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

LẬP TRÌNH C CƠ BẢN

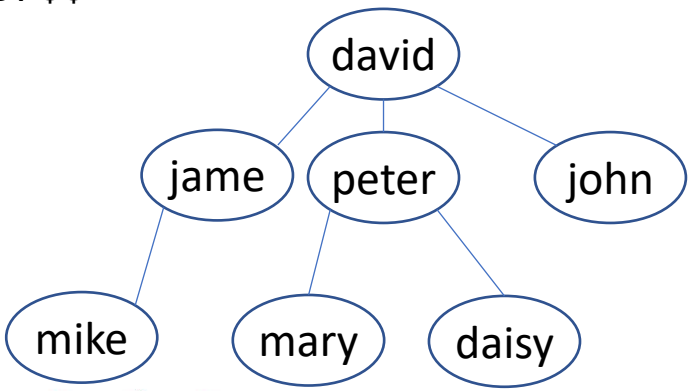
Cây – phần 1

Thao tác với cây tổng quát

- Mỗi nút của cây có cấu trúc dữ liệu như sau

```
typedef struct Node{
    char name[256];
    struct Node* leftMostChild; // pointer to the left-most child
    struct Node* rightSibling;// pointer to the right sibling
}Node;
```

- Dữ liệu về cây được lưu ở file text với định dạng như sau:
  - Mỗi dòng lưu 1 dãy các xâu ký tự s_0, s_1, …, s_k kết thúc bởi $ trong đó s_1, s_2, …, s_k là con của s_0 từ trái qua phải (s_1 là con trái nhất của s_0) (***ghi chú***: mỗi dòng (trừ dòng 1), s_0 là con của một nút nào đó đã xuất hiện trong dòng trước đó).
  - File được kết thúc bởi $$




```
david jame peter john $
peter mary daisy $
jame mike $
$$
```

Thao tác với cây tổng quát

- Viết chương trình C chạy ở chế độ tương tác để thao tác với cây biểu diễn sơ đồ phả hệ gia đình với các lệnh sau:
  - Load <filename>: nạp dữ liệu từ <filename> vào bộ nhớ và dựng cây
  - FindChildren <name>: in ra màn hình danh sách con của <name>
  - AddChild <name> <child>: thêm một con <child> vào cuối danh sách con của <name>
  - Print: In ra màn hình danh sách tất cả các thành viên trong gia đình
  - Height <name>: in ra chiều cao của <name> trên cây
  - Count: đếm số thành viên của gia đình (số nút của cây)
  - Store <filename>: Lưu dữ liệu cây ra file <filename>

Thao tác với cây tổng quát

```
#include <stdio.h>
typedef struct Node{
    char name[256];
    struct Node* leftMostChild;
    struct Node* rightSibling;
}Node;
Node* root;

Node* makeNode(char* name){
    Node* p = (Node*)malloc(sizeof(Node));
    strcpy(p->name,name);
    p->leftMostChild = NULL; p->rightSibling = NULL;
    return p;
}
```

Thao tác với cây tổng quát

```
Node* find(Node* r, char* name){
    if(r == NULL) return NULL;
    if(strcmp(r->name,name) == 0) return r;
    Node* p = r->leftMostChild;
    while(p != NULL){
        Node* q = find(p,name);
        if(q != NULL) return q;
        p = p->rightSibling;
    }
}
```

Thao tác với cây tổng quát

```
Node* addLast(Node* p, char*name){
    if(p == NULL) return makeNode(name);
    p->rightSibling = addLast(p->rightSibling, name);
    return p;
}
void addChild(char*name, char* child){
    Node* r = find(root,name);
    if(r == NULL) return;
    r->leftMostChild = addLast(r->leftMostChild,child);
}
```

Thao tác với cây tổng quát

```
void printTree(Node* r){
    if(r == NULL) return;
    printf("%s: ",r->name);
    Node* p = r->leftMostChild;
    while(p != NULL){
        printf("%s ",p->name);
        p = p->rightSibling;
    }
    printf("\n");
    p = r->leftMostChild;
    while(p != NULL){
        printTree(p);
        p = p->rightSibling;
    }
}
```

Thao tác với cây tổng quát

```
void printTreeF(Node* r, FILE* f){
    if(r == NULL) return;
    fprintf(f,"%s ",r->name);
    Node* p = r->leftMostChild;
    while(p != NULL){
        fprintf(f,"%s ",p->name);
        p = p->rightSibling;
    }
    fprintf(f," $\n");
    p = r->leftMostChild;
    while(p != NULL){
        printTreeF(p,f);
        p = p->rightSibling;
    }
}
```

Thao tác với cây tổng quát

```
void processFind(){
    char name[256];    scanf("%s",name);
    Node* p = find(root,name);
    if(p == NULL) printf("Not Found %s\n",name);
    else printf("Found %s\n",name);
}
```

Thao tác với cây tổng quát

```
void processFindChildren(){
    char name[256];    scanf("%s",name);
    Node* p = find(root,name);
    if(p == NULL) printf("Not Found %s\n",name);
    else{
        printf("Found %s with children: ",name);
        Node* q = p->leftMostChild;
        while(q != NULL){
            printf("%s ",q->name);     q = q->rightSibling;
        }
    }
    printf("\n");
}
```

Thao tác với cây tổng quát

```
int height(Node* p){
    if(p == NULL) return 0;
    int maxH = 0;
    Node* q = p->leftMostChild;
    while(q != NULL){
        int h = height(q);
        maxH = maxH < h ? h : maxH;
        q = q->rightSibling;
    }
    return maxH + 1;
}
```

Thao tác với cây tổng quát

```
void processHeight(){
    char name[256];
    scanf("%s",name);
    Node* p = find(root,name);
    if(p == NULL) printf("Not Found %s\n",name);
    else{
        printf("Found %s having height = %d\n",name,height(p));
    }
}
```

Thao tác với cây tổng quát

```
int count(Node* r){
    if(r == NULL) return 0;
    int cnt = 1;
    Node* q = r->leftMostChild;
    while(q != NULL){
        cnt += count(q);
        q = q->rightSibling;
    }
    return cnt;
}
void processCount(){
    printf("Number of members is %d\n",count(root));
}
```

Thao tác với cây tổng quát

```
void processStore(){
    char filename[256];
    scanf("%s",filename);
    FILE* f = fopen(filename,"w");
    printTreeF(root,f);
    fprintf(f,"$$");
    fclose(f);
}
```

Thao tác với cây tổng quát

```
void freeTree(Node* r){
    if(r == NULL) return;
    Node* p = r->leftMostChild;
    while(p != NULL){
        Node* sp = p->rightSibling;
        freeTree(p);
        p = sp;
    }
    printf("free node %s\n",r->name); free(r);
    r = NULL;
}
```

Thao tác với cây tổng quát

```
void main(){
    while(1){
        char cmd[256];
        printf("Enter command: ");  scanf("%s",cmd);
        if(strcmp(cmd,"Quit") == 0) break;
        else if(strcmp(cmd,"Load")==0) processLoad();
        else if(strcmp(cmd,"Print")==0) processPrint();
        else if(strcmp(cmd,"Find")==0) processFind();
        else if(strcmp(cmd,"FindChildren")==0) processFindChildren();
        else if(strcmp(cmd,"Height")==0) processHeight();
        else if(strcmp(cmd,"Count")==0) processCount();
        else if(strcmp(cmd,"AddChild")==0) processAddChild();
        else if(strcmp(cmd,"Store")==0) processStore();
    }
    freeTree(root);
}
```

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

soict.hust.edu.vn/ fb.com/groups/soict