LAPORAN PRAKTIKUM

MODUL 5

"SINGLE LINKED LIST (BAGIAN KEDUA)"

Disusun Oleh:

Tiurma Grace Angelina (2311104042)

SE-07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. **Tujuan**
   - Memahami penggunaan linked list dengan pointer operator-operator dalam program.
   - Memahami operasi-operasi dasar dalam linked list.
   - Membuat program dengan menggunakan linked list dengan prototype yang ada.

2. **Landasan Teori**
   - **Linked List**: Linked list adalah struktur data yang terdiri dari rangkaian elemen (node) yang masing-masing memiliki data dan penunjuk (pointer) ke elemen berikutnya dalam daftar. Berbeda dengan array yang memiliki ukuran tetap dan alamat memori yang berurutan, linked list dapat bertambah atau berkurang ukurannya secara dinamis, dan elemen-elemennya tidak disimpan di lokasi memori yang berdekatan. Terdapat beberapa jenis linked list, yaitu:
   - **Single Linked List**: Setiap node hanya memiliki pointer ke node berikutnya.
   - **Doubly Linked List**: Setiap node memiliki pointer ke node sebelumnya dan node berikutnya.
   - **Circular Linked List**: Linked list yang node terakhirnya menunjuk ke node pertama.

   - **Single Linked List**: Pada single linked list, setiap node hanya memiliki pointer yang menunjuk ke node berikutnya. Hal ini memungkinkan penelusuran elemen dari awal hingga akhir linked list, tetapi tidak memungkinkan penelusuran ke arah sebaliknya. Struktur ini lebih sederhana dibandingkan doubly linked list dan lebih hemat memori karena hanya memiliki satu pointer per node.
   - **Operasi Dasar pada Linked List**:
   - **Inisialisasi**: Menginisialisasi linked list dengan mengatur pointer First (atau head) ke NULL, yang menunjukkan bahwa linked list tersebut kosong.
   - **Penambahan Elemen**: Elemen dapat ditambahkan pada awal atau akhir linked list. Pada operasi penambahan di awal, elemen baru menjadi elemen pertama, sementara pada penambahan di akhir, elemen baru ditambahkan setelah elemen terakhir.
   - **Penghapusan Elemen**: Penghapusan elemen dapat dilakukan dengan memutuskan hubungan antara elemen yang akan dihapus dengan elemen lainnya, dan menghubungkan ulang elemen-elemen lain jika diperlukan.
   - **Pencarian Elemen**: Pencarian elemen dilakukan dengan membandingkan setiap nilai data dalam linked list dengan nilai yang dicari hingga ditemukan kecocokan.
   - **Penghitungan Total Elemen**: Untuk menghitung total nilai elemen, program melakukan iterasi melalui linked list dan menjumlahkan setiap nilai data (info) yang ditemukan.
   - **Pointer dan Dynamic Memory Allocation**: Linked list menggunakan konsep pointer dan dynamic memory allocation untuk mengelola node. new digunakan untuk mengalokasikan memori untuk node baru secara dinamis,

sedangkan delete digunakan untuk membebaskan memori node yang tidak lagi diperlukan, sehingga membantu menghindari kebocoran memori.

- **Fungsi dalam Program**: Beberapa fungsi penting dalam program ini antara lain:
- createList untuk menginisialisasi linked list.
- alokasi untuk mengalokasikan memori node baru.
- insertFirst untuk menambahkan elemen di awal linked list.
- findElm untuk mencari elemen berdasarkan nilai tertentu.
- sumInfo untuk menjumlahkan semua nilai data (info) dalam linked list.

