

Tugas Pendahuluan Modul 4
STRUKTUR DATA - Ganjil 2024/2025
"Single_Linked_List_Bagian_1"

Ketentuan Tugas Pendahuluan

1. Tugas Pendahuluan dikerjakan secara **Individu**.
2. TP ini bersifat **WAJIB**, tidak mengerjakan = **PENGURANGAN POIN JURNAL / TES ASESMEN**.
3. Hanya **MENGUMPULKAN** tetapi **TIDAK MENGERJAKAN** = **PENGURANGAN POIN JURNAL / TES ASESMEN**.
4. Deadline pengumpulan TP Modul 2 adalah Senin, 30 September 2024 pukul 07.30 WIB.
5. **TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP MAKA DIANGGAP TIDAK MENGERJAKAN**.
6. **DILARANG PLAGIAT (PLAGIAT = E)**.
7. Kerjakan TP dengan jelas agar dapat dimengerti.
8. Codingan diupload di Github dan upload Laporan di Lab menggunakan format **PDF** dengan ketentuan:
TP_MOD_[XX]_NIM_NAMA.pdf

CP (WA):

- Andini (082243700965)
- Imelda (082135374187)

SELAMAT MENGERJAKAN^^

LAPORAN PRAKTIKUM
PERTEMUAN 4
Single_Linked_List_Bagian_1



Nama :

Zulfa Mustafa Akhyar Iswahyudi (2311104010)

Dosen :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

A. Tujuan

Untuk melatih kompetensi Mahasiswa untuk memperdalam skill pemrograman C++

B. Tools

Codeblocks, VSCode, Github

TUGAS PENDAHULUAN

1.) List.h

Secara keseluruhan, dibawah ini adalah program untuk melakukan *reversal* (*pembalikan*) urutan beberapa inputan yang dilakukan oleh user ke sintaks program. Kita bagi programnya menjadi tiga bagian. Satu untuk sub-class **list.h**, satu untuk menampung seluruh method dan fungsi dalam **list.cpp**, dan terakhir adalah **main.cpp** yang menjadi class operasional utama.

=====

Untuk sub-class yaitu **list.h** adalah sub-class yang mendeklarasikan **definite/fungsi** dari elemen *infotype* dan *elmlist* yang akan menjadi alamat data dari **linkedlist**. Jangan lupa untuk konfigurasi pertama kali yang harus dilakukan adalah mendefinisikan fungsi **first(L)** yang dimana sebagai settingan default suatu nilai dalam barisan data.

Jadi untuk inisiasi pertama kali dalam **linkedlist**, data yang akan terbaca duluan adalah data paling depan dalam **linkedlist** data, jadi begitu simpelnya. Selanjutnya ada fungsi **next(p)** dan **info(p)**.

P adalah variabel data tertentu,

Sementara fungsi **next** adalah fungsi untuk memajukan variabel tertentu dalam **linkedlist** sebanyak 1 langkah. Sedangkan

info adalah fungsi untuk menampilkan hasil kalkulasi program terhadap data **linkedlist**. Lalu kita mulai membuat **constructor** pada *elmlist* yang diisikan fungsi *infotype* dengan inisialisasi **info**, lalu fungsi *address* dengan inisialisasi **next** dan *List* yang diisikan fungsi *address* dengan inisialisasi **first**.

=====

Langkah Paling akhir disini adalah membuat method untuk menciptakan **linkedlist**, membuat fungsi pengalokasian data, method untuk memasukkan data pada **linkedlist** berdasarkan alamatnya, dan terakhir adalah method untuk menampilkan data **linkedlist**-nya.

Glosarium :

- **List &L** : Ini adalah **sintaks** untuk pemanggilan **linkedList** yang dideklarasikan beserta dengan **alamat linkedlist** data-nya.
- **Infotype x** : Ini adalah sintaks untuk **menargetkan serta mengidentifikasi nilai/data tertentu pada sebuah linkedlist** yang bisa dimanipulasi seperti dihapus, dimajukan 1,2,3 langkah, atau dimundurkan sebanyak langkah sesuai inputan seperti halnya memajukan data.
- **Address p** : Ini adalah sintaks untuk pencarian **alamat** dari suatu **data(p)** tertentu agar bisa dialokasikan ke posisi tertentu dalam **linkedList**.

