Algorithms and Data Structures
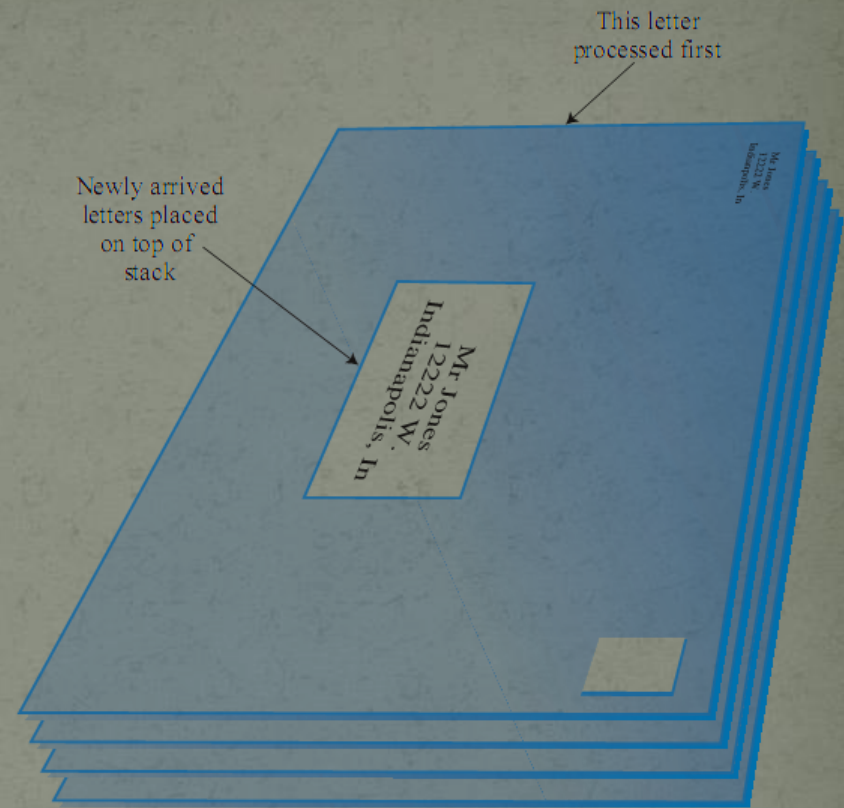
Andrzej Pisarski

Plan of the lecture

- Abstract Data Types (ADS)
- Stacks
- Queues
- Priority Queues
- Analysis of arithmetic expressions: checking brackets.

Abstract Data Types (ADS)

- Stacks and Queues:
  - Programmer's tools
    - conceptual aids than data storage objects (usually created for particular task for some function)
  - Restricted access to data
    - Restricted access is enforced by interface (access to other items like in array are not allowed)
  - ADS
    - Data type defined by their interface (inside it can be implement by different kind of data structures like arrays, linked list, etc.)

3

Stacks

- A stacks allows access to only one data item: the last item inserted.

- LIFO – last item inserted is first to be removed.

- *Pushing it* – placing a data item on the top of the stack.

- *Popping it* – removing from top of the stack.

This letter processed first

Newly arrived letters placed on top of stack

Mr Jones
12222 W.
Indianapolis, In

(*example from R. Lafore book*)

4

Stacks

Efficiency of Stacks
   Items can be pushed or popped in O(1) time (do not depend how many items are in the stack).

