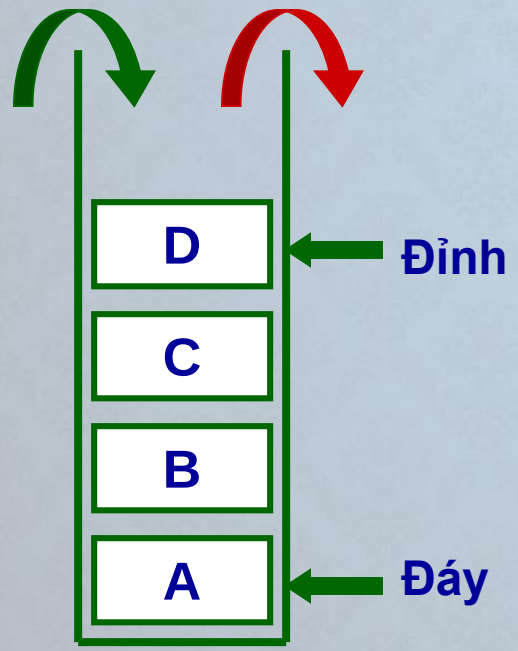
BÀI 4

DANH SÁCH TUYẾN TÍNH

4.5. NGĂN XẾP

4.5.1. Khái niệm ngăn xếp



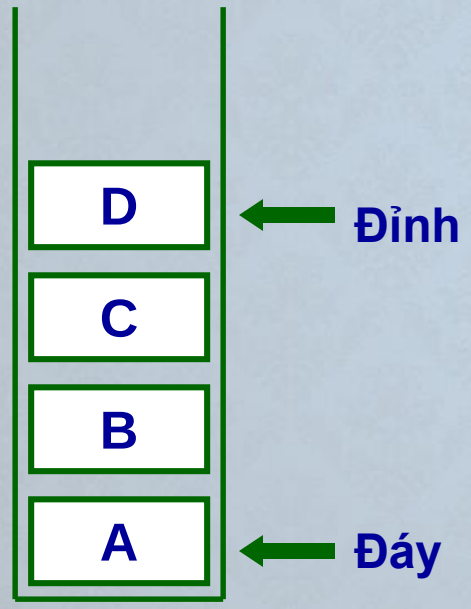
Ngăn xếp

- Là một danh sách tuyến tính.
- Việc bổ sung một phần tử vào ngăn xếp hoặc lấy một phần tử ra khỏi ngăn xếp chỉ thực hiện ở một đầu gọi là đỉnh ngăn xếp.
- Ngăn xếp được gọi là danh sách kiểu LIFO – Last In First Out.

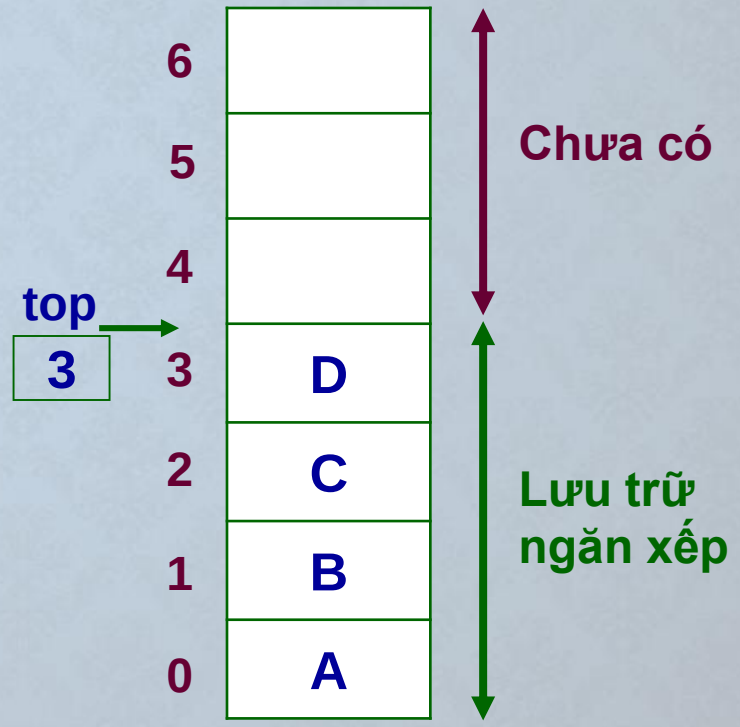
4.5.2. Cài đặt ngăn xếp bằng danh dách kế tiếp

