**LAPORAN PRAKTIKUM STRUKTUR**

**DATA DAN ALGORITMA**

**MODUL 4**

**LINKED LIST CIRCULAR DAN NON CIRCULAR**

Sebuah gambar berisi simbol, logo, Grafis, deasin

Deskripsi dibuat secara otomatis

**Disusun oleh:**

Faqih Abdullah

2311102048

**Dosen Pengampu**

Wahyu Andi Saputra, SPd., M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKUTLAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2024**

1. **DASAR TEORI**
2. Linked list Non Circular

Linked lisrt non circular merupakan linked list dengan node pertama (head) dan node terakhir (tail) yang tidak terhubung. Pointer terakhir (tail) pada linked list ini selalu bernilai ‘***NULL***’ sebagai pertanda data terakhir di dalam listnya. Linked list non circular dapat digambarkan sebagai berikut.

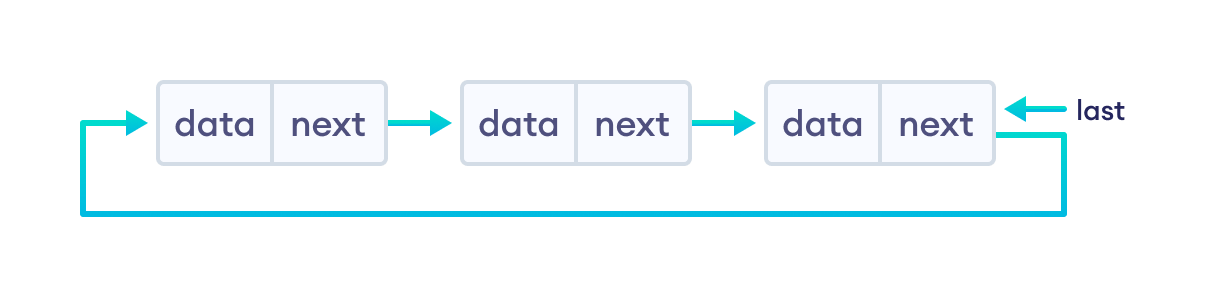
Sebuah gambar berisi Font, sketsa, putih, garis

Deskripsi dibuat secara otomatis

Gambar : linked list non circular

1. Linked List Circular

Linked list circular merupakan variasi dari linked list Dimana elemen pertama menunjuk ke elemen elemen terakhir dan elemen terakhir menunjuk ke elemen pertama. gambar linked list circular sebagai berikut



Gambar : Linked list circular

1. **GUIDED**
2. Linked list Non circular

**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() {      return head == NULL;  }  void insertDepan(int *nilai*) {  *Node* \*baru = **new** *Node*;      baru->data = *nilai*;      baru->next = NULL;      if (isEmpty()) {          head = tail = baru;      } else {          baru->next = head;          head = baru;      }  }  void insertBelakang(int *nilai*) {  *Node* \*baru = **new** *Node*;      baru->data = *nilai*;      baru->next = NULL;      if (isEmpty()) {          head = tail = baru;      } else {          tail->next = baru;          tail = baru;      }  }  int hitungList() {  *Node* \*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 = **new** *Node*();          baru->data = *data*;  *Node* \*bantu = head;          int nomor = 1;          while (nomor < *posisi* - 1) {              bantu = bantu->next;              nomor++;          }          baru->next = bantu->next;          bantu->next = baru;      }  }  void hapusDepan() {      if (!isEmpty()) {  *Node* \*hapus = head;          if (head->next != NULL) {              head = head->next;          } else {              head = tail = NULL;          }  **delete** hapus;      } else {          cout << "List kosong!" << endl;      }  }  void hapusBelakang() {      if (!isEmpty()) {  *Node* \*hapus = tail;          if (head != tail) {  *Node* \*bantu = head;              while (bantu->next != tail) {                  bantu = bantu->next;              }              tail = bantu;              tail->next = NULL;          } else {              head = tail = NULL;          }  **delete** hapus;      } else {          cout << "List kosong!" << endl;      }  }  void hapusTengah(int *posisi*) {      if (*posisi* < 1 || *posisi* > hitungList()) {          cout << "Posisi di luar jangkauan" << endl;      } else if (*posisi* == 1) {          cout << "Posisi bukan posisi tengah" << endl;      } else {  *Node* \*bantu = head;  *Node* \*hapus;  *Node* \*sebelum = NULL;          int nomor = 1;          while (nomor < *posisi*) {              sebelum = bantu;              bantu = bantu->next;              nomor++;          }          hapus = bantu;          if (sebelum != NULL) {              sebelum->next = bantu->next;          } else {              head = bantu->next;          }  **delete** hapus;      }  }  void ubahDepan(int *data*) {      if (!isEmpty()) {          head->data = *data*;      } else {          cout << "List masih kosong!" << endl;      }  }  void ubahTengah(int *data*, int *posisi*) {      if (!isEmpty()) {          if (*posisi* < 1 || *posisi* > hitungList()) {              cout << "Posisi di luar jangkauan" << endl;          } else if (*posisi* == 1) {              cout << "Posisi bukan posisi tengah" << endl;          } else {  *Node* \*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()) {          tail->data = *data*;      } else {          cout << "List masih kosong!" << endl;      }  }  void clearList() {  *Node* \*bantu = head;  *Node* \*hapus;      while (bantu != NULL) {          hapus = bantu;          bantu = bantu->next;  **delete** hapus;      }      head = tail = NULL;      cout << "List berhasil terhapus!" << endl;  }  void tampil() {  *Node* \*bantu = head;      if (!isEmpty()) {          while (bantu != NULL) {              cout << bantu->data << " ";              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 OUTPUT

