C Programming Basic - week 9

Chủ đề

- Tạo tệp thực thi sử dụng makefile
- Duyệt cây
  - Duyệt theo chiều sâu
    - Preorder
    - Inorder
    - Postorder
  - Duyệt theo chiều rộng
- Exercises

Makefile

- Chương trình nhỏ → một tệp
- Chương trình lớn:
  - Nhiều dòng code
  - Nhiều thành phần
  - Nhiều lập trình viên

- Vấn đề:
  - Nhiều tệp khó quản lý
    (cho cả lập trình viên và máy)
  - Mỗi thay đổi cần biên dịch lâu
  - Nhiều lập trình viên không thể sửa đổi một tệp đồng thời

Makefile (2)

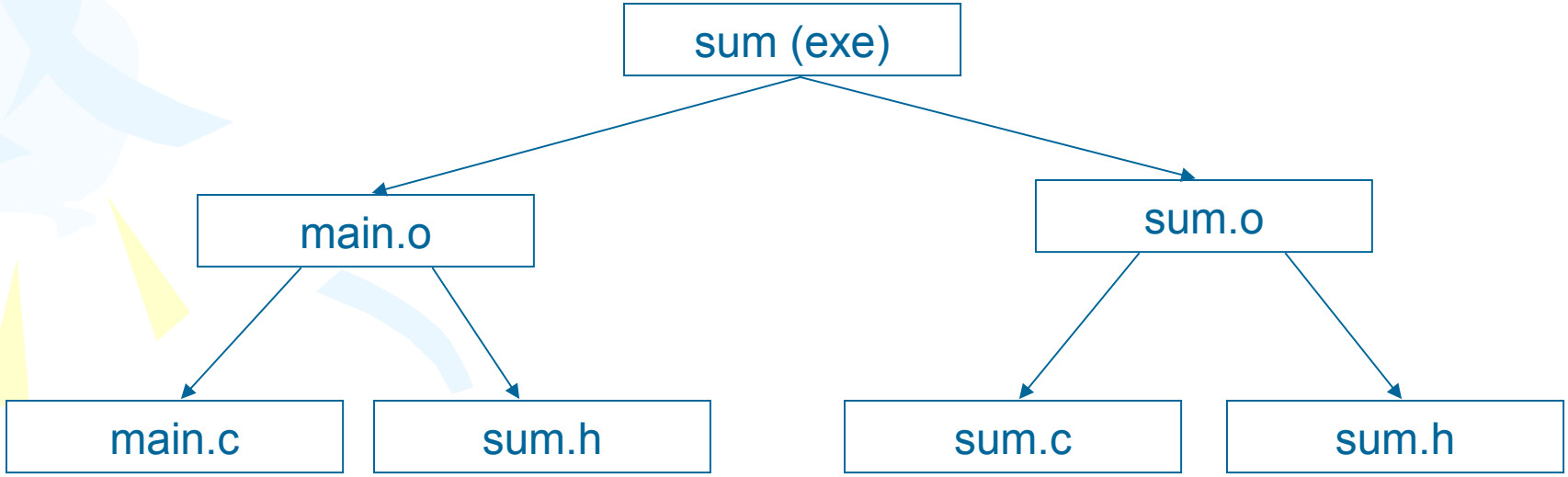
- Giải pháp : chia dự án thành nhiều tệp
- Mục tiêu:
  - Chia nhỏ thành các thành phần
  - Biên dịch tối thiểu khi có thay đổi
  - Dễ dàng bảo trì cấu trúc dự án và các phụ thuộc

Bảo trì dự án

- Sử dụng makefile trên Unix
- Makefile là một tệp (script) chứa:
  - Cấu trúc dự án (các tệp, các phụ thuộc)
  - Hướng dẫn để tạo tệp
- Lệnh make đọc một makefile, hiểu cấu trúc dự án và tạo tệp thực thi
- Makefile không chỉ dùng với chương trình C

