LAPORAN PRAKTIKUM

STRUKTUR DATA DAN ALGORITME

MODUL 8

**DOUBLE 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

Double Linked List (Senarai Berantai Ganda) merupakan jenis linked list yang hampir sama dengan Single Linked List. Yang membedakan adalah penggunaan pointernya. Double linked list menggunakan dua buah pointer yaitu next dan prev. Pointer next digunakan untuk menunjuk node/simpul selanjutnya dan pointer prev digunakan untuk menunjuk node/simpul sebelumnya. Keberadaan dua pointer menjadikan double linked list menjadi lebih fleksibel dibandingkan single linked list.

Seperti halnya single list, double linked list juga memiliki dua jenis yaitu Non-Circular dan Circular.

Double Linked List Non-Circular (DLLNC)

* Pointer prev pada node head menunjuk ke NULL, menandakan bahwa node head sebagai node awal.
* Pointer next pada node tail menunjuk ke NULL, menandakan bahwa node tail sebagai node akhir.

Chart

Description automatically generated

Double Linked List Circular (DLLC)

* Pointer prev pada node head menunjuk ke node tail.
* Pointer next pada node tail menunjuk ke node head.

Chart, waterfall chart

Description automatically generated

Operasi dasar double linked list

1. Deklarasi

Pendeklarasian DLL menggunakan tipe data struct yang terdiri atas 3 buah field: data, pointer next dan prev.

1. Inisialisasi

Diperlukan untuk memberikan nilai awal node pada list kosong dengan memberikan nilai NULL pada node head dan tail.

1. Create

Operasi untuk membuat node baru.

1. isEmpty

Operasi untuk memeriksa apakah suatu linked list masih kosong.

1. Insert

Operasi untuk menambahkan satu node ke dalam linked list.

1. Delete

Operasi untuk menghapus node pada linked list.

1. Clear

Operasi untuk menghapus atau mengosongkan seluruh data pada linked list.

1. Display

Operasi untuk menampilkan seluruh data linked list.

Ref:

Modul 8 : DOUBLE LINKED LIST

[Double Linked List | rizkidoank](https://www.rizkidoank.com/2016/10/17/double-linked-list/)

[Double Linked List pada C++ Pengertian dan Contoh Program - pintarkom](https://pintarkom.com/double-linked-list-pada-c-plus/)

