**LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA**

**MODUL IV**

A logo of a book

Description automatically generated**LINKED LIST CIRCULAR DAN NON CIRCULAR**

**Disusun oleh:**

**Alfin Ilham Berlianto**

**2311102047**

**Dosen pengampu:**

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA   
FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

MODUL 4

LINKED LIST CIRCULAR DAN NON CIRCULAR

**A. Tujuan**

1. Mahasiswa memahami perbedaan konsep Circular Linked List dan Non Circular.
2. Mahasiswa mampu menerapkan Circular Linked List dan Non Circular dalam pemrograman.

**B. Dasar Teori**

Linked List terdapat 2 jenis, yaitu Circular dan Non Circular.

**Non Circular Linked List**

Linked List merupakan suatu bentuk struktur data yang berisi Kumpulan data yang disebut sebagai node yang tersusun sexara sekuensisal, saling sambung menyambung, dan dinamis. Linked List sering juga disebut sebagai sebutan data berantai. Cara untuk menghubungkan satu node dengan node lainnya maka Linked List menggunkan pointer sebagai petunjuk node selanjutnya atau bisa disebut next.

Secara teoritis data pointer bersifat dinamis, sehingga suatu pointer dapat dijadikan sebuah list berkait atau Linked List. Gambaran logic list berkait dapat digambarkan sebagai berikut:

A diagram of a diagram

Description automatically generated

**Circular Linked List**

Circular Linked List adalah sekumpulan node atau simpul yang tidak terdapat nilai NULL pada satupun nodenya. Bentuk node pada Circular Linked List sama dengan Single Linked List yang mempunyai data dan pointer next. Yang membedakan adalah jika pada Single Linked List, node terakhir menuju ke NULL, tapi di Circular Single Linked List node terakhir menunjuk ke node pertama.