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

DESKRIPSI PROGRAM

*Program merupakan penerapan dari sebuah single linked list non circular dengan beberapa operasi dasar seperti insert (depan, belakang, Tengah), delete(depan, belakang, Tengah), modify(depan, belakang, Tengah).*

*1. struct node: mewakili sebuah node dalam linked list. Setiap node memiliki dua anggotta , yaitu data ( nilai yang disimpan) dan pointer next (pointer yang menunjuk ke node berikutnya).*

*2. Inisialisasi linked list: fungsi init() digunakan untuk menginisialisasi linked list dengan mengatur pointer head dan tail menjadi NULL.*

*3. isEmpty() : digunakan untuk memeriksa apakah linked list kosong atau tidak.*

*4. - insertDepan() : untuk menambahkan node baru dibagian depan linked list.*

*- insertBelakang() : untuk menambahkan node baru di bagian belakang linked list.*

*- insertTengah() : untuk menambahkan node baru di posisi tertentu di Tengah linked list.*

*5. hitungList() : untuk menghitung jumlah node dalam linked list.*

*6 delete operation :*

*- hapusDepan() : untuk menghapus node pertama linked list.*

*- hapusBelakang() : untuk menghapus node terakhir linked list.*

*- hapusTengah() : untuk menghapus node bagian tengah tertentu linked list.*

*7. modify operation :*

*- ubahDepan() : untuk mengubah nilai data pada node pertama.*

*- ubahBelakang() : untuk mengubaj nilai data pada node terakhir.*

*- ubahTengah() : untuk mengubah nilai data pada posisi tertentu di Tengah linked list.*

*8. clearList() : menghapus seluruh node dari LL.*

*9. tampil() : menampilkan isi LL.*

*10. main() : program melakukan inisialisasi linked list, memanggil operasi-operasi di LL dan menampilkan hasilnya.*

2. Linked list circular

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() {      return head == NULL;  }  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()) {          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()) {          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()) {          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()) {          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()) {          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()) {          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()) {          tail = head;          do {              cout << tail->data << " ";              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 OUTPUT

Sebuah gambar berisi teks, software, Software multimedia, Ikon komputer

Deskripsi dibuat secara otomatis

DESKRIPSI PROGRAM

*struct node : mewakili sebuah node dalam linked list. Setiap node memiliki dua anggota, yaitu data (nilai yang disimpan dan pointer next (pointer yang menunjuk ke node berikutnya).*

*Berikut fungsi fungsi yang ada di program diatas*

*1. init() : digunakan untuk menginisialisasi LL dengan mengatur pointer head dan tail menjadi NULL.*

*2. isEmpty() : untuk memeriksa apakah LL kosong atau tidak.*

*3. buatNode() : untuk sebuah node baru dengan nilai data yang ditentukan.*

*4. hitungList() : untuk megitung jumlah node dalam LL.*

*5. ada 3 operasi insert, yaitu “insertDepan(), insertBelakang(). insertTengah()”.*

*- insertDepan(): untuk menambahkan node baru dibagian depan dalam LL*

*- insertBelakang() : untuk menambahkan node baru dibagian belakang LL.*

*- insertTengah() : untuk menambahkan node baru dibagian tengah posisi yang ditentukan dalam LL.*

