CANTHO UNIVERSITY

Cấu trúc dữ liệu

CÁC KIỂU DỮ LIỆU TRỪU TƯỢNG CƠ BẢN (BASIC ABSTRACT DATA TYPES)

Bộ môn Công Nghệ Phần Mềm

MỤC TIÊU

- Nắm vững các kiểu dữ liệu trừu tượng như: danh sách, ngăn xếp, hàng đợi.
- Cài đặt các kiểu dữ liệu trừu tượng bằng ngôn ngữ lập trình cụ thể.
- Ứng dụng được các kiểu dữ liệu trừu tượng trong bài toán thực tế.

CANTHO UNIVERSITY

NỘI DUNG

- Kiểu dữ liệu trừu tượng danh sách (LIST)
- Kiểu dữ liệu trừu tượng ngăn xếp (STACK)
- Kiểu dữ liệu trừu tượng hàng đợi (QUEUE)
- Danh sách liên kết kép (Double – Lists)

NGĂN XẾP (STACK)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT
  - CÀI ĐẶT BẰNG DANH SÁCH LIÊN KẾT
  - CÀI ĐẶT BẰNG MẢNG

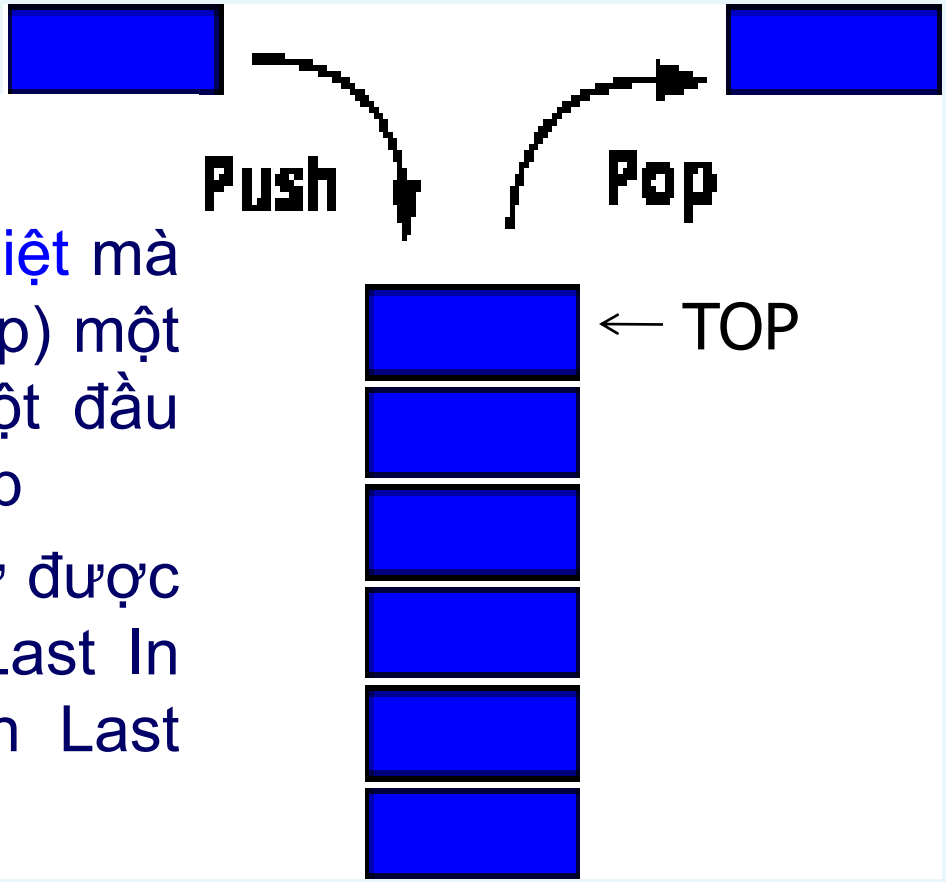
NGĂN XẾP (STACK)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT
  - CÀI ĐẶT BẰNG DANH SÁCH LIÊN KẾT
  - CÀI ĐẶT BẰNG MẢNG

ĐỊNH NGHĨA

Ngăn xếp:

- Là một dạng danh sách đặc biệt mà việc thêm (Push) hay xóa (Pop) một phần tử chỉ thực hiện tại một đầu gọi là đỉnh (TOP) của ngăn xếp
- Việc thêm và xóa một phần tử được thực hiện theo dạng LIFO (Last In First Out) hay FILO (First In Last Out)



NGĂN XẾP (STACK)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT
  - CÀI ĐẶT BẰNG DANH SÁCH LIÊN KẾT
  - CÀI ĐẶT BẰNG MẢNG

CÁC PHÉP TOÁN

Phép toán	Diễn giải
makenullStack(S)	Tạo một ngăn xếp rỗng (S)
emptyStack(S)	Kiểm tra xem ngăn xếp S có rỗng hay không. Hàm cho kết quả 1 (true) nếu ngăn xếp rỗng và 0 (false) trong trường hợp ngược lại.
full(S)	Kiểm tra xem ngăn xếp S có đầy hay không
push(*x,S*)	Thêm phần tử x vào đỉnh ngăn xếp S. Tương đương: insertList(x,first(S),S)
pop(S)	Xóa phần tử tại đỉnh ngăn xếp S. Tương đương: deleteList(first(S),S)
top(S)	Trả về phần tử đầu tiên trên đỉnh ngăn xếp S, tương đương: retrieve(first(S),S)