Cấu trúc dự án

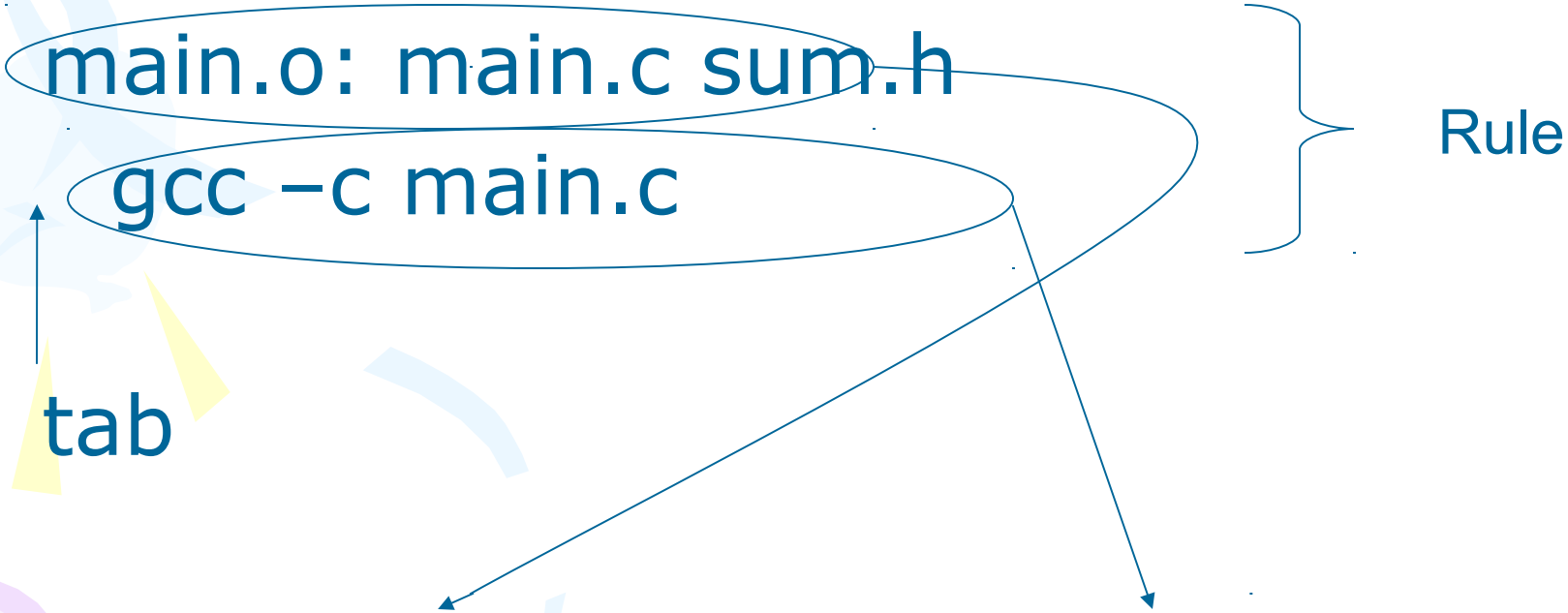
- Cấu trúc và phụ thuộc của dự án có thể biểu diễn bởi một DAG (Directed Acyclic Graph)
- VD :
  - Chương trình chứa 3 tệp
  - main.c., sum.c, sum.h
  - sum.h được khai báo trong cả hai tệp .c
  - Tệp thực thi có tên sum


sum (exe)
main.o
sum.o
main.c
sum.h
sum.c
sum.h

makefile

```
sum: main.o sum.o
    gcc –o sum main.o sum.o

main.o: main.c sum.h
    gcc –c main.c

sum.o: sum.c sum.h
    gcc –c sum.c
```

Cú pháp



Cách khác

- .o phụ thuộc (mặc định) vào tệp .c tương ứng

```
sum: main.o sum.o
    gcc –o sum main.o sum.o

main.o: sum.h
    gcc –c main.c

sum.o: sum.h
    gcc –c sum.c
```

Cách khác (2)

