LAPORAN PRAKTIKUM

PERTEMUAN 6

DOUBLE LINKED LIST

Nama :

RIFKI TAUFIKURROHMAN (2311104033)

Dosen :

YUDHA ISLAMI SULISTYA, S.Kom., M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

SOAL TP

1. Soal 1.cpp

```cpp
#include <iostream>
using namespace std;

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

void insertFirst(Node*& head, int data) {
  Node* newNode = new Node();
  newNode->data = data;
  newNode->prev = nullptr;
  newNode->next = head;
  if(head != nullptr) {
    head->prev = newNode;
  }
  head = newNode;
}

void insertLast(Node*& head, int data) {
  Node* newNode = new Node();
  newNode->data = data;
  newNode->next = nullptr;

  if (head == nullptr) {
    newNode->prev = nullptr;
    head = newNode;
    return;
  }

  Node* temp = head;
  while (temp->next != nullptr) {
    temp = temp->next;
  }

  temp->next = newNode;
  newNode->prev = temp;
}
```

```cpp
void displayList(Node* head) {
  Node* temp = head;
  cout << "DAFTAR ANGGOTA LIST: ";
  while (temp != nullptr) {
    cout << temp->data;
    if (temp->next != nullptr) {
        cout << " <-> ";
    }
    temp = temp->next;
  }
  cout << endl;
}

int main() {
  Node* head = nullptr;

  int firstElm, secondElm, thirdElm;
  cout << "Masukkan elemen pertama = ";
  cin >> firstElm;
  insertFirst(head, firstElm);

  cout << "Masukkan elemen kedua di awal = ";
  cin >> secondElm;
  insertFirst(head, secondElm);

  cout << "Masukkan elemen ketiga di akhir = ";
  cin >> thirdElm;
  insertLast(head, thirdElm);

  displayList(head);

  return 0;
}
```

Hasil Kode :

```
Masukkan elemen pertama = 10
Masukkan elemen kedua di awal = 5
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20
```

2. Soal 2.cpp

```cpp
#include <iostream>
using namespace std;

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

void insertFirst(Node*& head, int data) {
  Node* newNode = new Node();
  newNode->data = data;
  newNode->prev = nullptr;
  newNode->next = head;
  if (head != nullptr) {
    head->prev = newNode;
  }
  head = newNode;
}

void insertLast(Node*& head, int data) {
  Node* newNode = new Node();
  newNode->data = data;
  newNode->next = nullptr;

  if (head == nullptr) {
    newNode->prev = nullptr;
    head = newNode;
    return;
  }

  Node* temp = head;
  while (temp->next != nullptr) {
    temp = temp->next;
  }

  temp->next = newNode;
  newNode->prev = temp;
}

void deleteFirst(Node*& head) {
  if (head == nullptr) {
```

```
    cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl;
    return;
  }

  Node* temp = head;
  head = head->next;

  if (head != nullptr) {
    head->prev = nullptr;
  }

  delete temp;
}

void deleteLast(Node*& head) {
  if (head == nullptr) {
    cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl;
    return;
  }

  if (head->next == nullptr) {
    delete head;
    head = nullptr;
    return;
  }

  Node* temp = head;
  while (temp->next != nullptr) {
    temp = temp->next;
  }

  temp->prev->next = nullptr;
  delete temp;
}

void displayList(Node* head) {
  if (head == nullptr) {
    cout << "DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: List kosong." << endl;
    return;
  }
```

```
  Node* temp = head;
  cout << "DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: ";
  while (temp != nullptr) {
    cout << temp->data;
    if (temp->next != nullptr) {
        cout << " <-> ";
    }
    temp = temp->next;
  }
  cout << endl;
}

int main() {
  Node* head = nullptr;

  int firstElm, secondElm, thirdElm;
  cout << "Masukkan elemen pertama = ";
  cin >> firstElm;
  insertFirst(head, firstElm);

  cout << "Masukkan elemen kedua di akhir = ";
  cin >> secondElm;
  insertLast(head, secondElm);

  cout << "Masukkan elemen ketiga di akhir = ";
  cin >> thirdElm;
  insertLast(head, thirdElm);

  deleteFirst(head);
  deleteLast(head);
  displayList(head);
}
```

Hasil Kode :

```
Masukkan elemen pertama = 10
Masukkan elemen kedua di akhir = 15
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 15
```

3. Soal 3.cpp

```cpp
#include <iostream>
using namespace std;

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

void insertLast(Node*& head, int data) {
  Node* newNode = new Node();
  newNode->data = data;
  newNode->next = nullptr;

  if (head == nullptr) {
    newNode->prev = nullptr;
    head = newNode;
    return;
  }

  Node* temp = head;
  while (temp->next != nullptr) {
    temp = temp->next;
  }

  temp->next = newNode;
  newNode->prev = temp;
}

void displayForward(Node* head) {
  Node* temp = head;
  cout << "Daftar elemen dari depan ke belakang: ";
  while (temp != nullptr) {
    cout << temp->data;
    if (temp->next != nullptr) {
      cout << " <-> ";
    }
    temp = temp->next;
  }
  cout << endl;
}
```

```
void displayBackward(Node* head) {
  if (head == nullptr) return;

  Node* temp = head;
  while (temp->next != nullptr) {
    temp = temp->next;
  }

  cout << "Daftar elemen dari belakang ke depan: ";
  while (temp != nullptr) {
    cout << temp->data;
    if (temp->prev != nullptr) {
      cout << " <-> ";
    }
    temp = temp->prev;
  }
  cout << endl;
}

int main() {
  Node* head = nullptr;
  int n, data;

  cout << "Masukkan 4 elemen secara berurutan: ";
  for (int i = 0; i < 4; i++) {
    cin >> data;
    insertLast(head, data);
  }

  displayForward(head);
  displayBackward(head);
}
```