Queues

- FIFO – first item inserted is first to be removed.
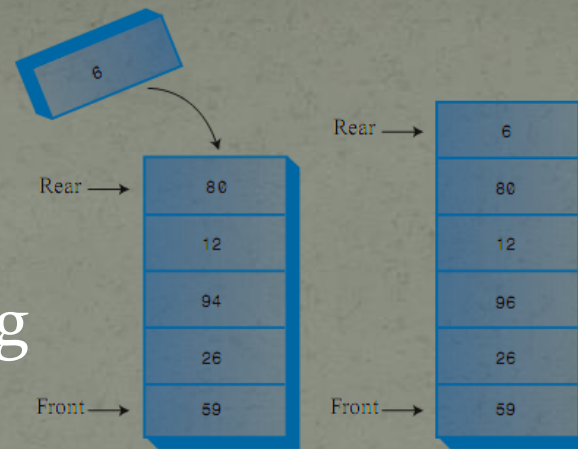
People join the
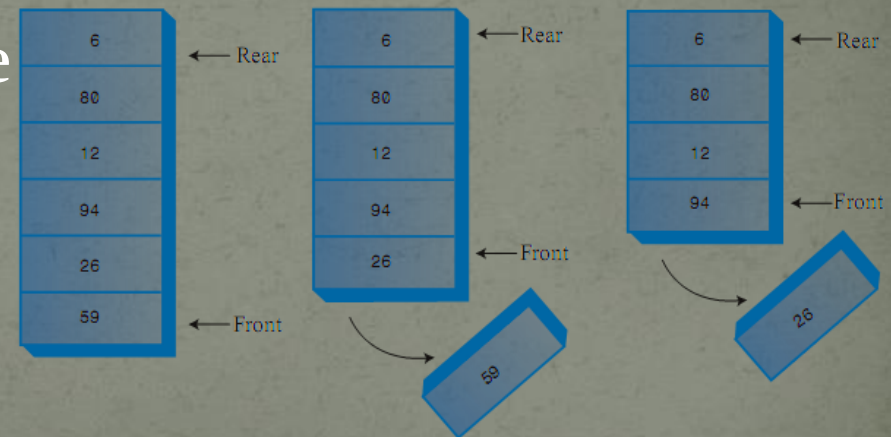queue at the rear

People leave the
queue at the front

(*example from R. Lafore book*)

Queues

- FIFO – first item inserted is first to be removed.
- *Push (pushing)* – inserting *(put* or *add* or *enque),*
- *Pop (popping)* – removing *(delete* or *get* or *de-que),*
- Rear (of the queue) – place where items are inserted *(back* or *tail* or *end),*
- Front – place where items are removed *(head).*

