

# Data Structures

# Stack – Array Implementation

Team Emertxe



# Stack – Array Implementation



# Operations

Create Stack

Insert an Element

Delete an Element

Print top Element

Stack – `create_stack(stack,size)`

A large, stylized arrow pointing to the right, composed of two overlapping shapes. The front shape is a dark purple chevron, and the back shape is a lighter purple chevron, creating a 3D effect. The arrow is positioned at the bottom of the slide, pointing towards the right edge.

# create\_stack(stack,size)

```
typedef struct
{
    unsigned int capacity;
    int top;
    int *item;
} stack_t;
```

# create\_stack(stack,size)



```
typedef struct
```

```
{
```

```
    unsigned int capacity;
```

```
    int top;
```

```
    int *item;
```

```
}
```

```
stack_t;
```

Max Stack size

# create\_stack(stack,size)



```
typedef struct  
{  
    unsigned int capacity;  
    int top;  
    int *item;  
} stack_t;
```

Max Stack size  
Top variable

# create\_stack(stack,size)



```
typedef struct
{
    unsigned int capacity;
    int top;
    int *item;
} stack_t;
```

Max Stack size

Top variable

Array holding the stack elements



# create\_stack(stack,size)



## Input Specification:

stack : Pointer that contains address of structure variable (stack\_t)

size : Size of array

## Output Specification:

Status : e\_true / e\_false

# create\_stack(stack,size)



size = 4

```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

```
    return e_false
```

```
stack —→ capacity = size
```

```
stack —→ top = -1
```

```
return e_true
```

# create\_stack(stack,size)



size = 4

```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

```
    return e_false
```

```
stack —→ capacity = size
```

```
stack —→ top = -1
```

```
return e_true
```

item



# create\_stack(stack,size)



```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

```
    return e_false
```

```
stack —→ capacity = size
```

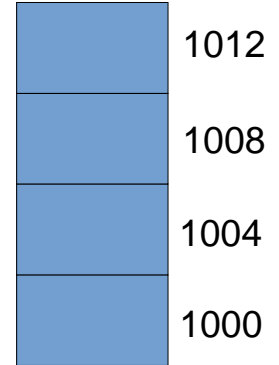
```
stack —→ top = -1
```

```
return e_true
```

size = 4

item

1000



# create\_stack(stack,size)

```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

```
    return e_false
```

```
stack —→ capacity = size
```

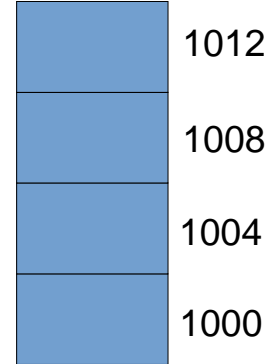
```
stack —→ top = -1
```

```
return e_true
```

size = 4

item

1000



# create\_stack(stack,size)

```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

```
    return e_false
```

```
stack —→ capacity = size
```

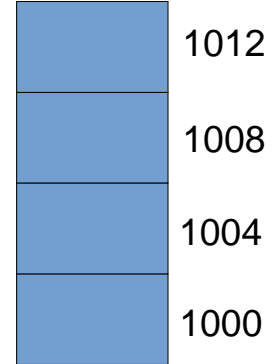
```
stack —→ top = -1
```

```
return e_true
```

size = 4

item

1000



# create\_stack(stack,size)



```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

```
    return e_false
```

```
stack —→ capacity = size
```

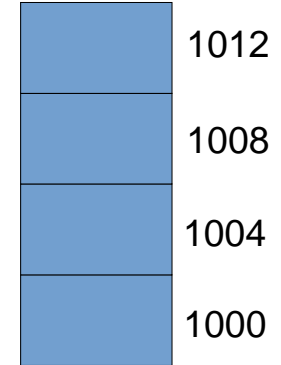
```
stack —→ top = -1
```

```
return e_true
```

size = 4

item

1000



# create\_stack(stack,size)



```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

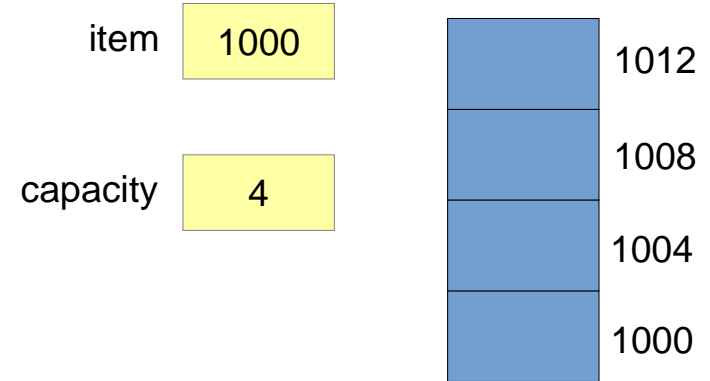
```
    return e_false
```