3. Guided

- **Array**

1. CODE :

```cpp
#include <iostream>

using namespace std;

// Struktur untuk node dalam linked list
struct Node {
    int data;
    Node* next;
};

// fungsi untuk menambahkan elemen baru ke awal linkded list
void insertFirst(Node*& head, Node*& tail, int new_data){
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = head;
    head = new_node;
    head = new_node;

    if (tail == nullptr){
        tail = new_node;
    }
}

// fungsi menambahkan elemen baru ke akhir linked list
void insertLast(Node*& head, Node*& tail, int new_data){
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = nullptr;

```

```
28      new_node->next = nullptr;
29
30      if (head == nullptr){
31          head = new_node;
32          tail = new_node;
33      } else {
34          tail->next = new_node;
35          tail = new_node;
36      }
37  }
38
39  // fungsi mencari elemen dalam linked list
40  int findElement(Node* head, int x){
41      Node* current = head;
42      int index = 0;
43
44      while (current != nullptr){
45          if (current->data == x){
46              return index;
47          }
48          current = current->next;
49          index++;
50      }
51      return -1;
52  }
53
54  // fungsi menampilkan elemen dalam linked list
55  void display(Node* node){
56      while (node != nullptr){
```

```
55  void display(Node* node){
56      while (node != nullptr){
57          cout << node->data << " ";
58          node = node->next;
59      }
60      cout << endl;
61  }
62
63  // fungsi menghapus elemen dari linked list
64  void deleteElement(Node*& head, int x){
65      if (head == nullptr){
66          cout << "Linked List Kosong" << endl;
67          return;
68      }
69
70      if (head->data == x){
71          Node* temp = head;
72          head = head->next;
73          delete temp;
74          return;
75      }
76
77      Node* current = head;
78      while (current->next != nullptr){
79          if (current->next->data == x){
80                  Node* temp = current->next;
81              current->next = current->next->next;
82              delete temp;
83              return;
```

```
            delete temp;
            return;
        }
        current = current->next;
    }
}

int main(){
    Node* head = nullptr;
    Node* tail = nullptr;

    insertFirst(head, tail, 3);
    insertFirst(head, tail, 5);
    insertFirst(head, tail, 7);

    insertLast(head, tail, 11);
    insertLast(head, tail, 14);
    insertLast(head, tail, 18);

    cout << "Elemen dalam linked list: ";
    display(head);

    int x;
    cout << "Masukkan elemen yang ingin dicari: ";
    cin >> x;

    int result = findElement(head, x);

    if (result == -1)
```

```
    int result = findElement(head, x);

    if (result == -1)
        cout << "Elemen tidak ditemukan dalam linked list" << endl;
    else
        cout << "Elemen ditemukan pada indeks " << result << endl;

    cout << "Masukkan elemen yang ingin dihapus: ";
    cin >> x;
    deleteElement(head, x);

    cout << "Elemen dalam linked list setelah penghapusan: ";
    display(head);

    return 0;
}
```

OUTPUT :

```
"C:\Users\USER\Documents\5 STD\guided\bin\Debug\cobba5.exe"
Elemen dalam linked list: 7 5 3 11 14 18
Masukkan elemen yang ingin dicari: 5
Elemen ditemukan pada indeks 1
Masukkan elemen yang ingin dihapus: 11
Elemen dalam linked list setelah penghapusan: 7 5 3 14 18

Process returned 0 (0x0)   execution time : 13.670 s
Press any key to continue.
```

- Node Struktur**: Setiap elemen dalam linked list disebut node, dan setiap node berisi data integer (`data`) dan pointer (`next`) yang menunjuk ke node berikutnya. Ini memungkinkan setiap node terhubung dalam rantai atau daftar berantai (linked list).
- insertFirst: Fungsi ini menambahkan node baru di awal linked list. Node baru akan menunjuk ke node yang sebelumnya berada di awal daftar, kemudian dijadikan sebagai `head` baru. Jika linked list sebelumnya kosong (tidak ada elemen), `tail` juga akan menunjuk ke node baru tersebut.

