LAPORAN PRAKTIKUM

STRUKTUR DATA DAN ALGORITME

MODUL V

**LINKED LIST**

Logo

Description automatically generated

**Disusun Oleh :**

Nama : Fatkhurrohman Purnomo

NIM : 21102125

**Dosen Pengampu**

Ipam Fuaddina Adam, S.T., M.Kom.

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**PURWOKERTO**

**2022**

# Dasar Teori

### Konsep Linked List

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Linked List sering disebut senarai berantai. Untuk menghubungkan satu node dengan node yang lainnya Linked List menggun akan pointer sebagai penunjuk node selanjutnya. Node sendiri merupakan sebuah struct yang terdiri dari beberapa field, minimal ada 2 buah field yaitu field untuk isi dari struct datanya sendiri, dan 1 field artbitrary bertipe pointer sebagai penunjuk node selanjutnya.

### Aturan Linked List

* + - 1. Data harus memiliki hubungan dengan yang lain.
      2. Data yang terhubung tidak boleh bercabang.

### Tipe Linked List

Salah satu tipe Linked List yang sederhana yaitu Single Linked List. Single Linked List merupakan Linked List yang memiliki hanya satu pointer penunjuk dengan arah data hanya satu arah saja. Single Linked List memiliki 2 macam bentuk yaitu Non Circular dan Circular. Non Circular Linked List merupakan Linked List dimana antara kepala (head) dan node terakhir (tail) tidak memiliki hubungan. Pada Linked List ini maka pointer terakhir selalu menunjuk NULL sebagai pertanda data terakhir dalam list-nya. Single Linked List Non Circular dapat digambarkan sebagai berikut.

Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.

Langkah membuat dan operasi pada sebuah Linked List adalah sebagai berikut :

1. Mendeklarasikan struct node

2. Membuat node head

3. Menginisialiasi node head

4. Menambahkan node baru baik di depan maupun di belakang

5. Menghapus node tertentu

### Perbedaan Array dan Linked List

ARRAY

* Statis
* Penambahan dan penghapusan data terbatas
* Random access
* Penghapusan array tidak mungkin

LINKED LIST

* Dinamis
* Penambahan dan penghapusan data tidak terbatas
* Sequential access
* Penghapusan mudah

Linked List banyak dimanfaatkan pada program kecerdasan buatan, fuzzy, maze solving, dan sebagainya.

Ref:

Modul 5 Linked List

[Pengertian Linked List Dalam C++ Beserta Contoh Programnya - syarif soden](https://syarifsoden.blogspot.com/2020/08/pengertian-linked-list-dalam-c-beserta.html)

[Materi dan Contoh Program Single Linked List pada C++ – Forumkomputer.Com](https://www.forumkomputer.com/program-linked-list/)

[Single Linked List C++ Disertai Contoh, Tugas & Jawaban - TeachMeSoft](https://www.teachmesoft.com/2019/03/single-linked-list-c-disertai-contoh.html)

