**LAPORAN PRAKTIKUM**

**Modul IV**

**LINKED LIST CIRCULAR DAN NON CIRCULAR**

Logo

Description automatically generated

**Disusun oleh:**

Muhammad Irsyad : **2211102048**

**Dosen**

Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2023**

# BAB I

## Tujuan Pembelajaran

1. Memahami konsep struktur data
2. Mengembangkan keterampilan pemrograman
3. Meningkatkan efisiensi program
4. Meningkatkan keahlian algoritma
5. Menyelesaikan masalah yang lebih kompleks

# BAB II

# Dasar Teori

1. **Linked List Non Circular**

Linked list non circular 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. Linked list non circular dapat digambarkan sebagai berikut.

1. Linked List Circular

Linked list circular adalah struktur data linked list yang memiliki sifat berputar atau mengulang dari node head ke node tail. Dalam linked list circular, node tail tidak menunjuk ke ‘NULL’, melainkan mengarah kembali ke node head, sehingga membentuk lingkaran. Pada node tail dalam linked list circular, pointer menunjuk kembali ke node head.

Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu dalam pemutar musik, daftar pesan dalam antrian, atau penggunaan memori yang berulang dalam suatu aplikasi.

# BAB III

## LATIHAN KELAS – GUIDED

### Guided 1

**Source Code**

|  |
| --- |
| #include <iostream>  using namespace std;  struct Node  {  int data;  Node \*next;  };  Node \*head;  Node \*tail;  void init()  {  head = NULL;  tail = NULL;  }  bool isEmpty()  {  if (head == NULL)  return true;  else  return false;  }  void insertDepan(int nilai)  {  Node \*baru = new Node;  baru->data = nilai;  baru->next = NULL;  if (isEmpty() == true)  {  head = tail = baru;  tail->next = NULL;  }  else  {  tail->next = baru;  tail = baru;  }  }  void insertBelakang(int nilai)  {  Node \*baru = new Node;  baru->data = nilai;  baru->next = NULL;  if (isEmpty() == true)  {  head = tail = baru;  tail->next = NULL;  }  else  {  tail->next = baru;  tail = baru;  }  }  int hitungList()  {  Node \*hitung;  hitung = head;  int jumlah = 0;  while (hitung != NULL)  {  jumlah++;  hitung = hitung->next;  }  return jumlah;  }  void insertTengah(int data, int posisi)  {  if (posisi < 1 || posisi > hitungList())  {  cout << "Posisi diluar jangkauan" << endl;  }  else if (posisi == 1)  {  cout << "Posisi bukan posisi tengah" << endl;  }  else  {  Node \*baru, \*bantu;  baru = new Node();  baru  ->data = data;  // tranversing  bantu = head;  int nomor = 1;  while (nomor < posisi - 1)  {  bantu = bantu  ->next;  nomor++;  }  baru  ->next = bantu  ->next;  bantu  ->next = baru;  }  }  void hapusDepan()  {  Node \*hapus;  if (isEmpty() == false)  {  if (head  ->next != NULL)  {  hapus = head;  head = head->next;  delete hapus;  }  else  {  head = tail = NULL;  }  }  else  {  cout << "List kosong!" << endl;  }  }  void hapusBelakang()  {  Node \*hapus;  Node \*bantu;  if (isEmpty() == false)  {  if (head != tail)  {  hapus = tail;  bantu = head;  while (bantu  ->next != tail)  {  bantu = bantu  ->next;  }  tail = bantu;  tail->next = NULL;  delete hapus;  }  else  {  head = tail = NULL;  }  }  else  {  cout << "List kosong!" << endl;  }  }  void hapusTengah(int posisi)  {  Node \*bantu, \*hapus, \*sebelum;  if (posisi < 1 || posisi > hitungList())  {  cout << "Posisi di luar jangkauan" << endl;  }  else if (posisi == 1)  {  cout << "Posisi bukan posisi tengah" << endl;  }  else  {  int nomor = 1;  bantu = head;  while (nomor <= posisi)  {  if (nomor == posisi - 1)  {  sebelum = bantu;  }  if (nomor == posisi)  {  hapus = bantu;  }  bantu = bantu->next;  nomor++;  }  sebelum->next = bantu;  delete hapus;  }  }  void ubahDepan(int data)  {  if (isEmpty() == 0)  {  head->data = data;  }  else  {  cout << "List masih kosong!" << endl;  }  }  void ubahTengah(int data, int posisi)  {  Node \*bantu;  if (isEmpty() == 0)  {  if (posisi < 1 || posisi > hitungList())  {  cout << "Posisi di luar jangkauan" << endl;  }  else if (posisi == 1)  {  cout << "Posisi bukan posisi tengah" << endl;  }  else  {  bantu = head;  int nomor = 1;  while (nomor < posisi)  {  bantu = bantu->next;  nomor++;  }  bantu->data = data;  }  }  else  {  cout << "List masih kosong!" << endl;  }  }  void ubahBelakang(int data)  {  if (isEmpty() == 0)  {  tail->data = data;  }  else  {  cout << "List masih kosong!" << endl;  }  }  void clearList()  {  Node \*bantu, \*hapus;  bantu = head;  while (bantu != NULL)  {  hapus = bantu;  bantu = bantu->next;  delete hapus;  }  head = tail = NULL;  cout << "List berhasil terhapus!" << endl;  }  void tampil()  {  Node \*bantu;  bantu = head;  if (isEmpty() == false)  {  while (bantu != NULL)  {  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;  } |

