Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

http://www.clipartpanda.com

Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

Schritt 1: Wir erzeugen einen leeren Stack s.

Das LIFO-Prinzip

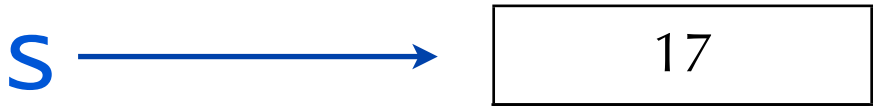
Demonstration der Arbeitsweise eines Stacks

Schritt 1: Wir erzeugen einen leeren Stack s.

Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

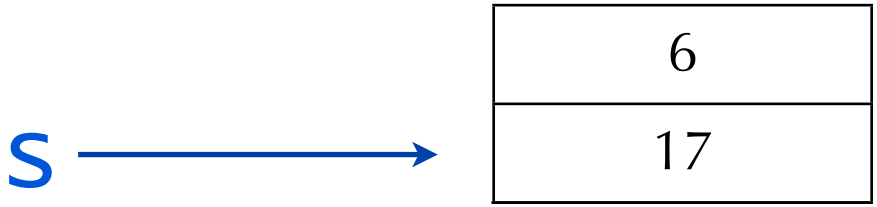
Schritt 2: Wir speichern die Zahl 17 in dem Stack.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

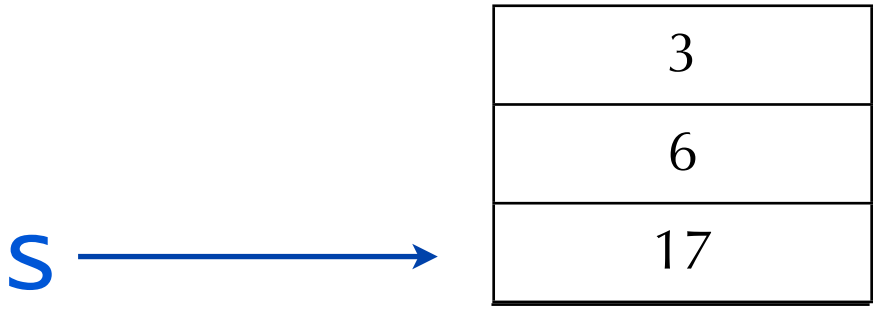
Schritt 3: Wir speichern die Zahl 6 in dem Stack.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

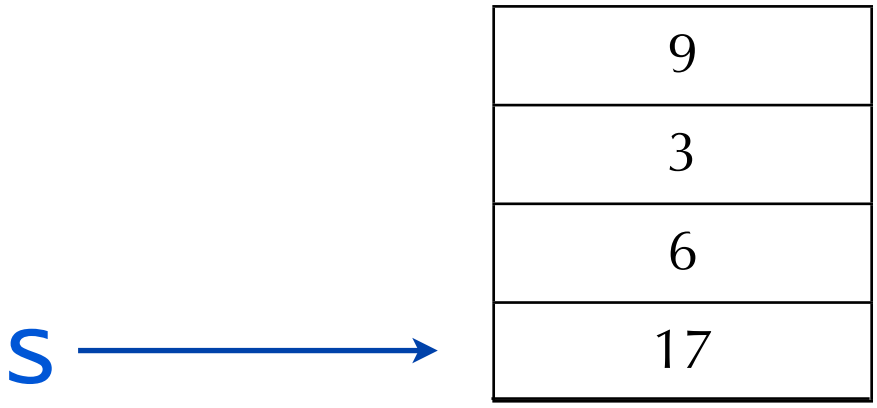
Schritt 4: Wir speichern die Zahl 3 in dem Stack.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

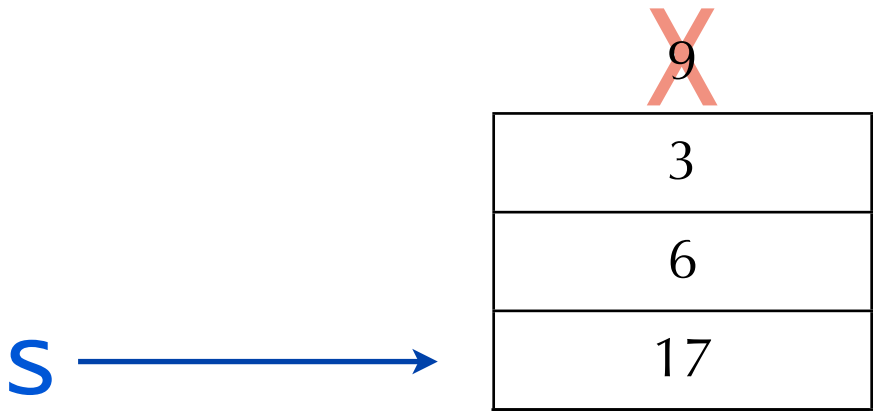
Schritt 5: Wir speichern die Zahl 9 in dem Stack.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

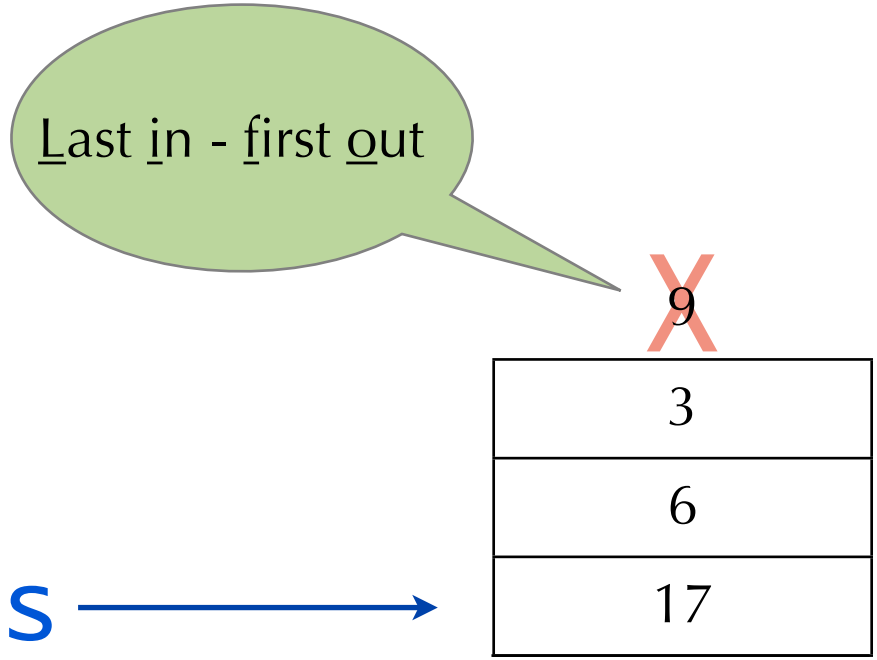
Schritt 6: Wir führen die Operation pop() aus.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

