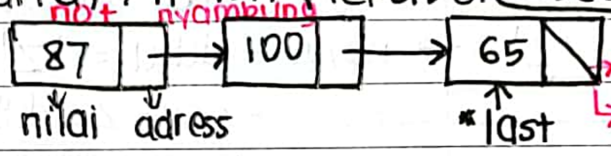
No
Date

* DEFINISI LINKED LIST

(berunit/berurutan/bertahap → tdk dpt lgn cat → dpt mjd kelemahannya)

- Yaitu model data sekuensial yg saling berkait
- Mirip dgn array, tp lbh fleksibel & sedikit kompleks (efisien)
- List Data →



(variabel / nama array)

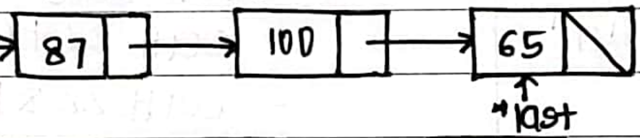
Bth 50 dpt gunakan 50, tp dpt ditambah

↳ grs menandakan stlh elemen ini tdk ada yg lain

- Array adl tipe data yg dpt menyimpan bny nilai dlm 1 var
- Kelemahan : krg fleksibel
- Jika dideklarasikan 100 mk hny dpt simpan 100 (max)

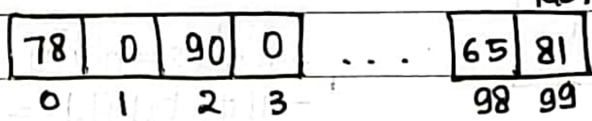
* LINKED LIST VS ARRAY

(L) * list Data →



} linked list tdk punya nomor indeks elemen
↳ cara akses dr not plg dpn

(AR) deret Nilai :



} aray punya nomor indeks elemen
↳ cara menampilkan deretNilai[5];

LINKED LIST	ARRAY
1) Fleksibel dlm +/- elemen data	Tdk dpt +/- elemen data
2) Tdk punya indeks no elemen	Punya
3) Bth tipe data pointer	Tdk bth
4) Nilai dr tiap elemen hny dpt diakses mll proses penc. sekuen	Nilai dr tiap elemen dpt diakses lgsng dgn tulis indeks elemen
5) Simpan data jml berubah² (dnms)	Simpan dt jml ttp (statis)

* KONSEP POINTER

- Pointer adl tipe data yg berfungsi membaca/ menyimpan alamat suatu variabel
- Pada C++ variabel bertipe pointer diberi tanda asterisk (*)
- Contoh :
  1) char *ptr;
  2) int *indeks;
  3) float *data;

No

Date

* CARA KERJA POINTER

Address	Value
0x0001	'a'
0x0002	'z' → 1 Byte
0x0003	→ indeks
0x0004	g5 → 4 byte krn bertipe int
0x0005	
0x0006	
....	...
0XFFFC	
0XFFFD	'm'
0XFFFE	
0XFFFF	't'

tipe data :
char = 1 byte
int = 4 byte
float = 8 byte
→ variabel

- char data;
- data = 'z';
- cout << data ; → menampilkan z
- cout << &data; → menampilkan 0x0002 → adres

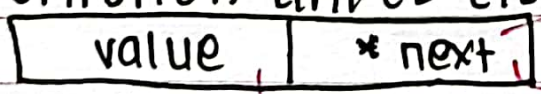
⊛ satuan terkecil dlm memori adl bit
8 bit = 1 byte