A diagram of a diagram

Description automatically generated

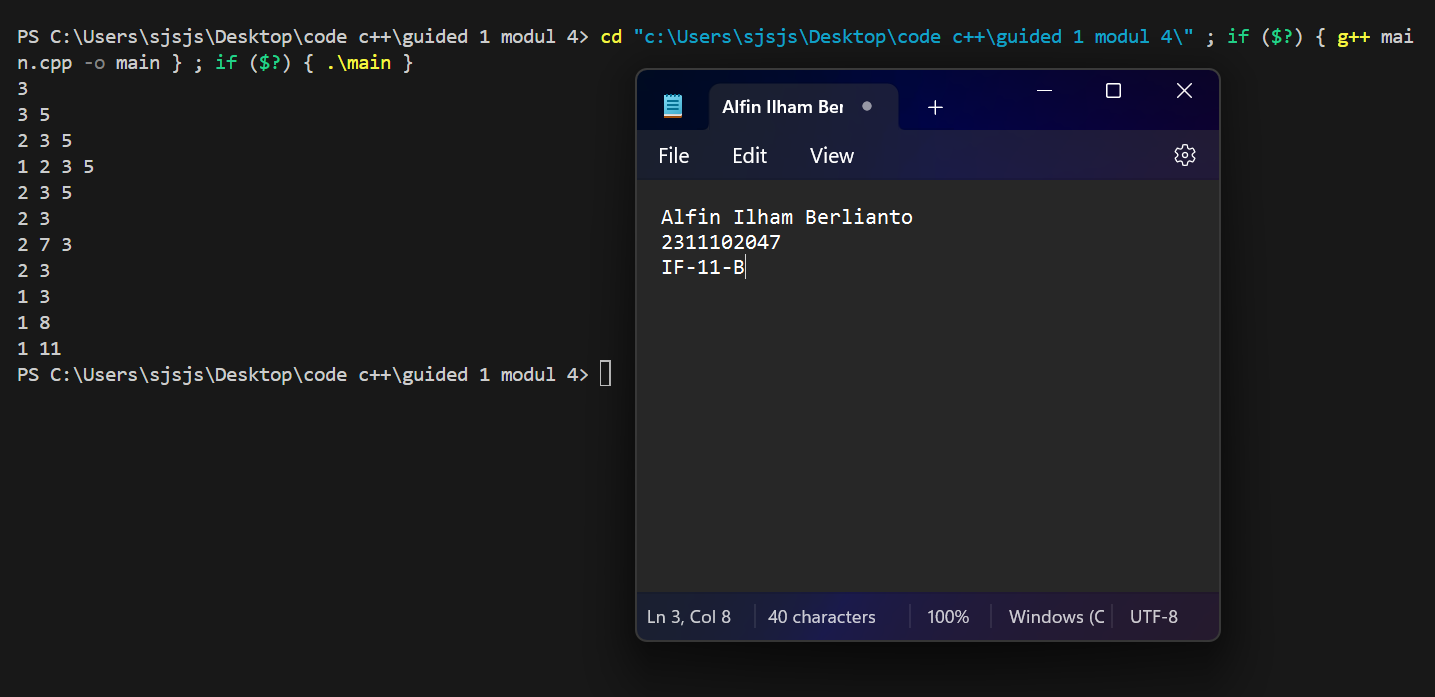
**C. Guided**

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() {  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



DESKRIPSI PROGRAM

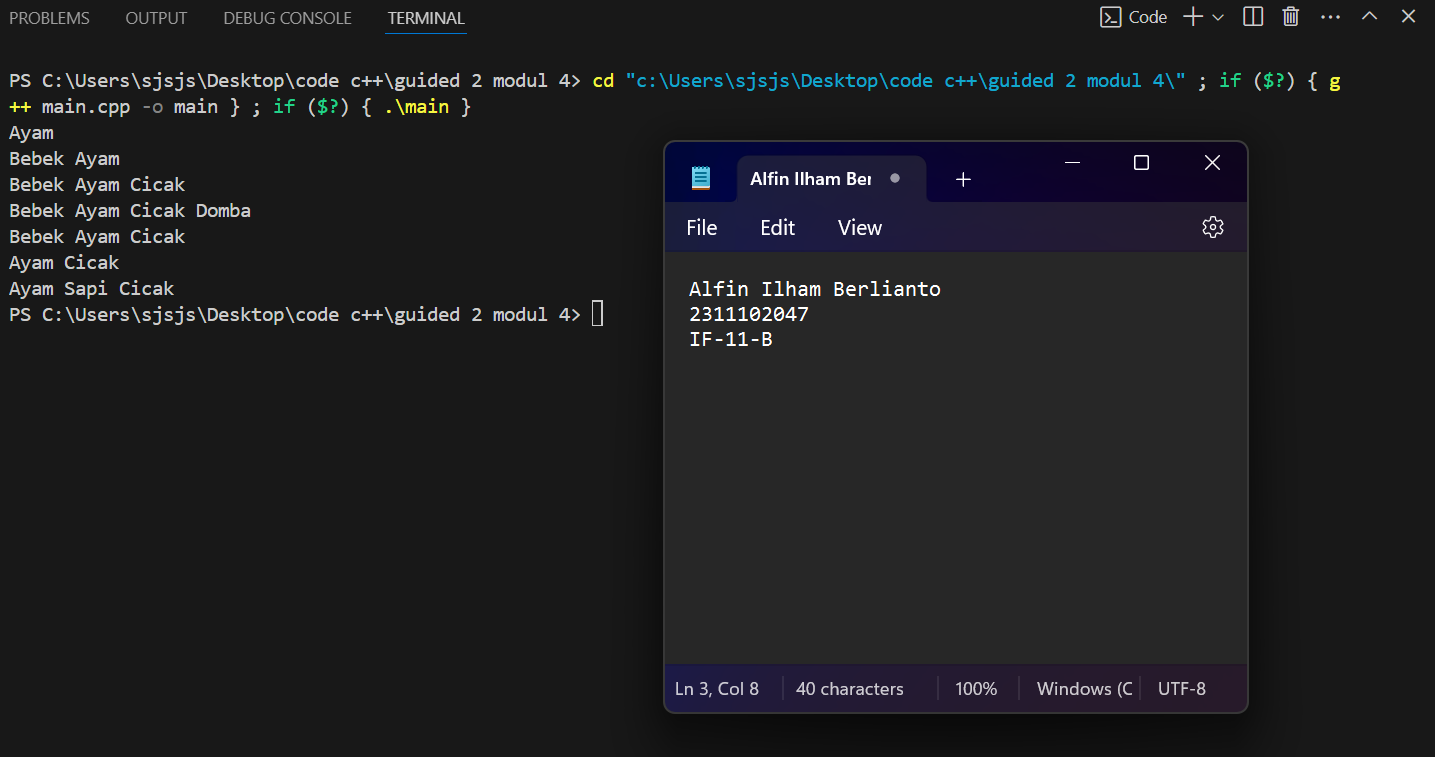
Pada program ini membuat data menggunakan Non Circular Linked List, Yang mana di dalam program ini memiliki 14 Fungsi atau prosedur yang di setiap fungsi atau prosedurnya memiliki kegunaan masing-masing,Seperti contoh yang digunakan di dalam main program ada init(), void ini adalah untuk menginisialisasikan Node awal yang head dan tail nya masih bernilai NULL,Lalu ada insertDepan(), fungsi ini kegunaanya untuk menambahkan node di awal data,insertBelakang() kegunaannya untuk menambahkan data setelah node awal dibuat,hapusBelakang() fungsinya untuk mengahapus node yang berada di belakang data. insertTengah() fungsinya menambahkan node di Tengah-tengah data yang sudah dibuat,hapusTengah() fungsinya untuk menghapus node yang berada di urutan Tengah atau setelah awal dan sebelum akhir,ubahDepan() berguna untuk mengedit node yang sudah dibuat diawal data, ubahTengah() berfungsi untuk mengedit node yang telah dibuat di Tengah-tengah data atau setelah awal node sampai sebelum akhir node.ubahBelakang() berfungsi untuk mengubah data atau node yang berada di urutan belakang.