Rear → | 6 |

Rear → | 80 |
| 12 |
| 94 |
| 26 |
Front → | 59 |

Rear → | 6 |
| 80 |
| 12 |
| 96 |
| 26 |
Front → | 59 |

New item inserted at rear of queue

| 6 | ← Rear
| 80 |
| 12 |
| 94 |
| 26 |
| 59 | ← Front

| 6 | ← Rear
| 80 |
| 12 |
| 94 |
| 26 | ← Front

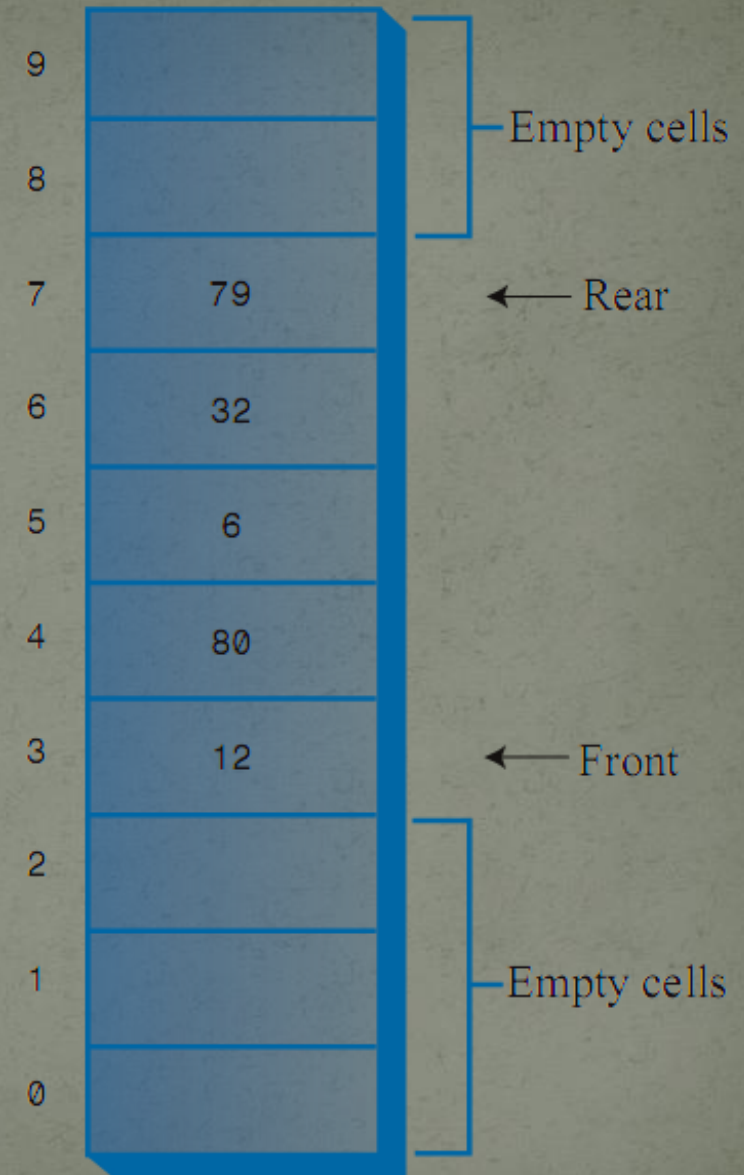| 59 |

| 6 | ← Rear
| 80 |
| 12 |
| 94 | ← Front

| 26 |

Two items removed from front of queue

(example from R. Lafore book)

7

Queues

- FIFO – first item inserted is first to be removed.
- *Push (pushing)* – inserting (*put* or *add* or *enque),*
- *Pop (popping)* – removing (*delete* or *get* or *de-que*),
- Rear (of the queue) – place where items are inserted (*back* or *tail* or *end*),
- Front – place where items are removed (*head*).

MaxSize-1 →

9	
8	
7	79
6	32
5	6
4	80
3	12
2	
1	
0	

Empty cells

← Rear

← Front

Empty cells

(example from R. Lafore book)

