LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL VI
SINGLY LINKED LIST 1

Disusun Oleh :

NAMA : Gien Darrel Adli

NIM : 10312430008

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Algoritma merupakan fondasi dalam pembuatan program komputer. Secara sederhana, algoritma adalah serangkaian langkah logis dan sistematis yang disusun untuk menyelesaikan suatu masalah. C++, berfungsi sebagai alat untuk mengimplementasikan algoritma tersebut agar dapat dimengerti dan dieksekusi oleh komputer. C++ sering digunakan sebagai bahasa pengantar untuk mempelajari konsep pemrograman dasar karena strukturnya yang terorganisir dan kemampuannya untuk menangani operasi tingkat rendah.

Untuk membangun logika dalam program sesuai dengan algoritma yang dirancang, C++ menyediakan struktur kontrol, yang terbagi menjadi dua jenis utama:

Struktur Percabangan (Conditional): Digunakan untuk pengambilan keputusan, di mana program akan menjalankan blok kode tertentu jika suatu kondisi terpenuhi. Struktur ini mencakup if-else untuk mengevaluasi kondisi boolean dan switch-case untuk memilih blok kode berdasarkan nilai dari sebuah variabel.

Struktur Perulangan (Looping): Digunakan untuk mengeksekusi blok kode yang sama secara berulang kali selama kondisi tertentu masih terpenuhi. C++ menyediakan tiga jenis perulangan utama: for, while, dan do-while, yang masing-masing memiliki karakteristik penggunaan yang spesifik dalam implementasi algoritma.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1 (main.cpp)

```
#include <iostream>

using namespace std;


struct Node
{
    int data;
    Node *prev;
    Node *next;
};


Node *ptr_first = NULL;
Node *ptr_last = NULL;
```

```
void add_first(int value)
{
   Node *newNode = new Node{value, NULL, ptr_first};

   if (ptr_first == NULL)
   {
      ptr_last = newNode;
   }
   else
   {
      ptr_first->prev = newNode;
   }
   ptr_first = newNode;
}

void add_last(int value)
{
   Node *newNode = new Node{value, ptr_last, NULL};

   if (ptr_last == NULL)
   {
      ptr_first = newNode;
   }
   else
   {
      ptr_last ->next = newNode;
   }
   ptr_last = newNode;
}

void add_target(int targetValue, int newValue)
{
```

```
    Node *current = ptr_first;
    while (current != NULL && current->data != targetValue)
    {
        current = current->next;
    }

    if (current != NULL)
    {
        if (current == ptr_last)
        {
            add_last(newValue);
        }
        else
        {
            Node *newNode = new Node{newValue, current, current->next};
            current->next->prev = newNode;
            current->next = newNode;
        }
    }
}

void view()
{
    Node *current = ptr_first;
    if (current == NULL)
    {
        cout << "List Kosong\n";
        return;
    }
    while (current != NULL)
    {
        cout << current->data << (current->next != NULL ? "<->" : "");
```

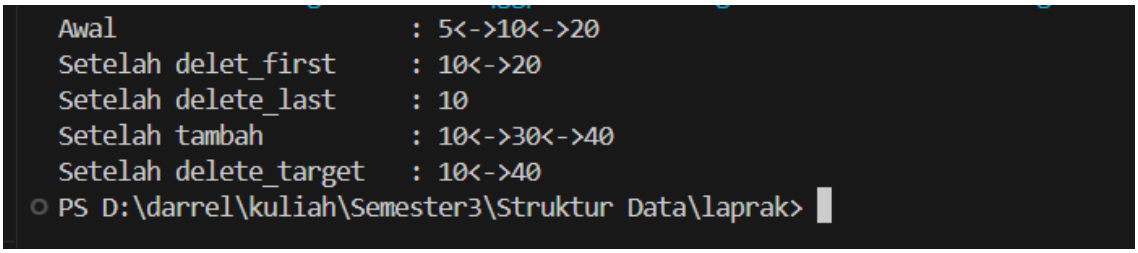
```
        current = current->next;
    }
    cout << endl;
}

void delete_first()
{
    if (ptr_first == NULL)
        return;

    Node *temp = ptr_first;

    if (ptr_first == ptr_last)
    {
        ptr_first = NULL;
        ptr_last = NULL;
    }
    else
    {
        ptr_first = ptr_first->next;
        ptr_first->prev = NULL;
    }
}

void delete_last()
{
    if (ptr_last == NULL)
        return;

    Node *temp = ptr_last;

    if (ptr_first == ptr_last)
```

```
    {
        ptr_first = NULL;
        ptr_last = NULL;
    }
    else
    {
        ptr_last = ptr_last->prev;
        ptr_last->next = NULL;
    }
    delete temp;
}

void delete_target(int targetValue)
{
    Node *current = ptr_first;
    while (current != NULL && current->data != targetValue)
    {
        current = current->next;
    }

    if (current != NULL)
    {
        if (current == ptr_first)
        {
            delete_first();
            return;
        }
        else if (current == ptr_last)
        {
            delete_last();
            return;
        }
```

```
        else
        {
            current->prev->next = current->next;
            current->next->prev = current->prev;
            delete current;
        }
    }
}

void edit_mode(int targetValue, int newValue)
{
    Node *current = ptr_first;
    while (current != NULL && current->data != targetValue)
    {
        current = current->next;
    }

    if (current != NULL)
    {
        current->data = newValue;
    }
}

int main()
{
    add_first(10);
    add_first(5);
    add_last(20);
    cout << "Awal\t\t\t: ";
    view();

    delete_first();
```

```
    cout << "Setelah delet_first\t: ";
    view();
    delete_last();
    cout << "Setelah delete_last\t: ";
    view();

    add_last(30);
    add_last(40);
    cout << "Setelah tambah\t\t: ";
    view();

    delete_target(30);
    cout << "Setelah delete_target\t: ";
    view();
}
```

