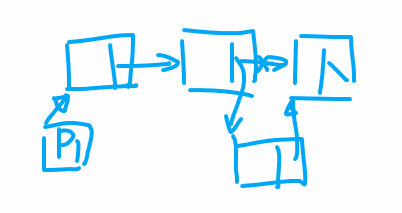
**March 15, 2021**

1. **Pointer (continued)**

* **Linked List**

****

**P1->next = p1->next->next; // delete a node**

**// traversal of a linked list**

**While( p1 != NULL)**

**{**

**Cout << p1->value << endl;**

**P1 = p1->next;**

**}**

1. **Stack**

**Precondition 1: you are allowed to insert or remove an item only from the top of the stack. In other words, you are not allowed to add or remove an item from the middle or bottom of the stack.**

**Two actions related to the stack:**

**Push: add one item to the top of the stack**

**Pop: remove one item from the top of the stack**

**Characteristic of the stack: First In Last Out (FILO)**

**Examples: a calling stack for recursion**

**A stack for transformation sequences**

**…**

**Structure:**

**Class Stack1**

**{**

**Private:**

**// static array: use an array name**

**// dynamic array: use a pointer**

**// linked list: use a pointer that points to the beginning of the linked list**

**Public:**

**makeEmpty( )**

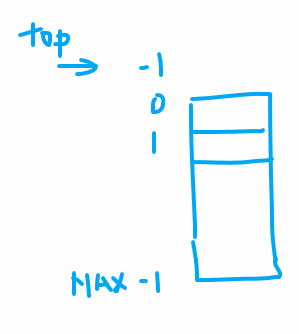
**isFull( )**

**isEmpty( )**

**push( )**

**pop( )**

**}**

****

**Template <class ItemType> // template <typename ItemType>**

**Class StackType**

**{**

**Private:**

**// dynamic array**

**Int top;**

**ItemType \*items;**

**Public:**

**StackType() { int len1; cin >> len1; items = new ItemType[len1];}**

**StackType(int len){ items=new ItemType[len];}**

**…**

**~StackType() { delete []items;}**

**};**

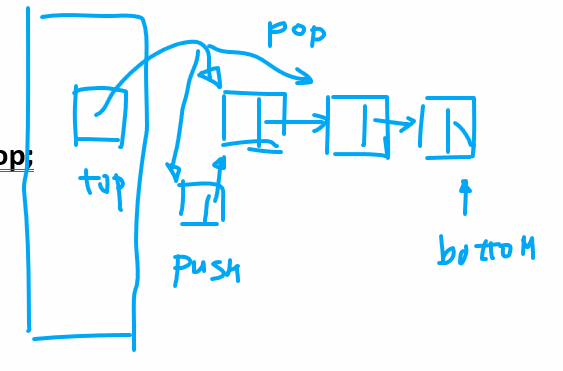
**Class StackLinkedList**

**{**

**Private:**

**ItemType \*top;**

**Int length;**

****

**};**