

**LAPORAN PRAKTIKUM  
STRUKTUR DATA**

**MODUL XIII**

**MULTI LINKED LIST**



**Disusun Oleh :**  
Nama : Besthian Guido Rafael Simbolon  
NIM : 103112430258

**Dosen**  
FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2025**

## A. Dasar Teori

Linked List adalah struktur data dinamis yang terdiri dari kumpulan node yang saling terhubung menggunakan pointer, sehingga tidak memiliki ukuran tetap seperti array. Salah satu pengembangannya adalah Multi-Linked List, yaitu struktur data yang digunakan untuk menyimpan data dengan hubungan satu ke banyak (one-to-many). Dalam implementasinya, struktur ini biasanya menggunakan Doubly Linked List, di mana setiap node memiliki pointer next dan prev untuk memudahkan penelusuran dua arah. Keunikan Multi-Linked List terletak pada adanya pointer tambahan pada node induk (parent) yang menunjuk ke node pertama pada daftar anak (child), sehingga satu data parent dapat memiliki banyak data child. Struktur ini membentuk pola seperti pohon atau graf dan memungkinkan pengelolaan data secara fleksibel dan dinamis tanpa batasan ukuran seperti pada array statis.

## B. Guided

### Guided 1

#### Guided.cpp

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

struct ChildNode
{
    string info;
    ChildNode *next;
    ChildNode *prev;
};

struct ParentNode
{
    string info;
    ChildNode *childHead;
    ParentNode *next;
    ParentNode *prev;
};

ParentNode *createParent(string info)
{
    ParentNode *newNode = new ParentNode;
    newNode->info = info;
    newNode->childHead = NULL;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

ChildNode *createChild(string info)
{
    ChildNode *newNode = new ChildNode;
    newNode->info = info;
    newNode->next = NULL;
    newNode->prev = ....;
}
```

---

Screenshots Output

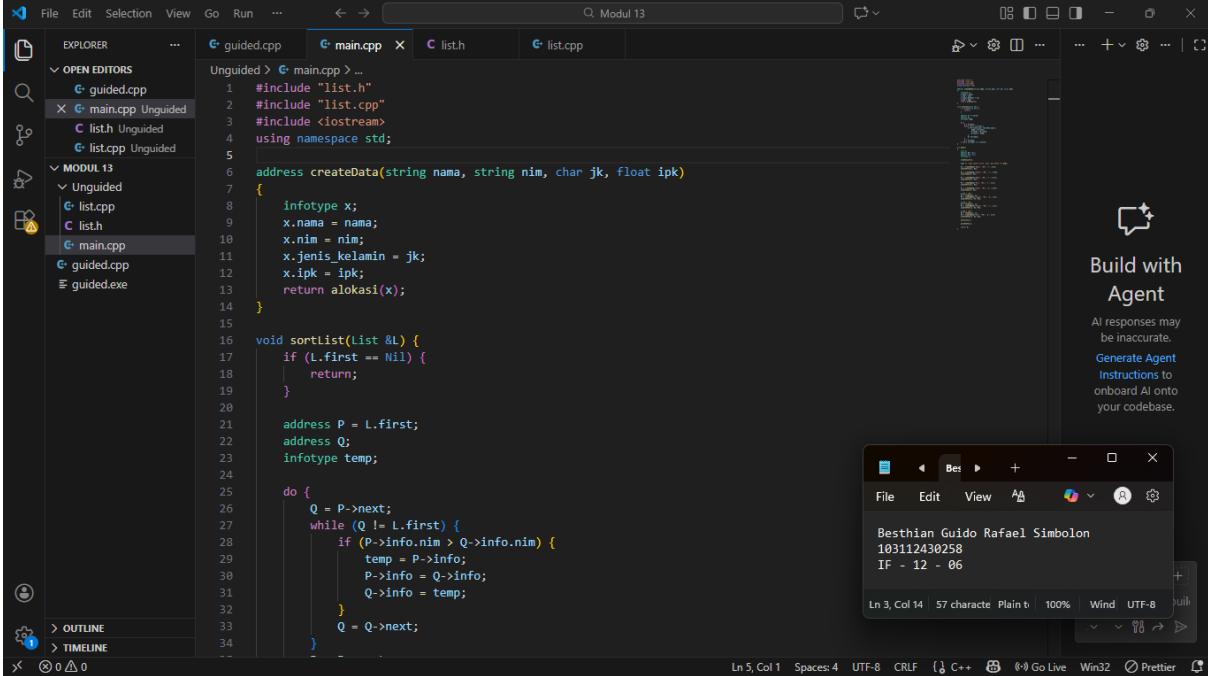
Deskripsi code :

Program ini merupakan implementasi sederhana struktur data Multi-Linked List (List of Lists) menggunakan bahasa C++, yang terdiri dari dua jenis node yaitu ParentNode sebagai induk dan ChildNode sebagai anak. Setiap node induk saling terhubung dalam bentuk doubly linked list, dan setiap node induk memiliki pointer khusus yang menunjuk ke daftar node anak yang juga tersusun sebagai doubly linked list. Program ini menyediakan operasi dasar CRUD (Create, Read, Update, Delete) yang memungkinkan pengguna menambah data parent dan child, menampilkan seluruh struktur data secara hierarkis, memperbarui data pada node tertentu, serta menghapus node parent atau child dengan tetap menjaga keterhubungan pointer agar struktur data tidak rusak. Pada fungsi main, program memperlihatkan alur penggunaan secara lengkap mulai dari penambahan data, pembaruan, hingga penghapusan untuk menunjukkan cara pengelolaan relasi antar node di dalam memori.

### C. Unguided

#### Unguided 1

##### Main.cpp



The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows project structure with files: guided.cpp, main.cpp, list.h, and list.cpp.
- Editor Area:** Displays the content of main.cpp. The code implements a doubly linked list where each parent node contains a list of child nodes. It includes functions for creating data, sorting the list, and printing the structure.
- Bottom Status Bar:** Shows file information like "Ln 5, Col 1" and "Spaces: 4".
- Right Sidebar:** Contains a "Build with Agent" section with AI-related instructions.

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

address createData(string nama, string nim, char jk, float ipk)
{
    infotype x;
    x.nama = nama;
    x.nim = nim;
    x.jenis_kelamin = jk;
    x.ipk = ipk;
    return alokasi(x);
}

void sortList(List &L) {
    if (L.first == Nil) {
        return;
    }

    address P = L.first;
    address Q;
    infotype temp;

    do {
        Q = P->next;
        while (Q != L.first) {
            if (P->info.nim > Q->info.nim) {
                temp = P->info;
                P->info = Q->info;
                Q->info = temp;
            }
            Q = Q->next;
        }
    } while (P != L.first);
}
```