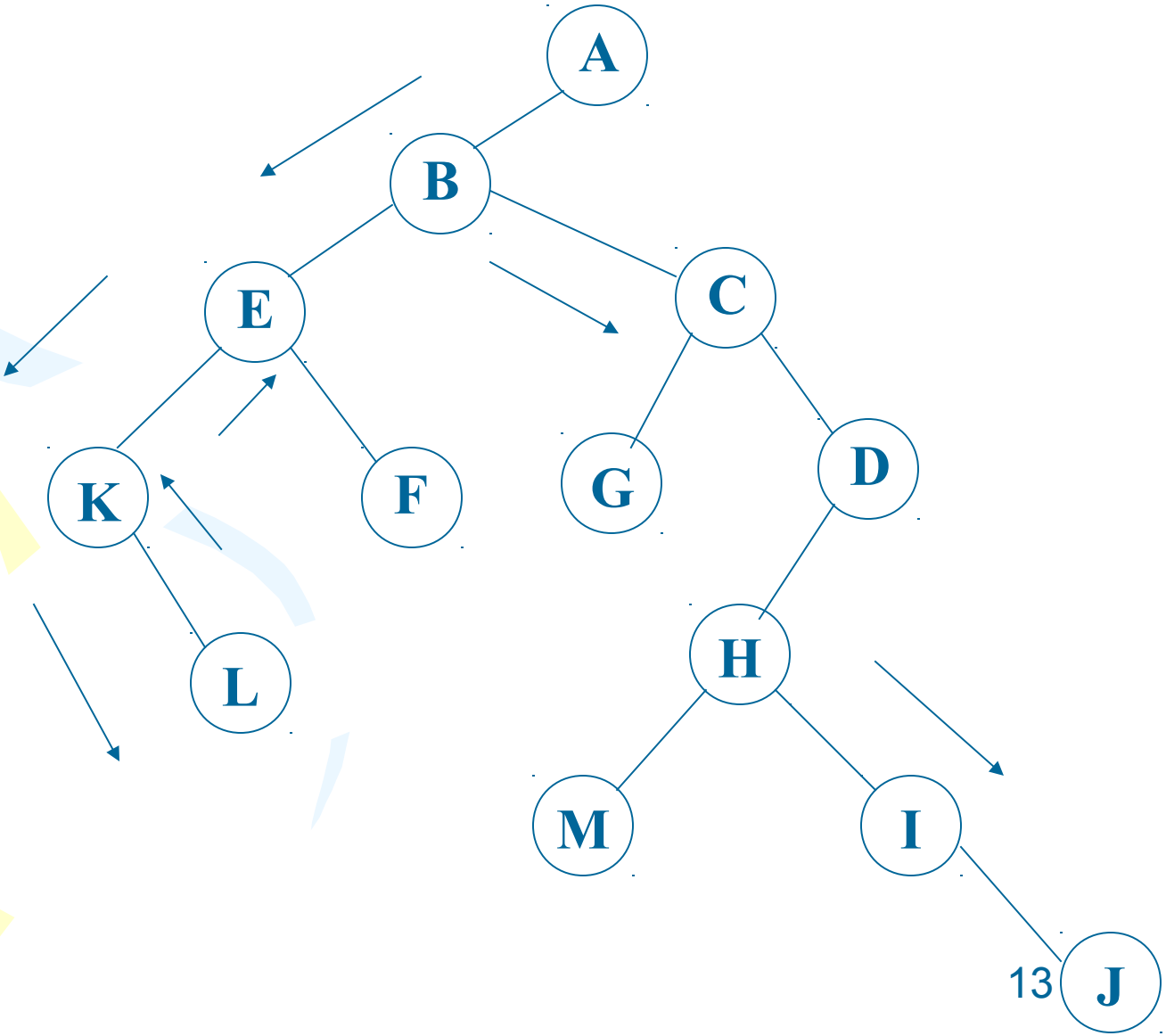
- Nén các phụ thuộc tương tự :

```
sum: main.o sum.o
    gcc –o $@ main.o sum.o

main.o sum.o: sum.h
    gcc –c $*.c
```

Duyệt cây nhị phân

- Rất nhiều thao tác trên cây nhị phân được thực hiện thông qua duyệt
- Mỗi nút được thăm một lần duy nhất
- Khi thăm một nút, tất cả các hành động (sao chép, hiển thị, đánh giá, cập nhật...) đối với nút này được thực hiện

