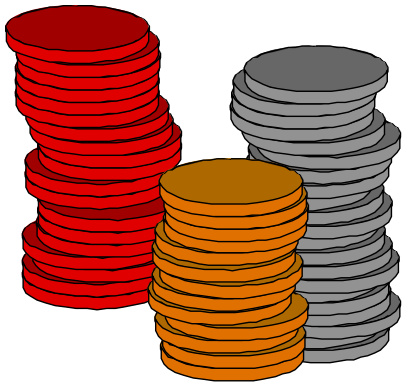
Stack

Stack

- Stack: what is it?
- Implementation
- Applications

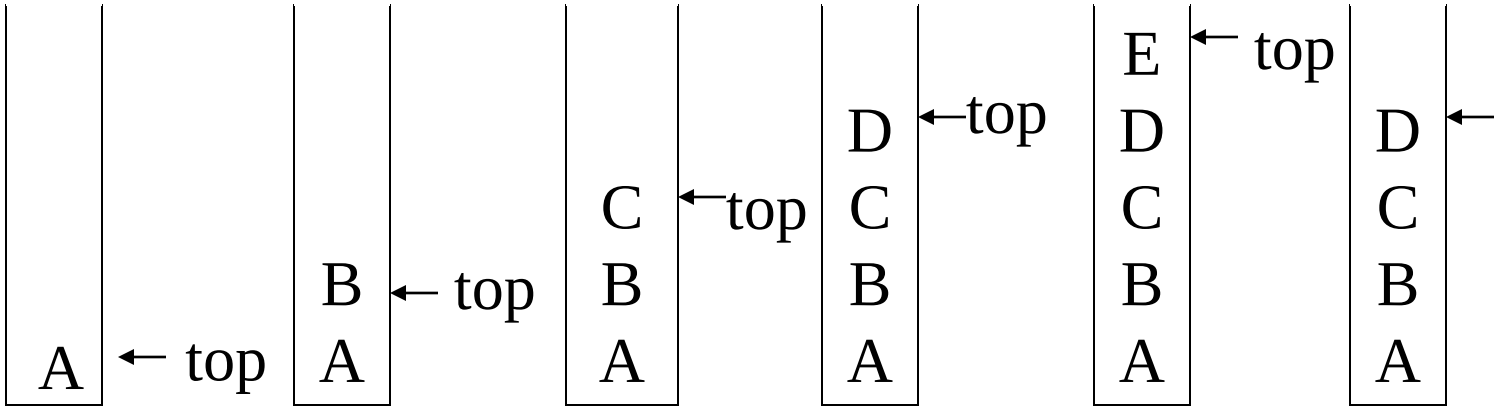
What is a stack?

- Stores a set of elements in a particular order
- Stack principle: LAST IN FIRST OUT
- = LIFO
- It means: the last element inserted is the first one to be removed
- Example



- Which is the first element to pick up?

Last In First Out



Stack Applications

- Real life
  - Pile of books
  - Plate trays
- More applications related to computer science
  - Program execution stack (read more from your text)
  - Evaluating expressions

Stack ADT

objects: *a finite ordered list with zero or more elements.*
methods:
for all stack $\in$ Stack, item $\in$ element, max_stack_size $\in$ *positive integer*
Stack *createS(*max_stack_size*) ::=*
create an empty stack whose maximum size is max_stack_size
Boolean *isFull(*stack, max_stack_size*) ::=*
if *(number of elements in* stack == max_stack_size*)*
return *TRUE*
else return *FALSE*
Stack *push(*stack, item*) ::=*
if *(IsFull(*stack*))* stack_full
else *insert* item *into top of* stack *and* ***return***

Stack ADT (cont'd)

Boolean *isEmpty(*stack*) ::=*
if(stack == *CreateS(*max_stack_size*))***
return *TRUE*
else return *FALSE*

Element *pop(*stack*) ::=*
if(*IsEmpty(*stack*))* ***return***
else *remove and return the* item *on the top of the stack.*

Array-based Stack Implementation

- Allocate an array of some size (pre-defined)
  - Maximum N elements in stack
- Bottom stack element stored at element 0
- last index in the array is the *top*
- Increment *top* when one element is pushed, decrement after pop

Stack Implementation: CreateS, isEmpty, isFull

Stack *createS(max_stack_size)* *::=*

```
#define MAX_STACK_SIZE 100 /* maximum stack size */
typedef struct {
        int key;
        /* other fields */
        } element;
element stack[MAX_STACK_SIZE];
int top = -1;
```

Boolean *isEmpty(Stack)* *::= top< 0;*

Boolean *isFull(Stack)* *::= top >= MAX_STACK_SIZE-1;*

Push

```
void push(int *top, element item)
{
 /* add an item to the global stack */
    if (*top >= MAX_STACK_SIZE-1) {
         stack_full( );
         return;
    }
    stack[++*top] = item;
}
```

Pop

```
element pop(int *top)
{
 /* return the top element from the stack */
    if (*top == -1)
        return stack_empty( );  /* returns and error key */
    return stack[(*top)--];
 }
```

C++ Stack

```
#include <stack>

stack<int>  st;
empty()     Test whether stack is empty;
size()      Return stack size;
top()       Access top element;
push()      Add element;
pop()       Remove element;
```

Java Stack

Stack()	Creates an Stack;
empty()	Test if empty;
peek()	Looks at the object at the top of this stack without removing it from the stack;
pop()	Removes the object at the top of this stack and returns that object;
push()	Pushes an item onto the top of this stack;
search()	Returns the 1-based position where an object is on this stack.