Laporan Final Projek Struktur Data

Program Manajemen Pemesanan dan Pengiriman Barang

Dibuat untuk memenuhi tugas akhir mata kuliah Struktur Data

Dosen Pengampu:

I Made Widiartha, S.Si., M.Kom

Disusun oleh:

Kelompok II

I Gede Widnyana (2208561016)

I Gede Widiantara Mega Saputra (2208561022)

Kelas: C

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

JIMBARAN

2023

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas berkat rahmat-Nya penulis dapat menyelesaikan makalah ini dengan tepat waktu. Laporan ini dibuat guna untuk memenuhi tugas akhir mata kuliah Struktur Data. Tujuan dari penulisan laporan ini adalah sebagai bentuk pertanggung jawaban tertulis dari program *final project* yang penulis susun.

Penulis mengucapkan terima kasih kepada Bapak I Made Widiartha, S.Si., M.Kom, selaku dosen pengampu mata kuliah struktur data yang telah membimbing penulis dalam menyelesaikan pembuatan program *final project*. Penulis menyadari bahwa dalam penulisan laporan ini masih terdapat kekurangan. Oleh karena itu, penulis sangat terbuka untuk menerima kritik dan saran dari para pembaca laporan ini. Penulis berharap laporan ini dapat memberikan wawasan kepada para pembaca.

Jimbaran, 2 Juli 2023

Penulis

DAFTAR ISI

KATA PENGANTAR i

DAFTAR ISI ii

BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Rumusan masalah 2

1.3 Tujuan 2

BAB II PEMBAHASAN 3

2.1 Flowchart 3

2.2 Alur program 3

BAB III KESIMPULAN 48

LAMPIRAN 49

BAB I

PENDAHULUAN

1.1 Latar Belakang

Persaingan pasar industri di Indonesia berkembang pesat mengikuti arus globalisasi yang berdampak terhadap kebiasaan masyarakat Indonesia yang semakin konsumtif. Hal itu memengaruhi perekonomian negara yang ditandai dengan banyaknya toko, pasar, minimarket, dan *online shop* yang muncul akibat tuntutan globalisasi serta kebutuhan masyarakat terhadap barang-barang semakin meningkat. Pelaku ekonomi yang dalam konteks ini adalah pedagang mengambil peran andil dalam memenuhi kebutuhan masyarakat melalui jasa dan barang yang dijual kepada konsumen melalui sebuah usaha dalam siklus ekonomi masyarakat. Usaha yang paling umum dilakukan adalah sebagai penjual barang eceran maupun grosir.

Perkembangan teknologi memberikan kemudahan bagi konsumen dan pedagang dalam transaksi jual beli maupun pengiriman barang. Pada suatu kondisi konsumen sering memesan barang melalui internet ataupun melalui media sosial dari sebuah toko kemudian penjual menggunakan jasa pengiriman barang untuk mengirim pesanan konsumen. Pada proses pemesanan barang melalui internet, pedagang tentunya akan memerlukan sebuah aplikasi yang membantu dalam mencatat dan memanajemen pesanan barang, status barang, dan sebagainya. Selain itu, pihak jasa pengiriman juga perlu mengetahui rute terpendek yang harus dilalui untuk mengirimkan pesanan konsumen agar lebih efisien waktu dan tenaga.

Melihat permasalahan tersebut, penulis menawarkan solusi berupa sebuah program yang membantu penjual barang eceran ataupun grosir dalam memanajemen pesanan konsumen, mengatur stok barang, mengawasi keluar masuknya barang, dan mengatur proses pengiriman barang. Solusi ini diharapkan akan membantu penjual dalam meningkatkan kelancaran dalam siklus jual beli barang, meningkatkan kepuasan konsumen, serta mengefisienkan waktu pengiriman barang sehingga berdampak pada meningkatnya keuntungan penjual.

1.2 Rumusan masalah

Berdasarkan latar belakang tersebut, dapat dirumuskan permasalahan antara lain sebagai berikut.

1. Bagaimana rancangan program atau flowchart yang merepresentasikan program yang dibuat?
2. Bagaimana alur program dari aplikasi manajemen pemesanan dan pengiriman barang?

1.3 Tujuan

Adapun tujuan dari penulisan laporan ini adalah.

1. Untuk menyusun rancangan program atau flowchart yang merepresentasikan program yang dibuat
2. Untuk menganalisa program dari aplikasi manajemen pemesanan dan pengiriman barang.

BAB II
PEMBAHASAN

2.1 Flowchart

Sebelum penulis membuat aplikasi, penulis merancang sebuah flowchart dengan tujuan menggambarkan alur sebuah program dari setiap prosesnya. Fungsi lainnya, yaitu memampatkan atau menyederhanakan penjelasan dari proses yang ada. Dengan begitu, alir program akan lebih mudah untuk dipahami oleh semua orang.

Berikut penulis lampirkan flowchart yang telah penulis rancang untuk mendeskripsikan program manajemen pemesanan dan pengiriman barang. Flowchart disajikan dalam bentuk link google drive berikut. https://drive.google.com/file/d/1cQ9hstSXhnzJHgnQux-xOeix2JJNBBDK/view?usp=drive_link.

