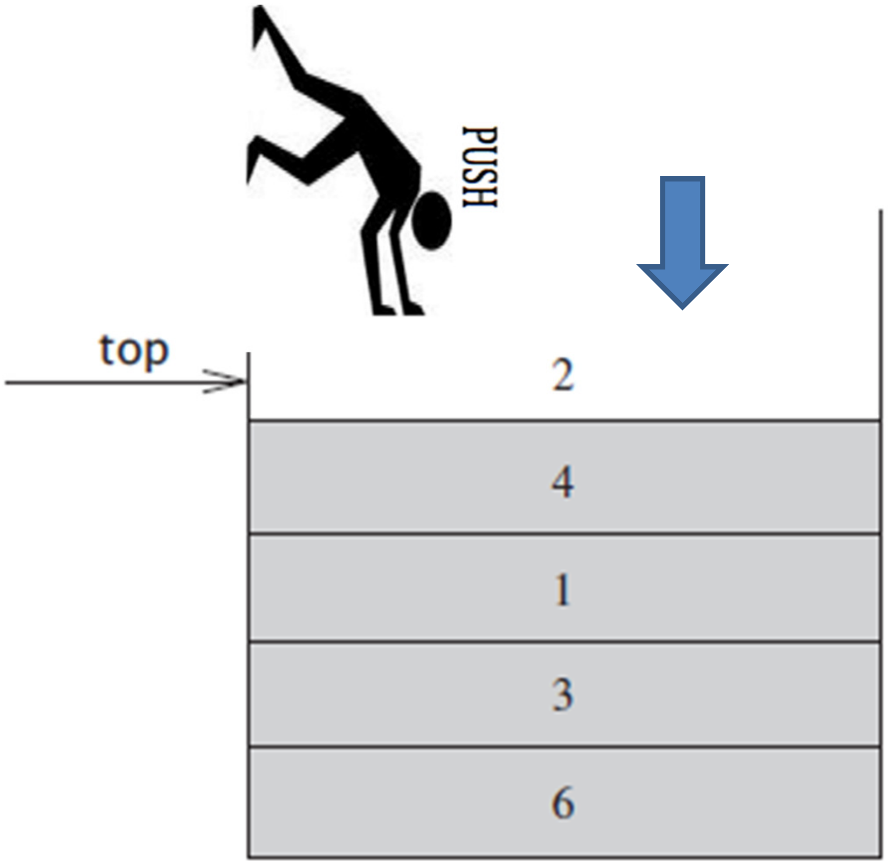
Stack

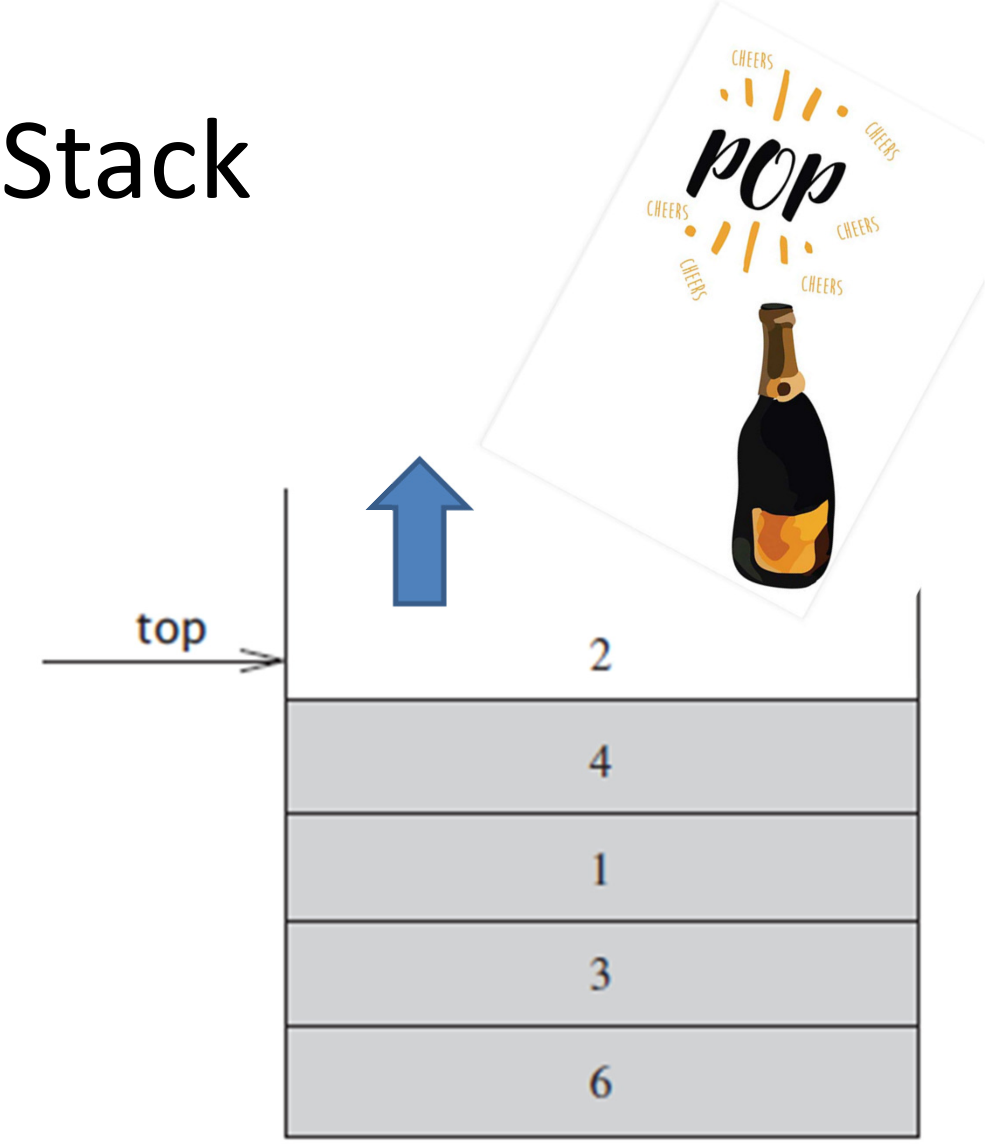
Stack

- A stack is a pile of objects (same type) which can be accessed only from the top.
- Operations
  - ***Push***: add an item to the *top* of the stack.
  - ***Pop***: remove an item from the *top* of the stack.
- It is also known as LIFO (last in, first out)

