",
                    guest->id, guest->first_name, guest->last_name, guest->city,
guest->phone);
            }
            else
            {
                printf("Guest not found.\n");
            }
            break;
        case 3:
            printf("Input the ID:\n");
            id = get_number_in_range(0, __INT_MAX__);
            printf("Deleting Guest with ID %d...\n", id);
```

```
            delete_guest(ht, id);
            break;
        default:
            printf("Sorry, that option is not found\n");

    }
}

//хеш-функція
unsigned int hash(int id)
{
    return id % table_size;
}

//ініціалізація хеш-таблиці
HashTable* create_table()
{
    HashTable *ht = (HashTable*)malloc(sizeof(HashTable));
    for (int i = 0; i < table_size; i++)
    {
        ht->table[i] = NULL;
    }
    return ht;
}

//додавання гостя
void add_guest(HashTable *ht, int id, const char *first_name, const char
*last_name, const char *city, const char *phone)
{
    int index = hash(id);
    Guest *new_guest = (Guest*)malloc(sizeof(Guest));
    new_guest->id = id;
    strncpy(new_guest->first_name, first_name, name_length);
    strncpy(new_guest->last_name, last_name, name_length);
    strncpy(new_guest->city, city, city_length);
    strncpy(new_guest->phone, phone, phone_length);
    new_guest->next = ht->table[index];
    ht->table[index] = new_guest;
}
```

```c
// Функція для читання гостей з файлу
void read_guests_from_file(const char *filename, HashTable *ht)
{
    FILE *file = fopen(filename, "r");
    if (file==NULL)
    {
        perror("Can't open the file.\n");
        return;
    }

    int id = 0;
    char city[city_length];
    char phone[phone_length];
    char first_name[name_length];
    char last_name[name_length];
    char line[city_length];
    while (fgets(line, sizeof(line), file)!=NULL)
    {
        if (sscanf(line,  "%d %49s %49s %49s %15s", &id, first_name, last_name,
city, phone) == 5)
        {
            add_guest(ht, id, first_name, last_name, city, phone);
        }
        else
        {
            printf("Error during manipulating the line: %s\n", line);
        }
    }

    fclose(file);
}

//вивід усіх гостей
void print_table(HashTable *ht)
{
    for (int i = 0; i < table_size; i++)
    {
        Guest *current = ht->table[i];
```

```
        printf("Index %d:\n", i);
        while (current!=NULL)
        {
            printf("    ID: %d, Name: %s %s, City: %s, Phone: %s\n",
                current->id, current->first_name, current->last_name,
current->city, current->phone);
            current = current->next;
        }
    }
}

//пошук гостя за ID
Guest* find_guest(HashTable *ht, int id)
{
    int index = hash(id);
    Guest *current = ht->table[index];
    while (current!=NULL)
    {
        if (current->id == id)
        {
            return current;
        }
        current = current->next;
    }
    return NULL;
}

//видалення гостя
void delete_guest(HashTable *ht, int id)
{
    int index = hash(id);
    Guest *current = ht->table[index];
    Guest *prev = NULL;
    while (current)
    {
        if (current->id == id)
        {
            if (prev!=NULL)
            {
```

```
                prev->next = current->next;
            }
            else
            {
                ht->table[index] = current->next;
            }
            free(current);
            printf("Guest with ID %d deleted.\n", id);
            printf("Table After Deletion:\n");
            print_table(ht);
            return;
        }
        prev = current;
        current = current->next;
    }
    printf("Guest with ID %d not found.\n", id);
}

//звільнення пам'яті
void free_table(HashTable *ht)
{
    for (int i = 0; i < table_size; i++)
    {
        Guest *current = ht->table[i];
        while (current!=NULL)
        {
            Guest *temp = current;
            current = current->next;
            free(temp);
        }
    }
    free(ht);
}

//перевіряє, що користувач ввів саме число, і що воно попадає між заданих чисел
int get_number_in_range(int left, int right)
{
    int var = 0;
    int bad_input = 0;
```

```
int i = 0;
int negative = 0;

while(1)
{
    var = 0, bad_input = 0, i = 0, negative = 0;
    char number[temp_length];

    if (fgets(number, sizeof(number), stdin) == NULL) {
        printf("Error reading input. Please try again.\n");
        clearerr(stdin);
        continue;
    }

    //перевіряє, що користувач не натиснув тільки Enter
    if(number[0]==10)
    {
        bad_input = 1;
    }

    //перевріяє, що користувач не ввів букви, або спец знаки.
    while(number[i]!=' '&&number[i]!='\n'&&number[i]!='\0')
    {
        if ((number[i] -'0')<0||(number[i] - '0')>9)
        {
            if(i==0&&number[i]=='-')
            {
                negative = 1;
                i++;
            }
            else
            {
                bad_input = 1;
                break;
            }
        }
        else
        {
            //якщо це не буква або знак, переводить у число
```

```
                var = var*10 + (number[i] - '0');
                i++;
            }
        }

        if(negative==1)
            var = -var;

        if(bad_input==1)
            printf("Please, try again, only numbers.\n");
        else
        {
            //якщо не попадає в рамки, просить ввести знову
            if(left==right && var != right)
                printf("Please, try again and beware of the range.\n");
            else
            {
                if(var < left || var > right)
                    printf("Please, try again and beware of the range.\n");
                else
                    //якщо попадає в задані рамки виход з циклу
                    break;
            }
        }
    }

    return var;
}
```