- Cài đặt cấu trúc dữ liệu
- Cài đặt các phép toán cơ bản
  - Khởi tạo ngăn xếp rỗng
  - Kiểm tra ngăn xếp rỗng
  - Kiểm tra ngăn xếp đầy
  - Bổ sung một phần tử vào đỉnh ngăn xếp
  - Lấy một phần tử ở đỉnh ngăn xếp

4.5.2.1. Cài đặt cấu trúc dữ liệu



Ngăn xếp



Mảng e lưu trữ các phần tử
của ngăn xếp

Cài đặt cấu trúc dữ liệu (tt)

- Giả sử N nguyên dương là số phần tử lớn nhất mà ngăn xếp có thể phát triển đến.
- Item là kiểu dữ liệu của các phần tử.
- Khi đó ngăn xếp là một cấu trúc gồm 2 thành phần
  - Biến *top* lưu chỉ số phần tử mảng lưu phần tử đỉnh ngăn xếp.
  - Mảng *e* lưu các phần tử của ngăn xếp.

Cài đặt cấu trúc dữ liệu (tt)

Khai báo kích thước mảng →

```
#define MAX N
```

Khai báo kiểu phần tử →

```
Định nghĩa kiểu Item
```

Khai báo kiểu ngăn xếp →

```
struct Stack
{
    Item e[MAX] ;
    int top ;
} ;
```

Khai báo biến ngăn xếp →

```
Stack S;
```

Cài đặt cấu trúc dữ liệu (tt) – Ví dụ

- Cài đặt cấu trúc dữ liệu của ngăn xếp lưu trữ danh sách 100 học sinh gồm các thông tin:
  - Mã học sinh.
  - Họ tên học sinh.
  - Tuổi.
  - Điểm trung bình.

- Cấu trúc dữ liệu
  - Số học sinh nhiều nhất có thể có N = 100.
  - Kiểu dữ liệu học sinh: Cấu trúc gồm 4 thành phần.
  - Ngăn xếp.

Cài đặt cấu trúc dữ liệu (tt) – Ví dụ

```
#define MAX 100
struct HocSinh {
   int maHs;
   char hoTen[30];
   int tuoi;
   float diemTb;
};
struct Stack {
   HocSinh e[MAX];
   int top;
};
Stack S;
```

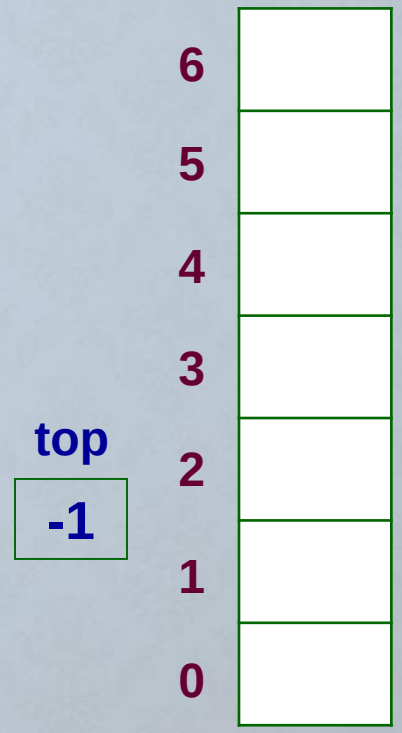
4.5.2.2. Các phép toán cơ bản

- **Khởi tạo ngăn xếp rỗng**

```
void initStack (Stack &S)
{
   S.top = -1;
}
```

- **Kiểm tra ngăn xếp rỗng**

```
int empty (Stack S)
{
   return (S.top == -1);
}
```

