C Programming Basic – week 9

Tree

Lecturer:

Do Quoc Huy

Dept of Computer Science

Hanoi University of Technology

Topics of this week

- Cách xây dựng chương trình bằng công cụ makefile
- Duyệt cây
  - Tìm kiếm theo chiều sâu
    - Duyệt thứ tự trước
    - Duyệt thứ tự giữa
    - Duyệt thứ tự sau
  - Tìm kiếm theo chiều rộng
- Các bài tập thực hành

Makefile – động lực

- Chương trình nhỏ ⟶ một file đơn lẻ
- Các chương trình "không nhỏ lắm" :
  - Nhiều dòng lệnh
  - Nhiều thành phần
  - Nhiều hơn một lập trình viên

- Vấn đề:
  - Các file dài khó quản lý hơn
    (cho cả lập trình viên lẫn máy tính)
  - Mỗi thay đổi sẽ yêu cầu biên dịch lâu hơn
  - Khi có nhiều lập trình viên thì không thể thay đổi file đồng thời

Makefile - motivation

- Giải pháp : chia chương trình thành nhiều file
- Mục đích:
  - Phân chia tốt các thành phần
  - Tối thiểu sự biên dịch khi có gì đó thay đổi.
  - Dễ dàng bảo trì cấu trúc project, sự phụ thuộc và sự sáng tạo.

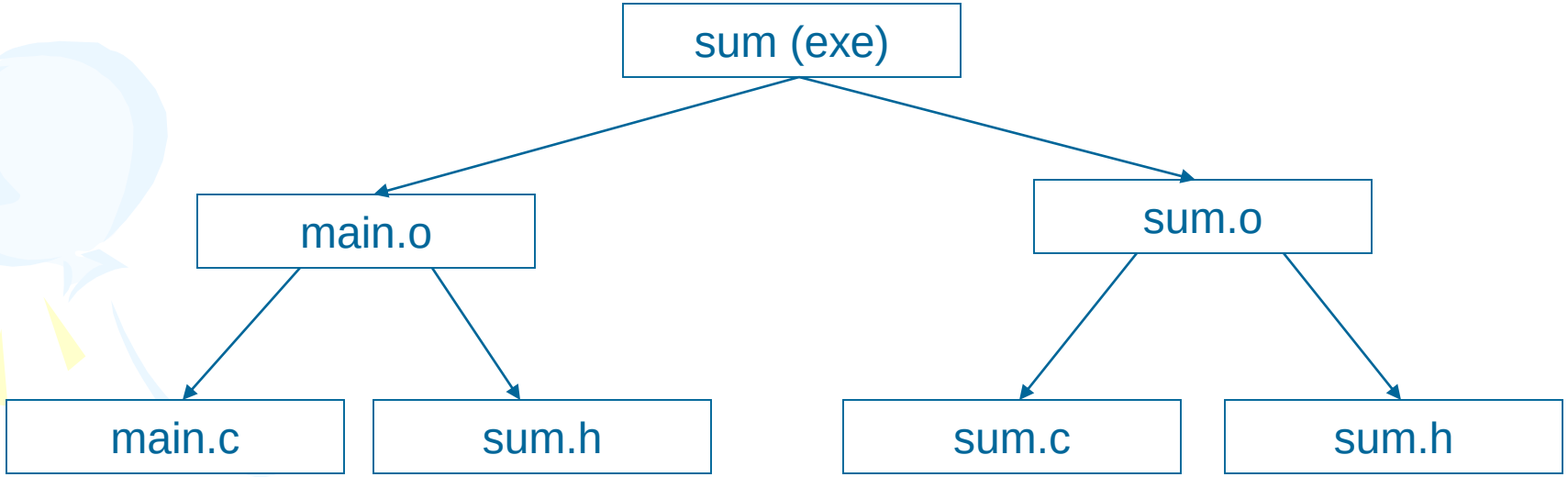
Bảo trì project

- Được thực hiện trong UNIX với công cụ Makefile
- makefile là 1 file (script - bản thảo) chứa:
  - Cấu trúc structure (files, thông tin phụ thuộc)
  - Hướng dẫn tạo các file
- Lệnh make đọc 1 makefile, hiểu cấu trúc project và tạo file thực thi
- Lưu ý rằng Makefile không chỉ giới hạn trong các chương trình C

cấu trúc Project

- Cấu trúc project và các thành phần phụ thuộc có thể biểu diễn bởi 1 DAG (Directed Acyclic Graph – Đồ thị có hướng không có chu trình)
- Ví dụ:

- Chương trình có 3 files
  - main.c., sum.c, sum.h
  - sum.h được include trong file both.c
  - File thực thi là file sum


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

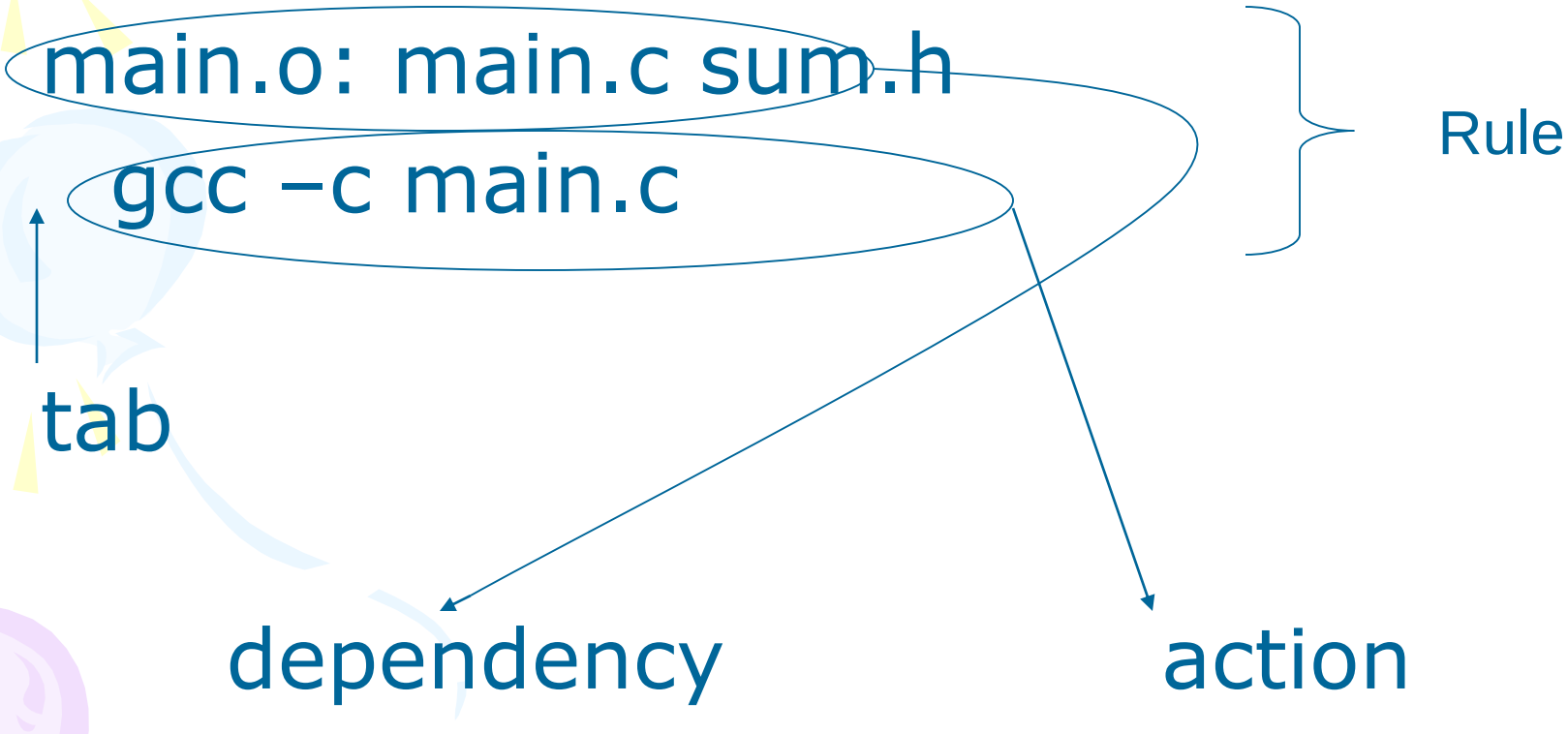
makefile

```
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
```