Penjelasan flowchart penulis sajikan dalam bentuk video di link google drive berikut:https://drive.google.com/file/d/1bGfFoPzYE6SGKkqlRn36OcamJo5yqMre/view?usp=drive_link.

2.2 Alur program

Berikut ini adalah penjelasan dari program yang penulis susun. Program ini dikerjakan menggunakan aplikasi text editor visual studio code, dengan bahasa pemrograman bahasa C, dan menerapkan konsep single linked list dan konsep graph (Algoritma Djikstra) pada program.

```c
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
#include <stdbool.h>
#include <limits.h>
#define INFINITY 9999
#define SIZE 100
#define V 10
#include<conio.h>

struct Barang{
    int kodebarang;
    char namabarang[20];
```

```
    int stock;
    struct Barang *next;
}; typedef struct Barang Brg;

struct Pemesanan{
    struct Pemesanan *next;
    int kodepesanan;
    char nama[20];
    int kodebarangpesanan;
    char namabarang[20];
    int jumlahpesanan;
    int kota_tujuan;
    int kota_asal;
    int tanggal;
    int bulan;
    int tahun;
    struct Barang Brg;
}; typedef struct Pemesanan Pesan;

struct Login{
    struct Login *next;
    char username[50];
    char password[50];
}akun[50]; typedef struct Login Lg;
```

1. Pertama-tama, beberapa header file termasuk `<stdio.h>`, `<stdlib.h>`, `<string.h>`, `<stdbool.h>`, dan `<limits.h>` diimpor. Header file ini menyediakan deklarasi fungsi dan tipe data yang akan digunakan dalam program.
2. Selanjutnya, beberapa konstanta didefinisikan menggunakan `#define`. Konstanta-konstanta tersebut adalah:
    - `INFINITY` dengan nilai 9999
    - `SIZE` dengan nilai 100
    - `V` dengan nilai 10

    Konstanta-konstanta ini akan digunakan dalam program untuk membatasi ukuran array dan operasi lainnya.
3. Setelah itu, sebuah struct `Barang` dideklarasikan. Struct ini memiliki empat anggota:
    - `kodebarang` (integer) untuk menyimpan kode barang
    - `namabarang` (array karakter) untuk menyimpan nama barang

- `stock` (integer) untuk menyimpan jumlah stok barang
- `next` (pointer ke struct Barang) untuk membentuk linked list

  Struct `Barang` ini digunakan untuk merepresentasikan informasi mengenai barang.

4. Selanjutnya, sebuah struct `Pemesanan` dideklarasikan. Struct ini memiliki sepuluh anggota:

- `next` (pointer ke struct Pemesanan) untuk membentuk linked list
- `kodepesanan` (integer) untuk menyimpan kode pesanan
- `nama` (array karakter) untuk menyimpan nama pelanggan
- `kodebarangpesanan` (integer) untuk menyimpan kode barang yang dipesan
- `namabarang` (array karakter) untuk menyimpan nama barang yang dipesan
- `jumlahpesanan` (integer) untuk menyimpan jumlah barang yang dipesan
- `kota_tujuan` (integer) untuk menyimpan kode kota tujuan pengiriman
- kota_asal` (integer) untuk menyimpan kode kota asal pengiriman
- `tanggal`, `bulan`, dan `tahun` (integer) untuk menyimpan tanggal pemesanan

Struct `Pemesanan` ini digunakan untuk merepresentasikan informasi mengenai pesanan yang dilakukan oleh pelanggan.

5. Setelah itu, sebuah struct `Login` dideklarasikan. Struct ini memiliki empat belas anggota:

- `next` (pointer ke struct Login) untuk membentuk linked list
- `username` (array karakter) untuk menyimpan username akun pengguna
- `password` (array karakter) untuk menyimpan password akun pengguna
- `pesan` (struct Pemesanan) untuk menyimpan informasi pesanan yang dilakukan oleh pengguna