The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a project structure under 'PRAKTIKUM_4'. The files listed are: .vscode, KodinganBare..., LatihanUnguid..., output, buatSLL.cpp, cariHitungLLL..., hapusNode.cpp, TugasPendahu..., output, list.cpp, list.h (selected), main.cpp, and main.exe. The main editor area shows the content of 'list.h' with line numbers 1 through 24. The code defines a linked list structure with macros for first, next, and info, and function prototypes for createList, allocate, insertFirst, and printInfo.

```
1  #include <iostream>
2  #define first(L) L.first
3  #define next(P) P->next
4  #define info(P) P->info
5
6  using namespace std;
7  typedef int infotype;
8  typedef struct elmlist *address;
9
10 struct elmlist
11 {
12     infotype info;
13     address next;
14 };
15
16 struct List
17 {
18     address first;
19 };
20
21 void createList(List &L);
22 address allocate(infotype x);
23 void insertFirst(List &L, address p);
24 void printInfo(List L);
```

2.) List.cpp

Lanjut ke file untuk implementasi variabel definisi beserta variabel construct yang sudah dibuat, yaitu list.cpp. Kita perlu memanggil sub-class list.h agar semua pendeklarasian dapat terdeteksi dan sistem dapat mengidentifikasi dengan baik keseluruhan deklarasinya.

=====

Buat method **createList** untuk menciptakan **linkedlist** yang parameter-nya itu **<List &L>** yang memuat fungsi insialisasi sebuah List **Linkedlist**. Didalam method-nya tambahkan juga definite/fungsi definisi **<first(L)>** yang kita atur kosong diawal.

=====

Selanjutnya adalah method khusus untuk pengalokasian sebuah nilai berdasarkan fungsi definisi alamat, parameter-nya **<infotype x>** yang memuat fungsi untuk menargetkan sebuah variabel tertentu.

Didalam method-nya berisi deklarasi baru dari constructor **elmList** yang diinisialisasi fungsi **<address p>**. Kemudian atur default **data(p)** agar bisa ditampilkan dan data selanjutnya adalah kosong dulu. Lalu buat pengembalian nilai yang akan menjadi output.

=====

Selanjutnya buat method **insertFirst** agar data inputan bisa masuk kedalam **linkedlist** yang sudah dibuat sebelumnya. Dengan dua parameter **<List &L>** dan **<address p>** kita sudah dapat identifikasi data beserta **linkedlist**-nya yang akan kita manipulasi. Didalam method ini kita hanya perlu mengatur bahwa **data(p)** akan ditaruh diurutan paling awal didepan **linkedlist**.

=====

Terakhir adalah method **printInfo** untuk menampilkan keseluruhan **linkedlist**, parameter utamanya adalah **<List L>**. Karena kita ingin menampilkan keseluruhan data **linkedlist**, maka kita gunakan notasi **L** dan bukan **&L**. Didalamnya adalah pengkondisian jika data **linkedlist** tidak kosong maka **linkedlist** akan ditampilkan dalam output. Satu per satu data akan ditampilkan seluruhnya dengan fungsi **<next(p)>** untuk membacaa data selanjutnya setelah data pertama.

EXPLORER

▼ PRAKTIKUM_4

> .vscode

▼ KodanganBare...

C++ gui_1.cpp U

C++ gui_2.cpp U

▼ LatihanUnguid...

> output

C++ buatSLL.cpp U

C++ cariHitungLLL.... U

C++ hapusNode.cpp U

▼ TugasPendahu...