Screenshots Output :



Deskripsi:

Program ini adalah implementasi double linked list di mana setiap node terhubung ke node sebelumnya dan sesudahnya. Program mendukung operasi tambah data di awal, akhir, dan setelah data tertentu, hapus data di awal, akhir, dan berdasarkan nilai, serta menampilkan isi list. Pada fungsi main, operasi-operasi tersebut dijalankan untuk memperlihatkan perubahan isi list setelah penambahan dan penghapusan data.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Doublylist.h

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H

#include <iostream>
#include <string>
using namespace std;

struct kendaraan {
   string nopol;
   string warna;
   int thnBuat;
};

typedef kendaraan infotype;
typedef struct ElmList *address;

struct ElmList {
   infotype info;
   address next;
   address prev;
};

struct List {
   address First;
   address Last;
};

void CreateList(List &L);
```

```
address alokasi(infotype x);

void dealokasi(address &P);

void printInfo(List L);

void insertLast(List &L, address P);


address findElm(List L, string nopol);

bool isNopolExist(List L, string nopol);

void deleteFirst(List &L, address &P);

void deleteLast(List &L, address &P);

void deleteAfter(address Prec, address &P);

bool deleteByNopol(List &L, string nopol);


#endif
```

Deskripsi:

File doublylist.h adalah file header yang berisi deklarasi struktur dan fungsi untuk implementasi double linked list kendaraan. Di dalamnya didefinisikan struktur data kendaraan, node (ElmList) dengan pointer next dan prev, struktur List dengan penunjuk First dan Last, serta prototype fungsi untuk membuat list, alokasi/dealokasi, insert, delete, pencarian, dan pengecekan nopol. File ini berfungsi sebagai antarmuka yang akan diimplementasikan di file doublylist.cpp.

doublylist.cpp

```
#include "doublylist.h"

void CreateList(List &L) {
   L.First = NULL;
   L.Last = NULL;
}

address alokasi(infotype X) {
   address P = new ElmList;
   P->info = X;
   P->next = NULL;
   P->prev = NULL;
   return P;
```

```
}

void dealokasi(address &P) {
   delete P;
   P = NULL;
}


void printInfo(List L) {
   if (L.First == NULL) {
      cout << "List kosong" << endl;
      return;
   }

   address P = L.First;
   while (P != NULL) {
      cout << "nopol : " << P->info.nopol << endl;
      cout << "warna : " << P->info.warna << endl;
      cout << "tahun : " << P->info.thnBuat << endl;
      cout << endl;
      P = P->next;
   }
}


void insertLast(List &L, address P) {
   if (L.First == NULL) {
      L.First = P;
      L.Last = P;
   } else {
      L.Last->next = P;
      P->prev = L.Last;
      L.Last = P;
   }
}


address findElm(List L, string nopol) {
   address P = L.First;
   while (P != NULL) {
      if (P->info.nopol == nopol) {
         return P;
      }
      P = P->next;
   }
   return NULL;
}
```

```
bool isNopolExist(List L, string nopol) {
   return findElm(L, nopol) != NULL;
}


void deleteFirst(List &L, address &P) {
   if (L.First == NULL) {
      P = NULL;
      return;
   }

   P = L.First;
   if (L.First == L.Last) {
      L.First = NULL;
      L.Last = NULL;
   } else {
      L.First = L.First->next;
      L.First->prev = NULL;
   }

   P->next = NULL;
   P->prev = NULL;
}


void deleteLast(List &L, address &P) {
   if (L.Last == NULL) {
      P = NULL;
      return;
   }

