

# *Topic 3*

## *Basic Data Structures*

CSE1303 Part A

Data Structures and Algorithms

# *Basic Data Structures*



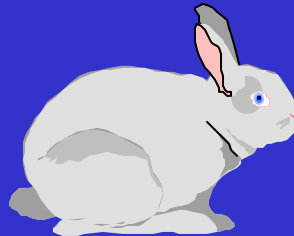
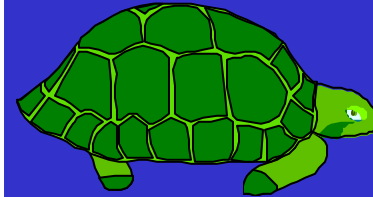
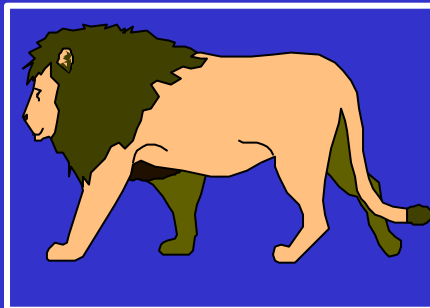
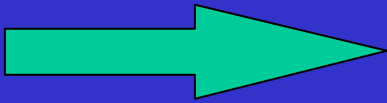
- Stacks
- Queues
- Lists

# *Overview of Stacks*

- What is a Stack?
- Operations on a Stack
  - push, pop
  - initialize
  - status: empty, full
- Implementation of a Stack.
- Example: Reversing a sequence.

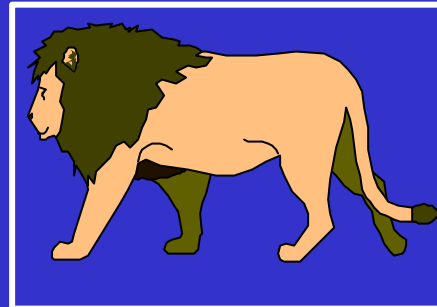
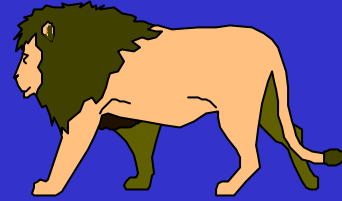
# *A Stack*

Top

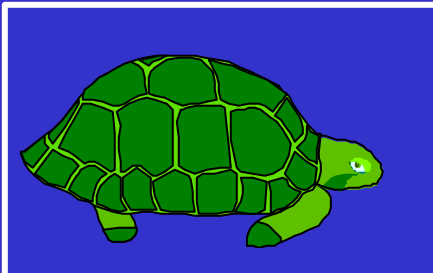


Items on  
the Stack

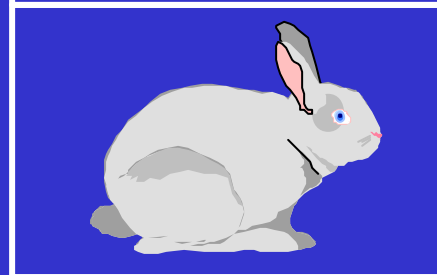
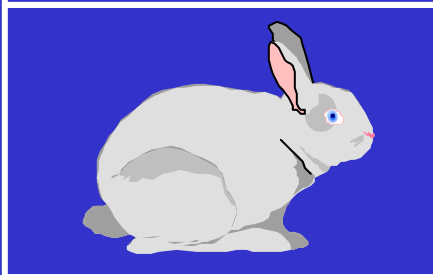
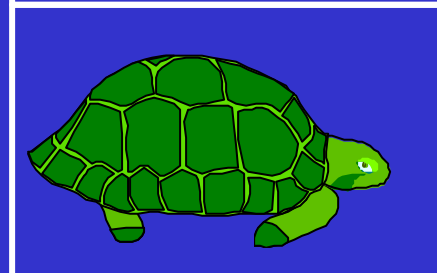
*Push*