- int harga = 15000;
- cout << harga; → tampilkan: 15000
- cout << &harga; → tampilkan 0x0003
- indeks = harga → error (ptr = int)
- int harga = 15000;
- int *indeks; → tampilkan 0x0000
- indeks = &harga; → tampilkan 0x0003
- harga = 17000;
- cout << *indeks; → tampilkan 17000

bisa 17000 krn indeks simpan alamat. Id ketika nilainya diubah mk akan berubah

- cout << indeks; → tampilkan 0x0003

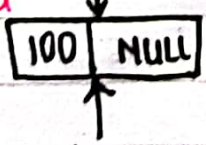
* KOMPONEN LINKED LIST



→ Node
→ pointer to next

* list Data → utk simpan data biasa
→ simpan alamat dr node berikutnya

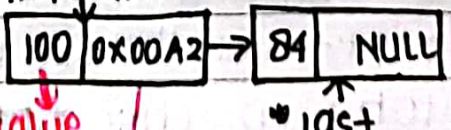
elemen pertama



* last

tunjukkan elemen terakhir dr LL

misal ditambah :

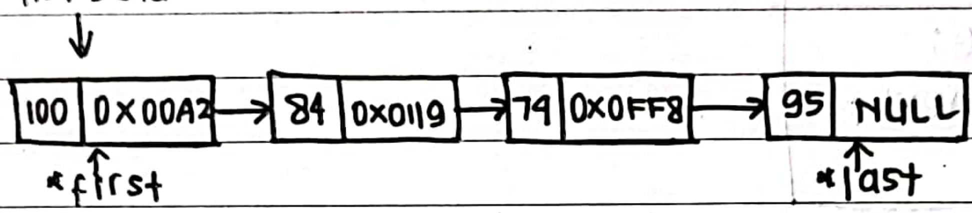


* list Data
value
* last

Fungsinya menunjukkan node yg ditempati 84

* LINKED LIST VS ARRAY

* list Data



deret Nilai :

100	84	74	95
0	1	2	3

* DEKLARASI LINKED LIST (C++)

- Deklarasi Record 'Node'

```
struct Node {          → Nama record (bebas)
    int value;         → variabel   (int → tipe)
    Node *next;        (Node → tipe data)
};
```

- Deklarasi Var. Bertipe 'Node'

```
Node *linkedList, *last;
```

(*linkedList → tunjukkan alamat dr Node)

- Deklarasi Record 'Node'

```
struct Node {
    string nim, nama, prodi;
    float IPK;
    int usia;
    Node *next;
};
```

- Deklarasi var. Bertipe 'Node'

```
Node *listMhs, *last;
```

No
Date

* MENGAKSES ELEMEN NODE (C++)

```
struct Node {
    int value;
    Node *next;
};
```

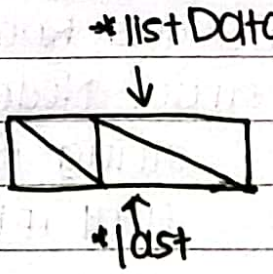
```
Node *listData, *last;
```

```
int main () {
    listData -> value = .. .. ;
    listData -> next = .. .. ;
}
```

→ Harusnya listData.value (dot) tapi karena di Node *listData adl pointer maka ditulis dgn "→"

* INISIALISASI LINKED LIST (C++)

```
struct Node {
    int value;
    Node *next;
};

Node *listData, *last;

int main () {
    //inisialisasi variabel linked list
    listData = null;
    last = listData;
}
```

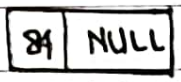


No
Date

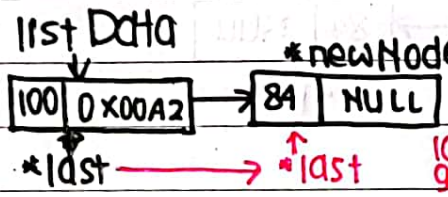
* MENAMBAH ELEMEN BARU LINKED LIST (C++)

① Insert Last()

1) Buat *newNode & deklarasikan



2) Tambahkan ke list yg sdh ada



Agar tsd fungsi pass by reference

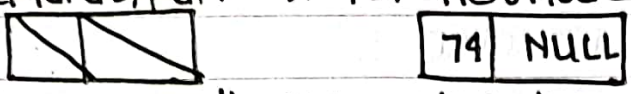
```
void insertLast(Node *&linkedList, Node *&last, int n){
    //create newNode.
    Node *newNode;
    newNode = new Node;

    //assign value of newNode
    newNode -> value = n;
    newNode -> next = NULL;

    if (linkedList == NULL){
        linkedList = newNode;
        last = linkedList;
    }
    else {
        last -> next = newNode;
        last = newNode;
    }
}
```

