First Assignment

Data structures

Alejandro Mart’in Sanchez

Carlos Garrido Juncos

Index

Analysis

- ADTs specification and operations

- Make a program to generate random data > 100.000
- Random data generate by the user
- Menu with several options
- Must use linear data structure
    - Stacks
    - Queues
    - Lists

Analysis

```
spec QUEUE [INTEGER]
 genre queue, integer
  operations
            empty : queue → bool
            makenull: queue → queue
            front: queue → integer
            enqueue: queue, integer → queue
            dequeue: queue → integer
endspec
```

```
spec STACK [INTEGER]
 genre stack, integer
  operations
            empty: stack → bool
             makenull: stack→ stack
            top: stack→ integer
            push: stack, integer → stack
            pop: stack → integer
endspe
```

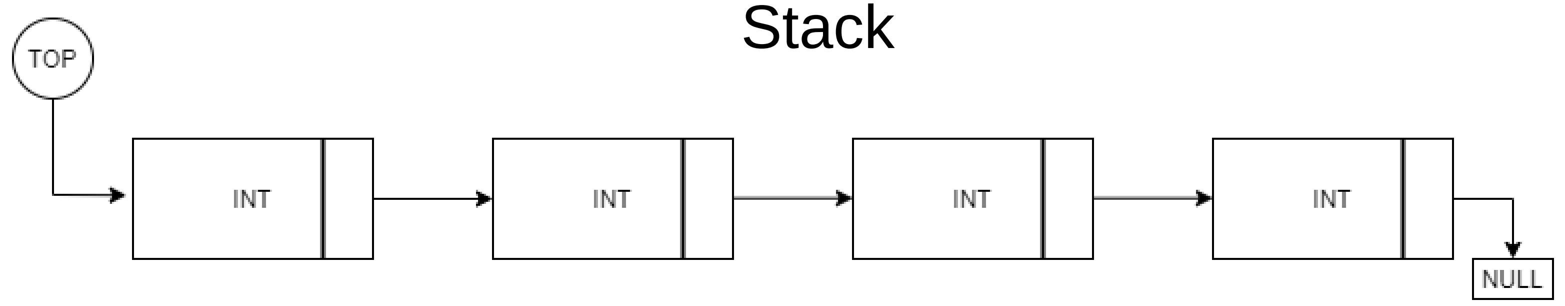
Analysis

```
spec LIST [INTEGER]
 genre list, integer
  operations
           empty : list → bool
           makenull: list → list
           front: queue → integer
           insert: list, integer → list
           delete: list → integer
           listlegnth: list → integer
endspec
```