## circularlist.cpp

A screenshot of the Visual Studio Code interface. The left sidebar shows an 'EXPLORER' view with files: guided.cpp, main.cpp, circularlist.h, and circularlist.cpp. The right pane displays the content of circularlist.cpp. The code defines a circular linked list structure and includes functions for creating the list, inserting nodes at the beginning, and dealocking memory. A status bar at the bottom shows file paths, line numbers, and other development details.

```
File Edit Selection View Go Run ... ← → 🔍 Modul 13
EXPLORER ... guided.cpp main.cpp circularlist.h circularlist.cpp
MODUL 13 Unguided circularlist.cpp
circularlist.h
main.cpp
main.exe
guided.cpp
guided.exe
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

File Edit View ⚡ - + 🌐 ⓘ
Bestian Guido Rafael Simbolon
103112430258
IF - 12 - 06
Ln 3, Col 14 57 character Plain text 100% Wind UTF-8
Ln 20, Col 39 Spaces: 4 UTF-8 CRLF ⓘ Go Live Win32 ⓘ Prettier
CircularList.h
1 #ifndef CIRCULARLIST_H_INCLUDED
2 #define CIRCULARLIST_H_INCLUDED
3
4 #include <string>
5 using namespace std;
6
7 #define Nil NULL
8
9 struct infotype {
10     string nama;
11     string nim;
12     char jenis_kelamin;
13     float ipk;
14 };
15
16 typedef struct ElmList *address;
17
18 struct ElmList {
19     infotype info;
20     address next;
21 };
22
23 struct List {
24     address first;
25 };
26
27 void createList(List &L);
28
29 address alokasi(infotype x);
30 void dealokasi(address &P);
31
32 void insertFirst(List &L, address P);
33 void insertAfter(List &L, address Prec, address P);
34 void insertLast(List &L, address P);
```