Duyệt cây nhị phân

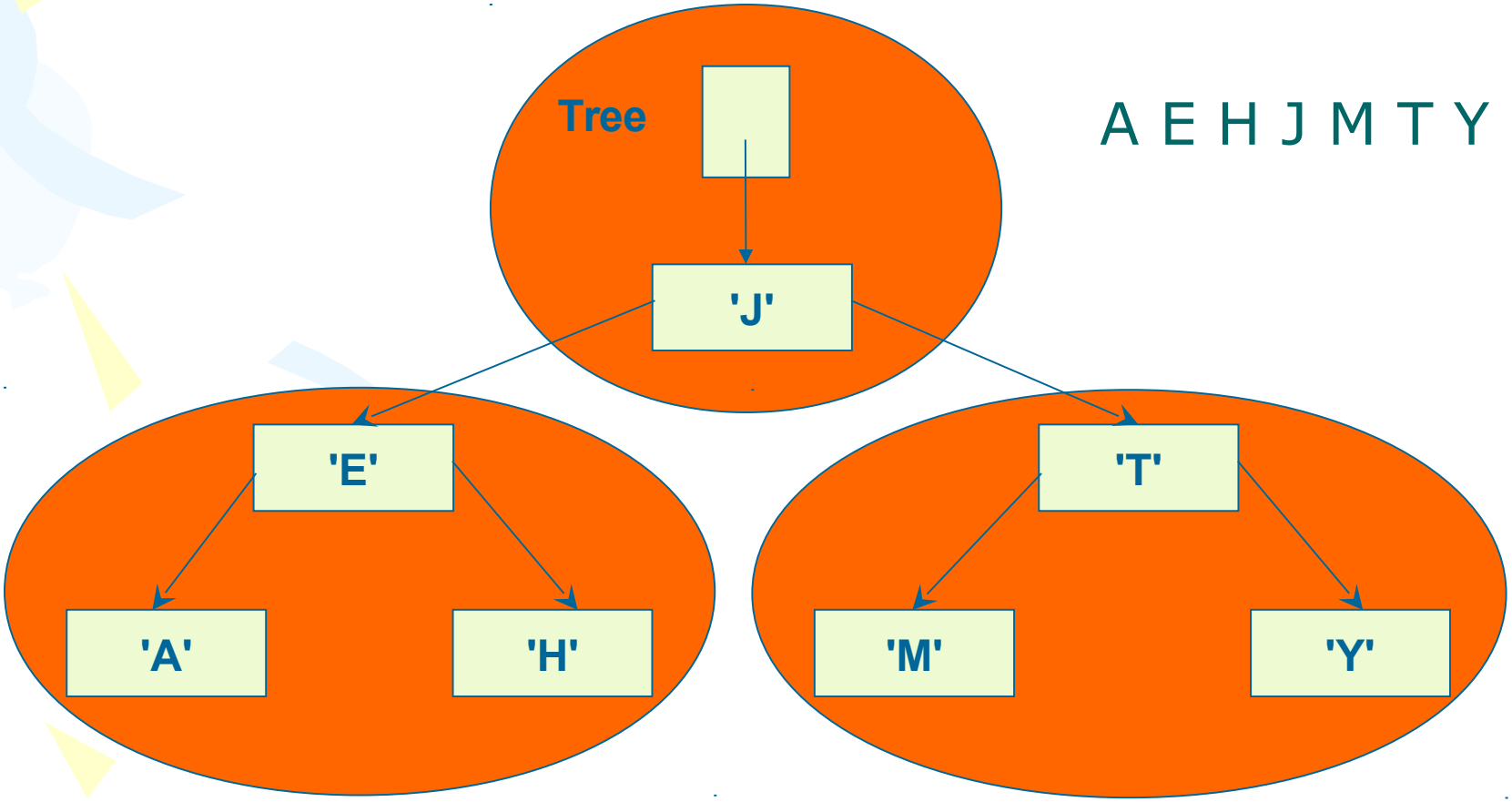


DFS

- Tìm kiếm theo chiều sâu: Duyệt sâu nhất có thể
- Các kiểu duyệt:
  - Preorder
  - Inorder
  - Postorder

