Modul 6. Single Linked List

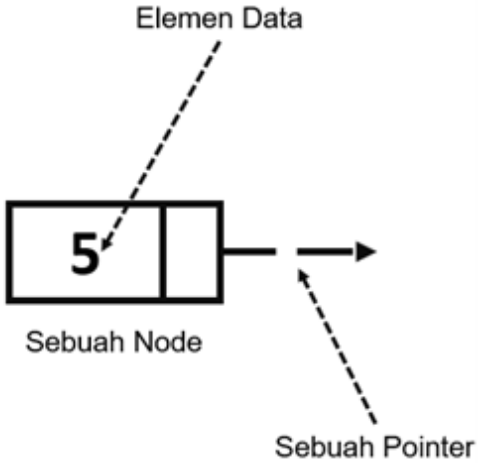
Jump to bottom

chintyatribhuanau edited this page 2 weeks ago · 4 revisions

Single Linked List

Pengertian Linked List

Linked List adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang tersusun secara sekuensial, saling sambung menyambung, dinamis dan terbatas. Data yang disimpan dalam bentuk list bisa berupa data tunggal maupun data majemuk. Data tunggal adalah data yang terdiri dari satu variabel. Sedangkan data majemuk adalah data yang terdiri dari berbagai tipe data, misalnya data mahasiswa, terdiri dari variabel nama yang bertipe data string, NIM bertipe integer dan sebagainya. Linked List dalam C terdiri dari simpul seperti struktur, yang selanjutnya dapat dibagi menjadi 2 bagian dalam kasus Single Linked List. Kedua bagian ini adalah-: Node – untuk menyimpan data. Pointer – untuk menyimpan alamat node berikutnya.



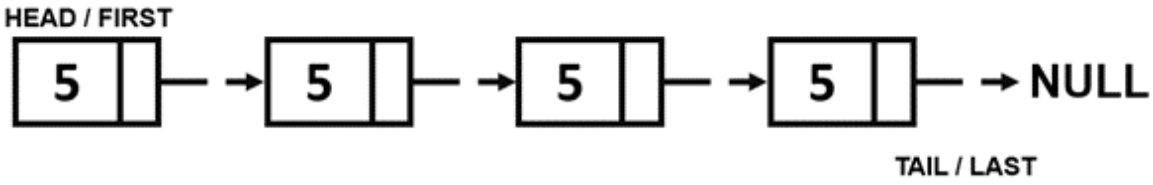
Setiap node akan berbentuk struct dan memiliki satu buah field bertipe struct yang sama, yang berfungsi sebagai pointer. Node pertama disebut Head/First. Jika linked list kosong, maka nilai dari head menunjuk ke NULL. Secara umum operasi yang sering digunakan pada linked list, yaitu :

1. Pembuatan dan inisialisasi list (Create List).
2. Pembuatan elemen baru (Create element)
3. Penyisipan elemen list (Insert).
4. Penghapusan elemen list (Delete).
5. Menampilkan elemen list (View).

Pengaplikasian Linked List

- Dapat digunakan untuk mengimplementasikan struktur data yang bermanfaat seperti stack dan queue.
- Dapat digunakan untuk mengimplementasikan tabel hash, setiap keranjang tabel hash dapat menjadi linked list..
- Dapat digunakan untuk mengimplementasikan grafik (representasi Adjacency List dari grafik)