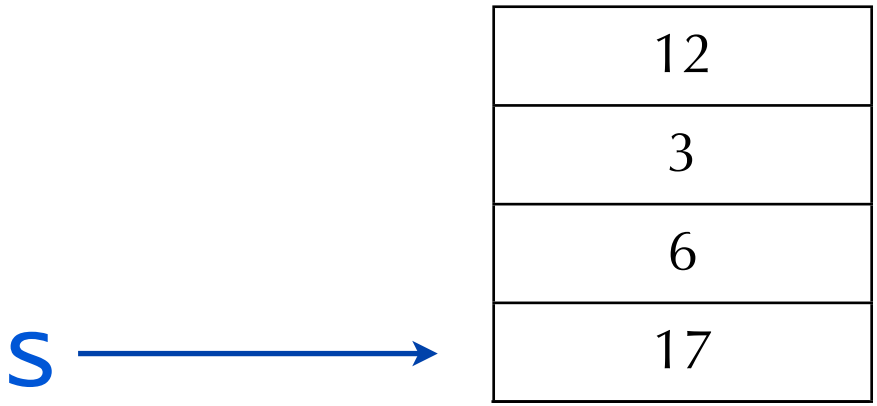
Schritt 6: Wir führen die Operation pop() aus.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

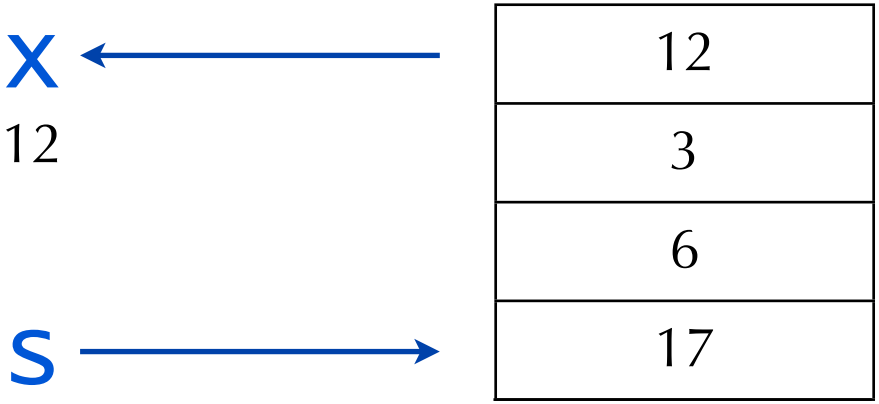
Schritt 7: Wir führen die Operation push(12) aus.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

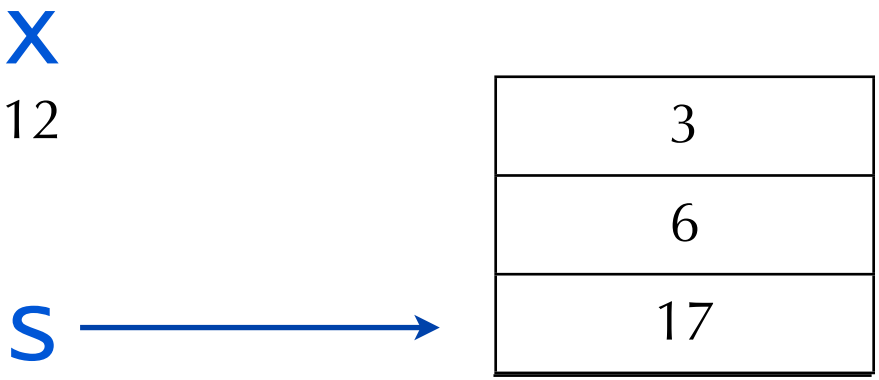
Schritt 8: Wir führen die Operation top() aus.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

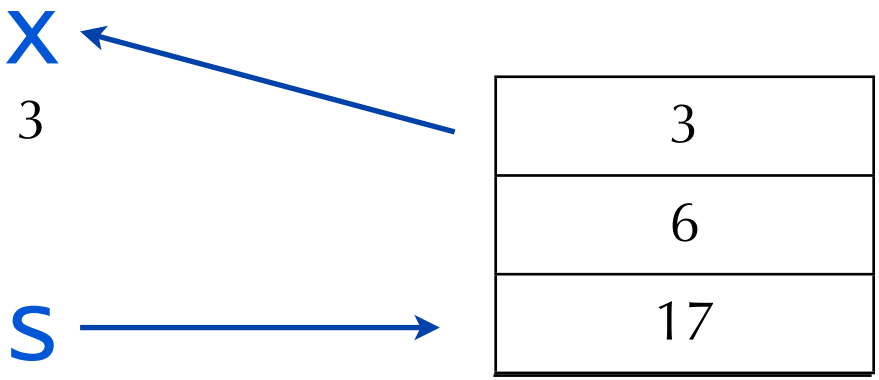
Schritt 9: Wir führen die Operation pop() aus.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

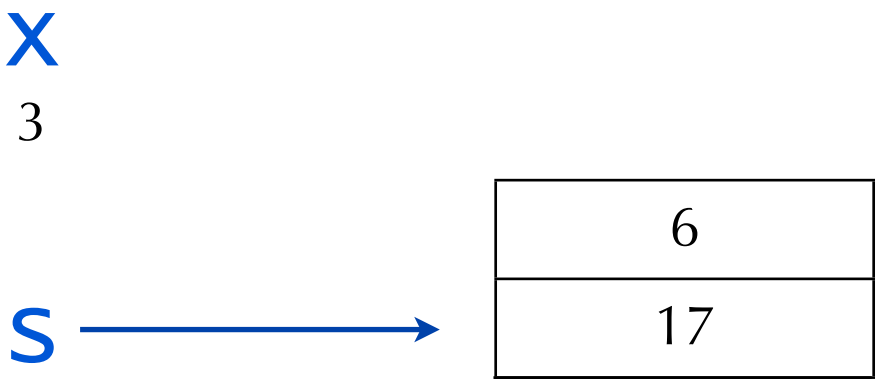
Schritt 10: Wir führen die Operation top() aus.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

Schritt 11: Wir führen die Operation pop() aus.



Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

Schritt 12: Wir führen die Operation pop() aus.

X

3

S → 17

Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

Schritt 13: Wir führen die Operation pop() aus.

X

3

S ⟶ _____

Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

Schritt 14: Wir führen die Operation pop() aus.

X

3

keine Fehlermeldung!

S ⟶ _____

Das LIFO-Prinzip

Demonstration der Arbeitsweise eines Stacks

Schritt 15: Wir führen die Operation top() aus.

X

-99

Fehlermeldung oder error-Wert* wie -99.

S ⟶ _____

*Dieser Wert darf nicht in der Wertemenge des Stacks vorkommen.

Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

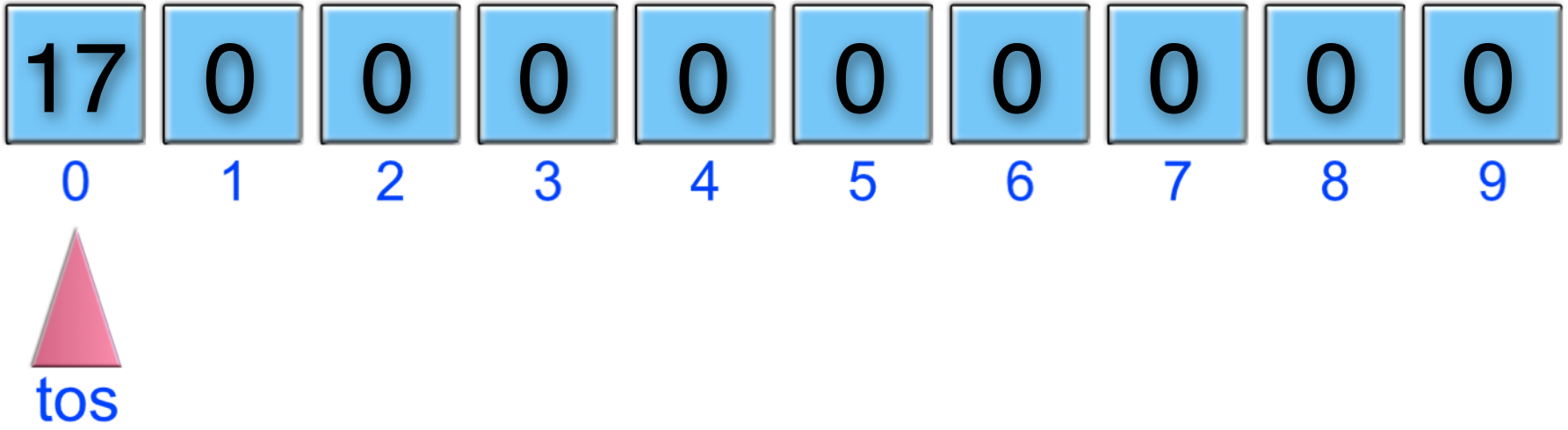
Schritt 1: Erzeugung eines leeren int-Arrays aus 10 Zahlen

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

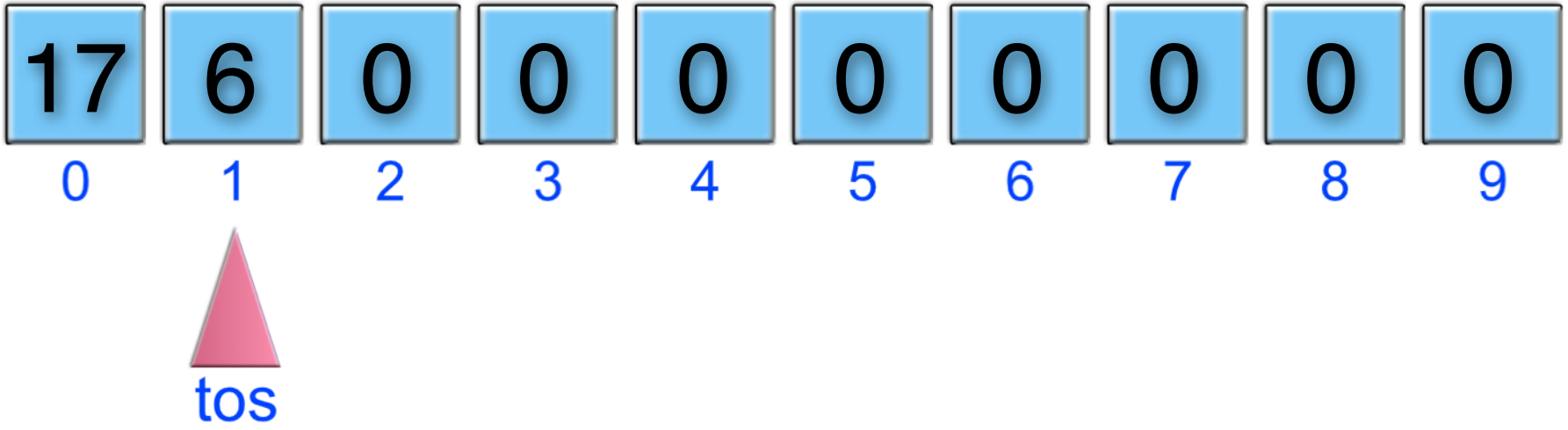
Schritt 2: Wir führen die Operation Push(17) aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