8

Queues

- FIFO – first item inserted is first to be removed.
- *Push* (*pushing*) – inserting (*put* or *add* or *enque*),
- *Pop* (*popping*) – removing (*delete* or *get* or *de-que*),
- Rear (of the queue) – place where items are inserted (*back* or *tail* or *end*),
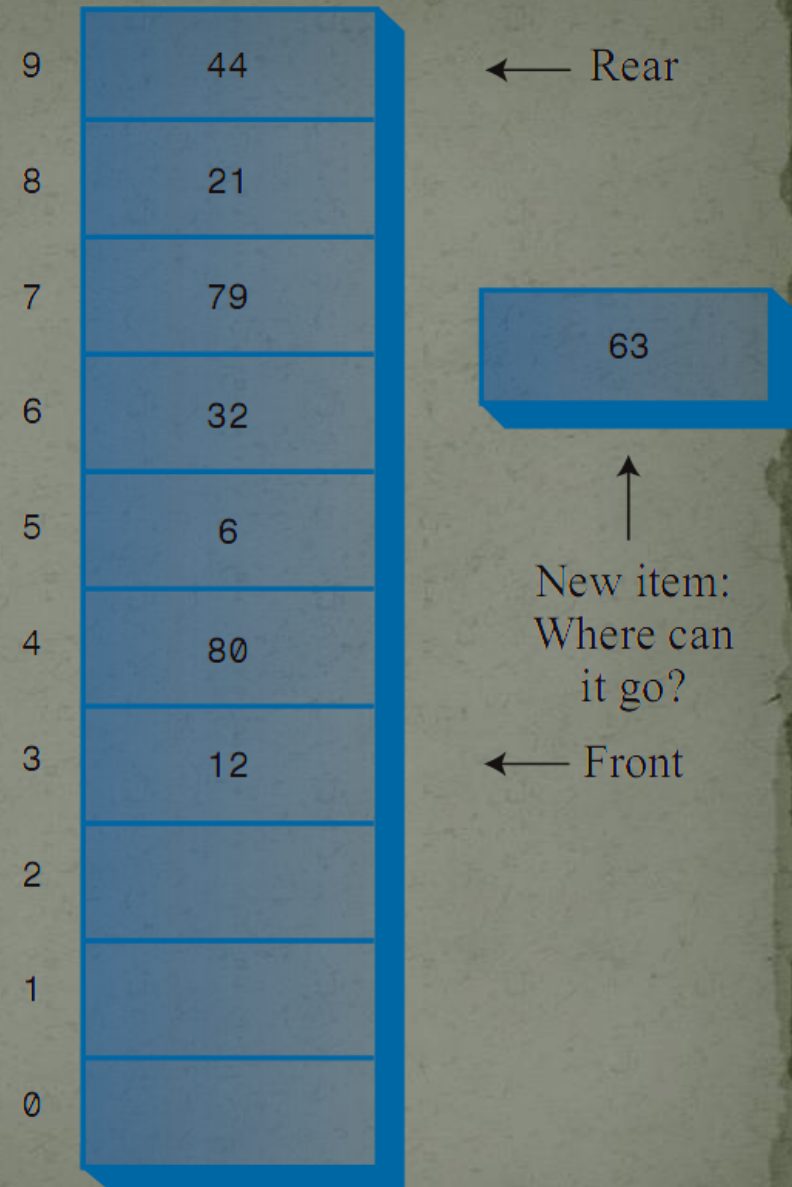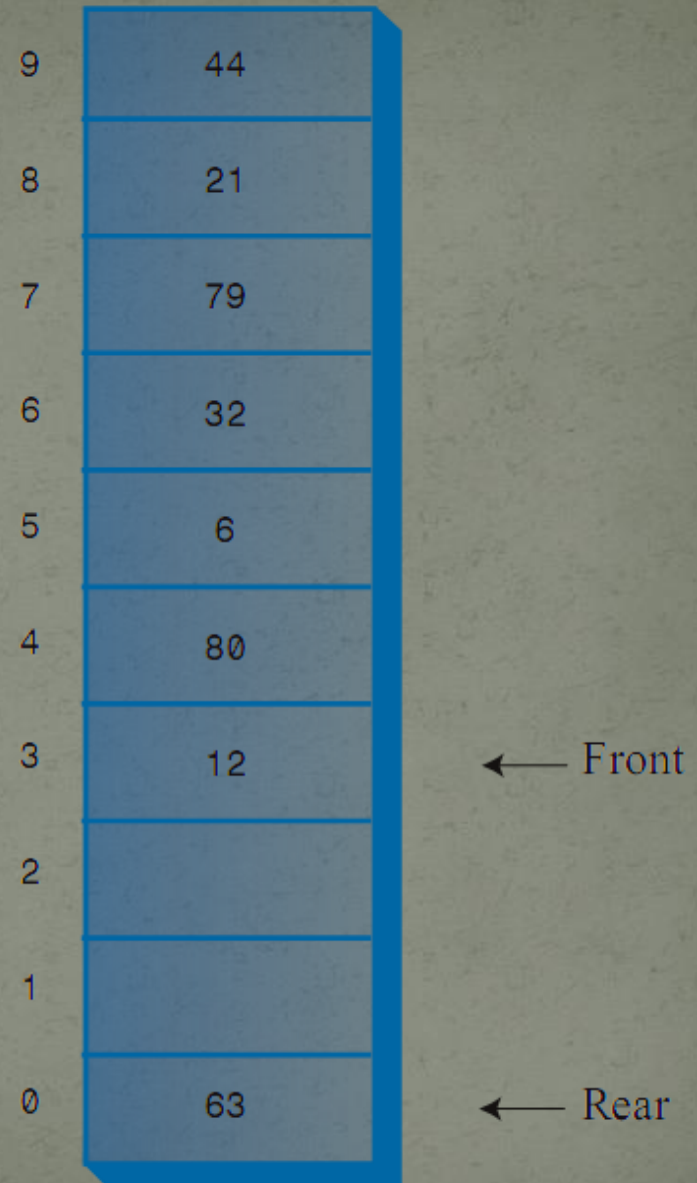- Front – place where items are removed (*head*).

MaxSize-1 ⟶

9	44	⟵ Rear
8	21	
7	79	
6	32	
5	6	
4	80	
3	12	⟵ Front
2		
1		
0		

63

New item: Where can it go?