Contoh Single Linked List



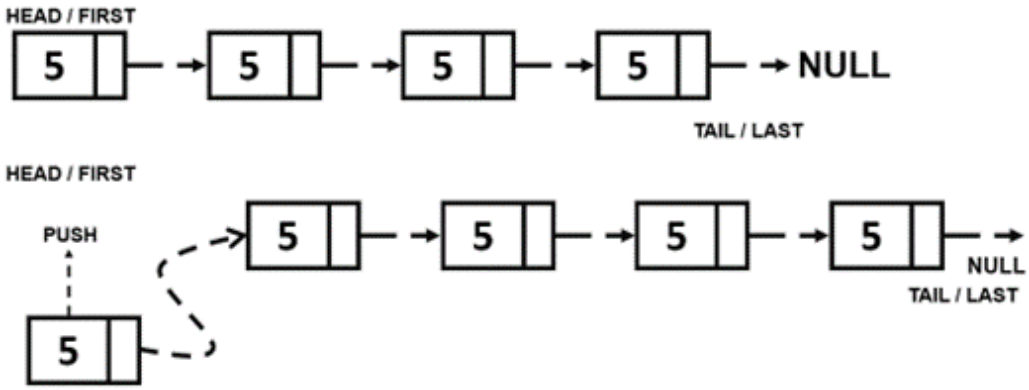
Operasi pada Single Linked List

Pada single linked list secara umum dikenal dua operasi, yakni operasi push dan pop. Operasi push adalah operasi yang menambahkan data pada list, sedangkan operasi pop adalah operasi yang menghapus data dari list. Untuk operasi push secara umum dapat dilakukan secara first yang menambahkan data pada head dan secara last yang menambahkan data pada tail. Sedangkan pop juga sama, dapat dilakukan secara first yang menghapus data pada head dan secara last yang menghapus data pada tail.

1. Create Node

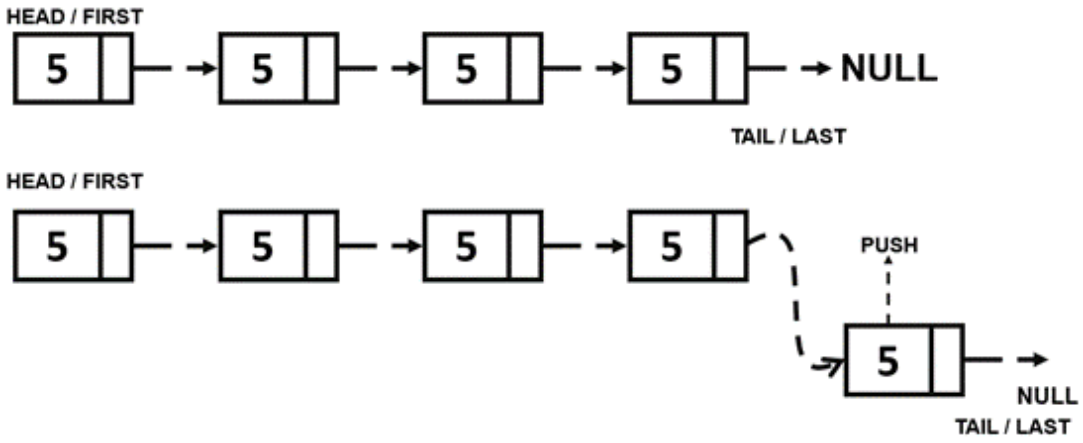
```
/Deklarasi pointer dengan tipe data struct node
typedef struct node *address;
// Isi dari struct
struct node{
        int isi;
        address next;
};
//fungsi membuat simpul baru
address createNode(int nilai){
    address p;
    //alokasi node
    p = (address)malloc(sizeof(struct node));
    p->isi = nilai;
    p->next = NULL;
    return (p);
}
```

2. Insert First / Push pada Head



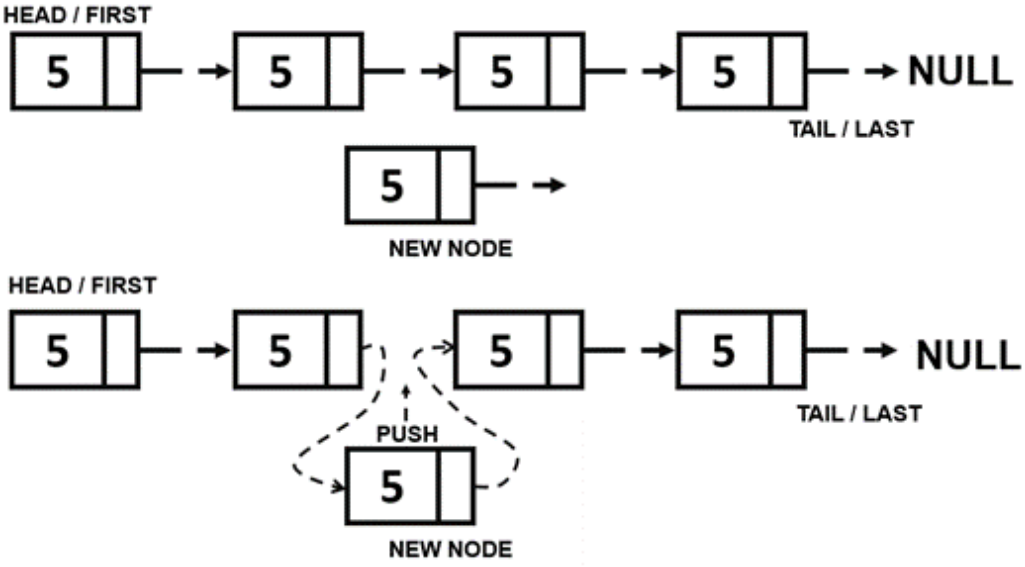

```
address insertFirst(address head, int nilai){
    address new_node = createNode(nilai);
    new_node->next = head;
    head = new_node;
    return(head);
}
```