   P = L.Last;
   if (L.First == L.Last) {
      L.First = NULL;
      L.Last = NULL;
   } else {
      L.Last = L.Last->prev;
      L.Last->next = NULL;
   }

   P->prev = NULL;
}


void deleteAfter(address Prec, address &P) {
   if (Prec == NULL || Prec->next == NULL) {
      P = NULL;
```

```
    return;
  }

  P = Prec->next;
  Prec->next = P->next;
  if (P->next != NULL) {
    P->next->prev = Prec;
  }

  P->next = NULL;
  P->prev = NULL;
}


bool deleteByNopol(List &L, string nopol) {
  address P = findElm(L, nopol);
  if (P == NULL) {
    return false;
  }

  address deletedNode;
  if (P == L.First) {
    deleteFirst(L, deletedNode);
    dealokasi(deletedNode);
  } else if (P == L.Last) {
    deleteLast(L, deletedNode);
    dealokasi(deletedNode);
  } else {
    deleteAfter(P->prev, deletedNode);
    dealokasi(deletedNode);
  }

  return true;
}
```

Deskripsi:

doublylist.cpp File ini berisi implementasi fungsi-fungsi double linked list kendaraan yang dideklarasikan di doublylist.h. Program mengatur list dengan dua pointer utama, yaitu First dan Last, serta menyediakan operasi alokasi dan dealokasi node, penambahan data di akhir, penampilan seluruh data kendaraan, pencarian berdasarkan nopol, dan penghapusan data baik dari awal, akhir, maupun berdasarkan nopol tertentu. Seluruh operasi menjaga keterhubungan pointer next dan prev agar struktur double linked list tetap konsisten.

main.cpp

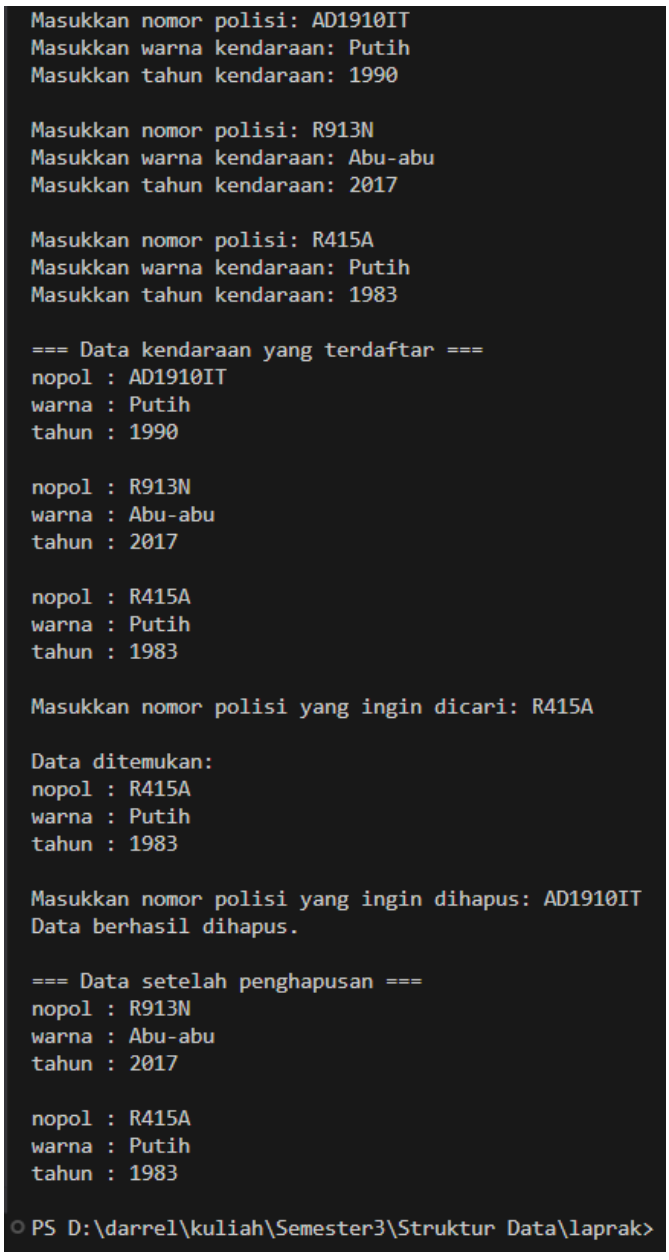
```
#include <iostream>
#include "doublylist.h"
#include "doublylist.cpp"
using namespace std;

int main() {
    List L;
    CreateList(L);

    int jumlahData = 0;
    const int MAX_DATA = 3;

    while (jumlahData < MAX_DATA) {
        infotype kendaraan;