Rule syntax



Các makefile tương đương

- .o phụ thuộc (mặc định) vào file .c tương ứng.Bởi vậy, makefile tương đương là:

```
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
```

Các makefile tương đương (tiếp)

- Ta có thể nén các sự phụ thuộc giống nhau và sử dụng các lệnh gắn liền để tạo 1 makefile tương đương khác (ngắn hơn):

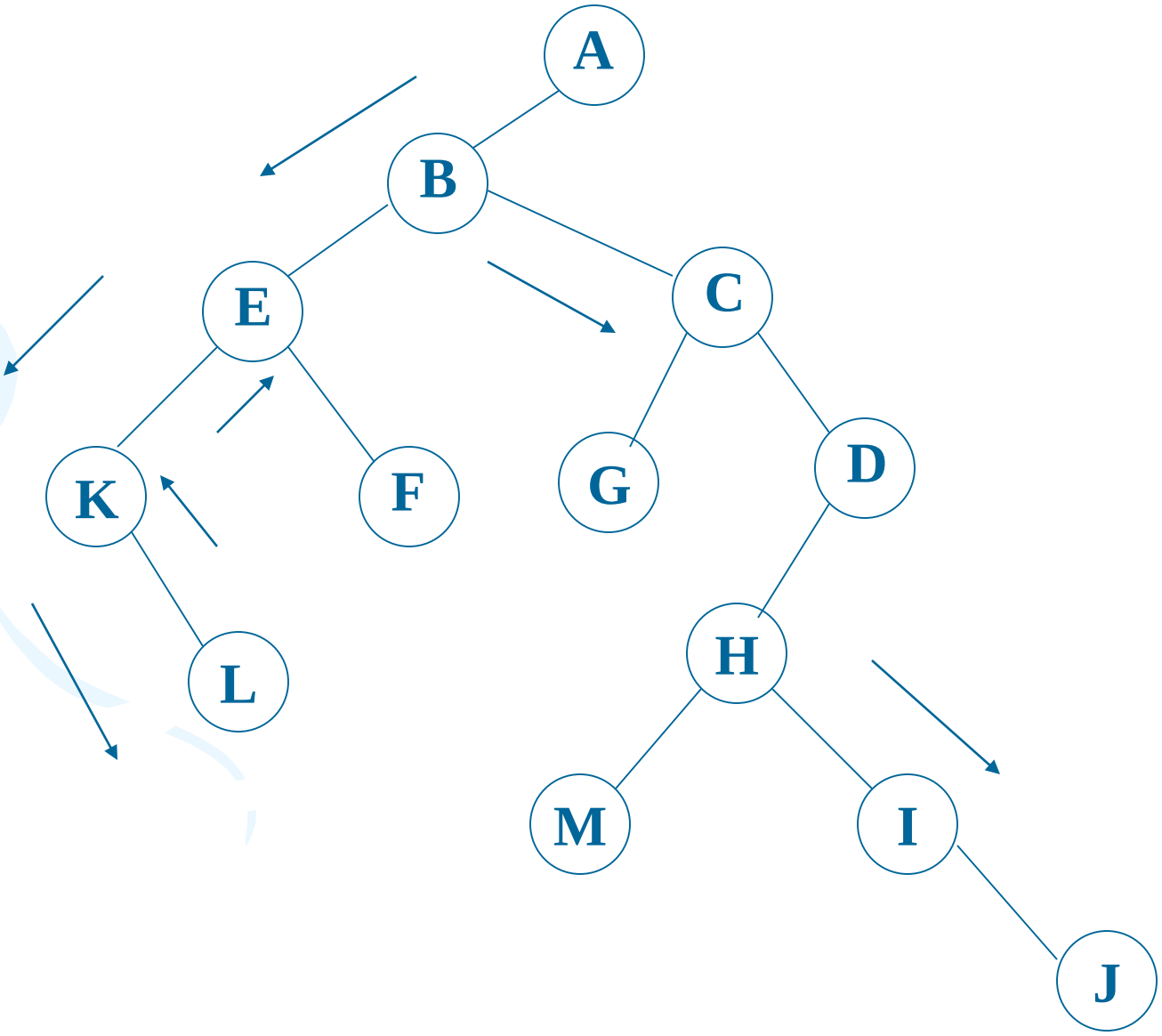
```
sum: main.o sum.o
    gcc –o $@ main.o sum.o