3. Insert Last / Push pada Tail




```
address insertLast(address head, int nilai){
    //iterasi mencari node terakhir
    address tail = head;
    while(tail->next != NULL){
        tail = tail->next;
    }
    //buat simpul baru
    address new_node = createNode(nilai);
    tail->next = new_node;
    return(head);
}
```

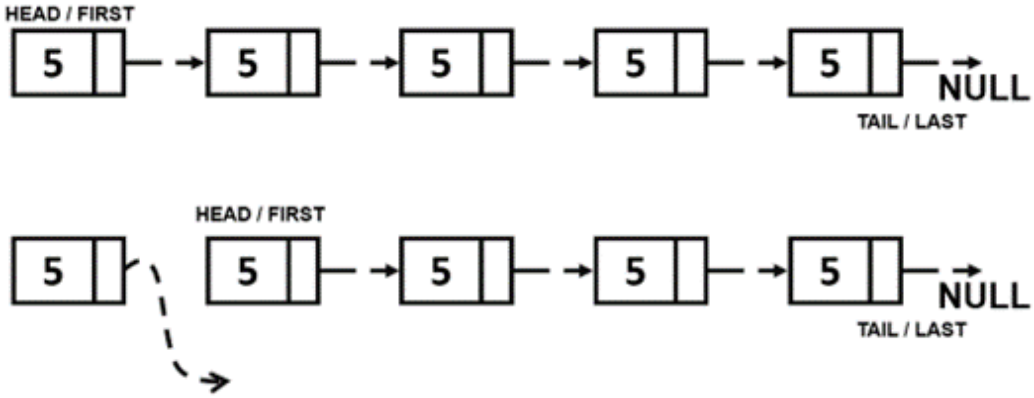
4. Insert After atau Push setelah index-N




```
address insertAfter(address head, int nilai, int prev_nilai){
//mencari simpul sebelumnya, mulai dari simpul pertama
    address cursor = head;
    while (cursor->isi != prev_nilai){
        cursor = cursor->next;
    }
    address new_node = createNode(nilai);
    new_node->next = cursor->next;
    cursor->next = new_node;
    return(head);
}
```

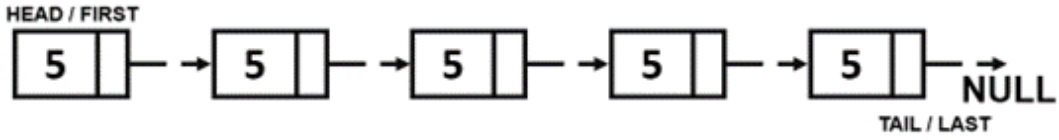
```
address insertBefore(address head, int nilai, int next_nilai){
    if(head->isi == next_nilai){
        head = insertFirst(head, nilai);
    } else {
        address cursor, prevcursor;
        cursor = head;
        do {
            prevcursor = cursor;
            cursor = cursor->next;
        } while (cursor->isi != next_nilai);

        address new_node = createNode(nilai);
        new_node->next = cursor;
        prevcursor->next = new_node;
    } return(head);
}
```

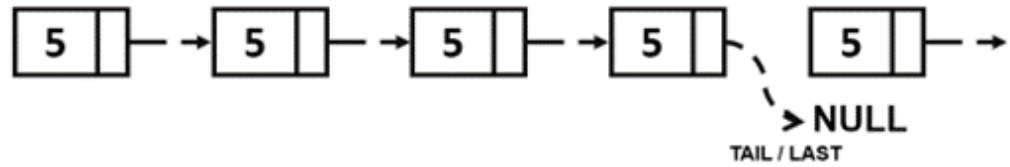
5. Delete First atau Pop pada Head




```
address deleteFirst(address head){
    if(head==NULL){
        return;
    }
    address first = head;
    head = head->next;
    first->next = NULL;
    free(first);
    return(head);
}
```