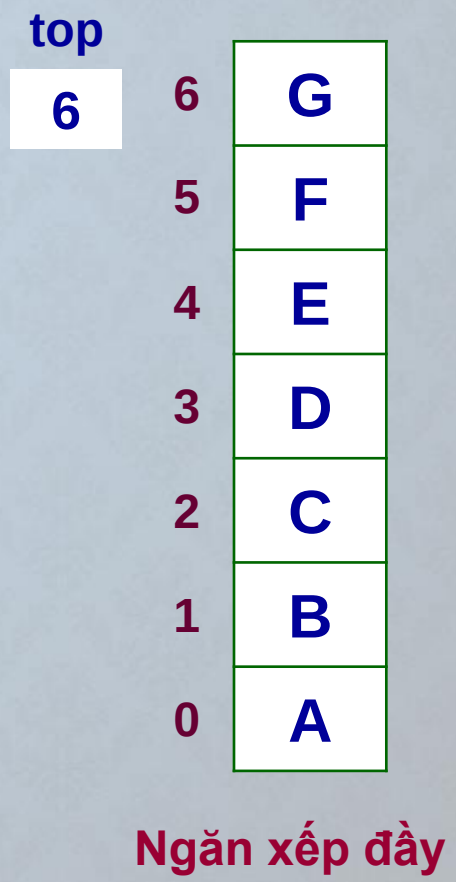


Ngăn xếp rỗng

Các phép toán cơ bản (tt)

- Kiểm tra ngăn xếp đầy

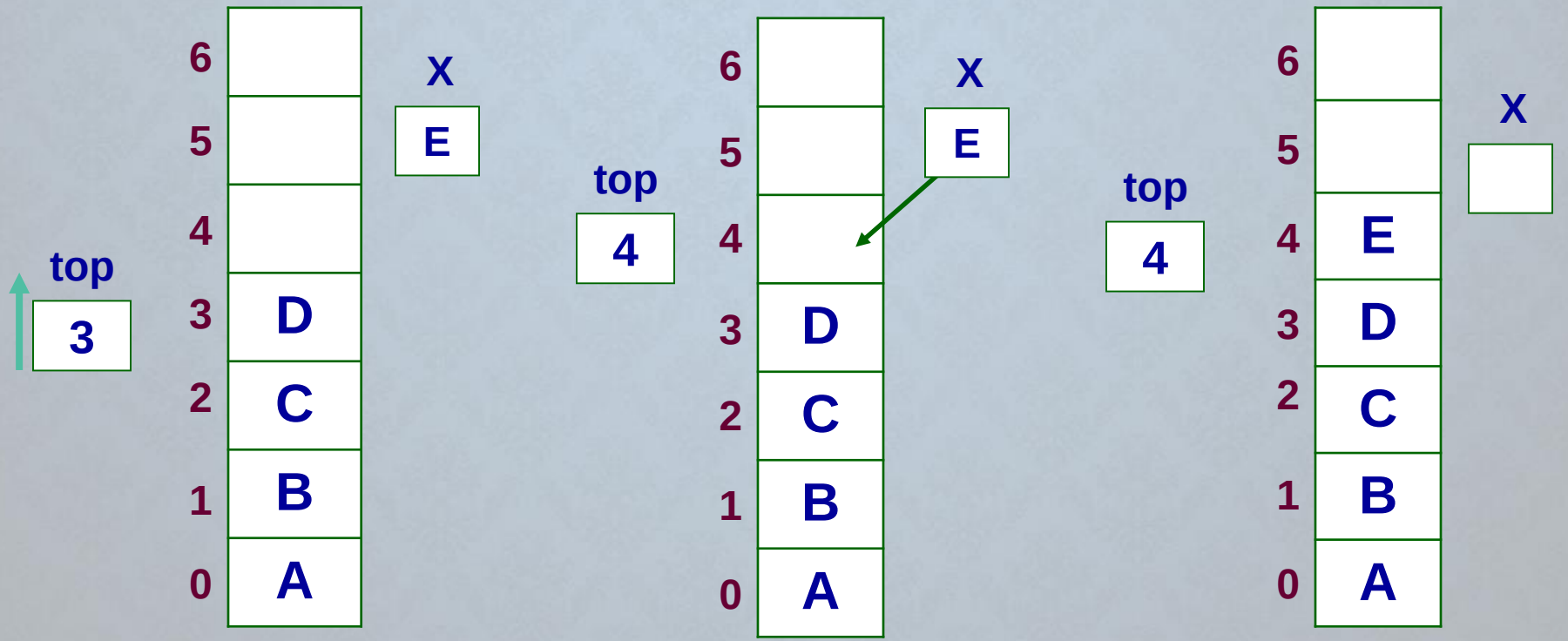
```
int full (Stack S)
{
   return (S.top == MAX-1);
}
```