Inorder

- Cây con trái → root → cây con phải

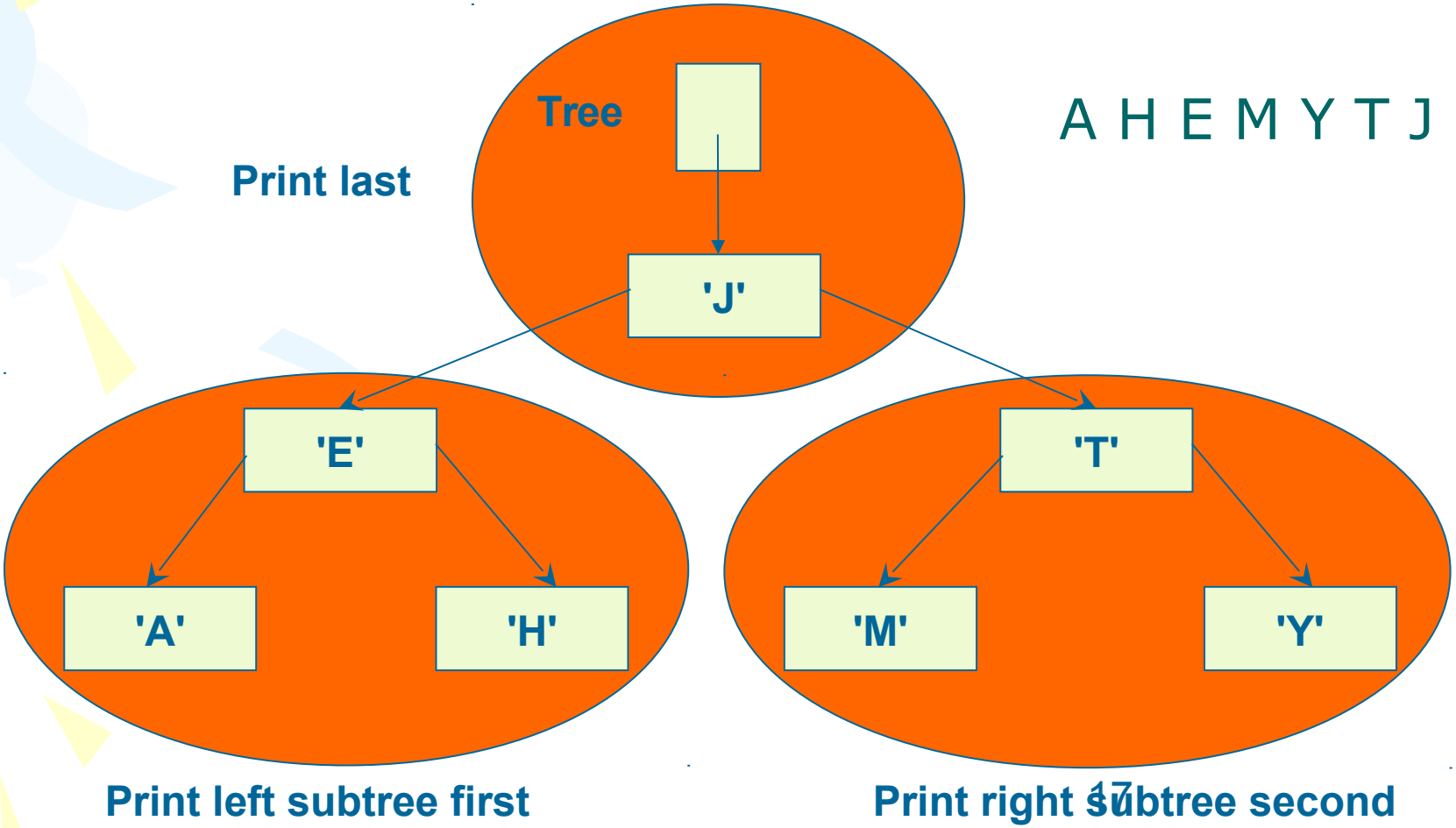


inorderprint

```
void inorderprint(TreeType tree)
{
  if (tree!=NULL)
  {
   inorderprint(tree->left);
   printf("%4d\n",tree->Key);
   inorderprint(tree->right);
  }
}
```

