بسم الله الرحمن الرحيم

الموضوع: Data structure التاريخ: / /

It is a container of data used to store and organize data. Its type depends on how it is been arranged on the computer

Data structure

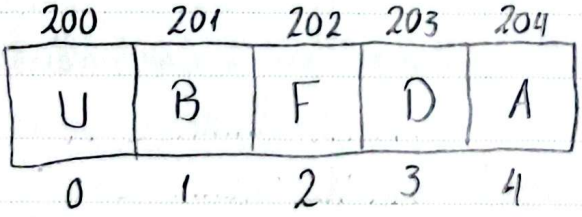
- Linear data structure (البيانات تخزن بالتتابع / جنب بعض)
  - Static data structure
    - Arrays
      - fixed size
  - Dynamic data structure
    - Stack
    - Stack
    - Linked list
      - size can by randomly updated during the runtime
- Non-Linear data structure (ترتيب)
  - Tree
  - Graph

---

* Arrays: container of same data types arranged sequentially on computer

⇒ has fixed size

⇒ It supports random access [O(1)]



* arr[0] = u ← how to access elements in array

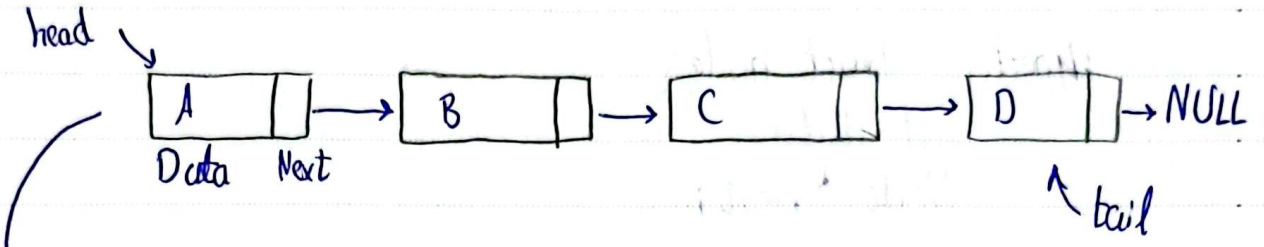
SENA

بسم الله الرحمن الرحيم

Data structure

التاريخ: / / الموضوع:

* Linked lists : elements in it are linked using pointers



→ Node : stores data and the address of the next node

1 : data : it holds the actual data or data associated with node

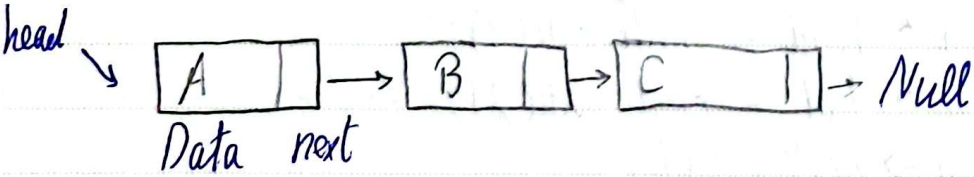
2 : Next pointer or reference : It stores the address of the next node in sequence

Why linked lists not Arrays?

→ because of the ease of insertion or delection in a linked list

Types of Linked lists :

① Singly linked lists : Simplest form in which every node contains some data and a pointer to the next node



بسم الله الرحمن الرحيم

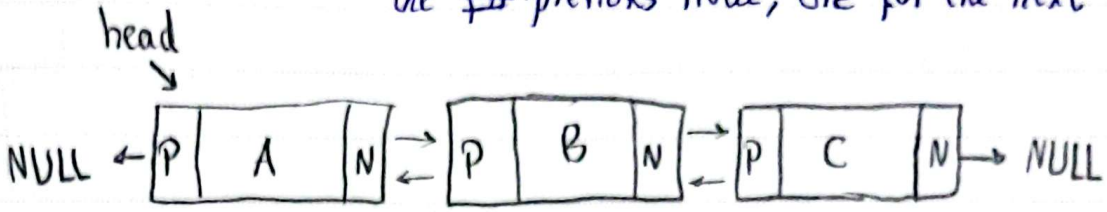
التاريخ: / / Data structure الموضوع

ex. on Singly linked list in C

```
typedef struct node{
        int data;
        Node * next;
} Node;

Node* creatNode(int data){
    Node *node = malloc(sizeof(Node));
    node -> data = data;
    node -> next = NULL;
    retum node;
}

        List
void printNode(Node *n){
    while(n != NULL){
        printf("%d ", n -> data);
        n = n -> next;
    }
}

int main(void){
    Node* head = createNode(1);
    Node* tail = create Node(2);
    head -> next = tail;
    printList(head);
}
```