```
stack —→ capacity = size
```

```
stack —→ top = -1
```

```
return e_true
```

size = 4





# create\_stack(stack,size)



```
stack —→ item = Memalloc(sizeof(int) * size)
```

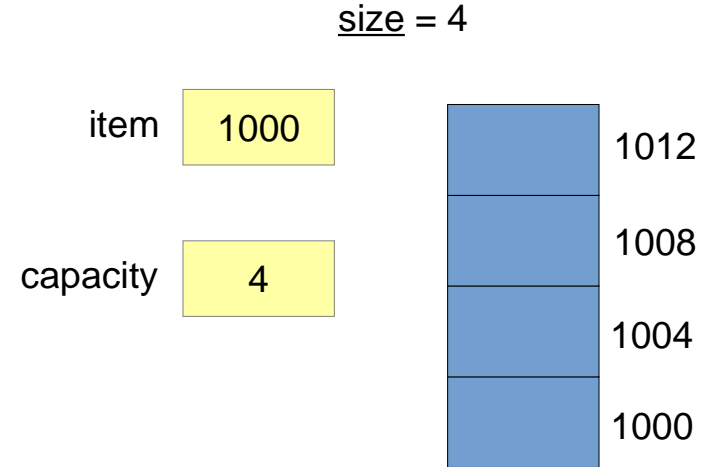
```
If (stack —→ item = NULL)
```

```
    return e_false
```

```
stack —→ capacity = size
```

```
stack —→ top = -1
```

```
return e_true
```



# create\_stack(stack,size)



```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

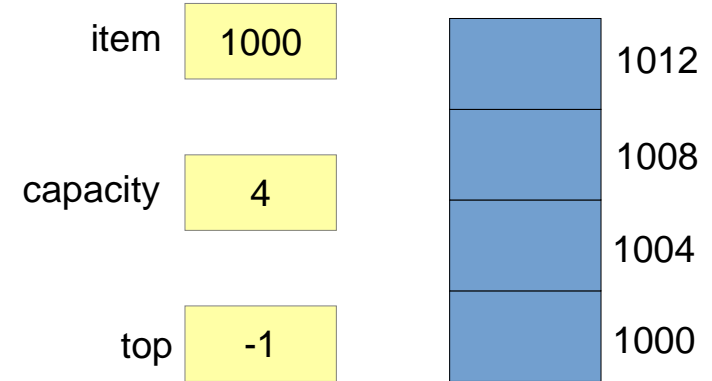
```
    return e_false
```

```
stack —→ capacity = size
```

```
stack —→ top = -1
```

```
return e_true
```

size = 4



# create\_stack(stack,size)



```
stack —→ item = Memalloc(sizeof(int) * size)
```

```
If (stack —→ item = NULL)
```

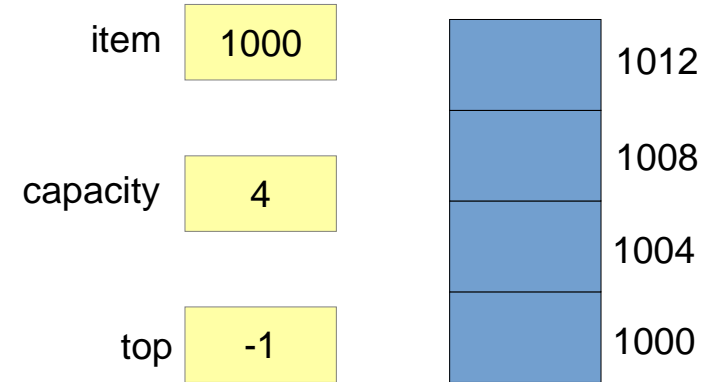
```
    return e_false
```

```
stack —→ capacity = size
```

```
stack —→ top = -1
```

```
return e_true
```

size = 4



Stack – push(stack, element)