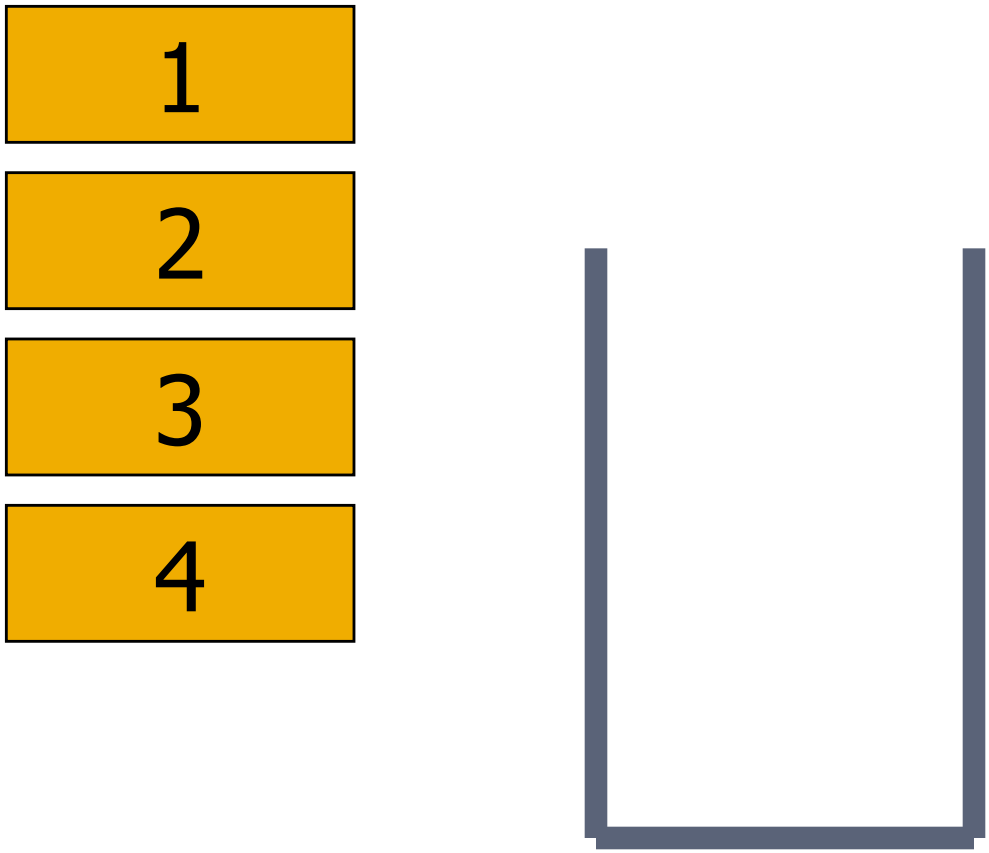
Lab 2: Two Stacks Implementation

# EPL231 – Data Structures and Algorithms

# Στοίβα (Stack)



# Υλοποίηση Στοίβας

- *MakeEmptyStack(S): Δημιουργία κενής στοίβας*
- *IsEmptyStack(S): Επιστρέφει true αν η στοίβα S είναι κενή*
- *Push(x,S): Τοποθετεί τον κόμβο x στην κορυφή της στοίβας S*
- *Pop(S): Διαγράφει τον κόμβο από την κορυφή της στοίβας S*
- *Top(S): Επιστρέφει τον κόμβο της κορυφής της στοίβας S*

# Υλοποίηση Στοίβας σε πίνακα

Ο τύπος δεδομένων που χρειάζεται για τη δομή είναι:

```
typedef struct _Stack{
        type list[size];
        int top;
} Stack;
```

Υλοποίηση πράξεων:

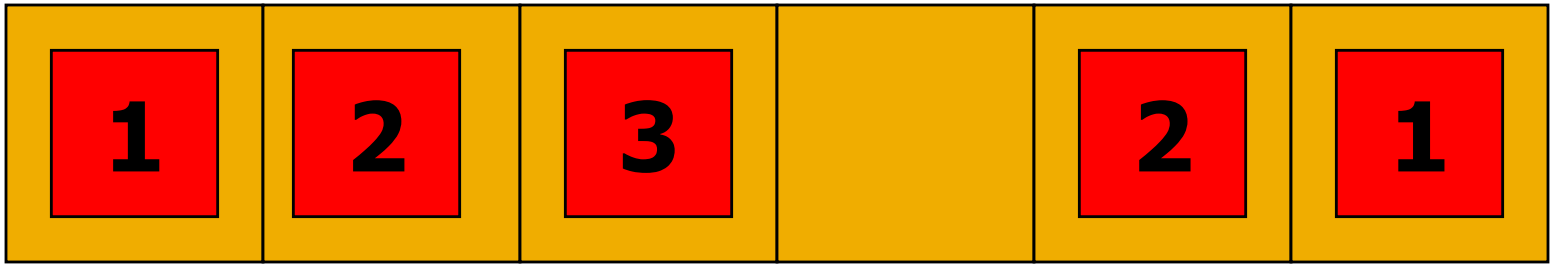
```
void makeEmpty (Stack *s){
   s->top = 0;
}
void push(type x, Stack *s){
   if (s->top <size) {
     s->list[s->top] = x;
     s->top++;
    }
}

int isEmpty(Stack *s){
   return (s->top == 0);
}
```

```
void pop(stack *s){
   if (!isEmpty(s))
       s->top--;
}

type top(stack *s){
   if !isEmpty(s)
     return s->list[s->top-1];
}
```

# Table Implementation of 2 Stacks



# Structures

```
#define MAX 25
#define LEFT 0
#define RIGHT 1

typedef int type;

typedef struct _TwoStack{
    type data[MAX];
    int  top1;
    int  top2;
}TwoStack;
```

# Begin Development [Part A]

- `void makeEmptyStack(TwoStack *s);`
- `int isEmpty(TwoStack *s);`
- `void push(int which, type x, TwoStack *s);`
- `void pop(int which, TwoStack *s);`
- `type top(int which, TwoStack *s);`
- `void printStack(TwoStack *s);`

# Solution

```
void makeEmptyStack(TwoStack *s){
    s->top1 = 0;
    s->top2 = MAX-1;
}


int isEmpty(TwoStack *s){
    return ((s->top1==0) &&(s->top2==(MAX-1)));
}
```

# Solution

```
void push(int i, type x, TwoStack *s){
   if (s->top1 <= s->top2){
       if ( i==LEFT ){
          s->data[s->top1]=x;
          s->top1++;
       }
       else if ( i==RIGHT ){
          s->data[s->top2]=x;
          s->top2--;
       }
       else {
           fprintf(stderr, "Invalid Stack Selection.\n");
           exit(1);
       }
   }
   else {
       fprintf(stderr, "Not Enough Space\n");
   }
}
```

# Solution

```
type top(int which, TwoStack *s)
{
   if ((which==LEFT) && (s->top1!=0))
       return s->data[s->top1-1];
   else if((which==RIGHT)&&(s->top2!=(MAX-1)))
       return s->data[s->top2+1];
   else
       fprintf(stderr, "top() operation cannot be executed!");
}
```

```
void pop (int which, TwoStack *s)
{
   if ((which==LEFT) && (s->top1!=0))
       s->top1--;
   else if((which==RIGHT)&&(s->top2!=(MAX-1)))
       s->top2++;
   else
       fprintf(stderr, "pop() operation cannot be executed!");
}
```

# Solution

```
void printStack(TwoStack *s){
   if(isEmpty(s))
       printf("The stack is empty.\n");
   else{
       int i = 0;
       printf("\n\nPrinting the stacks...\n|");
       printf("Stack 1: ");
       for(i=0; i<s->top1; i++)
              printf("%d\t|", s->data[i]);
       printf("Stack 2: ");
       for(i=MAX-1; i>s->top2; i--)
              printf("%d\t|", s->data[i]);
      printf("Done\n\n");
   }
}
```

# Begin Development [Part B]

- `typedef TwoStack *TwoStackPtr;`
  - `void` **`makeEmptyStack`**`(TwoStackPtr s);`
  - `int` **`isEmpty`**`(TwoStackPtr s);`
  - `void` **`push`**`(int i, type x, TwoStackPtr s);`
  - `void` **`pop`**`(int i, TwoStackPtr s);`
  - `type` **`top`**`(int i, TwoStackPtr s);`
  - `void` **`printStack`**`(TwoStackPtr s);`

# The End