- insertLast: Fungsi ini menambahkan node baru di akhir linked list. Jika linked list kosong, node baru menjadi `head` dan `tail`. Jika tidak kosong, node baru ditambahkan setelah node `tail`, dan `tail` diperbarui ke node baru tersebut.
- findElement: Fungsi ini mencari elemen dalam linked list berdasarkan nilai data yang diberikan. Fungsi melakukan pencarian dari awal hingga menemukan elemen yang dicari dan mengembalikan indeksnya. Jika elemen tidak ditemukan, fungsi mengembalikan `-1`.
- display: Fungsi ini menampilkan semua elemen dalam linked list dari `head` sampai ke `tail`. Fungsi ini melakukan iterasi melalui setiap node dan mencetak nilai `data` pada setiap node hingga mencapai akhir.
- deleteElement: Fungsi ini menghapus elemen dalam linked list berdasarkan nilai yang dicari. Jika elemen yang akan dihapus berada di awal (pada `head`), `head` diperbarui ke node berikutnya. Jika elemen ada di tengah atau akhir, fungsi ini menghubungkan node sebelumnya ke node setelahnya, lalu menghapus node yang sesuai.
- main: Fungsi utama untuk menguji linked list. Dalam `main`, beberapa elemen ditambahkan ke awal dan akhir linked list menggunakan `insertFirst` dan `insertLast`. Lalu, seluruh elemen dalam linked list ditampilkan dengan `display`. Program kemudian meminta pengguna memasukkan elemen untuk dicari dengan `findElement` dan menghapus elemen yang diinginkan dengan `deleteElement`. Setelah penghapusan, linked list kembali ditampilkan untuk menunjukkan hasil akhirnya.

  Secara keseluruhan, program ini membentuk linked list, memungkinkan penambahan di awal dan akhir, pencarian elemen, penghapusan elemen tertentu, dan penampilan semua elemen dalam linked list.

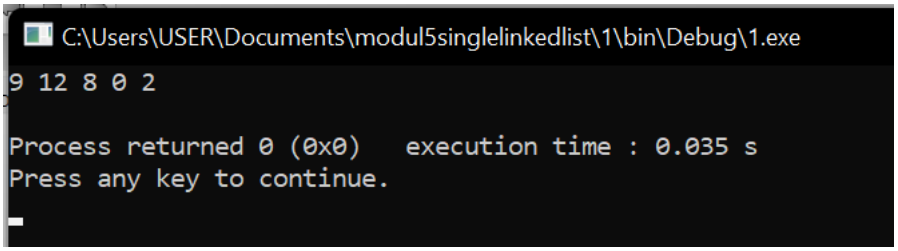
4. Unguided

1. CODE :

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

// Define the types
typedef int infotype;
struct ElmList; // Forward declaration
typedef ElmList* address;

// Element of the list
struct ElmList {
    infotype info;
    address next;
};

// List definition
struct List {
    address first;
};

// Initialize the list
void createList(List &L) {
    L.first = NULL;
}

// Allocate a new element
address alokasi(infotype x) {
    address p = new ElmList;
```

```
// Allocate a new element
address alokasi(infotype x) {
    address p = new ElmList;
    p->info = x;
    p->next = NULL;
    return p;
}

// Deallocate an element
void deAlokasi(address &p) {
    delete p;
    p = NULL;
}

// Print all elements in the list
void printInfo(const List &L) {
    address p = L.first;
    while (p != NULL) {
        cout << p->info << " ";
        p = p->next;
    }
    cout << endl;
}

// Insert an element at the beginning
void insertFirst(List &L, address p) {
    p->next = L.first;
    L.first = p;
}

int main() {
    List L;
    address P1, P2, P3, P4, P5;

    // Initialize pointers
    P1 = P2 = P3 = P4 = P5 = NULL;

    // Create the list
    createList(L);

    // Allocate nodes and insert them into the list
    P1 = alokasi(2);
    insertFirst(L, P1);

    P2 = alokasi(0);
    insertFirst(L, P2);

    P3 = alokasi(8);
    insertFirst(L, P3);

    P4 = alokasi(12);
    insertFirst(L, P4);

    // Allocate nodes and insert them into the list
    P1 = alokasi(2);
    insertFirst(L, P1);

    P2 = alokasi(0);
    insertFirst(L, P2);

