Cấu trúc dữ liệu và thuật toán

PGS. TS. Phạm Tuấn Minh
Trường Công nghệ Thông tin, Đại học Phenikaa
minh.phamtuan@phenikaa-uni.edu.vn
https://sites.google.com/site/phamtuanminh/

Chương 2: Mảng và danh sách liên kết

- Cấu trúc lưu trữ mảng
- Danh sách liên kết
- Hàng đợi
- **Ngăn xếp**

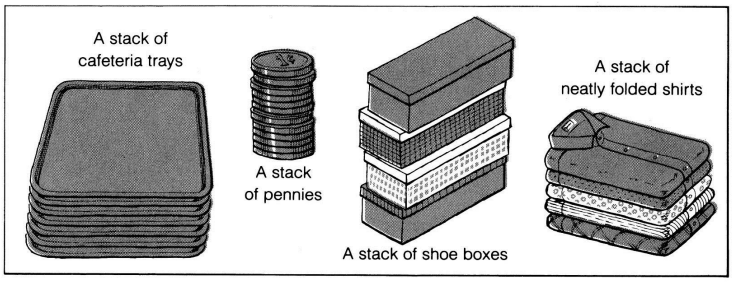
Ngăn xếp

- **Ví dụ ngăn xếp**
- Cấu trúc dữ liệu ngăn xếp
- Cài đặt ngăn xếp dùng danh sách liên kết
- Các thao tác trên ngăn xếp
  - push()
  - pop()
  - peek()
  - isEmptyStack()
- Ví dụ ứng dụng

1-3

Ví dụ

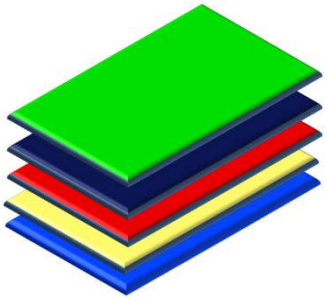
- Ngăn xếp là một nhóm có thứ tự các phần tử
  - Các phần tử được thêm vào và lấy ra từ đầu của danh sách