← Top



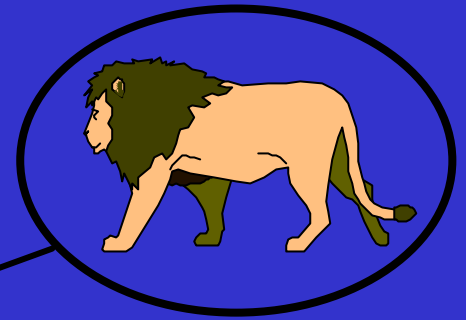
← Top



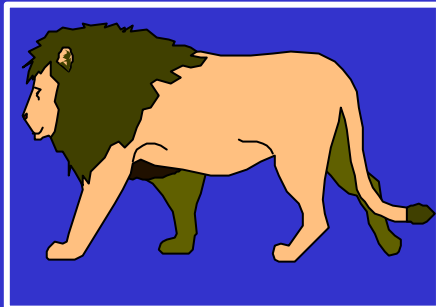
Before

After

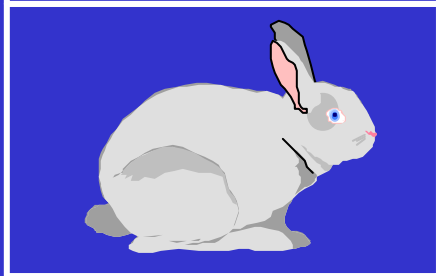
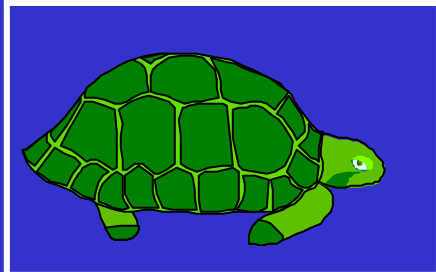
*Pop*



