LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL XIII (13)
MULTI LINKED LIST

Disusun Oleh :

NAMA : Gien Darrel Adli NIM : 10312430008

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Algoritma merupakan fondasi dalam pembuatan program komputer. Secara sederhana, algoritma adalah serangkaian langkah logis dan sistematis yang disusun untuk menyelesaikan suatu masalah. C++, berfungsi sebagai alat untuk mengimplementasikan algoritma tersebut agar dapat dimengerti dan dieksekusi oleh komputer. C++ sering digunakan sebagai bahasa pengantar untuk mempelajari konsep pemrograman dasar karena strukturnya yang terorganisir dan kemampuannya untuk menangani operasi tingkat rendah.

Untuk membangun logika dalam program sesuai dengan algoritma yang dirancang, C++ menyediakan struktur kontrol, yang terbagi menjadi dua jenis utama:

Struktur Percabangan (Conditional): Digunakan untuk pengambilan keputusan, di mana program akan menjalankan blok kode tertentu jika suatu kondisi terpenuhi. Struktur ini mencakup if-else untuk mengevaluasi kondisi boolean dan switch-case untuk memilih blok kode berdasarkan nilai dari sebuah variabel.

Struktur Perulangan (Looping): Digunakan untuk mengeksekusi blok kode yang sama secara berulang kali selama kondisi tertentu masih terpenuhi. C++ menyediakan tiga jenis perulangan utama: for, while, dan do-while, yang masing-masing memiliki karakteristik penggunaan yang spesifik dalam implementasi algoritma.

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

Guided 1 (multilinkedlist.h)

```
#ifndef MULTILINKEDLIST_H
#define MULTILINKEDLIST_H

#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);
ChildNode *createChild(string info);

void insertParent(ParentNode *&head, string info);
void insertChild(ParentNode *head, string parentInfo, string childInfo);

void updateParent(ParentNode *head, string oldInfo, string newInfo);
void updateChild(ParentNode *head, string parentInfo,
                 string oldChildInfo, string newChildInfo);

void deleteParent(ParentNode *&head, string info);
void deleteChild(ParentNode *head, string parentInfo, string childInfo);

void printAll(ParentNode *head);

#endif
```

Deskripsi:

File multilinkedlist.h adalah header yang mendefinisikan struktur multilinked list dengan node Parent dan Child. Parent dapat memiliki banyak child, masing-masing terhubung dengan pointer. File ini berisi deklarasi struktur data dan fungsi untuk membuat, menambah, mengubah, menghapus, serta menampilkan data parent dan child.

Guided 2 (multilinkedlist.cpp)

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

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

ChildNode *createChild(string info) {
    ChildNode *c = new ChildNode;
    c->info = info;
    c->next = NULL;
    c->prev = NULL;
    return c;
}

void insertParent(ParentNode *&head, string info) {
    ParentNode *newNode = createParent(info);
    if (head == NULL) {
        head = newNode;
    } else {
        ParentNode *temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
    }
}

void insertChild(ParentNode *head, string parentInfo, string childInfo) {
    ParentNode *p = head;
    while (p != NULL && p->info != parentInfo)
        p = p->next;

    if (p != NULL) {
        ChildNode *newChild = createChild(childInfo);
        if (p->childHead == NULL) {
            p->childHead = newChild;
```

```
        } else {
            ChildNode *c = p->childHead;
            while (c->next != NULL)
                c = c->next;
            c->next = newChild;
            newChild->prev = c;
        }
    }
}

void updateParent(ParentNode *head, string oldInfo, string newInfo) {
    while (head != NULL) {
        if (head->info == oldInfo) {
            head->info = newInfo;
            return;
        }
        head = head->next;
    }
}

void updateChild(ParentNode *head, string parentInfo,
                 string oldChildInfo, string newChildInfo) {
    while (head != NULL && head->info != parentInfo)
        head = head->next;

    if (head != NULL) {
        ChildNode *c = head->childHead;
        while (c != NULL) {
            if (c->info == oldChildInfo) {
                c->info = newChildInfo;
                return;
            }
            c = c->next;
        }
    }
}

void deleteChild(ParentNode *head, string parentInfo, string childInfo) {
    while (head != NULL && head->info != parentInfo)
        head = head->next;

    if (head != NULL) {
        ChildNode *c = head->childHead;
        while (c != NULL) {
```

```
            if (c->info == childInfo) {
                if (c == head->childHead) {
                    head->childHead = c->next;
                    if (head->childHead != NULL)
                        head->childHead->prev = NULL;
                } else {
                    c->prev->next = c->next;
                    if (c->next != NULL)
                        c->next->prev = c->prev;
                }
                delete c;
                return;
            }
            c = c->next;
        }
    }
}

void deleteParent(ParentNode *&head, string info) {
    ParentNode *p = head;
    while (p != NULL) {
        if (p->info == info) {
            ChildNode *c = p->childHead;
            while (c != NULL) {
                ChildNode *temp = c;
                c = c->next;
                delete temp;
            }

            if (p == head) {
                head = p->next;
                if (head != NULL)
                    head->prev = NULL;
            } else {
                p->prev->next = p->next;
                if (p->next != NULL)
                    p->next->prev = p->prev;
            }
            delete p;
            return;
        }
        p = p->next;
    }
}
```