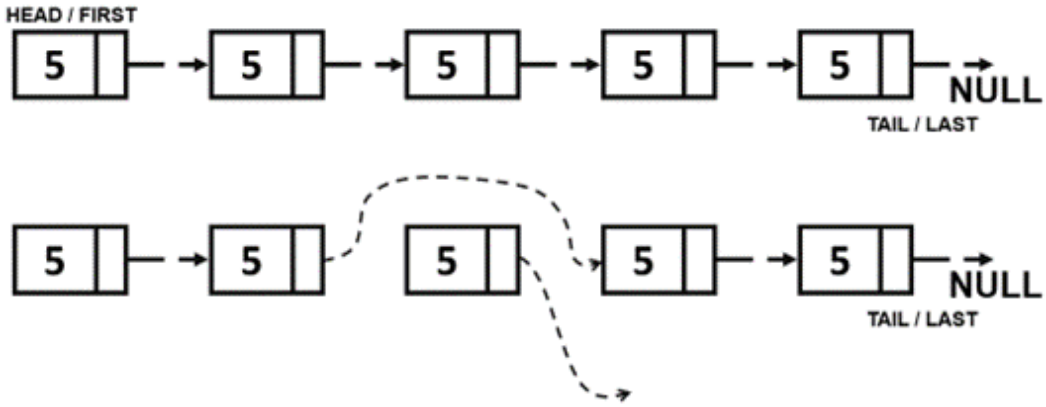
6. Delete Last atau Pop pada Tail






```
address deleteLast(address head){
    if (head==NULL){
        return;
    }
    address tail = head;
    address prevtail = NULL;
    while (tail->next != NULL){
        prevtail = tail;
        tail = tail->next;
    }
    prevtail->next = NULL;
    free(tail);
    return(head);
}
```

7. Delete After atau Pop setelah index-N




```
address deleteAfter(address head, int nilai){
    address cursor = head;
    while (cursor != NULL){
            if(cursor->next->isi == nilai){
                break; //keluar dari iterasi
            } cursor = cursor->next;
    }
    if (cursor != NULL){
        address tmp = cursor->next;
        cursor->next = tmp->next;
        tmp->next = NULL;
        free(tmp);
    }
    return(head);
}
```

8. Fungsi Tampil Nilai, Dispose, dan count

```
void tampilnilai(address head){
    address n = head;
    printf("Daftar nilai linked list : \n");

    while (n != NULL){
        printf("[%d] ", n->isi);
        n = n->next;
    };
    printf("\n");
}

//menghitung jumlah node
address count(address head){
    int count = 0;
    struct node *p;
    p = head;
    while (p != NULL){
        p = p->next;
        count++;
```

```
    }
    printf("Jumlah node adalah : %d\n", count);
    return(head);

}

address dispose(address head){
    if(head == NULL){
        return;
    }
    while(head != NULL){
        address tmp = head;
        head = head->next;
        tmp->next = NULL;
        free(tmp);
    }
    printf("Semua node (simpul) telah dihapus\n");
    return(head);
}
```

9. Fungsi Main

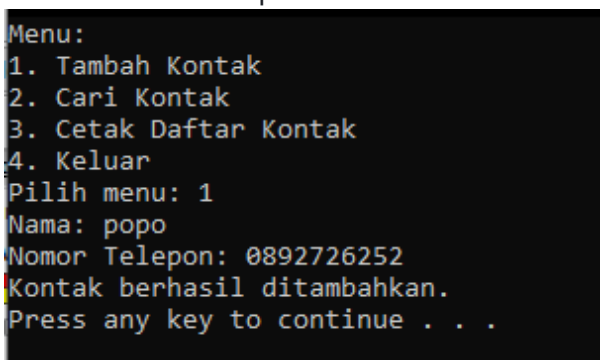
```
int main()
{
    int pilih, val, val1;
    address head = NULL;
    while(1){
    system("cls");
    tampilnilai(head);
    printf("\n======================== SINGLE LINKED LIST ========================\n");
    printf("1. Insert First\n");
    printf("2. Insert Last\n");
    printf("3. Insert After\n");
    printf("4. Insert Before\n");
    printf("5. Delete First\n");
    printf("6. Delete Last\n");
    printf("7. Delete After\n");
    printf("8. Hitung Jumlah Node\n");
    printf("9. Hapus Keseluruhan Node\n");
    printf("10. Keluar\n");
    printf("Pilihan Anda = ");
    scanf("%d", &pilih);
    switch (pilih){
    case 1:
        {
            printf("Masukkan nilai = ");
            scanf("%d", &val);
            head = insertFirst(head, val);
            system("cls");
            break;
        }
```