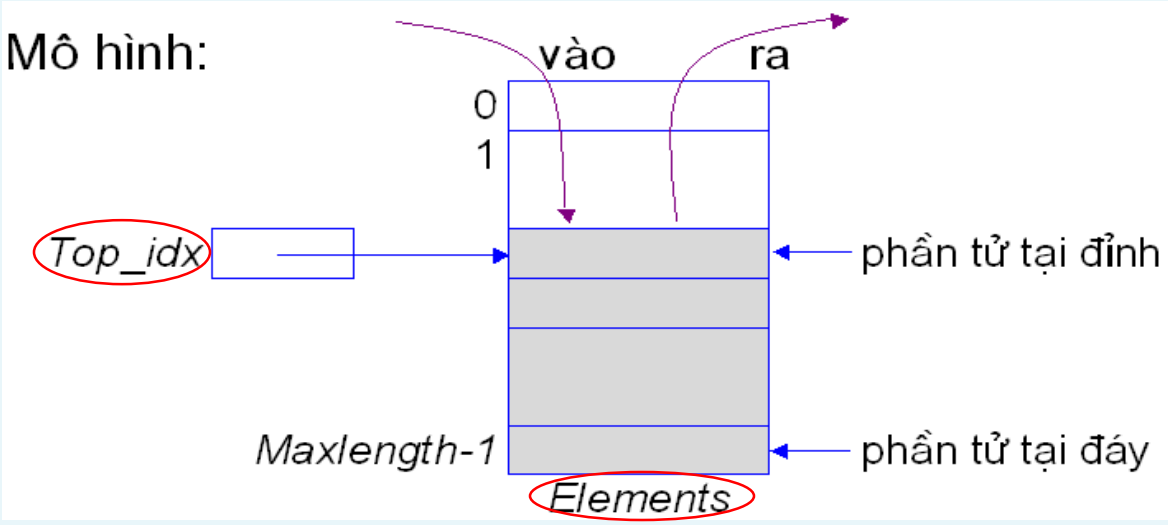
NGĂN XẾP (STACK)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT
  - CÀI ĐẶT BẰNG MẢNG
  - CÀI ĐẶT BẰNG DANH SÁCH

Kiểu dữ liệu trừu tượng - Lưu ý

- Cài đặt kiểu dữ liệu trừu tượng:
  - Tổ chức lưu trữ: **cấu trúc dữ liệu** (*khai báo dữ liệu*).
  - Viết **chương trình con** thực hiện các phép toán (khai báo phép toán).

CÀI ĐẶT NGĂN XẾP BẰNG MẢNG



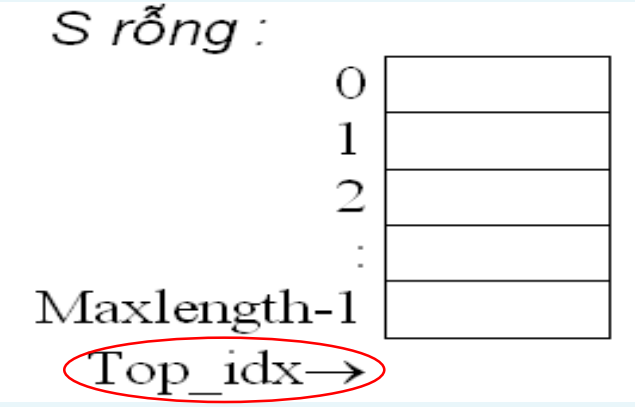
- Khai báo

```
#define MaxLength <n>                    //độ dài của mảng
typedef <datatype> ElementType;//kiểu phần tử của ngăn xếp
typedef struct {
    //Lưu nội dung của các phần tử
    ElementType Elements[MaxLength];
    int         Top_idx; //giữ vị trí đỉnh ngăn xếp
}Stack;
```

KHỞI TẠO NGĂN XẾP RỖNG

- Khai báo

```
#define MaxLength <n>                    //độ dài của mảng
typedef <datatype> ElementType;//kiểu phần tử của ngăn xếp
typedef struct {
    //Lưu nội dung của các phần tử
    ElementType Elements[MaxLength];
    int         Top_idx; //giữ vị trí đỉnh ngăn xếp
}Stack;
```

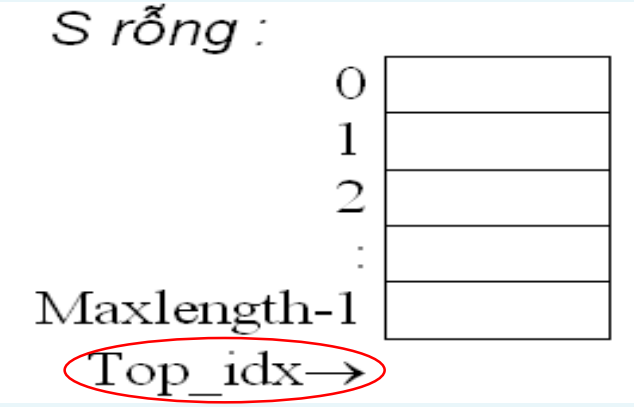



```
<kiểu kết quả>  Tên hàm (          Tham số hình thức
                    [<kiểu t số> <tham số>]
                    [,<kiểu t số><tham số>][…])
{
      [Khai báo biến cục bộ]
      [Các câu lệnh thực hiện hàm]
      [return [<Biểu thức>];]
}
```