# Guided

## Program Single Linked List Non-Circular

|  |
| --- |
| // Nama : Fatkhurrohman Purnomo  // NIM : 21102125  #include <iostream>  using namespace std;  ///PROGRAM SINGLE LINKED LIST NON-CIRCULAR  //Deklarasi Struct Node  struct Node  {      int data;      Node \*next;  };  Node \*head;  Node \*tail;  //Inisialisasi Node  void init()  {      head = NULL;      tail = NULL;  }  //Pengecekan  bool isEmpty()  {      if (head == NULL)          return true;      else          return false;  }  //Tambah Depan  void insertDepan(int nilai)  {      //Buat Node baru      Node \*baru = new Node;      baru->data = nilai;      baru->next = NULL;      // Jika list kosong      if (isEmpty() == true){          head = tail = baru;          tail->next = NULL;      }      else{          baru->next = head;          head = baru;      }  }  //Tambah Belakang  void insertBelakang(int nilai)  {      //Buat Node baru      Node \*baru = new Node;      baru->data = nilai;      baru->next = NULL;        // Jika list kosong      if (isEmpty() == true){          head = tail = baru;          tail->next = NULL;      }      else{          tail->next = baru;          tail = baru;      }  }  //Hitung Jumlah List  int hitungList()  {      Node \*hitung; //Node bantu      hitung = head; //Node bantu diinisialisasi dengan head      int jumlah = 0; //Inisialisasi jumlah dengan 0      // Hitung jumlah elemen      while( hitung != NULL ){          jumlah++;          hitung = hitung->next;      }      return jumlah;  }  //Tambah Tengah  void insertTengah(int data, int posisi)  {      if( posisi < 1 || posisi > hitungList() ){ //Jika posisi tidak sesuai          cout << "Posisi diluar jangkauan" << endl;      }else if( posisi == 1){ //Jika posisi pertama          cout << "Posisi bukan posisi tengah" << endl;      }else{ //Jika posisi tengah          Node \*baru, \*bantu; //Deklarasi Node baru dan bantu          baru = new Node(); //Membuat Node baru          baru->data = data; //Mengisi data          // tranversing          bantu = head;          int nomor = 1;            while( nomor < posisi - 1 ){ //Posisi sebelum posisi tengah              bantu = bantu->next;              nomor++;          }          baru->next = bantu->next; //Posisi tengah          bantu->next = baru; //Posisi sebelum posisi tengah      }  }  //Hapus Depan  void hapusDepan()      {      Node \*hapus;      if (isEmpty() == false){ //Jika list tidak kosong          if (head->next != NULL){ //Jika list hanya memiliki 1 elemen              hapus = head;              head = head->next;              delete hapus;          }          else{              head = tail = NULL;          }      }      else{          cout << "List kosong!" << endl;      }  }  //Hapus Belakang  void hapusBelakang()  {      Node \*hapus; //Node yang akan dihapus      Node \*bantu; //Node yang akan menunjuk ke node sebelum hapus      if (isEmpty() == false){ //Jika list tidak kosong          if (head != tail){ //Jika list tidak berisi 1 elemen              hapus = tail;              bantu = head;              while (bantu->next != tail){ //Transversing                  bantu = bantu->next;              }              tail = bantu;              tail->next = NULL;              delete hapus;          }          else{              head = tail = NULL;          }      }      else{          cout << "List kosong!" << endl;      }  }  // Hapus Tengah  void hapusTengah(int posisi)  {      Node \*bantu, \*hapus, \*sebelum; //Deklarasi variabel        if( posisi < 1 || posisi > hitungList() ){ //Jika posisi tidak sesuai          cout << "Posisi di luar jangkauan" << endl;      }else if( posisi == 1){ //Jika posisi pertama          cout << "Posisi bukan posisi tengah" << endl;      }else{ //Jika posisi tengah          int nomor = 1; //Nomor posisi          bantu = head;            while( nomor <= posisi ){ //Transversing          if( nomor == posisi-1 ){ //Jika posisi sebelumnya              sebelum = bantu;          }              if( nomor == posisi ){ //Jika posisi yang dihapus                  hapus = bantu;              }              bantu = bantu->next; //Transversing              nomor++;          }            sebelum->next = bantu;          delete hapus;      }  }  //Ubah Depan  void ubahDepan(int data)  {      if (isEmpty() == 0){ //Jika list tidak kosong          head->data = data;      }      else{          cout << "List masih kosong!" << endl;      }  }  //Ubah Tengah  void ubahTengah(int data, int posisi)      {        Node \*bantu;      if (isEmpty() == 0){ //Jika list tidak kosong          if( posisi < 1 || posisi > hitungList() ){ //Jika posisi tidak sesuai              cout << "Posisi diluar jangkauan" << endl;              cout << "Posisi di luar jangkauan" << endl;          }          else if( posisi == 1){ //Jika posisi pertama              cout << "Posisi bukan posisi tengah" << endl;          }          else{ //Jika posisi tengah              bantu = head;              int nomor = 1;                while (nomor < posisi){                  bantu = bantu->next;                  nomor++;              }              bantu->data = data;          }      }else{ //Jika list kosong          cout << "List masih kosong!" << endl;      }  }  //Ubah Belakang  void ubahBelakang(int data)  {      if (isEmpty() == 0){ //Jika list tidak kosong          tail->data = data;      }      else{          cout << "List masih kosong!" << endl;      }  }  //Hapus List  void clearList()  {      Node \*bantu, \*hapus; //bantu untuk transversing, hapus untuk menghapus      bantu = head; //bantu di set head      while (bantu != NULL){ //Transversing          hapus = bantu;          bantu = bantu->next;          delete hapus;      }      head = tail = NULL;      cout << "List berhasil terhapus!" << endl;  }  //Tampilkan List  void tampil()  {      Node \*bantu; //bantu untuk transversing      bantu = head; //bantu di set head      if (isEmpty() == false){ //Jika list tidak kosong          while (bantu != NULL){ //Transversing              cout << bantu->data << ends;              bantu = bantu->next;          }          cout << endl;      }      else{          cout << "List masih kosong!" << endl;      }  }  int main()  {      init();      insertDepan(3);      tampil();      insertBelakang(5);      tampil();      insertDepan(2);      tampil();      insertDepan(1);      tampil();      hapusDepan();      tampil();      hapusBelakang();      tampil();      insertTengah(7,2);      tampil();      hapusTengah(2);      tampil();      ubahDepan(1);      tampil();      ubahBelakang(8);      tampil();      ubahTengah(11, 2);      tampil();      return 0;  } |

