Stacks

Kuan-Yu Chen (陳冠宇)

2020/09/30 @ TR-212, NTUST

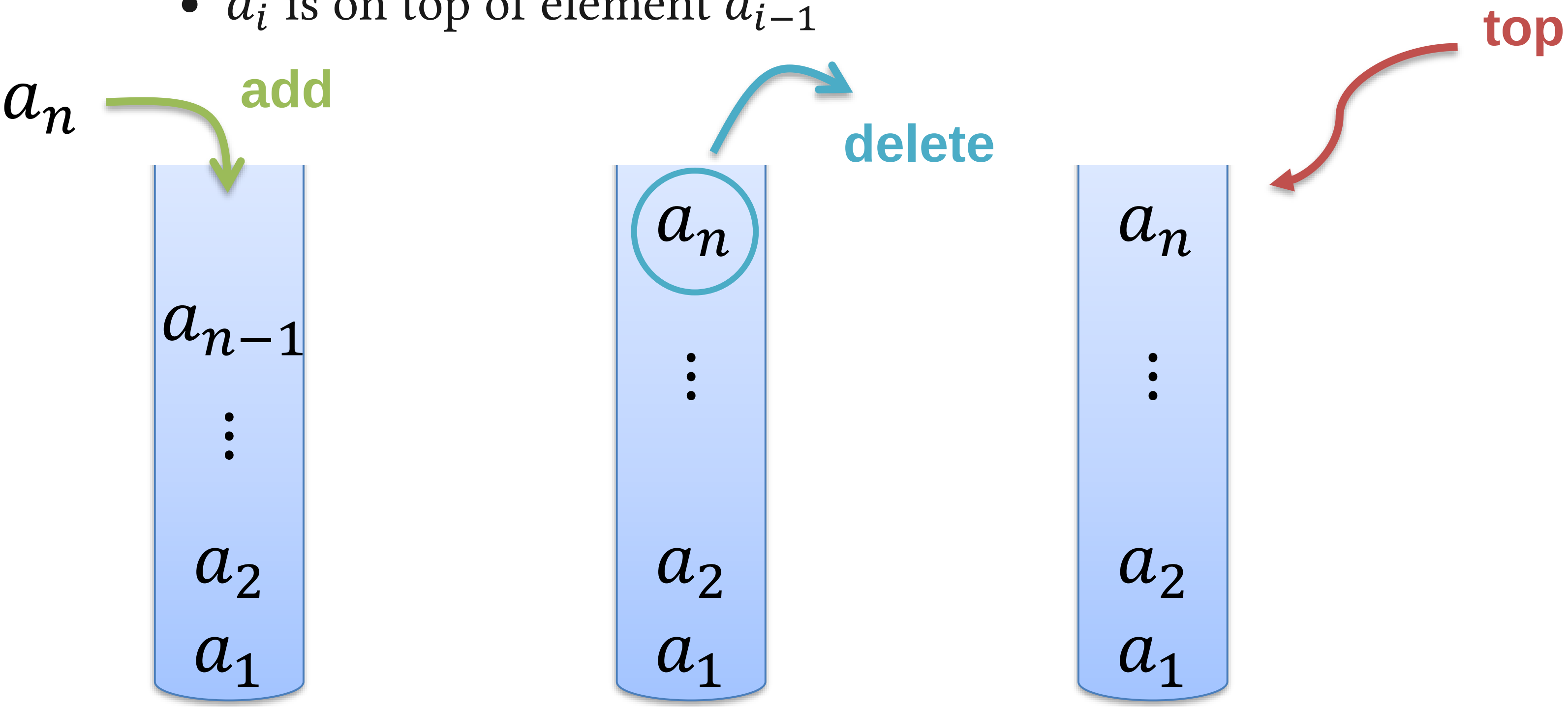
Review

- Array
  - An array is a set of pairs $< index, value >$, such that each index is associated with a value

- 2D Array = Matrix
  - Row-Major
  - Column-Major
  - Upper-Triangular
  - Lower-Triangular