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() {  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



DESKRIPSI PROGRAM

Pada program ini membuat menggunakan Circular Linked List, pada main program diawali dengan init(), yang mana head nya masih NULL dan tailnya diisi variable head, lalu insertDepan() dengan data yang diisi ayam menggunakan parameter, lalu insertDepan lagi dengan parameter bebek, insertBelakang() dengan nilai parameter cicak, insertBelakang() lagi dengan parameter yang berisi nilai Domba, lalu menghapus data belakang dengan hapusBelakang() yaitu data domba, kemudia hapus depan yaitu data ayam, lalu menambah data di Tengah dengan value sapi dan di posisikan ke 2,lalu menghapus data Tengah di posisi ke 2, dan yang terakhir menampilkan semua data yang telah di inialisasikan sebelumnya menggunakan fungsi tampil().

**D. Unguided**

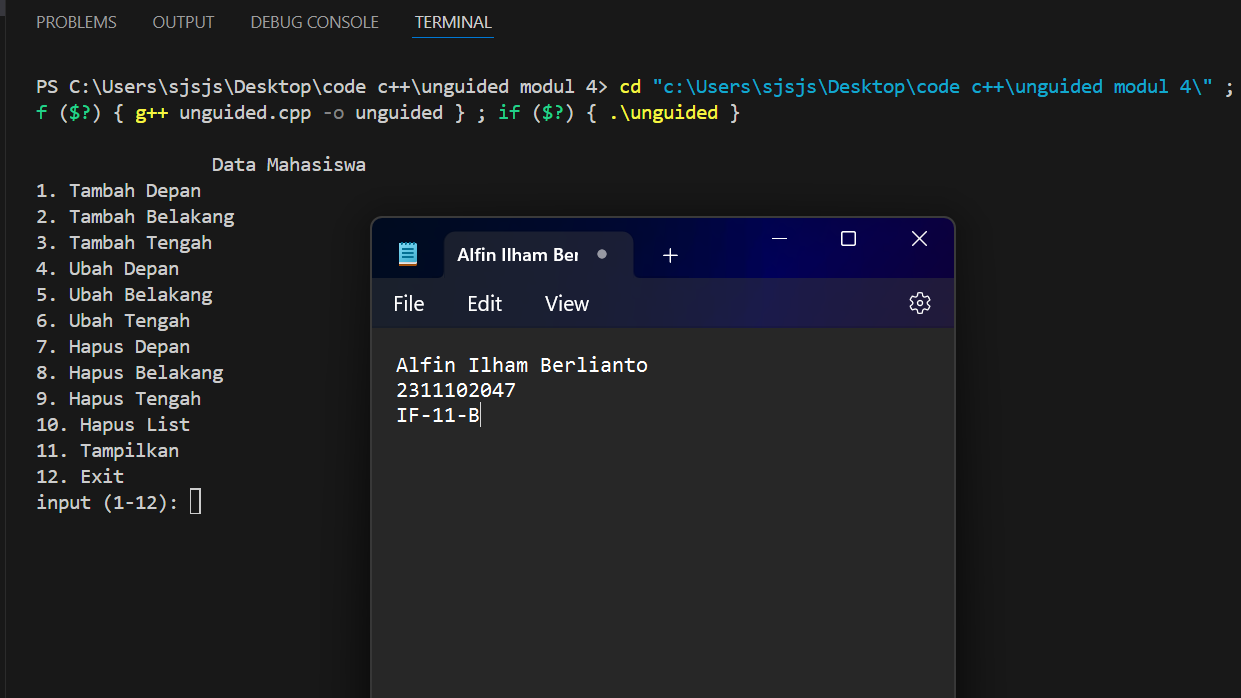
Buatlah program menu Linked List Non Circular untuk menyimpan Nama dan NIM Mahasiswa, dengan menggunakan input dari user.