Ngăn xếp đầy

Các phép toán cơ bản (tt)

- **Bổ sung một phần tử X vào đỉnh ngăn xếp S**



Các phép toán cơ bản (tt)

- **Bổ sung một phần tử X vào đỉnh ngăn xếp S**

  - **Kiểm tra ngăn xếp đầy**
    - **Đúng -> Bổ sung không thành công**
    - **Sai ->**
      - ✓ **Tăng biến top lên 1 đơn vị.**
      - ✓ **Gán giá trị X vào vị trí top mới.**
      - ✓ **Bổ sung thành công.**

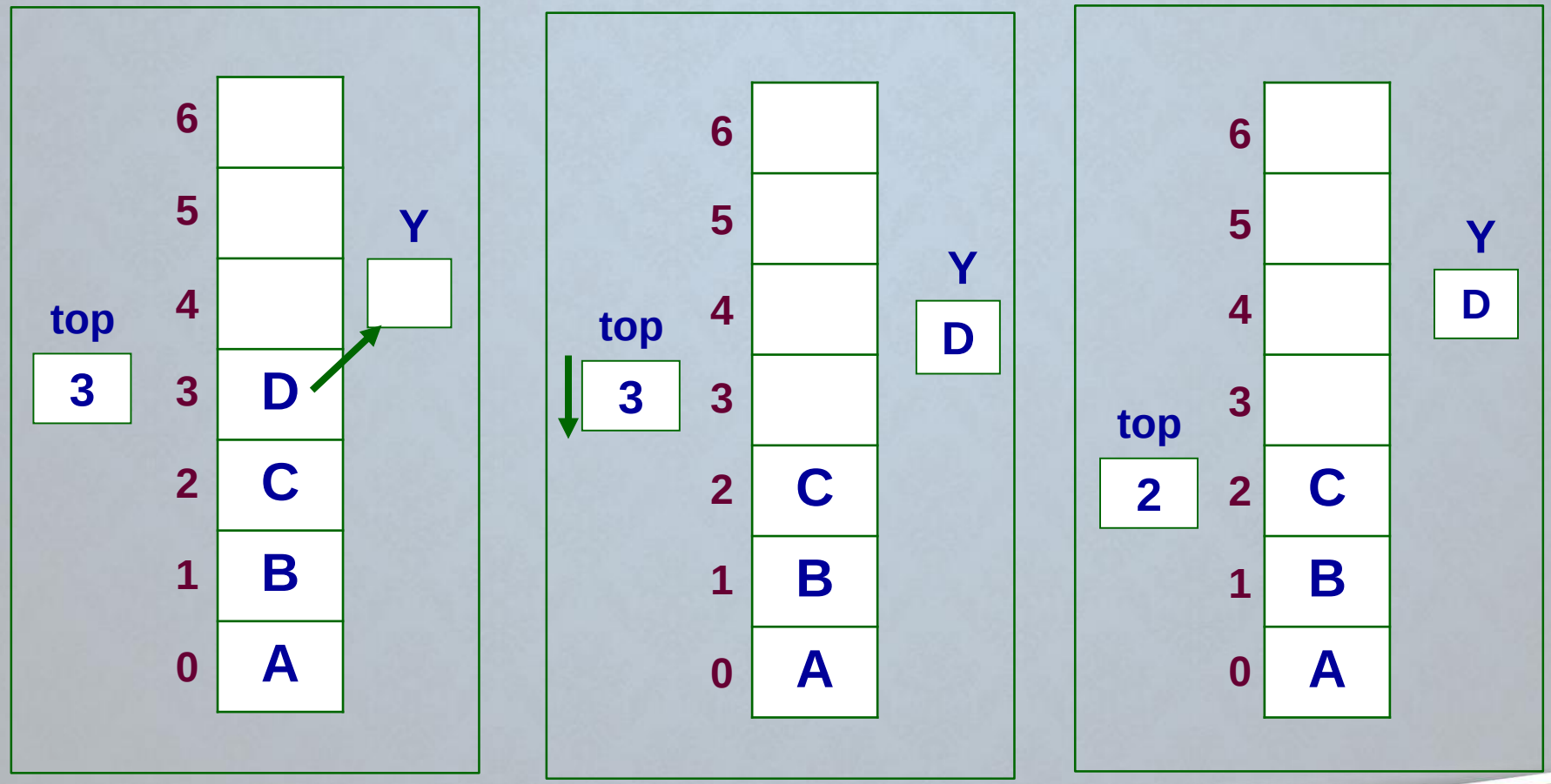
Các phép toán cơ bản (tt)