    P3 = alokasi(8);
    insertFirst(L, P3);

    P4 = alokasi(12);
    insertFirst(L, P4);

    P5 = alokasi(9);
    insertFirst(L, P5);

    // Print the elements in the list
    printInfo(L);

    return 0;
}
```

OUTPUT :



- **Pendefinisian Struktur dan Tipe**: Program ini mendefinisikan tipe data infotype sebagai int, address sebagai pointer ke ElmList, dan List sebagai struktur yang menyimpan pointer first untuk menunjuk ke node pertama dari linked list.
- **Struktur ElmList (Node)**: Struktur ElmList merepresentasikan satu node dari linked list, di mana setiap node memiliki:

- info untuk menyimpan nilai data (integer).

- next untuk menunjuk ke node berikutnya dalam linked list.

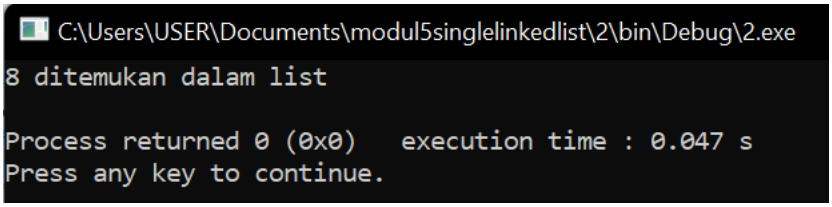
- **Inisialisasi Linked List**: Fungsi createList digunakan untuk menginisialisasi linked list sehingga kosong di awal. first pada linked list diatur ke NULL, yang menunjukkan bahwa list belum memiliki elemen.
- **Alokasi Node Baru**: Fungsi alokasi digunakan untuk membuat node baru dengan nilai data tertentu. Node yang baru dibuat diatur agar next-nya menunjuk ke NULL, menunjukkan bahwa node ini belum terhubung dengan node lain dalam linked list.
- **Dealokasi Node**: Fungsi deAlokasi menghapus atau mendeklarasikan kembali memori yang digunakan oleh node tertentu. Ini membantu menghindari kebocoran memori (walaupun fungsi ini tidak digunakan dalam program ini).
- **Menampilkan Elemen dalam Linked List**: Fungsi printInfo digunakan untuk mencetak semua elemen dalam linked list. Fungsi ini melakukan iterasi dari first hingga node terakhir, mencetak nilai info dari setiap node.
- **Menambahkan Node di Awal**: Fungsi insertFirst menambahkan node baru di awal linked list. Fungsi ini membuat node baru menunjuk ke node pertama yang ada, kemudian memperbarui first sehingga menunjuk ke node baru ini.
- **Fungsi main**: Fungsi main mengimplementasikan semua langkah-langkah yang dijelaskan di atas:
    - Membuat linked list kosong.
    - Menambahkan beberapa elemen ke awal linked list menggunakan insertFirst.
    - Menampilkan semua elemen yang ada dalam linked list dengan printInfo.
- **Alur Eksekusi**

Program ini membuat linked list kosong, kemudian menambahkan lima node di awal daftar dengan nilai 2, 0, 8, 12, dan 9 secara berurutan. Setelah penambahan selesai, daftar berisi elemen-elemen tersebut dalam urutan 9 12 8 0 2. Fungsi printInfo kemudian menampilkan isi linked list ke layar.

2. CODE :

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

typedef int infotype;
typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
};

struct List {
    address First;
};

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

address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = NULL;
    return P;
}

void insertFirst(List &L, address P) {
    P->next = L.First;
    L.First = P;
}

// Fungsi untuk mencari elemen dengan info tertentu
address findElm(List L, infotype x) {
    address P = L.First;
    while (P != NULL) {
        if (P->info == x) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}

int main() {
    List L;
    address P1, P2, P3, P4, P5 = NULL;

    createList(L);

    P1 = alokasi(2);
    insertFirst(L, P1);

    P2 = alokasi(0);
    insertFirst(L, P2);

    P3 = alokasi(8);
    insertFirst(L, P3);

    P4 = alokasi(12);
    insertFirst(L, P4);

    P5 = alokasi(9);
    insertFirst(L, P5);

    // Mencari elemen dengan info 8
    infotype searchValue = 8;
    address found = findElm(L, searchValue);
    if (found != NULL) {
        cout << searchValue << " ditemukan dalam list" << endl;
    } else {
        cout << searchValue << " tidak ditemukan dalam list" << endl;
    }

    return 0;
}
```