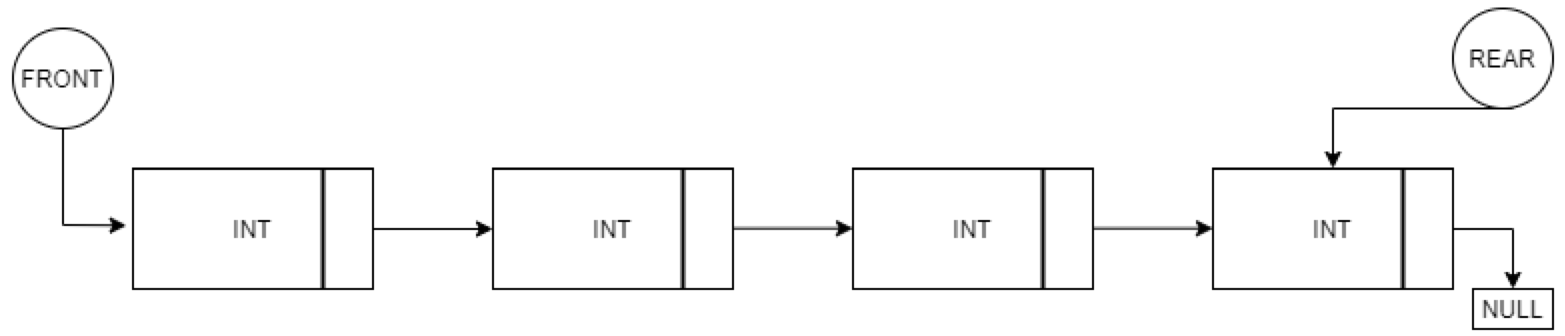
Design

- BOX DIAGRAM

Stack



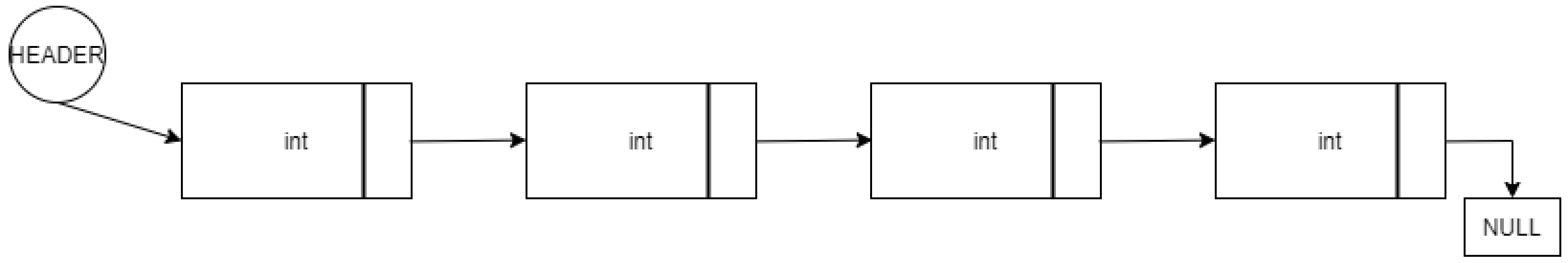
Queue



Design

- Box Diagram

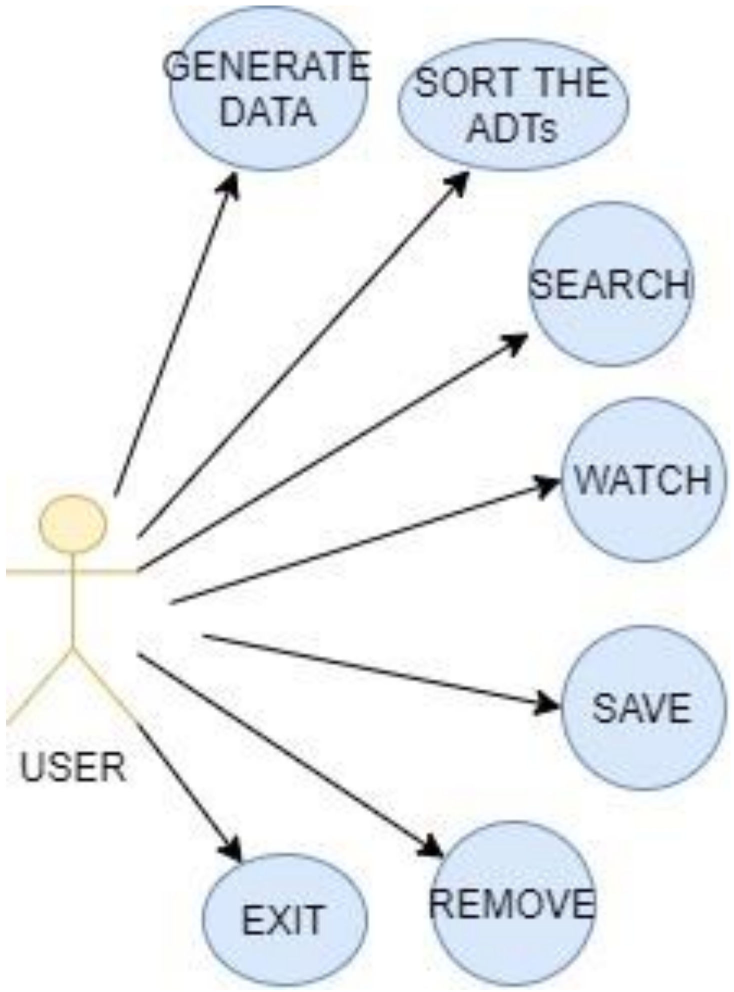
List