```
void printAll(ParentNode *head) {
   while (head != NULL) {
      cout << head->info;
      ChildNode *c = head->childHead;
      while (c != NULL) {
         cout << " -> " << c->info;
         c = c->next;
      }
      cout << endl;
      head = head->next;
   }
}
```

Deskripsi:

File **multilinkedlist.cpp** berisi implementasi fungsi-fungsi multilinked list. File ini mengatur pembuatan node parent dan child, proses penambahan, pengubahan, dan penghapusan data parent maupun child, serta menampilkan seluruh isi struktur multilinked list beserta relasinya.

Guided 3 (main.cpp)

```
#include <iostream>
#include "multilinkedlist.h"
#include "multilinkedlist.cpp"

using namespace std;

int main() {
   ParentNode *list = NULL;

   insertParent(list, "Parent A");
   insertParent(list, "Parent B");
   insertParent(list, "Parent C");

   cout << "\nSetelah InsertParent:\n";
   printAll(list);
```

```
    insertChild(list, "Parent A", "Child A1");
    insertChild(list, "Parent A", "Child A2");
    insertChild(list, "Parent B", "Child B1");

    cout << "\nSetelah InsertChild:\n";
    printAll(list);

    updateParent(list, "Parent B", "Parent B*");
    updateChild(list, "Parent A", "Child A1", "Child A1*");

    cout << "\nSetelah Update:\n";
    printAll(list);

    deleteChild(list, "Parent A", "Child A2");
    deleteParent(list, "Parent C");

    cout << "\nSetelah Delete:\n";
    printAll(list);

    return 0;
}
```

Deskripsi:

File **main.cpp** merupakan program utama yang menguji multilinked list dengan membuat data parent dan child, lalu menampilkan hasil setelah proses penambahan, pembaruan, dan penghapusan data.

Screenshots Output :

```
Setelah InsertParent:
Parent A
Parent B
Parent C

Setelah InsertChild:
Parent A -> Child A1 -> Child A2
Parent B -> Child B1
Parent C

Setelah Update:
Parent A -> Child A1* -> Child A2
Parent B* -> Child B1
Parent C

Setelah Delete:
Parent A -> Child A1*
Parent B* -> Child B1
PS D:\darrel\kuliah\Semester3\Struktur Data\laprak>
```

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

circularlist.h

```
#ifndef CIRCULARLIST_H_INCLUDED

#define CIRCULARLIST_H_INCLUDED



#include <iostream>

#include <string>

using namespace std;



#define Nil NULL



struct infotype

{

    string nama;
```

```
    string nim;
    char jenis_kelamin;
    float ipk;
};

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
```

Deskripsi:

File **circularlist.h** adalah file header yang mendefinisikan struktur data **circular singly linked list** untuk menyimpan data mahasiswa serta deklarasi fungsi untuk membuat list, menambah, mencari, menghapus, dan menampilkan data dalam list tersebut.

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;
  P = Nil;
}

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

```
    while (last->next != L.first)
    {
      last = last->next;
    }
    P->next = L.first;
    L.first = P;
    last->next = L.first;
  }
}

void insertLast(List &L, address P)
{
  if (L.first == Nil)
  {
    L.first = P;
    P->next = L.first;
  }
  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)
{
  if (Prec != Nil)
  {
    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)
  {
    cout << "List kosong" << endl;
    return;
  }
  address P = L.first;
  do
  {
    cout << "Nama: " << P->info.nama
         << ", NIM: " << P->info.nim
         << ", Jenis Kelamin: " << P->info.jenis_kelamin
         << ", IPK: " << P->info.ipk << endl;
    P = P->next;
  } while (P != L.first);
}
```

Deskripsi:

File **circularlist.cpp** berisi implementasi fungsi circular singly linked list, meliputi pembuatan list, alokasi dan dealokasi node, penyisipan data di awal, tengah, dan akhir, pencarian data, serta penampilan isi list secara melingkar.

main.cpp

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


address createData(string nama, string nim, char jenis_kelamin, float ipk)
{
    infotype x;
    x.nama = nama;
```