  Struct `Login` ini digunakan untuk menyimpan informasi akun pengguna, termasuk username, password, dan detail pesanan.

```
void lihatbarang();
void tambahbarang();
void updatebarang();
```

```
void hapusbarang();
void detailbarang();
void tambahpemesanan();
void lihatpemesanan();
void dijkstra(int G[V][V],int n, int startnode, int targetnode);
void ubahgraph();
void buatakun();

Lg* loginakun(Lg *);
void lihatakun(Lg *);
struct Barang *head = NULL;
struct Barang *p = NULL;
struct Pemesanan *head1 = NULL;
struct Pemesanan *p1=NULL;

int graph[V][V] =  {{0, 15, 33, 0, 0, 0, 0, 0, 0},
                    { 15, 0, 0, 11, 0, 0, 0, 0, 0},
                    { 33, 0, 0, 0, 33, 0, 0, 0, 0},
                    { 0, 11, 0, 0, 0, 0, 0, 0, 74},
                    { 0, 0, 33, 0, 0, 39, 0, 0, 0},
                    { 0, 0, 0, 0, 39, 0, 43, 0, 0},
                    { 0, 0, 0, 0, 0, 43, 0, 96, 0},
                    { 0, 0, 0, 0, 0, 0, 96, 0, 67},
                    { 0, 0, 0, 74, 0, 0, 0, 67, 0}};

int main(){
    int pilih, fitur;
    Lg *lg;
    lg = NULL;

    system("cls"); system("color 06");
    printf("\n\t=====================================================
\n");
    printf("\t=          FINAL PROJECT STRUKTUR
DATA             =\n");
    printf("\t=APLIKASI MANAJEMEN PEMESANAN DAN PENGIRIMAN
BARANG=\n");
    printf("\t=====================================================\n
");
    printf("\t=      I GEDE WIDNYANA                [2208561016]
=\n");
    printf("\t=      I GEDE WIDIANTARA MEGA SAPUTRA [2208561022]
=\n");
    printf("\t=====================================================\n
");
    printf("\tTekan sembarang untuk memulai program!");
    getch();
```

```
    do{
        menu2:
        system("cls");
        system("color 06");
        printf("\t================================================
================ \n");
        printf("\t=      APLIKASI MANAJEMEN PEMESANAN DAN PENGIRIMAN
BARANG         =\n");
        printf("\t================================================
================ \n");
        printf("\n\tSilakan login terlebih dahulu! \n");
        printf("\n\tMenu:");
        printf("\n\t1. Login");
        printf("\n\t2. Daftar Akun");
        printf("\n\t0. Keluar");
        printf("\n\tPilih menu: ");
        scanf("%d", &fitur);

        switch (fitur) {
        case 1:
            lg = loginakun(lg);
            if(lg == NULL){
                break;
                goto menu2;
            }
            else{
                break;
                goto menu;
            }

    case 2:
        buatakun();
        printf("\n\tAnda telah berhasil mendaftar akun!");
        printf("\n\tSilakan login dengan akun yang baru dibuat.");
        getch();
        system("cls");
        lg = loginakun(lg);
        if(lg == NULL){

            goto menu2;
        }
        else{
            break;
        }
    case 0:
        exit(1);
```

```
    default:
        printf("\n\tMaaf, pilihan tidak valid. Silakan coba lagi.");
        getch();
        goto menu2;

    }
    }while(fitur!=0 && lg==NULL);

    do{
        menu:
        system("cls"); system("color 06");
        printf("\n\t==============================================
====\n");
        printf("\t=                    Menu
Fitur                    =\n");
        printf("\t==============================================
==\n");
        printf("\t1. Informasi Barang\n");
        printf("\t2. Informasi Pemesanan\n");
        printf("\t3. Informasi Akun & Graph\n");
        printf("\t4. Keluar\n");
        printf("\t==============================================
==\n");
        printf("\tMasukkan Pilihan Anda : ");
        scanf("%d", &fitur);

        if(fitur==1){
            system("cls"); system("color 06");
            printf("\n\t==============================================
========\n");
            printf("\t=                    Menu
Fitur                    =\n");
            printf("\t==============================================
======\n");
            printf("\t1. Tambah Barang\n");
            printf("\t2. Ubah Barang\n");
            printf("\t3. Lihat Barang\n");
            printf("\t4. Hapus Barang\n");
            printf("\t5. Lihat Detail Barang\n");
            printf("\t==============================================
======\n");
            printf("\tMasukkan Pilihan Anda : ");
            scanf("%d", &pilih);
            switch(pilih){
                case 1 :
                    tambahbarang();
                    break; goto menu;
```

```
            case 2 :
                updatebarang();
                break; goto menu;
            case 3 :
                lihatbarang();
                break; goto menu;
            case 4 :
                hapusbarang();
                break; goto menu;
            case 5:
                detailbarang();
                break; goto menu;
        }
    }
```

4. Fungsi `loginakun`:
   - Fungsi ini digunakan untuk melakukan proses login pengguna.
   - Menerima argumen berupa pointer ke struct `Lg` yang akan menyimpan informasi login pengguna.
   - Fungsi ini akan menampilkan pesan untuk memasukkan username dan password.
   - Kemudian, fungsi akan membaca file `data.txt` untuk memeriksa apakah kombinasi username dan password yang dimasukkan oleh pengguna terdaftar.
   - Jika terdapat kombinasi yang cocok, maka fungsi akan mengembalikan pointer ke struct `Lg` yang berisi informasi login pengguna.
   - Jika tidak ditemukan kombinasi yang cocok, fungsi akan mengembalikan nilai `NULL`.
5. Fungsi `buatakun`:
   - Fungsi ini digunakan untuk membuat akun baru.
   - Fungsi ini akan meminta pengguna memasukkan username dan password baru.
   - Selanjutnya, fungsi akan menyimpan informasi akun baru tersebut ke dalam file `data.txt`.
   - Setelah itu, fungsi akan menampilkan pesan bahwa akun telah berhasil dibuat dan meminta pengguna untuk login dengan akun yang baru dibuat.
6. Variabel `head` dan `p`:

- Variabel `head` dan `p` bertipe `struct Barang` dan digunakan untuk menyimpan data barang.
- `head` merupakan pointer ke node pertama dalam linked list barang.
- `p` digunakan sebagai pointer untuk mengakses dan memanipulasi node-node barang dalam linked list.

