Data Structures

Linked list, Queue, Stack

Definitions

Data structures – a combination of simple data types and structuration methods.

Standard operations on structured data types:

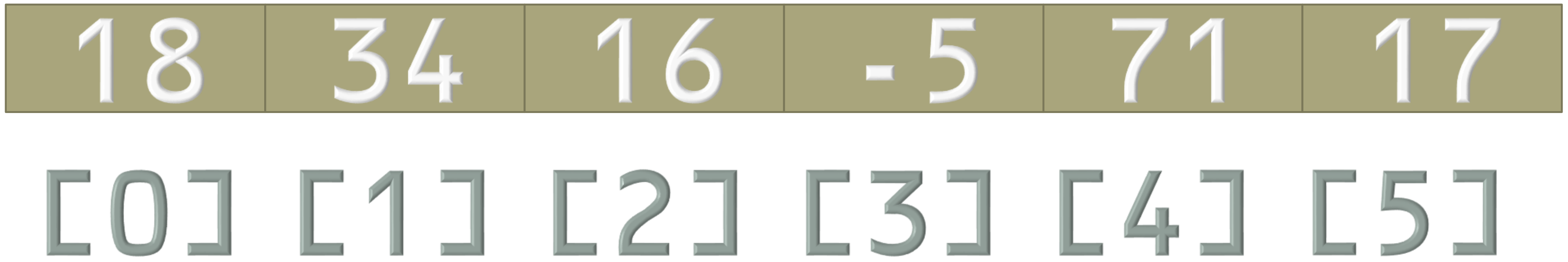
- 1. Definition
- 2. Initialization
- 3. Adding elements
- 3. Remove elements
- 4. Passing data structure

Types of linear data structures

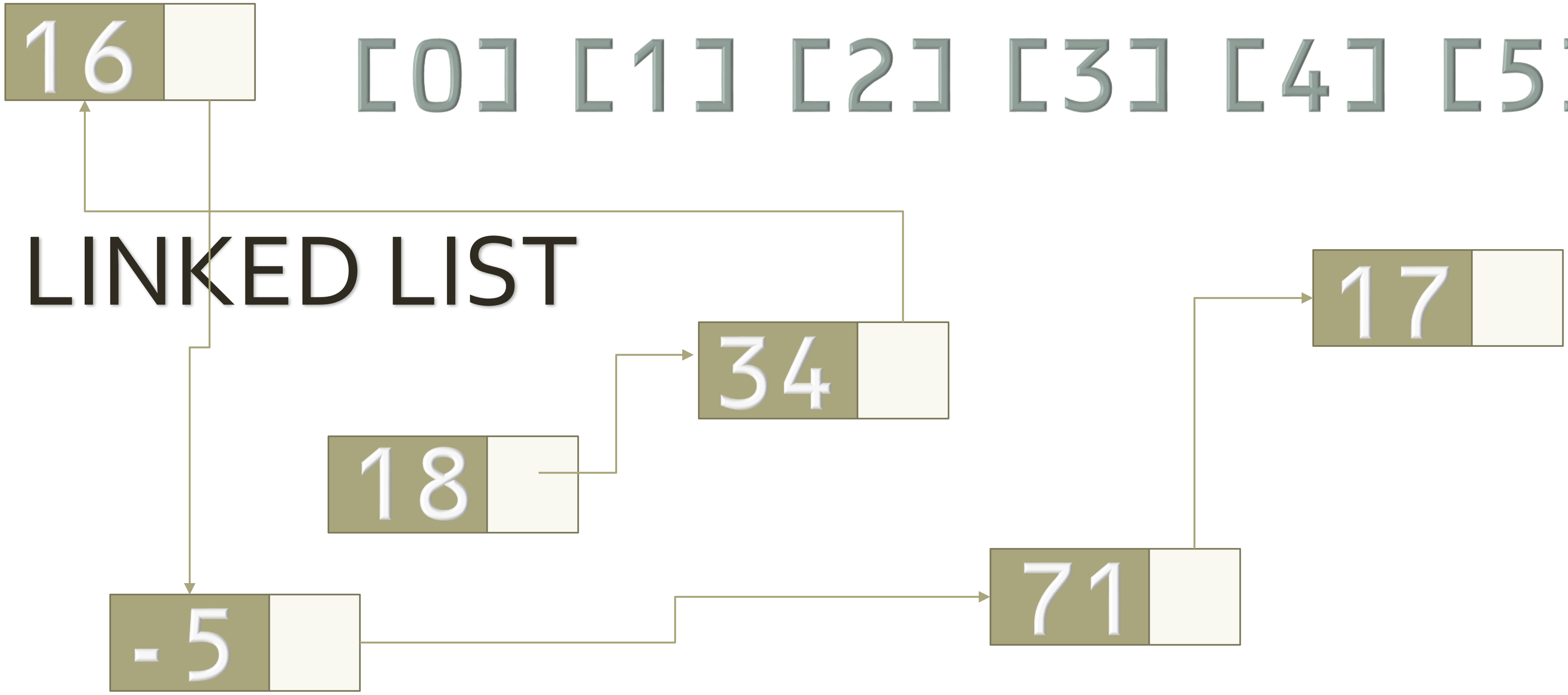
Indexed structures – arrays. Already discussed.