KHỞI TẠO NGĂN XẾP RỖNG

- Khai báo

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
    ElementType Elements[MaxLength];
    int         Top_idx;
}Stack;
```



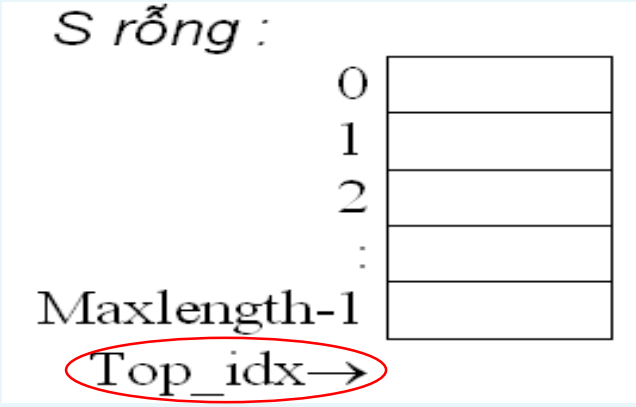
– Khi ngăn xếp S rỗng ta cho đỉnh ngăn xếp được khởi tạo bằng Maxlength

```
void makenullStack(Stack *pS){
        pS->Top_idx=MaxLength;
}
```

KIỂM TRA NGĂN XẾP RỖNG?

- Khai báo

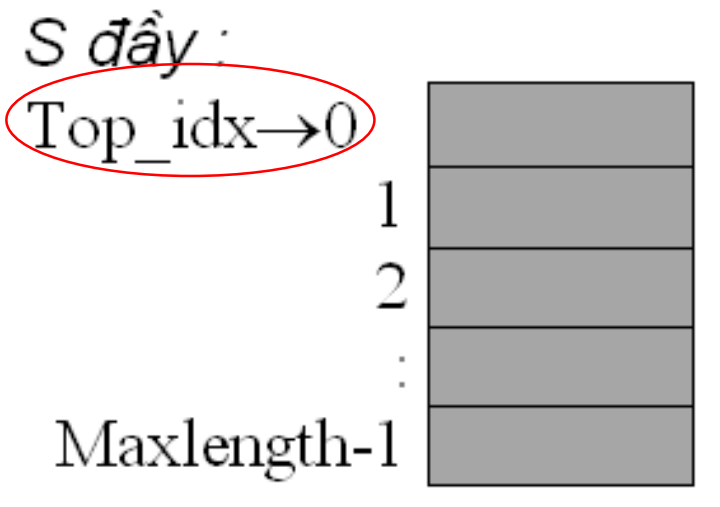
```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
    ElementType Elements[MaxLength];
    int         Top_idx;
}Stack;
```



– Ta kiểm tra xem đỉnh ngăn xếp có bằng MaxLength không?

```
int emptyStack(Stack S){
    return S.Top_idx==MaxLength;
}
```

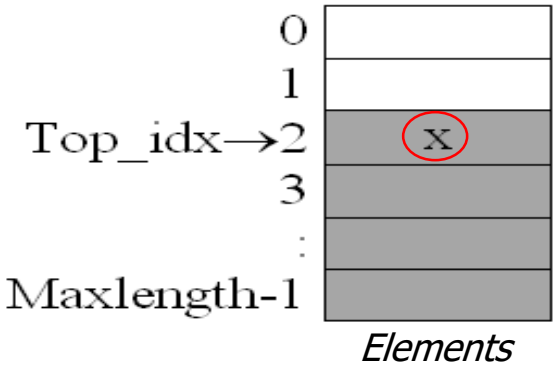
KIỂM TRA NGĂN XẾP ĐẦY?



– Ta kiểm tra xem Top_idx có chỉ vào 0 hay không?