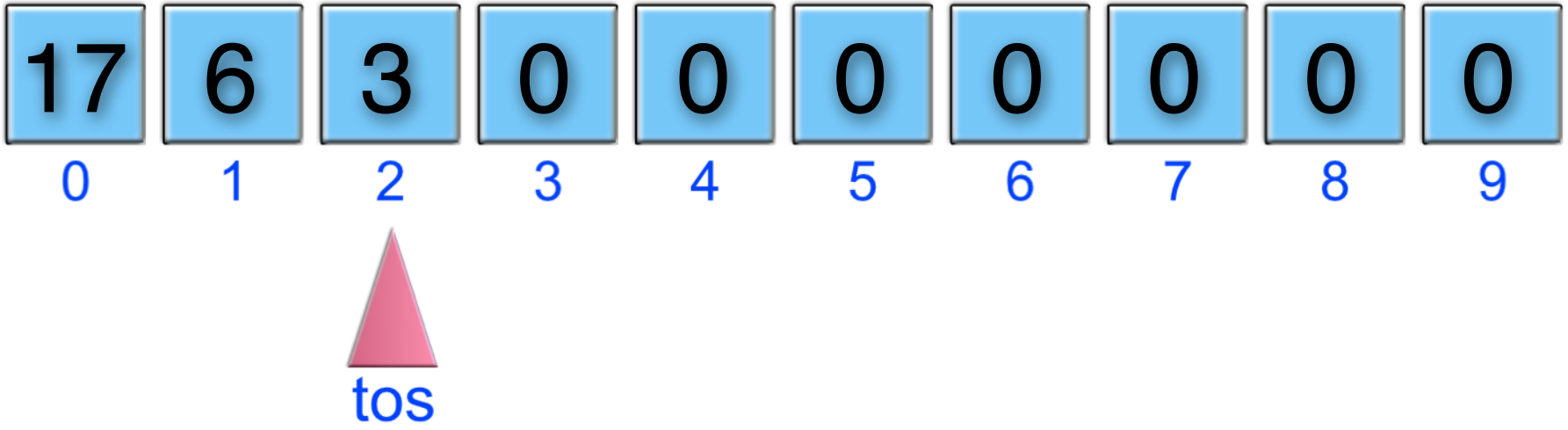
Schritt 3: Wir führen die Operation Push(6) aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

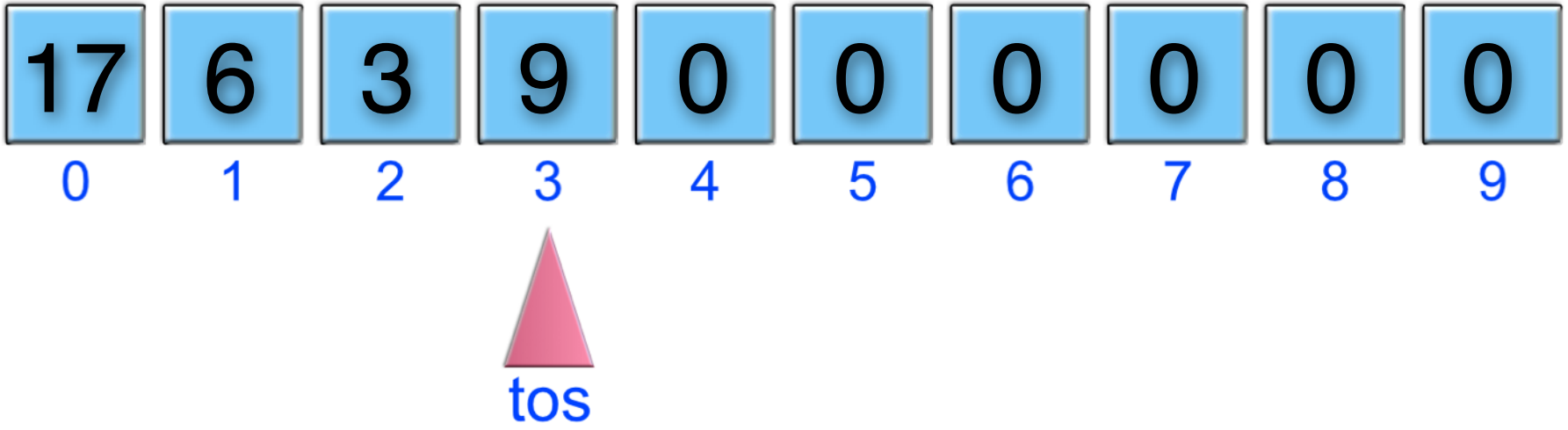
Schritt 4: Wir führen die Operation Push(3) aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

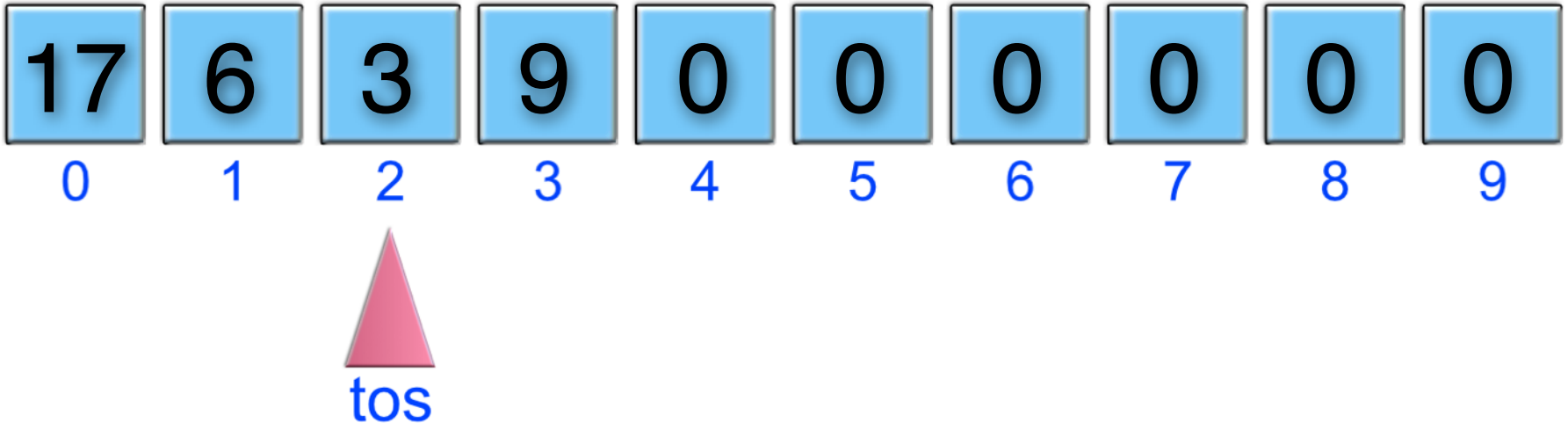
Schritt 5: Wir führen die Operation Push(9) aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