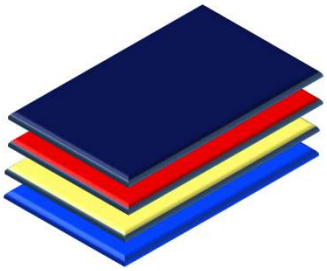
Ngăn xếp

- Bạn chỉ lấy một cái áo bên trên cùng
- Bạn chỉ thêm áo vào trên cùng

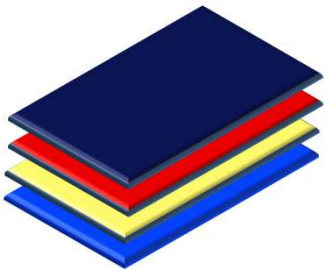
Hôm nay mặc áo nào



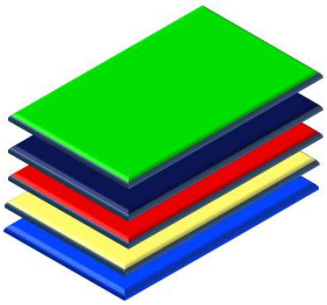
Hôm nay mặc áo nào



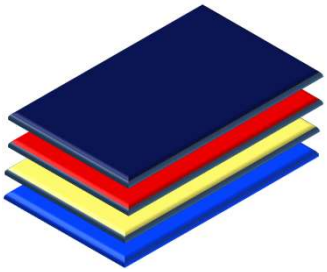
Hôm nay mặc áo nào



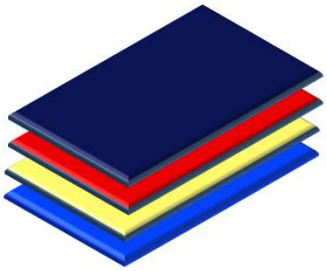
Hôm nay mặc áo nào



Hôm nay mặc áo nào

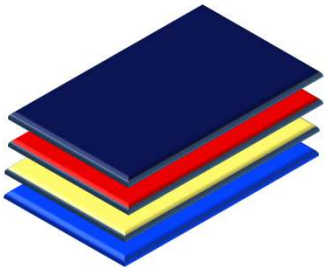


Hôm nay mặc áo nào

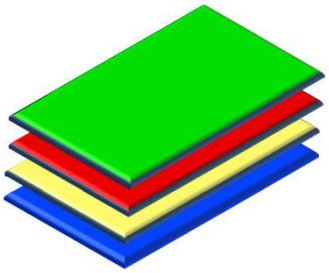


Bạn mặc cùng một cái áo mọi ngày

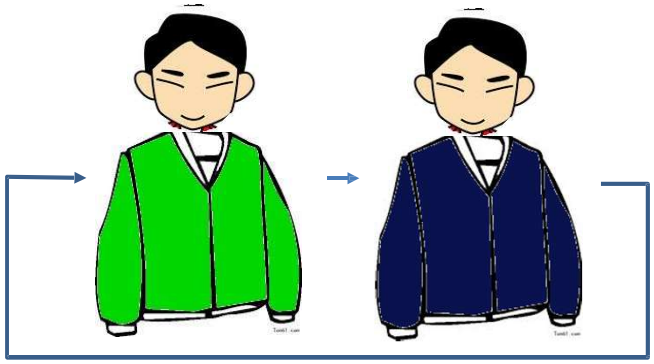
Nếu cần 1 ngày để áo khô



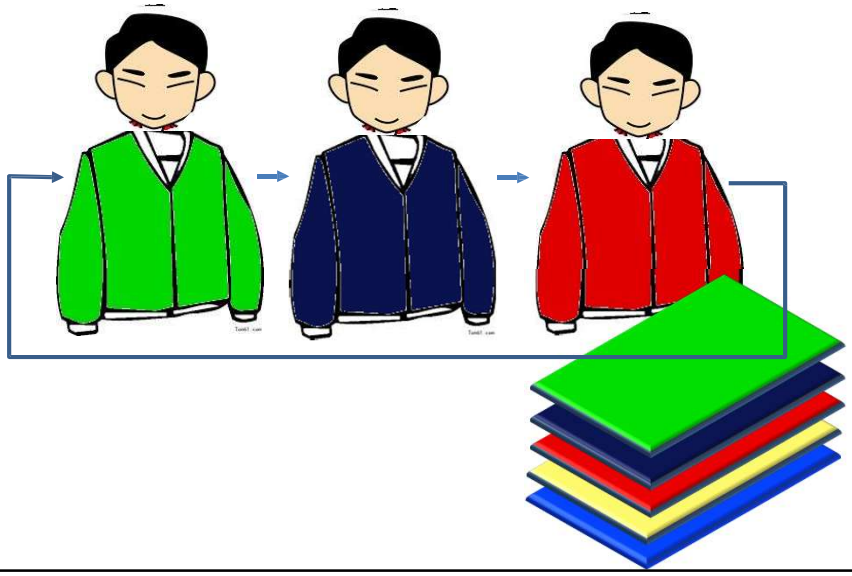
Nếu cần 1 ngày để áo khô



Bạn mặc luân phiên 2 áo



Nếu cần 2 ngày để áo khô



Ngăn xếp

- Ví dụ ngăn xếp
- **Cấu trúc dữ liệu ngăn xếp**
- Cài đặt ngăn xếp dùng danh sách liên kết
- Các thao tác trên ngăn xếp
  - push()
  - pop()
  - peek()
  - isEmptyStack()
- Ví dụ ứng dụng

Mảng, danh sách liên kết, hàng đợi, ngăn xếp

- Mảng
  - Cấu trúc dữ liệu truy cập ngẫu nhiên
  - Truy cập trực tiếp bất kì phần tử nào của mảng
    - array[index]
- Danh sách liên kết
  - Cấu trúc dữ liệu truy cập tuần tự
  - Để truy cập một phần tử phải đi qua các phần tử trước nó
    - cur->next
- Hàng đợi
  - Cấu trúc dữ liệu tuần tự truy cập có giới hạn:
    - Đến trước phục vụ trước (FIFO - First In First Out)
- Ngăn xếp
  - Cấu trúc dữ liệu tuần tự truy cập có giới hạn:
    - Đến cuối phục vụ trước (LIFO - Last In First Out)

Danh sách liên kết, hàng đợi, ngăn xếp

Cấu trúc dữ liệu ngăn xếp

- Ngăn xếp là một cấu trúc dữ liệu hoạt động như một ngăn xếp vật lý
  - Ví dụ ngăn xếp sách
  - Các phần tử chỉ có thể thêm vào hoặc lấy ra từ đỉnh ngăn xếp
- Nguyên tắc: Last-In, First-Out (LIFO)
  - Hoặc First-In, Last-Out (FILO)
- Ngăn xếp được xây dựng dựa trên các cấu trúc dữ liệu khác
  - Ví dụ như mảng, danh sách liên kết
  - Bài giảng tập trung phân tích cài đặt ngăn xếp dựa trên danh sách liên kết