**Screenshot Program**

A screenshot of a computer

Description automatically generated

**Deskripsi Program**

Kode program tersebut merupakan implementasi dari struktur data linked list dengan menggunakan bahasa pemrograman C++. Struktur data linked list adalah struktur data yang terdiri dari sejumlah simpul atau node yang saling berhubungan melalui pointer. Setiap node pada linked list terdiri dari data dan pointer ke simpul berikutnya.

Beberapa fungsi yang didefinisikan pada program tersebut adalah:

1. **init()** - fungsi ini digunakan untuk menginisialisasi nilai head dan tail dengan nilai null.
2. **isEmpty()** - fungsi ini mengembalikan nilai true jika linked list kosong dan false jika sebaliknya.
3. **insertDepan(int nilai)** - fungsi ini digunakan untuk menyisipkan nilai pada awal linked list.
4. **insertBelakang(int nilai)** - fungsi ini digunakan untuk menyisipkan nilai pada akhir linked list.
5. **hitungList()** - fungsi ini menghitung jumlah node pada linked list.
6. **insertTengah(int data, int posisi)** - fungsi ini digunakan untuk menyisipkan nilai pada posisi tertentu pada linked list.
7. **hapusDepan()** - fungsi ini digunakan untuk menghapus node pada awal linked list.
8. **hapusBelakang()** - fungsi ini digunakan untuk menghapus node pada akhir linked list.
9. **hapusTengah(int posisi)** - fungsi ini digunakan untuk menghapus node pada posisi tertentu pada linked list.
10. **ubahDepan(int data)** - fungsi ini digunakan untuk mengubah nilai pada node pertama.
11. **ubahTengah(int data, int posisi)** - fungsi ini digunakan untuk mengubah nilai pada node pada posisi tertentu pada linked list.
12. **ubahBelakang(int data)** - fungsi ini digunakan untuk mengubah nilai pada node terakhir.
13. **clearList()** - fungsi ini digunakan untuk menghapus seluruh isi dari linked list.

