資料結構 HW_1

110303581 陳俊瑋

```c
typedef struct KeyValue {
    char key[KEY_SIZE];
    char value[VALUE_SIZE];   //age
} KeyValue;

typedef struct Database {
    KeyValue* data;     //存一堆 KeyValue 的指標
    int size;
} Database;

Database* createDatabase() {
    Database* db = (Database*)malloc(sizeof(Database));
    db->size = 0;
    db->data = NULL;
    return db;
}
```

我先定義了一個結構叫作 KeyValue，其結構為一個 key 和一個 value，而後再定義一個結構叫做 DataBase，結構為一對 KeyValue(取名為 data)的指標和一個整數 size 用來追蹤現在有幾個 DataBase，讓之後在使用 for 迴圈在 db 裡尋找時有個界線。

之後再定義一個函式 createDatabase，之後再 main 的地方呼叫這函式以建立(初始化)一個資料庫的指標叫做 db，並用動態記憶體存取。

```c
//創建 inset
void insert(Database* db, const char* key, const char* value) {
    db->data = (KeyValue*)realloc(db->data, sizeof(KeyValue) *
(size_t)(db->size + 1));

    strcpy(db->data[db->size].key, key);
    strcpy(db->data[db->size].value, value);
    db->size++;

    printf("Insert [%s] Successfully.\n", key);
}

//讀取 read
void read(Database* db, const char* key) {
    for (int i = 0; i < db->size; i++) {
        if ((db->data[i].key, key) == 0) {
```

```
            printf("{\n");
            printf("    Name: %s\n", db->data[i].key);
            printf("    Age: %s\n", db->data[i].value);
            printf("}\n");
            return;
        }
    }
    printf("Not Found [%s] in Database.\n", key);
}

//更新 update
void update(Database* db, const char* key, const char* value) {
    for (int i = 0; i < db->size; i++) {
        if (strcmp(db->data[i].key, key) == 0) {
            strcpy(db->data[i].value, value);
            printf("Update [%s] Successfully\n", key);
            return;
        }
    }
    printf("[%s] Not Found. Cannot update.\n", key);
}


//刪除 delete
void delete(Database* db, const char* key) {
    for (int i = 0; i < db->size; i++) {
        if (strcmp(db->data[i].key, key) == 0) {
            // 將後面的資料往前移動
            for (int j = i; j < db->size - 1; j++) {
                strcpy(db->data[j].key, db->data[j + 1].key);
                strcpy(db->data[j].value, db->data[j + 1].value);
            }
            db->size--;

            printf("Deleted [%s] Successfully.\n", key);
            return;
        }
    }
    printf("Key not found. Cannot remove.\n");
}
```

這邊是用函式建立 CRUD 的操作

建立我取名叫做 insert，將 db 裡的 data 用動態記憶體配置一個空間，之後將值都帶入，並將 size 加 1。

讀取我取名叫做 read，用 for 迴圈的方式找尋資料，並用 `strcmp` 函式判斷當 key 為要尋找的目標時，將 Key 和 Value 都進行輸出。

更新我取名為 update，一樣是用 for 迴圈找尋 key，找到後將 value 進行更新。
刪除我取名為 delete，一樣用 for 迴圈找尋對應的 key，之後將其替換成後一個的值，同時將 size 減 1。

```
int main() {
    Database* db = createDatabase();
    char input[INPUT_SIZE];

    insert(db, "William", "20");
    insert(db, "Una", "19");

    do{
        printf("Enter a command (get/set/update/del): ");
        fgets(input, sizeof(input), stdin);
        input[strcspn(input, "\n")] = '\0';

        char command[COMMAND_SIZE];
        char key[KEY_SIZE];
        char value[VALUE_SIZE];

        if (sscanf(input, "%s %s %s", command, key, value) == 3)
        {
            if (strcmp(command, "set") == 0) {
                //SET
                insert(db, key, value);
            }
            else if (strcmp(command, "update") == 0)
            {
                //UPDATE
                update(db, key, value);
            }
            else {
                printf("Invalid command\n");
            }
        }

        else if(sscanf(input, "%s %s", command, key) == 2){
            if (strcmp(command, "get") == 0) {
                // GET
                read(db, key);
            }

            else if (strcmp(command, "del") == 0) {
                // DEL
                delete(db, key);
```

```
            }

            else {
                printf("Invalid command\n");
            }

        }



        else if (strcmp(input, "0") == 0)
        {
            printf("END");
            break;
        }

        else{
            printf("Invalid command\n");
        }


    }while (1);
```

一開始先初始化資料庫，再定義一個陣列(input)用來存取使用者的輸入。這裡我先建立兩筆資料在資料庫裏面。而後建立一個 do-while 迴圈讓使用者可以一直輸入新的指令，其中利用 `fgets` 函式抓取輸入並將尾巴的\n 替換成\0。
再利用 `sscanf` 去判斷輸入為三個 string 或是兩個 string，將值帶入 `command`, `key` 和 `value`。`strcmp` 是看 `command` 為 set、get、update 或是 del，之後呼叫對應的函式進行操作。同時我也設置一些防呆機制，在無法判斷使用者輸入時也做出 `printf("Invalid command\n");`
而當使用者輸入 0 時，則會跳出 do-while 迴圈。

```
    free(db->data);
    free(db);

    return 0;
}
```

最後將記憶體 free 掉並結束整個程式。

輸出結果:

```
Insert [Una] Successfully.
Enter a command (get/set/update/del): get Una
{
    Name: Una
    Age: 19
}
Enter a command (get/set/update/del): set mimi 18
Insert [mimi] Successfully.
Enter a command (get/set/update/del): get mimi
{
    Name: mimi
    Age: 18
}
Enter a command (get/set/update/del): del mimi
Deleted [mimi] Successfully.
Enter a command (get/set/update/del): get mimi
Not Found [mimi] in Database.
Enter a command (get/set/update/del): 0
END
```

可以看到程式裡面有先存取`[William]`和`[Una]`這兩筆資料，所以會 print

```
Insert [William] Successfully.
Insert [Una] Successfully.
```

之後輸入 `get Una` 得到對應的數據，再輸入 `set mimi 18` 設置一個數據，也可以看到也成功輸出。

後面將`[mimi]`從資料庫刪除，再嘗試可不可以用 get 去到這筆資料，而程式也成功顯示 `Not Found [mimi] in Database.`。

最後，輸入 0 以結束程式的 while 迴圈。