SOURCE CODE

|  |
| --- |
| #include <iostream>  using namespace std;  /// PROGRAM SINGLE LINKED LIST NON-CIRCULAR  // Deklarasi Struct Node  struct Node  {  string nama;  string nim;  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(string nama, string nim)  {  // Buat Node baru  Node \*baru = new Node;  baru->nama = nama;  baru->nim = nim;  baru->next = NULL;  if (isEmpty() == true)  {  head = tail = baru;  tail->next = NULL;  }  else  {  baru->next = head;  head = baru;  }  }  // Tambah Belakang  void insertBelakang(string nama, string nim)  {  // Buat Node baru  Node \*baru = new Node;  baru->nama = nama;  baru->nim = nim;  baru->next = NULL;  if (isEmpty() == true)  {  head = tail = baru;  tail->next = NULL;  }  else  {  tail->next = baru;  tail = baru;  }  }  // Hitung Jumlah List  int hitungList()  {  Node \*hitung;  hitung = head;  int jumlah = 0;  while (hitung != NULL)  {  jumlah++;  hitung = hitung->next;  }  return jumlah;  }  // Tambah Tengah  void insertTengah(string nama, string nim, 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->nama = nama;  baru->nim = nim;  // tranversing  bantu = head;  int nomor = 1;  while (nomor < posisi - 1)  {  bantu = bantu->next;  nomor++;  }  baru->next = bantu->next;  bantu->next = baru;  }  }  // Hapus Depan  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;  }  }  // Hapus Belakang  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;  }  }  // Hapus Tengah  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;  }  }  // Ubah Depan  void ubahDepan(string nama, string nim)  {  if (isEmpty() == 0)  {  head->nama = nama;  head->nim = nim;  }  else  {  cout << "List masih kosong!" << endl;  }  }  // Ubah Tengah  void ubahTengah(string nama, string nim, 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->nama = nama;  bantu->nim = nim;  }  }  else  {  cout << "List masih kosong!" << endl;  }  }  // Ubah Belakang  void ubahBelakang(string nama, string nim)  {  if (isEmpty() == 0)  {  tail->nama = nama;  tail->nim = nim;  }  else  {  cout << "List masih kosong!" << endl;  }  }  // Hapus List  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;  }  // Tampilkan List  void tampil()  {  Node \*bantu;  bantu = head;  if (!isEmpty())  {  cout << endl << endl;  cout << "\t\t Output Data" <<endl;  cout << "\t\t\_\_\_\_\_\_\_" << endl << endl;  cout << " Nama\t\t| Nim\t\t|" << endl;  while (bantu != NULL)  {  cout << bantu->nama <<"\t\t|"<< bantu->nim <<"\t" << endl;  bantu = bantu->next;  }  }  else  {  cout << "List masih kosong!" << endl;  return;  }  }  int main()  {  string nama;  string nim;  int pilihan;  int posisi;  do  {  cout << endl;  cout << "\t\tData Mahasiswa" << 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" << endl;  cout << "12. Exit" << endl;  cout << "input (1-12): ";  cin >> pilihan;  switch (pilihan)  {  case 1:  {    cout << "Masukkan Nama :";  cin.ignore();  getline(cin, nama);  cout << "Masukkan Nim: ";  cin.ignore();  cin >> nim;  insertDepan(nama, nim);  cout << "Data " << nama << " berhasil diinput!";  break;  }  case 2:  {  cout << "Masukkan Nama :";  cin.ignore();  getline(cin, nama);  cout << "Masukkan Nim: ";  cin >> nim;  insertBelakang(nama, nim);    break;  }  case 3:  {    cout << "Masukkan Nama :";  cin.ignore();  getline(cin, nama);  cout << "Masukkan Nim: ";  cin >> nim;  cout << "Masukan Posisi: ";  cin >> posisi;  insertTengah(nama, nim, posisi);  cout << "Data " << nama << " Berhasil diinput!" << endl;  break;  }  case 4:  {  cout << "Masukkan Nama :";  cin.ignore();  getline(cin, nama);  cout << "Masukkan Nim: ";  cin >> nim;  ubahDepan(nama, nim);  break;  }  case 5:  {    cout << "Masukkan Nama :";  cin.ignore();  getline(cin, nama);  cout << "Masukkan Nim: ";  cin >> nim;  ubahBelakang(nama, nim);  break;  }  case 6:  {  cout << "Masukkan Nama :";  cin.ignore();  getline(cin, nama);  cout << "Masukkan Nim: ";  cin >> nim;  cout << "Masukkan Posisi: ";  cin >> posisi;  ubahTengah(nama, nim, posisi);  break;  }  case 7:  {  hapusDepan();  cout << "Data Depan berhasil terhapus!";  break;  }  case 8:  {    hapusBelakang();  cout << "Data Belakang berhasil terhapus!";  break;  }  case 9:  {    cout << "Masukkan Posisi: ";  cin >> posisi;  hapusTengah(posisi);  cout << "Data Tengah berhasil terhapus!";  break;  }  case 10:  {  clearList();  break;  }  case 11:  {  tampil();  break;  }  case 12:  {  cout << "Terima Kasih!" << endl;  }  default:  {  cout << "Pilihan tidak Valid!" << endl;  break;  }  }  } while (pilihan != 12);  return 0;  } |

