



ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# LẬP TRÌNH C CƠ BẢN

## Danh sách tuyển tính

# Nội dung

- Cài đặt các thao tác cơ bản trên danh sách liên kết đơn
- Bài toán quản lý hồ sơ
- Cài đặt bằng danh sách liên kết đơn
- Cài đặt bằng danh sách liên kết đôi

# Các thao tác cơ bản trên danh sách liên kết đơn

- Mỗi nút của danh sách liên kết đơn có cấu trúc sau

```
typedef struct Node{  
    int value;  
    struct Node* next;  
}Node;
```

- Cài đặt các hàm
  - Node\* insertLast(Node\* h, int v);// insert a node at that last position
  - Node\* removeFirst(Node\* h, int v);// remove a first node having value v
  - Node\* removeAll(Node\* h, int v);// remove all nodes having value v
  - int count(Node\* h);// count number of nodes
  - Node\* reverse(Node\* h);// reverse the linked list

# Các thao tác cơ bản trên danh sách liên kết đơn

- Định nghĩa cấu trúc dữ liệu

```
#include <stdio.h>

typedef struct Node{
    int value;
    struct Node* next;// point to the next
    //element of the current element
}Node;

Node*makeNode(int v){// allocate memory for a new node
    Node* p = (Node*)malloc(sizeof(Node));
    p->value = v; p->next = NULL;
    return p;
}
```

# Các thao tác cơ bản trên danh sách liên kết đơn

- Chèn thêm 1 nút vào cuối danh sách (sử dụng và không sử dụng đệ quy)

```
Node* insertLast(Node* h, int v){
    Node* p = h;
    if(h == NULL){
        return makeNode(v);
    }
    // general case
    while(p->next != NULL)
        p = p->next;

    Node* q = makeNode(v);
    p->next = q;
    return h;
}
```

```
Node* insertLastRecursive(Node* h, int v){
    if(h == NULL){
        return makeNode(v);
    }
    h->next = insertLastRecursive(h->next, v);
    return h;
}
```

# Các thao tác cơ bản trên danh sách liên kết đơn

- Loại bỏ 1 nút có giá trị bằng v (không dùng đệ quy)

```
Node* removeNode(Node* h, int v){
    Node* p = h;
    if(h == NULL) return NULL;
    if(h->value == v
        Node* tmp = h; h = h->next;
        free(tmp); return h;
    }
    while(p->next != NULL){
        if(p->next->value == v) break;
        p = p->next;
    }
    if(p->next != NULL){
        Node* q = p->next; p->next = q->next; free(q);
    }
    return h;
}
```

# Các thao tác cơ bản trên danh sách liên kết đơn

- Loại bỏ 1 nút có giá trị bằng v (sử dụng đệ quy)

```
Node* removeNodeRecursive(Node* h, int v){
    if(h == NULL) return NULL;
    if(h->value == v){
        Node* tmp = h; h = h->next; free(tmp); return h;
    }
    h->next = removeNodeRecursive(h->next, v);
    return h;
}
```

# Các thao tác cơ bản trên danh sách liên kết đơn

- Loại bỏ tất cả các nút có giá trị bằng v (sử dụng đệ quy)

```
Node* removeAll(Node* h, int v){
    // remove all nodes having value v from the linked list headed by h
    if(h == NULL) return NULL;
    if(h->value == v){
        Node* tmp = h; h = h->next; free(tmp);
        h = removeAll(h,v); // continue to remove other elements having value v
        return h;
    }
    h->next = removeAll(h->next,v);
    return h;
}
```



# Các thao tác cơ bản trên danh sách liên kết đơn

- Đếm số nút trên danh sách (sử dụng và không sử dụng đệ quy)

```
int countRecursive(Node* h){  
    if(h == NULL) return 0;  
    return 1+countRecursive(h->next);  
}
```

```
int count(Node* h){  
    int cnt = 0;  
    Node* p = h;  
    while(p != NULL){  
        cnt += 1;  
        p = p->next;  
    }  
    return cnt;  
}
```

# Các thao tác cơ bản trên danh sách liên kết đơn

- Đảo ngược thứ tự các nút trong danh sách

```
Node* reverse(Node *h){  
    Node* p = h;  
    Node* pp = NULL;  
    Node* np = NULL;  
    while(p != NULL){  
        np = p->next;  
        p->next = pp;  
        pp = p;  
        p = np;  
    }  
    return pp;  
}
```