7. Variabel `head1` dan `p1`:
   - Variabel `head1` dan `p1` bertipe `struct Pemesanan` dan digunakan untuk menyimpan data pemesanan.
   - `head1` merupakan pointer ke node pertama dalam linked list pemesanan.
   - `p1` digunakan sebagai pointer untuk mengakses dan memanipulasi node-node pemesanan dalam linked list.
8. Array `graph`:
   - Array `graph` merupakan matriks yang digunakan untuk merepresentasikan graf yang menyimpan jarak antara lokasi-lokasi pengiriman barang.
   - Setiap elemen matriks menunjukkan jarak antara dua lokasi pengiriman.
   - Nilai 0 menunjukkan bahwa tidak ada koneksi langsung antara dua lokasi tersebut.
9. Fungsi `main`:
   - Fungsi utama dari program.
   - Pertama, akan ditampilkan header aplikasi dan user diminta untuk login atau membuat akun baru.
   - Jika pengguna berhasil login atau membuat akun baru, pengguna dapat mengakses menu fitur aplikasi seperti informasi barang, informasi pemesanan, informasi akun & graph, atau keluar dari aplikasi.
   - Di dalam menu fitur, pengguna dapat memilih untuk melakukan berbagai operasi seperti menambah, mengubah, melihat, atau menghapus barang, atau melihat detail barang.
   - Setelah pengguna selesai menggunakan fitur, pengguna akan kembali ke menu fitur hingga memilih untuk keluar dari aplikasi.

Penjelasan menu informasi barang sebagai berikut.

a. Jika pengguna memilih fitur "Informasi Barang", maka akan ditampilkan sub-menu fitur barang dengan menggunakan `printf()`. Pengguna akan diminta untuk memasukkan nomor pilihan.
b. Jika pengguna memilih pilihan 1, yaitu "Tambah Barang", maka fungsi `tambahbarang()` akan dipanggil. Fungsi ini digunakan untuk menambahkan barang baru ke dalam sistem. Pengguna akan diminta untuk memasukkan kode barang, nama barang, dan jumlah stok barang. Data barang baru akan disimpan dalam linked list yang telah didefinisikan sebelumnya.
c. Jika pengguna memilih pilihan 2, yaitu "Ubah Barang", maka fungsi `updatebarang()` akan dipanggil. Fungsi ini digunakan untuk mengubah informasi barang yang sudah ada dalam sistem. Pengguna akan diminta untuk memasukkan kode barang yang ingin diubah. Jika kode barang tersebut ditemukan dalam linked list, pengguna akan diminta untuk memasukkan informasi baru seperti nama barang dan jumlah stok barang.
d. Jika pengguna memilih pilihan 3, yaitu "Lihat Barang", maka fungsi `lihatbarang()` akan dipanggil. Fungsi ini digunakan untuk menampilkan informasi semua barang yang tersedia dalam sistem. Data barang akan ditampilkan dengan menggunakan perulangan untuk mengakses linked list barang.
e. Jika pengguna memilih pilihan 4, yaitu "Hapus Barang", maka fungsi `hapusbarang()` akan dipanggil. Fungsi ini digunakan untuk menghapus barang dari sistem berdasarkan kode barang yang dimasukkan oleh pengguna. Jika kode barang tersebut ditemukan dalam linked list, barang akan dihapus dari sistem.
f. Jika pengguna memilih pilihan 5, yaitu "Lihat Detail Barang", maka fungsi `detailbarang()` akan dipanggil. Fungsi ini digunakan untuk menampilkan detail lengkap dari suatu barang berdasarkan kode barang yang dimasukkan oleh pengguna. Jika kode barang tersebut ditemukan dalam linked list, informasi lengkap tentang barang tersebut akan ditampilkan.