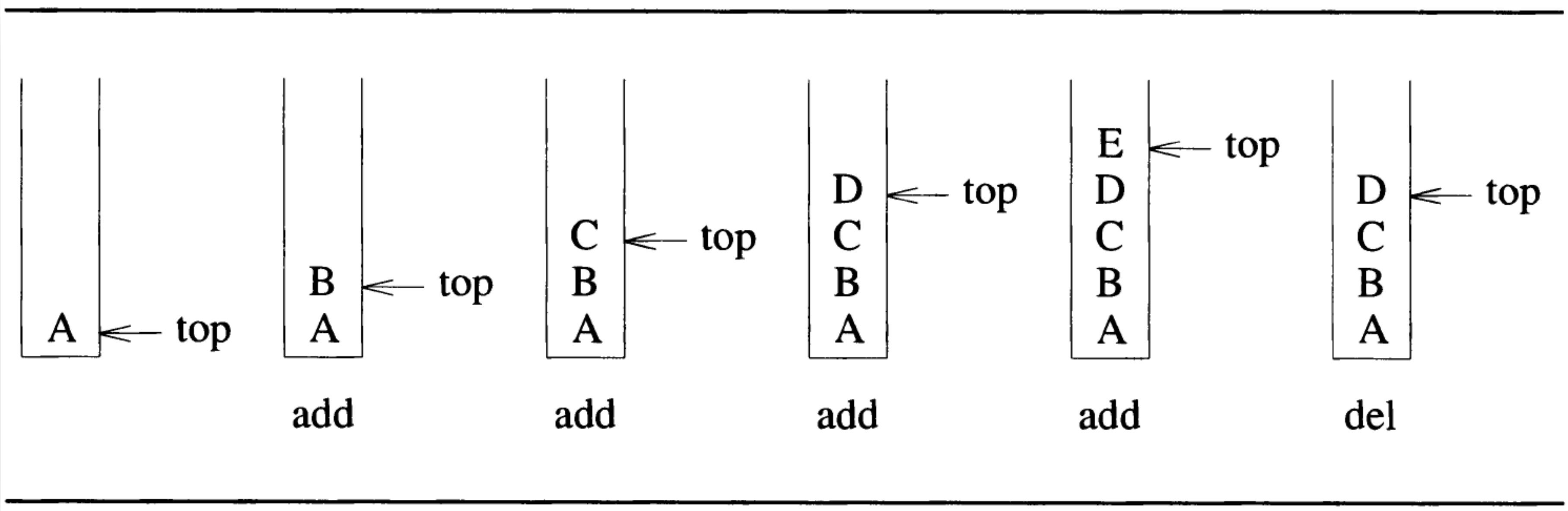
Stacks.

- A **stack** is an **ordered** list in which insertions and deletions are made at one end called the **top**
  - Given a stack $S = (a_1, a_2, \ldots, a_n)$
    - a_1 is the bottom element
    - a_n is the top element
    - a_i is on top of element a_{i-1}