Design

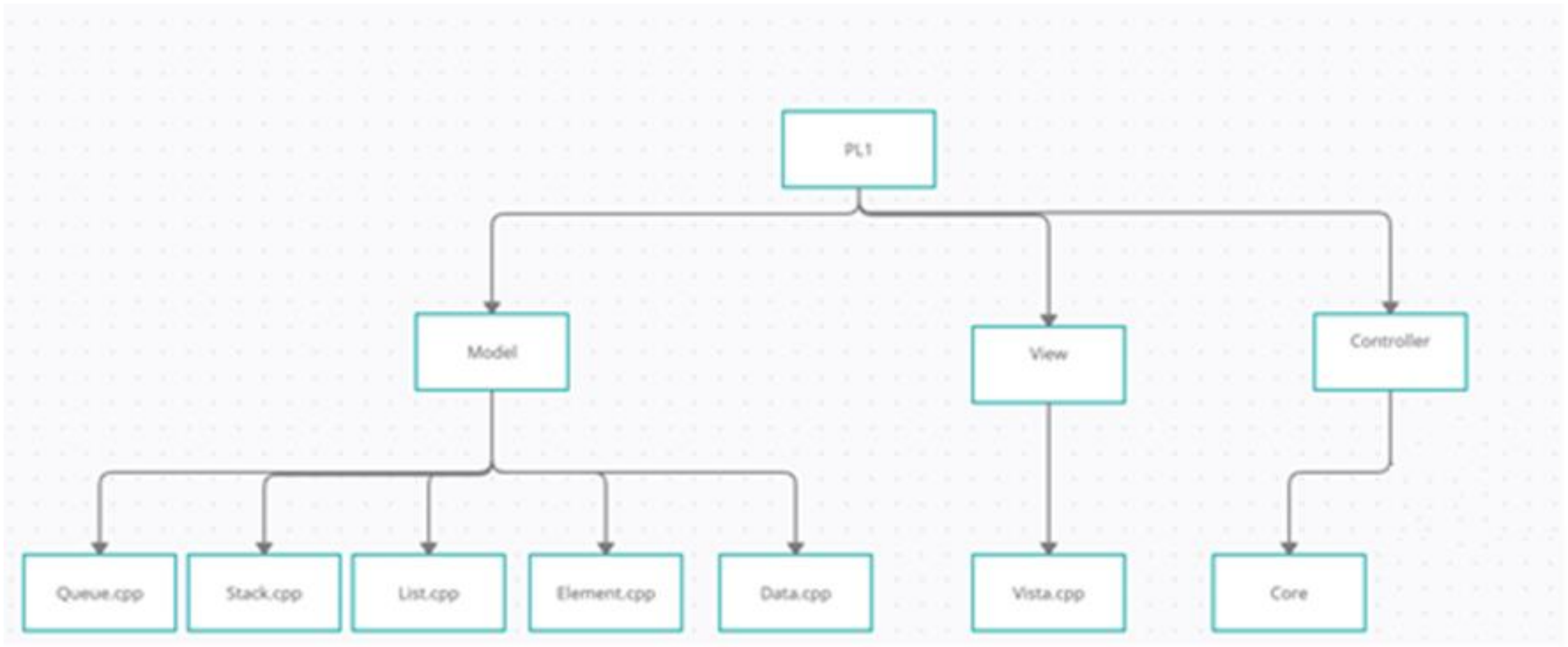
Universidad de Alcalá

- Use case diagram



Design

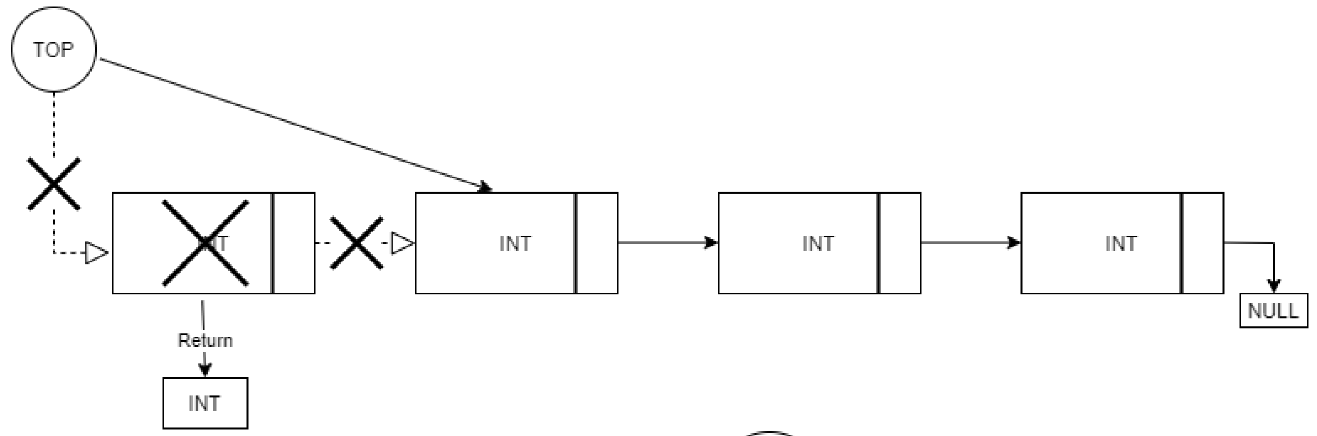
- Class Diagram



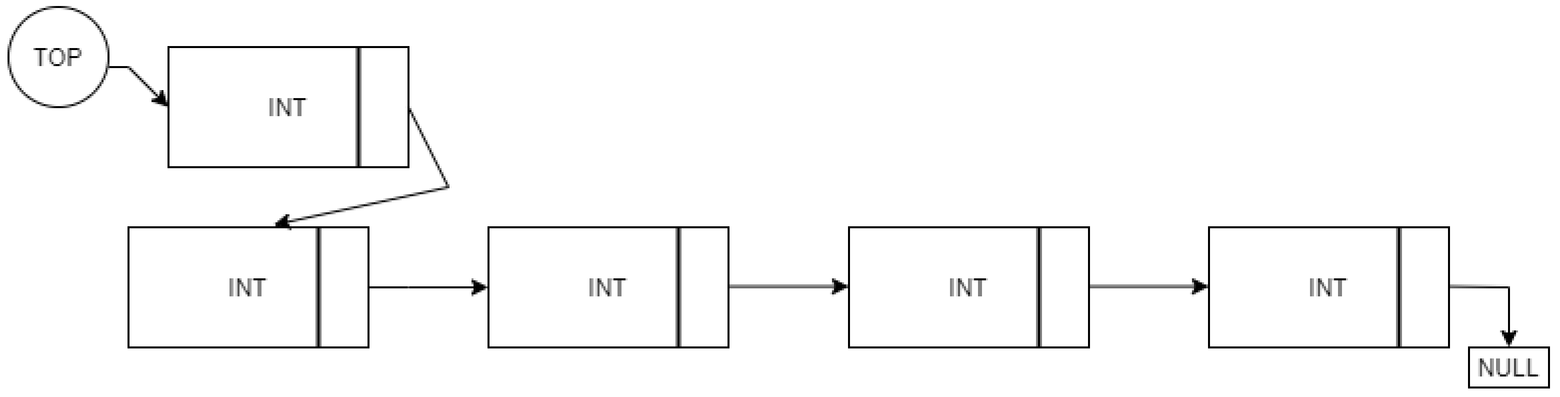