Linked Lists - linear collections of data elements, whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence.

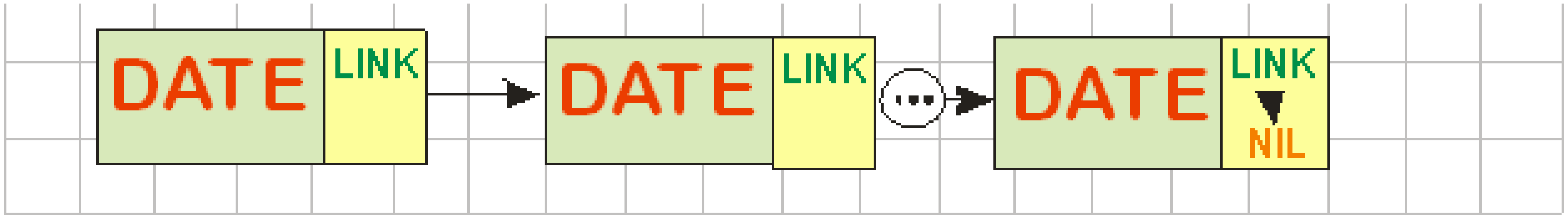
ARRAY



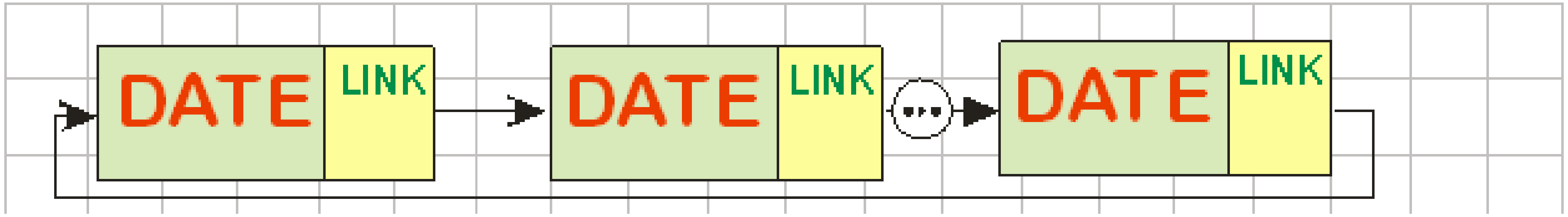
LINKED LIST



A

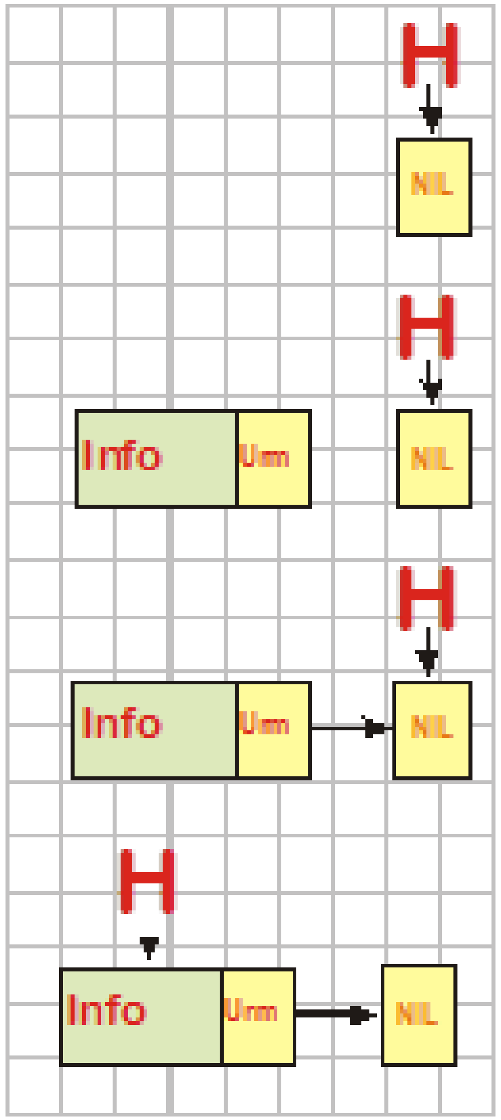


B

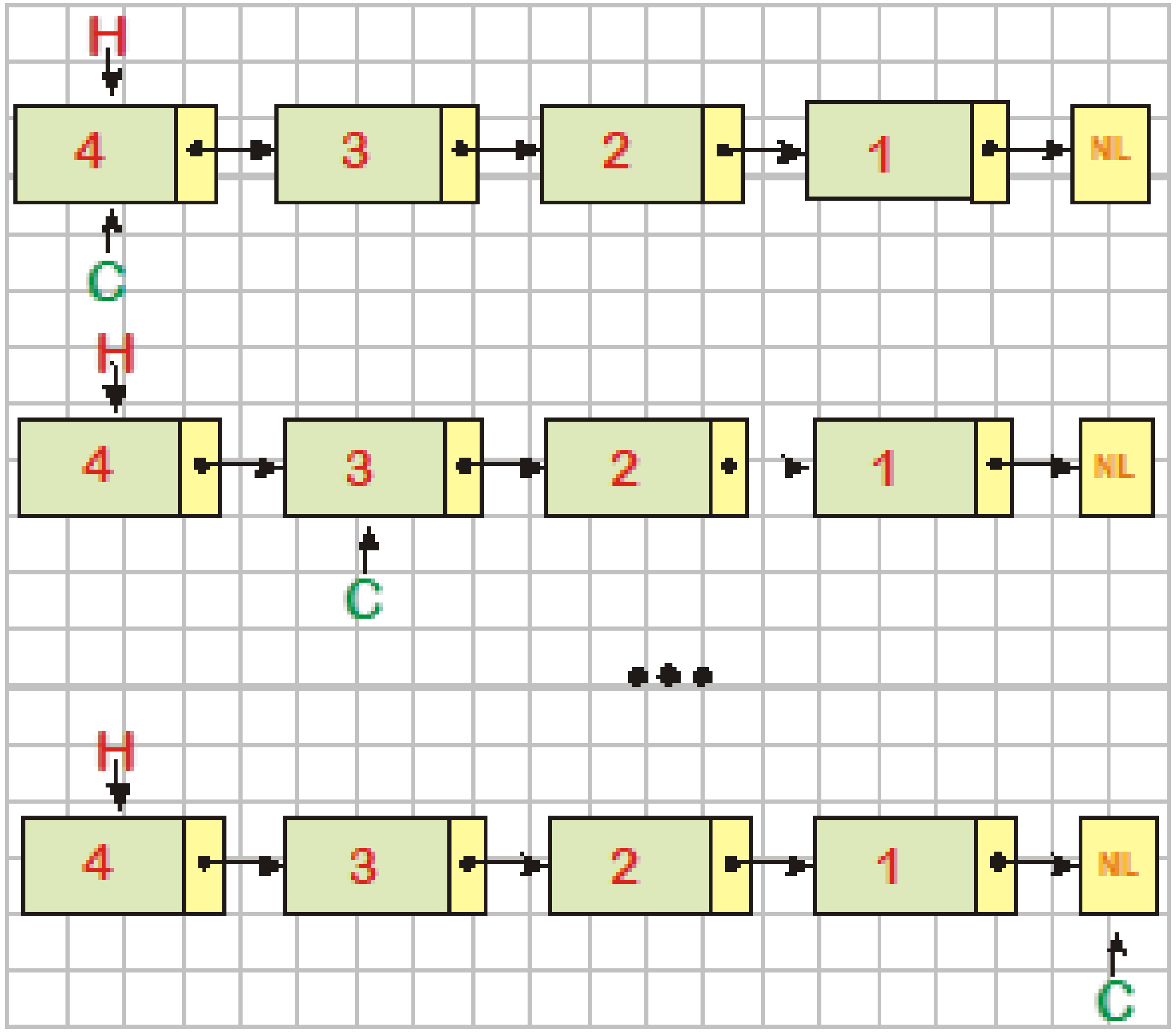


Types of lists: one-directional (A), circular (B)