```
int full(Stack S){
      return S.Top_idx==0;
}
```

TRẢ VỀ PHẦN TỬ ĐẦU NGĂN XẾP



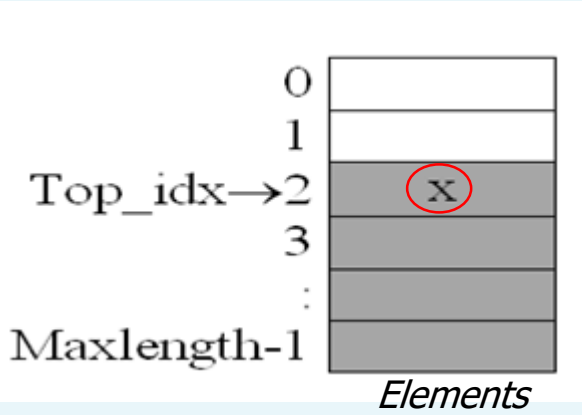
Kết quả của phép toán trên ngăn xếp là x

- Giải thuật:
  - Nếu ngăn xếp rỗng thì thông báo lỗi
  - Ngược lại, trả về giá trị được lưu trữ tại ô có chỉ số là Top_idx

TRẢ VỀ PHẦN TỬ ĐẦU NGĂN XẾP

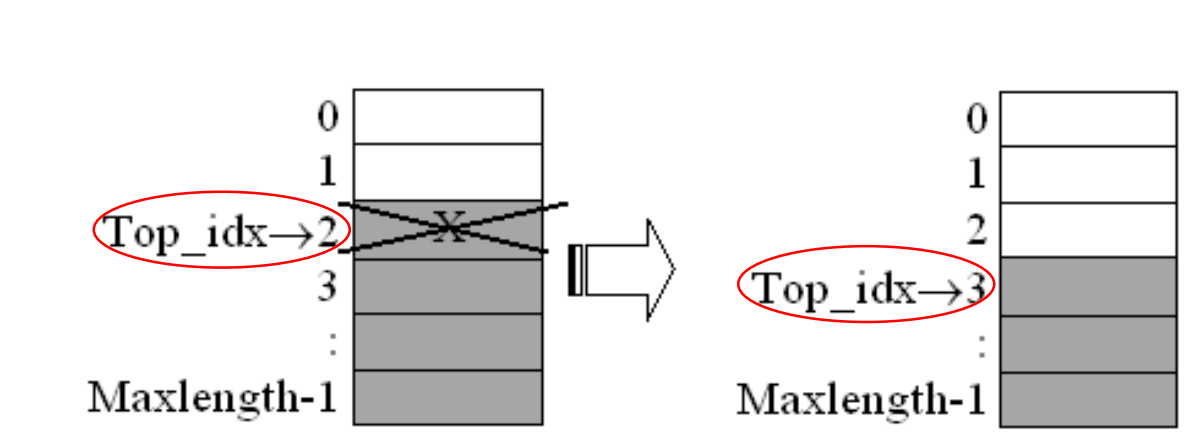
- Khai báo

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
    ElementType Elements[MaxLength];
    int         Top_idx;
}Stack;
```



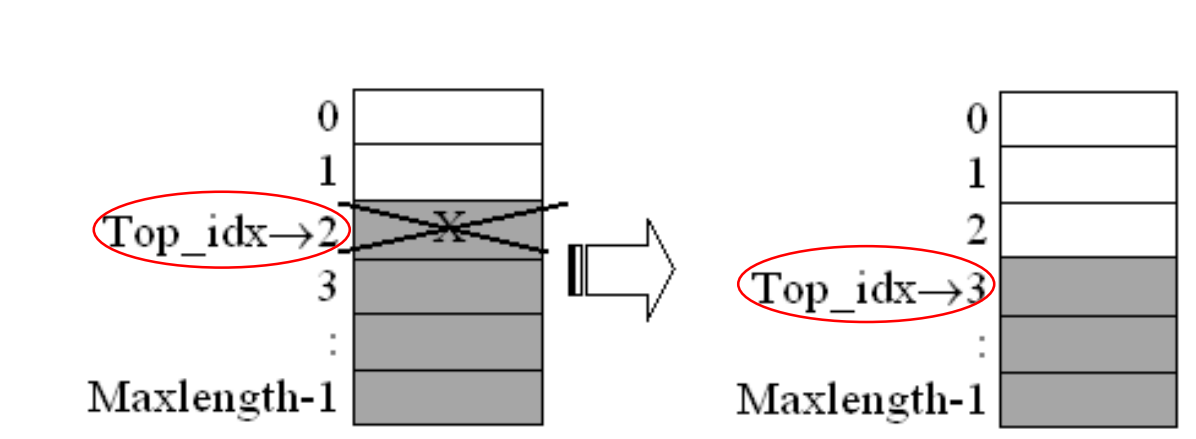

```
ElementType top(Stack S){
    if (!emptyStack(S))
        return S.Elements[S.Top_idx];
    else printf("Loi! Ngan xep rong");
}
```

XÓA PHẦN TỬ TẠI ĐỈNH NGĂN XẾP



- Giải thuật:
  - Nếu ngăn xếp rỗng thì thông báo lỗi
  - Ngược lại, tăng Top_idx lên 1 đơn vị