Setelah menjalankan salah satu fitur dari sub-menu barang, program akan kembali ke menu utama dengan menggunakan `goto menu`. Hal ini memungkinkan pengguna untuk memilih fitur lain atau keluar dari program.

```
       else if(fitur==2){
            system("cls"); system("color 06");
            printf("\n\t=============================================
========\n");
            printf("\t=                    Menu
Fitur                    =\n");
            printf("\t=============================================
======\n");
            printf("\t1.  Pemesanan Barang\n");
            printf("\t2.  Lihat Proses Pengiriman\n");
            printf("\t=============================================
======\n");
            printf("\tMasukkan Pilihan Anda : ");
            scanf("%d", &pilih);
            switch(pilih){
                case 1 :
                    tambahpemesanan();
                    break; goto menu;
                case 2 :
                    lihatpemesanan(graph);
                    break; goto menu;
            }
        }
        else if(fitur==3){
            system("cls"); system("color 06");
            printf("\n\t=============================================
========\n");
            printf("\t=                    Menu
Fitur                    =\n");
            printf("\t=============================================
======\n");
            printf("\t1.  Lihat Akun Saya\n ");
            printf("\t2.  Ubah Data Graph\n");
            printf("\t=============================================
======\n");
            printf("\tMasukkan Pilihan Anda : ");
            scanf("%d", &pilih);
            switch(pilih){
                case 1 :
                    lihatakun(lg);
                    break; goto menu;
                case 2 :
                    ubahgraph();
                    break; goto menu;
            }
```

```
        }
        else if(fitur==4){
            system("cls");
            printf("\n\tTerimakasih Telah Menggunakan Program
ini.");
            getch(); exit(0); break;
        }
        else{
            printf("\n\tMaaf Pilihan Fitur Tidak Tersedia, Silahkan
Menginput Ulang!");
            getch(); break; goto menu;
        }
    }

while(pilih != 10);
return 0;
akhir:
    exit(0);
}
struct Login* loginakun(struct Login* lg) {
    struct Login* log = (struct Login*)malloc(sizeof(struct Login));
    log->next = NULL;

    system("cls");
    system("color 06");

    printf("\n\t=====================================================
\n");
    printf("\t=                    LOGIN
AKUN                    =\n");
    printf("\t=====================================================\n
");

    fflush(stdin);

    printf("\tUSERNAME    : ");
    scanf("%s", log->username);

    printf("\tPASSWORD    : ");
    scanf("%s", log->password);

    bool validCredentials = false;

    FILE* file = fopen("data.txt", "r");

    if (file == NULL) {
        printf("\n\tDatabase file error!\n");
```

```
        getch();
        free(log);
        return NULL;
    }

    char buffer[100];

    while (fgets(buffer, sizeof(buffer), file) != NULL) {
    char username[50], password[50];
    sscanf(buffer, "%s %s", username, password);

    if (strcmp(log->username, username) == 0 && strcmp(log-
>password, password) == 0) {
        validCredentials = true;
        break;
    }
}
    fclose(file);

    if (!validCredentials) {
        printf("\n\tUsername atau password Anda salah. Silakan login
kembali.\n");
        getch();
        free(log);
        return NULL;
    }

    if (lg == NULL) {
        lg = log;
    } else {
        log->next = lg;
        lg = log;
    }

    printf("\n");
    return lg;
}

void buatakun() {
    struct Login* newUser = (struct Login*)malloc(sizeof(struct
Login));

    printf("\nMasukkan Username: ");
    scanf("%s", newUser->username);

    printf("\nMasukkan Password: ");
    scanf("%s", newUser->password);
```

```
    newUser->next = NULL;

    FILE* file = fopen("data.txt", "a");

    if (file == NULL) {
        printf("\n\tDatabase file error!\n");
        getch();
        free(newUser);
        return;
    }

    fprintf(file, "%s %s\n", newUser->username, newUser->password);

    fclose(file);
}
```

10. Bagian `else if(fitur==2)`:
    - Kode ini akan dieksekusi jika nilai variabel `fitur` sama dengan 2.
    - Pertama, layar akan dibersihkan menggunakan `system("cls")` dan warna tampilan akan diubah menjadi biru menggunakan `system("color 06")`.
    - Selanjutnya, menu fitur terkait pemesanan barang dan melihat proses pengiriman akan ditampilkan menggunakan `printf`.
    - Pengguna akan diminta untuk memasukkan pilihan fitur menggunakan `scanf`, dan nilainya akan disimpan dalam variabel `pilih`.
    - Nilai `pilih` kemudian akan diproses dalam `switch` case.
    - Jika pengguna memilih fitur 1, fungsi `tambahpemesanan()` akan dipanggil untuk menambahkan pemesanan barang.
    - Setelah itu, program akan melanjutkan ke label `menu` dengan menggunakan `goto`.
    - Jika pengguna memilih fitur 2, fungsi `lihatpemesanan(graph)` akan dipanggil untuk melihat proses pengiriman dengan menggunakan argumen `graph`.
    - Setelah itu, program akan melanjutkan ke label `menu` dengan menggunakan `goto`.
11. Bagian `else if(fitur==3)`:

- Kode ini akan dieksekusi jika nilai variabel `fitur` sama dengan 3.
- Pertama, layar akan dibersihkan menggunakan `system("cls")` dan warna tampilan akan diubah menjadi biru menggunakan `system("color 06")`.
- Selanjutnya, menu fitur terkait melihat akun dan mengubah data graph akan ditampilkan menggunakan `printf`.
- Pengguna akan diminta untuk memasukkan pilihan fitur menggunakan `scanf`, dan nilainya akan disimpan dalam variabel `pilih`.
- Nilai `pilih` kemudian akan diproses dalam `switch` case.
- Jika pengguna memilih fitur 1, fungsi `lihatakun(lg)` akan dipanggil untuk melihat akun dengan menggunakan argumen `lg`.
- Setelah itu, program akan melanjutkan ke label `menu` dengan menggunakan `goto`.
- Jika pengguna memilih fitur 2, fungsi `ubahgraph()` akan dipanggil untuk mengubah data graph.
- Setelah itu, program akan melanjutkan ke label `menu` dengan menggunakan `goto`.