# Guided

## Program Double Linked List Non-Circular (DLLNC)

|  |
| --- |
| #include <iostream>  using namespace std;  ///PROGRAM DOUBLE LINKED LIST NON-CIRCULAR (DLLNC)  //Deklarasi DLLNC  struct Node  {      int data;      Node \*next;      Node \*prev;  };  Node \*head, \*tail, \*baru, \*bantu, \*bantu2, \*hapus;  //Inisialisasi Node  void init()  {      head = NULL;      tail = NULL;  }  //IsEmpty  int isEmpty()  {      if (head == NULL && tail == NULL)      return 1; //true      else      return 0; //false  }  //Buat Node Baru  void create(int input)  {      baru = new Node;      baru->data = input;      baru->next = NULL;      baru->prev = NULL;  }  //Tambah Depan  void insertDepan(int input)  {      create(input);      if (isEmpty() == 1){          head = baru;          tail = baru;      }      else{          baru->next = head;          head->prev = baru;          head = baru;      }  }  //Tambah Belakang  void insertBelakang(int input)  {      create(input);      if (isEmpty() == 1){          head = baru;          tail = baru;      }      else{          tail->next = baru;          baru->prev = tail;          tail = baru;      }  }  //Hitung List  int countList()  {      int counter = 0;      bantu = head;      while(bantu != NULL){          counter++;          bantu = bantu->next;      }      return counter;  }  //Tambah Tengah  void insertTengah(int input, int posisi)  {      create(input);      if (posisi < 1 || posisi > countList()){          cout << " Posisi di luar jangkauan!" << endl;      }      else if ( posisi == 1){          cout << " Posisi bukan posisi tengah!" << endl;      }      else{          bantu = head;          int counter = 1;          while (counter < posisi-1){              bantu = bantu->next;              counter++;          }            bantu2 = bantu->next;          baru->prev = bantu;          baru->next = bantu2;          bantu->next = baru;          bantu2->prev = baru;      }  }  //Hapus Depan  void hapusDepan()  {      if (isEmpty() == 0){          hapus = head;          head = head->next;          head->prev = NULL;          delete hapus;      }      else{          cout << " List masih kosong!" << endl;      }  }  //Hapus Belakang  void hapusBelakang()  {      if (isEmpty() == 0){          hapus = tail;          tail = tail->prev;          tail->next = NULL;          delete hapus;      }      else{          cout << " List masih kosong!" << endl;      }  }  //Hapus Tengah  void hapusTengah(int posisi)  {      if (isEmpty() == 0){          if (posisi < 1 || posisi > countList()){              cout << " Posisi di luar jangkauan!" << endl;          }          else if (posisi == 1 || posisi == countList()){              cout << " Posisi bukan posisi tengah!" << endl;          }          else{              int counter = 1;              bantu = head;                while (counter < posisi-1){                  bantu = bantu->next;                  counter++;          }            hapus = bantu->next;          bantu2 = hapus->next;          bantu->next = bantu2;          bantu2->prev = bantu;          delete hapus;          }      }      else{          cout << " List masih kosong!" << endl;      }  }  //Ubah Depan  void ubahDepan(int input)  {      if (isEmpty() == 0){          head->data = input;      }      else{          cout << " List masih kosong!" << endl;      }  }  //Ubah Belakang  void ubahBelakang(int input)  {      if (isEmpty() == 0){          tail->data = input;      }      else{          cout << " List masih kosong!" << endl;      }  }  //Ubah Tengah  void ubahTengah(int input, int posisi)  {      if (isEmpty() == 0){          if (posisi < 1 || posisi > countList()){              cout << " Posisi di luar jangkauan!" << endl;          }          else if (posisi == 1 || posisi == countList()){              cout << " Posisi bukan posisi tengah!" << endl;          }          else{              bantu = head;              int counter = 1;            while (counter < posisi){              bantu = bantu->next;              counter++;          }              bantu->data = input;          }      }      else{          cout << " List masih kosong!" << endl;      }  }  //Hapus List  void clearList()  {      bantu = head;      while (bantu != NULL){          hapus = bantu;          bantu = bantu->next;          delete hapus;      }        head = tail = NULL;      cout << " List berhasil terhapus!" << endl;  }  //Tampilkan List  void tampil()  {      bantu = head;      if (isEmpty() == 0){          while (bantu != NULL){              cout << ends << 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:**

Yang pertama adalah membuat Deklarasi dan inisialisasi dari struct itu sendiri, dan pointernya. Lalu melakukan inisialisasi node, dan membuat fungsi untuk mengecek apa data kosong. Membuat fungsi untuk menambah depan, belakang, dan tengah. Kemudian fungsi hapus depan, belakang, dan tengah. Fungsi ubah depan, belakang, dan tengah. Fungsi untuk menghapus data dan menampilkan data. Yang terakhir menu utama untuk memanggil fungsi.