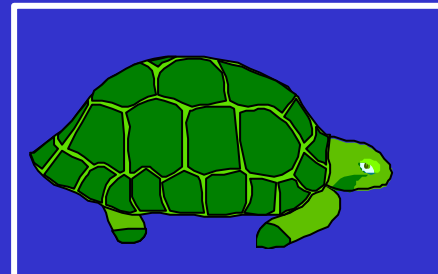
This comes off the stack



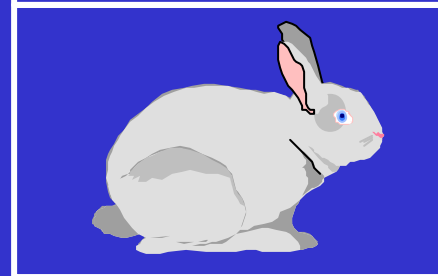
← Top



Before



← Top



After

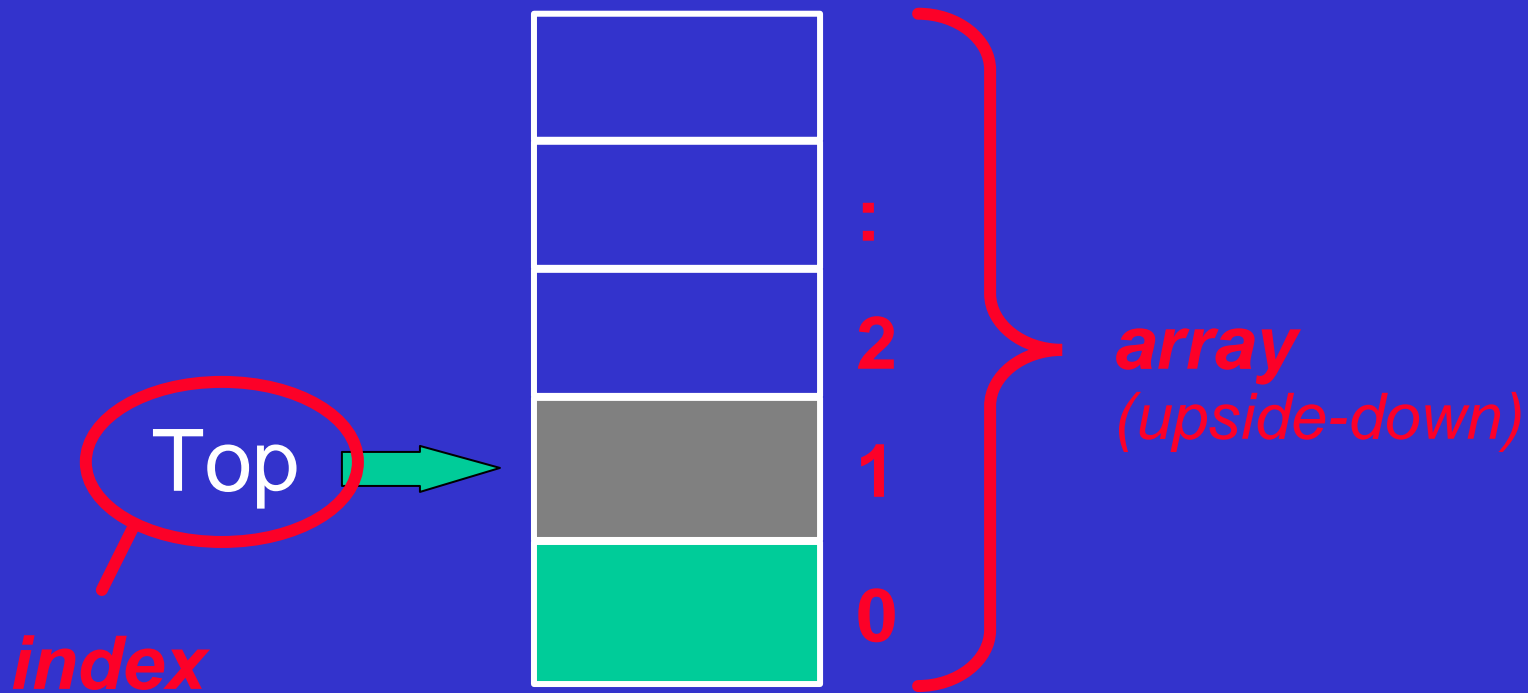
# *Operations*

- Initialize the Stack.
- Pop an item off the top of the stack.
- Push an item onto the top of the stack.
- Is the Stack empty?
- Is the Stack full?
- Clear the Stack
- Determine Stack Size