*6. ada 3 operasi delete*

*- hapusDepan() : untuk menghapus node pertama dalam LL.*

*- hapusBelakang() : untuk menghapus node terakhir dalam LL.*

*- hapusTengah() : untuk menghapus node urutan tertentu dibagian Tengah dalam LL.*

*7. clearList(): untuk menghapus seluruh node dari linked list.*

*8. tampil() : untuk menampilkan isi LL.*

*9 main() : program melakukan inisialisasi linked list, memanggil operasi-operasi di LL dan menampilkan hasilnya. Program ini mengimplementasikan circular linked list, Dimana tail selalu menunjuk Kembali ke head, sehingga membentuk lingkaran.*

1. **UNGUIDED**
2. Buatlah program menu linked list non circular untuk menyimpan **Nama** dan **NIM** Mahasiswa, dengan menggunakan input dari user  
   Tampilan Menu:

PROGRAM SINGLE LINKED LIST NON-CIRCULAR

* + - 1. Tambah Depan
      2. Tambah Belakang
      3. Tambah Tengah
      4. Ubah Depan
      5. Ubah Belakang
      6. Ubah Tengah
      7. Hapus Depan
      8. Hapus Belakang
      9. Hapus Tengah
      10. Hapus List
      11. TAMPILKAN

0. KELUAR

Pilih Operasi :

* + - Tampilan Operasi Tambah:

-Tambah Depan

Masukkan Nama : Masukkan NIM :

Data telah ditambahkan

-Tambah Tengah

Masukkan Nama :

Masukkan NIM :

Masukkan Posisi :

Data telah ditambahkan

* + - Tampilan Operasi Hapus:

-Hapus Belakang

Data (nama mahasiswa yang dihapus) berhasil dihapus

-Hapus Tengah

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus

* + - Tampilan Operasi Ubah:

|  |
| --- |
| -Ubah Belakang  Masukkan nama : Masukkan NIM :  Data (nama lama) telah diganti dengan data (nama baru) |

-Ubah Belakang

Masukkan nama :

Masukkan NIM :

Masukkan posisi :

Data (nama lama) telah diganti dengan data (nama baru)

* + - Tampilan Operasi Tampil Data:

DATA MAHASISWA

NAMA NIM

Nama1 NIM1

Nama2 NIM2

**\*Buat tampilan output sebagus dan secantik mungkin sesuai kreatifitas anda masing-masing, jangan terpaku pada contoh output yang diberikan**