> output

C++ list.cpp U

h list.h U

C++ main.cpp U

main.exe U

OUTLINE

C++ list.cpp U X

C++ gui_2.cpp U

C++ main.cpp U

TugasPendahuluan > C++ list.cpp > printInfo(List)

1

#include <iostream>

2

#include "list.h"

3

using namespace std;

4

Qodo Gen: Options | Test this function

5

void createList(List &L)

6

{

7

first(L) = NULL;

8

}

Qodo Gen: Options | Test this function

9

address allocate(infotype x)

10

{

11

address p = new elmlist;

12

info(p) = x;

13

next(p) = NULL;

14

return p;

15

}

16

Qodo Gen: Options | Test this function

17

void insertFirst(List &L, address p)

18

{

19

next(p) = first(L);

20

first(L) = p;

21

}

22

Qodo Gen: Options | Test this function

23

void printInfo(List L)

24

{

25

address p = first(L);

26

while (p != NULL)

27

{

28

cout << info(p) << " ";

29

p = next(p);

30

}

31

cout << endl;

32

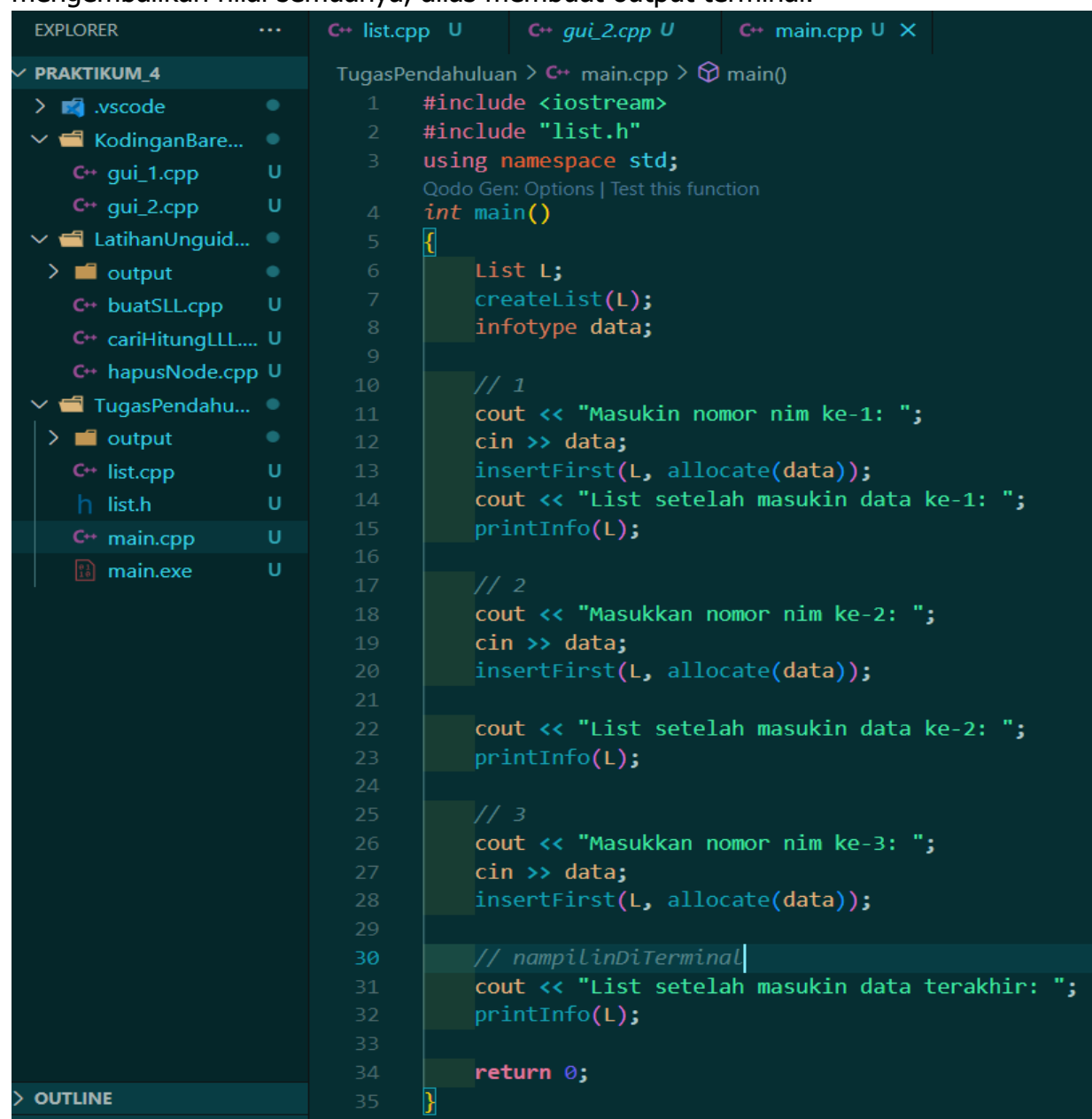
}

3.) Main.cpp

Masuk Class utama yang mengoperasikan seluruh method dan fungsi, main.cpp. Kita perlu meng-import sub-class **list.h** agar seluruh deklarasinya dapat terdeteksi dan diimplementasikan dengan lancar. Deklarasikan **<List L>** lalu buat **linkedlist** dengan fungsi **createList** dengan inisialisasi **L**, kemudian deklarasi variabel **infotype** untuk menargetkan data didalam **linkedlist**.

=====

Saatnya mengeksekusi seluruh deklarasi, **sekarang kita buat fungsi inputan dari user dengan inisiasi data inputannya akan dialokasikan kedalam linkedlist dengan fungsi insertFirst, kemudian ditampilkan di Output, ulangi proses ini sampai tiga kali.** Setelahnya buat return untuk mengembalikan nilai semuanya, alias membuat output terminal.



```
EXPLORER
PRAKTIKUM_4
  .vscode
  KodinganBare...
    gui_1.cpp
    gui_2.cpp
  LatihanUnguid...
    output
    buatSLL.cpp
    cariHitungLLL....
    hapusNode.cpp
  TugasPendahu...
    output
    list.cpp
    list.h
    main.cpp
    main.exe