- **Bổ sung một phần tử X vào đỉnh ngăn xếp S**

```
int push(Stack &S, Item X)
{
   if (full(S)) return 0;
   else
   {
      S.top = S.top + 1;
      S.e[S.top] = X;
      return 1;
   }
}
```

Các phép toán cơ bản (tt)

- Lấy phần tử ở đỉnh ngăn xếp S



Các phép toán cơ bản (tt)

- **Lấy phần tử ở đỉnh ngăn xếp S**

  - **Kiểm tra ngăn xếp rỗng**
    - **Đúng -> Lấy không thành công**
    - **Sai ->**
      - ✓ **Gán giá trị ở đỉnh cho biến Y.**
      - ✓ **Giảm giá trị biến top đi 1 đơn vị.**
      - ✓ **Lấy thành công.**

Các phép toán cơ bản (tt)

- **Lấy phần tử ở đỉnh ngăn xếp S**

```
int pop (Stack &S, Item &Y)
{
   if  (empty(S))
      return 0;
   else
   {
      Y = S.e[S.top];
      S.top = S.top - 1;
      return 1;
   }
}
```

4.5.2.3. Ứng dụng ngăn xếp

- **Chuyển đổi số thập phân sang dạng nhị phân tương ứng.**
- **Bài toán gồm các yêu cầu như sau:**

  - **Nhập số nguyên dương N.**
  - **Đổi số N sang dạng mã nhị phân tương ứng của nó.**
  - **In kết quả ra màn hình.**