Insertion

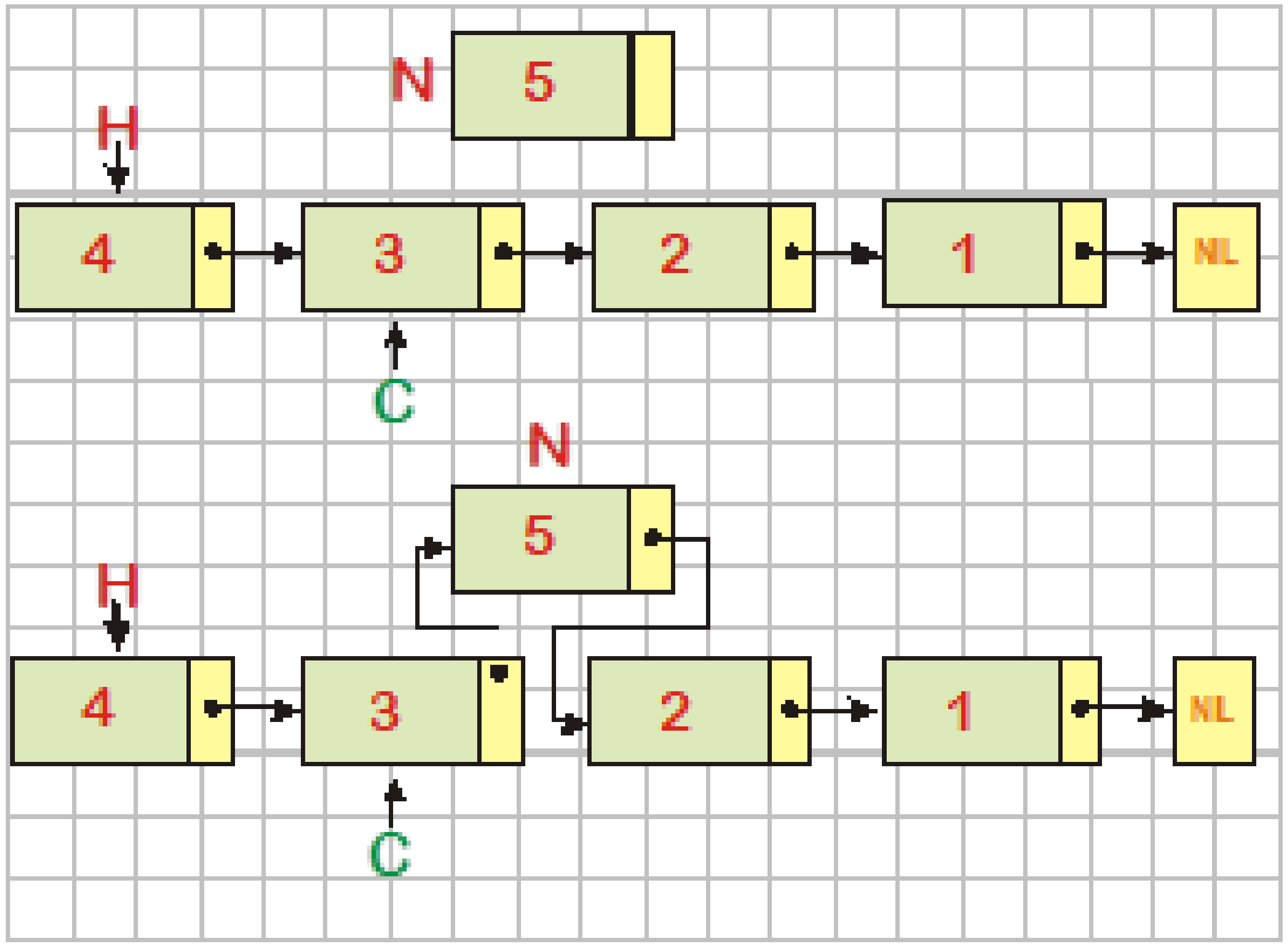


Passing the List

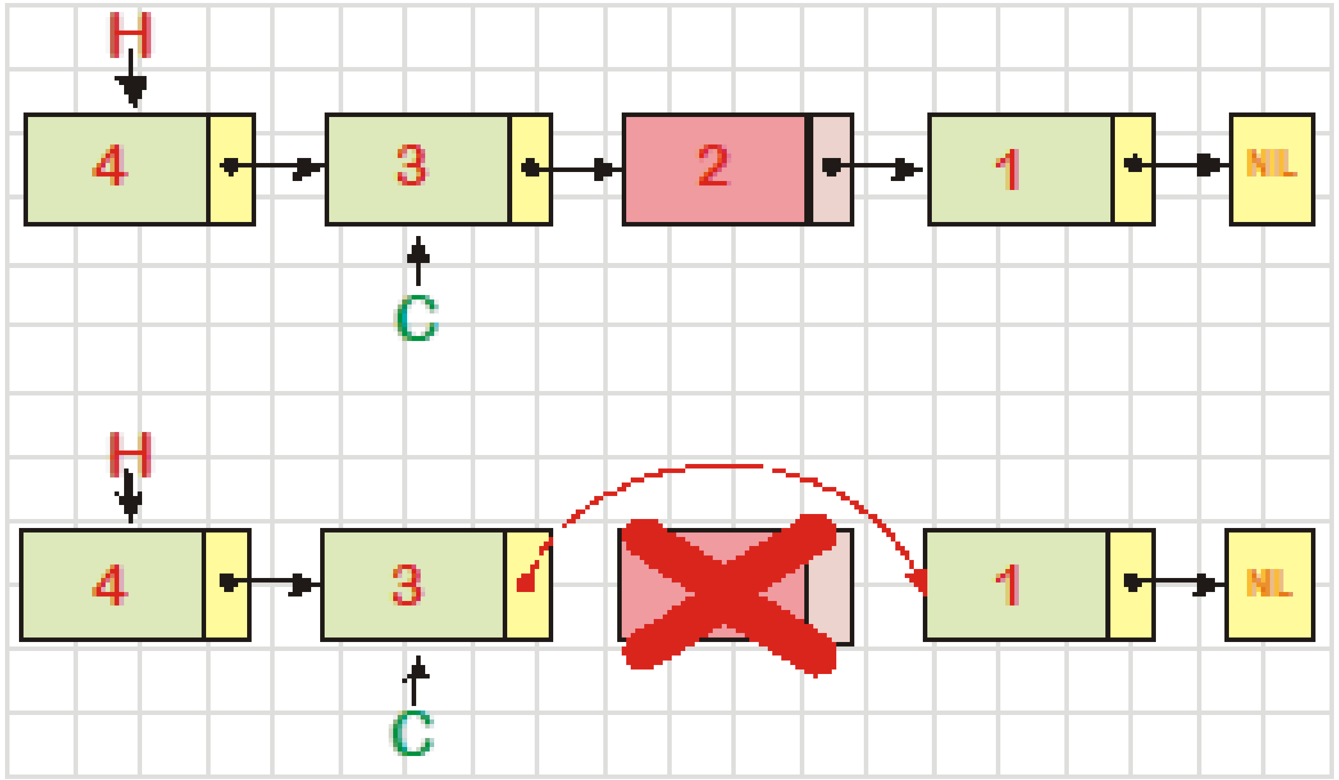


