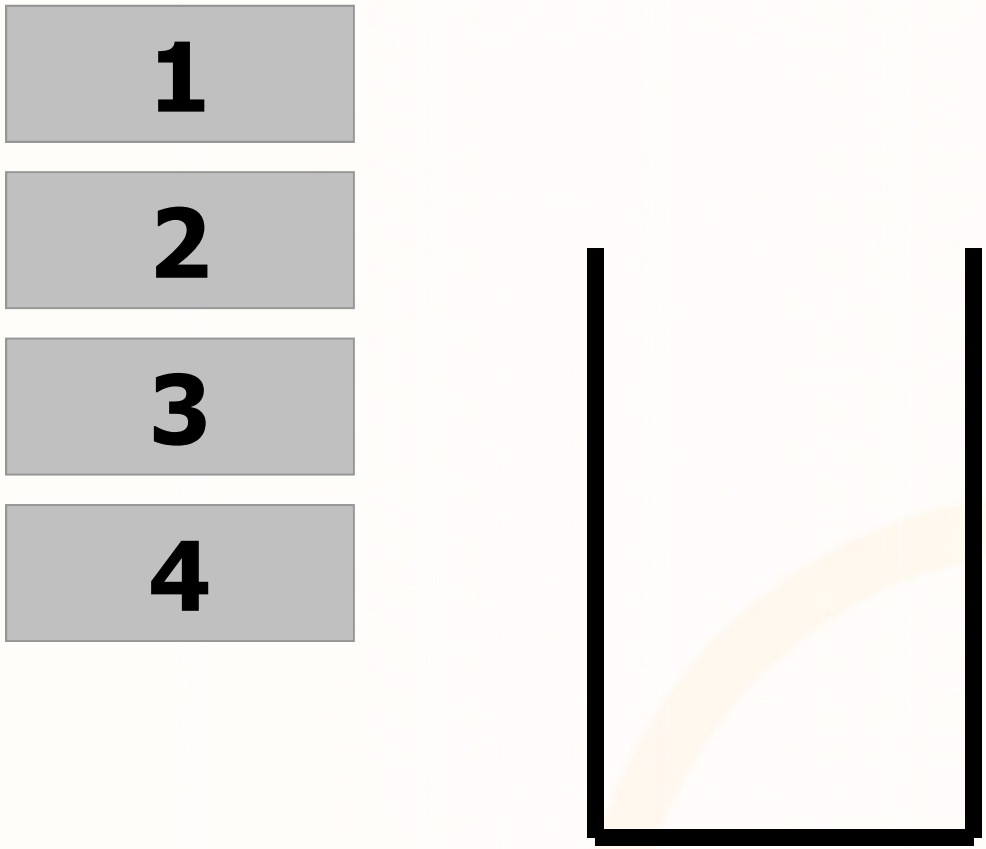
Εργαστήριο 3: Υλοποίηση Αφηρημένου Τύπου Δεδομένων: Διπλή Στοίβα

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

- Στοίβες
- Υλοποίηση διπλής Στοίβας με στατική δέσμευση μνήμης

Παράδειγμα στοίβας



Υλοποίηση Στοίβας

Τυπικές πράξεις σε μία στοίβα:

- **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

Επανάληψη: Στοίβα με Στατική Δέσμευση Μνήμης

- Ο τύπος δεδομένων για τη **Στοίβα** με **Στατική** Δέσμευση Μνήμης είναι:

```
typedef struct {
    type list[ size ]; //π.χ., size=10
    int Length;
} STACK;
```