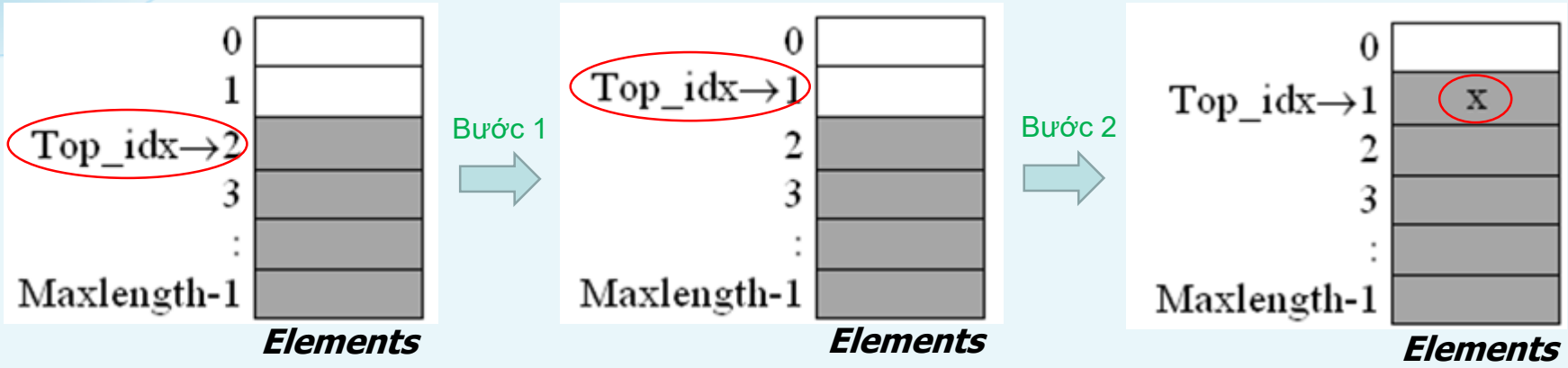
XÓA PHẦN TỬ TẠI ĐỈNH NGĂN XẾP



- Giải thuật:
  - Nếu ngăn xếp rỗng thì thông báo lỗi
  - Ngược lại, tăng Top_idx lên 1 đơn vị

```
void pop(Stack *pS){
    if (!emptyStack(*pS))
        pS->Top_idx=pS->Top_idx+1;
    else printf("Loi! Ngan xep rong!");
}
```

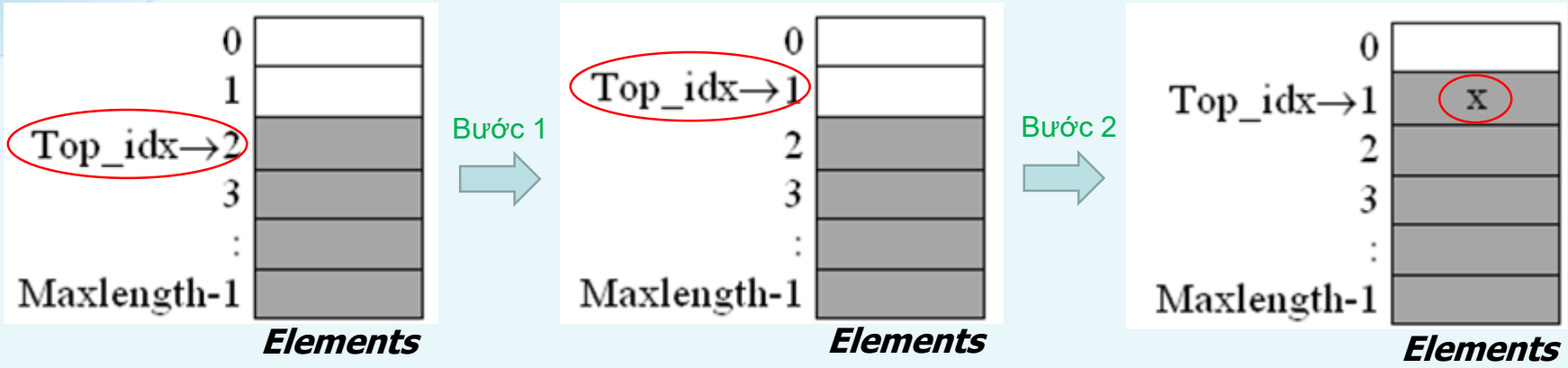
THÊM PHẦN TỬ x VÀO NGĂN XẾP



Giải thuật :

- Nếu ngăn xếp đầy thì thông báo lỗi
- Ngược lại, giảm Top_idx xuống 1 đơn vị rồi đưa giá trị x vào ô có chỉ số Top_idx

THÊM PHẦN TỬ x VÀO NGĂN XẾP




```
void push(ElementType x, Stack *pS){
   if (full(*pS))
         printf("Loi! Ngan xep day!");
   else {
         pS->Top_idx=pS->Top_idx-1;
         pS->Elements[pS->Top_idx]=x;
    }
}
```