### Guided 2

**Source Code**

|  |
| --- |
| #include <iostream>  using namespace std;  struct Node  {  string data;  Node \*next;  };  Node \*head, \*tail, \*baru, \*bantu, \*hapus;  void init()  {  head = NULL;  tail = head;  }  int isEmpty()  {  if (head == NULL)  return 1;  else  return 0;  }  void buatNode(string data)  {  baru = new Node;  baru->data = data;  baru->next = NULL;  }  int hitungList()  {  bantu = head;  int jumlah = 0;  while (bantu != NULL)  {  jumlah++;  bantu = bantu->next;  }  return jumlah;  }  void insertDepan(string data)  {  buatNode(data);  if (isEmpty() == 1)  {  head = baru;  tail = head;  baru->next = head;  }  else  {  while (tail->next != head)  {  tail = tail->next;  }  baru->next = head;  head = baru;  tail->next = head;  }  }  void insertBelakang(string data)  {  buatNode(data);  if (isEmpty() == 1)  {  head = baru;  tail = head;  baru->next = head;  }  else  {  while (tail->next != head)  {  tail = tail->next;  }  tail->next = baru;  baru->next = head;  }  }  void insertTengah(string data, int posisi)  {  if (isEmpty() == 1)  {  head = baru;  tail = head;  baru->next = head;  }  else  {  baru->data = data;  int nomor = 1;  bantu = head;  while (nomor < posisi - 1)  {  bantu = bantu->next;  nomor++;  }  baru->next = bantu->next;  bantu->next = baru;  }  }  void hapusDepan()  {  if (isEmpty() == 0)  {  hapus = head;  tail = head;  if (hapus->next == head)  {  head = NULL;  tail = NULL;  delete hapus;  }  else  {  while (tail->next != hapus)  {  tail = tail->next;  }  head = head->next;  tail->next = head;  hapus->next = NULL;  delete hapus;  }  }  else  {  cout << "List masih kosong!" << endl;  }  }  void hapusBelakang()  {  if (isEmpty() == 0)  {  hapus = head;  tail = head;  if (hapus->next == head)  {  head = NULL;  tail = NULL;  delete hapus;  }  else  {  while (hapus->next != head)  {  hapus = hapus->next;  }  while (tail->next != hapus)  {  tail = tail->next;  }  tail->next = head;  hapus->next = NULL;  delete hapus;  }  }  else  {  cout << "List masih kosong!" << endl;  }  }  void hapusTengah(int posisi)  {  if (isEmpty() == 0)  {  int nomor = 1;  bantu = head;  while (nomor < posisi - 1)  {  bantu = bantu->next;  nomor++;  }  hapus = bantu->next;  bantu->next = hapus->next;  delete hapus;  }  else  {  cout << "List masih kosong!" << endl;  }  }  void clearList()  {  if (head != NULL)  {  hapus = head->next;  while (hapus != head)  {  bantu = hapus->next;  delete hapus;  hapus = bantu;  }  delete head;  head = NULL;  }  cout << "List berhasil terhapus!" << endl;  }  void tampil()  {  if (isEmpty() == 0)  {  tail = head;  do  {  cout << tail->data << ends;  tail = tail->next;  } while (tail != 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;  } |

**Screenshot Program**

A screenshot of a computer

Description automatically generated

**Deskripsi Program**

Program di atas adalah implementasi dari struktur data linked list dalam bahasa C++. Pada program ini, terdapat beberapa fungsi dan prosedur untuk mengelola linked list, yaitu:

* **struct Node** adalah struktur data yang berisi dua atribut, yaitu **data** bertipe string yang menyimpan nilai dari suatu node, dan **next** bertipe pointer ke Node yang menyimpan alamat node selanjutnya dalam linked list.
* **Node \*head, \*tail, \*baru, \*bantu, \*hapus;** adalah deklarasi variabel pointer untuk menunjuk ke node-head, node-tail, node-baru (yang akan ditambahkan ke linked list), node-bantu (digunakan untuk melakukan iterasi pada linked list), dan node-hapus (yang akan dihapus dari linked list).
* **void init()** adalah prosedur untuk menginisialisasi linked list dengan membuat **head** dan **tail** bernilai NULL.
* **int isEmpty()** adalah fungsi yang mengembalikan nilai 1 jika linked list kosong, dan nilai 0 jika tidak kosong.
* **void buatNode(string data)** adalah prosedur untuk membuat node baru dengan nilai **data** yang diberikan.
* **int hitungList()** adalah fungsi untuk menghitung jumlah node dalam linked list.
* **void insertDepan(string data)** adalah prosedur untuk menambahkan node baru pada bagian depan (head) linked list.
* **void insertBelakang(string data)** adalah prosedur untuk menambahkan node baru pada bagian belakang (tail) linked list.
* **void insertTengah(string data, int posisi)** adalah prosedur untuk menambahkan node baru pada posisi tertentu dalam linked list.
* **void hapusDepan()** adalah prosedur untuk menghapus node pada bagian depan (head) linked list.
* **void hapusBelakang()** adalah prosedur untuk menghapus node pada bagian belakang (tail) linked list.
* **void hapusTengah(int posisi)** adalah prosedur untuk menghapus node pada posisi tertentu dalam linked list.
* **void clearList()** adalah prosedur untuk menghapus seluruh node dalam linked list.
* **void tampil()** adalah prosedur untuk menampilkan nilai dari seluruh node dalam linked list.