12. Bagian `else if(fitur==4)`:
    - Kode ini akan dieksekusi jika nilai variabel `fitur` sama dengan 4.
    - Pertama, layar akan dibersihkan menggunakan `system("cls")`.
    - Pesan "Terimakasih Telah Menggunakan Program ini." akan ditampilkan menggunakan `printf`.
    - Program akan menunggu pengguna menekan tombol apa pun menggunakan `getch()`.
    - Setelah itu, program akan keluar menggunakan `exit(0)`.
13. Bagian `else`:
    - Kode ini akan dieksekusi jika nilai variabel `fitur` tidak sama dengan 1, 2, 3, atau 4.
    - Pesan "Maaf Pilihan Fitur Tidak Tersedia, Silahkan Menginput Ulang!" akan ditampilkan menggunakan `printf`.
    - Program akan menunggu pengguna menekan tombol apa pun menggunakan `getch()`.

- Selanjutnya, program akan melanjutkan ke label `menu` dengan menggunakan `goto`.

14. Fungsi `loginakun(struct Login* lg)`:
    - Fungsi ini bertujuan untuk memvalidasi login pengguna dengan memeriksa apakah username dan password yang dimasukkan sesuai dengan data yang disimpan dalam file "data.txt".
    - Fungsi menerima argumen `lg` yang merupakan pointer ke linked list `struct Login`.
    - Pertama, layar akan dibersihkan dan pesan login akan ditampilkan menggunakan `printf`.
    - Pengguna akan diminta untuk memasukkan username dan password menggunakan `scanf`.
    - Variabel `validCredentials` akan digunakan untuk menyimpan status validasi kredensial.
    - File "data.txt" akan dibuka dan dibaca baris per baris.
    - Setiap baris akan dibaca menggunakan `fgets` dan kemudian dibandingkan dengan username dan password yang dimasukkan oleh pengguna.
    - Jika ditemukan kecocokan, `validCredentials` akan diubah menjadi `true` dan loop akan berhenti.
    - Setelah itu, file akan ditutup.
    - Jika kredensial valid, node baru akan dibuat untuk menyimpan informasi login pengguna.
    - Jika linked list kosong, node baru akan menjadi node pertama.
    - Jika linked list sudah berisi node, node baru akan ditambahkan di awal linked list.
    - Fungsi akan mengembalikan pointer ke linked list yang telah diperbarui.
15. Fungsi `buatakun()`:
    - Fungsi ini bertanggung jawab untuk membuat akun baru dengan meminta pengguna memasukkan username dan password baru.
    - Pertama, sebuah node baru akan dibuat untuk menyimpan username dan password baru.

- Pengguna akan diminta memasukkan username dan password menggunakan `scanf`.
- File "data.txt" akan dibuka dalam mode append menggunakan `fopen`.
- Jika file tidak dapat dibuka, pesan error akan ditampilkan.
- Jika file berhasil dibuka, username dan password baru akan ditambahkan ke dalam file menggunakan `fprintf`.
- Setelah itu, file akan ditutup.

16. Struktur Data Linked List:
    - Struktur data linked list digunakan untuk menyimpan informasi login pengguna. Setiap node dalam linked list memiliki dua elemen: `username` dan `password`.
    - Struktur data tersebut didefinisikan sebagai `struct Login`, yang terdiri dari `char username[50]`, `char password[50]`, dan `struct Login* next`.
    - `struct Login* next` merupakan pointer yang menunjuk ke node berikutnya dalam linked list. Jika node merupakan node terakhir, nilai pointer ini akan menjadi `NULL`.
17. Fungsi `loginakun(struct Login* lg)`:
    - Fungsi ini menerima argumen `lg` yang merupakan pointer ke linked list `struct Login`.
    - Jika `lg` memiliki nilai `NULL`, artinya linked list kosong, sehingga node baru yang diperoleh akan menjadi node pertama.
    - Jika `lg` tidak `NULL`, node baru akan ditambahkan di awal linked list dengan mengatur `next` pada node baru untuk menunjuk ke `lg` yang sudah ada, dan kemudian mengubah pointer `lg` untuk menunjuk ke node baru.
    - Dengan menggunakan pendekatan ini, node baru yang ditambahkan akan menjadi node pertama dalam linked list.
18. Fungsi `buatakun()`:
    - Fungsi ini bertanggung jawab untuk membuat akun baru dengan meminta pengguna memasukkan username dan password baru.

- Pertama, sebuah node baru akan dibuat dengan alokasi memori menggunakan `malloc(sizeof(struct Login))`. Ini akan mengalokasikan memori untuk node baru dalam ukuran yang sesuai dengan `struct Login`.
- Pengguna akan diminta memasukkan username dan password menggunakan `scanf`, dan nilai-nilai ini akan disimpan dalam `newUser->username` dan `newUser->password`.
- Setelah itu, node baru akan dihubungkan ke linked list dengan mengatur `next` pada node baru untuk menunjuk ke `NULL`.
- File "data.txt" akan dibuka dalam mode append menggunakan `fopen`.
- Jika file tidak dapat dibuka, pesan error akan ditampilkan.
- Jika file berhasil dibuka, username dan password baru akan ditambahkan ke dalam file menggunakan `fprintf`.
- Terakhir, file akan ditutup.