**Deskripsi:**

Program Single Linked List Non-Circular ini bisa melakukan penyimpanan data tanpa array, bisa digunakan dengan memanggil fungsi ditambah isi yang ingin di input. Nantinya inputan tersebut akan disimpan di memori, program juga bisa menghapus data yang sudah di inputkan.

**Output:**

A picture containing text

Description automatically generated

## Program Single Linked List Circular

|  |
| --- |
| #include <iostream>  using namespace std;  ///PROGRAM SINGLE LINKED LIST CIRCULAR  //Deklarasi Struct Node  struct Node  {      string data;      Node \*next;  };  Node \*head, \*tail, \*baru, \*bantu, \*hapus; // membuat variabel head, tail, baru, bantu, hapus  //Inisialisasi Node  void init()  {      head = NULL;      tail = head;  }  //Pengecekan  int isEmpty()  {      if (head == NULL) // jika head kosong          return 1; //true      else          return 0; //false  }  //Buat Node Baru  void buatNode(string data)  {      baru = new Node; // membuat node baru      baru->data = data; // mengisi data      baru->next = NULL; // mengisi next dengan NULL  }  //Hitung List  int hitungList()  {      bantu = head; // membuat bantu menunjuk head      int jumlah = 0; // membuat variabel jumlah      while (bantu != NULL){ // jika bantu tidak kosong          jumlah++;          bantu = bantu->next;      }      return jumlah;  }  //Tambah Depan  void insertDepan(string data)  {      //Buat Node baru      buatNode(data);      if (isEmpty() == 1){ // jika list kosong          head = baru;          tail = head;          baru->next = head;      }      else{ // jika list tidak kosong          while (tail->next != head){ // jika tail tidak menunjuk head              tail = tail->next;              }              baru->next = head;              head = baru;              tail->next = head;      }  }  //Tambah Belakang  void insertBelakang(string data)  {      //Buat Node baru      buatNode(data);      if (isEmpty() == 1){ // jika list kosong          head = baru;          tail = head;          baru->next = head;      }      else{ // jika list tidak kosong          while (tail->next != head){ // jika tail tidak menunjuk head              tail = tail->next;              }              tail->next = baru;              baru->next = head;      }  }  //Tambah Tengah  void insertTengah(string data, int posisi)  {      if (isEmpty() == 1){ // jika list kosong          head = baru;          tail = head;          baru->next = head;      }      else{ // jika list tidak kosong          baru->data = data;        //transversing      int nomor = 1;      bantu = head;        while (nomor < posisi - 1){ // jika nomor kurang dari posisi - 1          bantu = bantu->next;          nomor++;      }      baru->next = bantu->next; // mengisi next dengan bantu->next      bantu->next = baru; // mengisi bantu->next dengan baru      }  }  //Hapus Depan  void hapusDepan()  {      if (isEmpty() == 0){ // jika list tidak kosong          hapus = head;          tail = head;          if (hapus->next == head){ // jika head menunjuk head              head = NULL;              tail = NULL;              delete hapus;          }          else{ // jika head tidak menunjuk head              while (tail->next != hapus){                  tail = tail->next;              }              head = head->next;              tail->next = head;              hapus->next = NULL;              delete hapus; // menghapus data          }      }      else{          cout << "List masih kosong!" << endl;      }  }  //Hapus Belakang  void hapusBelakang()  {      if (isEmpty() == 0){ // jika list tidak kosong          hapus = head; // membuat hapus menunjuk head          tail = head; // membuat tail menunjuk head          if (hapus->next == head){ // jika head menunjuk head              head = NULL;              tail = NULL;              delete hapus;          }          else{ // jika head tidak menunjuk head              while (hapus->next != head){ // jika hapus tidak menunjuk head                  hapus = hapus->next;              }              while (tail->next != hapus){ // jika tail tidak menunjuk hapus                  tail = tail->next;              }              tail->next = head; // mengisi tail->next dengan head              hapus->next = NULL; // mengisi hapus->next dengan NULL              delete hapus;          }      }      else{          cout << "List masih kosong!" << endl;      }  }  //Hapus Tengah  void hapusTengah(int posisi)  {      if (isEmpty() == 0){ // jika list tidak kosong          //transversing          int nomor = 1;          bantu = head;            while (nomor < posisi - 1){ // jika nomor kurang dari posisi - 1              bantu = bantu->next;              nomor++;          }          hapus = bantu->next; // membuat hapus menunjuk bantu->next          bantu->next = hapus->next; // mengisi bantu->next dengan hapus->next          delete hapus;      }      else{          cout << "List masih kosong!" << endl;      }  }  //Hapus List  void clearList()  {      if (head != NULL){ // jika head tidak kosong          hapus = head->next;          while (hapus != head){ // jika hapus tidak menunjuk head              bantu = hapus->next;              delete hapus;              hapus = bantu;          }          delete head;          head = NULL;      }      cout << "List berhasil terhapus!" << endl;  }  //Tampilkan List  void tampil()  {      if (isEmpty() == 0){ // jika list tidak kosong          tail = head;          do{ // jika tail tidak menunjuk head              cout << tail->data << ends;              tail = tail->next;          }while (tail != head); // jika tail tidak menunjuk head          cout << endl;      }      else{          cout << "List masih kosong!" << endl;      }  }  int main()  {      init();      insertDepan("Ayam");      tampil();      insertDepan("Bebek");      tampil();      insertBelakang("Cicak");      tampil();      insertBelakang("Domba");      tampil();      hapusBelakang();      tampil();      hapusDepan();      tampil();      insertTengah("Sapi", 2);      tampil();      hapusTengah(2);      tampil();      return 0;  } |