Design

Universidad de Alcalá

- Explanation ADTs - Stack

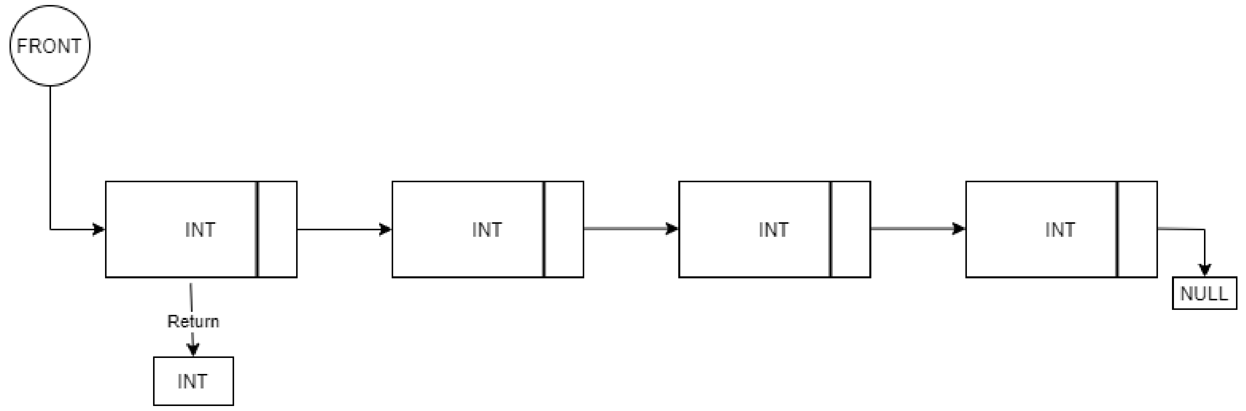


Pop(): return the value of the element pointed by top and delete the element