## circularlist.h

A screenshot of the Visual Studio Code interface. The left sidebar shows an 'EXPLORER' view with files: guided.cpp, main.cpp, circularlist.h, and circularlist.cpp. The right pane displays the content of circularlist.h. The code defines a circular linked list structure using a header guard, string for names, and a struct for information. It also includes definitions for NULL and ElmList pointers. A status bar at the bottom shows file paths, line numbers, and other development details.

```
File Edit Selection View Go Run ... ← → 🔍 Modul 13
EXPLORER ... guided.cpp main.cpp circularlist.h circularlist.cpp
MODUL 13 Unguided circularlist.h
circularlist.cpp
main.cpp
main.exe
guided.cpp
guided.exe
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

File Edit View ⚡ - + 🌐 ⓘ
Bestian Guido Rafael Simbolon
103112430258
IF - 12 - 06
Ln 3, Col 14 57 character Plain text 100% Wind UTF-8
Ln 44, Col 7 Spaces: 4 UTF-8 CRLF ⓘ Go Live Win32 ⓘ Prettier
CircularList.h
1 #ifndef CIRCULARLIST_H_INCLUDED
2 #define CIRCULARLIST_H_INCLUDED
3
4 #include <string>
5 using namespace std;
6
7 #define Nil NULL
8
9 struct infotype {
10     string nama;
11     string nim;
12     char jenis_kelamin;
13     float ipk;
14 };
15
16 typedef struct ElmList *address;
17
18 struct ElmList {
19     infotype info;
20     address next;
21 };
22
23 struct List {
24     address first;
25 };
26
27 void createList(List &L);
28
29 address alokasi(infotype x);
30 void dealokasi(address &P);
31
32 void insertFirst(List &L, address P);
33 void insertAfter(List &L, address Prec, address P);
34 void insertLast(List &L, address P);
```

---

## Screenshot Output

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows a tree view of files and folders. Opened editors include `guided.cpp`, `main.cpp` (Unguided), `circularlist.h`, and `circularlist.cpp`. A folder named `MODUL 13` contains `Unguided`, `circularlist.cpp`, `circularlist.h`, `main.cpp`, `main.exe`, `guided.cpp`, and `guided.exe`. Each file has associated student information: Nama, NIM, L/P, and IPK.
- TERMINAL:** Tab labeled "Modul 13".
- CODE EDITOR:** Tab labeled "Code". It displays the content of `main.cpp` which includes:

```
Besthian Guido Rafael Simbolon
103112430258
IF - 12 - 06 |
```

With status bar text: Ln 3, Col 14 | 57 character Plain t | 100% | Wind | UTF-8 | C++ | Go Live | Win32 | Prettier |
- RIGHT SIDE:** A sidebar titled "Build with Agent" with a note about AI responses being inaccurate and a link to "Generate Agent Instructions".

Deskripsi:

Program ini merupakan implementasi struktur data **Circular Single Linked List** menggunakan bahasa C++ yang digunakan untuk mengelola data mahasiswa, meliputi nama, NIM, jenis kelamin, dan IPK. Pada struktur ini, node terakhir tidak menunjuk ke NULL, melainkan kembali ke node pertama (first), sehingga membentuk sebuah lingkaran. Program disusun secara modular dengan pemisahan file header, file implementasi, dan file `main` agar kode lebih rapi dan mudah dipahami. Operasi dasar yang disediakan meliputi pembuatan node, penyisipan data di awal, akhir, atau setelah node tertentu, penghapusan, serta pencarian data. Pada fungsi `main`, program menampilkan contoh penggunaan dengan memasukkan beberapa data mahasiswa secara acak, kemudian mengurutkannya berdasarkan NIM menggunakan algoritma sederhana, dan akhirnya menampilkan seluruh data ke layar.

#### D. Kesimpulan

**Multi Linked List** yang telah dilakukan, dapat disimpulkan bahwa struktur data ini sangat efektif untuk mengelola data yang memiliki hubungan **satu ke banyak (one-to-many)**. Dengan adanya node induk (parent) yang terhubung ke beberapa node anak (child) melalui pointer, data dapat disusun secara hierarkis dan lebih terorganisir. Praktikum ini juga menunjukkan pentingnya pemahaman terhadap penggunaan pointer dan alokasi memori dinamis agar proses penambahan, penghapusan, dan pencarian data dapat berjalan dengan baik tanpa merusak struktur list. Secara keseluruhan, Multi Linked List memberikan fleksibilitas tinggi dalam pengelolaan data yang kompleks dan menjadi solusi yang tepat untuk kasus data bertingkat.

#### E. Referensi

- Kadir, A. (2012). *Algoritma dan struktur data*. Yogyakarta: Andi Offset.
- Kurniawan, D. (2016). *Struktur data menggunakan C++*. Jakarta: Elex Media Komputindo.
- Rosa, A. S., & Shalahuddin, M. (2018). *Rekayasa perangkat lunak terstruktur dan berorientasi objek*. Bandung: Informatika.
- Sutabri, T. (2012). *Konsep sistem informasi*. Yogyakarta: Andi Offset.
- Program Studi Informatika. (Tahun). *Modul praktikum struktur data: Linked list dan pengembangannya*. Indonesia: Perguruan Tinggi terkait.