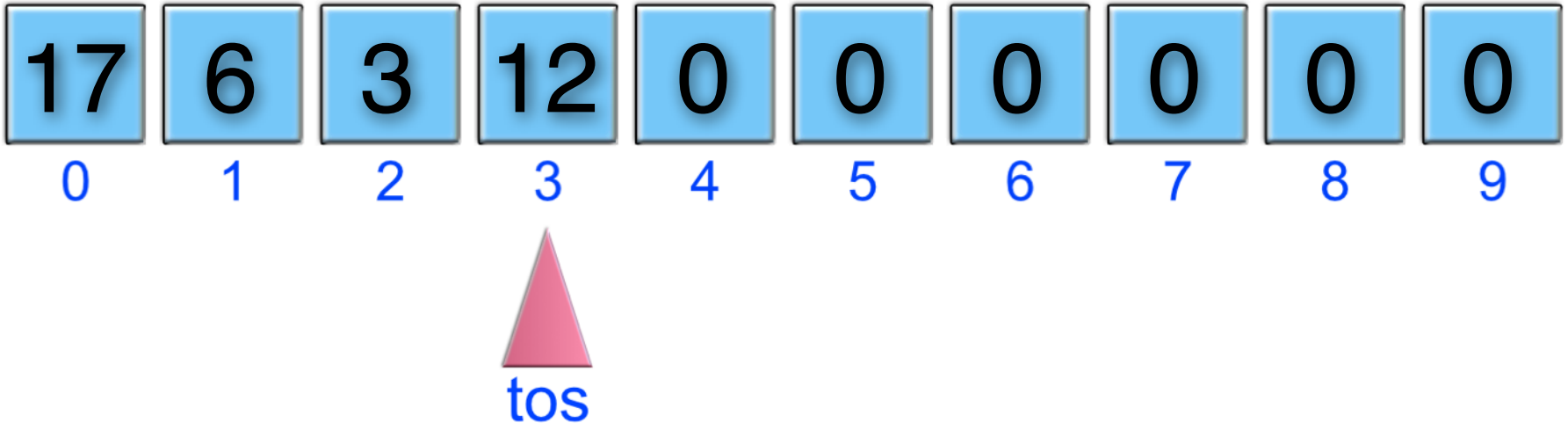
Schritt 6: Wir führen die Operation Pop() aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

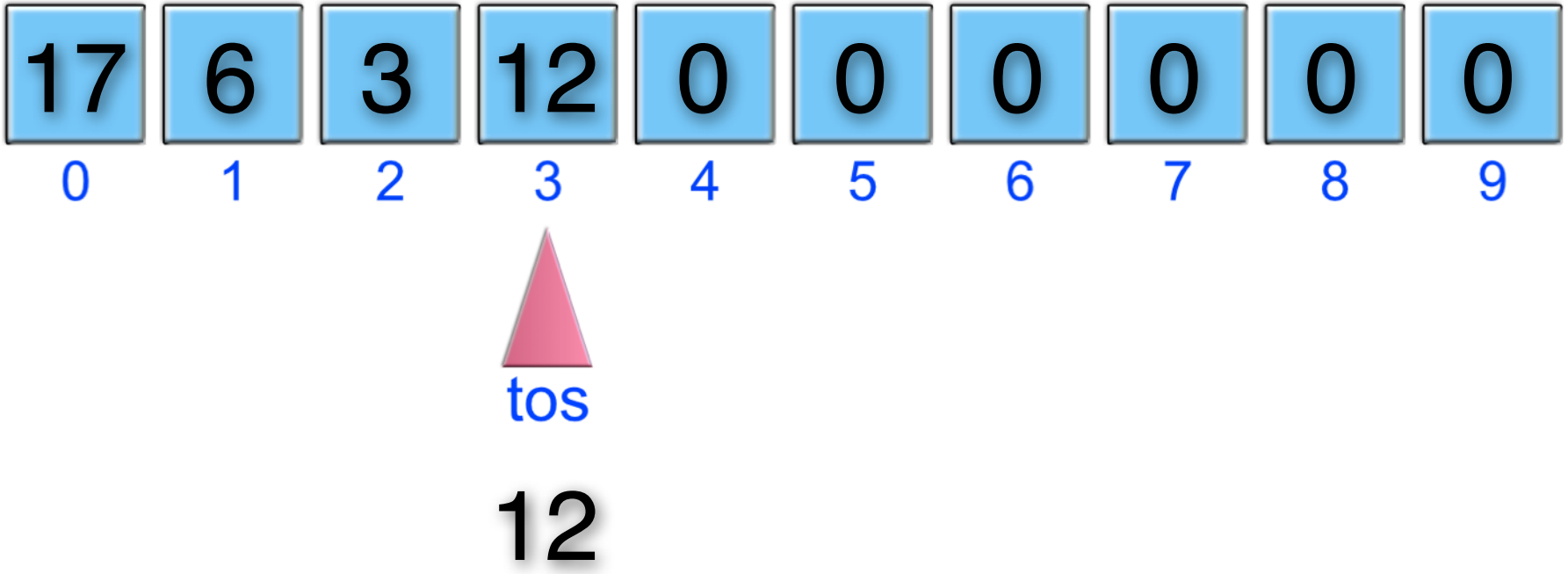
Schritt 7: Wir führen die Operation Push(12) aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

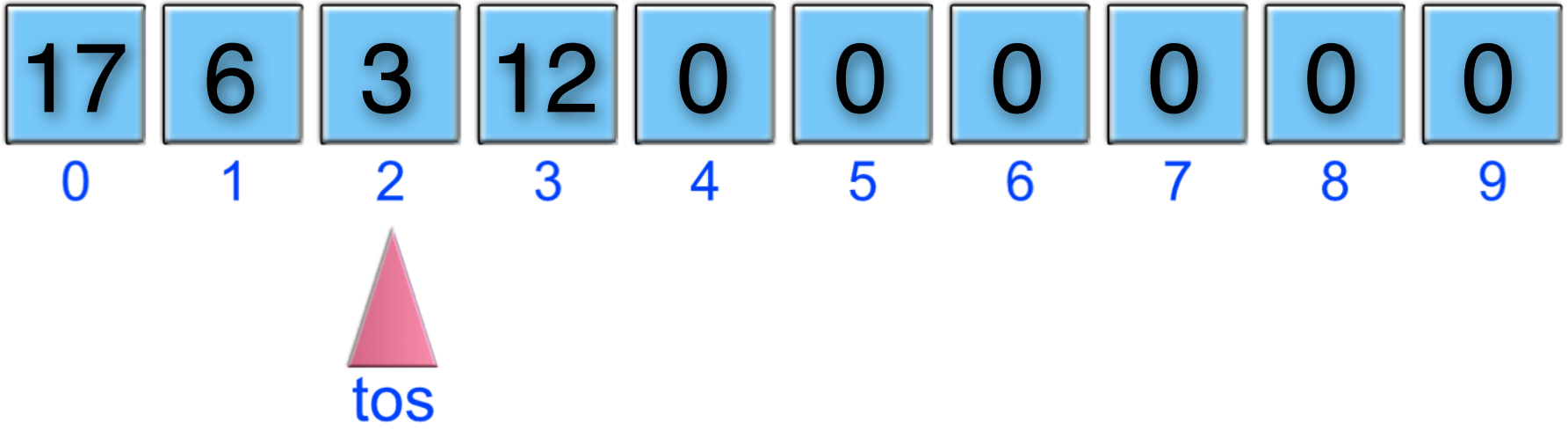
Schritt 8: Wir führen die Operation Top() aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

