Symbol Tables

Instructor: Thanh-Chung Dao
(chungdt@soict.hust.edu.vn)
Slides by Dr. Ta Tuan Anh

1

1

ADT

Key-value pair abstraction.

- Insert a value with specified key.
- Given a key, search for the corresponding value.

Example: DNS lookup.

- Insert URL with specified IP address.
- Given URL, find corresponding IP address

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

Can interchange roles: given IP address find corresponding URL

2

2

Example applications

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

3

3

Phone Book

- Define a structure to store name and phone number pair. The key is name and the value is number.

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

4

4

Using array for implementation

- Key-value pairs are stored in an **ordered array** as the following

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

- The memory to store the entries should be allocated dynamically according to the maximal size of the phone book.
- When the total number of entries exceeds the maximal size, the memory have to be reallocated with a new maximal size

5

5

API

- Let define two constant numbers to represent the initial size of a phone book and the gap size of the array each to time to be reallocated

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

- Write two functions to create a new phone book and drop the entries of a phone book as the following

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

6

6

API

- Add an entry in the phone book

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

NB: If the entry exists, the value should be overwritten.

- Find an entry in the phone book

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

returns null if the entry does not exist

7

7

Quiz 1

- Complete the API specified for the phone book.
- Write a phone book program using the given API

8

8

Hint

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

9

9

Hint

- 1 . PhoneBook createPhoneBook()
  - declare PhoneBook p
  - using malloc to allocate memory for p
  - assign beginning values to p
- 2. void dropPhoneBook(PhoneBook* book);
  - free memory in p
  - assign ending values to p

10

10

Hint

- 3'. int binarySearch(PhoneEntry* entries, int l, int r, char * name, int* found)
  - found = 0 or 1
  - return index to insert name if we don't find or the found index
- 3. void addPhoneNumber(char * name, long number, PhoneBook* book);
  - call binarySearch
  - if (found==1) update the number
  - else, using memcpy to move copy data
  - insert new (name, number) pair

11

11

Hint

- 4. PhoneEntry * getPhoneNumber(char * name, PhoneBook book);
  - call pos = binarySearch(PhoneEntry* entries, int l, int r, char * name, int* found)
  - if (found==1) return &book.entries[pos]
  - else return NULL

12

12

Solution

- phonebook.c

13

13

Generic symbol tables

- Define a generic structure for entries

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

- Define a generic structure for symbol tables

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

makeNode is a function pointer to refer to a function to create a node with a key and a value passed

compare is a function to refer to a function to compare two keys

14

14

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);
```

NB: Free the memory allocated for each entry when a table is dropped

15

15

Example

```
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);
```

16

16

Quiz 2

- Rewrite the phone book program using the generic symbol table

17

17

Solution

- phonebook_generic.c

18

18