Di dalam fungsi **main()**, program ini melakukan beberapa operasi pada linked list seperti menambahkan node pada bagian depan, belakang, dan tengah, menghapus node pada bagian depan, belakang, dan tengah, serta menampilkan isi dari linked list setelah operasi dilakukan.

## TUGAS – UNGUIDED

### Unguided 1

**Source Code**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  struct Mahasiswa  {  string nama;  string nim;  Mahasiswa \*next;  };  Mahasiswa \*head = NULL;  Mahasiswa \*tail = NULL;  void tambahDepan()  {  Mahasiswa \*new\_mhs = new Mahasiswa;  cout << "Masukkan Nama Mahasiswa: ";  cin >> new\_mhs->nama;  cout << "Masukkan NIM Mahasiswa: ";  cin >> new\_mhs->nim;  new\_mhs->next = NULL;  if (head == NULL)  {  head = new\_mhs;  tail = new\_mhs;  }  else  {  new\_mhs->next = head;  head = new\_mhs;  }  cout << "Mahasiswa berhasil ditambahkan di depan.\n\n";  }  void tambahBelakang()  {  Mahasiswa \*new\_mhs = new Mahasiswa;  cout << "Masukkan Nama Mahasiswa: ";  cin >> new\_mhs->nama;  cout << "Masukkan NIM Mahasiswa: ";  cin >> new\_mhs->nim;  new\_mhs->next = NULL;  if (head == NULL)  {  head = new\_mhs;  tail = new\_mhs;  }  else  {  tail->next = new\_mhs;  tail = new\_mhs;  }  cout << "Mahasiswa berhasil ditambahkan di belakang.\n\n";  }  void tambahTengah()  {  int posisi;  cout << "Masukkan Posisi yang ingin ditambahkan: ";  cin >> posisi;  if (posisi == 1)  {  tambahDepan();  }  else  {  Mahasiswa \*new\_mhs = new Mahasiswa;  cout << "Masukkan Nama Mahasiswa: ";  cin >> new\_mhs->nama;  cout << "Masukkan NIM Mahasiswa: ";  cin >> new\_mhs->nim;  new\_mhs->next = NULL;  Mahasiswa \*temp = head;  for (int i = 1; i < posisi - 1; i++)  {  temp = temp->next;  }  new\_mhs->next = temp->next;  temp->next = new\_mhs;  cout << "Mahasiswa berhasil ditambahkan di tengah.\n\n";  }  }  void ubahDepan()  {  if (head == NULL)  {  cout << "Linked List kosong.\n\n";  }  else  {  cout << "Data Mahasiswa di Depan: " << head->nama << " - " << head->nim << "\n";  cout << "Masukkan Nama Mahasiswa Baru: ";  cin >> head->nama;  cout << "Masukkan NIM Mahasiswa Baru: ";  cin >> head->nim;  cout << "Data Mahasiswa berhasil diubah di depan.\n\n";  }  }  void ubahBelakang()  {  if (head == NULL)  {  cout << "Linked List kosong.\n\n";  }  else  {  cout << "Data Mahasiswa di Belakang: " << tail->nama << " - " << tail->nim << "\n";  cout << "Masukkan Nama Mahasiswa Baru: ";  cin >> tail->nama;  cout << "Masukkan NIM Mahasiswa Baru: ";  cin >> tail->nim;  cout << "Data Mahasiswa berhasil diubah di belakang.\n\n";  }  }  void ubahTengah()  {  int posisi;  cout << "Masukkan Posisi yang ingin diubah: ";  cin >> posisi;  Mahasiswa \*temp = head;  for (int i = 1; i < posisi; i++)  {  temp = temp->next;  }  cout << "Data Mahasiswa di Posisi " << posisi << ": " << temp->nama << " - " << temp->nim << "\n";  cout << "Masukkan Nama Mahasiswa Baru: ";  cin >> temp->nama;  cout << "Masukkan NIM Mahasiswa Baru: ";  cin >> temp->nim;  cout << "Data Mahasiswa berhasil diubah di tengah.\n\n";  }  void hapusDepan()  {  if (head == NULL)  {  cout << "Linked List kosong.\n\n";  }  else  {  Mahasiswa \*temp = head;  head = head->next;  delete temp;  cout << "Mahasiswa berhasil dihapus di depan.