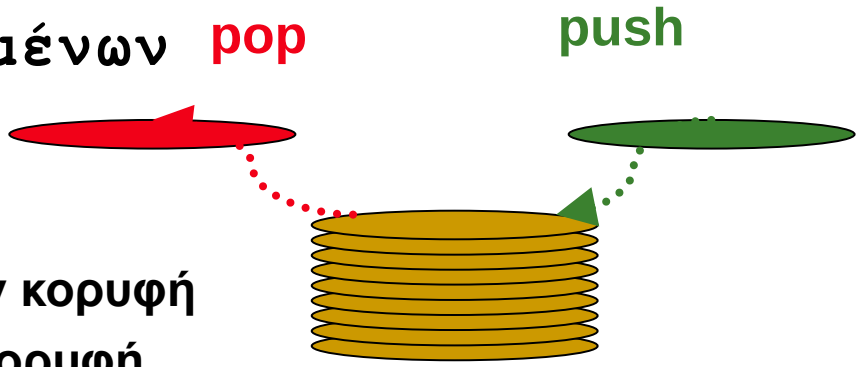
Δομές Δεδομένων

Στοίβες - Υλοποίηση

Στοίβα

- Δομή αποθήκευσης δεδομένων
  - Λειτουργία LIFO
- Δυνατότητες
  - Προσθήκης νέου δεδομένου στην κορυφή
  - Ανάκτησης δεδομένου από την κορυφή
- Ο χρήστης της δομής ενδιαφέρεται για:
  - τον τύπο δεδομένων (int, char, float, struct ? κ.λ.π.)
  - λειτουργία σύμφωνη με LIFO
  - τρόπο χειρισμού της δομής



An Application Programming Interface (API) is a set of routines, data structures, object classes and/or protocols provided by libraries and/or operating system services in order to support the building of applications.

Παράδειγμα API στοίβας

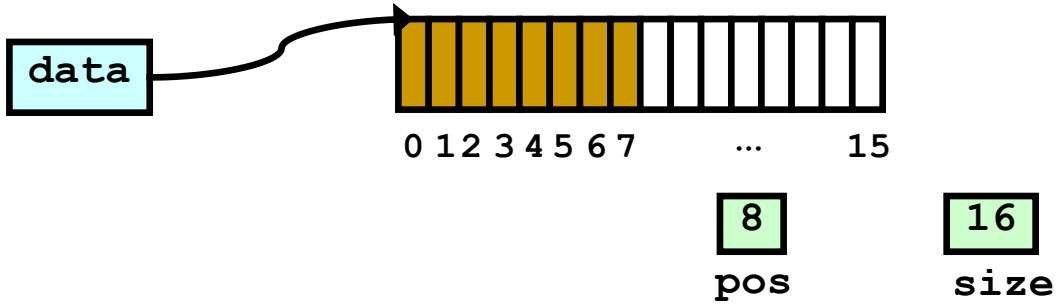
- Όνομα Κλάσης: **Stack**
- Δομή Δεδομένων: **int**
- Αρχικοποίηση Στοιβας:
  - `Stack mystack = new Stack();`
    `// Κενή στοίβα προκαθορισμένου αρχικού μεγέθους (10)`
  - `Stack mystack = new Stack(int k);`
    `// Κενή στοίβα αρχικού μεγέθους (k)`
  - `Stack mystack = new Stack(Stack &oldstack);`
    `// Στοίβα αντίγραφο της oldstack`
- Χειρισμός Στοιβας:
  - `bool pop(int &element);`

    `// Σε άδεια στοίβα επιστρέφει false. Αλλιώς επιστρέφει true και το πρώτο στοιχείο στο element`
  - `bool push(int element);`

    `// Σε γεμάτη στοίβα επιστρέφει false. Αλλιώς επιστρέφει true και τοποθετεί στη κορυφή το element`
  - `bool isEmpty();`