Postorder

- Cây con trái → cây con phải → root

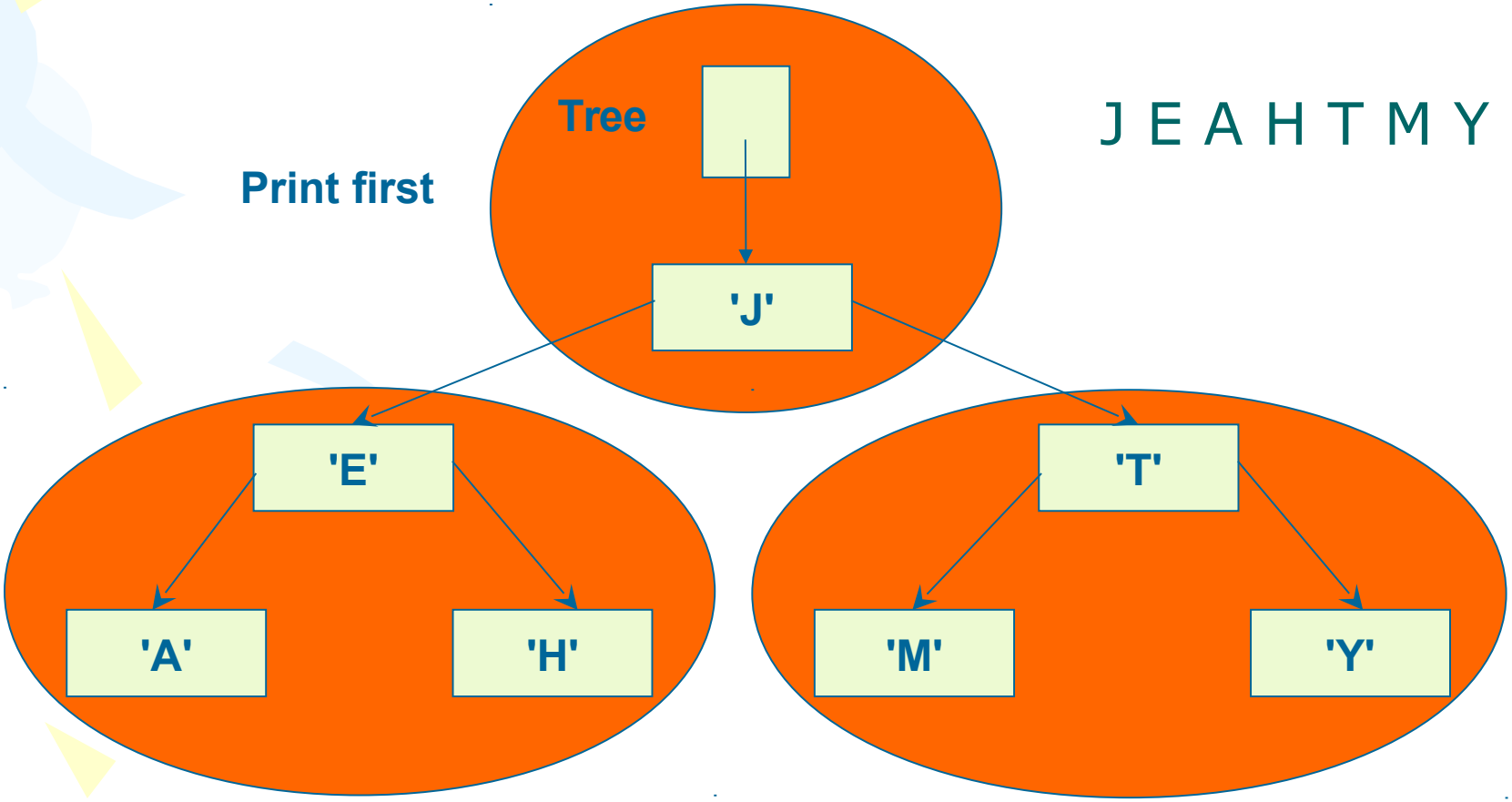


postorderprint

```
void postorderprint(TreeType tree)
{
  if (tree!=NULL)
  {
   postorderprint(tree->left);
   postorderprint(tree->right);
   printf("%4d\n",tree->Key);
  }
}
```