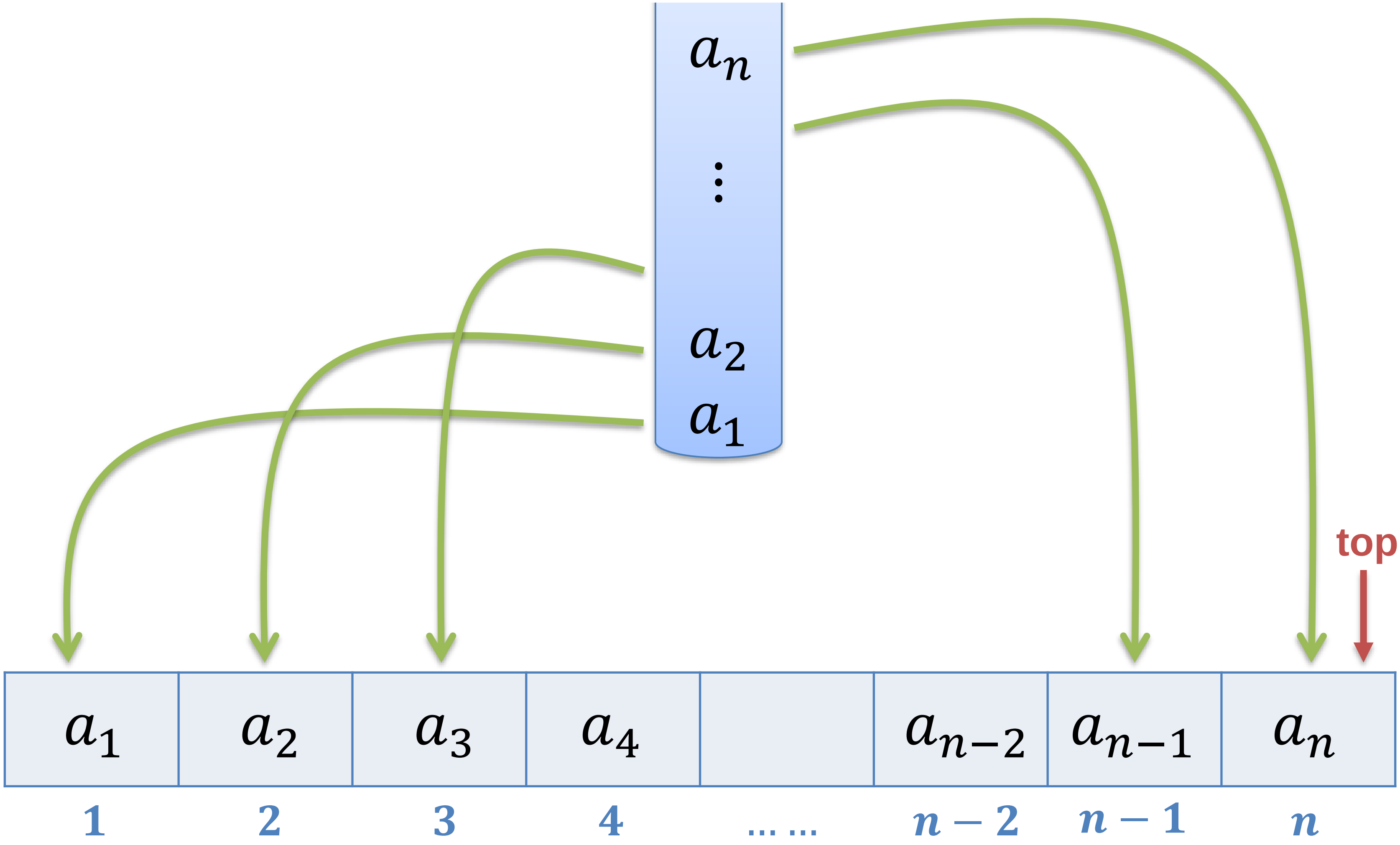


Stacks..

- By the definition of stack, if we add the elements A, B, C, D, E to the stack, in that order, then E is the first element we delete from the stack
  - **Last-In-First-Out**