OUTPUT :



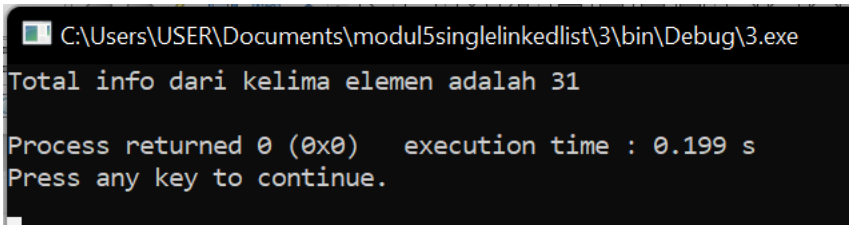
- **Pendefinisian Struktur dan Tipe**: Program ini mendefinisikan tipe data infotype sebagai int, address sebagai pointer ke ElmList, dan List sebagai struktur yang berisi pointer First yang menunjuk ke elemen pertama dari linked list.
- **Struktur ElmList (Node)**: Struktur ElmList merepresentasikan satu node dalam linked list, di mana setiap node memiliki:
    - info untuk menyimpan nilai data (integer).
    - next untuk menunjuk ke node berikutnya dalam linked list.
- **Inisialisasi Linked List**: Fungsi createList menginisialisasi linked list sehingga kosong pada awalnya dengan mengatur First ke NULL. Ini menunjukkan bahwa list belum memiliki elemen.
- **Alokasi Node Baru**: Fungsi alokasi digunakan untuk membuat node baru dengan nilai data tertentu. Node yang baru dibuat diatur agar next-nya menunjuk ke NULL, menunjukkan bahwa node ini belum terhubung dengan node lain dalam linked list.
- **Menambahkan Node di Awal**: Fungsi insertFirst menambahkan node baru di awal linked list. Node baru ini menjadi First, sementara next-nya menunjuk ke node yang sebelumnya ada di posisi pertama.
- **Mencari Elemen Berdasarkan Nilai Info**: Fungsi findElm melakukan pencarian elemen berdasarkan nilai info yang diberikan. Fungsi ini melakukan iterasi dari node First hingga node terakhir untuk menemukan node yang memiliki nilai info yang cocok. Jika elemen ditemukan, fungsi mengembalikan pointer ke node tersebut. Jika tidak ditemukan, fungsi mengembalikan NULL.
- **Fungsi main**: Di dalam main:
    - Sebuah linked list kosong dibuat dengan createList.
    - Lima elemen ditambahkan ke awal linked list menggunakan insertFirst dengan nilai 2, 0, 8, 12, dan 9, sehingga urutannya dari awal ke akhir adalah 9 12 8 0 2.
    - Kemudian, fungsi findElm digunakan untuk mencari elemen dengan nilai info tertentu, dalam hal ini 8. Jika ditemukan, program mencetak bahwa nilai tersebut ada dalam linked list; jika tidak ditemukan, akan dicetak bahwa nilai tersebut tidak ada.