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

Binary Tree Traversal (duyệt cây nhị phân)

- Rất nhiều phép toán trên cây nhị phân được thực hiện bởi trình diễn duyệt 1 cây nhị fân.
- Trong duyệt, thì mỗi phần tử của cây nhị phân chỉ được thăm đúng 1 lần.
- Khi thăm 1 phần tử, tất cả các hành động (tạo 1 bản sao, đưa ra màn hình, tính giá trị phép toán…) với sự lưu tâm tới phần tử này được thực hiện.

Binary Tree Traversal



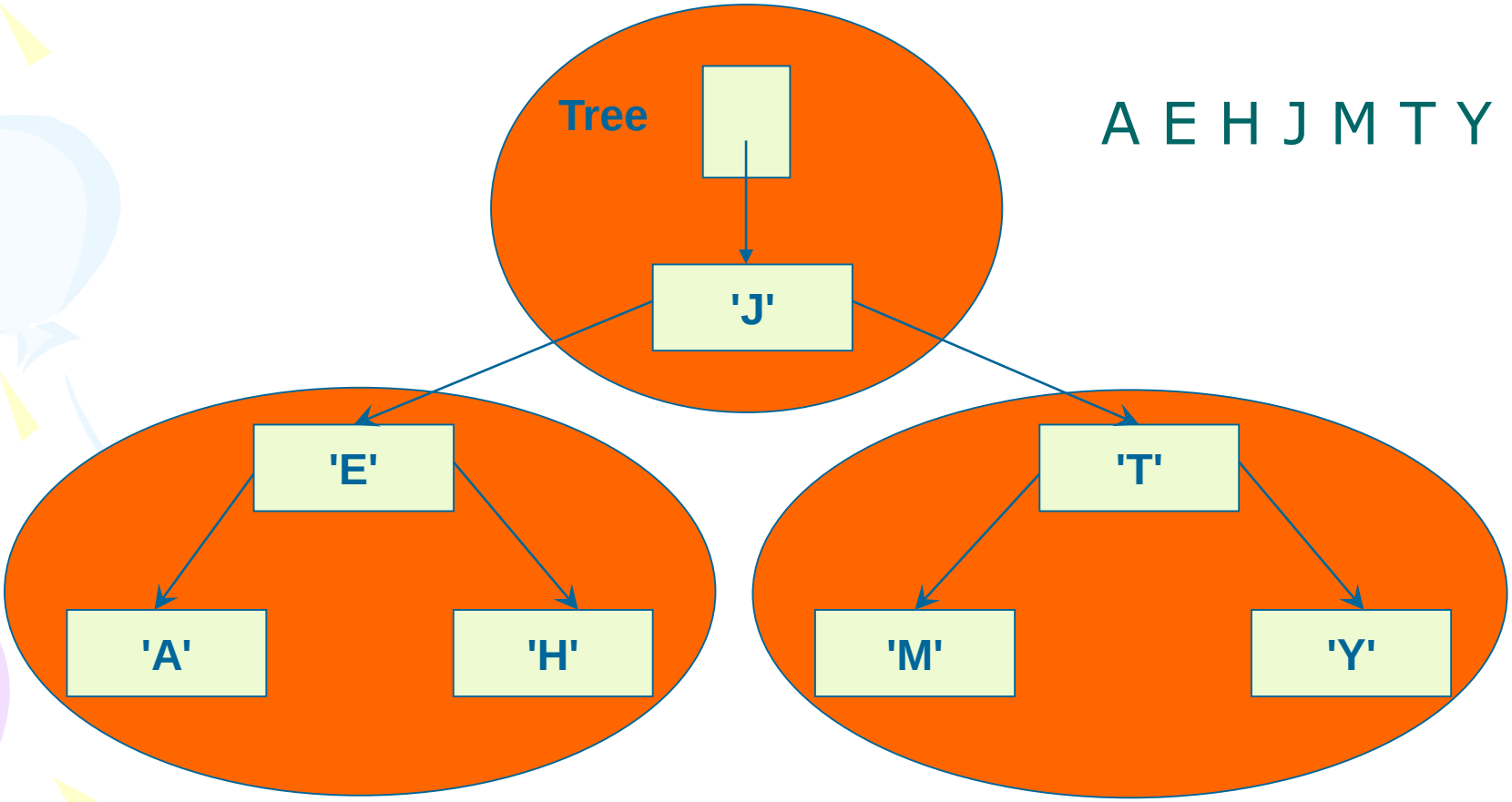
