

LAPORAN PRAKTIKUM
Modul 5
Single Linked List (BagianKedua)



Disusun Oleh:
Yogi Hafidh Maulana - 2211104061
SE06-02

Dosen :
Wahyu Andi

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

- Memahami Konsep Dasar Linked List.
- Menggunakan Pointer dalam Program
- Melaksanakan Operasi Dasar pada Linked List.
- Mengimplementasikan ADT Linked List.
- Menerapkan Fungsi Pencarian dan Agregasi Data.

2. Landasan Teori

- Pengertian Linked List
Linked List adalah salah satu bentuk struktur data yang menyimpan serangkaian elemen yang saling terhubung satu sama lain melalui pointer. Setiap elemen dalam linked list disebut node, yang terdiri dari dua bagian: data (informasi yang disimpan dalam node) dan pointer (penunjuk ke node berikutnya). Berbeda dengan array, linked list bersifat dinamis, sehingga ukurannya dapat berubah sesuai kebutuhan (bertambah atau berkurang).
- Single Linked List
Single Linked List adalah salah satu varian dari linked list di mana setiap node hanya memiliki satu pointer, yaitu penunjuk ke node berikutnya (successor). Dalam single linked list, pengaksesan elemen hanya dapat dilakukan secara sekuensial (dari depan ke belakang), karena node hanya memiliki satu arah penunjuk.
- Keunggulan Linked List Dibandingkan Array
Linked list memiliki beberapa keunggulan dibandingkan array, di antaranya:
Ukuran Dinamis: Linked list dapat tumbuh dan mengecil sesuai kebutuhan, sedangkan array memiliki ukuran tetap.
Efisiensi dalam Penyisipan dan Penghapusan: Pada linked list, penambahan atau penghapusan elemen pada posisi awal atau akhir bisa dilakukan dengan efisien tanpa perlu menggeser elemen-elemen lain, seperti yang harus dilakukan pada array.
Namun, linked list juga memiliki kelemahan, seperti waktu akses yang lebih lambat dibandingkan array, karena linked list tidak menyediakan akses langsung ke elemen tertentu (harus dilakukan pencarian sekuensial).

3. Guided

a) Guided 1

Code:

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

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

// Fungsi untuk menambahkan elemen baru ke awal linked 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;

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

// Fungsi untuk 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;

    if (head == nullptr)
    {
        head = new_node;
        tail = new_node;
    }
    else
    {
        tail->next = new_node;
        tail = new_node;
    }
}
```

```
// Fungsi untuk mencari elemen dalam linked list
int findElement(Node *head, int x)
{
    Node *current = head;
    int index = 0;

    while (current != nullptr)
    {
        if (current->data == x)
        {
            return index;
        }
        current = current->next;
        index++;
    }
    return -1;
}

// Fungsi untuk menampilkan elemen dalam linked list
void display(Node *node)
{
    while (node != nullptr)
    {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}
```

```
// Fungsi untuk menghapus elemen dari linked list
void deleteElement(Node *&head, int x)
{
    if (head == nullptr)
    {
        cout << "Linked List kosong" << endl;
        return;
    }

    if (head->data == x)
    {
        Node *temp = head;
        head = head->next;
        delete temp;
        return;
    }

    Node *current = head;
    while (current->next != nullptr)
    {
        if (current->next->data == x)
        {
            Node *temp = current->next;
            current->next = current->next->next;
            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 << "Masukan elemen yang ingin dicari: ";
    cin >> x;

    int result = findElement(head, x);

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

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

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

    return 0;
}

```

Output:

```

-dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Elemen dalam linked list: 7 5 3 11 14 18
Masukan elemen yang ingin dicari: 3
Elemen ditemukan dalam index 2
Masukan elemen yang ingin dihapus: 14
Elemen dalam linked list setelah penghapusan: 7 5 3 11 18
PS D:\PROJECT\C++ Project\Laprak5>

```

Deskripsi Program:

insertFirst: Fungsi ini digunakan untuk menambahkan elemen baru ke awal linked list. Ia menerima dua argumen referensi, yaitu head dan tail, serta new_data yang merupakan nilai baru yang akan dimasukkan. Fungsi ini membuat node baru dengan data yang diberikan, menghubungkannya dengan node yang sudah ada, dan memperbarui head. Jika linked list sebelumnya kosong (tail adalah nullptr), tail juga diatur ke node baru.

insertLast: Fungsi ini berfungsi untuk menambahkan elemen baru ke akhir linked list. Seperti insertFirst, ia menerima dua argumen referensi (head dan tail) serta new_data. Jika linked list kosong, node baru yang dibuat menjadi head dan tail. Jika tidak, node baru dihubungkan ke akhir linked list, dan tail diperbarui untuk menunjuk ke node baru.

findElement: Fungsi ini digunakan untuk mencari elemen tertentu dalam linked list. Fungsi ini menerima argumen head (awal linked list) dan x (nilai yang dicari). Fungsi ini mengiterasi melalui linked list hingga menemukan elemen yang cocok, mengembalikan indeks elemen jika ditemukan atau -1 jika tidak ditemukan.

display: Fungsi ini digunakan untuk menampilkan semua elemen dalam linked list. Ia menerima pointer ke node (awal linked list) dan mencetak nilai setiap node hingga mencapai akhir linked list (nullptr).

deleteElement: Fungsi ini berfungsi untuk menghapus elemen tertentu dari linked list. Fungsi ini menerima head dan nilai x yang ingin dihapus. Jika elemen yang ingin dihapus adalah elemen pertama, head diperbarui. Jika elemen tersebut berada di tengah atau akhir, fungsi ini mengiterasi melalui linked list untuk menemukan elemen dan memperbarui pointer agar mengabaikan node yang dihapus.

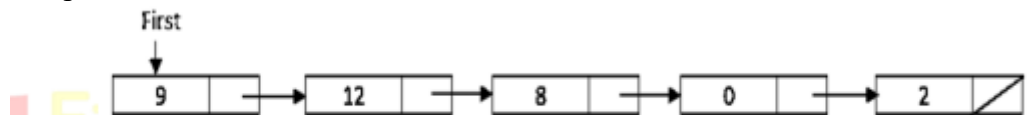
main: Fungsi ini adalah titik awal eksekusi program. Di sini, linked list diinisialisasi dengan head dan tail yang diset ke nullptr. Beberapa elemen ditambahkan menggunakan insertFirst dan insertLast, dan kemudian elemen dalam linked list ditampilkan. Program meminta pengguna untuk memasukkan elemen yang ingin dicari dan dihapus, menggunakan fungsi yang sesuai untuk mencari dan menghapus elemen tersebut, diakhiri dengan menampilkan linked list yang telah diperbarui.

4. Unguided

- a) Buatlah ADT Single Linked list sebagai berikut di dalam file “singlelist.h”:

```
Type infotype : int
Type address  : pointer to ElmList
Type ElmList <
    info : infotype
    next : address
>
Type List : < First : address >
prosedur CreateList( in/out L : List )
fungsi alokasi( x : infotype ) : address
prosedur dealokasi( in/out P : address )
prosedur printInfo( in L : List )
prosedur insertFirst( in/out L : List, in P : address )
```

Kemudian buat implementasi ADT Single Linked list pada file “singlelist.cpp”.
Adapun isi data



Gambar 5-1 Ilustrasi elemen

Cobalah hasil implementasi ADT pada file “main.cpp”

```
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);

    printInfo(L)
    return 0;
}
```


Contoh Output

```
9 12 8 0 2
Process returned 0 (0x0)   execution time : 0.019 s
Press any key to continue.
```


Code:

Singlelist.h

```


#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);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(List L);
void insertFirst(List &L, address P);

#endif
```

Singlelist.cpp

```
#include "singlelist.h"

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

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

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

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

void printInfo(List L)
{
    address P = L.First;
    while (P != NULL)
    {
        cout << P->info << " ";
        P = P->next;
    }
    cout << endl;
}
```

Main.cpp

```

#include "singlelist.h"

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);

    printInfo(L);

    return 0;
}
```

Output:

```
D:\PROJECT\C++ Project\Laprak5>program.exe
9 12 8 0 2
```

Deskripsi Code:

singlelist.h (File Header) File ini berfungsi sebagai deklarasi atau blueprint dari struktur data dan fungsi-fungsi yang akan diimplementasikan di file lain, dalam hal ini singlelist.cpp. Di dalam singlelist.h, kita mendefinisikan tipe-tipe data yang akan digunakan, seperti infotype (yang bertipe int), serta address (sebagai pointer yang menunjuk ke elemen di dalam linked list). Kita juga mendefinisikan struktur ElmList, yang merupakan elemen dalam linked list, yang berisi nilai info dan pointer next yang menunjuk ke elemen berikutnya. Selain itu, kita juga mendeklarasikan tipe List, yang hanya memiliki satu anggota, yaitu pointer First yang menunjuk ke elemen pertama di dalam list. Terakhir, file ini mendeklarasikan beberapa prosedur dan fungsi, seperti createList, alokasi, dealokasi, insertFirst, dan printInfo, yang akan diimplementasikan di file singlelist.cpp.

singlelist.cpp (Implementasi) File ini berisi implementasi dari fungsi-fungsi yang dideklarasikan dalam singlelist.h. Pertama, fungsi createList digunakan untuk menginisialisasi linked list agar kosong, di mana pointer First diset ke NULL. Fungsi alokasi bertugas untuk mengalokasikan memori baru untuk sebuah elemen list, mengisinya dengan nilai info, dan memastikan pointer next menunjuk ke NULL. Prosedur dealokasi digunakan untuk melepaskan memori yang dialokasikan sebelumnya, membantu menghindari kebocoran memori. Fungsi insertFirst memungkinkan penambahan elemen baru di awal linked list, dengan cara mengubah pointer next dari elemen baru agar menunjuk ke elemen pertama yang lama, lalu memperbarui pointer First agar menunjuk ke elemen baru tersebut. Akhirnya, prosedur printInfo digunakan untuk mencetak semua elemen yang ada di dalam linked list, mulai dari elemen pertama hingga akhir list, dengan memanfaatkan loop untuk menelusuri pointer next.

main.cpp (Main Program) File ini merupakan program utama yang menjalankan semua fungsi yang telah didefinisikan dan diimplementasikan dalam dua file sebelumnya. Di dalam main.cpp, pertama-tama kita mendeklarasikan sebuah list L dan beberapa alamat P1 hingga P5 yang bertugas untuk menyimpan alamat elemen-elemen linked list. Prosedur createList(L) dipanggil untuk menginisialisasi list kosong. Selanjutnya, kita menggunakan fungsi alokasi untuk menciptakan elemen-elemen dengan nilai tertentu (sesuai dengan gambar yang disediakan), dan memasukkannya ke dalam list menggunakan prosedur insertFirst, sehingga elemen-elemen tersebut disusun dari yang terakhir diinput hingga yang pertama. Terakhir, prosedur printInfo dipanggil untuk mencetak isi dari linked list, yang akan menghasilkan output berupa angka-angka yang disusun sesuai urutan elemen yang dimasukkan.

- b) Carilah elemen dengan info 8 dengan membuat fungsi baru. fungsi findElm(L : List, x : infotype) : address

```
8 ditemukan dalam list
Process returned 0 (0x0)   execution time : 0.020 s
Press any key to continue.
```

Code:

```
address findElm(const List &L, infotype x) {
    address P = L.First;
    while (P != nullptr) {
        if (P->info == x) {
            return P;
        }
        P = P->next;
    }

    return nullptr;
}
```

Output:

```
D:\PROJECT\C++ Project\Laprak5>program.exe
9 12 8 0 2
8 ditemukan dalam list
```

Deskripsi Code:

Untuk menjalankan program ini, Anda perlu membuat tiga file yaitu singlelist.h (file header), singlelist.cpp (file implementasi), dan main.cpp (program utama). Setelah menyiapkan kode di masing-masing file, Anda bisa mengompilasi menggunakan GCC. Pada sistem Linux atau Mac, Anda dapat menjalankannya dengan perintah `g++ main.cpp singlelist.cpp -o program` di terminal, lalu jalankan dengan `./program`. Jika menggunakan Windows dengan MinGW, Anda dapat menggunakan perintah `g++ main.cpp singlelist.cpp -o program.exe` dan menjalankannya dengan `program.exe`. Program ini akan menampilkan hasil yang sesuai dengan ilustrasi dan tangkapan layar, yakni mencetak urutan elemen dari daftar terhubung, dan menemukan elemen dengan nilai 8, lalu mencetak pesan bahwa 8 ditemukan dalam list.

- c) Hitunglah jumlah total info seluruh elemen ($9+12+8+0+2=31$).

Code:

```
int sumInfo(const List &L)
{
    address P = L.First;
    int sum = 0;
    while (P != nullptr)
    {
        sum += P->info;
        P = P->next;
    }
    return sum;
}
```

Output:

```
D:\PROJECT\C++ Project\Laprak5>program.exe
9 12 8 0 2
8 ditemukan dalam list
Total info dari kelima elemen adalah 31
```

Deskripsi Code:

Fungsi sumInfo berfungsi untuk menjelajahi setiap elemen dalam linked list, menjumlahkan nilai info dari setiap elemen, lalu mengembalikan jumlah totalnya. Di dalam program utama (main.cpp), setelah mencetak elemen-elemen dari linked list, fungsi ini dipanggil untuk menghitung total nilai dari semua elemen yang ada, kemudian hasilnya ditampilkan.

5. Kesimpulan

Setelah mempelajari modul ini, saya telah memahami konsep dasar Single Linked List, terutama dalam hal operasi-operasi seperti menambahkan elemen di awal list, mencari elemen berdasarkan nilai, serta menghitung total nilai dari semua elemen dalam list. Saya juga mempelajari bagaimana mengelola memori secara dinamis, termasuk cara mengalokasikan dan mendekalokasikan node dengan tepat. Modul ini membantu saya memahami pentingnya linked list dalam pemrograman dan bagaimana menerapkan fungsi-fungsi dasar yang relevan dalam aplikasi nyata. Implementasi dan hasil latihan memberikan pemahaman yang lebih dalam tentang operasi dasar linked list serta kemampuan saya untuk mengembangkan program berbasis linked list.
