Data Structures
Stack – Array Implementation

Team Emertxe

EMERTXE

Stack – Array Implementation

Data Structure –Stack-Array Implementation

Operations

- Create Stack
- Insert an Element
- Delete an Element
- Print top Element

Stack – create_stack(stack,size)

Data Structure –Stack-Array Implementation

create_stack(stack,size)

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

Data Structure –Stack-Array Implementation

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

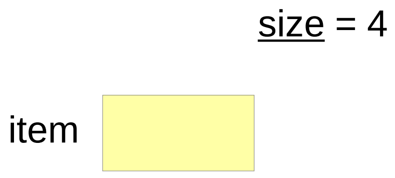
create_stack(stack,size)

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)

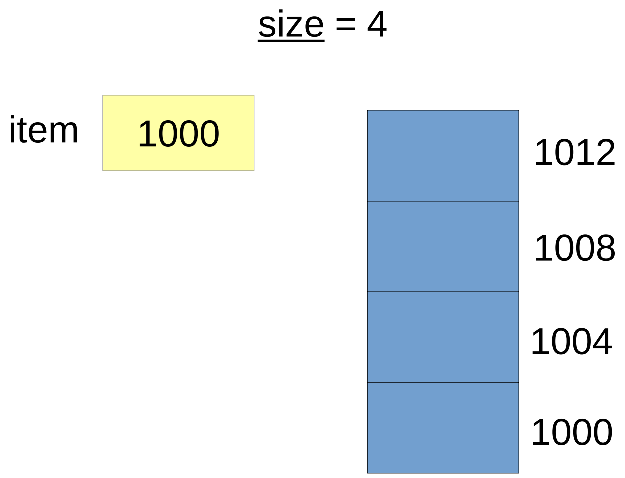
```
stack → item = Memalloc(sizeof(int) * size)
If (stack → item  = NULL)
    return e_false
stack → capacity = size
stack → top = -1
return e_true
```



Data Structure –Stack-Array Implementation

create_stack(stack,size)

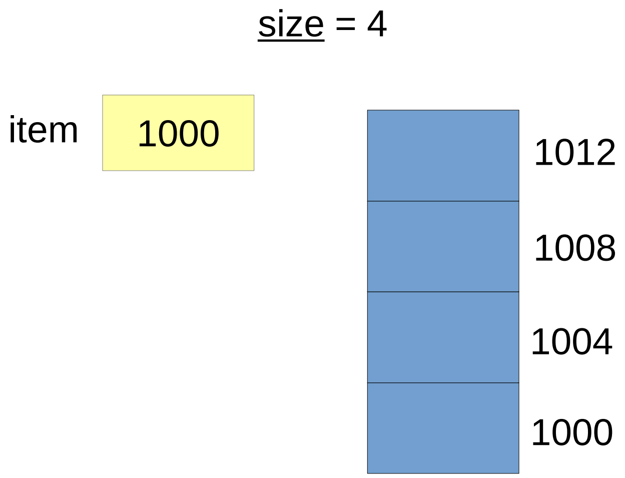
```
stack ⟶ item = Memalloc(sizeof(int) * size)
If (stack ⟶ item  = NULL)
      return e_false
stack ⟶ capacity = size
stack ⟶  top = -1
return e_true
```



Data Structure –Stack-Array Implementation

create_stack(stack,size)

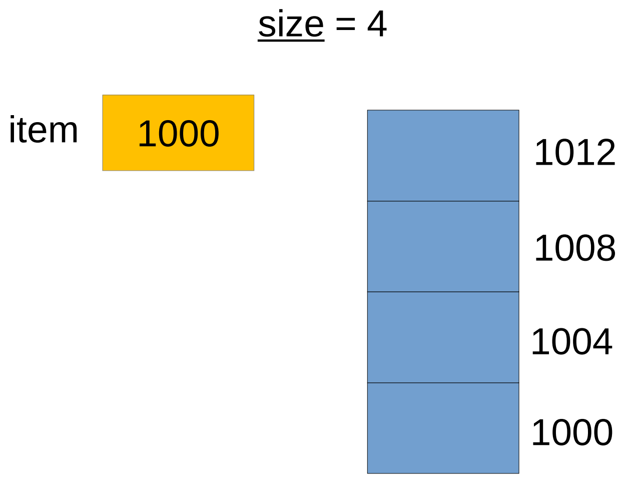
```
stack ⟶ item = Memalloc(sizeof(int) * size)
If (stack ⟶ item  = NULL)
      return e_false
stack ⟶ capacity = size
stack ⟶ top = -1
return e_true
```



Data Structure –Stack-Array Implementation

create_stack(stack,size)

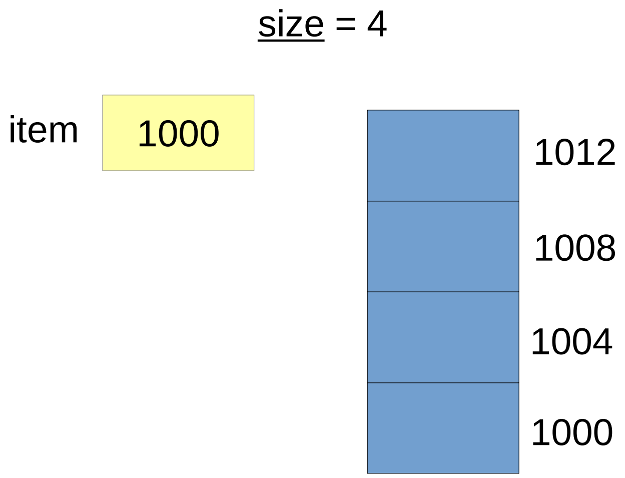
```
stack → item = Memalloc(sizeof(int) * size)
If (stack → item  = NULL)
      return e_false
stack → capacity = size
stack → top = -1
return e_true
```