Add an element

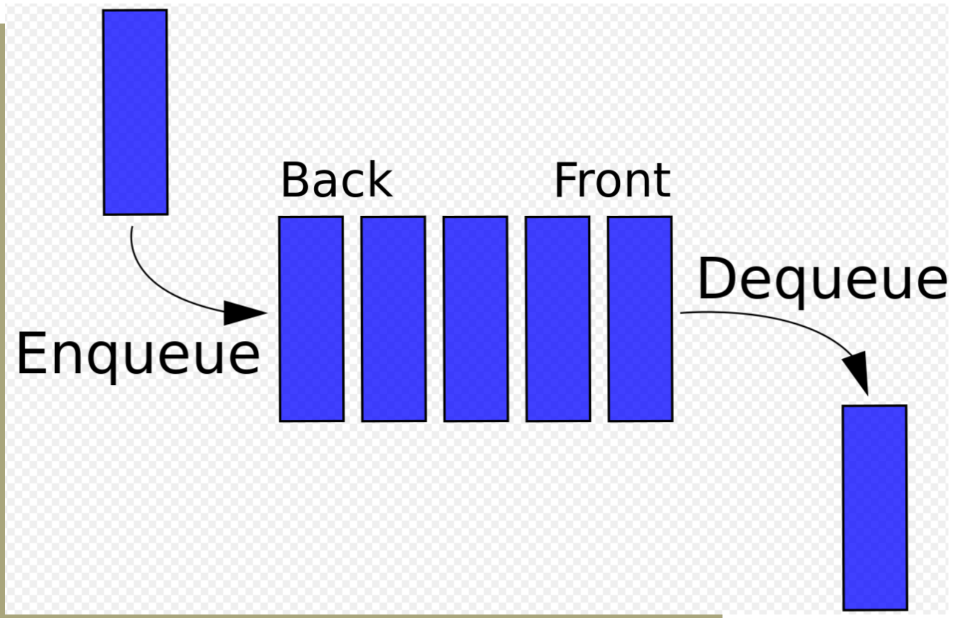
(after selected)