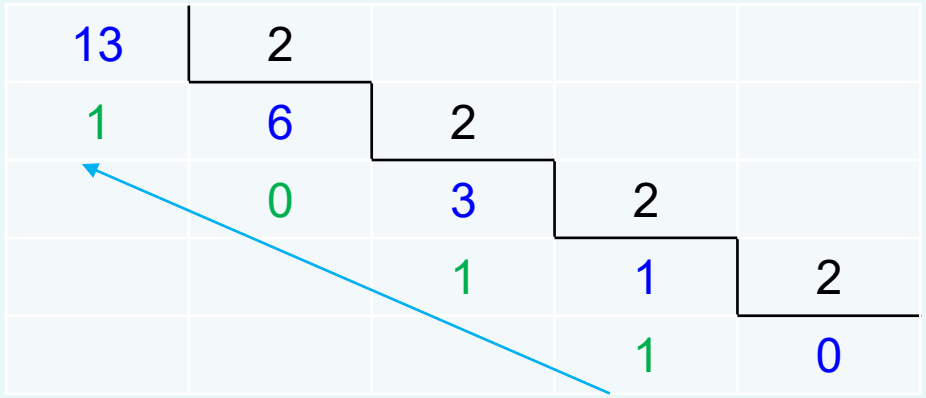
BÀI TẬP

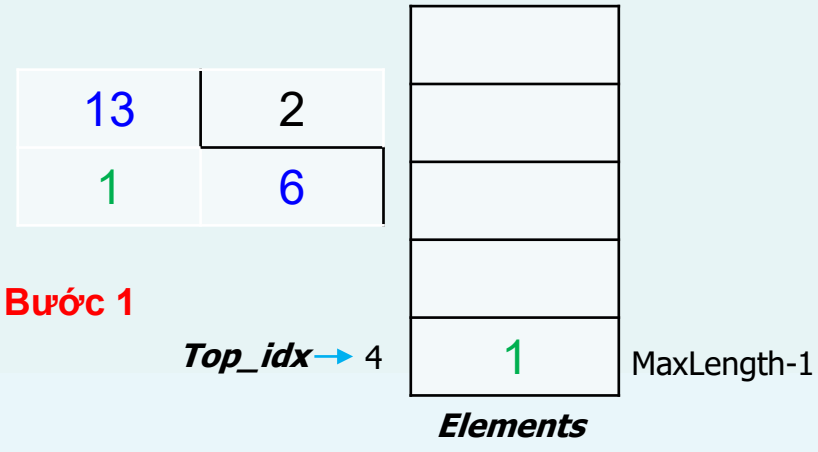
– Sử dụng các phép toán trên ngăn xếp, viết hàm **void Print_Binary(int n)**, nhận vào 1 số nguyên không âm n và in ra biểu diễn nhị phân của số n?

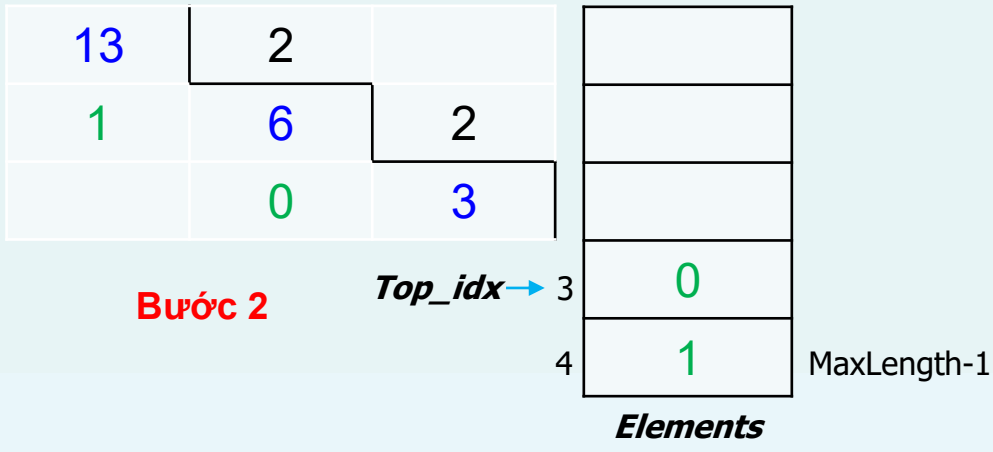
CANTHO UNIVERSITY

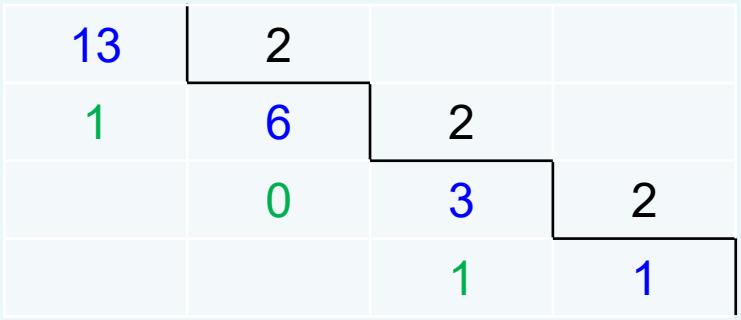
BÀI TẬP

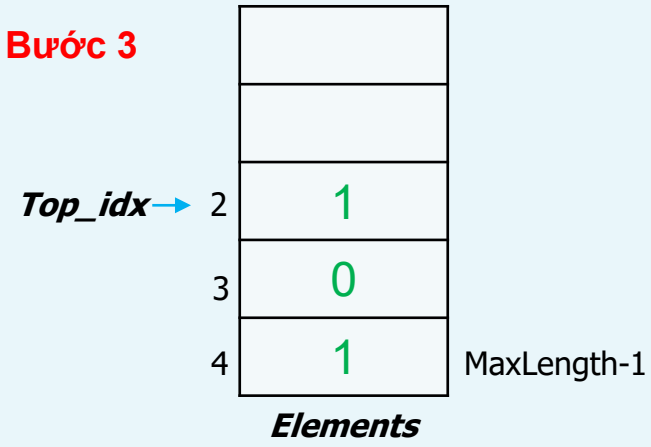
– Sử dụng các phép toán trên ngăn xếp, viết hàm **void Print_Binary(int n)**, nhận vào 1 số nguyên không âm n và in ra biểu diễn nhị phân của số n?