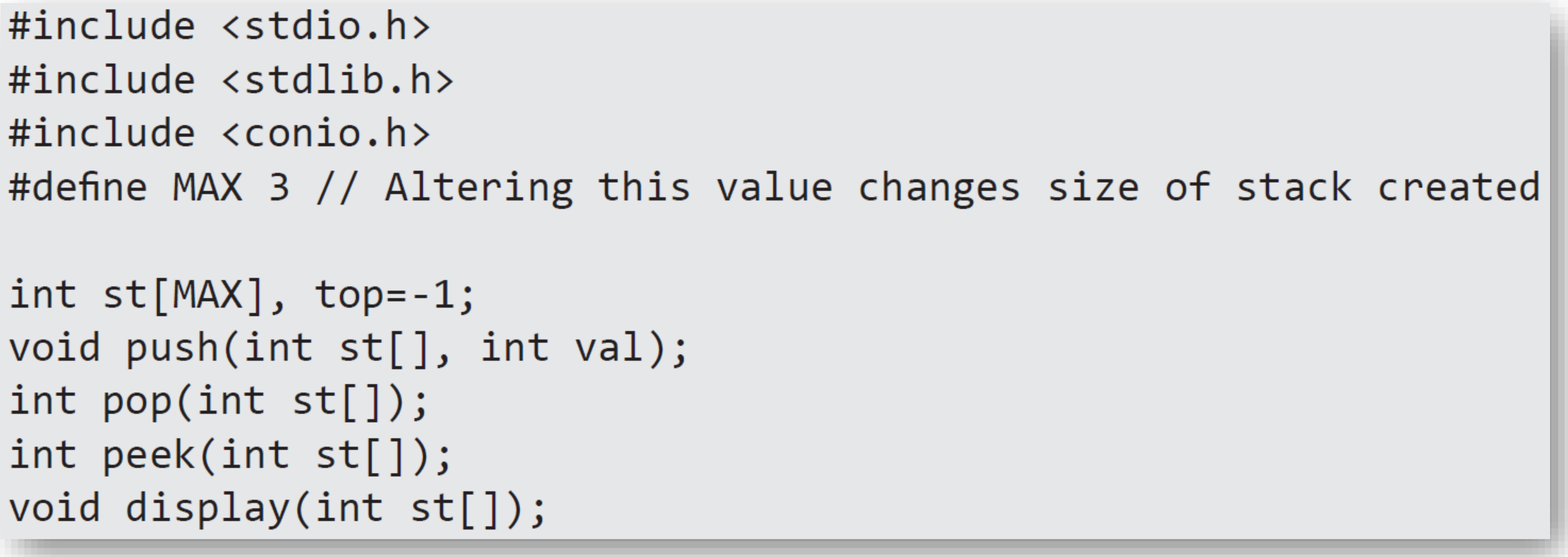
Leverage Array to Implement Stack



a_1	a_2	a_3	a_4		a_{n-2}	a_{n-1}	a_n
1	2	3	4	… …	$n-2$	$n-1$	n

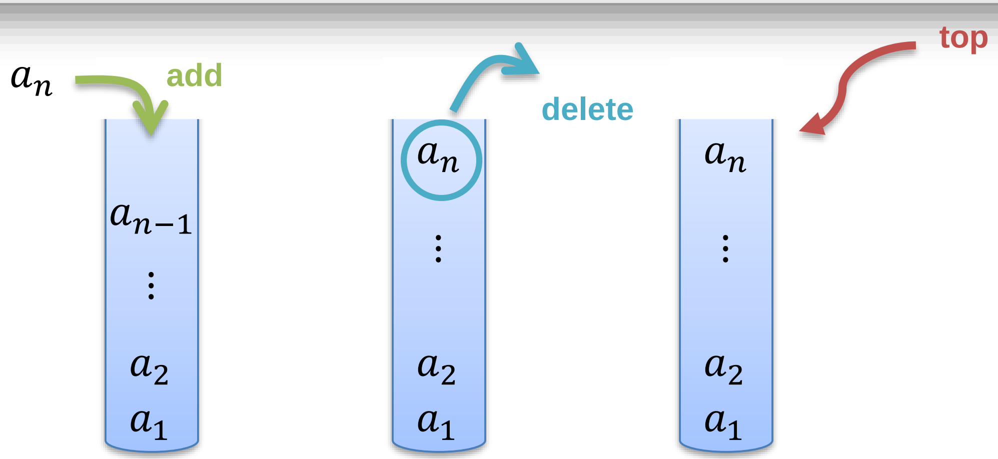
Implementation for Stack by Array.