**Dekripsi:**

Program Single Linked List Circular ini sama seperti program sebelumnya, bisa melakukan penyimpanan data tanpa array, bisa digunakan dengan memanggil fungsi ditambah isi yang ingin di input. Nantinya inputan tersebut akan disimpan di memori, program juga bisa menghapus data yang sudah di inputkan.

**Output:**

Text

Description automatically generated

# Tugas (Unguided)

**Kode**

|  |
| --- |
| #include <iostream>  using namespace std;  // struct Node  struct Node  {      int data;      string nama;      Node \*next;  };  Node \*head;  Node \*tail;  // tambah depan  void tambahDepan(Node \*&head, string nama, int nim)  {      Node \*baru = new Node; // membuat node baru      baru ->data = nim; // mengisi data      baru ->nama = nama; // mengisi nama      baru ->next = NULL; // mengisi next dengan NULL        if (head == NULL)      {          head = tail = baru;          tail ->next = NULL;      }      else      {          baru ->next = head;          head = baru;      }  }  // tambah belakang  void tambahBelakang(Node \*&head, string nama, int nim)  {      Node \*baru = new Node;      baru ->data = nim;      baru ->nama = nama;      baru ->next = NULL;        if (head == NULL)      {          head = tail = baru;          tail ->next = NULL;      }      else      {          tail ->next = baru;          tail = baru;      }  }  // tambah tengah  void tambahTengah(Node \*&head, string nama, int nim, int posisi)  {      Node \*baru = new Node;      baru ->data = nim;      baru ->nama = nama;      baru ->next = NULL;        if (head == NULL)      {          head = tail = baru;          tail ->next = NULL;      }      else      {          Node \*temp = head;          for (int i = 1; i < posisi; i++) // looping untuk mencari posisi          {              temp = temp ->next;          }          baru ->next = temp ->next;          temp ->next = baru;      }  }  // ubah depan  void ubahDepan(Node \*&head, string nama, int nim)  {      if (head == NULL)      {          cout << "List kosong" << endl;      }      else      {          cout << "Data " << head ->nama << " berhasil dihapus" << endl;          head ->data = nim;          head ->nama = nama;      }  }  // ubah belakang  void ubahBelakang(Node \*&head, string nama, int nim)  {      if (head == NULL)      {          cout << "List kosong" << endl;      }      else      {          cout << "Data " <<  tail ->nama << " berhasil diubah" << endl;          tail ->data = nim;          tail ->nama = nama;      }  }  // ubah tengah  void ubahTengah(Node \*&head, string nama, int main, int posisi)  {      if (head == NULL)      {          cout << "List kosong" << endl;      }      else      {          Node \*temp = head;          for (int i = 1; i < posisi; i++)          {              temp = temp ->next;          }          Node \*current = head;          for (int i = 1; i < posisi; i++)          {              current = current ->next;          }          cout << "Data " << current ->nama << " berhasil diubah" << endl;          temp ->data = main;          temp ->nama = nama;      }  }  // hapus depan  void hapusDepan(Node \*&head)  {      // jika list kosong      if (head == NULL)      {          cout << "List kosong" << endl;      }      else      {          // jika list hanya satu elemen          cout << "Data " << head ->nama << " berhasil dihapus" << endl;          if (head->next == NULL)          {              delete head;              head = NULL;              tail = NULL;          }          else          {              // jika list lebih dari satu elemen              Node \*hapus = head;              head = head->next;              delete hapus;          }      }  }  // hapus belakang  void hapusBelakang(Node \*&head)  {      // jika list kosong      if (head == NULL)      {          cout << "List kosong" << endl;      }      else      {          // jika list hanya satu elemen          cout << "Data " <<  tail ->nama << " berhasil dihapus" << endl;          if (head->next == NULL)          {              delete head;              head = NULL;              tail = NULL;          }          else          {              // jika list lebih dari satu elemen              Node \*previous = NULL;              Node \*current = head;              // looping untuk mencari elemen terakhir              while (current->next != NULL)              {                  previous = current;                  current = current->next;              }              delete current;              previous->next = NULL;              tail = previous;          }      }  }  // hapus tengah  void hapusTengah(Node \*&head, int posisi)  {      // jika list kosong      if (head == NULL)      {          cout << "List kosong" << endl;      }      else      {          // jika list hanya satu elemen          if (posisi == 1)          {              cout << "Data " << head ->nama << " berhasil dihapus" << endl;              if (head->next == NULL)              {                  delete head;                  head = NULL;                  tail = NULL;              }              else              {                  Node \*hapus = head;                  head = head->next;                  delete hapus;              }          }          else          {              Node \*previous = NULL;              Node \*current = head;              // looping untuk mencari elemen terakhir              for (int i = 1; i < posisi; i++)              {                  previous = current;                  current = current->next;              }              cout << "Data " << current ->nama << " berhasil dihapus" << endl;              previous->next = current->next;              delete current;          }      }  }  // hapus List  void hapusList(Node \*&head)  {      // jika list kosong      if (head == NULL)      {          cout << "List kosong" << endl;      }      else      {          Node \*current = head;          Node \*next;          // looping untuk menghapus semua elemen          while (current != NULL)          {              next = current->next;              delete current;              current = next;          }          head = NULL;          tail = NULL;      }  }  // tampilkan list  void tampilkanList(Node \*head)  {      if (head == NULL)      {          cout << endl << "List kosong" << endl;      }      else      {          // looping untuk menampilkan semua elemen          Node \*current = head;          cout << endl << "Data Mahasiswa" << endl;          cout << " NIM       |       Nama" << endl;          while (current != NULL)          {              cout << " " << current->data << "  |  " << current->nama << endl;              current = current->next;          }          cout << endl;      }  }  // mencari nilai  bool cariNilai(Node \*head, int nilai)  {      // jika list kosong      if (head == NULL)      {          return false;      }      else      {          Node \*current = head;          // looping untuk mencari elemen          while (current != NULL)          {              if (current->data == nilai)              {                  return true;              }              current = current->next;          }      }      return false;  }  int main(){      // deklarasi variabel      Node \*head = NULL;      int pilihan;      int posisi;      string nama;      int nim;      do      {          cout << "===========================================================" << endl;          cout << "|                        Daftar Menu                      |" << endl;          cout << "===========================================================" << endl;          cout << "   1. Tambah depan" << endl;          cout << "   2. Tambah belakang" << endl;          cout << "   3. Tambah tengah" << endl;          cout << "   4. Ubah depan" << endl;          cout << "   5. Ubah belakang" << endl;          cout << "   6. Ubah tengah" << endl;          cout << "   7. Hapus depan" << endl;          cout << "   8. Hapus belakang" << endl;          cout << "   9. Hapus tengah" << endl;          cout << "  10. Hapus list" << endl;          cout << "  11. Tampilkan list" << endl;          cout << "  12. Cari nilai" << endl;          cout << "  0.  Keluar" << endl;          cout << "  Pilihan : ";          cin >> pilihan;          // pilihan menu          switch (pilihan)          {          case 1:              cout << "Masukan Nama       : ";              cin >> nama;              cout << "Masukan NIM        : ";              cin >> nim;              tambahDepan(head, nama, nim);              cout << "Data " << nama << " Berhasil Diinput" << endl << endl;              break;          case 2:              cout << "Masukan Nama       : ";              cin >> nama;              cout << "Masukan NIM        : ";              cin >> nim;              tambahBelakang(head, nama, nim);              cout << "Data " << nama << " Berhasil Diinput" << endl << endl;              break;          case 3:              cout << "Masukan Nama       : ";              cin >> nama;              cout << "Masukan NIM        : ";              cin >> nim;              cout << "Masukan Posisi     : ";              cin >> posisi;              tambahTengah(head, nama, nim, posisi);              cout << "Data " << nama << " Berhasil Diinput" << endl << endl;              break;          case 4:              cout << "Masukan Nama       : ";              cin >> nama;              cout << "Masukan NIM        : ";              cin >> nim;              ubahDepan(head, nama, nim);              break;          case 5:              cout << "Masukan Nama       : ";              cin >> nama;              cout << "Masukan NIM        : ";              cin >> nim;              ubahBelakang(head, nama, nim);              break;          case 6:              cout << "Masukan Nama       : ";              cin >> nama;              cout << "Masukan NIM        : ";              cin >> nim;              cout << "Masukan Posisi     : ";              cin >> posisi;              ubahTengah(head, nama, nim, posisi);              break;          case 7:              hapusDepan(head);              break;          case 8:              hapusBelakang(head);              break;          case 9:              cout << "Posisi : ";              cin >> posisi;              hapusTengah(head, posisi);              break;          case 10:              hapusList(head);              cout << "Data di List Berhasil Dihapus" << endl << endl;              break;          case 11:              tampilkanList(head);              break;          case 12:              cout << "Masukan NIM : ";              cin >> nim;              if (cariNilai(head, nim))              {                  cout << endl << "NIM " << nim << " Ditemukan" << endl << endl;              }              else              {                  cout << endl << "NIM " << nim << " Tidak Ditemukan" << endl << endl;              }              break;          case 13:              break;          default:              cout << "Pilihan tidak ada" << endl << endl;              break;          }      } while (pilihan != 0);      return 0;  } |

