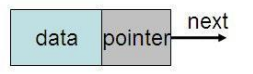
**Single Linked List**

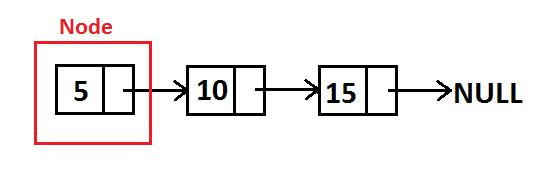
Linked List merupakan koleksi linear dari data yang disebut sebagai nodes, dimana setiap node akan menunjuk pada node lain melalui sebuah pointer. Linked List dapat didefinisikan pula sebagai kumpulan nodes yang merepresentasikan sebuah sequence.



setiap node terdiri atas 2 bagian

* Data / informasi yang disimpan (char, int, float, dll)
* Next, yaitu pointer yang menyimpan alamat node selanjutnya

Representasi sebuah linked list dapat digambarkan melalui gambar di bawah ini:



Sebuah linked list yang hanya memiliki 1 penghubung ke node lain disebut sebagai single linked list.

Di dalam sebuah linked list, ada 1 pointer yang menjadi gambaran besar, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri.

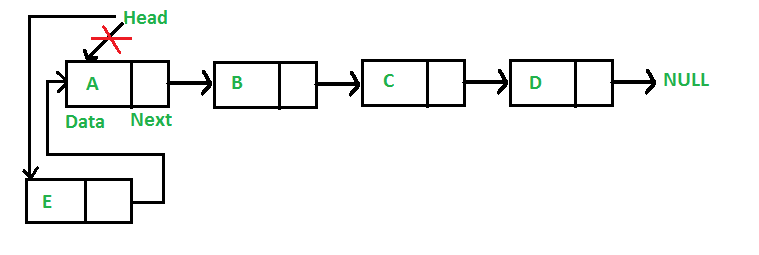


Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL.

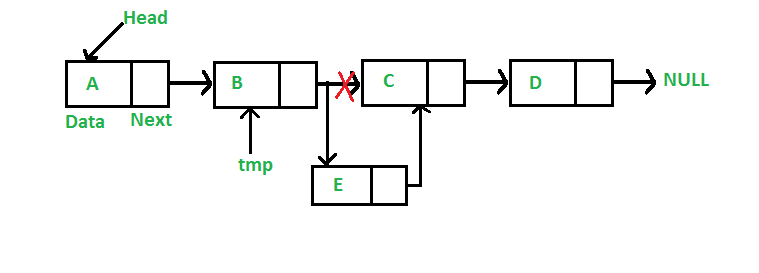
***“Secara teori, linked list adalah sejumlah node yang dihubungkan secara linier dengan bantuan pointer”.***

**Single Linked List - INSERT**

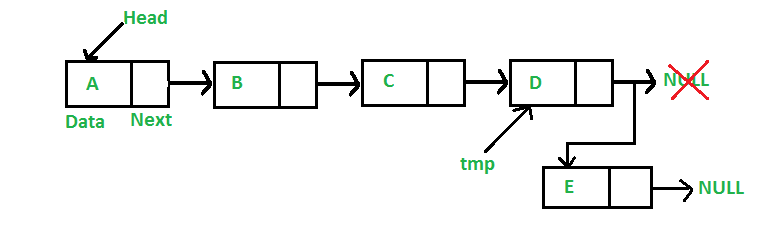
Istilah Insert berarti menambahkan sebuah simpul (Node) baru ke dalam suatu linked list.



|  |
| --- |
| /\* Given a reference (pointer to pointer) to the head of a list  and an int, inserts a new node on the front of the list. \*/  void push(struct Node\*\* head\_ref, int new\_data)  {  /\* 1. allocate node \*/  struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));  /\* 2. put in the data \*/  new\_node->data = new\_data;  /\* 3. Make next of new node as head \*/  new\_node->next = (\*head\_ref);  /\* 4. move the head to point to the new node \*/  (\*head\_ref) = new\_node;  } |



|  |
| --- |
| /\* Given a node prev\_node, insert a new node after the given  prev\_node \*/  void insertAfter(struct Node\* prev\_node, int new\_data)  {  /\*1. check if the given prev\_node is NULL \*/  if (prev\_node == NULL)  {  printf("the given previous node cannot be NULL");  return;  }    /\* 2. allocate new node \*/  struct Node\* new\_node =(struct Node\*) malloc(sizeof(struct Node));  /\* 3. put in the data \*/  new\_node->data = new\_data;  /\* 4. Make next of new node as next of prev\_node \*/  new\_node->next = prev\_node->next;  /\* 5. move the next of prev\_node as new\_node \*/  prev\_node->next = new\_node;  } |



|  |
| --- |
| /\* Given a reference (pointer to pointer) to the head  of a list and an int, appends a new node at the end \*/  void append(struct Node\*\* head\_ref, int new\_data)  {  /\* 1. allocate node \*/  struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));  struct Node \*last = \*head\_ref; /\* used in step 5\*/  /\* 2. put in the data \*/  new\_node->data = new\_data;  /\* 3. This new node is going to be the last node, so make next  of it as NULL\*/  new\_node->next = NULL;  /\* 4. If the Linked List is empty, then make the new node as head \*/  if (\*head\_ref == NULL)  {  \*head\_ref = new\_node;  return;  }    /\* 5. Else traverse till the last node \*/  while (last->next != NULL)  last = last->next;  /\* 6. Change the next of last node \*/  last->next = new\_node;  return;  } |