TugasPendahuluan > C++ main.cpp > main()
1  #include <iostream>
2  #include "list.h"
3  using namespace std;
4  int main()
5  {
6      List L;
7      createList(L);
8      infotype data;
9
10     // 1
11     cout << "Masukin nomor nim ke-1: ";
12     cin >> data;
13     insertFirst(L, allocate(data));
14     cout << "List setelah masukin data ke-1: ";
15     printInfo(L);
16
17     // 2
18     cout << "Masukkan nomor nim ke-2: ";
19     cin >> data;
20     insertFirst(L, allocate(data));
21
22     cout << "List setelah masukin data ke-2: ";
23     printInfo(L);
24
25     // 3
26     cout << "Masukkan nomor nim ke-3: ";
27     cin >> data;
28     insertFirst(L, allocate(data));
29
30     // nampilinDiTerminal
31     cout << "List setelah masukin data terakhir: ";
32     printInfo(L);
33
34     return 0;
35 }
```


Output :

```
• n> g++ main.cpp list.cpp -o main
PS C:\Users\HUAWEI\OneDrive\Documents\ALL_I
• n> ./main
Masukin nomor nim ke-1: 0
List setelah masukin data ke-1: 0
Masukkan nomor nim ke-2: 1
List setelah masukin data ke-2: 1 0
○ Masukkan nomor nim ke-3: 0
List setelah masukin data terakhir: 0 1 0
```

Latihan – UNGUIDED

1.) Membuat Single Linked List

Kali ini adalah program untuk menciptakan sebuah barisan **linkedlist** berbasis SLL. Sebelum kita menginputkan secara manual data-data yang ingin kita bariskan, lebih baik buat dulu method-method penting untuk menambahkan data didepan, menambahkan data dibelakang, juga jangan lupa buat method untuk menampilkan barisan list **LinkedList**-nya.

Untuk deklarasi awal ada constructor untuk Node yang isi konfigurasi-nya ada variabel data yang di-set dengan integer dan Node itu sendiri yang di-set *next agar data-data inputan bisa masuk kedalam barisan list **LinkedList**.

Menurut saya pribadi, Node itu adalah variabel untuk membentuk struktur **LinkedList**. Jadi barisan list data yang kita buat sumber utamanya adalah Node yang dideklarasikan diawal.