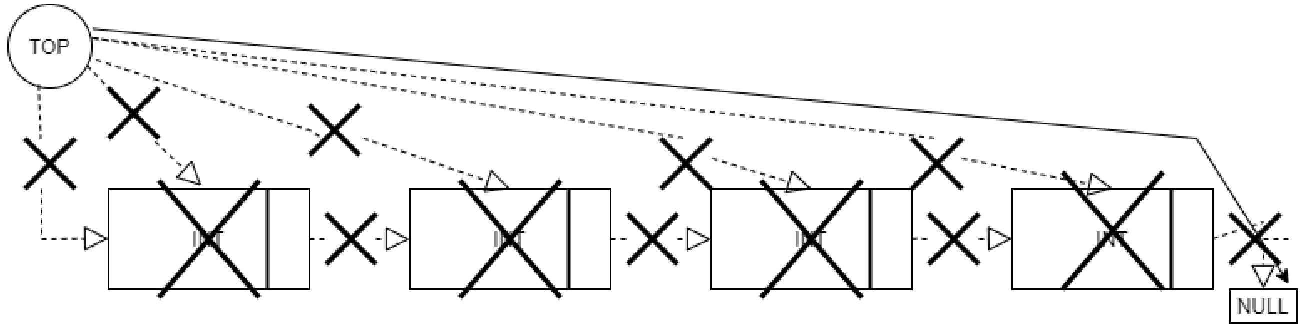


Push(): Insert a new element at the top of the stack

Design



Top(); return the value of the element pointed by top without deleting the element