The Stack



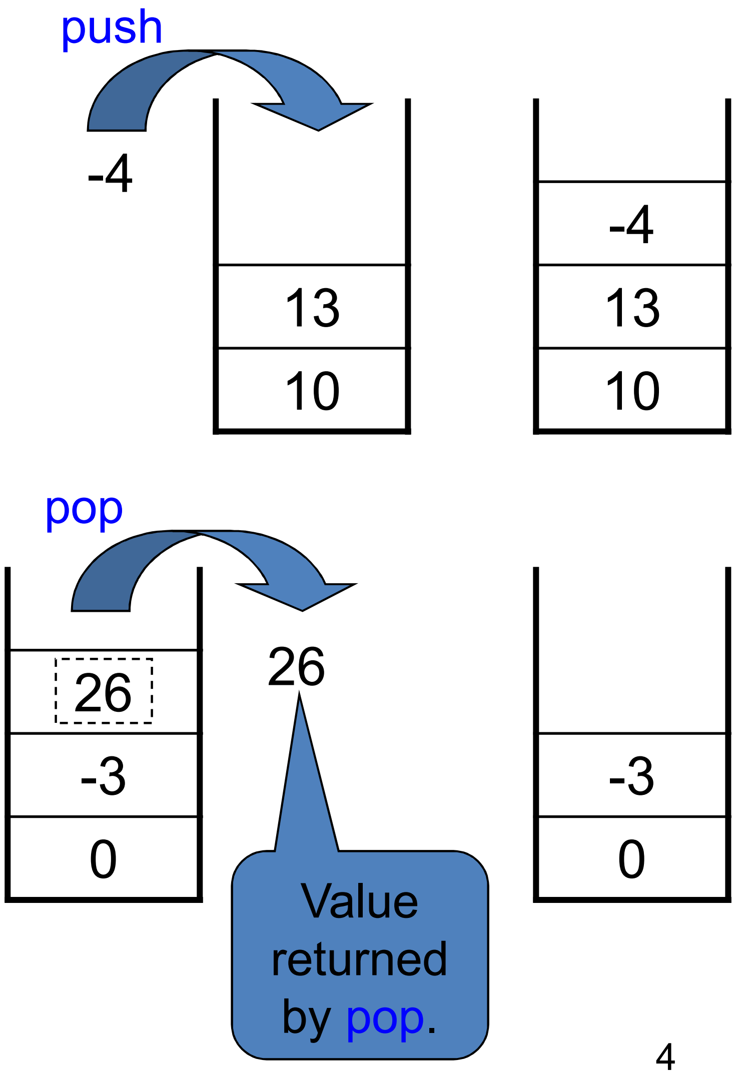
PUSH



POP

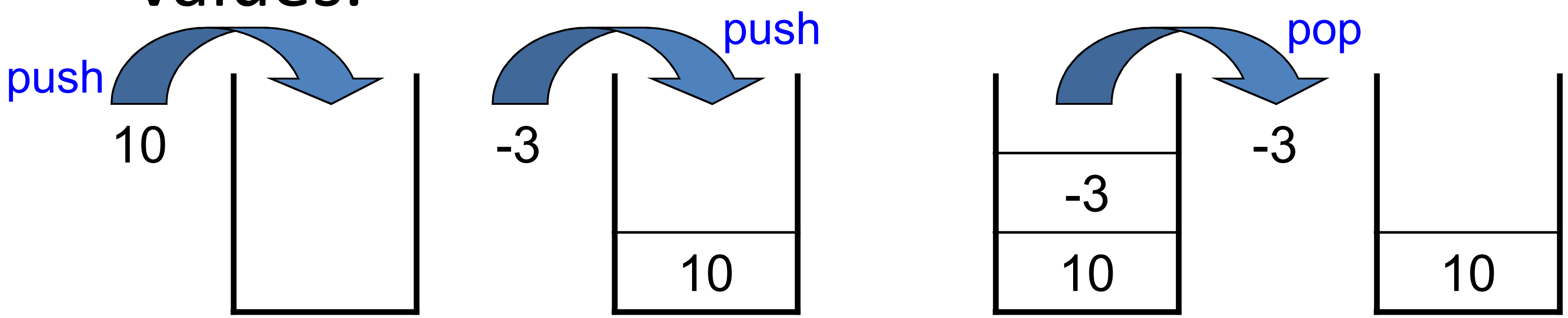
Stack Operations: Push and Pop

- Push
  - Puts a value on the top of a stack.
- Pop
  - Takes a value from the top of a stack.
  - The value is *returned*.
- A stack is a ***Last-In-First-Out*** (LIFO) data structure.

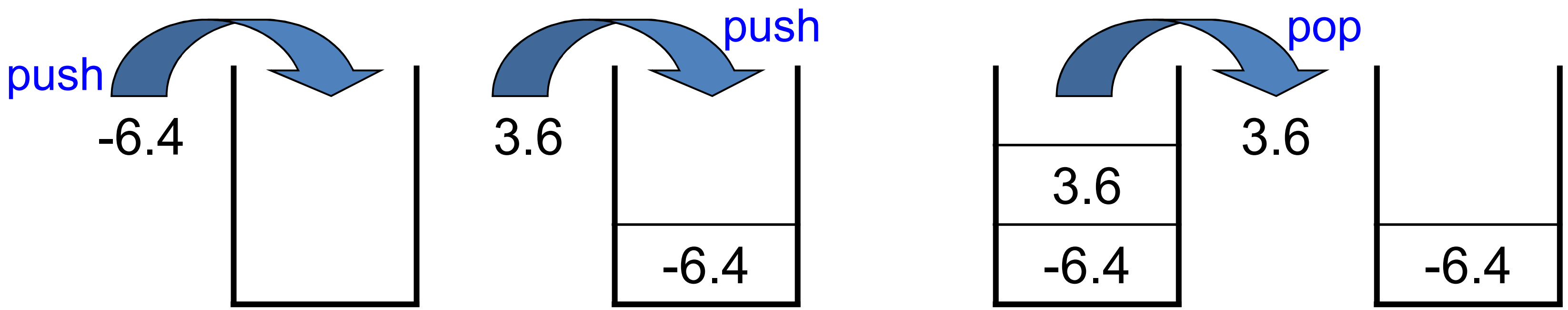


Stack

- As a data structure, a stack is used to store values.