\n\n";  }  }  void hapusBelakang()  {  if (head == NULL)  {  cout << "Linked List kosong.\n\n";  }  else if (head == tail)  {  delete head;  head = NULL;  tail = NULL;  cout << "Mahasiswa berhasil dihapus di belakang.\n\n";  }  else  {  Mahasiswa \*temp = head;  while (temp->next != tail)  {  temp = temp->next;  }  delete tail;  tail = temp;  tail->next = NULL;  cout << "Mahasiswa berhasil dihapus di belakang.\n\n";  }  }  void hapusTengah()  {  int posisi;  cout << "Masukkan Posisi yang ingin dihapus: ";  cin >> posisi;  if (posisi == 1)  {  hapusDepan();  }  else if (head == tail)  {  delete head;  head = NULL;  tail = NULL;  cout << "Mahasiswa berhasil dihapus di tengah.\n\n";  }  else  {  Mahasiswa \*temp1 = head;  for (int i = 1; i < posisi - 1; i++)  {  temp1 = temp1->next;  }  Mahasiswa \*temp2 = temp1->next;  temp1->next = temp2->next;  delete temp2;  cout << "Mahasiswa berhasil dihapus di tengah.\n\n";  }  }  void hapusList()  {  if (head == NULL)  {  cout << "Linked List kosong.\n\n";  }  else  {  while (head != NULL)  {  Mahasiswa \*temp = head;  head = head->next;  delete temp;  }  head = NULL;  tail = NULL;  cout << "Seluruh data mahasiswa berhasil dihapus.\n\n";  }  }  void tampilkan()  {  if (head == NULL)  {  cout << "Linked List kosong.\n\n";  }  else  {  cout << "Daftar Mahasiswa:\n";  Mahasiswa \*temp = head;  while (temp != NULL)  {  cout << temp->nama << " - " << temp->nim << "\n";  temp = temp->next;  }  cout << "\n";  }  }  int main()  {  int pilihan;  do  {  cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR\n";  cout << "1. Tambah Depan\n";  cout << "2. Tambah Belakang\n";  cout << "3. Tambah Tengah\n";  cout << "4. Ubah Depan\n";  cout << "5. Ubah Belakang\n";  cout << "6. Ubah Tengah\n";  cout << "7. Hapus Depan\n";  cout << "8. Hapus Belakang\n";  cout << "9. Hapus Tengah\n";  cout << "10. Hapus List\n";  cout << "11. Tampilkan\n";  cout << "0. Keluar\n";  cout << "Pilih Operasi: ";  cin >> pilihan;  switch (pilihan)  {  case 1:  tambahDepan();  break;  case 2:  tambahBelakang();  break;  case 3:  tambahTengah();  break;  case 4:  ubahDepan();  break;  case 5:  ubahBelakang();  break;  case 6:  ubahTengah();  break;  case 7:  hapusDepan();  break;  case 8:  hapusBelakang();  break;  case 9:  hapusTengah();  break;  case 10:  hapusList();  break;  case 11:  tampilkan();  break;  case 0:  break;  default:  cout << "Pilihan tidak tersedia.\n\n";  }  } while (pilihan != 0);  return 0;  } |

**Screenshot Program**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Deskripsi Program**

Program di atas adalah program linked list sederhana dalam bahasa pemrograman C++. Linked list adalah salah satu struktur data yang terdiri dari elemen-elemen yang terhubung satu sama lain dengan menggunakan pointer. Setiap elemen pada linked list terdiri dari dua bagian yaitu data dan pointer yang menunjuk ke elemen selanjutnya.

Program ini terdiri dari beberapa fungsi yang berfungsi untuk menambah, mengubah, dan menghapus elemen pada linked list. Setiap elemen pada linked list direpresentasikan sebagai struktur Mahasiswa yang terdiri dari nama, nim, dan pointer next yang menunjuk ke elemen selanjutnya.