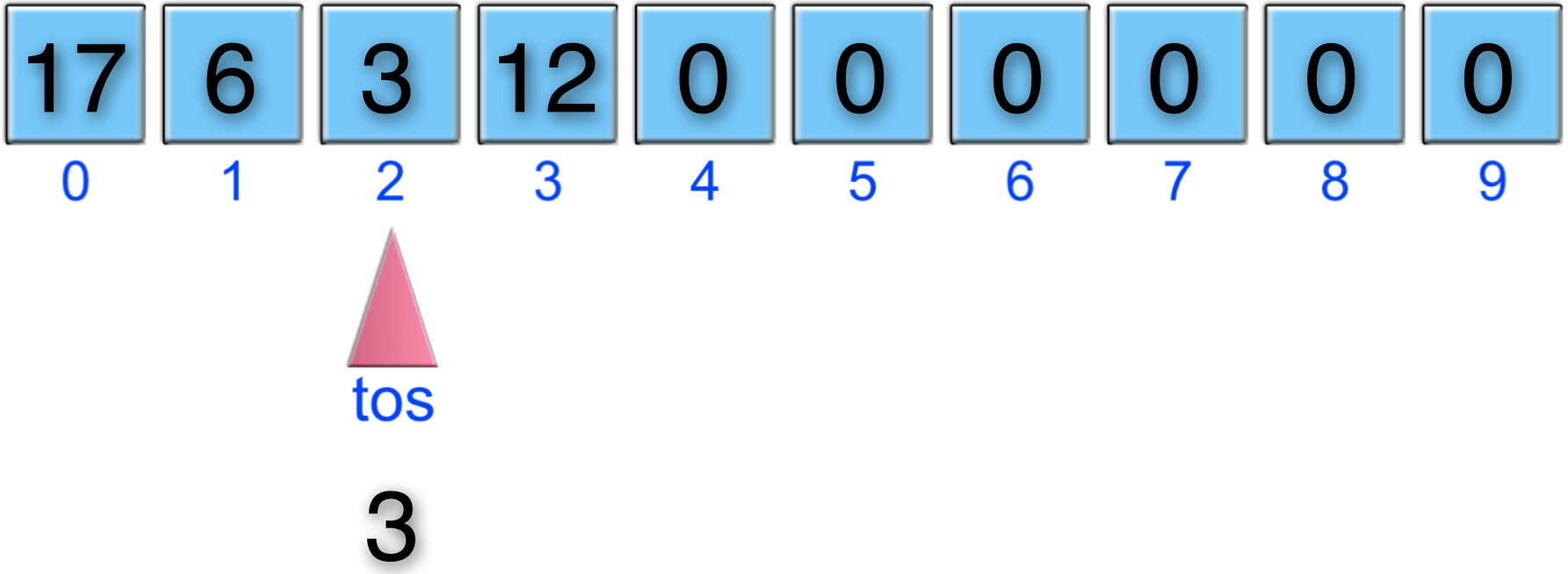
Schritt 9: Wir führen die Operation Pop() aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

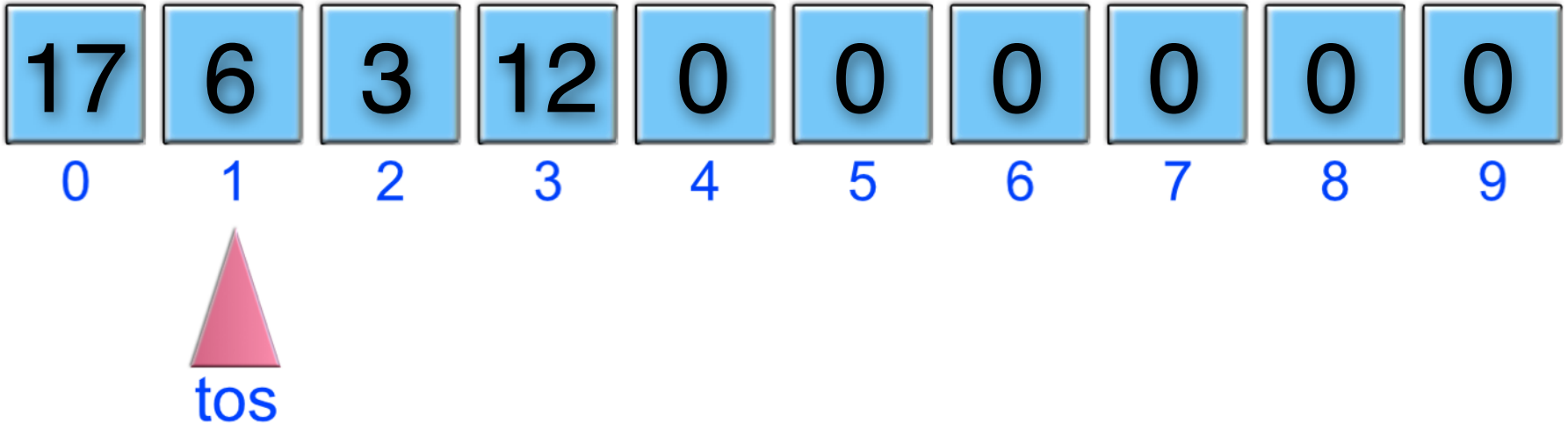
Schritt 10: Wir führen die Operation Top() aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

