

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL 13
PENGENALAN CODE BLOCKS**



Disusun Oleh :
NAMA : Muhamad Ilham Syahid
NIM : 103112400155

Dosen
WAHYU ANDI SAPUTRA

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

A. Dasar Teori

Multi Linked List merupakan struktur data yang terdiri dari beberapa linked list yang saling terhubung, di mana satu list berperan sebagai induk dan list lainnya sebagai anak. Setiap elemen induk dapat menunjuk ke sebuah list anak sehingga mampu merepresentasikan hubungan satu-ke-banyak. Struktur ini mendukung operasi penambahan, penghapusan, dan penelusuran data baik pada level induk maupun anak secara dinamis dan efisien.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided

1

multilist.cpp

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

/* ===== LIST INDUK ===== */
boolean ListEmpty(listinduk L){
    return (L.first == Nil);
}

void CreateList(listinduk &L){
    L.first = Nil;
    L.last = Nil;
}

address alokasi(infotypeinduk x){
    address P = new elemen_list_induk;
    P->info = x;

    // WAJIB
    CreateListAnak(P->lanak);

    P->next = Nil;
    P->prev = Nil;
    return P;
}

void dealokasi(address P){
    delete P;
}

address findElm(listinduk L, infotypeinduk X){
    address P = L.first;
    while(P != Nil){
```

```

        if(P->info == X) return P;
        P = P->next;
    }
    return Nil;
}

void insertLast(listinduk &L, address P){
    if(ListEmpty(L)){
        L.first = P;
        L.last = P;
    } else {
        L.last->next = P;
        P->prev = L.last;
        L.last = P;
    }
}

void delP(listinduk &L, infotypeinduk X){
    address P = findElm(L, X);
    if(P != Nil){
        if(P == L.first){
            L.first = P->next;
        } else if(P == L.last){
            L.last = P->prev;
        } else {
            P->prev->next = P->next;
            P->next->prev = P->prev;
        }
        dealokasi(P);
    }
}

/* ===== LIST ANAK ===== */
boolean ListEmptyAnak(listanak L){
    return (L.first == Nil);
}

void CreateListAnak(listanak &L){
    L.first = Nil;
    L.last = Nil;
}

address_anak alokasiAnak(infotypeanak x){
    address_anak P = new elemen_list_anak;
    P->info = x;
    P->next = Nil;
    P->prev = Nil;
    return P;
}

```

```
}
```

```
void dealokasiAnak(address_anak P){
    delete P;
}
```

```
void insertLastAnak(listanak &L, address_anak P){
    // PAKSA node baru steril
    P->next = Nil;
    P->prev = Nil;

    if(L.first == Nil){
        L.first = P;
        L.last = P;
    } else {
        L.last->next = P;
        P->prev = L.last;
        L.last = P;
    }
}
```

```
void delPAnak(listanak &L, infotypeanak X){
    address_anak P = L.first;
    while(P != Nil){
        if(P->info == X){
            if(P == L.first){
                L.first = P->next;
            } else if(P == L.last){
                L.last = P->prev;
            } else {
                P->prev->next = P->next;
                P->next->prev = P->prev;
            }
            dealokasiAnak(P);
            return;
        }
        P = P->next;
    }
}

void printInfo(listinduk L){
    address P = L.first;
    while(P != Nil){
        cout << "Induk: " << P->info << " -> Anak: ";
        printInfoAnak(P->lanak);
        cout << endl;
        P = P->next;
    }
}
```

```

    }

}

void printInfoAnak(listanak L){
    address_anak P = L.first;
    int pengaman = 0;

    while(P != Nil && pengaman < 20){
        cout << P->info << " ";
        P = P->next;
        pengaman++;
    }
    cout << endl;
}

```

Guided 2

Multilist.h

```

#ifndef MULTILIST_H_INCLUDED
#define MULTILIST_H_INCLUDED

#include <iostream>
using namespace std;

#define Nil NULL
typedef bool boolean;

/* ===== TYPE DATA ===== */
typedef int infotypeanak;
typedef int infotypeinduk;

typedef struct elemen_list_induk *address;
typedef struct elemen_list_anak *address_anak;

/* ===== LIST ANAK ===== */
struct elemen_list_anak{
    infotypeanak info;
    address_anak next;
    address_anak prev;
};

struct listanak{
    address_anak first;
    address_anak last;
};

```

```

/* ===== LIST INDUK ===== */
struct elemen_list_induk{
    infotypeinduk info;
    listanak lanak;
    address next;
    address prev;
};

struct listinduk{
    address first;
    address last;
};