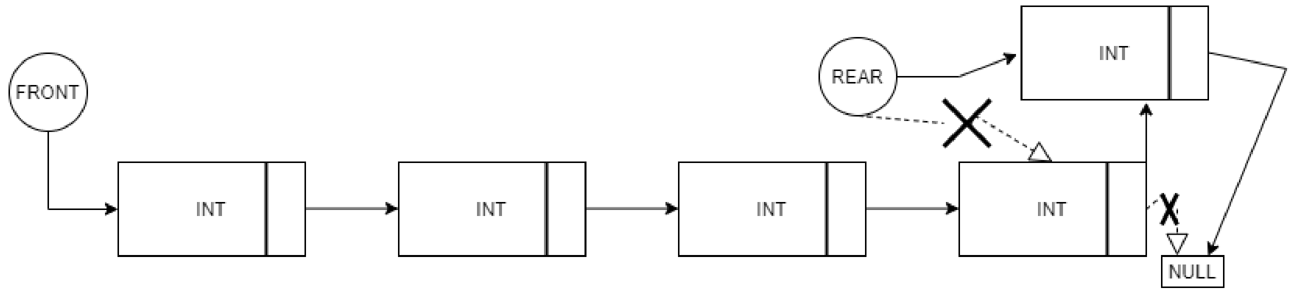


Makenull(): Empties the stack

Eempty(): return if top equals to null

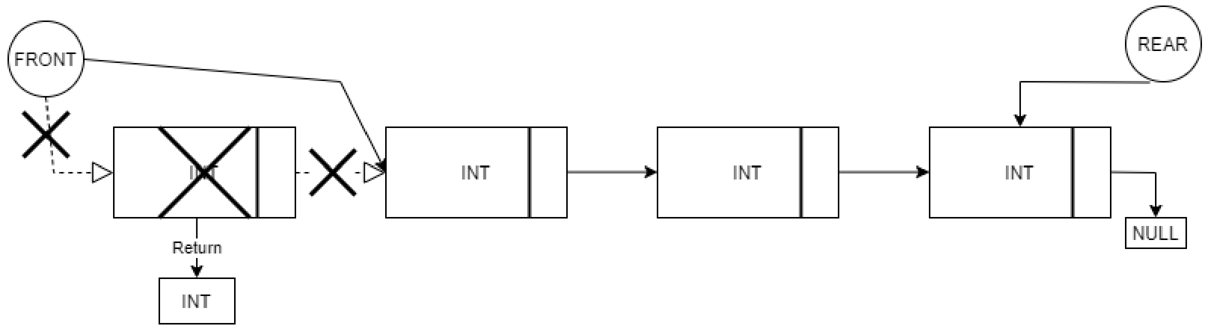
Design

- Explanation ADTs - Queues

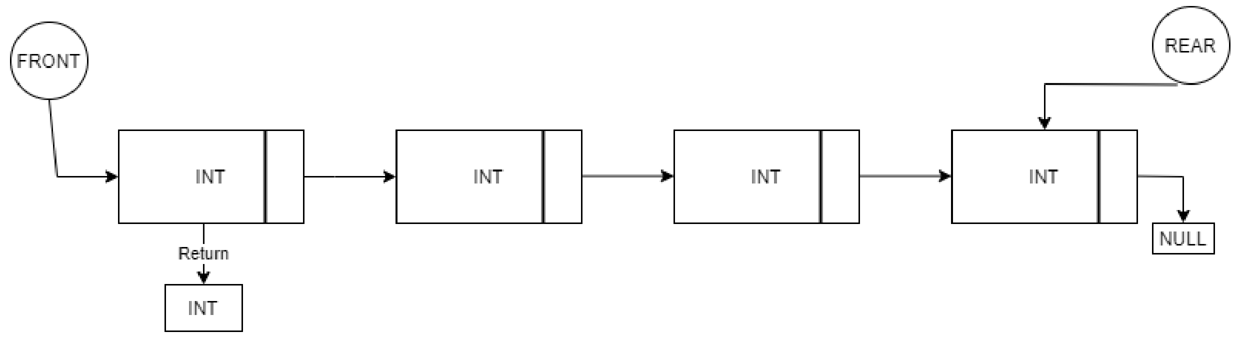


Enqueue(): Insert a new element at the rear of the queue

Dequeue(): return the value of the element pointed by front and delete the element



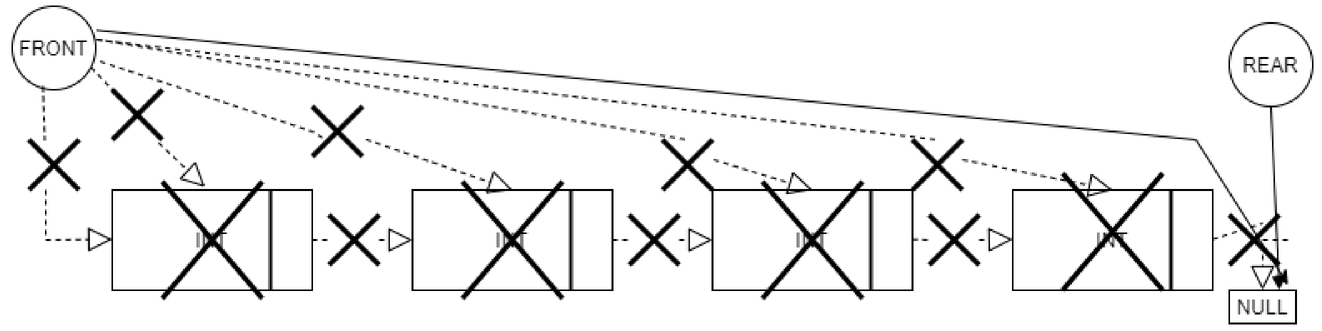
Design



Front(): return the value of the element pointed by front without deleting the element