=====

Untuk method 'depan', kita perlu **masukkan Node beserta alamatnya dan variabel baru yaitu nilai dengan set integer kedalam parameter-nya agar data dapat diidentifikasi dengan inputan berupa angka**. Didalam method ini kita perlu deklarasi Node menjadi variabel fungsi baru supaya struktur **linkedlist** tetap solid saat ada tambahan data baru.

Atur NewNode yang sudah dideklarasikan ulang tadi agar data diidentifikasi sebagai nilai dan fungsi definisi next sebagai head. Head ini selain dia adalah data pertama dalam linkedlist, diidentifikasi sebagai newNode juga. **Artinya tujuan dari korelasi antara fungsi definisi next dengan variabel Head(nilai**

pertama dalam linkedlist) adalah agar linkedlist dapat terus meregenerasi indeks-nya saat inputan baru masuk kedalam linkedlist.

=====

Selanjutnya ada method 'belakang'. Saya ga perlu panjang lagi jelasinnya, tapi intinya sintaks dan deklarasi didalam method ini jelas sama seperti method 'depan' hanya saja perbedaannya adalah deklarasi newNode dengan next adalah di-set menjadi nullptr yang dimana itu merujuk ke nilai paling akhir/belakang dalam **linkedlist**. Artinya data yang diinputkan tidak akan berada didepan data sebelumnya.

Dari kekosongan posisi data ini, kita perlu kondisikan kalo nilai pertama dalam **linkedlist** ada di paling belakang **linkedlist**, maka 'head' ini akan tetap meregenerasi indeks **linkedlist**. Namun jika tidak kosong, maka variabel 'head' akan diinisialisasikan sebagai Node sementara. Kondisikan lagi selama data selanjutnya setelah Node sementara itu bukan dibelakang **linkedlist**, maka Node sementara tadi akan menjadi indeks baru dalam **linkedlist**, artinya Node sementara tadi berubah menjadi nilai asli didalam **linkedlist**. Namun posisinya akan berada dibelakang nilai awal.

=====

Lalu ada method 'ngasihLiatList'. Gampangnya ini method untuk menampilkan seluruh baris **linkedlist**. Cukup parameter-kan Node dengan nilai pertamanya, kemudian didalam method ini kondisikan selama Node sementara bukan nilai paling belakang, maka data **linkedlist** akan ditampilkan secara urut.

=====

Terakhir adalah program untuk inputannya. **Kita ga pake cout untuk outputnya, tetapi cukup panggil method yang sudah dibuat lalu deklarasikan jumlah nilainya dalam bentuk angka.** Jangan lupa deklarasikan 'head' yang mewakili variabel nilai yang akan masuk ke dalam **linkedlist**.

Sama diatasnya itu deklarasiin dulu Node-nya dengan alamat nilainya. Diakhir line, panggil method 'ngasihLiatList' dengan parameter data didalamnya agar terminal menampilkan barisan **linkedlist**-nya'

EXPLORER

PRAKTIKUM_4

> .vscode

> KodinganBare...

> gui_1.cpp

> gui_2.cpp

> LatihanUnguid...

> output

> buatSLLcpp

> cariHitungLLL...

> hapusNode.cpp

> TugasPendahu...

C- buatSLLcpp U X

LatihanUnguid > C- buatSLLcpp > main()

```
1  #include <iostream>
2  using namespace std;
3  struct Node
4  {
5      int data;
6      Node *next;
7  };
8  Qodo Gen: Options | Test this function
9  void depan(Node *&head, int nilai)
10 {
11     Node *newNode = new Node();
12     newNode->data = nilai;
13     newNode->next = head;
14     head = newNode;
15 }
16 Qodo Gen: Options | Test this function
17 void belakang(Node *&head, int nilai)
18 {
19     Node *newNode = new Node();
20     newNode->data = nilai;
21     newNode->next = nullptr;
22
23     if (head == nullptr)
24     {
25         head = newNode;
26     }
27     else
28     {
29         Node *temp = head;
30         while (temp->next != nullptr)
31         {
32             temp = temp->next;
33         }
34         temp->next = newNode;
35     }
36 }
37 Qodo Gen: Options | Test this function
38 void ngasihLiatList(Node *head)
39 {
40     Node *temp = head;
41     while (temp != nullptr)
42     {
43         cout << temp->data << " -> ";
44         temp = temp->next;
45     }
46     cout << "NULL" << endl;
47 }
48 Qodo Gen: Options | Test this function
49 int main()
50 {
51     Node *head = nullptr;
52     depan(head, 10);
53     belakang(head, 20);
54     depan(head, 5);
55     belakang(head, 30);
56     depan(head, 40);
57     depan(head, 50);
58     belakang(head, 60);
59     belakang(head, 70);
60
61     ngasihLiatList(head);
62
63     return 0;
64 }
```