Data Structure –Stack-Array Implementation

create_stack(stack,size)

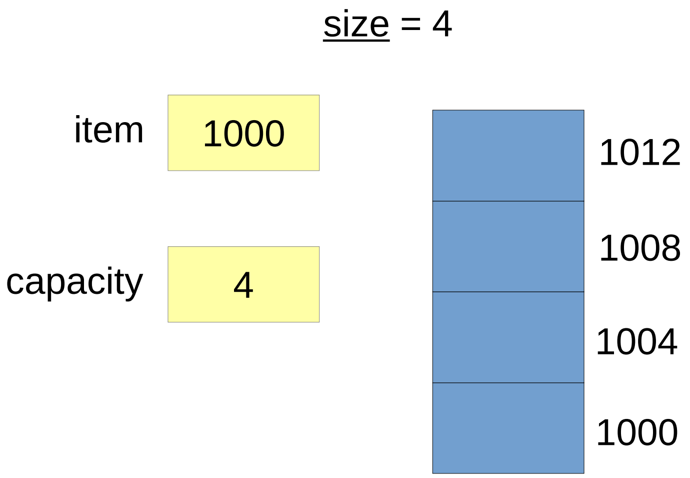
```
stack ⟶item = Memalloc(sizeof(int) * size)
If (stack ⟶ item  = NULL)
     return e_false
stack ⟶ capacity = size
stack ⟶   top = -1
return e_true
```



Data Structure –Stack-Array Implementation

create_stack(stack,size)

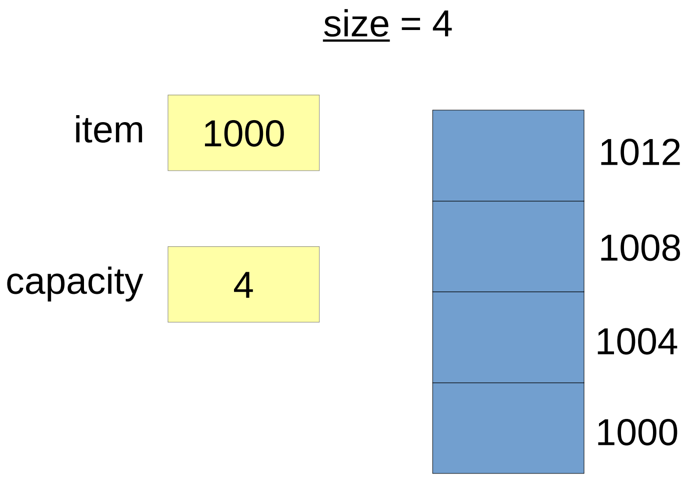
```
stack → item = Memalloc(sizeof(int) * size)
If (stack → item  = NULL)
      return e_false
stack → capacity = size
stack →   top = -1
return e_true
```



Data Structure –Stack-Array Implementation

create_stack(stack,size)

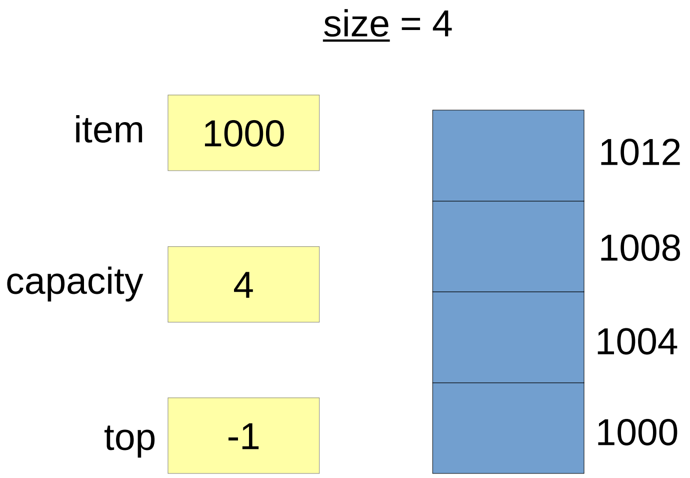
```
stack ⟶ item = Memalloc(sizeof(int) * size)
If (stack ⟶ item  = NULL)
      return e_false
stack ⟶ capacity = size
stack ⟶ top = -1
return e_true
```



Data Structure –Stack-Array Implementation

create_stack(stack,size)

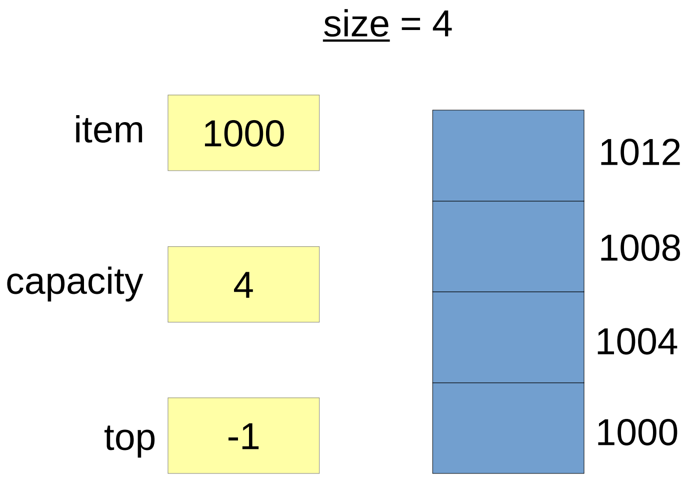
```
stack → item = Memalloc(sizeof(int) * size)
If (stack → item  = NULL)
      return e_false
stack → capacity = size
stack → top = -1
return e_true
```



Data Structure –Stack-Array Implementation

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
```



Stack – push(stack, element)