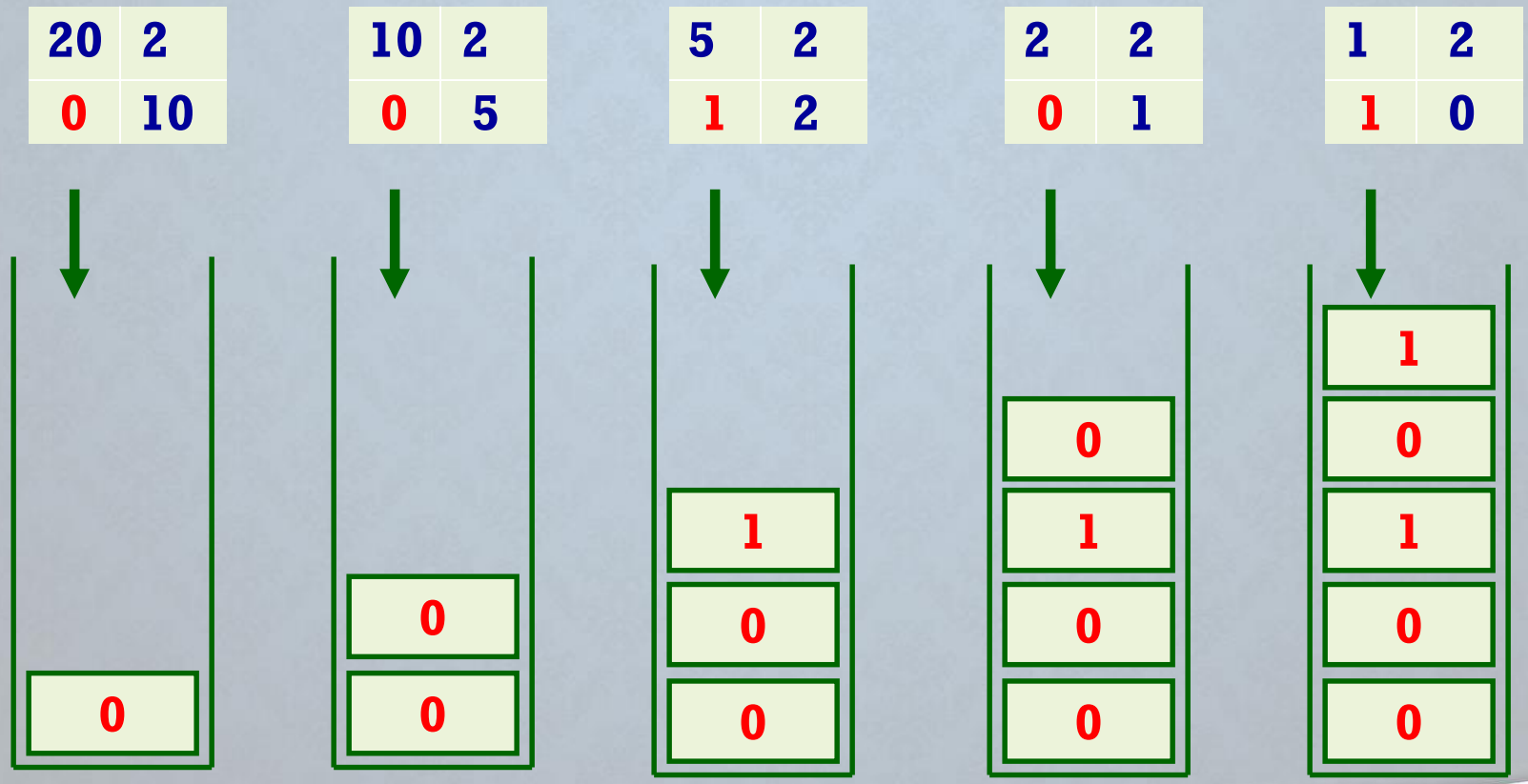
Ứng dụng ngăn xếp (tt)

- **Phương pháp chuyển đổi số thập phân sang dạng số nhị phân tương ứng.**

  - Chia liên tiếp số nguyên N cho 2 cho đến khi N = 0.
  - Lưu trữ các số dư trong mỗi lần chia.
  - Lấy các số dư theo thứ tự ngược lại của thứ tự chia ta được số nhị phân tương ứng.