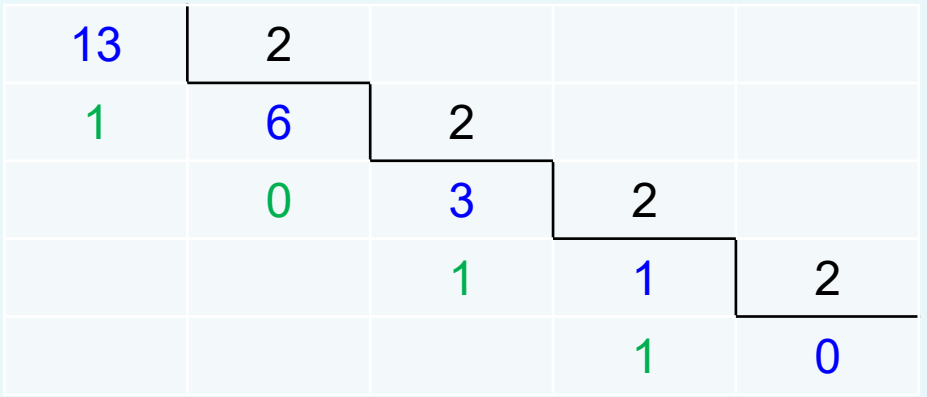


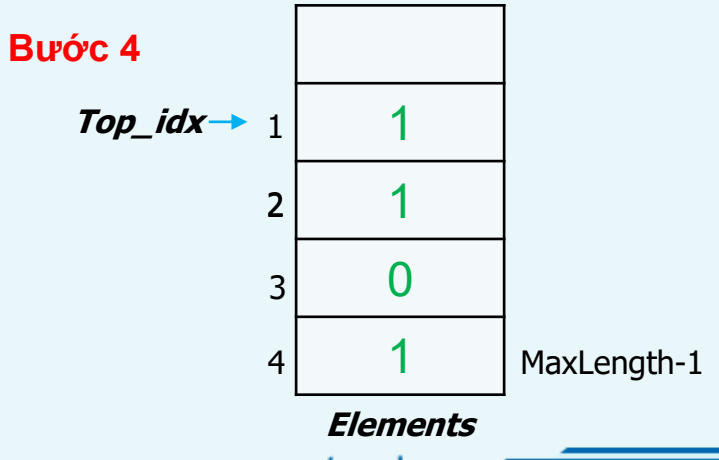

13 2
1 6
Bước 1
Top_idx → 4
1
MaxLength-1
Elements


13 2
1 6 2
0 3
Bước 2
Top_idx → 3
0
4
1
MaxLength-1
Elements


13 2
1 6 2
0 3 2
1 1

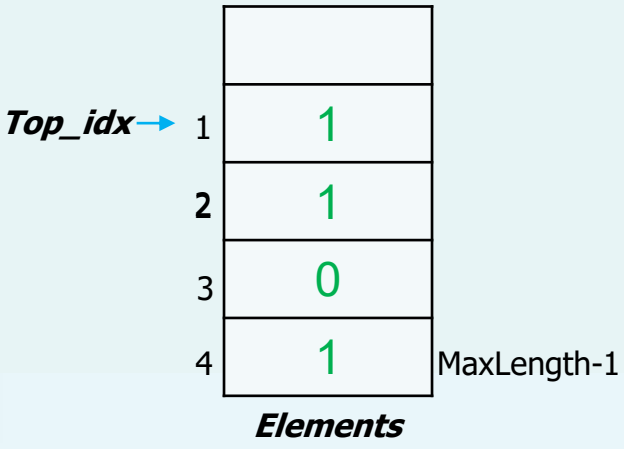

Bước 3
Top_idx → 2
1
3
0
4
1
MaxLength-1
Elements


13 2
1 6 2
0 3 2
1 1 2
1 0

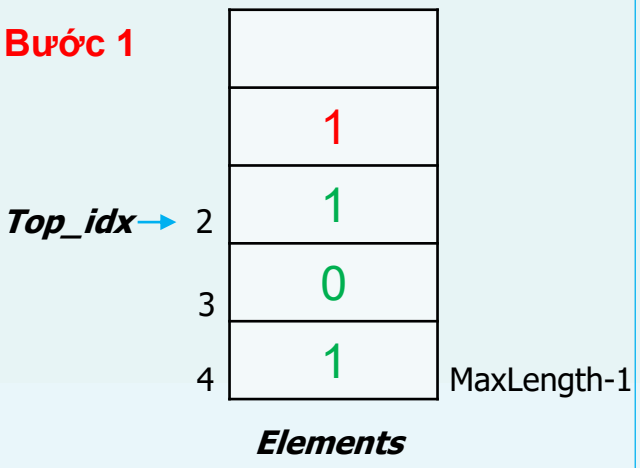

Bước 4
Top_idx → 1
1
2
1
3
0
4
1
MaxLength-1
Elements

BÀI TẬP

```
void print_binary(int n){
    Stack S;
    makenullStack(&S);
    while(n!=0){
        push(n%2,&S);
        n=n/2;
    }
    printf("So nhi phan tuong ung la:");
    while(!emptyStack(S)){
        printf("%d",top(S));
        pop(&S);
    }
}
```