**SOURCE CODE**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  // Deklarasi struktur Node  struct *Node* {  *string* nama;  *string* nim;  *Node*\* next;  };  // Deklarasi kelas LinkedList  class *LinkedList* {  private:  *Node*\* head;  public:      LinkedList() {          head = nullptr;      }      // Fungsi untuk menambahkan node di awal linked list      void addFront(*string* *nama*, *string* *nim*) {  *Node*\* newNode = **new** *Node*;          newNode->nama = *nama*;          newNode->nim = *nim*;          newNode->next = head;          head = newNode;          cout << "Data telah ditambahkan." << endl;      }      // Fungsi untuk menambahkan node di belakang linked list      void addBack(*string* *nama*, *string* *nim*) {  *Node*\* newNode = **new** *Node*;          newNode->nama = *nama*;          newNode->nim = *nim*;          newNode->next = nullptr;          if (head == nullptr) {              head = newNode;              cout << "Data telah ditambahkan." << endl;              return;          }  *Node*\* temp = head;          while (temp->next != nullptr) {              temp = temp->next;          }          temp->next = newNode;          cout << "Data telah ditambahkan." << endl;      }      // Fungsi untuk menambahkan node di tengah linked list      void addMiddle(int *pos*, *string* *nama*, *string* *nim*) {  *Node*\* newNode = **new** *Node*;          newNode->nama = *nama*;          newNode->nim = *nim*;          if (*pos* == 1) {              newNode->next = head;              head = newNode;              cout << "Data telah ditambahkan." << endl;              return;          }  *Node*\* temp = head;          for (int i = 1; i < *pos* - 1; i++) {              if (temp == nullptr) {                  cout << "Posisi yang dimaksud tidak valid." << endl;                  return;              }              temp = temp->next;          }          newNode->next = temp->next;          temp->next = newNode;          cout << "Data telah ditambahkan." << endl;      }      // Fungsi untuk mengubah data di depan linked list      void modifyFront(*string* *nama*, *string* *nim*) {          if (head == nullptr) {              cout << "Linked list kosong. Tidak ada data yang dapat diubah." << endl;              return;          }          cout << "Data " << head->nama << " telah diganti dengan data " << *nama* << "." << endl;          head->nama = *nama*;          head->nim = *nim*;      }      // Fungsi untuk mengubah data di belakang linked list      void modifyBack(*string* *nama*, *string* *nim*) {          if (head == nullptr) {              cout << "Linked list kosong. Tidak ada data yang dapat diubah." << endl;              return;          }  *Node*\* temp = head;          while (temp->next != nullptr) {              temp = temp->next;          }          cout << "Data " << temp->nama << " telah diganti dengan data " << *nama* << "." << endl;          temp->nama = *nama*;          temp->nim = *nim*;      }      // Fungsi untuk mengubah data di tengah linked list      void modifyMiddle(int *pos*, *string* *nama*, *string* *nim*) {          if (head == nullptr) {              cout << "Linked list kosong. Tidak ada data yang dapat diubah." << endl;              return;          }  *Node*\* temp = head;          int count = 0;          while (temp != nullptr) {              count++;              if (count == *pos*) {                  cout << "Data " << temp->nama << " telah diganti dengan data " << *nama* << "." << endl;                  temp->nama = *nama*;                  temp->nim = *nim*;                  return;              }              temp = temp->next;          }          cout << "Posisi yang dimaksud tidak valid." << endl;      }      // Fungsi untuk menghapus data di depan linked list      void deleteFront() {          if (head == nullptr) {              cout << "Linked list kosong. Tidak ada data yang dapat dihapus." << endl;              return;          }  *Node*\* temp = head;          head = head->next;  **delete** temp;          cout << "Data depan berhasil dihapus." << endl;      }      // Fungsi untuk menghapus data di tengah linked list      void deleteMiddle(int *pos*) {          if (head == nullptr) {              cout << "Linked list kosong. Tidak ada data yang dapat dihapus." << endl;              return;          }  *Node*\* temp = head;  *Node*\* prev = nullptr;          int count = 0;          while (temp != nullptr) {              count++;              if (count == *pos*) {                  if (prev == nullptr) { // jika yang dihapus adalah head                      head = head->next;                  } else {                      prev->next = temp->next;                  }              cout << "Data " << temp->nama << " berhasil dihapus." << endl;  **delete** temp;                  return;              }              prev = temp;              temp = temp->next;          }          cout << "Posisi yang dimaksud tidak valid." << endl;      }      // Fungsi untuk menghapus data di belakang linked list      void deleteBack() {          if (head == nullptr) {              cout << "Linked list kosong. Tidak ada data yang dapat dihapus." << endl;              return;          }          if (head->next == nullptr) {              cout << "Data " << head->nama << " berhasil dihapus." << endl;  **delete** head;              head = nullptr;              return;          }  *Node*\* temp = head;          while (temp->next->next != nullptr) {              temp = temp->next;          }          cout << "Data " << temp->next->nama << " berhasil dihapus." << endl;  **delete** temp->next;          temp->next = nullptr;      }      // Fungsi untuk menampilkan seluruh data mahasiswa dalam linked list      void display() {          if (head == nullptr) {              cout << "Linked list kosong." << endl;              return;          }  *Node*\* temp = head;          cout << "Data Mahasiswa:" << endl;          cout << "Nama <=====> \tNIM" << endl;          while (temp != nullptr) {              cout << "Nama: " << temp->nama << "\tNIM: " << temp->nim << endl;              temp = temp->next;          }          cout << endl;      }      // Fungsi untuk menghapus seluruh data dalam linked list      void clear() {          while (head != nullptr) {  *Node*\* temp = head;              head = head->next;  **delete** temp;          }          cout << "Seluruh data berhasil dihapus." << endl;      }  };  int main() {  *LinkedList* list;      int choice, pos;  *string* nama, nim;      cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;      do {          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 Tengah" << endl;          cout << "9. Hapus Belakang" << endl;          cout << "10. Tampilkan" << endl;          cout << "11. Hapus List" << endl;          cout << "0. Keluar" << endl;          cout << "Pilih Operasi: ";          cin >> choice;          switch (choice) {              case 1:                  cout << "Masukkan Nama : ";                  cin >> nama;                  cout << "Masukkan NIM : ";                  cin >> nim;                  list.addFront(nama, nim);                  break;              case 2:                  cout << "Masukkan Nama : ";                  cin >> nama;                  cout << "Masukkan NIM : ";                  cin >> nim;                  list.addBack(nama, nim);                  break;              case 3:                  cout << "Masukkan nama : ";                  cin >> nama;                  cout << "Masukkan NIM : ";                  cin >> nim;                  cout << "Masukkan posisi: ";                  cin >> pos;                  list.addMiddle(pos, nama, nim);                  break;              case 4:                  cout << "Masukkan nama baru mahasiswa depan: ";                  cin >> nama;                  cout << "Masukkan NIM baru mahasiswa depan: ";                  cin >> nim;                  list.modifyFront(nama, nim);                  break;              case 5:                  cout << "Masukkan nama : ";                  cin >> nama;                  cout << "Masukkan NIM : ";                  cin >> nim;                  list.modifyBack(nama, nim);                  break;              case 6:                  cout << "Masukkan nama baru mahasiswa: ";                  cin >> nama;                  cout << "Masukkan NIM baru mahasiswa: ";                  cin >> nim;                  cout << "Masukkan posisi: ";                  cin >> pos;                  list.modifyMiddle(pos, nama, nim);                  break;              case 7:                  list.deleteFront();                  break;              case 8:                  cout << "Masukkan posisi : ";                  cin >> pos;                  list.deleteMiddle(pos);                  break;              case 9:                  list.deleteBack();                  break;              case 10:                  list.display();                  break;              case 11:                  list.clear();                  break;              case 0:                  cout << "Program selesai." << endl;                  break;              default:                  cout << "Pilihan tidak valid. Silakan coba lagi." << endl;          }      } while (choice != 0);      return 0;  } |