1-20

Ngăn xếp

- Ví dụ ngăn xếp
- Cấu trúc dữ liệu ngăn xếp
- **Cài đặt ngăn xếp dùng danh sách liên kết**
- Các thao tác trên ngăn xếp
  - push()
  - pop()
  - peek()
  - isEmptyStack()
- Ví dụ ứng dụng

1-21

Cài đặt ngăn xếp dùng danh sách liên kết

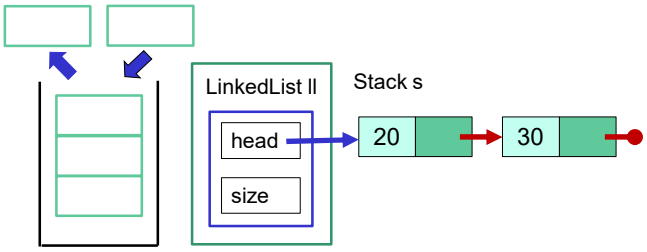
- Cài đặt cấu trúc Stack dựa trên danh sách liên kết

```
typedef struct _stack{
   LinkedList ll;
} Stack;
```

- Sử dụng danh sách liên kết để chứa dữ liệu
- Cần điều chỉnh các thao tác thêm, xóa phần tử

```
typedef struct _listnode {
        int num;
        struct _listnode *next;
} ListNode;

typedef struct _linkedlist {
    ListNode *head;
    int size;
} LinkedList;
```



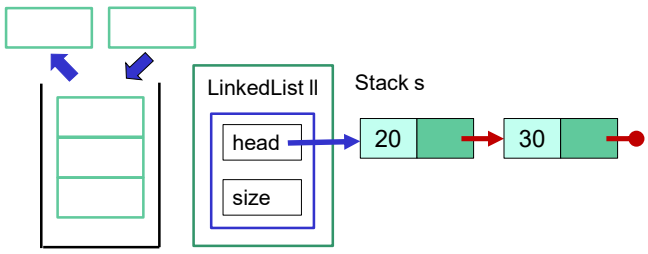
Ngăn xếp

- Ví dụ ngăn xếp
- Cấu trúc dữ liệu ngăn xếp
- Cài đặt ngăn xếp dùng danh sách liên kết
- **Các thao tác trên ngăn xếp**
  - push()
  - pop()
  - peek()
  - isEmptyStack()
- Ví dụ ứng dụng

1-23

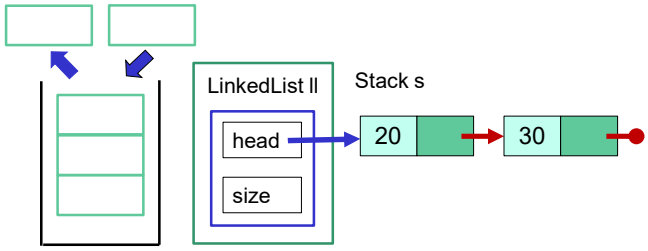
push()

- push() là cách duy nhất để thêm một phần tử vào cấu trúc dữ liệu ngăn xếp
- push() chỉ thao tác trên đỉnh của ngăn xếp
- Sử dụng danh sách liên kết để chứa dữ liệu của ngăn xếp, phần tử đầu tiên của danh sách liên kết là đỉnh hay đáy của ngăn xếp?



Cài đặt hàm push()

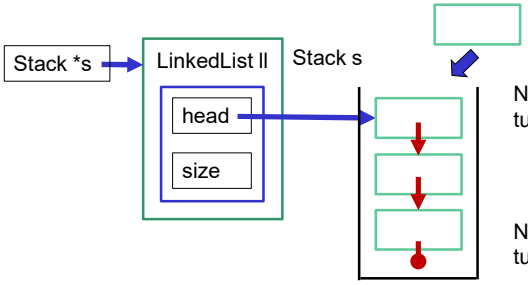
- Cài đặt hàm push()
  - Khai báo prototype
  - Cài đặt hàm
- Yêu cầu
  - Sử dụng các hàm của LinkedList đã cài đặt
  - Chỉ thao tác trên đỉnh ngăn xếp