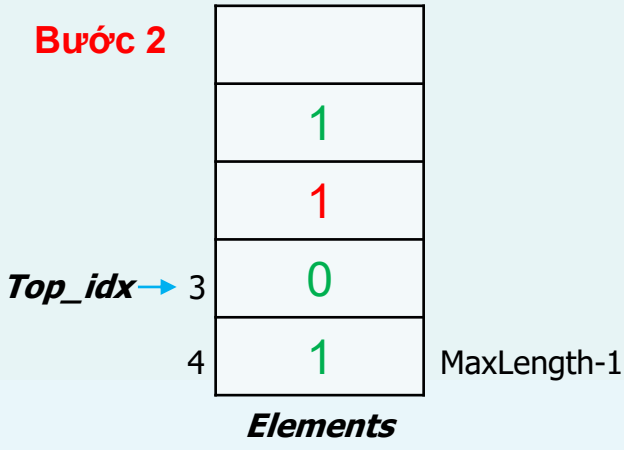


Bước 1



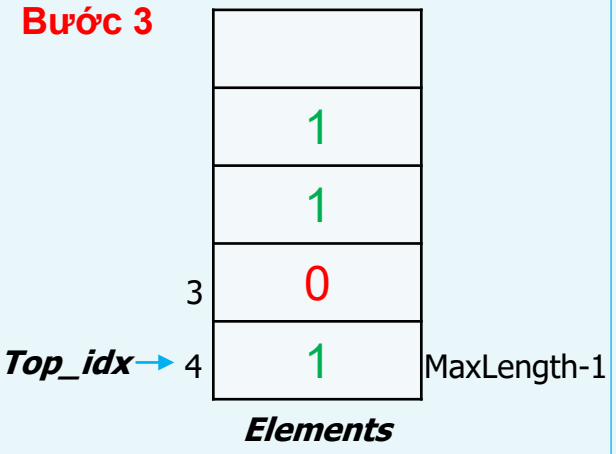
In p.tử trên đỉnh ngăn xếp;
Xóa p.tử trên đỉnh ngăn xếp

Bước 2



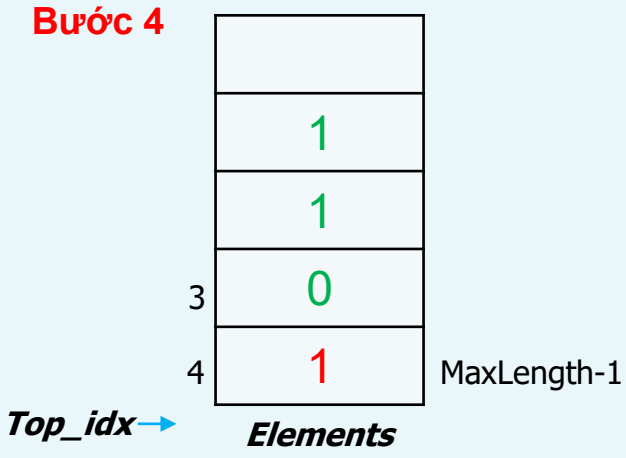
In p.tử trên đỉnh ngăn xếp;
Xóa p.tử trên đỉnh ngăn xếp

Bước 3



In p.tử trên đỉnh ngăn xếp;
Xóa p.tử trên đỉnh ngăn xếp

Bước 4



In p.tử trên đỉnh ngăn xếp;
Xóa p.tử trên đỉnh ngăn xếp

Kết quả in:
- Bước 1: 1
- Bước 2: 1
- Bước 3: 0
- Bước 4: 1

BÀI TẬP

```
void print_binary(int n){
    Stack S;
    makenullStack(&S);
    while(n!=0){
        push(n%2,&S);
        n=n/2;
    }
    printf("So nhi phan tuong ung la:");
    while(!emptyStack(S)){
        printf("%d",top(S));
        pop(&S);
    }
}
```

NGĂN XẾP (STACK)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT
  - CÀI ĐẶT BẰNG MẢNG
  - CÀI ĐẶT BẰNG DANH SÁCH

CÀI ĐẶT NGĂN XẾP BẰNG DANH SÁCH

- Khai báo

```
typedef List Stack;
```

- Tạo ngăn xếp rỗng

```
void makenullStack(Stack *pS) {
     makenullList(pS);
}
```

- Kiểm tra ngăn xếp rỗng

```
int emptyStack(Stack S) {
     return emptyList(S);
}
```

CÀI ĐẶT NGĂN XẾP BẰNG DANH SÁCH

- Trả về phần tử ở đỉnh ngăn xếp

```
ElementType top(Stack S) {
        retrieve(first(S),S);
}
```

- Xóa phần tử ra khỏi ngăn xếp

```
void pop(Stack *pS) {
        deleteList(first(*pS),pS);
}
```

- Thêm phần tử vào ngăn xếp

```
void push(Elementtype x, Stack *pS) {
        insertList(x,first(*pS),pS);
}
```

Q&A?