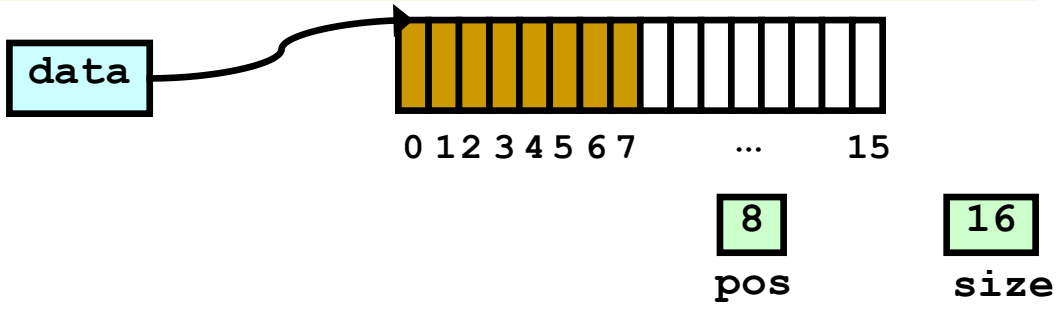
    `// Επιστρέφει true αν η στοίβα είναι άδεια, αλλιώς false`

Κλάση Στοίβας



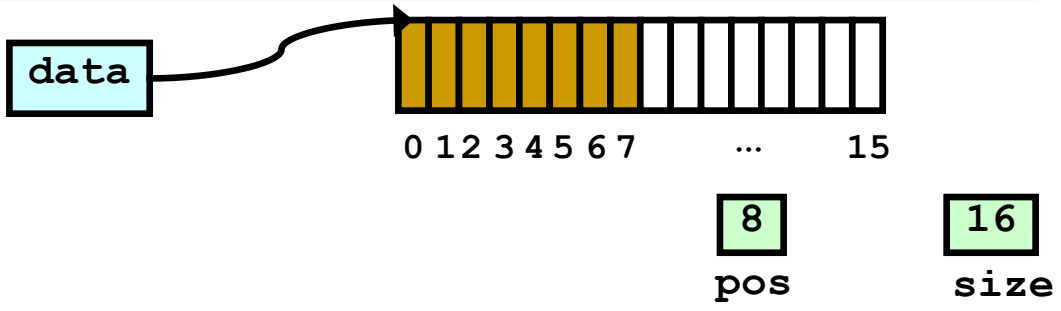

```
class Stack
{
   private:
     int size; // Μέγεθος πίνακα
     int pos;  // 1η ελεύθερη θέση
     int *data; // Πίνακας δεδομένων
   public:
     bool pop(int &element);
     bool push(int element);
     bool isEmpty();
     Stack();
     Stack(int n);
     Stack(Stack &other);
     ~Stack();
};
```

Κλάση Στοίβας



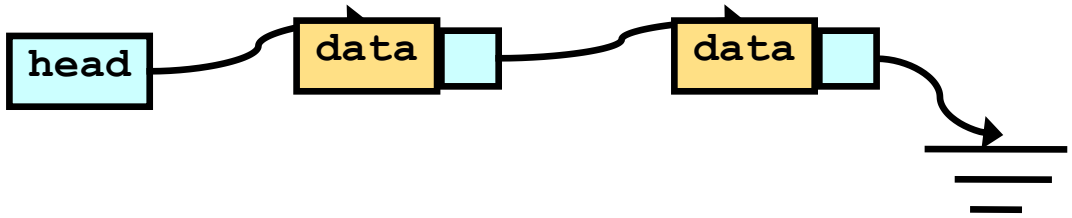

```
struct mydata
{
    int col, line;
};
class Stack
{
   private:
     int size; // Μέγεθος πίνακα
     int pos;  // 1η ελεύθερη θέση
     mydata *data; // Πίνακας δεδομένων
   public:
     bool pop(mydata &element);
     bool push(mydata element);
     bool isEmpty();
     Stack();
     Stack(int n);
     Stack(Stack &other);
     ~Stack();
};
```

Κλάση Στοίβας




```
class mydata
{
    int col, line;
};
class Stack
{
   private:
     int size; // Μέγεθος πίνακα
     int pos;  // 1η ελεύθερη θέση
     mydata *data; // Πίνακας δεδομένων
   public:
     bool pop(mydata &element);
     bool push(mydata element);
     bool isEmpty();
     Stack();
     Stack(int n);
     Stack(Stack &other);
     ~Stack();
};
```

Κλάση Στοίβας




```
class Stack
{
  private:
      struct node
      {
          int data;
          node *next;
      } *head;
  public:
      bool pop(int &element);
      bool push(int element);
      bool isEmpty();
      Stack();
      Stack(int n);
      Stack(Stack &other);
      ~Stack();
};
```

```
class Stack
{
    private:
      int size;
      int pos;
      int *data;
    public:
      bool pop(int &element);
      bool push(int element);
      bool isEmpty();
      Stack();
      Stack(int n);
      Stack(Stack &other);
      ~Stack();
};
```

