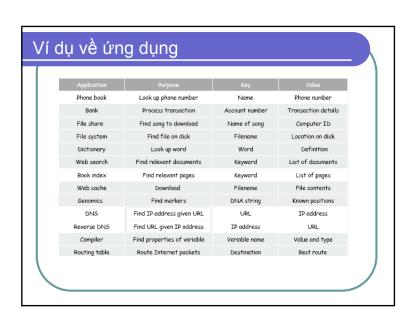
# Bảng ký hiệu (Symbol table)

Created by <a href="mailto:anhtt-fit@mail.hut.edu.vn">anhtt-fit@mail.hut.edu.vn</a>
Updated by <a href="mailto:hut.edu.vn">hut.edu.vn</a>



# Cặp khóa-giá trị trừ tượng (key-value abstraction) Chèn 1 giá trị trừ tượng (key-value abstraction) Chèn 1 giá trị với khóa xác định Cho 1 khóa, tìm giá trị tương ứng Ví dụ: Tìm DNS. Chèn URL với địa chỉ IP xác định Cho URL, tìm địa chỉ IP tương ứng URL IP address VMW. Sprinceton. edu 128.112.136.11 VMW. princeton. edu 128.112.136.11 VMW. princeton. edu 128.103.060.55 VMW. simpsons. com 209.052.165.60 key value Có thể thay đổi vai trò: cho địa chỉ IP, tìm URL tương ứng.

#### Cài đặt

• Định nghĩa cấu trúc lưu trữ cặp khóa-giá trị

Ví dụ: phonebook typedef struct { long number; char name[80] } PhoneEntry;

- Khóa:số điện thoại
- Giá trị: tên người

## Phép cài đặt sử dụng mảng

 Các cặp khóa-giá trị được lưu trong mảng đã sắp xếp

```
Ví dụ:
#define MAX_PHONE_NUMBER 1000
typedef struct {
    PhoneEntry entries[MAX_PHONE_NUMBER];
    int total;
} PhoneBook;
```

#### **API**

- Thêm 1 mục tin vào sổ điện thoại
   void addPhoneNumber(long number, char \* name, PhoneBook\* book);
   Nếu đã có mục này, giá tri sẽ bị ghi đè
- Tìm 1 mục tin trong sổ điện thoại
   char \* getPhoneNumber(long number, const PhoneBook\* book);
   Trả về null nếu không có mục này

#### Bài tập 1

 Viết chương trình thêm và tìm kiếm các số điện thoại trong sổ điện thoại. Chương trình cài đặt sử dụng mảng.

# Sử dụng mảng động

 Mảng lưu trữ các mục tin cần lưu trữ theo kiểu cấp phát động (do kích thước của sổ điện thoại)

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

Khi tổng số bản ghi vượt quá kích thước size, các mục tin trong bộ nhớ cần phải khai báo lại kích thước

```
Nhắc lại: cấp phát mảng động
int *ip;
ip = (int *) malloc(100*sizeof(int));
```

#### **API**

#define INITIAL\_SIZE 100 k /thuoc ban dau #define INCREMENTAL\_SIZE 10 kich thuoc Ion them

• Tạo 1 sổ điện thoại với kích thước khởi tạo.

PhoneBook createPhoneBook();

Xóa 1 phone book
 void dropPhoneBook(PhoneBook\* book);

## Bài tập 2

 Viết lại chương trình phone book sử dụng mảng động

### Tổng quát hóa bảng ký hiệu

```
Định nghĩa cấu trúc tổng quát cho các mục typedef struct {
void * key;
void * value;
} Entry;
Định nghĩa cấu trúc tổng quát cho các bảng ký hiệu typedef struct {
Entry * entries;
int size, total;
Entry (makeNode*)(void*, void*);
int (compare*)(void*, void*);
} SymbolTable;
makeNode là hàm con trỏ trỏ đến hàm tạo nút với các tham số đầu vào là khóa và giá trị
compare là hàm so sánh 2 khóa
```

#### **API**

Chú ý: Cần giải phóng bộ nhớ cho mỗi mục tin khi xóa bảng

# Example

```
Entry makePhoneBook(void* phone, void* name) {
    Entry res;
    res.key = malloc(sizeof(int));
    memcpy( res.key, phone, sizeof(int) );
    res.value = strdup( (char*)name );
    return res;
}
int comparePhone(void * key1, void* key2) {
    int num1 = *( (int*) key1 );
    int num2 = *( (int*) key2 );
    if (num1==num2) return 0;
    else if (num1 < num2) return -1;
    else return 1;
}
SymbolTable phoneBook = createSymbolTable(makePhoneBook, comparePhone);
```

# Bài tập về nhà

 Tạo 1 bảng ký hiệu sử dụng cây tìm kiếm nhị phân (binary search tree), sau đó dùng cấu trúc dữ liệu này để viết chương trình phone book.

## Bài tập 3

 Viết lại chương trình phone book sử dụng bảng ký hiệu tổng quát