Cài đặt hàm push()

```
void push(Stack *s, int item ) {
    insertNode(&(s->ll.head), 0, item);
    s->ll.size++;
}
```

Đẩy một nút mới vào ngăn xếp -> thêm một nút mới vào đầu của danh sách liên kết



Nút đầu tiên của danh sách liên kết tương ứng với đỉnh của ngăn xếp

Nút cuối cùng của danh sách liên kết tương ứng với đáy của ngăn xếp

Cài đặt hàm push()

```
void push(Stack *s, int item ) {
    insertNode(&(s->ll.head), 0, item);
    s->ll.size++;
}
```

Hiệu quả
Độ phức tạp O(1)

- Có thể thêm nút mới vào cuối của danh sách liên kết
  - Nếu nút cuối biểu diễn nút đỉnh của ngăn xếp
  - Khi đó cần sử dụng con trỏ tail để thực hiện thao tác hiệu quả

isEmptyStack()

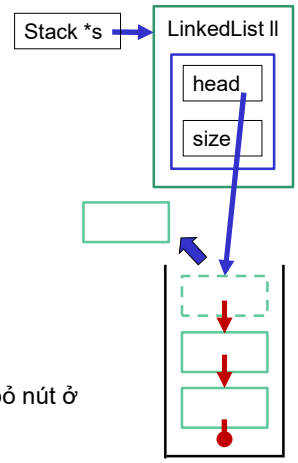
- Kiểm tra xem số phần tử trong ngăn xếp có phải bằng 0 không
- Sử dụng biến size trong cấu trúc LinkedList
- Độ phức tạp O(1)

```
int isEmptyStack(Stack *s) {
    if ((s->ll).size == 0) return 1;
    return 0;
}
```

pop()

- Thao tác lấy một giá trị khỏi ngăn xếp gồm 2 bước
  - Lấy giá trị của nút ở đỉnh của danh sách liên kết
  - Xóa nút này khỏi danh sách liên kết

```
int pop(Stack *s) {
    int item;
    If (!isEmptyStack(s)) {
        item = ((s->ll).head)->num;
        removeNode(&(s->ll.head), 0);
        (s->ll).size--;
        return item;
    }
    else return NULL;
}
```



- Cần biến item để chứa giá trị trước khi loại bỏ nút ở đỉnh ngăn xếp
- Độ phức tạp O(1)

peek()

- peek()
  - Lấy giá trị của nút ở đỉnh của danh sách liên kết
  - Không xóa nút này khỏi danh sách liên kết

```
int peek(Stack *s) {
    if ((s->ll).head)!=NULL)
        return ((s->ll).head)->num;
    else return NULL_VALUE;
}
```

- Độ phức tạp O(1)

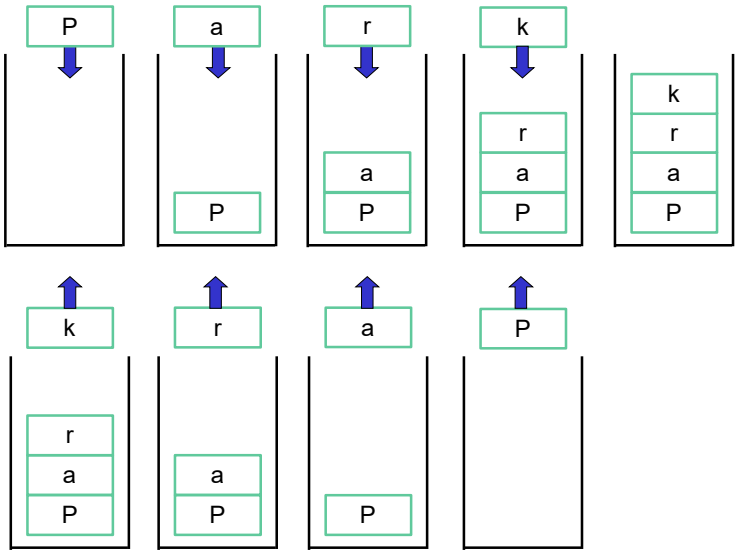
Ngăn xếp

- Ví dụ ngăn xếp
- Cấu trúc dữ liệu ngăn xếp
- Cài đặt ngăn xếp dùng danh sách liên kết
- Các thao tác trên ngăn xếp
  - push()
  - pop()
  - peek()
  - isEmptyStack()
- **Ví dụ ứng dụng**