**Deskripsi:**

Program membuat daftar list nama dan NIM, dibuat menggunakan Program Single Linked List Non-Circular. Yang pertama membuat struct data kemudian dilanjut dengan membuat fungsi-fungsi seperti tambah data, ubah data, hapus data, tampilkan data, dan mencari data. Kemudian dibuat menu utama untuk memanggil fungsi-fungsi tadi yang telah dibuat. User tinggal memilih menu yang ada kemudian mengikuti perintah yang ditampilkan.

Contoh user bisa memilih menu 1 untuk menambahkan data awal, kemudian input datanya. Selanjutnya pilih menu 1 untuk data akhir, lalu inputkan datanya. Lalu pilih menu 11 untuk menampilkan hasil inputan. User juga bisa mengubah datanya, menghapus satu persatu ataupun menghapus seluruhnya. Bahkan bisa juga melakukan search NIM yang nantinya program akan memberitahu NIM yang dimaksud ada di dalam list atau tidak.

#### Masukan nama sesuai urutan

Text

Description automatically generated

#### Hapus data Karin



#### Tambah Cika diantara Joko dan Friska

Graphical user interface, text

Description automatically generated

#### Hapus data Joko

Text

Description automatically generated

#### Tambah Dimas di awal

Graphical user interface, text

Description automatically generated

#### Tambah Vina diantara Dimas dan Anda

Graphical user interface, text

Description automatically generated

#### Ubah Gabriel menjadi Jamal

Graphical user interface, text

Description automatically generated

#### Ubah Niken menjadi April

Graphical user interface, text

Description automatically generated

#### Tambah Budi di akhir

Graphical user interface, text

Description automatically generated

#### Tampilkan list

Text

Description automatically generated

# Kesimpulan

1. Bisa membuat structure
2. Belajar lebih dalam structure
3. Saya lebih mahir dalam menggunakan bahasa C++
4. Saya bisa melakukan problem solving bagi program yang error
5. Lebih paham dalam membuat program
6. Melatih daya pikir, imajinasi, dan langkah-langkah dalam membuat program
7. Structure dapat diandalkan dalam berbagai masalah
8. Membuat linked list
9. Belajar mengenai penyimpanan memori