



HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# C Programming Basic

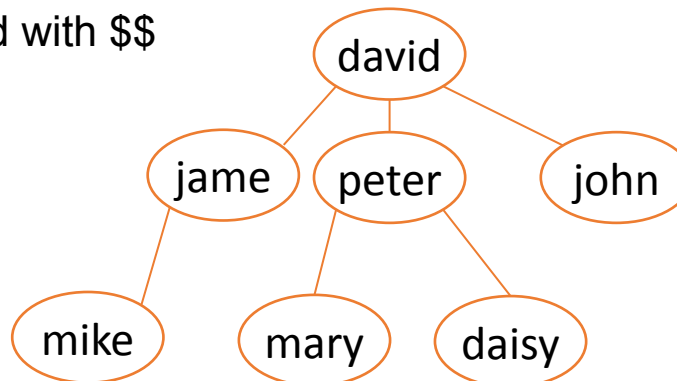
## Trees – part 1

# Manipulation with general trees

- Each node of a tree has the following structure

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

- The data of a tree is stored in an external text file with the format:
  - Each line contains a sequence of strings  $s_0, s_1, \dots, s_k$  terminated by \$ in which  $s_1, s_2, \dots, s_k$  are children of  $s_0$  from left to right ( $s_1$  is the left-most child) (**note**: in each line (except line 1), the string  $s_0$  is a child of some node appearing in previous lines).
  - The file is terminated with \$\$



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

# Manipulation with general trees

---

- Write a program running in an interactive mode for manipulating general trees representing members of a family with following instructions:
  - Load <filename>: load data from a text file and build the family tree
  - FindChildren <name>: print children of a given <name>
  - AddChild <name> <child>: add a new child to the children list of <name>
  - Print: print all members of the family
  - Height <name>: print the height of <name> in the tree
  - Count: print the number of members of the family
  - Store <filename>: store the family tree to a text file <filename>

# Manipulation with general trees

```
#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;
}
```

# Manipulation with general trees

```
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;
    }
}
```

# Manipulation with general trees

```
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);
}
```

# Manipulation with general trees

```
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;
    }
}
```

# Manipulation with general trees

```
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;
    }
}
```



# Manipulation with general trees

---

```
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);
}
```

# Manipulation with general trees

```
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");
}
```

# Manipulation with general trees

```
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;  
}
```

# Manipulation with general trees

```
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));
    }
}
```

# Manipulation with general trees

```
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));
}
```

# Manipulation with general trees

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

# Manipulation with general trees

```
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;
}
```

# Manipulation with general trees

```
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);
}
```





25 YEARS ANNIVERSARY  
**SOICT**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



[soict.hust.edu.vn/](http://soict.hust.edu.vn/)



[fb.com/groups/soict](https://fb.com/groups/soict)