Preorder

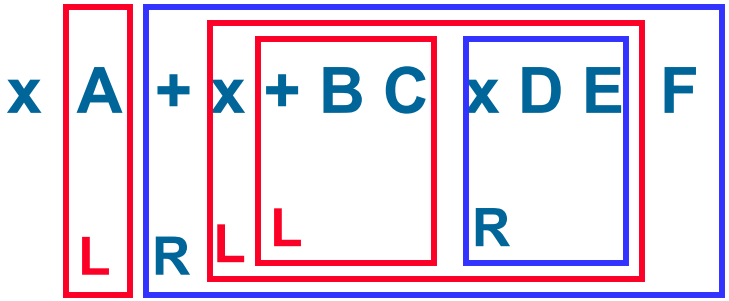
- Root → cây con trái → cây con phải

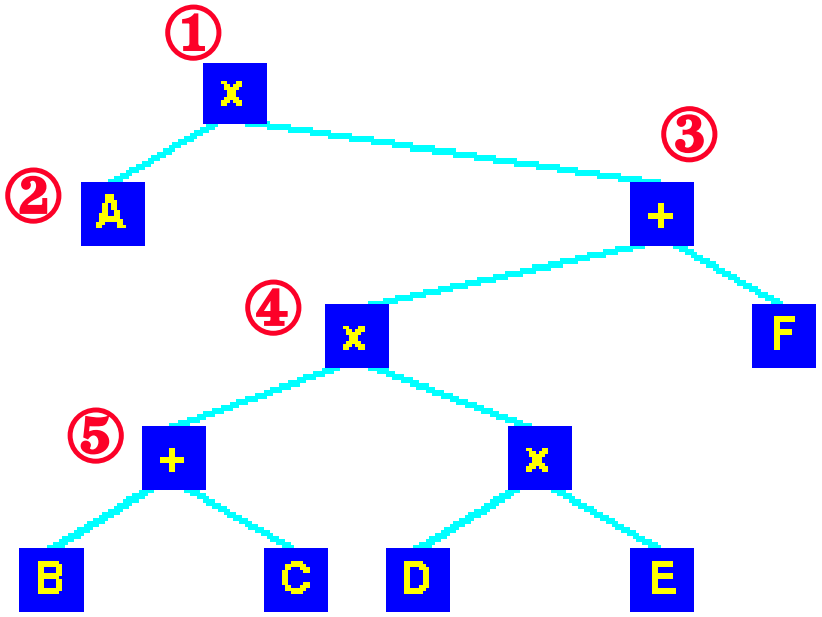


J E A H T M Y

Pre_order

❶ Pre-order
- Root
- Left sub-tree
- Right sub-tree





preorderprint

```
void preorderprint(TreeType tree)
{
  if (tree!=NULL)
  {
   printf("%4d\n",tree->Key);
   preorderprint(tree->left);
   preorderprint(tree->right);
  }
}
```

Exercise 9.1

- Sử dụng cây nhị phân tìm kiếm để lưu danh bạ điện thoại với email tăng dần
- Hiển thị danh bạ theo thứ tự tăng dần của email

Hướng dẫn: Sử dụng InOrderTraversal

Khử đệ quy inorder

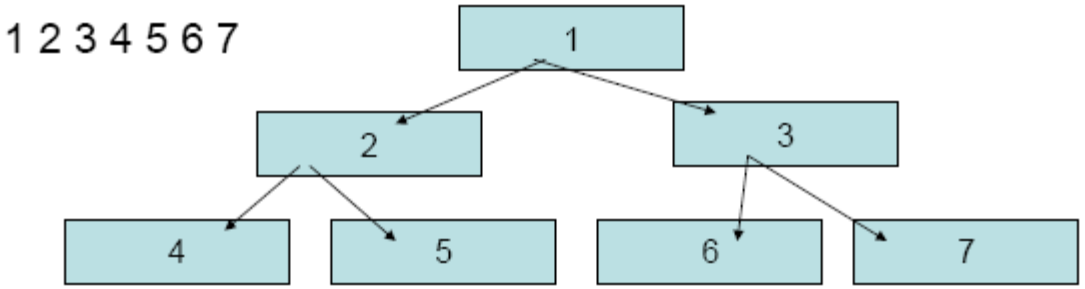
```
void iter_inorder(TreeType node)
{
  int top= -1; /* initialize stack */
  TreeType stack[MAX_STACK_SIZE];
  for (;;) {
   for (; node; node=node->left)
     add(&top, node);/* add to stack */
   node= delete(&top);/*delete from stack*/

   if (node==NULL) break;/* stack is empty */
   printf("%d", node->key);
   node = node->right;
  }
}
```

Exercise 9.2

- Hiển thị danh bạ theo thứ tự tăng dần của name
  - ra màn hình
  - ra tệp

Breadth First Search

- Duyệt theo chiều rộng
- Thăm tất cả các nút cùng mức từ trái qua phải



Breadth First Search

- Sử dụng một hàng đợi
- Thêm nút root vào hàng đợi
- Với một nút trong hàng đợi
  - Thăm nút
  - Thêm nút con trái vào hàng đợi
  - Thêm nút con phải vào hàng đợi