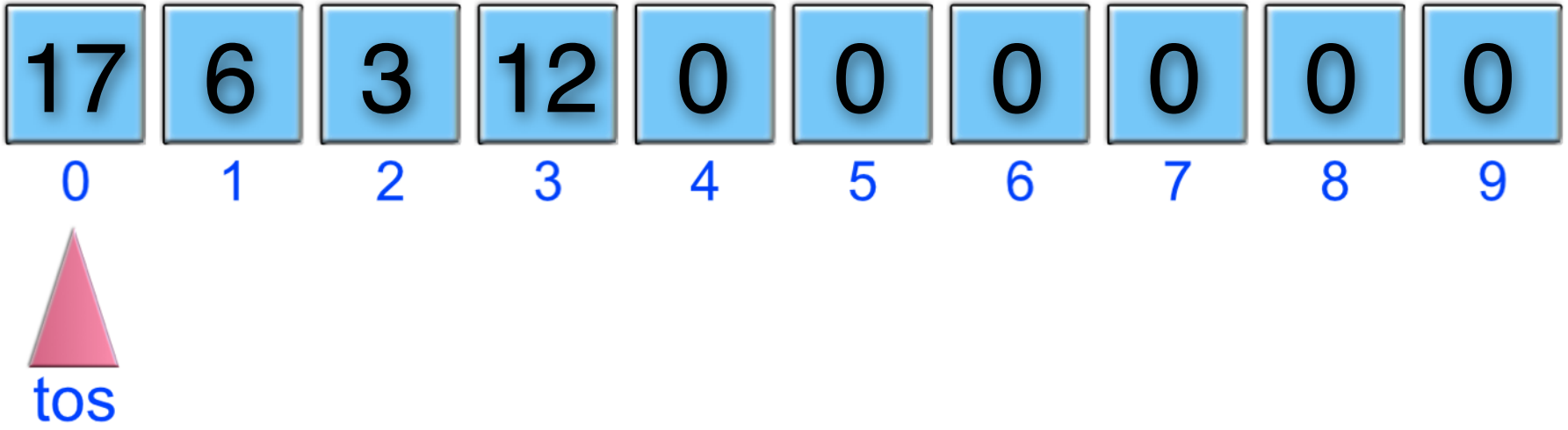
Schritt 11: Wir führen die Operation Pop() aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

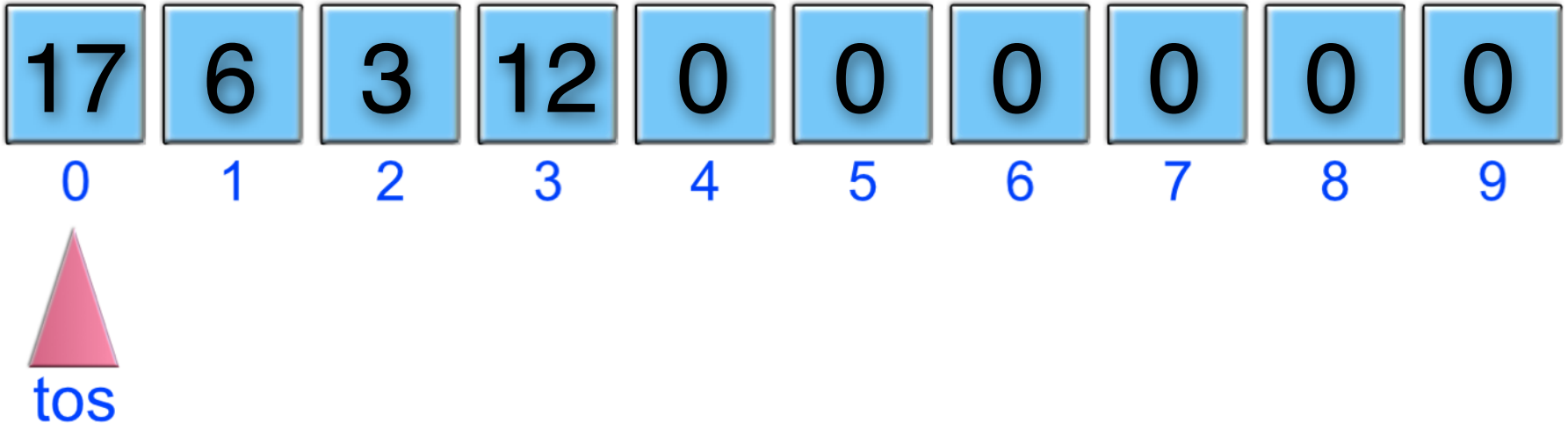
Schritt 12: Wir führen die Operation Pop() aus.



Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

Schritt 13: Wir führen die Operation Empty() aus.



false

Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

Schritt 14: Wir führen die Operation Pop() aus.

17	6	3	12	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

Schritt 14: Wir führen die Operation Empty() aus.

17	6	3	12	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

true

Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

Schritt 15: Wir führen die Operation Top() aus.

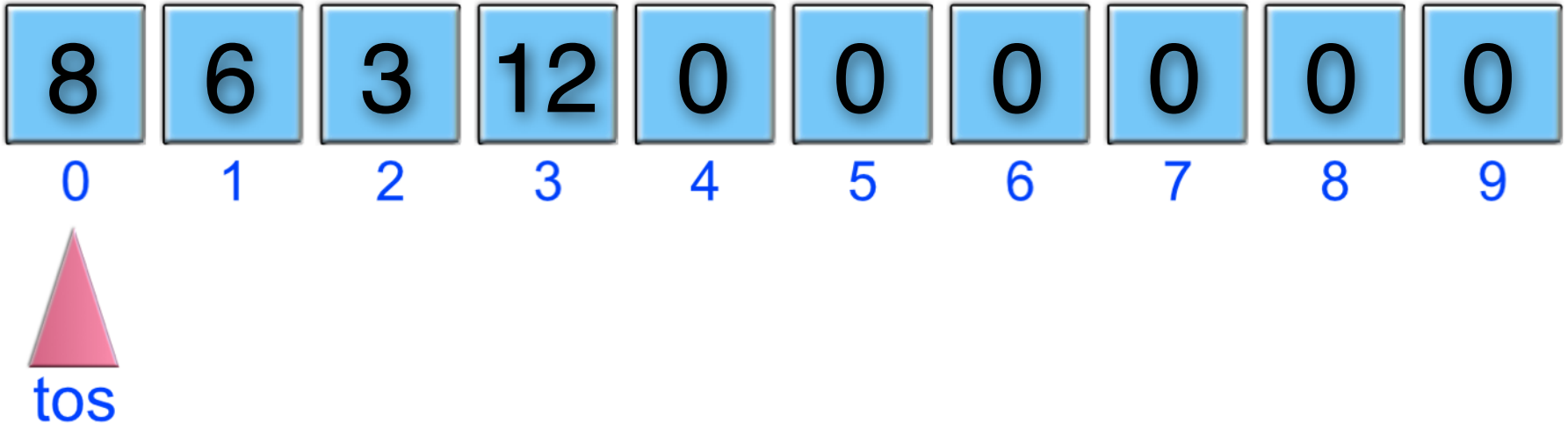
17	6	3	12	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

error

Stack-Implementation mit einem Array

Demonstration der Arbeitsweise eines Array-Stacks.