Output :

```
PS C:\Users\HUAWEI\OneDrive\Documents\ALL_ITTP_SEMEST  
'buatSLL.exe'  
50 -> 40 -> 5 -> 10 -> 20 -> 30 -> 60 -> 70 -> NULL
```

2.) Menghapus Node pada Linked List

Selanjutnya adalah sintaks program untuk menghapus sebuah Node atau data dalam LinkedList. Class kali ini tidak berbeda dengan Class sebelumnya, namun ada penambahan method baru, yaitu method **hapusNode**.

Parameter-kan dulu deklarasi Node beserta seluruh alamat data **linkedlist**, serta deklarasi nilai yang di-set integer. **Didalamnya ada pengkondisian bercabang yang rumit, pertama adalah pengkondisian dimana data pertama di linkedlist ada di paling belakang, maka method akan mengembalikan nilai.**

Pada pengkondisian berikutnya, **jika data tertentu dalam barisan linkedlist ditemukan nilai-nya, maka data tersebut akan menjadi Node sementara kemudian dihapus dari linkedlist-nya**, lalu method akan mengembalikan nilai.

Sekarang deklarasikan bahwa Node sementara adalah data dalam list **linkedlist** juga. **Buat perulangan dimana selama nilai sementara dari Node tadi tidak berada dibelakang linkedlist dan nilai sementara dalam keseluruhan data dalam linkedlist tidak ditemukan nilai yang ditargetkan**, maka nilai sementara tadi akan dimajukan posisinya untuk menggantikan posisi data yang telah hilang.

Kondisikan lagi jika nilai sementara dan nilai seterusnya tidak ada dibelakang **linkedlist**, maka Node akan mendeklarasikan nilai sementara ini dengan **nodeToDelete** yang nantinya akan menghapus nilai yang sesuai.

Lalu pada penginputan manual di **main()**, kita bisa tambahkan sintaks baru yaitu 'hapusNode', kemudian deklarasikan variabel head dengan data yang ingin dihapus. Disitu kita contohkan 50, maka di terminal angka 50 sudah hilang.

```

35 void hapusNode(Node *&head, int nilai)
36 {
37     if (head == nullptr)
38         return;
39
40     if (head->data == nilai)
41     {
42         Node *temp = head;
43         head = head->next;
44         delete temp;
45         return;
46     }
47     Node *temp = head;
48     while (temp->next != nullptr && temp->next->data != nilai)
49     {
50         temp = temp->next;
51     }
52
53     if (temp->next != nullptr)
54     {
55         Node *nodeToDelete = temp->next;
56         temp->next = temp->next->next;
57         delete nodeToDelete;
58     }
59 }

```

Output :

```

PS C:\Users\HUAWEI\OneDrive\Documents\ALL_ITTP_SE...
'hapusNode.exe'
40 -> 5 -> 10 -> 20 -> 30 -> 60 -> 70 -> NULL

```

3.) Mencari dan Menghitung Panjang Linked List

Untuk terakhir kalinya dalam laporan ini, adalah Class untuk mencari sebuah nilai tertentu beserta panjang list **linkedlist**-nya.

=====

Pada method '**cariNode**', kita akan parameter-kan Node dengan seluruh data dan deklarasi nilai bertipe integer. Kondisikan langsung apabila nilai dalam data **linkedlist** sesuai dengan nilai yang kita cari, maka method akan mengembalikan nilai **True yang artinya nilai yang ditargetkan telah ditemukan**.