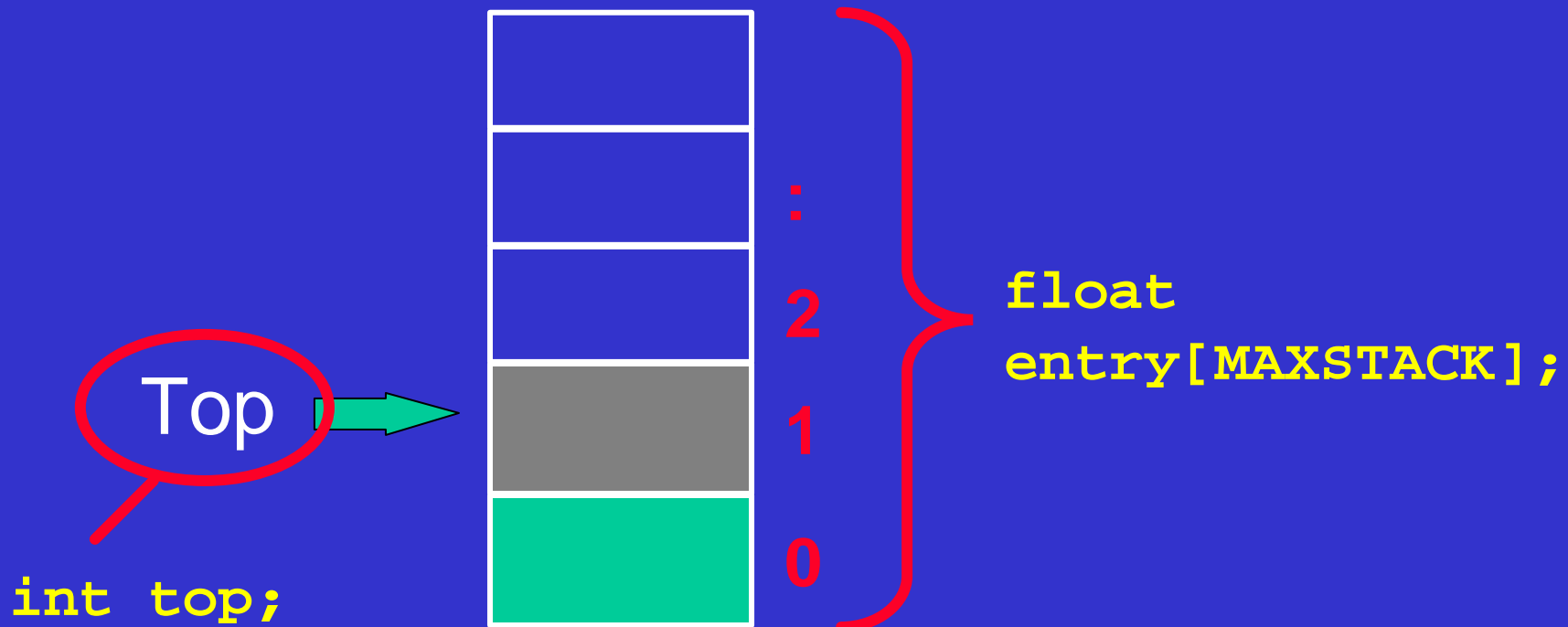
# *Stack Properties*

- Sequence of items, where insertions and deletions are done at the top.
- Main operations are pop and push.
- Last-In First Out (LIFO).
- Used when calling functions.
- Used when implementing recursion.

# *Implementation*



# *Implementation*



```
#ifndef STACK.H
#define STACK.H
#include <stdbool.h>
#define MAXSTACK 20

struct StackRec
{
    int    top;
    float  entry[MAXSTACK];
};

typedef struct StackRec Stack;

void initializeStack(Stack* stackPtr);
bool stackEmpty(const Stack* stackPtr);
bool stackFull(const Stack* stackPtr);
void push(Stack* stackPtr, float item);
float pop(Stack* stackPtr);

#endif
```

```
#define MAXSTACK 20

struct StackRec
{
    int top;
    float entry[MAXSTACK];
};
```

```
typedef struct StackRec Stack;
```

Stack:



```
#include <stdio.h>
#include <stdlib.h>
#include "stack.h"
```

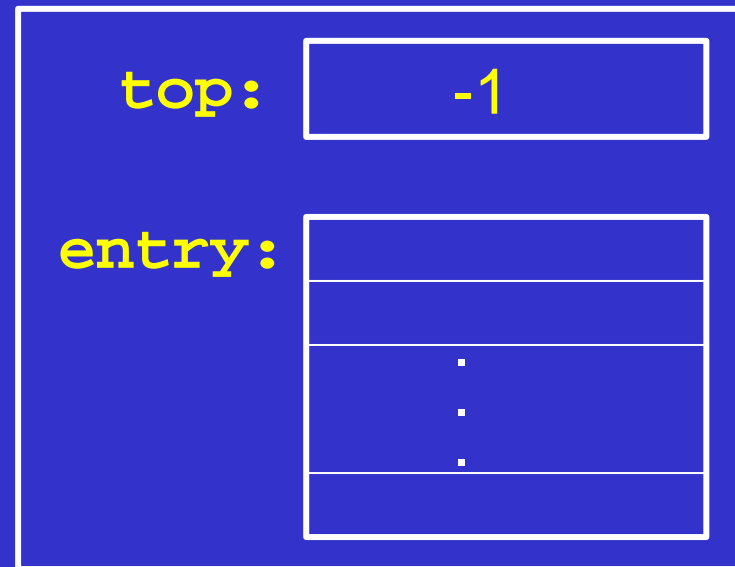
```
void initializeStack(Stack* stackPtr)
{
    stackPtr -> top = -1;
}
```

**stackPtr:**

*addr of Stack*



**Stack:**



```
bool stackEmpty(const Stack* stackPtr)
{
    if (stackPtr-> top < 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

```
bool stackFull(const Stack* stackPtr)
{
    if (stackPtr -> top >= MAXSTACK-1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

```
void push(Stack* stackPtr, float item)
{
    if (stackFull(stackPtr))
    {
        fprintf(stderr, "Stack is full\n");
        exit(1);
    }
    else
    {
        stackPtr-> top++;
        stackPtr-> entry[stackPtr-> top] = item;
    }
}
```

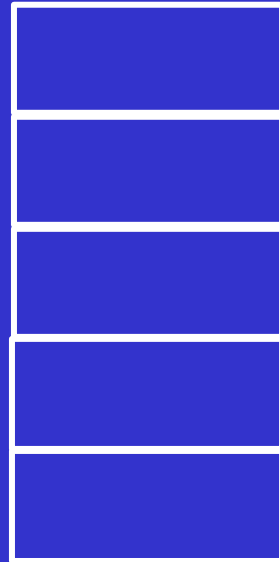
```
float pop(Stack* stackPtr)
{
    float item;

    if (stackEmpty(stackPtr))
    {
        fprintf(stderr, "Stack is empty\n");
        exit(1);
    }
    else
    {
        item = stackPtr-> entry[stackPtr-> top];
        stackPtr-> top--;
    }

    return item;
}
```