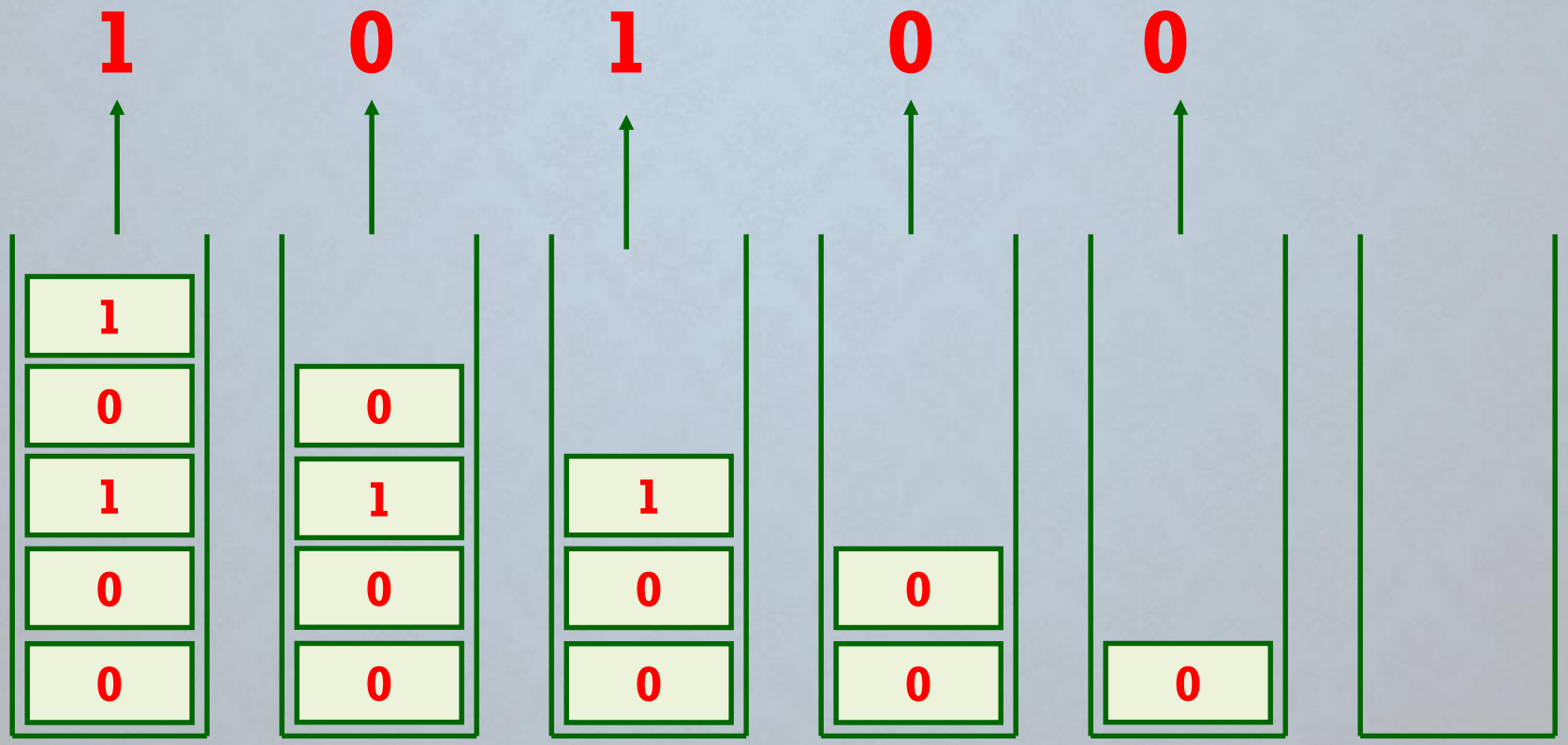
Các số dư được lưu như thế nào ?

- Chia số N liên tiếp cho 2 và lưu phần dư trong mỗi lần chia vào ngăn xếp S.
- Với N = 20 ta có:



Ứng dụng ngăn xếp (tt)

- Đọc mã nhị phân từ ngăn xếp S và hiển thị ra màn hình.



- **Cài đặt cấu trúc dữ liệu của bài toán.**

  - Số nguyên được lưu với kích thước 4 byte = 32 bit, nên N = 32 là kích thước của ngăn xếp.
  - Mã nhị phân là các giá trị 0, 1 -> dữ liệu trong ngăn xếp là số nguyên.

```
#define MAX 32
typedef unsigned int MaNhiPhan;
struct Stack
{
   int top;
   MaNhiPhan e[MAX];
};
```

Ứng dụng ngăn xếp (tt)

- Cài đặt hàm chuyển đổi số thập phân sang dạng nhị phân, kết quả lưu trong ngăn xếp S.

```
void change(unsigned long N, Stack &S)
{
    initStack(S);
    while (N > 0 && push(S, N % 2))
    {
        N = N / 2;
    }
}
```

Ứng dụng ngăn xếp (tt)

- Cài đặt hàm đọc mã nhị phân lưu trong ngăn xếp S và hiển thị kết quả ra màn hình.

```
void display(Stack S)
{
   MaNhiPhan Y;
   while (pop(S, Y))
   {
      cout<<Y<<" ";
   }
}
```

Bài tập

- **Cài đặt chương trình thực hiện các yêu cầu sau:**
  - Cài đặt ngăn xếp lưu trữ danh sách các số nguyên
  - Tạo ngăn xếp chứa n số nguyên (dữ liệu nhập từ bàn phím).
  - Đưa danh sách lên màn hình.
  - Thêm một số nguyên vào đáy ngăn xếp, hiển thị lại ngăn xếp.
  - Xóa số nguyên thứ 2 tính từ đáy ngăn xếp, hiển thị lại ngăn xếp.

TRÂN TRỌNG CẢM ƠN…!