**Output:**

Text

Description automatically generated

## Program Double Linked List Circular (DLLC)

|  |
| --- |
| #include <iostream>  using namespace std;  ///PROGRAM DOUBLE LINKED LIST CIRCULAR  //Deklarasi DLLC  struct Node{      string data;      Node \*next;      Node \*prev;  };  //Deklarasi Node  Node \*head, \*tail, \*baru, \*bantu, \*bantu2, \*hapus;  //Inisialisasi Node  void init(){      head = NULL;      tail = head;  }  //isEmpty  int isEmpty(){      if (head == NULL && tail == NULL)      return 1; //true      else      return 0; //false  }  //Buat Node Baru  void create(string input){      baru = new Node;      baru->data = input;      baru->next = NULL;      baru->prev = NULL;  }  //Tambah Depan  void insertDepan(string input){      create(input);      if (isEmpty() == 1){          head = baru;          tail = head;          head->prev = tail;          tail->next = head;      }      else{          baru->next = head;          head->prev = baru;          head = baru;          head->prev = tail;          tail->next = head;      }  }  //Tambah Belakang  void insertBelakang(string input){      create(input);      if (isEmpty() == 1){          head = baru;          tail = head;          head->prev = tail;          tail->next = head;      }      else{          baru->prev = tail;          tail->next = baru;          tail = baru;          tail->next = head;          head->prev = tail;      }  }  //Tambah Tengah  void insertTengah(string input, int posisi){      create(input);      if (isEmpty() == 1){          cout << " List masih kosong!" << endl;      }      else{          if (posisi == 1){              cout << " Posisi bukan posisi tengah!" << endl;          }          else if (posisi < 1){              cout << " Posisi di luar jangkauan!" << endl;          }          else{              int counter = 1;              bantu = head;                while (counter < posisi-1){                  bantu = bantu->next;                  counter++;              }                bantu2 = bantu->next;              bantu->next = baru;              bantu2->prev = baru;              baru->prev = bantu;              baru->next = bantu2;          }      }  }  //Hapus Depan  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;      }  }  //Hapus Belakang  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;      }  }  //Hapus Tengah  void hapusTengah(int posisi){      if (isEmpty() == 1){          cout << " List masih kosong!" << endl;      }      else{          if (posisi == 1){              cout << " Posisi bukan posisi tengah!" << endl;          }          else if (posisi < 1){              cout << " Posisi di luar jangkauan!" << endl;          }          else{              int counter = 1;              bantu = head;              while (counter < posisi-1){              bantu = bantu->next;              counter++;          }            hapus = bantu->next;          bantu2 = hapus->next;          bantu->next = bantu2;          bantu2->prev = bantu;          delete hapus;          }      }  }  //Hapus List  void clearList(){        if (isEmpty() == 0){          hapus = head;          while (hapus->next != head){              hapus = bantu;              bantu = bantu->next;              delete hapus;          }            head = NULL;          tail = NULL;      }      cout << "List berhasil terhapus!" << endl;  }  //Tampil  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();      insertDepan("Babi");      tampil();      hapusDepan();      tampil();      insertTengah("Sapi", 2);      tampil();      hapusTengah(2);      tampil();      return 0;  } |

**Dekripsi:**

Yang pertama adalah membuat Deklarasi dan inisialisasi dari stack itu sendiri, di lanjut dengan pointernya. Membuat fungsi untuk mengecek apa list kosong atau tidak, dan membuat fungsi create untuk tambah data. Fungsi insert depan, tengah, dan belakang untuk menambah data di depan, tengah, atau belakang. Ada fungsi hapus depan, belakang, dan tengah, untuk hapus data persatuan. Ada fungsi Clear untuk menghapus semua data. Fungsi tampil untuk menampilkan data. Dan yang terakhir fungsi utama untuk memanggil fungsi tadi.

**Output:**

Text

Description automatically generated

# Tugas (Unguided)

1. Buatlah program menu Double Linked List Circular dengan minimal 2 data input user. Tambahkan operasi search/find untuk mencari data pada linked list!