- Declare



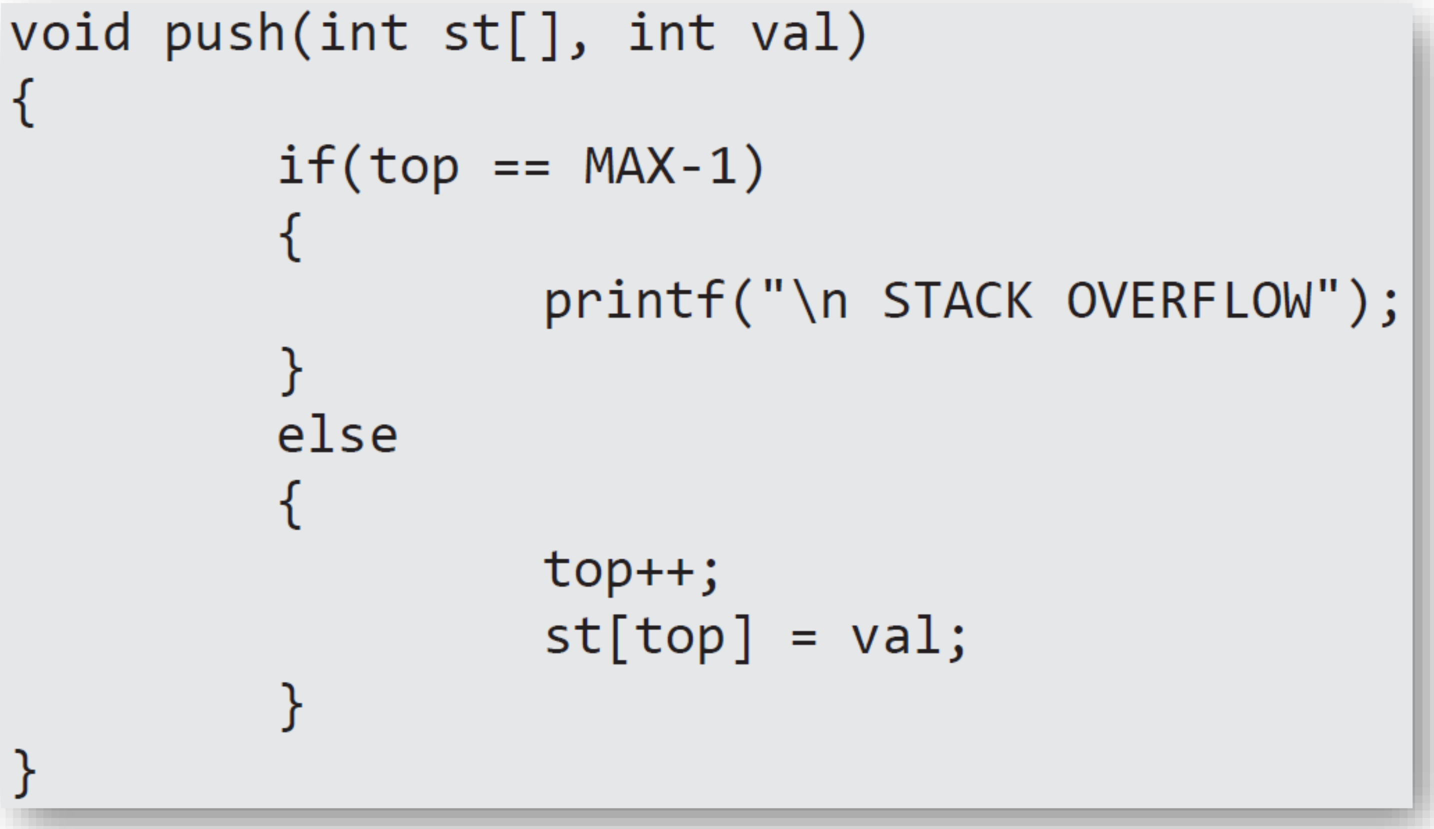
```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 3 // Altering this value changes size of stack created

int st[MAX], top=-1;
void push(int st[], int val);
int pop(int st[]);
int peek(int st[]);
void display(int st[]);
```



Implementation for Stack by Array..