Fungsi-fungsi pada program ini antara lain:

* tambahDepan() : menambahkan elemen pada posisi depan linked list.
* tambahBelakang() : menambahkan elemen pada posisi belakang linked list.
* tambahTengah() : menambahkan elemen pada posisi tertentu pada linked list.
* ubahDepan() : mengubah data elemen pada posisi depan linked list.
* ubahBelakang() : mengubah data elemen pada posisi belakang linked list.
* ubahTengah() : mengubah data elemen pada posisi tertentu pada linked list.
* hapusDepan() : menghapus elemen pada posisi depan linked list.
* hapusBelakang() : menghapus elemen pada posisi belakang linked list.
* hapusTengah() : menghapus elemen pada posisi tertentu pada linked list.

Selain itu, terdapat dua variabel global yaitu head dan tail yang digunakan untuk menyimpan elemen pertama dan terakhir pada linked list. Variabel ini bertipe pointer dan awalnya diinisialisasi dengan NULL.

**SOAL!**

Buatlah program menu Linked List Non Circular untuk menyimpan **Nama** dan

**NIM mahasiswa**, dengan menggunakan inputan dari user.

Lakukan operasi berikut:

1. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah)

[Nama\_Anda] [NIM\_Anda] → data pertama yang diinput adalah Nama & NIM Anda

|  |  |
| --- | --- |
| **Nama** | **NIM** |
| **Alvin** | **22200001** |
| **Candra** | **22200002** |
| **Niken** | **22200005** |
| **Joko** | **22200008** |
| **Friska** | **22200015** |
| **Gabrie** | **22200040** |
| **Karin** | **22200020** |

A screenshot of a computer

Description automatically generated

1. Hapus data Karin

A screenshot of a computer

Description automatically generated

1. Tambahkan data berikut diantara data Joko dan Friska:

**Cika 22200003**

A screenshot of a computer

Description automatically generated

1. Hapus data Joko

A screenshot of a computer

Description automatically generated

1. Tambahkan data berikut di awal:

**Dimas 22200010**

A screenshot of a computer

Description automatically generated

1. Tambahkan data berikut diantara data Dimas dan Anda:

**Vina 22200022**

A screenshot of a computer

Description automatically generated

1. Ubah data Gabriel menjadi data berikut:

**Jamal 22200033**

A screenshot of a computer

Description automatically generated

Ubah data Niken menjadi data berikut:

**April 22200017**

A screenshot of a computer

Description automatically generated

Tambahkan data berikut di akhir:

**Budi 22200000**

A screenshot of a computer

Description automatically generated

Ubah data NIM Candra menjadi:

**21200055**

A screenshot of a computer

Description automatically generated

Tampilkan seluruh data

A screenshot of a computer

Description automatically generated

# BAB IV

## KESIMPULAN

Program di atas merupakan implementasi dari struktur data linked list dalam bahasa pemrograman C++. Linked list adalah struktur data yang terdiri dari simpul atau node yang terhubung satu sama lain melalui pointer. Setiap node pada linked list memiliki data dan pointer ke simpul berikutnya.

Program ini memiliki beberapa fungsi yang berfungsi untuk mengelola linked list, seperti menambahkan, mengubah, dan menghapus elemen pada linked list. Beberapa fungsi yang didefinisikan pada program tersebut adalah **init()**, **isEmpty()**, **insertDepan()**, **insertBelakang()**, **hitungList()**, **insertTengah()**, **hapusDepan()**, **hapusBelakang()**, **hapusTengah()**, **ubahDepan()**, **ubahTengah()**, **ubahBelakang()**, dan **clearList()**.

Dalam fungsi **main()**, program melakukan beberapa operasi pada linked list seperti menambahkan, mengubah, dan menghapus elemen pada linked list serta menampilkan isi dari linked list setelah operasi dilakukan. Program ini juga memiliki struktur data **Node** yang terdiri dari data dan pointer ke simpul berikutnya. Sedangkan struktur **Mahasiswa** merupakan representasi setiap elemen pada linked list yang terdiri dari nama, nim, dan pointer next yang menunjuk ke elemen selanjutnya.

Top of Form