Namun jika pendeteksian dilakukan terus pada seluruh nilai seterusnya dan sampai akhir tidak ada, maka akan **False yang artinya nilai tak ditemukan**.

=====

Selanjutnya adalah method '**panjangList**', cukup Node beserta seluruh datanya sebagai parameter. Lalu set awal 'count' jadi nol sebagai awal mula pengecekan, kemudian atur nilai Node awal dimulai dari nilai pertama dalam **linkedlist**.

Kondisikan selama nilai yang dicari tak ada dibelakang **linkedlist**, maka fungsi 'count' tadi akan terus bertambah untuk mencari nilai yang kita cari. Setelah ditemukan maka method akan mengembalikan nilai 'count'.

=====

Disini 'count' mewakili pencarian nilai yang kita cari sebelumnya.

```
39  bool cariNode(Node *head, int nilai)
40  {
41      Node *temp = head;
42      while (temp != nullptr)
43      {
44          if (temp->data == nilai)
45          {
46              return true;
47          }
48          temp = temp->next;
49      }
50      return false;
51  }
52
53  Qodo Gen: Options | Test this function
54  int panjangList(Node *head)
55  {
56      int count = 0;
57      Node *temp = head;
58      while (temp != nullptr)
59      {
60          count++;
61          temp = temp->next;
62      }
63      return count;
64  }
```

Ah iya, jangan lupa juga pada **main()**, kita buat inputan manualnya dengan nilai angka-angka yang masih sama. **Namun kondisikan apabila method yang mencari nilai 60 menemukan nilainya, maka akan terbentuk output identifikasi yang sesuai nilainya.** Namun jika tidak, maka program akan mengatakan kalo Node = 60 tidak ditemukan.
Terakhir, buat output yang menampilkan keseluruhan panjang **linkedlist**-nya.

Output :