SCREENSHOT OUTPUT

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

1. Setelah membuat menu tersebut, masukkan data sesuai urutan berikut lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang, atau Tengah).

|  |  |
| --- | --- |
| **Nama** | **NIM** |
| **Jawad** | **23300001** |
| **[Nama Anda]** | **[NIM Anda]** |
| **Farrel** | **23300003** |
| **Denis** | **23300005** |
| **Anis** | **23300008** |
| **Bowo** | **23300015** |
| **Gahar** | **23300040** |
| **Udin** | **23300048** |
| **Ucok** | **23300050** |
| **Budi** | **23300099** |

SCREENSHOT OUTPUT

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

DESKRIPSI PROGRAM

*Output diatas merupakan hasil keluaran dari input user yang berupa Nama dan Nim mahasiswa. di urutan kedua merupakan nama dan NIM saya.*

1. Lakukan perintah berikut :

a) tambahkan data berikut diantara Farrel dan Denis: **wati 2330004**

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

Deskripsi Program

*Menambahkan wati pada posisi ke empat. Setelah farrel dan dennis.*

B) hapus data dennis  
  
Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

C) Tambahkan data berikut di awal: **Owi 2330000**

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

D) Tambahkan data berikut di akhir: **David 23300100**

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

E) Ubah data Udin menjadi berikut : **Idin 2300045**

**Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis**

F) ubah data terakhir menjadi berikut: **Lucy 23300101**

Sebuah gambar berisi teks, elektronik, cuplikan layar, software

Deskripsi dibuat secara otomatis

g) Hapus data awal

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

h) Ubah data awal menjadi berikut: **Bagas 233002**

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

I) Hapus data akhir

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

j) Tampilkan seluruh data

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

1. **KESIMPULAN**

1. struktur : Linked list non-circular memiliki simpul terakhir yang menunjuk ke ‘NULL’, sedangkan linked list circular memiliki simpul terakhir yang merujuk Kembali ke simpul pertama.

2. linked list non-circular membutuhkan pengecekan apakah pointer berada di simpul terakhir (‘null’), sementara linked list circular tidak memiliki simpul terakhir yang sebenarnya, sehingga perulangan dapat berlangsung tanpa batas. Dan tidak perlu mengecek akhir.

1. REFERENSI

Andriano, I. (2017, November 19). *Struktur Data - Single Linked List*. Retrieved from medium.com: https://medium.com/codelabs-unikom/struktur-data-single-linked-list-93bbd56b6ed1

Universitas Negri Malang. (2016). Single Dan Doubly Linked List. *Modul Praktikum Algoritma dan Struktur Data*, 3-4.