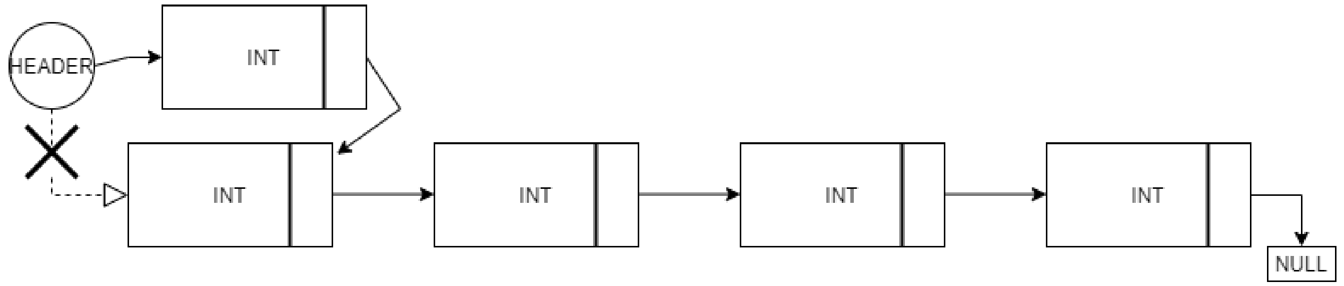
Makenull(): Empties the queue

Eempty(): return if front equals null and rear equals to null



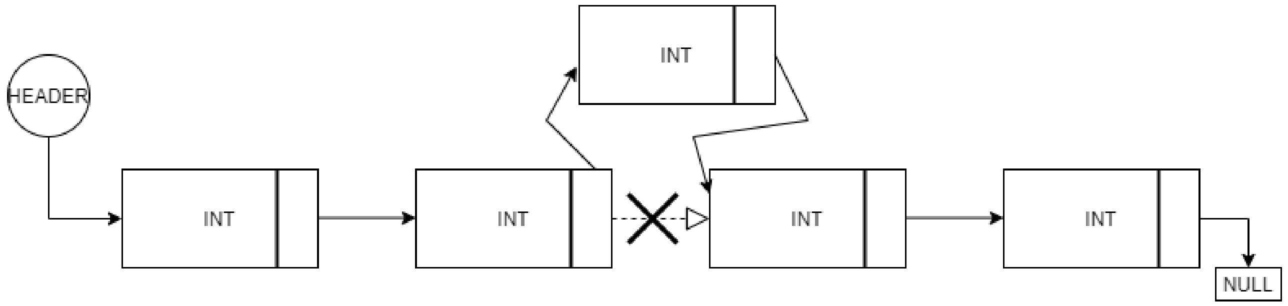
Design

- Explanation ADTs - Lists

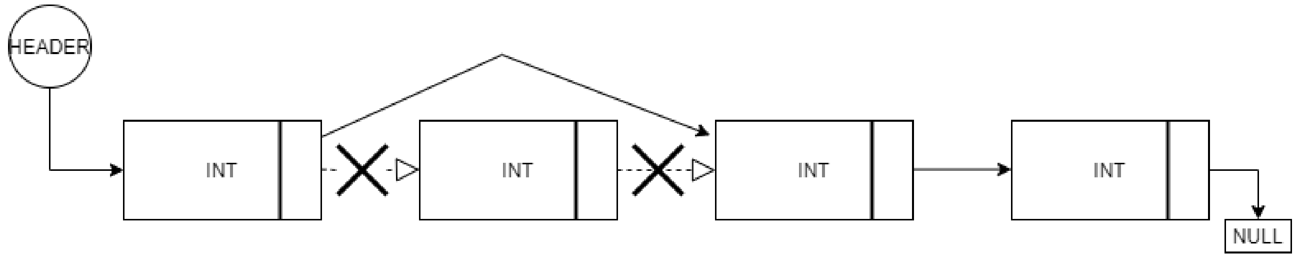


Insert(): Insert a new element at the head of the list

InsertPosition(): Insert a new element at a position on the list

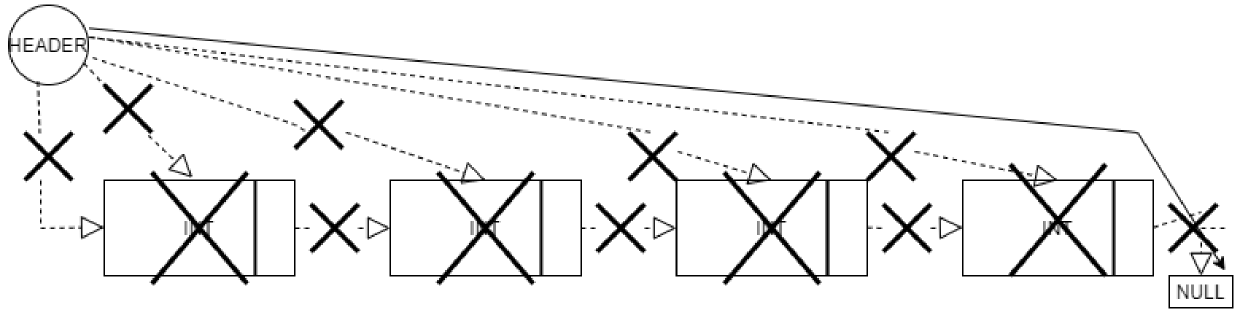


Design



Delete(): return the value of the element in the position to delete

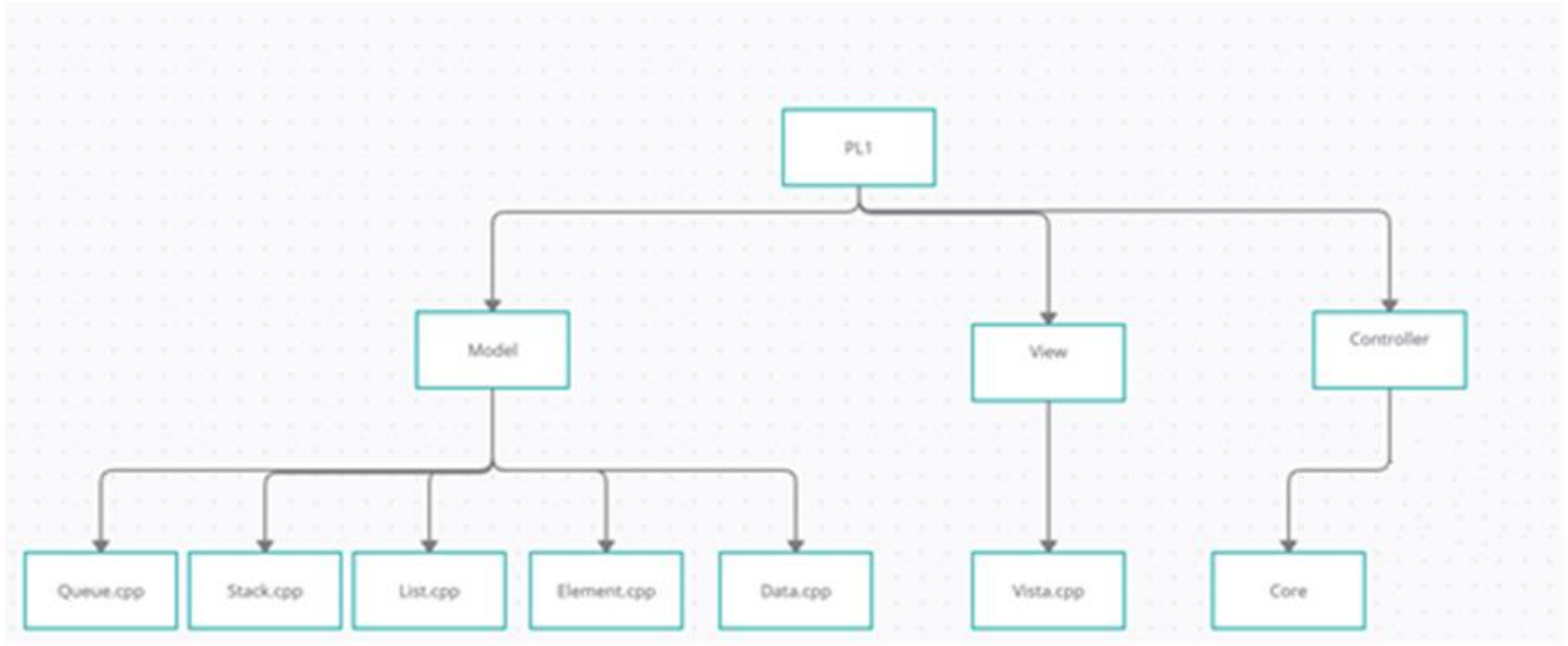
Makenull(): Empties the list



Implementation

- Behavior of the program
    1. Generate data number
    2. Display the menu.
        1. Sort the ADTs
        2. Search Options
        3. Watch Data
        4. Remove Data
        5. Add Data
        6. Save File
        7. Exit

Implementation



Main: invoke MainInterface

Vista: interface

Core: alogic

Model: All the ADTs and the element