(*example from R. Lafore book*)

9

Queues

- FIFO – first item inserted is first to be removed.
- *Push* *(pushing)* – inserting *(put* or *add* or *enque)*,
- *Pop* *(popping)* – removing *(delete* or *get* or *de-que)*,
- Rear (of the queue) – place where items are inserted *(back* or *tail* or *end)*,
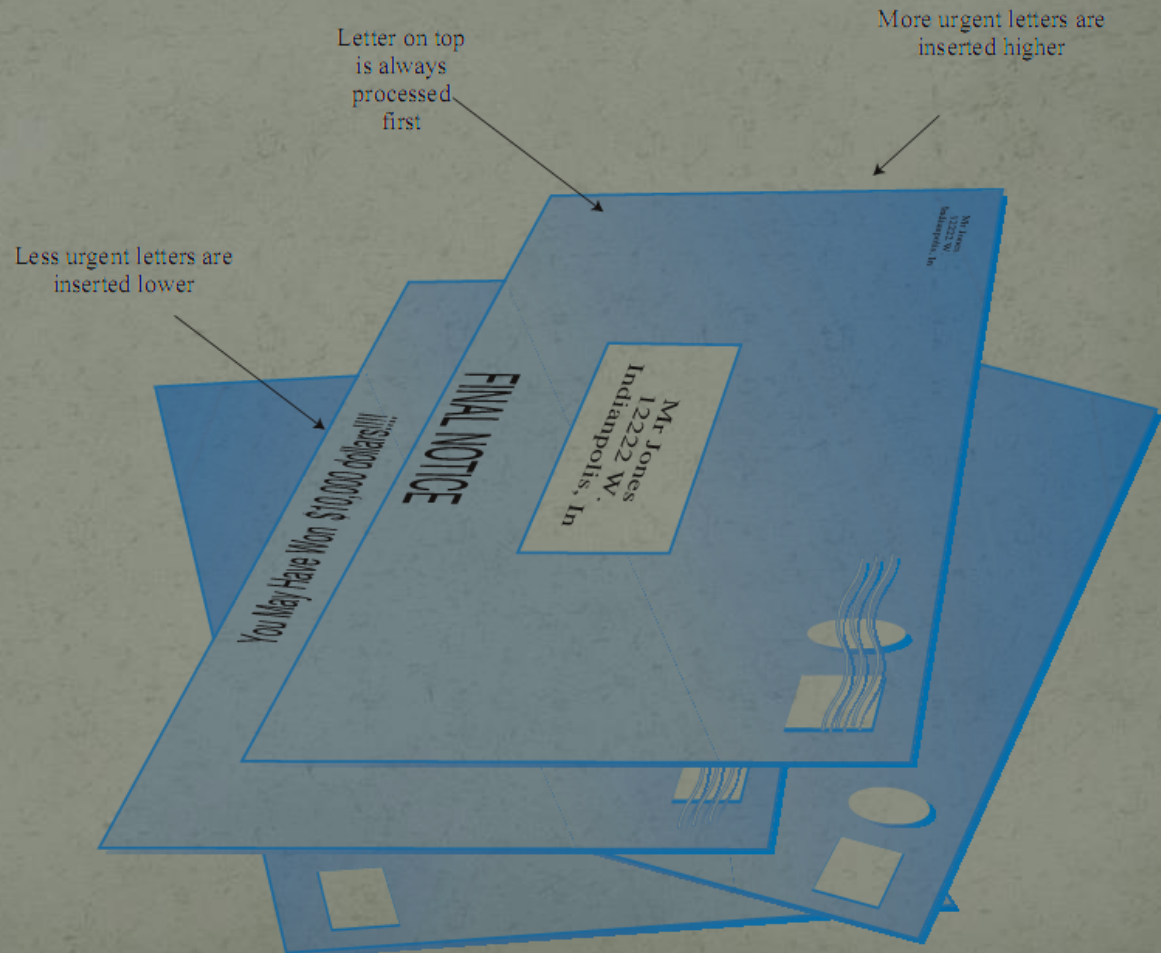- Front – place where items are removed *(head)*.

MaxSize-1 →

9	44
8	21
7	79
6	32
5	6
4	80
3	12	← Front
2	
1	
0	63	← Rear

(example from R. Lafore book)

10

Priority Queues

(example from R. Lafore book)

More urgent letters are inserted higher

Letter on top is always processed first

Less urgent letters are inserted lower

FINAL NOTICE

You May Have Won $10,000.00!!!