Schritt 15: Wir führen die Operation Push(8) aus.



Stack-Implementation mit einem Array

Erzeugung einer Java-Klasse Stack.
Das Attribut liste stellt den eigentlichen Stack dar.

```
public class Stack
{
   private int[] liste;
   private int   tos;

    public Stack()
   {
      liste = new int[10];
      tos   = -1;
   }
```

Stack-Implementation mit einem Array

Die push()-Methode

```
public void push(int x)
{
   if (tos < 10)
     liste[++tos] = x;
}
```

Stack-Implementation mit einem Array

Die push()-Methode

```
public void push(int x)
{
   if (tos < 10)
     liste[++tos] = x;
}
```

tos wird zuerst um 1 erhöht

Stack-Implementation mit einem Array

Die push()-Methode

```
public void push(int x)
{
   if (tos < 10)
     liste[++tos] = x;
}
```

danach wird x gespeichert

Stack-Implementation mit einem Array

Die pop()-Methode

```
public void pop()
{
   if (!empty()) tos--;
}
```

tos wird einfach dekrementiert

Stack-Implementation mit einem Array

Die top()-Methode

```
public int top()
{
   if (!empty())
      return liste[tos];
   else
      return -99;
}
```

Rückgabe eines error-Wertes

Stack-Implementation mit einem Array

Die empty()-Methode

```
public boolean empty()
{
    return (tos == -1);
}
```

Stack-Implementation mit einem Array

```
public class Stack
{
       private int[] liste;
       private int   tos;

       public Stack()
       {
          liste = new int[10];
          tos   = -1;
       }

       public void push(int x)
       {
          if (tos < 10)
            liste[++tos] = x;
       }

       public void pop()
       {
          if (! empty()) tos--;
       }

       public int top()
       {
          if (! empty())
             return liste[tos];
          else
             return -1;
       }

       public boolean empty()
       {
          return (tos == -1);
       }
}
```

Objekt-Arrays

Objekt-Arrays enthalten keine primitiven Datentypen (wie int), sondern Zeiger auf Objekte.

Objektstack-Implementation mit einem Array

Erzeugung einer Java-Klasse Stack.
Das Attribut liste stellt den eigentlichen Stack dar.

```
public class Stack
{
    private Object[] liste;
    private int tos;

    public Stack()
    {
        liste = new Object[100];
        tos = -1;
    }
```

Fast die gleiche Implementation wie int-Stack, nur Object[] statt int[].

Objektstack-Implementation mit einem Array

Implementation der Operation push()

```
public void push(Object x)
{
   if (tos < 99)
   {
      tos++;
      liste[tos] = x;
   }
}
```

Fast die gleiche Implementation wie int-Stack, nur Object statt int.

Objektstack-Implementation mit einem Array

Implementation der Operation pop()

```
public void pop()
{
    if (!empty()) tos--;
}
```

Exakt die gleiche Implementation wie int-Stack!

Objektstack-Implementation mit einem Array

Implementation der Operation top()

```
public Object top()
{
    if (!empty())
        return liste[tos];
    else
        return null;
}
```

Fast die gleiche Implementation wie int-Stack, nur Object statt int.

return null statt return -99 oder error!

Objektstack-Implementation mit einem Array

Implementation der Operation empty()

```
public Object top()
{
    if (!empty())
        return liste[tos];
    else
        return null;
}
```

Fast die gleiche Implementation wie int-Stack, nur Object statt int.

return null statt return -99 oder error!

Objektstack-Implementation mit einem Array

Anwendungsbeispiel für einen Objektstack

```
public class Test
{
    Stack s;

    public Test()
    {
       s = new Stack();
       s.push(new Bruch(3,4));
       s.push(new Bruch(5,8));
       s.push(new Bruch(1,7));
    }
    Bruch oben = (Bruch) s.top();
}
```

Objektstack-Implementation mit einem Array

Anwendungsbeispiel für einen Objektstack

```
public class Test
{
    Stack s;

    public Test()
    {
        s = new Stack();
        s.push(new Bruch(3,4));
        s.push(new Bruch(5,8));
        s.push(new Bruch(1,7));
    }
    Bruch oben = (Bruch) s.top();
}
```

Typecasting erforderlich, da im Stack nur Zeiger auf Instanzen der Klasse Object gespeichert sind.

Objektstack-Implementation mit einem Array

Weiteres Anwendungsbeispiel für einen Objektstack

```
Stack rucksack;
...
rucksack.push(new Gegenstand("Schwert",12,3);
rucksack.push(new Gegenstand("Messer",7,2);
rucksack.push(new Gegenstand("Dolch",9,3);
rucksack.sort();
waffe = rucksack.top();
rucksack.pop();
```

Objektstack-Implementation mit einem Array

Weiteres Anwendungsbeispiel für einen Objektstack

```
Stack rucksack;
...
rucksack.push(new Gegenstand("Schwert",12,3);
rucksack.push(new Gegenstand("Messer",7,2);
rucksack.push(new Gegenstand("Dolch",9,3);
rucksack.sort();
waffe = rucksack.top();
rucksack.pop();
```

Objekte werden direkt beim Aufruf von push() "an Ort und Stelle" erzeugt mit **new**...

Objektstack-Implementation mit einem Array

Implementation von push() aus der Klausuraufgabe

```
public void push(Gegenstand neu)
{
   if (anzahl >= 12) return; // Inventar voll

   anzahl++;
   ding[anzahl] = neu;
}
```

Objektstack-Implementation mit einem Array

Implementation von pop() aus der Klausuraufgabe

```
public void pop()
{
    if (!empty()) anzahl--;
}
```

Objektstack-Implementation mit einem Array

Implementation von top() aus der Klausuraufgabe

```
public Gegenstand top()
{
    if (!empty())
        return ding[anzahl];
    else
        return null;
}
```