Thuật toán

```
void breadth_first(TreeType node)
{
    QueueType queue; // queue of pointers
    if (node!=NULL) {
        enq(node,queue);
        while (!empty(queue)) {
            node=deq(queue);
            printf(node->key);
            if (node->left !=NULL)
                enq(node->left,queue);
            if (node->right !=NULL)
                enq(node->right,queue);
        }
    }
}
```

Exercise 9.3

- Cài đặt BFS
- Thêm hàm vào thư viện cây nhị phân
- Kiểm tra chương trình danh bạ điện thoại, in ra tất cả tên trong danh bạ
- Ghi kết quả ra tệp

Exercise 9.4

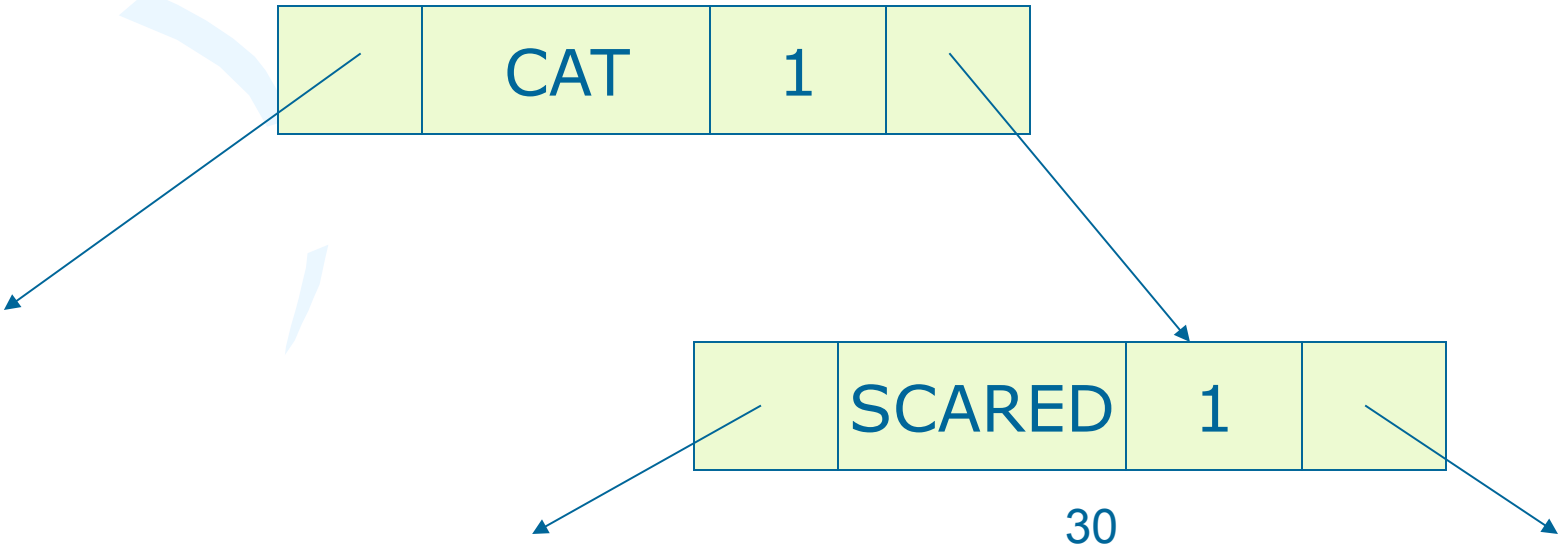
- Viết chương trình WordCount đọc một tệp văn bản, sau đó phân tích tần suất từ. Kết quả được lưu trong một tệp. Khi người dùng đưa vào một từ, chương trình hiển thị tần suất xuất hiện của từ trong tệp
- VD, nếu tập có nội dung: *A black black cat saw a very small mouse and a very scared mouse.*
- Tần suất của các từ sẽ là:

AND 1
CAT 1
SAW 1
SCARED 1

SMALL 1
BLACK 2
MOUSE 2
VERY 2
A 3

Hint

- Sử dụng BST để lưu dữ liệu
- Mỗi nút chứa ít nhất hai trường:
  - word: string
  - count: int
- Từ được lưu trong các nút của cây



Homework 1

- Viết chương trình quản lý giải đấu: Sử dụng cây nhị phân có nút cha có giá trị lớn nhất trong hai con; nút root có giá trị lớn nhất
- Giá trị các nút là được đưa vào từ một mảng
- Gợi ý: Sử dụng chiến thuật chia-để-trị