DFS

- Depth-first search (traversal): Duyệt theo chiều sâu: Chiến lược này bao gồm tìm kiếm theo chiều sâu của cây bất cứ khi nào có thể.
- Các loại:

- Thứ tự trước
- Thứ tự giữa
- Thứ tự sau

Inorder Traversal (duyệt theo thứ tự giữa)

- Thăm các node trong cây con trái, sau đó thăm root của cây, rồi thăm các node trong cây con phải.

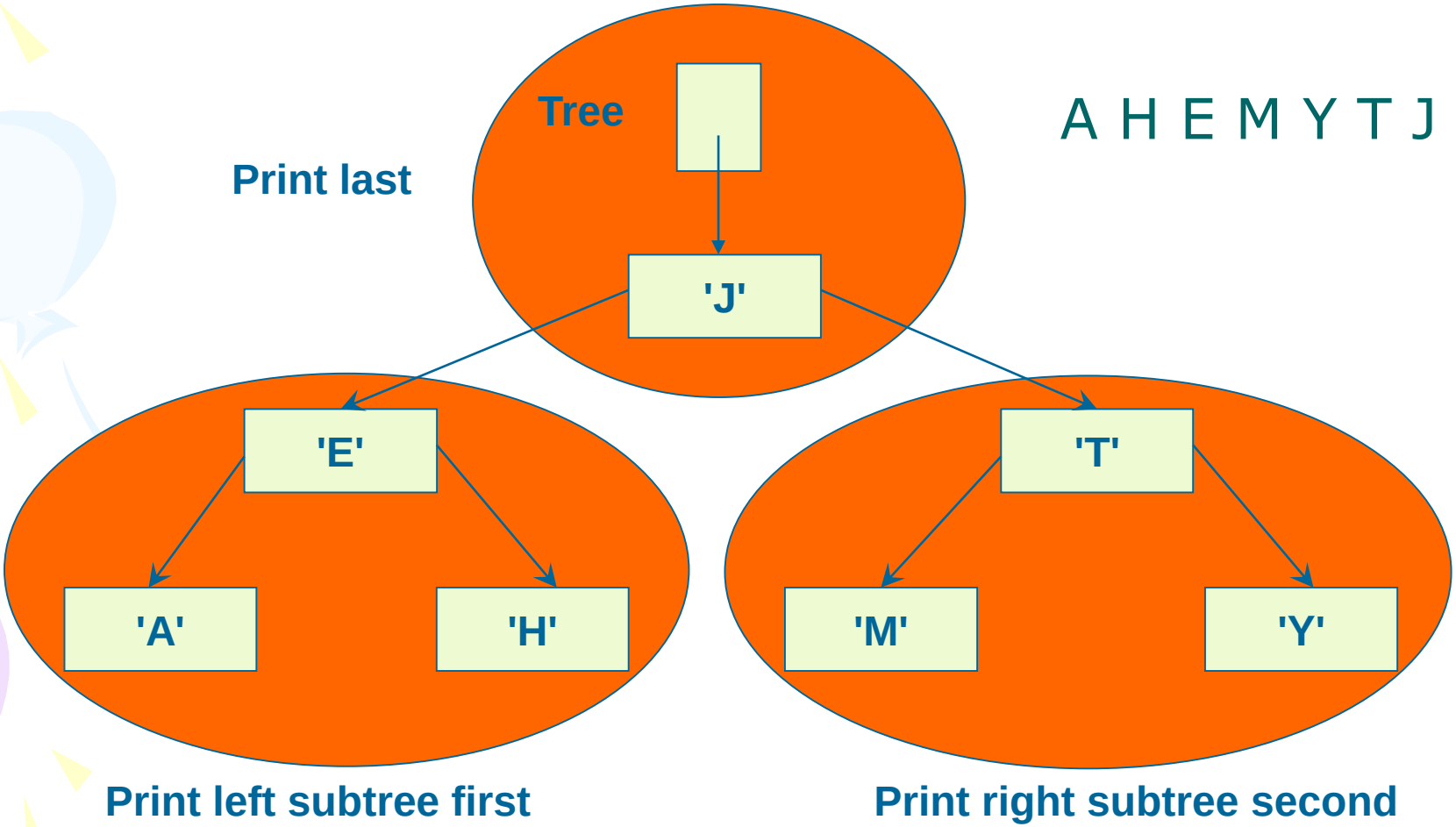


Hàm in theo thứ tự giữa

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