  Dengan menggunakan linked list, setiap kali pengguna melakukan login atau membuat akun baru, node baru akan ditambahkan di awal linked list. Ini memungkinkan kita untuk menyimpan informasi login pengguna dalam urutan terbalik, yaitu node terbaru berada di awal linked list. Ketika memeriksa akun pengguna atau melakukan operasi lain yang melibatkan informasi login, kita dapat mengunjungi setiap node dalam linked list dari awal hingga akhir.

Penjelasan rinci terhadap bagian berikut.

```
struct Login* loginakun(struct Login* lg) {
    struct Login* log = (struct Login*)malloc(sizeof(struct Login));
    log->next = NULL;
```

Pada bagian kode di atas, terdapat fungsi `loginakun` yang digunakan untuk melakukan proses login akun pengguna. Berikut adalah penjelasan lebih detail tentang bagian kode tersebut:

1. `struct Login* log = (struct Login*)malloc(sizeof(struct Login));`:

- Di sini, sebuah pointer `log` untuk node login baru dialokasikan menggunakan fungsi `malloc`.
- `sizeof(struct Login)` digunakan untuk mendapatkan ukuran memori yang tepat yang dibutuhkan untuk menyimpan satu node dalam linked list.

2. `log->next = NULL;`:

- Setelah melakukan alokasi memori untuk node baru, pointer `next` pada node tersebut diatur menjadi `NULL`.
- Hal ini menandakan bahwa node tersebut akan menjadi node terakhir dalam linked list.

3. Mengatur Return Value:

- Fungsi ini menggunakan pointer `lg` sebagai argumen dan akan mengembalikan pointer yang menunjuk ke node login baru.
- Jika linked list masih kosong, maka pointer `lg` akan menjadi `NULL` dan node login baru akan menjadi node pertama dalam linked list.
- Jika linked list sudah berisi node-node sebelumnya, node login baru akan ditambahkan di awal linked list dan pointer `lg` akan diubah untuk menunjuk ke node baru.

```
void lihatakun(Lg *lg){

    system("cls");
    printf("\n\t=========================================================\
n");
    printf("\t=                    Lihat Akun
Saya                  =\n");
    printf("\t=========================================================="
);

    while(lg != NULL){
        printf("\n\tUsername    : %s",lg->username);
        printf("\n\tPassword    : ******");
        lg= lg->next;
    } getch(); system("cls");
}

void masukpemesanan(int kodpes,int kodbarpes, int jumpes,int tgl,int
bln,int thn,char nampes[],char nambar[],int tujuan,int asal){
    struct Pemesanan* newnode = (struct
Pemesanan*)malloc(sizeof(struct Pemesanan));
    newnode->kodepesanan = kodpes;
    strcpy(newnode->nama, nampes);
    newnode->kodebarangpesanan = kodbarpes;
    strcpy(newnode->namabarang, nambar);
```

```
    newnode->jumlahpesanan = jumpes;
    newnode->tanggal = tgl;
    newnode->bulan = bln;
    newnode->tahun = thn;
    newnode->kota_asal= asal;
    newnode->kota_tujuan = tujuan;
    newnode->next = head1;
    head1 = newnode;
}
```

1. Fungsi `lihatakun()`:
    - Fungsi `lihatakun()` bertanggung jawab untuk menampilkan informasi akun yang tersimpan dalam linked list akun.
    - Pertama, layar dikosongkan menggunakan `system("cls")` agar tampilan menjadi bersih.
    - Kemudian, judul "Lihat Akun Saya" ditampilkan di layar menggunakan `printf()`.
    - Selanjutnya, menggunakan loop `while`, kita iterasi melalui linked list akun (`lg`) untuk mencetak informasi setiap akun.
    - Di setiap iterasi, `printf()` digunakan untuk mencetak username dan password dari akun saat ini (`lg`).
    - **Pernyataan lg = lg->next digunakan dalam loop while di fungsi lihatakun(). Fungsinya adalah memindahkan pointer lg ke elemen berikutnya dalam linked list akun.** Dalam loop **while**, setiap kali iterasi dilakukan, **lg** diubah untuk menunjuk ke elemen berikutnya dalam linked list dengan menggunakan pernyataan **lg = lg->next**. Ini adalah langkah yang penting untuk memastikan bahwa kita melanjutkan ke elemen berikutnya dalam linked list sehingga kita dapat mencetak informasi dari semua akun yang ada. Secara lebih rinci, ketika loop **while** pertama kali dimulai, **lg** menunjuk ke elemen pertama dalam linked list akun. Setelah mencetak informasi akun pertama, **lg = lg->next** memindahkan **lg** ke elemen kedua dalam linked list. Hal ini berlanjut sampai **lg** mencapai **NULL**, yang menandakan bahwa kita telah mencapai akhir dari linked list akun. Dengan menggunakan **lg = lg->next**, kita dapat melintasi seluruh linked list akun

dan mencetak informasi dari setiap akun tanpa kehilangan data atau melupakan elemen-elemen tertentu dalam linked list.