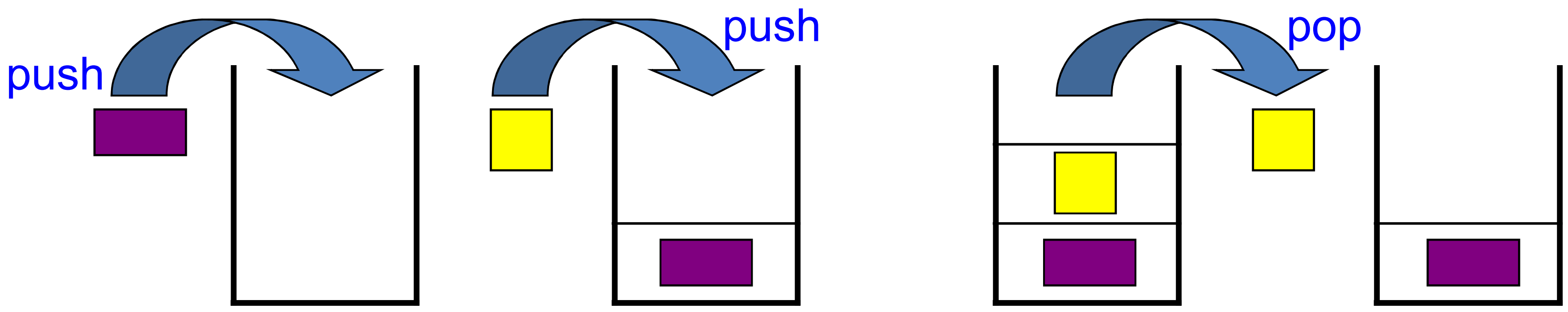


Stack of integers

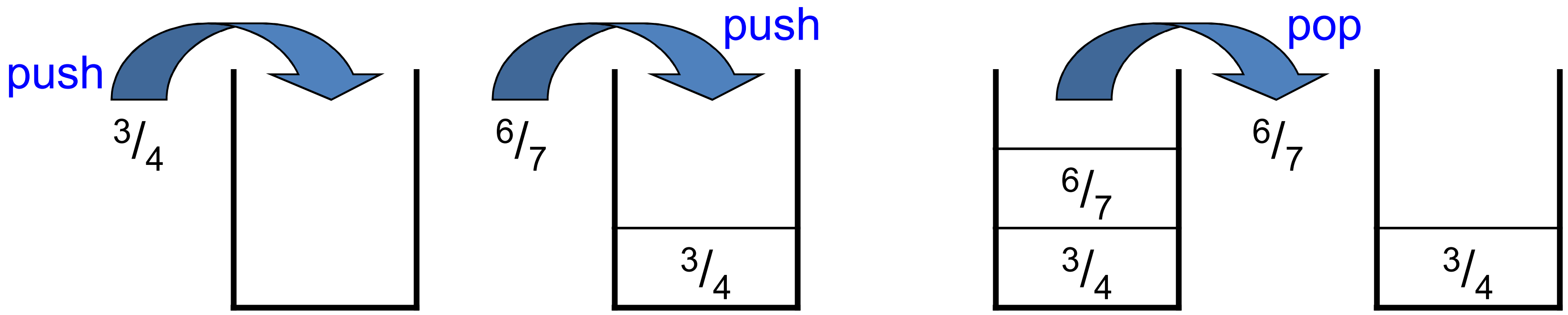


Stack of real numbers

Stack

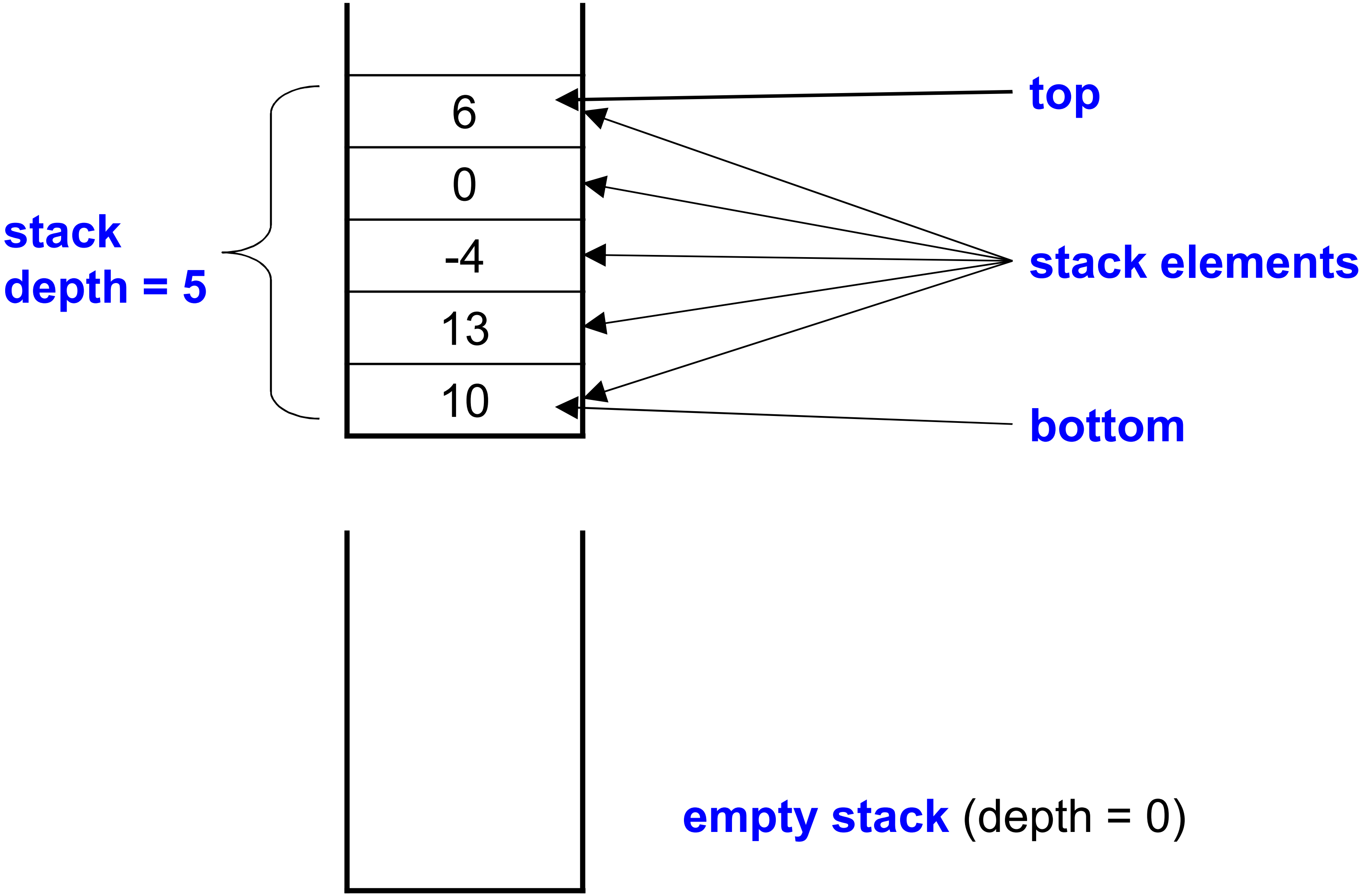


Stack of rectangles

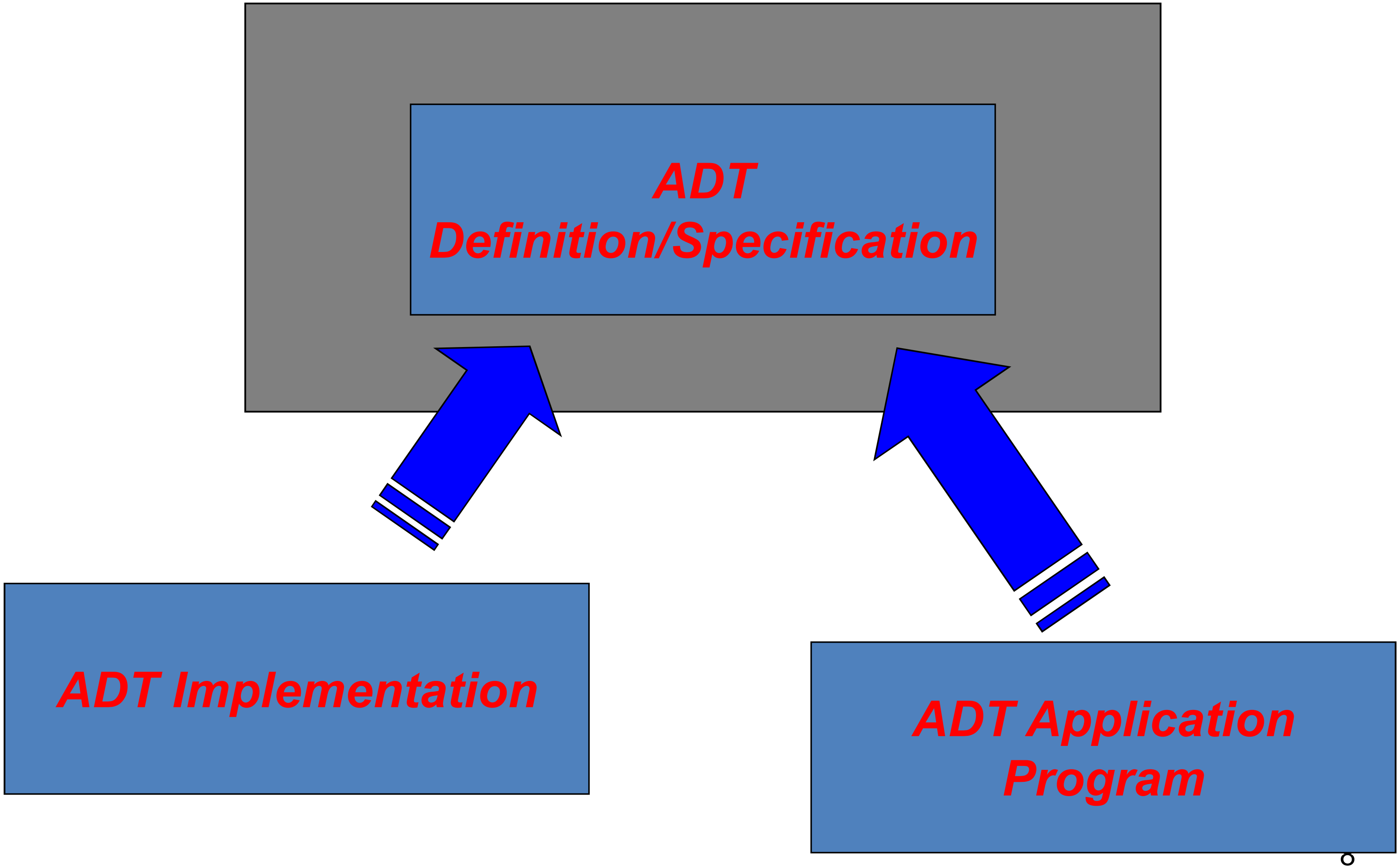


Stack of rational numbers

Stack Terminologies



Stack as an ADT



ADT Definition/Specification

ADT Implementation

ADT Application Program

Defining a Stack ADT: stack.h

```
typedef struct stackCDT *stackADT;

typedef int stackElementT;

stackADT EmptyStack();
void Push(stackADT stack, stackElementT element);
stackElementT Pop(stackADT stack);
int StackDepth(stackADT stack);
int StackIsEmpty(stackADT stack);
```

- This version of stack.h defines stackElementT to be int.