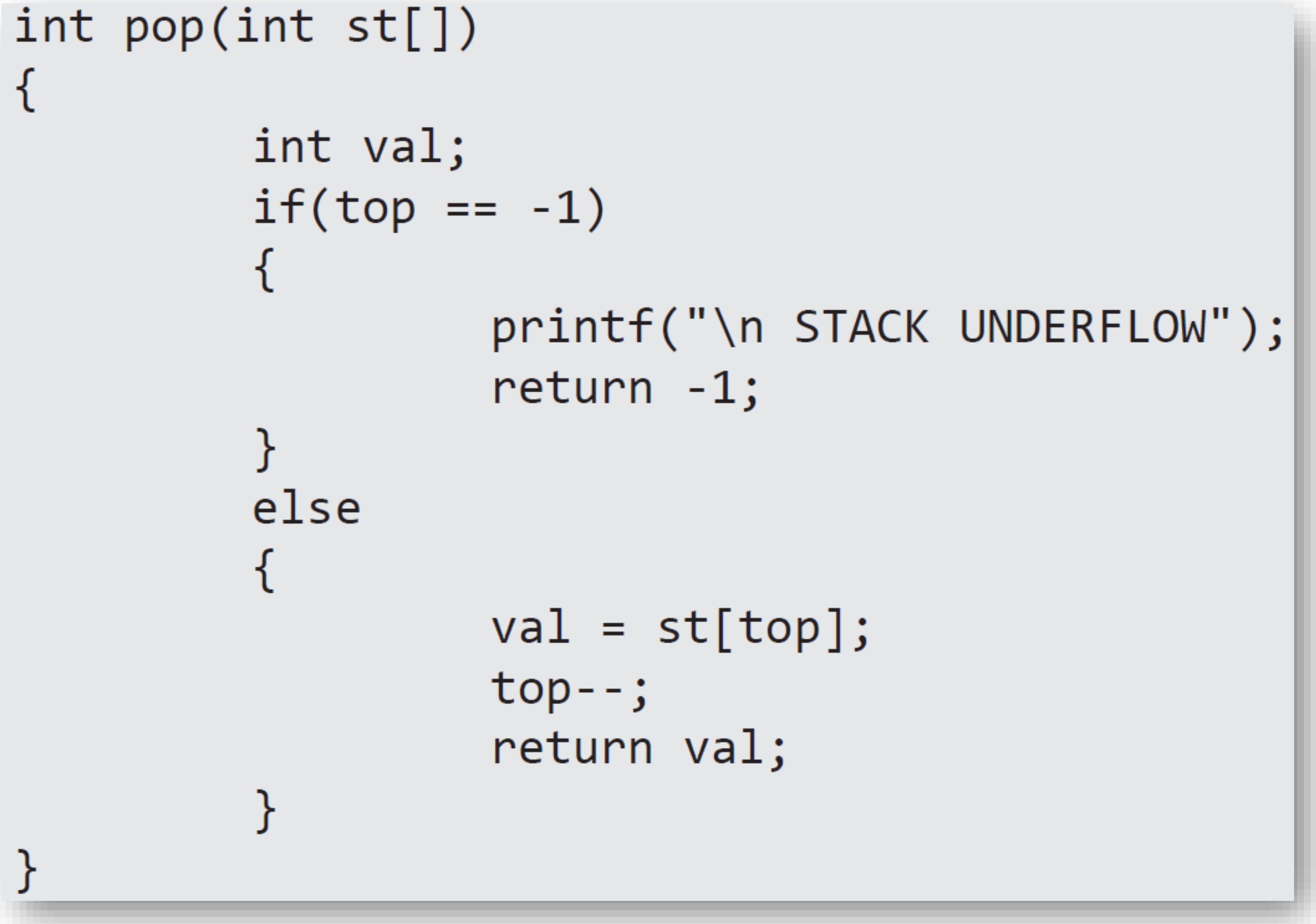
- For “push”



```
void push(int st[], int val)
{
        if(top == MAX-1)
        {
                printf("\n STACK OVERFLOW");
        }
        else
        {
                top++;
                st[top] = val;
        }
}
```

