Bảng ký hiệu

ADT (Abstract Data Type) – Kiểu dữ liệu trừu tượng

ADT: trừu tượng vì đặc tả bên trong đối tượng được ẩn đi từ thao tác không liên quan

Cặp key-value (khóa – giá trị)

- Insert (chèn) một giá trị với một khóa cụ thể
- Đưa ra một khóa, search (tìm kiếm) giá trị tương ứng của nó

Ví dụ: DNS

- Chèn URL với IP cụ thể
- Đưa ra URL => tìm IP tương ứng

URL	IP address
www.cs.princeton.edu	128.112.136.11
www.princeton.edu	128.112.128.15
www.yale.edu	130.132.143.21
www.harvard.edu	128.103.060.55
www.simpsons.com	209.052.165.60
key	value

Có thể thay đổi vai trò: đưa ra một IP, tìm URL tương ứng

Các ứng dụng

Application	Purpose	Key	Value
Phone book	Look up phone number	Name	Phone number
Bank	Process transaction	Account number	Transaction details
File share	Find song to download	Name of song	Computer ID
File system	Find file on disk	Filename	Location on disk
Dictionary	Look up word	Word	Definition
Web search	Find relevant documents	Keyword	List of documents
Book index	Find relevant pages	Keyword	List of pages
Web cache	Download	Filename	File contents
Genomics	Find markers	DNA string	Known positions
DNS	Find IP address given URL	URL	IP address
Reverse DNS	Find URL given IP address	IP address	URL
Compiler	Find properties of variable	Variable name	Value and type
Routing table	Route Internet packets	Destination	Best route

Danh bạ điện thoại

- Định nghĩa một cấu trúc lưu cặp: tên – số điện thoại. Trong đó khóa là tên, giá trị là số điện thoại

```
typedef struct {
    char name[80];
    long number;
} PhoneEntry;
```

Sử dụng mảng để cài đặt

- Cặp key-value được lưu trong một mảng có thứ tự (đã sắp xếp) như sau:

```
typedef struct {
    PhoneEntry * entries;
    int total;
    int size;
} PhoneBook;
```

- Bộ nhớ lưu trữ danh bạ: cấp phát động dựa trên số lượng tối đa các cặp (entity)
- Khi số lượng các cặp (entity) vượt quá số lượng tối đa ban đầu, bộ nhớ sẽ được cấp phát lại với kích thước tối đa mới

API

- Định nghĩa 2 hằng thể hiện số lượng tối đa ban đầu, và độ tăng mỗi khi cần cấp phát lại

```
#define INITIAL_SIZE 10
#define INCREMENTAL_SIZE 5
```

- Định nghĩa 2 hàm tạo danh bạ mới và loại bỏ danh bạ

```
PhoneBook  createPhoneBook();
void  dropPhoneBook(PhoneBook* book);
```

API

- Hàm thêm một mục (cặp – entity) vào danh bạ

void addPhoneNumber(char * name, long number, PhoneBook* book);

Chú ý: nếu mục này đã tồn tại, giá trị này sẽ ghi đè

- Tìm một cặp trong danh bạ

PhoneEntry * getPhoneNumber(char * name, PhoneBook book);

Trả về NULL nếu mục này không tồn tại

Bài 1

- Hoàn thành các API đã mô tả cho danh bạ điện thoại
- Viết chương trình danh bạ điện thoại sử dụng API đã đưa ra

Gợi ý: các hàm cần có

1. PhoneBook createPhoneBook();
2. void dropPhoneBook(PhoneBook* book);
3. void addPhoneNumber(char * name, long number, PhoneBook* book);
4. PhoneEntry * getPhoneNumber(char * name, PhoneBook book);

Gợi ý

- 1 . PhoneBook createPhoneBook()

  - Khai báo PhoneBook p

  - Sử dụng malloc để cấp phát vùng nhớ của thành phần entries cho p

  - Gán giá trị khởi tạo cho p

- 2. void dropPhoneBook(PhoneBook* book);

  - Giải phóng vùng nhớ của entries trong p

Gợi ý

- 3'. int binarySearch(PhoneEntry* entries, int l, int r, char * name, int* found)

  - found = 1 (tìm thấy) trả về chỉ số tìm được

  - found = 0 (không tìm thấy) trả về chỉ số cần chèn

- 3. void addPhoneNumber(char * name, long number, PhoneBook* book);

  - gọi hàm binarySearch

  - if (found==1) cặp nhật lại số điện thoại

  - else sử dụng memcpy để di chuyển dữ liệu và chèn cặp (name, number)

Gợi ý

- 4. PhoneEntry * getPhoneNumber(char * name, PhoneBook book);

  - pos = binarySearch(PhoneEntry* entries, int l, int r, char * name, int* found)

  - if (found==1) return &book.entries[pos]

  - else return NULL

Hàm main

```
PhoneBook book;
//Khoi tao & chen
book = createPhoneBook();
addPhoneNumber(“Do Lam”, 909090, &book);
addPhoneNumber(“Tuan Dung”, 929292, &book);
addPhoneNumber(“Anh Thang”, 919191, &book);
addPhoneNumber(“Ngo Phong”, 949494, &book);
//Hien thi danh sach book
//Tim kiem
PhoneEntry entry=PhoneEntry(“Do Lam”, book);
if(entry==NULL) printf(“Khong tim thay\n”);
else printf(“%s\n”, entry.number);
```

Bảng ký hiệu tổng quát

- Định nghĩa một cấu trúc tổng quát các mục

```
typedef struct {
    void * key;
    void * value;
} Entry;
```

- Định nghĩa một cấu trúc tổng quát cho bảng ký hiệu

```
typedef struct {
    Entry * entries;
    int size, total;
    Entry (*makeNode)(void*, void*);
    int (*compare)(void*, void*);
} SymbolTable;
```

makeNode là một con trỏ hàm, trỏ tới hàm tạo một nuút ới cặp khóa và giá trị được truyền

compare là một con trỏ hàm, trỏ tới hàm so sánh hai khóa

API

```
#define INITIAL_SIZE 100
#define INCREMENTAL_SIZE 10
SymbolTable createSymbolTable(
        Entry (*makeNode)(void*, void*),
        int (*compare)(void*, void*)
);
void dropSymbolTable(SymbolTable* tab);
void addEntry(void* key, void* value, SymbolTable*
    book);
Entry* getEntry(void* key, SymbolTable book);
```

Lưu ý: Giải phóng vùng nhớ được cấp phát cho mỗi mục khi bảng được loại bỏ

Ví dụ

```
Entry makePhone(void* name, void* phone) {
    Entry res;
    res.key = strdup( (char*)name );
    res.value = malloc(sizeof(long));
    memcpy( res.value, phone, sizeof(long) );
    return res;
}
int comparePhone(void * key1, void* key2) {
    return strcmp((char*)key1, (char*)key2);
}

SymbolTable phoneBook = createSymbolTable(makePhone,
    comparePhone);
long number = 983984775;
char name[] = “Ta Tuan Anh”;
addEntry(name, &number, &phoneBook);
```

Bài 2

- Viết lại chương trình danh bạ điện thoại sử dụng bảng kí hiệu tổng quát