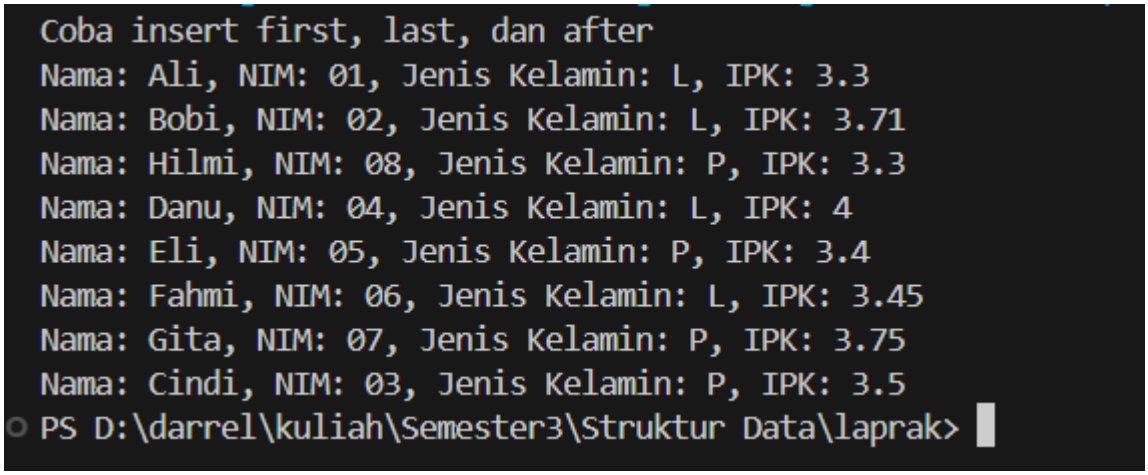
```
    x.nim = nim;
    x.jenis_kelamin = jenis_kelamin;
    x.ipk = ipk;
    return alokasi(x);
}
int main()
{
    List L;
    address P1, P2;
    infotype x;

    createList(L);

    cout << "Coba insert first, last, dan after" << endl;
    P1 = createData("Danu", "04", 'L', 4.0);
    insertFirst(L, P1);

    P1 = createData("Fahmi", "06", 'L', 3.45);
    insertLast(L, P1);

    P1 = createData("Bobi", "02", 'L', 3.71);
    insertFirst(L, P1);

    P1 = createData("Ali", "01", 'L', 3.3);
    insertFirst(L, P1);

    P1 = createData("Gita", "07", 'P', 3.75);
    insertLast(L, P1);

    x.nim = "07";
    P1 = findElm(L, x);
    P2 = createData("Cindi", "03", 'P', 3.5);
```

```
    insertAfter(L, P1, P2);

    x.nim = "02";
    P1 = findElm(L, x);
    P2 = createData("Hilmi", "08", 'P', 3.3);
    insertAfter(L, P1, P2);

    x.nim = "04";
    P1 = findElm(L, x);
    P2 = createData("Eli", "05", 'P', 3.4);
    insertAfter(L, P1, P2);

    printInfo(L);

    return 0;
}
```

Deskripsi:

Program main.cpp ini berfungsi untuk mencoba dan memastikan bahwa operasi pada circular linked list berjalan dengan benar. Program dimulai dengan membuat list kosong, lalu menambahkan beberapa data mahasiswa ke dalam list menggunakan operasi insert di awal, di akhir, dan setelah data tertentu. Untuk penyisipan di tengah, program terlebih dahulu mencari data berdasarkan NIM. Setelah semua data dimasukkan, isi list ditampilkan ke layar untuk melihat apakah urutan dan keterkaitan antar node sudah sesuai.

Screenshots Output Unguided:



D. Kesimpulan

Kesimpulan **modul 13** ini adalah mempelajari dan mengimplementasikan struktur data **Multi Linked List** menggunakan bahasa C++. Pada modul ini dibuat file header untuk mendefinisikan struktur node induk (parent) dan node anak (child) beserta relasinya, serta deklarasi fungsi-fungsi yang digunakan. File implementasi dibuat untuk merealisasikan operasi pada multi linked list seperti pembuatan list, alokasi node parent dan child, penyisipan data parent dan child, pencarian data, penghapusan data, serta penelusuran dan penampilan seluruh hubungan antara parent dan child. Seluruh fungsi tersebut kemudian diuji melalui program utama sehingga konsep kerja multi linked list, khususnya hubungan satu parent dengan banyak child dan pengelolaan struktur data yang saling terhubung, dapat dipahami dan berjalan sesuai dengan teori.

E. Referensi

Kaswar, A. B., & Zain, S. G. (2021). Mudah Belajar Pemrograman Dasar C++. Syiah Kuala University Press.

Hanief, S., Jepriana, I. W., & Kom, S. (2020). Konsep Algoritme dan Aplikasinya dalam Bahasa Pemrograman C++. Penerbit Andi.

Imamuddin, A., & Sobarnas, M. A. (2021). PEMBELAJARAN JARAK JAUH PEMROGRAMAN DASAR MENGGUNAKAN BAHASA C++ UNTUK UMUM: SEBUAH PROGRAM PENGABDIAN KEPADA MASYARAKAT. BEMAS: Jurnal Bermasyarakat, 1(2), 59-67.