Hasil Kode :

```
Masukkan 4 elemen secara berurutan: 1 2 3 4
Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1
```

UNGUIDED

1. Latihan 1
    - File doublelist.h

```cpp
#ifndef DOUBLELIST_H
#define DOUBLELIST_H
#include <iostream>
#include <string>
using namespace std;

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

struct ElmList;
typedef ElmList* address;

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

struct List {
  address First;
  address Last;
};

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

address alokasi(infotype x) {
  address newElm = new ElmList;
  newElm->info = x;
  newElm->next = nullptr;
  newElm->prev = nullptr;
  return newElm;
}

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

bool isNopolExists(const List &L, const string &nopol) {
  address temp = L.First;
  while (temp != nullptr) {
    if (temp->info.nopol == nopol) {
      return true;
    }
    temp = temp->next;
  }
  return false;
}

address findElm(const List &L, const string &nopol) {
  address temp = L.First;
  while (temp != nullptr) {
    if (temp->info.nopol == nopol) {
      return temp;
    }
    temp = temp->next;
  }
  return nullptr;
}

void printSearchResult(const List &L, const string &nopol) {
  address found = findElm(L, nopol);
  if (found != nullptr) {
    cout << "\nNomor Polisi : " << found->info.nopol << endl;
    cout << "Warna : " << found->info.warna << endl;
    cout << "Tahun : " << found->info.thnBuat << endl;
  } else {
    cout << "Nomor polisi " << nopol << " tidak ditemukan." << endl;
  }
}

void deleteElm(List &L, const string &nopol) {
  address found = findElm(L, nopol);
  if (found != nullptr) {
    if (found == L.First) {
      L.First = found->next;
      if (L.First != nullptr) {
        L.First->prev = nullptr;
      }
    } else if (found == L.Last) {
      L.Last = found->prev;
      if (L.Last != nullptr) {
        L.Last->next = nullptr;
      }
    } else {
      found->prev->next = found->next;
      found->next->prev = found->prev;
    }
    delete found;
    cout << "Data dengan nomor polisi " << nopol << " berhasil dihapus." <<
endl;
  } else {
    cout << "Nomor polisi " << nopol << " tidak ditemukan." << endl;
  }
}

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

  address temp = L.First;
  cout << "\nDATA LIST\n\n";
  while (temp != nullptr) {
    cout << "no polisi : " << temp->info.nopol << endl;
    cout << "warna : " << temp->info.warna << endl;
    cout << "tahun : " << temp->info.thnBuat << endl << endl;
    temp = temp->next;
  }
}

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

#endif
```

- File latihan1.cpp

```cpp
#include "doublelist.h"
int main() {
  List L;
  CreateList(L);
  infotype kendaraan;
  string nopol;

  for (int i = 0; i < 4; i++) {
    cout << "masukkan nomor polisi: ";
    cin >> kendaraan.nopol;

    if (isNopolExists(L, kendaraan.nopol)) {
        cout << "nomor polisi sudah terdaftar" <<
endl;   cout << endl;
        continue;
    }

    cout << "masukkan warna kendaraan: ";
    cin >> kendaraan.warna;
    cout << "masukkan tahun kendaraan: ";
    cin >> kendaraan.thnBuat;
    cout << endl;

    address P = alokasi(kendaraan);
    insertLast(L, P);
  }

  printInfo(L);

  string searchNopol;
  cout << endl;
  cout << "Masukkan Nomor Polisi yang dicari : ";
  cin >> searchNopol;

  printSearchResult(L, searchNopol);

  string deleteNopol;
  cout << endl;
  cout << "Masukkan Nomor Polisi yang akan dihapus : ";
  cin >> deleteNopol;

  deleteElm(L, deleteNopol);

  printInfo(L);
}
```

Hasil Kode :

```
masukkan nomor polisi: D001
masukkan warna kendaraan: HITAM
masukkan tahun kendaraan: 90

masukkan nomor polisi: D003
masukkan warna kendaraan: PUTIH
masukkan tahun kendaraan: 70

masukkan nomor polisi: D001
nomor polisi sudah terdaftar

masukkan nomor polisi: D004
masukkan warna kendaraan: KUNING
masukkan tahun kendaraan: 90


DATA LIST

no polisi : D001
warna : HITAM
tahun : 90

no polisi : D003
warna : PUTIH
tahun : 70

no polisi : D004
warna : KUNING
tahun : 90
```

Cari nomer polisi dan hapus nomer polisi :

```
Masukkan Nomor Polisi yang dicari : D001

Nomor Polisi : D001
Warna : HITAM
Tahun : 90

Masukkan Nomor Polisi yang akan dihapus : D003
Data dengan nomor polisi D003 berhasil dihapus.

DATA LIST

no polisi : D001
warna : HITAM
tahun : 90

no polisi : D004
warna : KUNING
tahun : 90
```