|  |
| --- |
| // Nama : Fatkhurrohman Purnomo  // NIM : 21102125  #include <iostream>  using namespace std;  ///PROGRAM DOUBLE LINKED LIST CIRCULAR  //Deklarasi DLLC  struct Node  {      int nomor ;      string nama;      char karakter;      Node \*next;      Node \*prev;  };  //Deklarasi Node  Node \*head, \*tail, \*baru, \*bantu, \*bantu2, \*hapus;  //Inisialisasi Node  void init()  {      head = NULL;      tail = head;  }  //isEmpty  int isEmpty()  {      if (head == NULL && tail == NULL)          return 1; //true      else          return 0; //false  }  //Buat Node Baru  void create(int nomor , string nama, char karakter)  {      baru = new Node;      baru->nomor = nomor ;      baru->nama = nama;      baru->karakter = karakter;      baru->next = NULL;      baru->prev = NULL;  }  //Tambah Depan  void insertDepan(int nomor , string nama, char karakter)  {      create(nomor, nama, karakter);      if (isEmpty() == 1){          head = baru;          tail = head;          head->prev = tail;          tail->next = head;      }      else{          baru->next = head;          head->prev = baru;          head = baru;          head->prev = tail;          tail->next = head;      }  }  //Tambah Belakang  void insertBelakang(int nomor , string nama, char karakter)  {      create(nomor, nama, karakter);      if (isEmpty() == 1){          head = baru;          tail = head;          head->prev = tail;          tail->next = head;      }      else{          baru->prev = tail;          tail->next = baru;          tail = baru;          tail->next = head;          head->prev = tail;      }  }  //Tambah Tengah  void insertTengah(int posisi, int nomor , string nama, char karakter)  {      create(nomor, nama, karakter);      if (isEmpty() == 1){          cout << " List masih kosong!" << endl;      }      else{          if (posisi == 1){              cout << " Posisi bukan posisi tengah!" << endl;          }          else if (posisi < 1){              cout << " Posisi di luar jangkauan!" << endl;          }          else{              int counter = 1;              bantu = head;              while (counter < posisi-1){                  bantu = bantu->next;                  counter++;              }              bantu2 = bantu->next;              bantu->next = baru;              bantu2->prev = baru;              baru->prev = bantu;              baru->next = bantu2;          }      }  }  //Hapus Depan  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;      }  }  //Hapus Belakang  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;      }  }  //Hapus Tengah  void hapusTengah(int posisi)  {      if (isEmpty() == 1){          cout << " List masih kosong!" << endl;      }      else{          if (posisi == 1){              cout << " Posisi bukan posisi tengah!" << endl;          }          else if (posisi < 1){              cout << " Posisi di luar jangkauan!" << endl;          }          else{              int counter = 1;              bantu = head;              while (counter < posisi-1){                  bantu = bantu->next;                  counter++;              }              hapus = bantu->next;              bantu2 = hapus->next;              bantu->next = bantu2;              bantu2->prev = bantu;              delete hapus;          }      }  }  //Hapus List  void clearList()  {   if (isEmpty() == 0){          hapus = head;          while (hapus->next != head){              hapus = bantu;              bantu = bantu->next;              delete hapus;          }          head = NULL;          tail = NULL;      }      cout << "List berhasil terhapus!" << endl;  }  //Tampil  void tampil()  {      if (isEmpty() == 0){          tail = head;          cout << "==========================================================" << endl;          cout << "| No | Huruf | Karakter |" << endl;          cout << "==========================================================" << endl;          do{              cout << "  " << tail->nomor << "  : " << tail->nama << "    : " << tail->karakter << endl;              tail = tail->next;          }while (tail != head);          cout << endl;      }      else{          cout << "List masih kosong!" << endl;      }  }  // search  void search(string nama)  {      if (isEmpty() == 0){          tail = head;          cout << "==========================================================" << endl;          cout << "| No | Huruf | Karakter |" << endl;          cout << "==========================================================" << endl;          do{              if (tail->nama == nama){                  cout << "  " << tail->nomor << "  : " << tail->nama << "    : " << tail->karakter << endl;              }              tail = tail->next;          }while (tail != head);          cout << endl;      }      else{          cout << "List masih kosong!" << endl;      }  }  int main()  {      int pilih, nomor , posisi;      string nama;      char karakter;      init();      do {          system("cls");          cout << endl;          cout << "==========================================================" << endl;          cout << "|                         Pilih Aksi                     |" << endl;          cout << "==========================================================" << endl;          cout << " 1. Tambah Depan" << endl;          cout << " 2. Tambah Belakang" << endl;          cout << " 3. Tambah Tengah" << endl;          cout << " 4. Hapus Depan" << endl;          cout << " 5. Hapus Belakang" << endl;          cout << " 6. Hapus Tengah" << endl;          cout << " 7. cari" << endl;          cout << " 8. Hapus List" << endl;          cout << " 9. Tampil" << endl;          cout << " 10. Keluar" << endl;          cout << " Masukkan pilihan anda : ";          cin >> pilih;          cout << "==========================================================" << endl << endl;          switch (pilih){              case 1:                  cout << " Masukkan No. : ";                  cin >> nomor ;                  cout << " Masukkan nama : ";                  cin >> nama;                  cout << " Masukkan kode : ";                  cin >> karakter;                  insertDepan(nomor, nama, karakter);                  break;              case 2:                  cout << " Masukkan No. : ";                  cin >> nomor ;                  cout << " Masukkan nama : ";                  cin >> nama;                  cout << " Masukkan kode : ";                  cin >> karakter;                  insertBelakang(nomor, nama, karakter);                  break;              case 3:                  cout << " Masukkan posisi : ";                  cin >> posisi;                  cout << " Masukkan No. : ";                  cin >> nomor ;                  cout << " Masukkan nama : ";                  cin >> nama;                  cout << " Masukkan kode : ";                  cin >> karakter;                  insertTengah(posisi, nomor , nama, karakter);                  break;              case 4:                  hapusDepan();                  break;              case 5:                  hapusBelakang();                  break;              case 6:                  cout << " Masukkan posisi : ";                  cin >> posisi;                  hapusTengah(posisi);                  break;              case 7:                  cout << " Masukkan nama : ";                  cin >> nama;                  search(nama);                  break;              case 8:                  clearList();                  break;              case 9:                  tampil();                  break;              default:                  cout << " Pilihan tidak ada!" << endl;                  break;          }      }while (pilih != 10);      cout << "==========================================================" << endl;      return 0;  } |

**Deskripsi:**

Sama seperti latihan no 2, hanya saja menggunakan 3 tipe data, yaitu integer, string dan char. Lalu ada fungsi cari sebagai tambahan. Pada fungsi utama sendiri dibuat menu untuk mempermudah user menggunakan program.