No

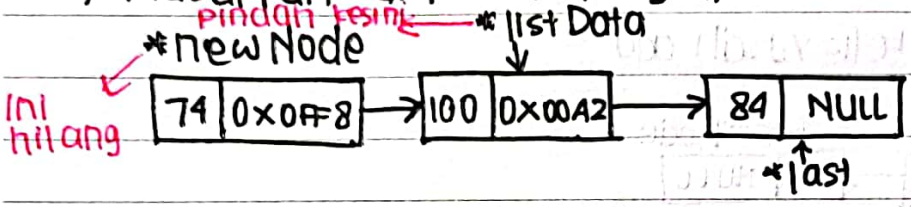
Date

② insertFirst( )

1) Deklarasikan & isi *newNode nya



2) Masukkan di posisi ping dpn




```
void insertFirst(Node *&linkedList, Node *&last, int n){
    //create newNode
    Node *newNode;
    newNode = new Node;

    //assign value of newNode
    newNode->value = n;
    newNode->next = NULL;

    if (linkedList == NULL){
        linkedList = newNode;
        last = linkedList;
    }
    else {
        newNode->next = linkedList;
        linkedList = newNode;
    }
}
```

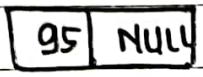
No

Date

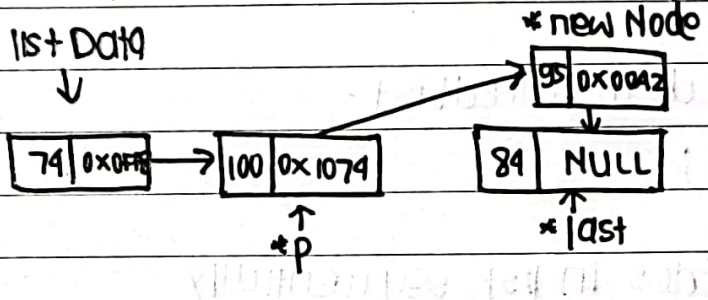
③ insertAfter ()

1) Deklarasikan new node dan isi

* new Node



2) Butuh pointer baru, misal * p




```
void insertAfter(Node *& linkedList, Node *& last, int n, int check){
    //create new Node
    Node *newNode;
    newNode = new Node;

    //assign value of new Node
    newNode-> value = n;
    newNode-> next = NULL;

    if (linkedList == NULL) {        -> cek linked list kosong atau
        linkedList = newNode;
        last       = linkedList;
    }
    else {
        Node *p = linkedList;
        //pakai while krn kita tdk tahu berhenti dmn
        while (p-> value != check) {
            p = p-> next;
        }
        newNode -> next = p-> next;
        p -> next = newNode;
    }
}
```

No
Date

* MENGAKSES ELEMEN LINKED LIST (C++)

```
void displayList(Node *& linkedList){
    //create new pointer as iterator
    Node *p;

    // p is connected to linkedList.
    p = linkedList;

    // display all nodes in list sequentially
    while (p != NULL){
        //show the element of node that is pointed by *p
        cout << p->value << endl;

        // move *p to the next node
        p = p-> next;
    }
}
```

* MENGHAPUS ELEMEN LINKED LIST (C++)

① Delete last ()

```
void deleteLast(Node *& linkedList, Node *& last){
    if (linkedList == NULL){
        cout << "linkedList is empty!";
    }
    else {
        Node *temp, *p;
        temp = last;
        p = linkedList;
        while (p-> next != last)
            p = p-> next;
        last = p;
        last-> next = NULL;
        delete (temp);
    }
}
```

JOYKO 30 Lines, 6 mm