3. Buatlah program yang dapat memberikan *input* dan *output* sbb:
CODE :

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

typedef int infotype;
typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
};

struct List {
    address First;
};

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

address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = NULL;
    return P;
}

void insertFirst(List &L, address P) {
    P->next = L.First;
    L.First = P;
}

// Fungsi untuk menghitung total info
int sumInfo(List L) {
    address P = L.First;
    int total = 0;
    while (P != NULL) {
        total += P->info;
        P = P->next;
    }
    return total;
}

int main() {
    List L;
    address P1, P2, P3, P4, P5 = NULL;

    createList(L);

    P1 = alokasi(2);
    insertFirst(L, P1);

    P2 = alokasi(0);
    insertFirst(L, P2);

```

```
50      insertFirst(L, P1);
51
52      P2 = alokasi(0);
53      insertFirst(L, P2);
54
55      P3 = alokasi(8);
56      insertFirst(L, P3);
57
58      P4 = alokasi(12);
59      insertFirst(L, P4);
60
61      P5 = alokasi(9);
62      insertFirst(L, P5);
63
64      // Menghitung total info
65      int total = sumInfo(L);
66      cout << "Total info dari kelima elemen adalah " << total << endl;
67
68      return 0;
69  }
70
```

OUTPUT :



- **Pendefinisian Struktur dan Tipe**: Program ini mendefinisikan tipe data infotype sebagai int, address sebagai pointer ke ElmList, dan List sebagai struktur yang berisi pointer First, yang menunjuk ke elemen pertama dari linked list.
- **Struktur ElmList (Node)**: Struktur ElmList merepresentasikan satu node dalam linked list, di mana setiap node memiliki:
- info untuk menyimpan nilai data (integer).
- next untuk menunjuk ke node berikutnya dalam linked list.
- **Inisialisasi Linked List**: Fungsi createList menginisialisasi linked list sehingga kosong pada awalnya dengan mengatur First ke NULL. Ini menunjukkan bahwa list belum memiliki elemen.
- **Alokasi Node Baru**: Fungsi alokasi digunakan untuk membuat node baru dengan nilai data tertentu. Node yang baru dibuat diatur agar next-nya menunjuk ke NULL, menunjukkan bahwa node ini belum terhubung dengan node lain dalam linked list.
- **Menambahkan Node di Awal**: Fungsi insertFirst menambahkan node baru di awal linked list. Node baru ini menjadi First, sementara next-nya menunjuk ke node yang sebelumnya ada di posisi pertama.
- **Menghitung Total Nilai info**: Fungsi sumInfo menghitung jumlah nilai info dari semua node dalam linked list. Fungsi ini melakukan iterasi dari node First hingga node terakhir, menambahkan setiap nilai info ke variabel total untuk mendapatkan jumlah keseluruhan.
- **Fungsi main**: Di dalam main:
    - Sebuah linked list kosong dibuat dengan createList.

- Lima elemen ditambahkan ke awal linked list menggunakan insertFirst dengan nilai 2, 0, 8, 12, dan 9, sehingga urutannya dari awal ke akhir adalah 9 12 8 0 2.
- Fungsi sumInfo kemudian dipanggil untuk menghitung total nilai info dari semua elemen dalam linked list. Hasil total ini disimpan dalam variabel total dan ditampilkan ke layar.

5. Kesimpulan

Program yang telah dibuat menunjukkan penerapan dasar dari single linked list dengan beberapa operasi utama, seperti menambahkan elemen di awal, mencari elemen berdasarkan nilai tertentu, serta menghitung total nilai elemen. Linked list menyediakan fleksibilitas dalam alokasi memori karena elemen-elemen dalam linked list tidak harus berada dalam lokasi memori yang berurutan dan dapat ditambahkan atau dihapus secara dinamis. Struktur linked list lebih efisien untuk operasi penambahan dan penghapusan elemen dibandingkan dengan array, terutama jika elemen ditambahkan atau dihapus dari posisi selain di akhir.

Secara keseluruhan, pemahaman dasar tentang linked list serta penggunaannya dalam program adalah penting dalam ilmu komputer, terutama dalam pengelolaan data yang fleksibel dan dinamis. Dengan menggunakan konsep pointer, program dapat membuat dan mengelola linked list yang ukurannya dapat berubah sesuai kebutuhan, serta melakukan berbagai operasi dasar yang diperlukan dalam pemrosesan data.