Remove element



QUEUE



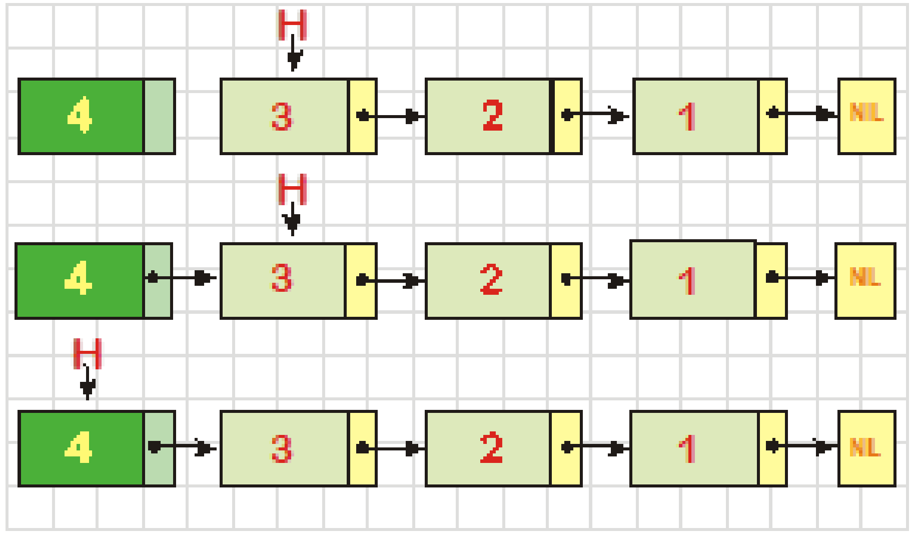
Definition

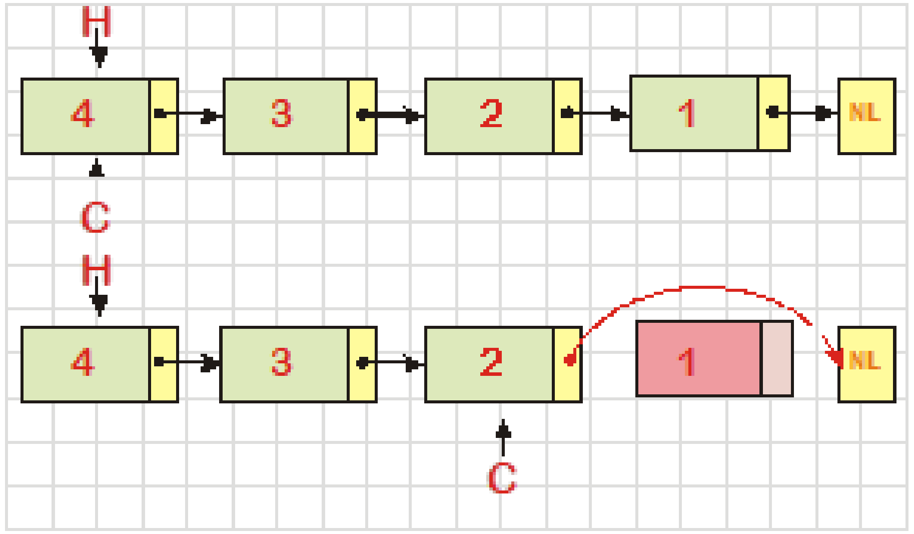
QUEUE from latin CAUDA (TAIL) – COADĂ

Queue is a collection in which the entities are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position, known as enqueue, and removal of entities from the front terminal position, known as dequeue. This makes the queue a **First-In-First-Out** (FIFO) data structure