- Changing int to double defines stacks of doubles.

Defining an ADT in an Interface

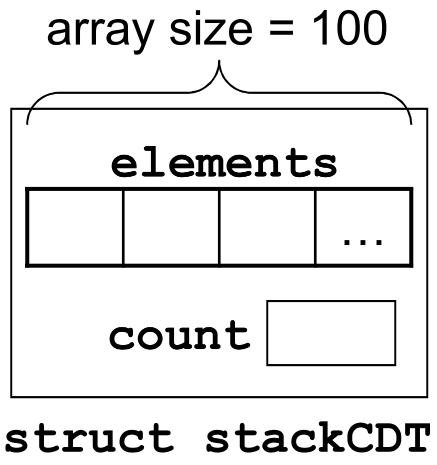
- To define a type in an interface so that its underlying representation remains hidden from users, we write

```
typedef struct nameCDT *nameADT;
```

  - struct *name*CDT is the name of the concrete type.

An example of *name*CDT :

```
struct stackCDT {
   stackElementT elements[100];
   int count;
};
```



Defining an ADT in an Interface

- To define a type in an interface so that its underlying representation remains hidden from users, we write

```
typedef struct nameCDT *nameADT;
```

  - struct *name*CDT is the name of the concrete type.
  - *name*ADT is the name of the abstract type
    - The type *name*ADT is a pointer to a structure identified by the name struct *name*CDT, which is hidden from users

Defining a Stack ADT

```
stackADT EmptyStack();
```

- Creates and returns a new empty stack to the user.

```
void Push(stackADT stack,
          stackElementT element);
```

- Adds the element `element` to the top of the stack `stack`. Nothing is returned.

Defining a Stack ADT

```
stackElementT Pop(stackADT stack);
```

- Removes an element from the top of `stack` and returns the element.

```
int StackDepth(stackADT stack);
```

- Returns the depth of `stack`.

```
int StackIsEmpty(stackADT stack);
```

- Returns 1 if `stack` is empty; 0 otherwise.