        cout << "Masukkan nomor polisi: ";
        cin >> kendaraan.nopol;
        cout << "Masukkan warna kendaraan: ";
        cin >> kendaraan.warna;
        cout << "Masukkan tahun kendaraan: ";
        cin >> kendaraan.thnBuat;
        cout << endl;

        if (isNopolExist(L, kendaraan.nopol)) {
            cout << "Nomor polisi sudah terdaftar!" << endl << endl;
        } else {
            address P = alokasi(kendaraan);
            insertLast(L, P);
            jumlahData++;
        }
    }

    cout << "=== Data kendaraan yang terdaftar ===" << endl;
    printInfo(L);
```

```
    string nopolCari;
    cout << "Masukkan nomor polisi yang ingin dicari: ";
    cin >> nopolCari;

    address hasilCari = findElm(L, nopolCari);
    if (hasilCari != NULL) {
        cout << endl << "Data ditemukan:" << endl;
        cout << "nopol : " << hasilCari->info.nopol << endl;
        cout << "warna : " << hasilCari->info.warna << endl;
        cout << "tahun : " << hasilCari->info.thnBuat << endl;
    } else {
        cout << "Data dengan nopol tersebut tidak ditemukan." << endl;
    }

    cout << endl;

    string nopolHapus;
    cout << "Masukkan nomor polisi yang ingin dihapus: ";
    cin >> nopolHapus;

    if (deleteByNopol(L, nopolHapus)) {
        cout << "Data berhasil dihapus." << endl;
    } else {
        cout << "Data tidak ditemukan." << endl;
    }

    cout << endl << "=== Data setelah penghapusan ===" << endl;
    printInfo(L);

    return 0;
}
```

Deskripsi:

Program main ini berfungsi sebagai penguji (driver) dari modul double linked list kendaraan. Program meminta pengguna memasukkan data kendaraan (nomor polisi, warna, dan tahun) dengan batas maksimal tertentu, lalu memastikan tidak ada nomor polisi yang duplikat sebelum data disimpan ke dalam list. Setelah itu, program menampilkan seluruh data kendaraan, melakukan pencarian kendaraan berdasarkan nomor polisi, dan menyediakan fitur penghapusan data kendaraan berdasarkan nomor polisi yang dimasukkan pengguna. Hasil setiap operasi langsung ditampilkan agar perubahan pada list dapat terlihat.

Screenshots Output Unguided:



```
Masukkan nomor polisi: AD1910IT
Masukkan warna kendaraan: Putih
Masukkan tahun kendaraan: 1990

Masukkan nomor polisi: R913N
Masukkan warna kendaraan: Abu-abu
Masukkan tahun kendaraan: 2017

Masukkan nomor polisi: R415A
Masukkan warna kendaraan: Putih
Masukkan tahun kendaraan: 1983

=== Data kendaraan yang terdaftar ===
nopol : AD1910IT
warna : Putih
tahun : 1990

nopol : R913N
warna : Abu-abu
tahun : 2017

nopol : R415A
warna : Putih
tahun : 1983

Masukkan nomor polisi yang ingin dicari: R415A

Data ditemukan:
nopol : R415A
warna : Putih
tahun : 1983

Masukkan nomor polisi yang ingin dihapus: AD1910IT
Data berhasil dihapus.

=== Data setelah penghapusan ===
nopol : R913N
warna : Abu-abu
tahun : 2017

nopol : R415A
warna : Putih
tahun : 1983

PS D:\darrel\kuliah\Semester3\Struktur Data\laprak>
```

D. Kesimpulan

Program ini berhasil menerapkan struktur data double linked list untuk mengelola data kendaraan secara terorganisir. Data kendaraan disimpan dalam node yang saling terhubung dua arah sehingga memungkinkan proses penambahan, pencarian, dan penghapusan data dilakukan dengan fleksibel. Pemisahan kode ke dalam file header, file implementasi, dan file utama membuat program lebih modular, mudah dipahami, serta mudah dikembangkan. Selain itu, pengecekan nomor polisi memastikan tidak terjadi data ganda, sehingga integritas data tetap terjaga.

E. Referensi

Kaswar, A. B., & Zain, S. G. (2021). Mudah Belajar Pemrograman Dasar C++. Syiah Kuala University Press.

Hanief, S., Jepriana, I. W., & Kom, S. (2020). Konsep Algoritme dan Aplikasinya dalam Bahasa Pemrograman C++. Penerbit Andi.

Imamuddin, A., & Sobarnas, M. A. (2021). PEMBELAJARAN JARAK JAUH PEMROGRAMAN DASAR MENGGUNAKAN BAHASA C++ UNTUK UMUM: SEBUAH PROGRAM PENGABDIAN KEPADA MASYARAKAT. BEMAS: Jurnal Bermasyarakat, 1(2), 59-67.