QUEUE

ADD
/
REMOVE



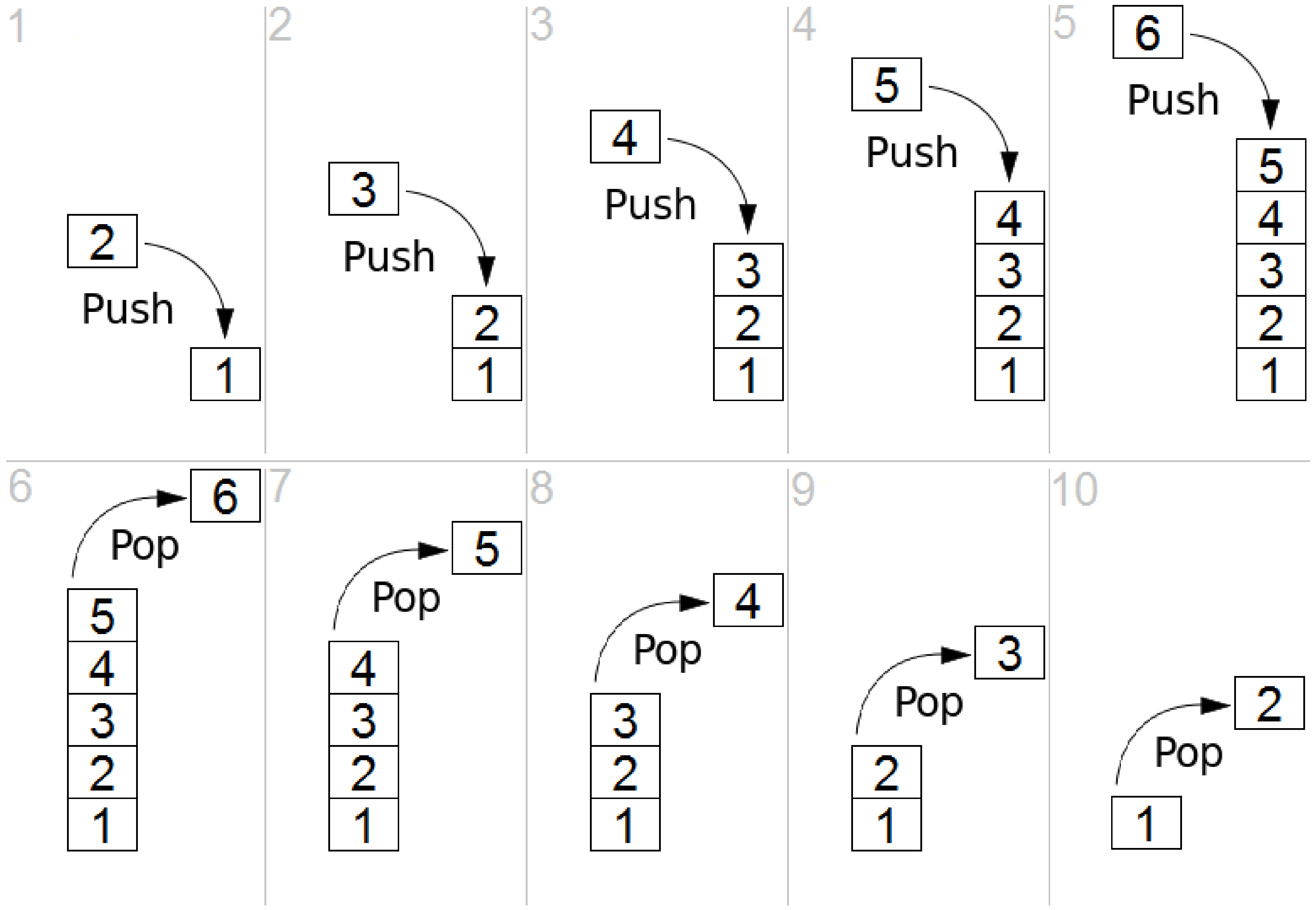


STACK

Definition

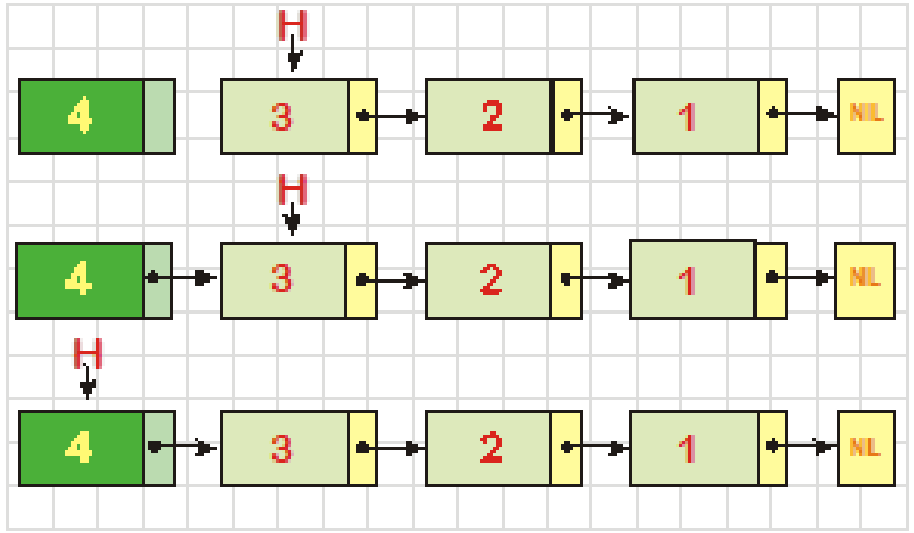
In computer science, a stack is an abstract data type that serves as a collection of elements, with two principal operations:

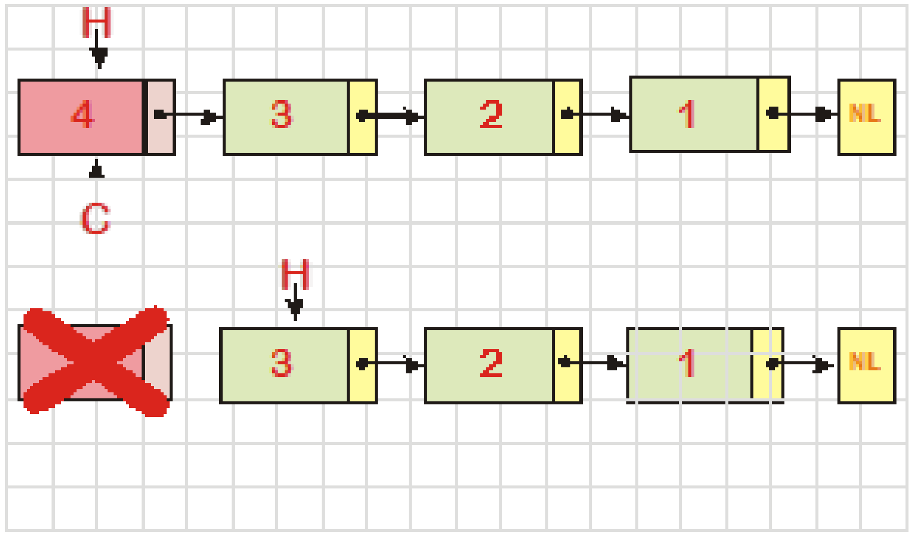
- **push,** which adds an element to the collection, and
- **pop,** which removes the most recently added element that was not yet removed.


1
2
Push
1
2
3
Push
2
1
3
4
Push
3
2
1
4
5
Push
4
3
2
1
5
6
Push
5
4
3
2
1
6
Pop
6
5
4
3
2
1
7
Pop
5
4
3
2
1
8
Pop
4
3
2
1
9
Pop
3
2
1
10
Pop
2
1

STACK

ADD / REMOVE

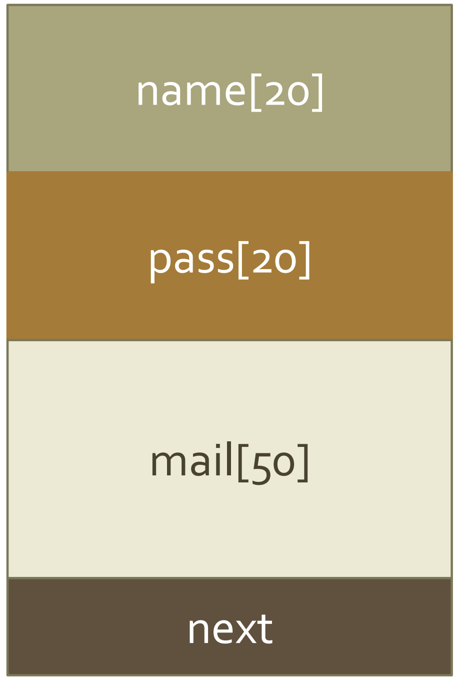




Code example

```
#include <stdio.h>
#include <stdlib.h>

struct User {
    char name[20];
    char pass[20];
    char mail[50];
    struct User *next;
};
struct User* head = NULL;
```



List (stack & queue) traversal

```c
void traverseList() {
    struct User* current = NULL;
    current = head;

    while (current != NULL) {
        printf("User: %s\n", current->name);
        printf("Password: %s\n", current->pass);
        printf("Email: %s\n\n", current->mail);

        current = current->next;
    }
}
```

Add element

```
void addToList(char *name, char *pass, char *mail) {
    struct User* item = NULL;
    item = (struct User*)malloc(sizeof(struct User));

    strcpy(item->name, name);
    strcpy(item->pass, pass);
    strcpy(item->mail, mail);

    if (head != NULL) {
        item->next = head;
    }
    else {
        item->next = NULL;
    }
    head = item;
}
```

Remove from stack

```
void removeFromListStack() {
    struct User* current = NULL;
    struct User* temp = NULL;

    if (head == NULL) {
        printf("No elements in list");
        return;
    }

    current = head;

    if (current != NULL) {
        head = head->next;
        free(current);
    }
}
```

Remove from queue

```
void removeFromListQueue() {
    struct User* current = NULL;
    struct User* temp = NULL;

    if (head == NULL) {
        printf("No elements in list");
        return;
    }

    if (head->next == NULL) {
        head = NULL;
        free(head);
        return;
    }

    current = head;
    while(current->next->next != NULL) current = current->next;


    temp = current->next;
    current->next = NULL;
    free(temp);
}
```

The entire program is attached

Be carefull with Stack and Queue!

STACK GUN & QUEUE GUN