Implementation for Stack by Array…

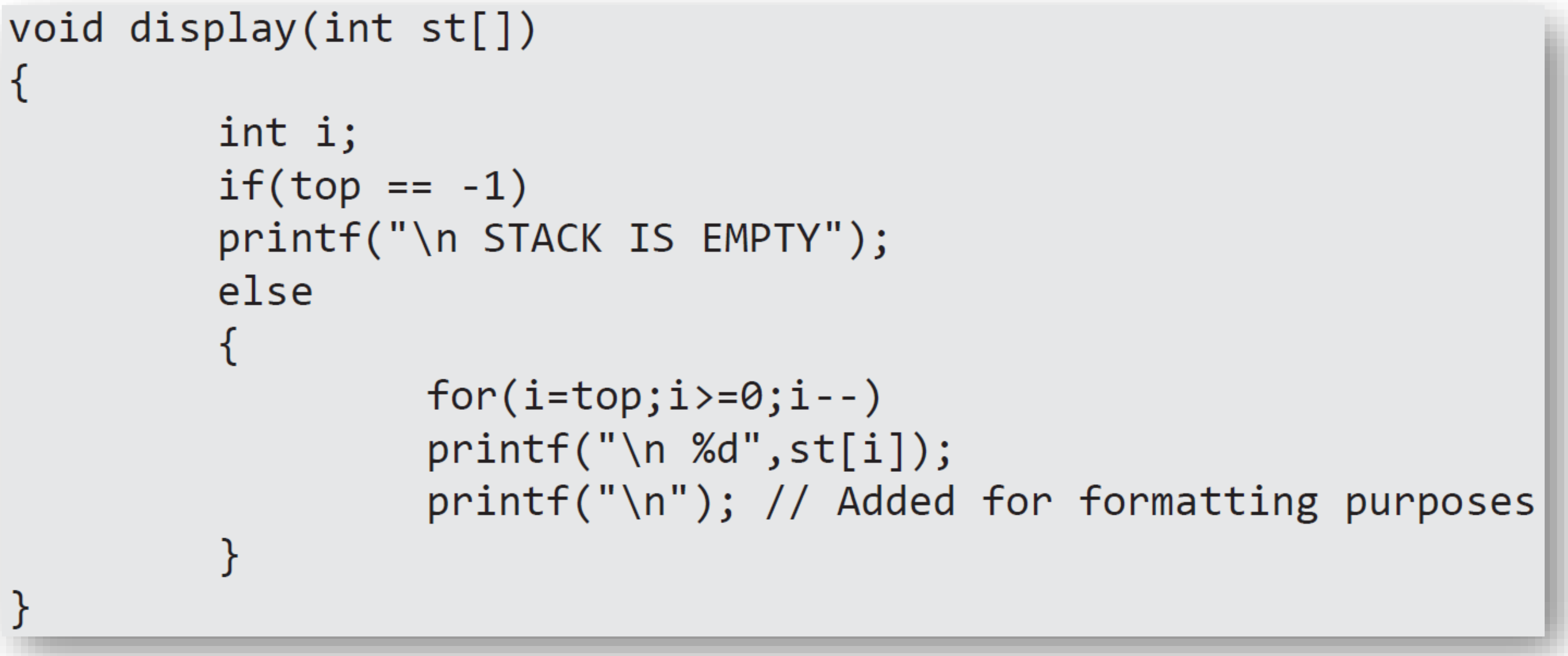
- For “pop”



```
int pop(int st[])
{
        int val;
        if(top == -1)
        {
                printf("\n STACK UNDERFLOW");
                return -1;
        }
        else
        {
                val = st[top];
                top--;
                return val;
        }
}
```

Implementation for Stack by Array….