# *Reversing a Sequence*

Take a sequence: **-1.5 2.3 6.7**

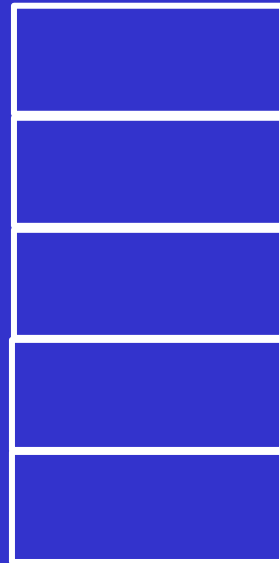


Push onto a stack one number at a time.

Then pop each number off the stack

# *Reversing a Sequence*

-1.5 2.3 6.7

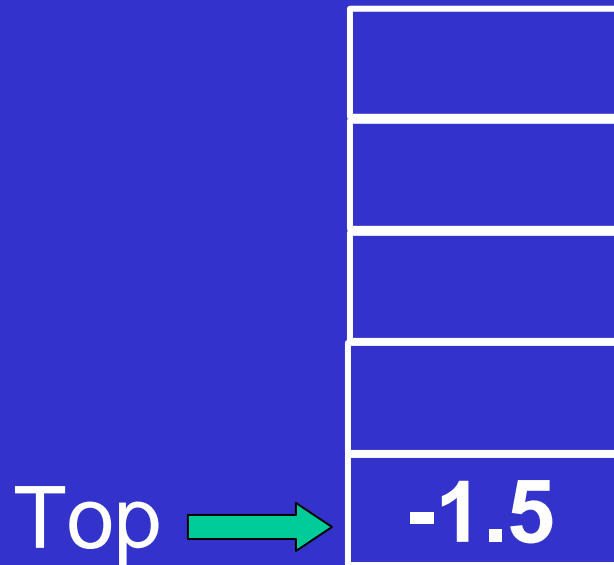


Push onto a stack one number at a time.

Top →

# *Reversing a Sequence*

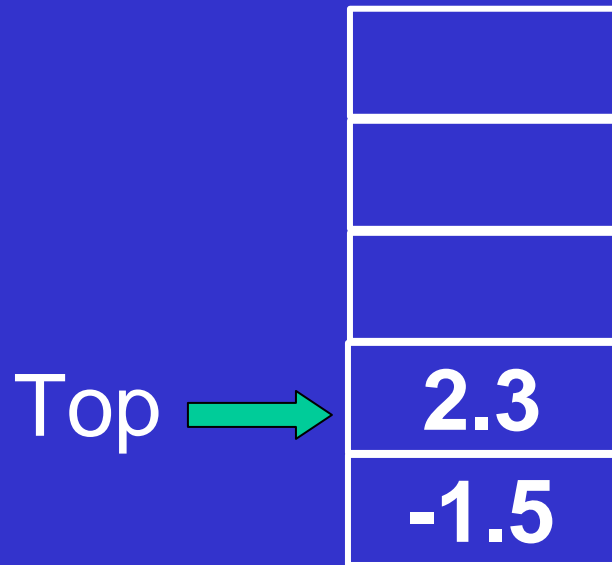
2.3 6.7



Push onto a stack one number at a time.

# *Reversing a Sequence*

6.7



Push onto a stack one number at a time.

# *Reversing a Sequence*



Push onto a stack one number at a time.