SCREENSHOT OUTPUT

Tampilan menu



Operasi tambah

A black and white text

Description automatically generated

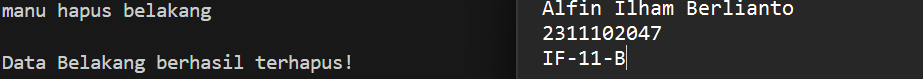
A screen shot of a computer

Description automatically generated

Operasi hapus

A black background with white text

Description automatically generated



Operasi Ubah

A black background with white text

Description automatically generated with medium confidence

A black background with white text

Description automatically generated

Tampil data

A screenshot of a computer

Description automatically generated

Tambah diantara farrel dan denis

A screenshot of a computer

Description automatically generated

Hapus data denis

A screenshot of a computer

Description automatically generated

Tambah data awal

A screenshot of a computer

Description automatically generated

Tambah data akhir

A screenshot of a computer

Description automatically generated

Ubah data udin

A screenshot of a computer

Description automatically generated

Ubah data akhir

A screenshot of a computer

Description automatically generated

Hapus data awal

A screenshot of a computer

Description automatically generated

Ubah data awal

A screenshot of a computer

Description automatically generated

Hapus data akhir dan tampilkan seluruh data

A screenshot of a computer

Description automatically generated

**E. KESIMPULAN**

Kesimpulan yang dapat diambil dari materi Linked List circular dan Non Circular ini adalah jika Linked List Circular Tailnya itu mengarah ke Head lagi (looping), sedangkan Linked List Non Circular ini tailnya mengarah ke NULL.

**Referensi**

1. Kuliah Informatika, Belajar Konsep Linked List secara visual (Kuliah Struktur Data - bag. 4)

<https://www.youtube.com/watch?v=VunTar7dl-0&t=3480>

3. REPOSITTORY UNIVERSITAS DIAN NUSWANTORO (2014. 29 Juli) SINGLE LINKED LIST NON CIRCULAR. Diakses pada 24 September 2018, dari

<https://repository.dinus.ac.id/docs/ajar/Pertemuan_11_Single_Linked_List_Circular.pdf>