Review

- Running Time
- Stack.cpp:
    - Push() -> O(1)
    - Pop() -> O(1)
    - Empty() -> O(1)
    - getTop() -> O(1)
    - makeNull() -> O(n)
    - DeleteStack() -> O(n)
- Queue.cpp:
    - Enqueue() -> O(1)
    - Dequeue() -> O(1)
    - Front() -> O(1)
    - Qempty -> O(1)
    - Makenull() -> O(n)
    - QueueDelete() -> O(n)

Review

- List.cpp:
  - Insert() -> O(1)
  - Concatenate -> O(n)
  - ListLenght -> O(n)
  - ListDelete -> O(n)
  - Makenull() -> O(n)
  - Lempty() -> O(1)
  - PositionInsert() -> O(n)
  - ListDeletetePosition -> O(n)

- Core.cpp
  - Generation() -> O(n)
  - SortStack() -> O(n^2)
  - SortQueue() -> O(n^2)
  - QuickSortList() -> O(n log n)
  - sequentiaSearchStackHigherNumber -> O(n)
  - sequentiaSearchQueueHigherNumber -> O(n)
  - sequentiaSearchQueueHigherNumber -> O(n)
  - sequentiaSearchStack100Number -> O(n^2)
  - sequentiaSearchQueue100Number -> O(n^2)
  - sequentiaSearchList100Number -> O(n^2)

END