**Output:**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

1. Buatlah program Double Linked List dengan input dari user dimana data yang ditampilkan secara terbalik (dari belakang ke depan)!

|  |
| --- |
| // Nama : Fatkhurrohman Purnomo  // NIM : 21102125  #include <iostream>  using namespace std;  ///PROGRAM DOUBLE LINKED LIST NON-CIRCULAR (DLLNC)  //Deklarasi DLLNC  struct Node  {      int angka;      string huruf;      Node \*next;      Node \*prev;  };  Node \*head, \*tail, \*baru, \*bantu, \*bantu2, \*hapus;  //Inisialisasi Node  void init()  {      head = NULL;      tail = NULL;  }  //IsEmpty  int isEmpty()  {      if (head == NULL && tail == NULL)          return 1;   //true      else          return 0;   //false  }  //Buat Node Baru  void create(int angka, string huruf)  {      baru = new Node;      baru->angka = angka;      baru->huruf = huruf;      baru->next = NULL;      baru->prev = NULL;  }  //Tambah Depan  void insertDepan(int angka, string huruf)  {      create(angka, huruf);      if (isEmpty() == 1){          head = baru;          tail = head;      }      else{          baru->next = head;          head->prev = baru;          head = baru;      }  }  //Tambah Belakang  void insertBelakang(int angka, string huruf)  {      create(angka, huruf);      if (isEmpty() == 1){          head = baru;          tail = head;      }      else{          tail->next = baru;          baru->prev = tail;          tail = baru;      }  }  //Hitung List  int countList()  {      int counter = 0;      bantu = head;      while(bantu != NULL){          counter++;          bantu = bantu->next;      }      return counter;  }  //Tambah Tengah  void insertTengah(int angka, int posisi, string huruf)  {      create(angka, huruf);      if (posisi < 1 || posisi > countList()){              cout << " Posisi di luar jangkauan!" << endl;      }      else if ( posisi == 1){          cout << " Posisi bukan posisi tengah!" << endl;      }      else{          bantu = head;          int counter = 1;          while (counter < posisi-1){              bantu = bantu->next;              counter++;          }          bantu2 = bantu->next;          baru->prev = bantu;          baru->next = bantu2;          bantu->next = baru;          bantu2->prev = baru;      }  }  //Hapus Depan  void hapusDepan()  {      if (isEmpty() == 0){          hapus = head;          head = head->next;          head->prev = NULL;          delete hapus;      }      else{          cout << " List masih kosong!" << endl;      }  }  //Hapus Belakang  void hapusBelakang()  {      if (isEmpty() == 0){          hapus = tail;          tail = tail->prev;          tail->next = NULL;          delete hapus;      }      else{          cout << " List masih kosong!" << endl;      }  }  //Hapus Tengah  void hapusTengah(int posisi)  {      if (isEmpty() == 0){          if (posisi < 1 || posisi > countList()){              cout << " Posisi di luar jangkauan!" << endl;          }          else if (posisi == 1 || posisi == countList()){              cout << " Posisi bukan posisi tengah!" << endl;          }          else{              int counter = 1;              bantu = head;              while (counter < posisi-1){                  bantu = bantu->next;                  counter++;              }              hapus = bantu->next;              bantu2 = hapus->next;              bantu->next = bantu2;              bantu2->prev = bantu;              delete hapus;          }      }      else{          cout << " List masih kosong!" << endl;      }  }  //Ubah Depan  void ubahDepan(int angka, string huruf)  {      if (isEmpty() == 0){          head->angka = angka;          head->huruf = huruf;      }      else{          cout << " List masih kosong!" << endl;      }  }  //Ubah Belakang  void ubahBelakang(int angka, string huruf)  {      if (isEmpty() == 0){          tail->angka = angka;          tail->huruf = huruf;      }      else{          cout << " List masih kosong!" << endl;      }  }  //Ubah Tengah  void ubahTengah(int posisi, int angka, string huruf)  {      if (isEmpty() == 0){          if (posisi < 1 || posisi > countList()){              cout << " Posisi di luar jangkauan!" << endl;          }          else if (posisi == 1 || posisi == countList()){              cout << " Posisi bukan posisi tengah!" << endl;          }          else{              bantu = head;              int counter = 1;              while (counter < posisi){                  bantu = bantu->next;                  counter++;              }              bantu->angka = angka;              bantu->huruf = huruf;          }      }      else{          cout << " List masih kosong!" << endl;      }  }  //Hapus List  void clearList()  {      bantu = head;      while (bantu != NULL){          hapus = bantu;          bantu = bantu->next;          delete hapus;      }      head = tail = NULL;      cout << " List berhasil terhapus!" << endl;  }  //Tampilkan List  void tampil()  {      bantu = tail;      cout << endl << " List : " << endl;      if (isEmpty() == 0){          while (bantu != NULL){              cout << bantu->angka << " = " << bantu->huruf << endl;              bantu = bantu->prev;          }          cout << endl;      }      else{          cout << " List masih kosong!" << endl;      }  }  int main()  {      int pilih, nomor, posisi;      string nama;      do{          cout << endl << " Menu : " << endl;          cout << " 1. Tambah Depan" << endl;          cout << " 2. Tambah Belakang" << endl;          cout << " 3. Tambah Tengah" << endl;          cout << " 4. Hapus Depan" << endl;          cout << " 5. Hapus Belakang" << endl;          cout << " 6. Hapus Tengah" << endl;          cout << " 7. Ubah Depan" << endl;          cout << " 8. Ubah Belakang" << endl;          cout << " 9. Ubah Tengah" << endl;          cout << " 10. Hapus List" << endl;          cout << " 11. Tampilkan List" << endl;          cout << " Masukkan pilihan: ";          cin >> pilih;          cout << endl;          switch(pilih){              case 1:                  cout << " Masukkan nomor: ";                  cin >> nomor;                  cout << " Masukkan huruf: ";                  cin >> nama;                  insertDepan(nomor, nama);                  break;              case 2:                  cout << " Masukkan nomor: ";                  cin >> nomor;                  cout << " Masukkan huruf: ";                  cin >> nama;                  insertBelakang(nomor, nama);                  break;              case 3:                  cout << " Masukkan nomor: ";                  cin >> nomor;                  cout << " Masukkan huruf: ";                  cin >> nama;                  cout << " Masukkan posisi: ";                  cin >> posisi;                  insertTengah(posisi, nomor, nama);                  break;              case 4:                  hapusDepan();                  break;              case 5:                  hapusBelakang();                  break;              case 6:                  cout << " Masukkan posisi: ";                  cin >> posisi;                  hapusTengah(posisi);                  break;              case 7:                  cout << " Masukkan nomor: ";                  cin >> nomor;                  cout << " Masukkan huruf: ";                  cin >> nama;                  ubahDepan(nomor, nama);                  break;              case 8:                  cout << " Masukkan nomor: ";                  cin >> nomor;                  cout << " Masukkan huruf: ";                  cin >> nama;                  ubahBelakang(nomor, nama);                  break;              case 9:                  cout << " Masukkan nomor: ";                  cin >> nomor;                  cout << " Masukkan huruf: ";                  cin >> nama;                  cout << " Masukkan posisi: ";                  cin >> posisi;                  ubahTengah(posisi, nomor, nama);                  break;              case 10:                  clearList();                  break;              case 11:                  tampil();                  break;              default:                  cout << " Pilihan tidak ada!" << endl;          }      }while(pilih != 0);      return 0;  } |

**Deskripsi:**

Sama seperti latihan no 1, hanya saja menggunakan 2 tipe data, yaitu integer, string. Lalu pada saat menampilkan data dari belakang. Pada fungsi utama sendiri dibuat menu untuk mempermudah user menggunakan program.

**Output:**

Input program menggunakan:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Output dari input:

A screenshot of a computer

Description automatically generated with medium confidence

# Kesimpulan

1. Bisa membuat double linked list
2. Belajar lebih dalam double linked list
3. Lebih mahir dalam menggunakan bahasa C++
4. 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