Postorder Traversal (duyệt theo thứ tự sau)

- Thăm các node ở cây con trái, sau đó là các node của cây con phải, rồi đến root.

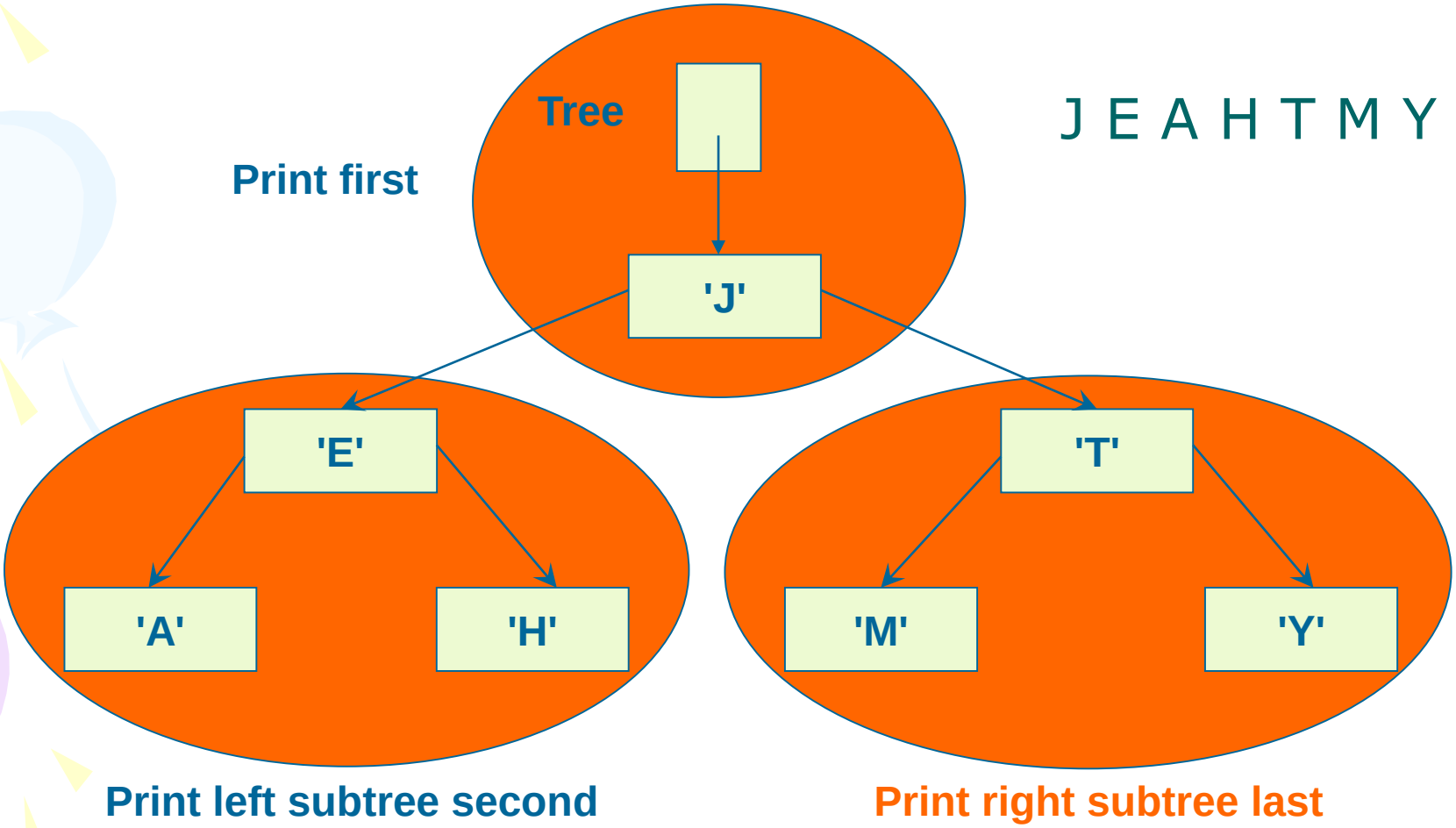


Hàm in theo thứ tự sau

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

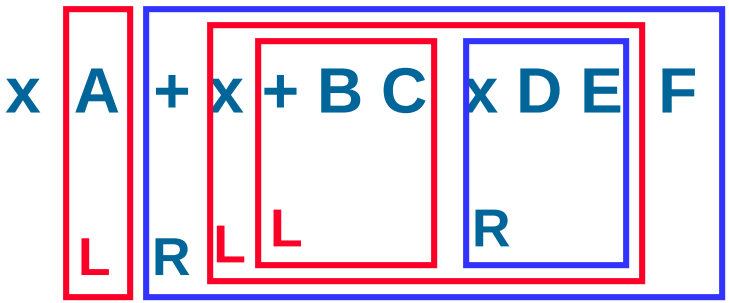
Preorder Traversal (duyệt theo thứ tự trước)

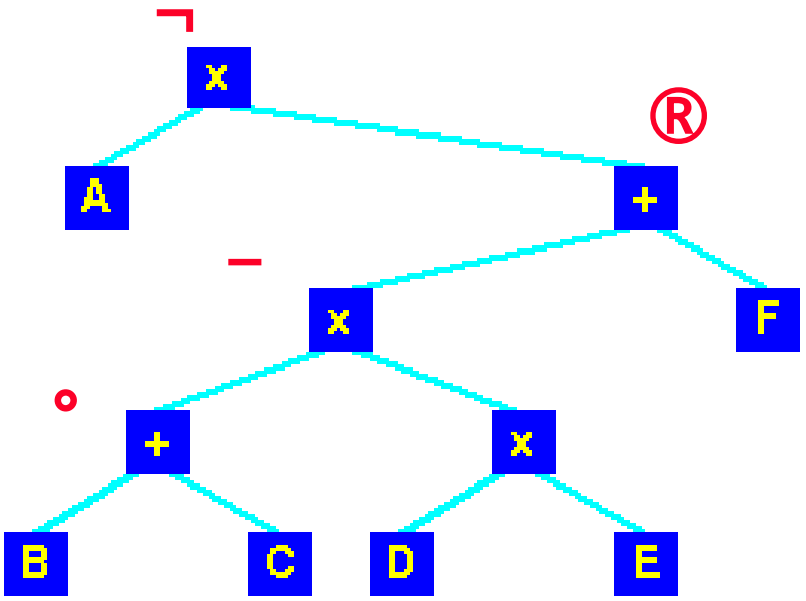
- Thăm root đầu tiên, sau đó thăm đến các node của cây con trái, rồi cây con phải.



Thứ tự trước

- Thứ tự trước
  - Root
  - Cây con trái
  - Cây con phải





Hàm in theo thứ tự trước

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

Exercise

- Trở lại bài tập của tuần trước.Ta đã có 1 cây trữ danh bạ điện thoại.
- Bây giờ, đưa tất cả dữ liệu trong file nhị phân ra theo thứ tự tăng dần của địa chỉ email.
- Gợi ý
  - Chỉ cần sử dụng hàm InOrderTraversal()

Duyệt theo thứ tự giữa bằng cách lặp