Κενός Κατασκευαστής

```
class Stack
{
   private:
     int size; // Μέγεθος πίνακα
     int pos;  // 1η ελεύθερη θέση
     int *data; // Πίνακας δεδομένων
     …
};
```

```
Stack::Stack()
{
     size = 10;
     pos=0;
     data = new int[size];
}
```

- `Stack mystack = new Stack();`
  `// Κενή στοίβα προκαθορισμένου αρχικού μεγέθους (10)`
- `Stack mystack = new Stack(int k);`
  `// Κενή στοίβα αρχικού μεγέθους (k)`
- `Stack mystack = new Stack(Stack &oldstack);`
  `// Στοίβα αντίγραφο της oldstack`

Κατασκευαστής με μέγεθος

```
class Stack
{
   private:
      int size; // Μέγεθος πίνακα
      int pos;  // 1η ελεύθερη θέση
      int *data; // Πίνακας δεδομένων
      …
};
```

```
Stack::Stack(int n)
{
    size = n;
    pos=0;
    data = new int[size];
}
```

- `Stack mystack = new Stack();`
  `// Κενή στοίβα προκαθορισμένου αρχικού μεγέθους (10)`
- `Stack mystack = new Stack(int k);`
  `// Κενή στοίβα αρχικού μεγέθους (k)`
- `Stack mystack = new Stack(Stack &oldstack);`
  `// Στοίβα αντίγραφο της oldstack`

Κατασκευαστής αντιγράφου

```
class Stack
{
   private:
     int size; // Μέγεθος πίνακα
     int pos;  // 1η ελεύθερη θέση
     int *data; // Πίνακας δεδομένων
     …
};
```

```
Stack::Stack(Stack &other)
{
    size = other.size;
    pos = other.pos;
    data = new int[size];
    for (int i=0;i<size;i++)
    {
        data[i]=other.data[i];
    }
}
```

- `Stack mystack = new Stack();`
  // Κενή στοίβα προκαθορισμένου αρχικού μεγέθους (10)
- `Stack mystack = new Stack(int k);`
  // Κενή στοίβα αρχικού μεγέθους (k)
- `Stack mystack = new Stack(Stack &oldstack);`
  // Στοίβα αντίγραφο της oldstack

Καταστροφέας

```
Stack::~Stack()
{
     delete[] data;
}
```

Συνάρτηση isEmpty()

```
bool Stack::isEmpty()
{
     return (pos==0);
}
```

Συνάρτηση pop()

```
bool Stack::pop(int &element)
{
    if (pos==0)
    {
        return false; // Stack empty
    }
    pos--;
    element=data[pos];
    return true; // completed normally
}
```

Συνάρτηση push()

```
bool Stack::push(int element)
{
    if (pos==size)
    {
        return false; // Stack full
    }
    data[pos]=element;
    pos++;
    return true; // completed normally
}
```

Κλάση Stack – 2η Έκδοση

```
class Stack
{
   private:
      int size;
      int pos;
      int *data;
      bool moreMemory(int n);
   public:
      bool pop(int &element);
      bool push(int element);
      bool isEmpty();
      Stack();
      Stack(int n);
      Stack(Stack &other);
      ~Stack();
};
```

Συνάρτηση push() - 2η Έκδοση

```
bool Stack::push(int element)
{
    if (pos==size)
    {
        if (!moreMemory(size+5))
            return false; // memory full
    }
    data[pos]=element;
    pos++;
    return true; // completed normally
}
```

Συνάρτηση moreMemory()

```
bool Stack::moreMemory(int n)
{
    int *temp;
    temp = new int[n];
    if (temp==NULL)
       return false;
    for (int i=0;i<size;i++)
       temp[i]=data[i];
    delete[] data;
    data=temp;
    size=n;
    return true;
}
```

Ένα παράδειγμα χρήσης της στοίβας

```
main()
{
        Stack s(30);
        int i, k;
        for (i=0;i<30;i++)
        {
            s.push(i);
        }
        for (i=0;i<30;i++)
        {
            if (s.pop(k))
                cout<<k<<endl;
            else
                cout<<"Error"<<endl;
        }
        getchar();
}
```