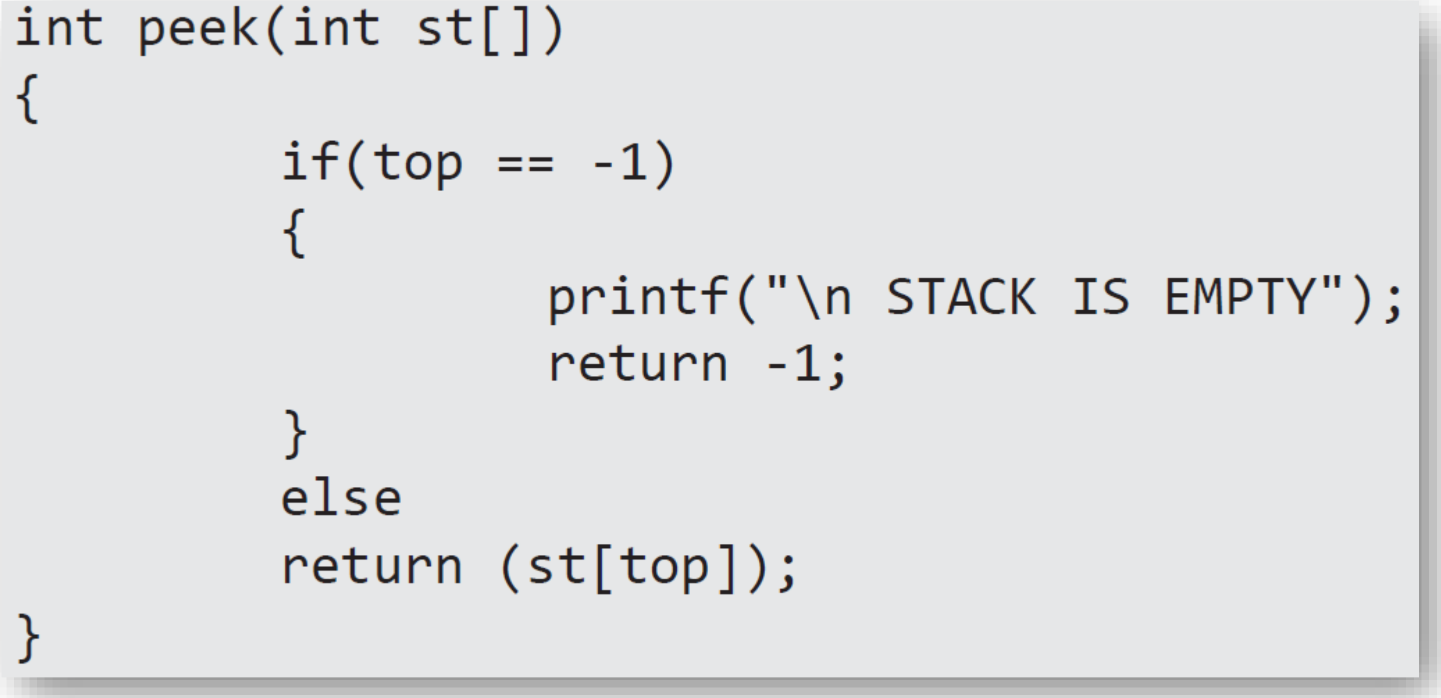
- For “display”



```
void display(int st[])
{
        int i;
        if(top == -1)
        printf("\n STACK IS EMPTY");
        else
        {
                for(i=top;i>=0;i--)
                printf("\n %d",st[i]);
                printf("\n"); // Added for formatting purposes
        }
}
```

Implementation for Stack by Array…..

- For “peek”



```
int peek(int st[])
{
        if(top == -1)
        {
                printf("\n STACK IS EMPTY");
                return -1;
        }
        else
        return (st[top]);
}
```

Stack Permutation.

- Given a sequence of elements and a empty stack, if a permutation can be generated by these elements and the stack, the permutation is called "stack permutation"
  - Stack-sortable permutation

- For a given sequence of elements $\{A, B, C\}$, please write down its stack permutation
  - ABC
    - push A, pop A, push B, pop B, push C, pop C
  - ACB
  - BAC
    - Push A, push B, pop B, pop A, push C, pop C
  - BCA
  - CBA
    - Push A, push B, push C, pop C, pop B, pop A

Stack Permutation..

- Given a sequence of n elements and a empty stack, the number of possible stack permutations can be calculated by
  - Catalan number
  - https://en.wikipedia.org/wiki/Catalan_number

$$\frac{1}{n+1}\mathrm{C}_n^{2n}$$

  - For a sequence of 3 elements, the number of possible stack permutations is

$$\frac{1}{n+1}\mathrm{C}_n^{2n} = \frac{1}{3+1}\mathrm{C}_3^{6} = \frac{1}{3+1}\frac{6\times 5\times 4}{3\times 2\times 1} = 5$$

Questions?

kychen@mail.ntust.edu.tw