SOURCE CODE WAKTU DI KELAS – KODINGAN ASPRAK

```
Gu1_1.cpp

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

// Deklarasi Struct untuk mahasiswa
struct mahasiswa
{
    char nama[50];
    char nim[10];
};

// Deklarasi Struct Node
struct Node
{
    mahasiswa data;
    Node *next;
};

Node *head;
Node *tail;

// Inisialisasi List
void init()
{
    head = nullptr;
    tail = nullptr;
}

// Mengecek apakah list kosong
bool isEmpty()
{
    return head == nullptr;
}

// Tambah Depan
void insertDepan(const mahasiswa &data)
{
    Node *baru = new Node;
    baru->data = data;
    baru->next = head;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

// Tambah Belakang
void insertBelakang(const mahasiswa &data)
{
    Node *baru = new Node;
    baru->data = data;
    baru->next = nullptr;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

// Hitung Jumlah List
int hitungList()
{
    Node *current = head;
    int jumlah = 0;
    while (current != nullptr)
    {
        jumlah++;
        current = current->next;
    }
    return jumlah;
}

// Hapus Depan
void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        head = head->next;
        delete hapus;
        if (head == nullptr)
        {
            tail = nullptr; // jika list menjadi kosong
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

// Hapus Belakang
void hapusBelakang()
{
    if (!isEmpty())
    {
        if (head == tail)
        {
            delete head;
            head = tail = nullptr; // list menjadi kosong
        }
        else
        {
            Node *bantu = head;
            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }
            delete tail;
            tail = bantu;
            tail->next = nullptr;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

// Tampilkan List
void tampil()
{
    Node *current = head;
    if (!isEmpty())
    {
        while (current != nullptr)
        {
            cout << "Nama: " << current->data.nama << ", NIM: " << current->data.nim << endl;
            current = current->next;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus List
void clearList()
{
    Node *current = head;
    while (current != nullptr)
    {
        Node *hapus = current;
        current = current->next;
        delete hapus;
    }
    head = tail = nullptr;
    cout << "List berhasil terhapus!" << endl;
}

// Main function
int main()
{
    init();

    // Contoh data mahasiswa
    mahasiswa m1 = {"Alice", "123456"};
    mahasiswa m2 = {"Bob", "789012"};
    mahasiswa m3 = {"Charlie", "112233"};

    // Menambahkan mahasiswa ke dalam list
    insertDepan(m1);
    tampil();
    insertBelakang(m2);
    tampil();
    insertDepan(m3);
    tampil();

    // Menghapus elemen dari list
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();

    // Menghapus seluruh list
    clearList();
    return 0;
}
```




```

#include <iostream>
using namespace std;

// Definisi struktur untuk elemen list
struct Node
{
    int data; // Menyimpan nilai elemen
    Node *next; // Pointer ke elemen berikutnya
};

// Fungsi untuk mengalokasikan memori untuk node baru
Node *alokasi(int value)
{
    Node *newNode = new Node; // Alokasi memori untuk elemen baru
    if (newNode != nullptr)
    {
        newNode->data = value; // Jika alokasi berhasil
        newNode->next = nullptr; // Mengisi data node
        // Set next ke nullptr
    }
    return newNode; // Mengembalikan pointer node baru
}

// Fungsi untuk dealokasi memori node
void dealokasi(Node *node)
{
    delete node; // Mengembalikan memori yang digunakan oleh node
}

// Pengecekan apakah list kosong
bool isListEmpty(Node *head)
{
    return head == nullptr; // List kosong jika head adalah nullptr
}

// Menambahkan elemen di awal list
void insertFirst(Node *head, int value)
{
    Node *newNode = alokasi(value); // Alokasi memori untuk elemen baru
    if (newNode != nullptr)
    {
        newNode->next = head; // Menghubungkan elemen baru ke elemen pertama
        head = newNode; // Menetapkan elemen baru sebagai elemen pertama
    }
}

// Menambahkan elemen di akhir list
void insertLast(Node *head, int value)
{
    Node *newNode = alokasi(value); // Alokasi memori untuk elemen baru
    if (newNode != nullptr)
    {
        if (isListEmpty(head))
        {
            head = newNode; // Jika list kosong
        }
        else
        {
            Node *temp = head;
            while (temp->next != nullptr)
            {
                temp = temp->next;
            }
            temp->next = newNode; // Menambahkan elemen baru di akhir list
        }
    }
}

// Menampilkan semua elemen dalam list
void printList(Node *head)
{
    if (isListEmpty(head))
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        Node *temp = head;
        while (temp != nullptr)
        {
            cout << temp->data << " "; // Selama belum mencapai akhir list
            temp = temp->next; // Menampilkan data elemen
            // Melanjutkan ke elemen berikutnya
        }
        cout << endl;
    }
}

// Menghitung jumlah elemen dalam list
int countElements(Node *head)
{
    int count = 0;
    Node *temp = head;
    while (temp != nullptr)
    {
        count++; // Menambah jumlah elemen
        temp = temp->next; // Melanjutkan ke elemen berikutnya
    }
    return count; // Mengembalikan jumlah elemen
}

// Menghapus semua elemen dalam list dan dealokasi memori
void clearList(Node *head)
{
    while (head != nullptr)
    {
        Node *temp = head; // Simpan pointer ke node saat ini
        head = head->next; // Pindahkan ke node berikutnya
        dealokasi(temp); // Dealokasi node
    }
}

int main()
{
    Node *head = nullptr; // Membuat list kosong

    // Menambahkan elemen ke dalam list
    insertFirst(head, 10); // Menambahkan elemen 10 di awal list
    insertLast(head, 20); // Menambahkan elemen 20 di akhir list
    insertLast(head, 30); // Menambahkan elemen 30 di akhir list

    // Menampilkan isi list
    cout << "Isi List: ";
    printList(head);

    // Menampilkan jumlah elemen
    cout << "Jumlah elemen: " << countElements(head) << endl;

    // Menghapus semua elemen dalam list
    clearList(head);

    // Menampilkan isi list setelah penghapusan
    cout << "Isi list setelah penghapusan: ";
    printList(head);

    return 0;
}

```



2311104010_ZULFA MUSTAFA AKHYAR ISWAHYUDI
 Han-Guo-Rang

Semoga Selalu diberi kemudahan^^