# Bài tập quản lý hồ sơ sinh viên

---

- Hồ sơ sinh viên bao gồm:
  - name: tên của sinh viên
  - email: địa chỉ email của sinh viên
- Viết chương trình trong chế độ tương tác dòng lệnh thực hiện các nghiệp vụ cơ bản trong quản lý hồ sơ sinh viên
  - Đọc dữ liệu từ file văn bản vào danh sách
  - In danh sách sinh viên
  - Thêm 1 hồ sơ vào cuối danh sách
  - Xóa 1 hồ sơ
  - Tìm kiếm hồ sơ
  - Lưu hồ sơ vào file văn bản

# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

- Khai báo dữ liệu

```
#include <stdio.h>

#define MAX_L 256

typedef struct Profile{
    char name[MAX_L]; // ten sinh vien
    char email[MAX_L]; // email cua sinh vien
    struct Profile* next;
}Profile;

Profile* first, *last;
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

```
Profile* makeProfile(char* name, char* email){
    Profile* node = (Profile*)malloc(sizeof(Profile));
    strcpy(node->name,name);
    strcpy(node->email,email);
    node->next = NULL;
    return node;
}

void initList(){
    first = NULL; last = NULL;
}

int listEmpty(){
    return first == NULL && last == NULL;
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

```
void insertLast(char* name, char* email){
    Profile* profile = makeProfile(name,email);
    if(listEmpty()){
        first = profile; last = profile;
    }else{
        last->next = profile; last = profile;
    }
}

void printList(){
    for(Profile* p = first; p != NULL; p = p->next)
        printf("%s, %s\n",p->name, p->email);
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

```
Profile* removeProfile(Profile* f, char* name){
    if(listEmpty()) return NULL;
    if(strcmp(f->name,name) == 0){
        Profile* tmp = f->next;
        free(f);
        if(tmp == NULL) last = NULL;
        return tmp;
    }else{
        f->next = removeProfile(f->next,name);
        return f;
    }
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

```
void load(char* filename){
    FILE* f = fopen(filename,"r");
    if(f == NULL) printf("Load data -> file not found\n");

    while(!feof(f)){
        char name[256], email[256];
        fscanf(f,"%s%s",name, email);
        insertLast(name,email);
    }
    fclose(f);
}
```



# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

```
void processFind(){
    char name[256];
    scanf("%s",name);
    Profile* profile = NULL;
    for(Profile* p = first; p != NULL; p = p->next){
        if(strcmp(p->name,name)==0){
            profile = p; break;
        }
    }
    if(profile == NULL){
        printf("NOT FOUND profile %s\n",name);
    }else{
        printf("FOUND profile %s, %s\n",profile->name,profile->email);
    }
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

```
void processLoad(){// load du lieu tu file vao bo nho
    char filename[256];
    scanf("%s",filename);
    load(filename);
}
void processStore(){// luu du lieu DS vao file van ban
    char filename[256];
    scanf("%s",filename);
    FILE* f = fopen(filename,"w");
    for(Profile* p = first; p != NULL; p = p->next){
        fprintf(f,"%s %s",p->name,p->email);
        if(p->next != NULL) fprintf(f,"\n");
    }
    fclose(f);
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

```
void processInsert(){
    char name[256], email[256];
    scanf("%s%s",name,email);
    insertLast(name,email);
}

void processRemove(){
    char name[256];
    scanf("%s",name);
    first = removeProfile(first,name);
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đơn

```
int main(){// chạy chương trình trong chế độ interactive
    initList();
    while(1){
        printf("Enter command: ");
        char cmd[256];
        scanf("%s",cmd);
        if(strcmp(cmd,"Quit")==0) break;
        else if(strcmp(cmd,"Load")==0) processLoad();
        else if(strcmp(cmd,"Print")==0) printList();
        else if(strcmp(cmd,"Find")==0) processFind();
        else if(strcmp(cmd,"Insert")==0) processInsert();
        else if(strcmp(cmd,"Remove")==0) processRemove();
        else if(strcmp(cmd,"Store")==0) processStore();
    }
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