# *Reversing a Sequence*



Then pop each number off the stack

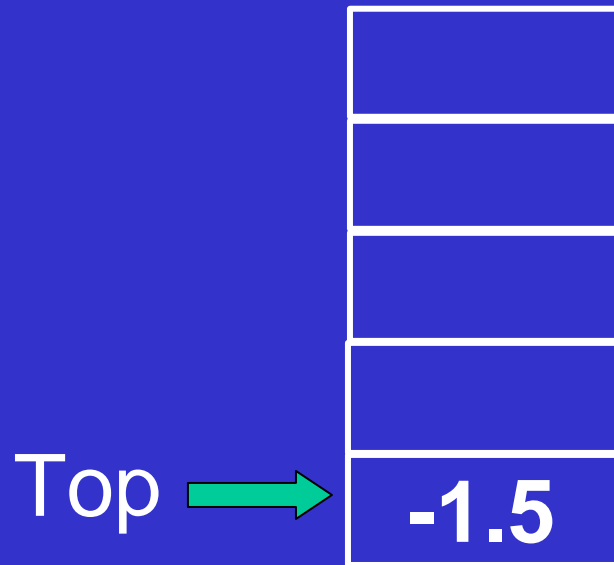
# *Reversing a Sequence*



Then pop each number off the stack

**6.7**

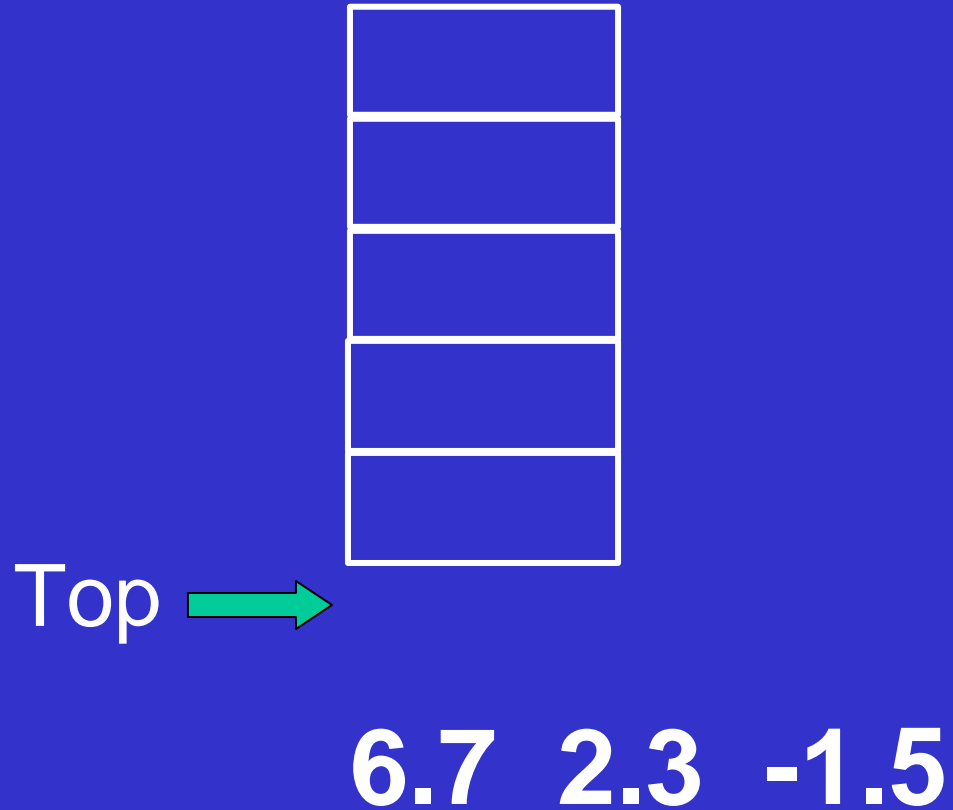
# *Reversing a Sequence*



Then pop each number off the stack

**6.7 2.3**

# *Reversing a Sequence*



```
module reverse ()
```

```
{
```

```
  initialize the Stack
```

```
  loop{
```

```
    input nextNumber
```

```
    if (end of input) then {exit loop}
```

```
    if (Stack is not full) then {push nextNumber onto Stack}
```

```
  }
```

```
  loop {
```

```
    if (Stack is empty) then {exit loop}
```

```
    pop nextNumber off the Stack
```

```
    output nextNumber
```

```
  }
```

```
}
```

```
#include <stdio.h>
#include "stack.h"
int main()
{
    Stack theStack;
    float next;

    initializeStack(&theStack);
    printf("Enter number sequence: ");
    while (scanf("%f", &next) != EOF) {
        if (!stackFull(&theStack)) {
            push(&theStack, next);
        }
    }
    while (!stackEmpty(&theStack)) {
        next = pop(&theStack);
        printf("%f", next);
    }
    printf("\n");
}
```

# *Revision*

- Stack
- Operations on a Stack
  - push, pop
  - initialize
  - status: empty, full
  - others: clear, size
- Example: Reversing a sequence.
- Implementation.

## *Next Lecture*

- Queue
- Main Operations
- Implementation

## *Preparation*

- Read 4.1 to 4.2 in Kruse et al.
- Deitel & Deitel (2e) 12.6