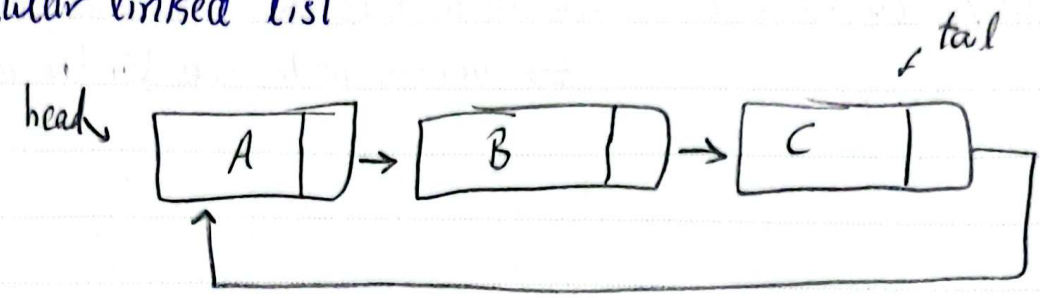
Output: 1 2

2. Doubly linked list : the node has two pointers one for the ~~for~~ previous node, one for the next



ex.
```
typedef struct {
    int data;
    Node * prev;
    Node *next;
} Node;

Node *createNode(int data) {
    Node *node = (Node*) malloc(sizeof(Node));
    node -> data = data;
    node -> prev = node -> next = NULL;
    return node;
}

void forwardTraversal(Node * head) {
    Node *curr = head;
    while(curr != NULL) {
        printf("%d", curr -> data);
        curr = curr -> next;
    }
}
```

3. Circular linked list

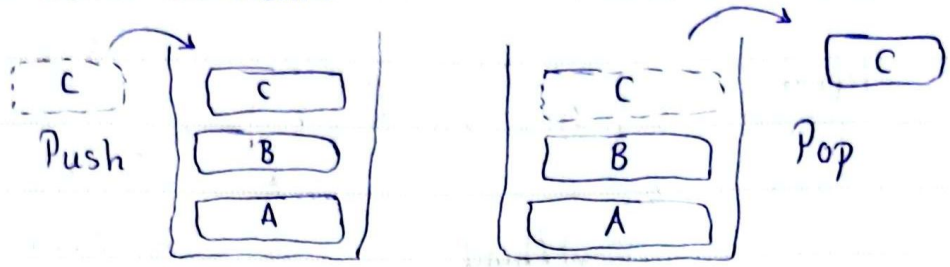



```
typedef struct {
    int data;
    Node *next;
}

Node *createNode (int data) {
    Node *node = (Node*) malloc (sizeof (Node));
    node -> data = data;
    node -> next = NULL;
    return node;
}
int main (void) {
    Node *first = createNode (2);
    first -> next = createNode (3);
    first -> next -> next = createNode (4);
    Node *last = first -> next -> next;
    last -> next = first;
}
```

بسم الله الرحمن الرحيم

الموضوع: Data Structure التاريخ: / /

★ Stack data structure



(LIFO) → Last in, first out principle

⇐ لو عندك صندوق فيه كتب وعاوز تضيف كتاب هتحطه فوق عادي

ولو عاوز تشيله هتشيله عادي برضو ، بس لاتيجي تعوز تشيل كتاب

تحت شوية هتعوز تشيل كل الكتب اللي فوقه وبعد كدا هتبقى قادر تشيله

Types of stack :

⇒ Static (Fixed size) : implemented with arrays زي الصندوق

⇒ Dynamic : implemented with linked list حجمها بيبقى على قد العناصر

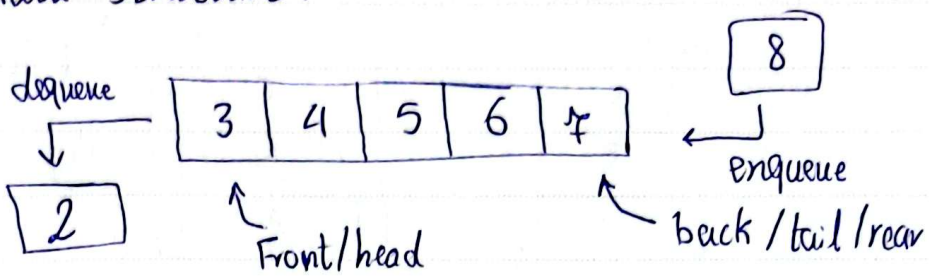
Operations :

push()	زود عنصر	isEmpty()	(Empty)? true : false
pop()	احذف عنصر	isFull()	(Full)? true : false
top()	بترجع أول عنصر		

بسم الله الرحمن الرحيم

الموضوع: Data structure التاريخ: / /

* Queue data structure :



FIFO ⇒ طابور ← الأول يتـ access الأول

← عاوز تنزور داتا ⇐ اقف فى الطابور (back)

← عاوز تشيل داتا ⇐ مش الراجل عشانه جاى منه بدرى