/* ===== PROTOTYPE ===== */
boolean ListEmpty(listinduk L);
boolean ListEmptyAnak(listanak L);

void CreateList(listinduk &L);
void CreateListAnak(listanak &L);

address alokasi(infotypeinduk x);
address_anak alokasiAnak(infotypeanak x);

void dealokasi(address P);
void dealokasiAnak(address_anak P);

address findElm(listinduk L, infotypeinduk X);

void insertLast(listinduk &L, address P);
void insertLastAnak(listanak &L, address_anak P);

void delP(listinduk &L, infotypeinduk X);
void delPAnak(listanak &L, infotypeanak X);

void printInfo(listinduk L);
void printInfoAnak(listanak L);

#endif

```

Guided 3

Main.cpp

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

int main(){
    listinduk L;
    CreateList(L);

    address peg1 = alokasi(1);
    insertLast(L, peg1);

    insertLastAnak(peg1->lanak, alokasiAnak(10));
    insertLastAnak(peg1->lanak, alokasiAnak(20));

    printInfo(L);
    return 0;
}
```

Screenshots Output

```
PS C:\Pratikum Struktur Data\modul 13 laprak> g++ main.cpp multilist.cpp -o main
>>
PS C:\Pratikum Struktur Data\modul 13 laprak> ./main
Induk: 1 -> Anak: 10 20
PS C:\Pratikum Struktur Data\modul 13 laprak> ./main
Induk: 1 -> Anak: 10 20
```

Deskripsi:

Modul 13 membahas implementasi Multi Linked List untuk merepresentasikan hubungan satu-ke-banyak antara data induk dan data anak, serta operasi dasar seperti insert, delete, dan traversal pada struktur tersebut.

C. Kesimpulan

Multi Linked List memudahkan pengelolaan data yang saling berelasi secara hierarkis dan dinamis, serta melatih ketelitian dalam penggunaan pointer pada struktur data linked list.

D. Referensi

Modul 13

Circularlist.h

```
#ifndef CIRCULARLIST_H_INCLUDED
#define CIRCULARLIST_H_INCLUDED
#include <string>
using namespace std;

#define Nil NULL

struct mahasiswa{
    string nama;
    string nim;
    char jenis_kelamin;
    float ipk;
};

typedef mahasiswa 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 insertFirst(List &L, address P);
void insertAfter(List &L, address Prec, address P);
void insertLast(List &L, address P);

void deleteFirst(List &L, address &P);
void deleteAfter(List &L, address Prec, address &P);
void deleteLast(List &L, address &P);

address findElm(List L, infotype x);
void printInfo(List L);

#endif
```

Circularlist.cpp

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

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

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

void dealokasi(address P){
    delete P;
}

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

void insertLast(List &L, address P){
    if(L.First == Nil){
        insertFirst(L,P);
    } else {
        address last = L.First;
        while(last->next != L.First)
            last = last->next;
        last->next = P;
        P->next = L.First;
    }
}

void insertAfter(List &L, address Prec, address P){
    P->next = Prec->next;
    Prec->next = P;
```

```

}

address findElm(List L, infotype x){
    if(L.First == Nil) return Nil;
    address P = L.First;
    do{
        if(P->info.nim == x.nim)
            return P;
        P = P->next;
    } while(P != L.First);
    return Nil;
}

void printInfo(List L){
    if(L.First == Nil) return;
    address P = L.First;
    do{
        cout << P->info.nama << " | "
            << P->info.nim << " | "
            << P->info.jenis_kelamin << " | "
            << P->info.ipk << endl;
        P = P->next;
    } while(P != L.First);
}

```

Main.cpp

```

#include "circularlist.h"
#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);
}

int main(){
    List L;
    createList(L);

    insertFirst(L, createData("Danu","04",'l',4.0));
    insertLast(L, createData("Fahmi","06",'l',3.45));
    insertFirst(L, createData("Bobi","02",'l',3.71));
    insertFirst(L, createData("Ali","01",'l',3.3));
}

```

```
insertLast(L, createData("Gita", "07", 'p', 3.75));

infotype x;
x.nim = "07";
insertAfter(L, findElm(L, x), createData("Cindi", "03", 'p', 3.5));

printInfo(L);
return 0;
}
```

Screenshot

```
PS C:\Pratikum Struktur Data\modul 13 laprak\circularlist> ./main
Ali | 01 | 1 | 3.3
Bobi | 02 | 1 | 3.71
Danu | 04 | 1 | 4
Fahmi | 06 | 1 | 3.45
Gita | 07 | p | 3.75
Cindi | 03 | p | 3.5
PS C:\Pratikum Struktur Data\modul 13 laprak\circularlist> []
```