```
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;
  }
}
```

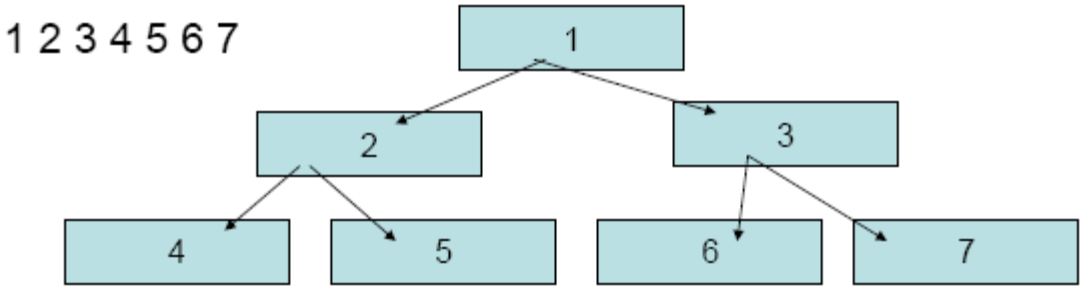
Exercise 2

- Đưa tất cả dữ liệu trong cây phonebook ra theo thứ tự tăng dần theo thứ tự từ điển của name. Đưa ra:

- Màn hình
- File

Breadth First Search (duyệt theo chiều rộng)

- Thay vì đi xuống các con trước thì đi ngang sang anh em.
- Thăm tất cả các node trên 1 mức theo thứ tự từ trái sang phải

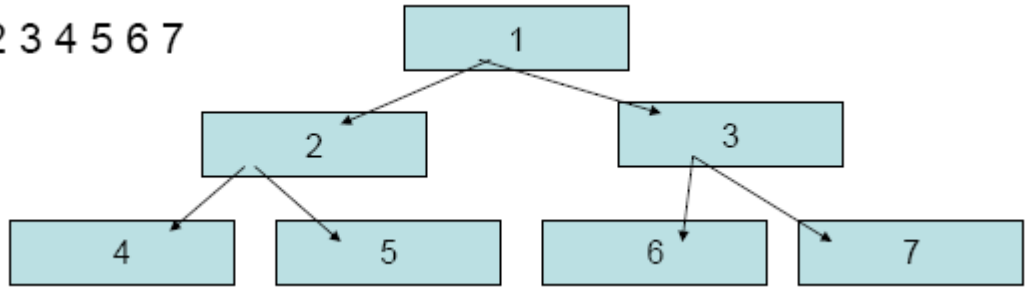


Breadth First Search

- Để xử lí tìm kiếm theo chiều rộng, ta cần 1 hàng đợi thay vì 1 stack.
- Đẩy node root vào hàng đợi
- Với mỗi node từ hàng đợi:

- Thăm node
- Thêm node con trái vào hàng đợi
- Thêm node con phải vào hàng đợi

1 2 3 4 5 6 7



Giả 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

- Khai triển thuật toán BFS trong ngôn ngữ C
- Thêm hàm này vào thư viện cây nhị phân.
- Test nó bằng chương trình quản lí phone book, in ra tất cả các tên trong cây.
- Đưa kết quả ra file.

Exercise

- Viết 1 chương trình xây dựng: 1 cây nhị phân mà dữ liệu của mỗi node trong là bản copy của dữ liệu lớn hơn trong 2 con của nó. Vì thế root sẽ là phần tử lớn nhất.Dữ liệu của các lá tuỳ sở thích.
- Đầu vào được trữ trong 1 mảng.
- Hint (gợi ý): Sử dụng chiến lược chia để trị.

Exercise: Đếm tần suất của từ

- Viết chương trình WordCount đọc 1 file text và phân tích tần suất của các từ. Kết quả được chứa trong 1 file. Khi người sử dụng cung cấp 1 từ, chương trình sẽ trả lại số lần xuất hiện của từ đó trong file.
- Ví dụ: giả sử nội dung file đầu vào như sau:
  - *A black black cat saw a very small mouse and a very scared mouse.*
- Tần suất của các từ trong file này như sau:

AND 1
CAT 1
SAW 1
SCARED 1
SMALL 1
BLACK 2
MOUSE 2
VERY 2
A 3

Gợi ý

- Sử dụng thư viện cây nhị phân (hoặc tốt hơn là cây AVL – cây nhị phân tìm kiếm cân bằng) để lưu trữ dữ liệu.
- 1 node trong cây này chứa ít nhất 2 trường:
- word: string
- count: int //biến đếm số lần
- Các từ được chứa trong các node theo thứ tự từ điển.