2. The Stack ADT

The Stack ADT

- A stack is a block of memory that allows operations on the data structure to be done on one end of the memory called the top of stack (TOS)
- Also known as LIFO (last in, first out) structure

The Stack ADT

- A stack is a block of memory that allows operations on the data structure to be done on one end of the memory called the top of stack (TOS)
- Also known as LIFO (last in, first out) structure
- Two operations:
  - push (insert)
  - pop (delete)

The Stack ADT

- push
  - operation for storing values on top of stack
  - updates the stack pointer to indicate that the stack has grown
- pop
  - takes out one value from the top of stack
  - updates the top of stack pointer to indicate that the stack has shrink by one value

The Stack ADT

Possible Errors

- Stack Underflow
  - attempt to pop a value from an empty stack
- Stack Overflow
  - attempt to push a value into a full stack

The Stack ADT

Implementation

- Array
- Linked list

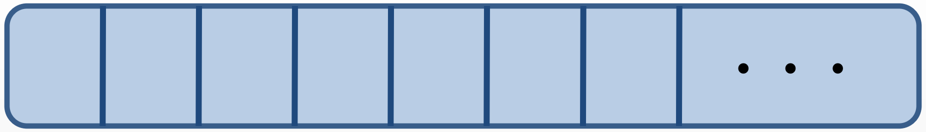
2. The Stack ADT

2.1 Array Implementation

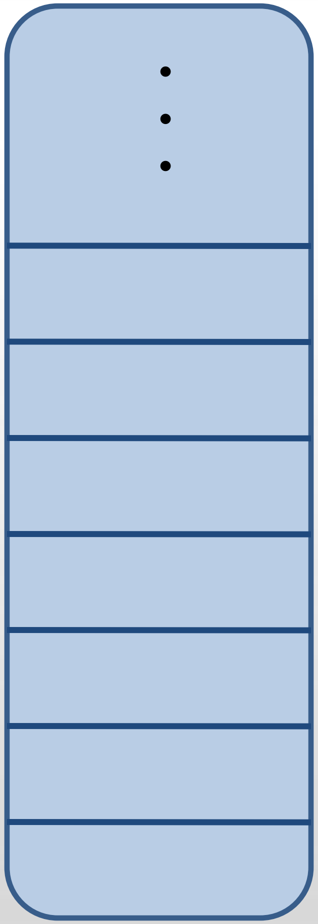
Stack – Array Implementation

- Unlike ordinary array where every element can be accessed, the array used to represent a stack is accessible only at one end
- A top of stack variable is defined to point to that part of the memory that is accessible

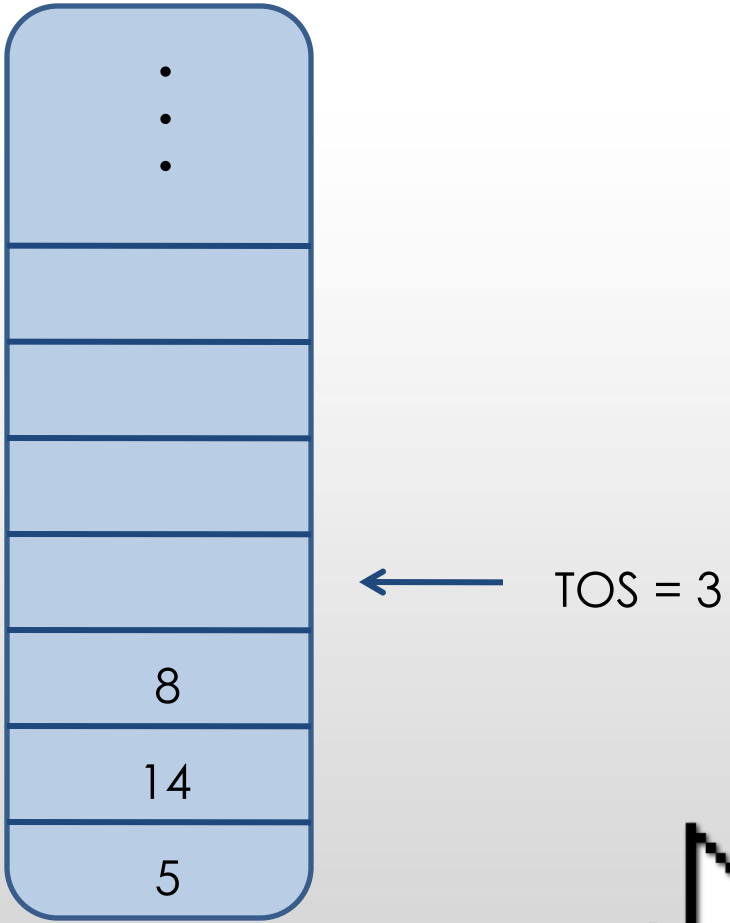
The Stack ADT



The Stack ADT



The Stack ADT



⋮

8

14

5

TOS = 3

Array Implementation

```
#define LIMIT 1000
int stack[LIMIT];
int top=0;

void push(int x){
  if (top < LIMIT)
    stack[top++]=x;
  else {
    printf(“stack overflow”);
    exit(1);
  }
}
```

```
int pop(){
  if (_______)
    _____________________
  else {
    printf(“stack underflow”);
    exit(1);
  }
}
```

Array Implementation

```
#define LIMIT 1000
int stack[LIMIT];
int top=0;

void push(int x){
  if (top < LIMIT)
    stack[top++]=x;
  else {
    printf(“stack overflow”);
    exit(1);
  }
}
```

```
int pop(){
  if (top > 0 )
    _____________________________
  else {
    printf(“stack underflow”);
    exit(1);
  }
}
```

Array Implementation

```
#define LIMIT 1000
int stack[LIMIT];
int top=0;

void push(int x){
  if (top < LIMIT)
    stack[top++]=x;
  else {
    printf(“stack overflow”);
    exit(1);
  }
}
```

```
int pop(){
  if (top > 0 )
    return(stack[--top]);
  else {
    printf(“stack underflow”);
    exit(1);
  }
}
```

Activity

```
#define LIMIT 1000
int stack[LIMIT];
int top=0;

void push(int x){
  if (top < LIMIT)
    stack[top++]=x;
  else {
    printf(“stack overflow”);
    exit(1);
  }
}
```

```
int pop(){
  if (top > 0 )
    return(stack[--top]);
  else {
    printf(“stack underflow”);
    exit(1);
  }
}
```

push(3); push(4); x=pop(); push(24); x=pop(); y=pop();