Mr Jones
12222 W.
Indianpolis, In

Priority Queues
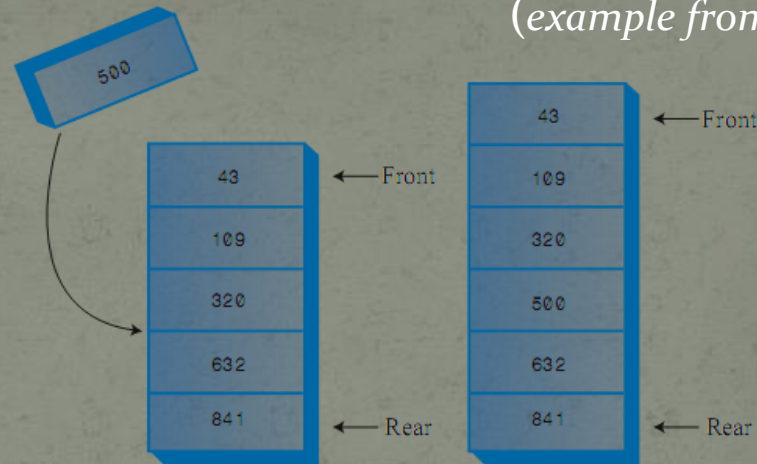
FIFO – first item inserted is first to be removed.
Push (pushing) – inserting *(put* or *add* or *enque),*
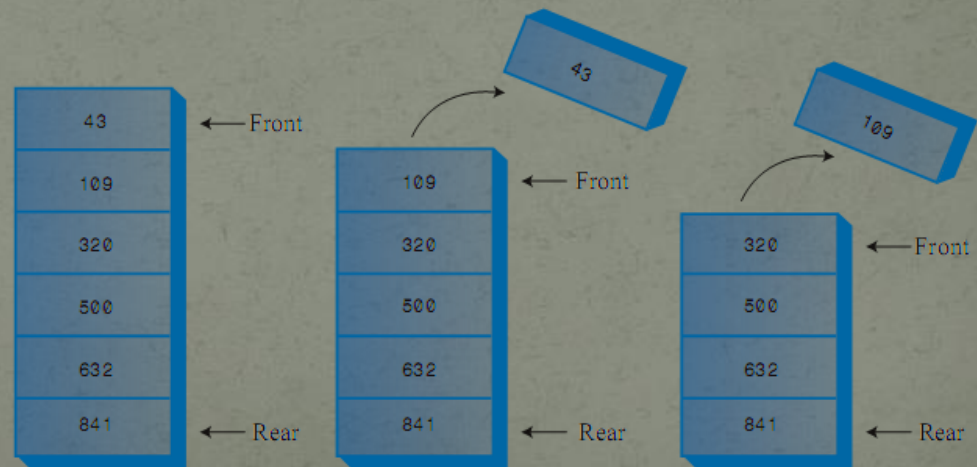*Pop (popping)* – removing *(delete* or *get* or *de-que),*
Rear (of the queue) – place where items are inserted *(back* or *tail* or *end),*
Front – place where items are removed *(head).*

New item inserted in priority queue

Two items removed from front of priority queue

12

Efficiency of Queues
Items can be pushed or popped in O(1) time (do not depend how many items are in the stack).

Analysis of arithmetic expressions: checking brackets.

- Program that checks the delimiters order (braces „{" and „}", brackets „[" and „]" parentheses „(" and „)").
- Stacks can be used to parse certain text strings (it can be strings of code lines parsing by compilers).

Examples (*R. Lafore book*) :
```
c[d]           // correct
a{b[c]d}e   // correct
a{b(c]d}e   // not correct; ] doesn't match (
a[b{c]d]e}  // not correct; nothing matches final }
a{b{c        // not correct; nothing matches opening {
```

14

Analysis of arithmetic expressions: checking brackets.

Character Read	Stack Contents
a	
{	{
b	{
(	{ (
c	{ (
[	{ ([
d	{ ([
]	{ (
e	{ (
)	{
f	{
}	

(example from R. Lafore book)