Ứng dụng 1

- Đảo ngược một chuỗi: Park -> kraP
- Dùng ngăn xếp:
  - Đẩy lần lượt kí tự vào ngăn xếp
  - Khi không còn kí tự nào trong chuỗi ban đầu, lấy lần lượt từng kí tự ra khỏi ngăn xếp

Ứng dụng 1



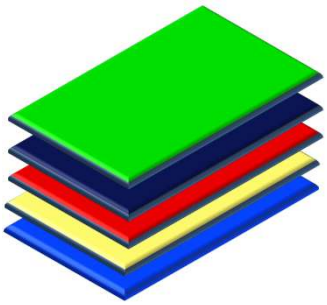
Ứng dụng 2

- Tương tự ứng dụng 1, nhưng với số

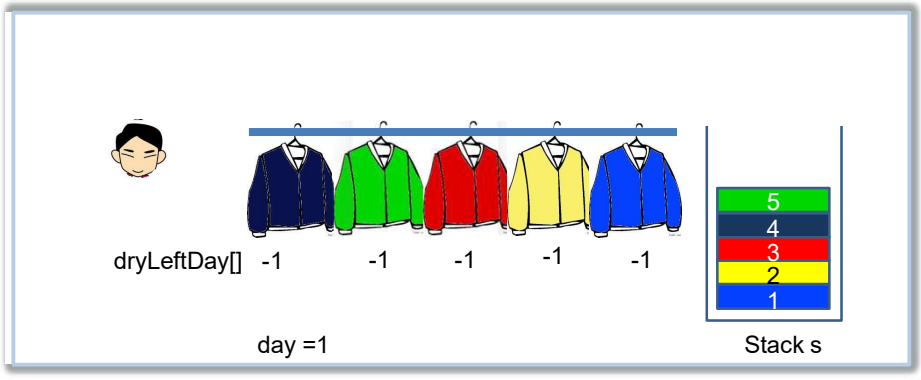
```
int main() {
    int i = 0;
    Stack s;
    s.ll.head = NULL;
    printf("Enter a number: ");
    scanf("%d", &i);
    while (i != -1) {
        push(&s, i);
        printf("Enter a number: ");
        scanf("%d", &i);
    }
    printf("Popping stack: ");
    while (!isEmptyStack(&s))
        printf("%d ", pop(&s));
    return 0;
}
```

Ứng dụng 3: Trang phục hôm nay

- Bạn có 5 áo với màu khác nhau: 1-blue, 2-yellow, 3-red, 4-deep blue, 5-green
- Giả sử cần M=2 ngày để khô áo
- Mô phỏng trang phục của bạn trong 10 ngày



whichClother()



whichClother()

```
void main(){
    int day = 1;
    int j, clothes;
    Stack s;
    int dryLeftDay[6]; //với mỗi áo (số 1 tới số 5), cần bao ngày nữa sẽ khô
    s.ll.head = NULL; s.ll.size=0; //khởi tạo stack s.
    for (j=1; j<=5; j++) push(&s, j);
    for (j=1; j<=5; j++) dryLeftDay[j]= -1; //<0 nghĩa là áo khô
    while (day <=10 ) {
        for (j=1; j<=5; j++)
            dryLeftDay[j]--; //giảm 1 ngày cần để khô
        clothes=pop(&s);
        printf("Day %d is clothes No. %d. \n", day, clothes);
        dryLeftDay[clothes]=M; //ví dụ, M=2, cần 2 ngày để khô
        for (j=1; j<=5; j++)
            if (dryLeftDay[j]==0) //nghĩa là áo vừa khô
                push(&s, j);
        day++;
    }
}
```

Ứng dụng 4

Viết chương trình thực hiện đổi biểu thức trung tố thành biểu thức hậu tố.
Ví dụ:
Biểu thức trung tố: a+b*(c^d-e)^(f+g*h)-i
Biểu thức hậu tố tương ứng: abcd^e-fgh*+^*+i-

Cho khai báo của hàm chuyển đổi như sau:
void infixToPostfix(char* exp, char* postfix)
Input:
- exp là string của biểu thức trung tố.
- postfix là địa chỉ của string chứa biểu thức hậu tố.
Output:
- postfix là string của biểu thức hậu tố.

Chương 2: Mảng và danh sách liên kết

- Cấu trúc lưu trữ mảng
- Danh sách liên kết
- Hàng đợi
- Ngăn xếp

Cấu trúc dữ liệu và giải thuật

- Nội dung bài giảng được biên soạn bởi PGS. TS. Phạm Tuấn Minh.