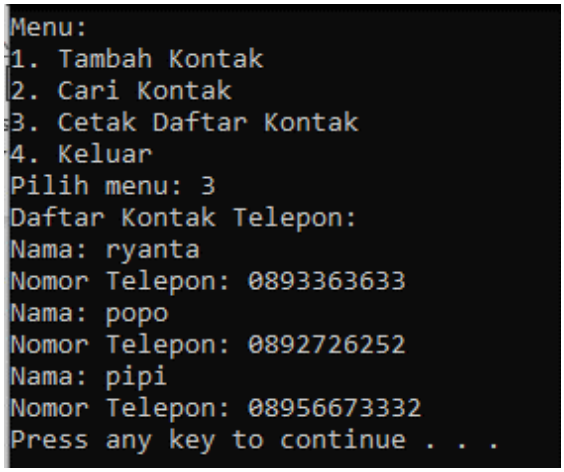
```
        case 2 :
            {
                printf("Masukkan nilai = ");
                scanf("%d", &val);
                head = insertLast(head, val);
                system("cls");
                break;
            }
        case 3 :
            {
                printf("Masukkan nilai = ");
                scanf("%d", &val);
                printf("Masukkan nilai tersebut setelah = ");
                scanf("%d", &val1);
                head = insertAfter(head, val, val1);
                system("cls");
                break;
            }
        case 4 :
            {
                printf("Masukkan nilai = ");
                scanf("%d", &val);
                printf("Masukkan nilai tersebut sebelum = ");
                scanf("%d", &val1);
                head = insertBefore(head, val, val1);
                system("cls");
                break;
            }
        case 5 :
            {
                head = deleteFirst(head);
                system ("cls");
                break;
            }
        case 6 :
            {
                head = deleteLast(head);
                system ("cls");
                break;
            }
        case 7 :
            {
                printf("Masukkan nilai yang ingin dihapus = ");
                scanf("%d", &val);
                head = deleteAfter(head, val);
                system ("cls");
                break;
            }
        case 8 :
            {
                head = count(head);
                system ("Pause");
                break;
```

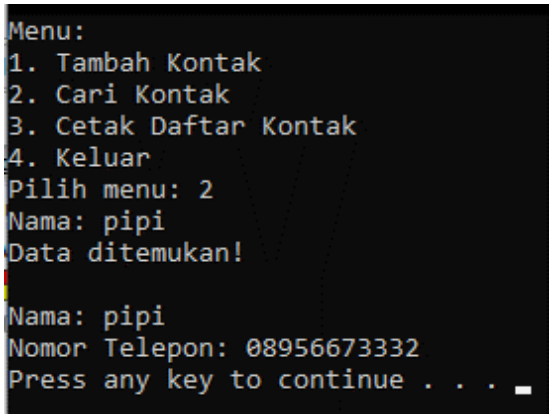
```
            }
        case 9 :
            {
                head = dispose(head);
                system ("Pause");
                break;
            }
        case 10 :
            {
                exit(1);
            }
        default :
            {
                printf("Pilihan tersebut tidak ada");
                break;
            }
        }
        }
        return 0;
    }
```

Soal Latihan

Buatlah program menggunakan linked-list untuk menyimpan daftar kontak telepon. Setiap simpul dalam linked-list berisi informasi kontak seperti nama dan nomor telepon. Program tersebut harus dapat menambahkan kontak baru, mencari kontak berdasarkan nama, dan mencetak seluruh daftar kontak. Output:







Kuis: https://forms.gle/wY7pfyXV3gszT2UY7

Asisten Praktikum Algoritma Struktur Data Informatika 2023

Pages 8

Home

- Modul 0. Pendahuluan
- Modul 1. Array

- Modul 2. Pointer dan Fungsi
- Modul 3. ADT dan Struct
- Modul 4. Stack
- Modul 5. Queue
- Modul 6. Single Linked List

Clone this wiki locally

```
https://github.com/fzl-22/Algoritma-dan-Struktur-Data_Informatika.wiki.git
```