- Định nghĩa cấu trúc dữ liệu

```
#include <stdio.h>

#define MAX_L 256

typedef struct Profile{
    char name[MAX_L];
    char email[MAX_L];
    struct Profile* next;// pointer to the next element
    struct Profile* prev;// pointer to the predecessor
}Profile;

Profile* first, *last;
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
Profile* makeProfile(char* name, char* email){  
    Profile* node = (Profile*)malloc(sizeof(Profile));  
    strcpy(node->name,name);  
    strcpy(node->email,email);  
    node->next = NULL;  
    node->prev = NULL;  
    return node;  
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
void initList(){
    first = NULL; last = NULL;
}
int listEmpty(){
    return first == NULL && last == NULL;
}
void printListLeft2Right(){
    for(Profile* p = first; p != NULL; p = p->next)
        printf("%s, %s\n",p->name, p->email);
}
void printListRight2Left(){
    for(Profile* p = last; p != NULL; p = p->prev)
        printf("%s, %s\n",p->name, p->email);
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
void insertLast(char* name, char* email){
    Profile* profile = makeProfile(name,email);
    if(listEmpty()){
        first = profile; last = profile;
    }else{
        last->next = profile;  profile->prev = last;
        last = profile;
    }
}
```



# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
Profile* find(char*name){
    for(Profile* p = first; p != NULL; p = p->next){
        if(strcmp(p->name,name)==0){
            return p;
        }
    }
    return NULL;
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
void removeProfile(char* name){
    if(listEmpty()) return NULL;
    Profile* profile = find(name);
    if(profile == NULL){
        printf("NOT FOUND %s\n",name);
    }else{
        Profile* left = profile->prev;
        Profile* right = profile->next;
        if(left != NULL) left->next = right;
        if(right != NULL) right->prev = left;
        if(left == NULL) first = right;
        if(right == NULL) last = left;
        free(profile);
    }
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
void load(char* filename){
    FILE* f = fopen(filename,"r");
    if(f == NULL) printf("Load data -> file not found\n");
    initList();
    while(!feof(f)){
        char name[256], email[256];
        fscanf(f,"%s%s",name, email);
        insertLast(name,email);
        printf("insert %s, %s\n",name,email);
    }
    fclose(f);
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
void processFind(){
    char name[256];
    scanf("%s",name);
    Profile* profile = find(name);
    if(profile == NULL){
        printf("NOT FOUND profile %s\n",name);
    }else{
        printf("FOUND profile %s, %s\n",profile->name,profile->email);
    }
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
void processLoad(){
    char filename[256];
    scanf("%s",filename);
    load(filename);
}

void processStore(){
    char filename[256];
    scanf("%s",filename);
    FILE* f = fopen(filename,"w");
    for(Profile* p = first; p != NULL; p = p->next){
        fprintf(f,"%s %s",p->name,p->email);
        if(p->next != NULL) fprintf(f,"\n");
    }
    fclose(f);
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
void processInsert(){
    char name[256], email[256];
    scanf("%s%s",name,email);
    insertLast(name,email);
}

void processRemove(){
    char name[256];
    scanf("%s",name);
    removeProfile(name);
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
void processPrintList(){
    printf("Danh sach tu trai qua phai\n");
    printListLeft2Right();
    printf("Danh sach tu phai qua trai\n");
    printListRight2Left();
}

void finalize(){
    Profile* p = first;
    while(p != NULL){
        Profile* np = p->next;
        free(p);
        p = np;
    }
}
```

# Quản lý hồ sơ sinh viên: danh sách liên kết đôi

```
int main(){
    initList();
    while(1){
        printf("Enter command: ");
        char cmd[256];
        scanf("%s",cmd);
        if(strcmp(cmd,"Quit")==0) break;
        else if(strcmp(cmd,"Load")==0) processLoad();
        else if(strcmp(cmd,"Print")==0) processPrintList();
        else if(strcmp(cmd,"Find")==0) processFind();
        else if(strcmp(cmd,"Insert")==0) processInsert();
        else if(strcmp(cmd,"Remove")==0) processRemove();
        else if(strcmp(cmd,"Store")==0) processStore();
    }
    finalize();
}
```





25  
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