→ **Στατική Δέσμευση size θέσεων μνήμης**

- **Υλοποίηση πράξεων:**

```
void MakeEmpty(STACK *S){
    S->Length = 0;
}
```

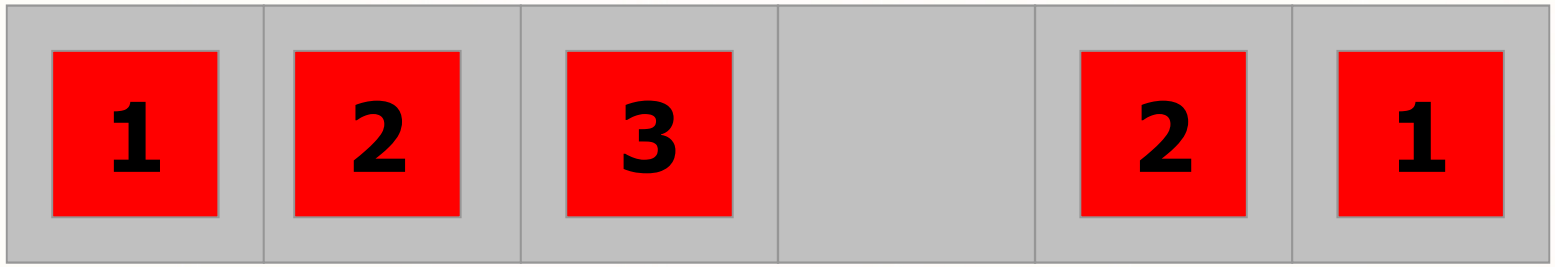
```
int IsEmpty(STACK *S){
    return (S->Length == 0);
}
```

```
type Top(STACK *S){
  if ( !IsEmpty(S) )
    return S->list[(S->Length)-1];
}
```

```
void Push(type x, STACK *S){
  if ((S->Length) < size) {
    S->list[S->Length]= x;
    (S->Length)++;
  }
}
```

```
void Pop(STACK *S){
   if ( !IsEmpty(S) )
      (S->Length)--;
}
```

Παράδειγμα διπλής στοίβας με τον ίδιο πίνακα



Υλοποίηση Διπλής Στοίβας με πίνακα

Δομές που θα χρειαστείτε:

```
//MAX: max elements in stack (TwoStack)
#define MAX 10


//LEFT: stack 1 - left stack (begining of table)
//RIGHT: stack 2 - right stack (end of table)
enum stackenum{LEFT, RIGHT};


//easily change the implementation from int to
other type, e.g., typedef double type;
typedef int type;
```

Υλοποίηση Διπλής Στοίβας με πίνακα (συν.)

Δομές που θα χρειαστείτε:

```
//The TwoStack data structure
typedef struct TwoStack{
  typedata[MAX];

  //index to top of stack [1]
  inttop1;

  //index to top of stack [2]
  inttop2;

}TwoStack;
```

Συναρτήσεις που πρέπει να υλοποιήσετε

- `void MakeEmptyStack(TwoStack *s);`
- `bool IsEmpty(TwoStack *s);`
- `void PrintStack(TwoStack *s);`
- `void Push(stackenum direction, type x, TwoStack *s);`
- `void Pop(stackenum direction, TwoStack *s);`
- `type Top(stackenum direction, TwoStack *s);`

Λύσεις

```
void MakeEmptyStack(TwoStack *s) {
    (*s).top1 = 0;
    (*s).top2 = MAX-1;
    for(int i=0; i<MAX; i++)
        (*s).data[i] = 0;
}
```

Λύσεις (συν.)

```
bool IsEmpty(TwoStack *s){
    return (((*s).top1==0)
            &&((*s).top2==(MAX-1)));
}
```

Λύσεις (συν.)

```
void PrintStack(TwoStack *s){
  if(IsEmpty(s)) printf("The stack is
empty.\n");
  else{
    printf("\n\nPrinting the stack...\n|");
     for(int i=0; i<MAX; i++)
        printf("%d\t|", (*s).data[i]);
   printf("Done\n\n");
  }
}
```

Λύσεις (συν.)

```
void Push(stackenum direction, type x, TwoStack *s){
      if ((*s).top1 <= (*s).top2){
            if ( direction==LEFT ){
               (*s).data[(*s).top1]=x;
               (*s).top1++;
            }
            if ( direction==RIGHT ){
               (*s).data[(*s).top2]=x;
               (*s).top2--;
            }
      }
}
```

Λύσεις (συν.)

```
type Top(stackenum direction, TwoStack *s){
     if(IsEmpty(s)) printf("The stack is
empty.\n");
     else{
        if ( direction==LEFT )
          return (*s).data[(*s).top1-1];
        if ( direction==RIGHT )
          return (*s).data[(*s).top2+1];
     }
}
```

Λύσεις (συν.)

```
void Pop(stackenum direction, TwoStack *s){
   if(IsEmpty(s)) printf("The stack is
empty.\n");
   else{
         if ( (direction==LEFT)
              && ((*s).top1!=0) )
            (*s).top1--;
         if( (direction==RIGHT)
              &&((*s).top2!=(MAX-1)))
            (*s).top2++;
     }
}
```