- Untuk setiap elemen akun, username ditampilkan menggunakan printf() dan password ditampilkan sebagai "******" untuk menjaga keamanan password.
- Setelah mencetak informasi akun, program akan menunggu input pengguna menggunakan `getch()` agar pengguna dapat membaca informasi dengan nyaman.
- Setelah pengguna memberikan input, layar dikosongkan lagi dengan `system("cls")`.

2. Fungsi `masukpemesanan()`:
   - Fungsi `masukpemesanan()` digunakan untuk menambahkan pemesanan baru ke linked list pemesanan.
   - Pada awalnya, sebuah node baru untuk pemesanan (`newnode`) dialokasikan menggunakan `malloc(sizeof(struct Pemesanan))`.
   - Variabel `newnode` bertipe `struct Pemesanan*` dan berfungsi sebagai pointer ke node baru yang akan ditambahkan ke linked list.
   - Nilai-nilai pemesanan seperti `kodepesanan`, `nama`, `kodebarangpesanan`, `namabarang`, `jumlahpesanan`, `tanggal`, `bulan`, `tahun`, `kota_asal`, dan `kota_tujuan` diatur sesuai dengan argumen yang diberikan kepada fungsi.
   - Perhatikan bahwa kita menggunakan fungsi `strcpy()` untuk menyalin string `nampes` dan `nambar` ke `newnode->nama` dan `newnode->namabarang` secara berturut-turut.
   - Setelah semua nilai pemesanan diatur dengan benar, `newnode->next` diatur untuk menunjuk ke `head1`, yang merupakan elemen pertama dalam linked list pemesanan.
   - Kemudian, `head1` diperbarui untuk menunjuk ke `newnode`, sehingga `newnode` menjadi elemen pertama dalam linked list pemesanan.
   - Dengan menggunakan kedua pecahan kode ini, kita dapat dengan mudah menampilkan informasi akun yang ada dalam linked list akun menggunakan fungsi `lihatakun()`. Selain itu, kita juga dapat menambahkan pemesanan

baru ke linked list pemesanan menggunakan fungsi `masukpemesanan()`. Hal ini membantu dalam mengelola dan memanipulasi informasi terkait akun dan pemesanan dalam program.

- **struct Pemesanan* newnode = (struct Pemesanan*)malloc(sizeof(struct Pemesanan));**

  Pada baris ini, kita membuat sebuah pointer newnode yang bertipe struct Pemesanan*. Fungsi malloc() digunakan untuk mengalokasikan memori yang diperlukan untuk menyimpan node baru dalam linked list. sizeof(struct Pemesanan) memberikan ukuran yang sesuai untuk tipe data struct Pemesanan, dan malloc() mengembalikan pointer ke memori yang dialokasikan.

- **newnode->kodepesanan = kodpes;**

  Di sini, nilai kodepesanan dari node baru diatur dengan nilai yang diterima sebagai argumen kodpes. Ini memasukkan kode pesanan ke dalam node baru.

- **strcpy(newnode->nama, nampes);**

  Fungsi strcpy() digunakan untuk menyalin string yang terdapat pada variabel nampes ke dalam array nama pada node baru. Ini menginisialisasi nama pemesan pada node baru.

- **newnode->kodebarangpesanan = kodbarpes;**

  Nilai **kodebarangpesanan** pada node baru diatur dengan nilai **kodbarpes**. Ini memasukkan kode barang pesanan ke dalam node baru.

- **strcpy(newnode->namabarang, nambar);**

  Seperti sebelumnya, fungsi **strcpy()** digunakan untuk menyalin string yang terdapat pada variabel **nambar** ke dalam array **namabarang** pada node baru. Ini menginisialisasi nama barang pada node baru.

- **newnode->jumlahpesanan = jumpes;**

  Nilai **jumlahpesanan** pada node baru diatur dengan nilai **jumpes**. Ini memasukkan jumlah pesanan ke dalam node baru.

- **newnode->tanggal = tgl;**

  Nilai tanggal pada node baru diatur dengan nilai tgl. Ini memasukkan tanggal pemesanan ke dalam node baru.

- **newnode->bulan = bln;**

  Nilai **bulan** pada node baru diatur dengan nilai **bln**. Ini memasukkan bulan pemesanan ke dalam node baru.
- **newnode->tahun = thn;**

  Nilai tahun pada node baru diatur dengan nilai thn. Ini memasukkan tahun pemesanan ke dalam node baru.
- **newnode->kota_asal = asal;**

  Nilai **kota_asal** pada node baru diatur dengan nilai **asal**. Ini memasukkan kode kota asal ke dalam node baru.
- **newnode->kota_tujuan = tujuan;**

  Nilai kota_tujuan pada node baru diatur dengan nilai tujuan. Ini memasukkan kode kota tujuan ke dalam node baru.
- **newnode->next = head1;**

  Pointer next pada node baru diatur untuk menunjuk ke head1, yaitu node pertama dalam linked list pemesanan sebelumnya (jika ada). Dengan mengatur next ke head1, node baru akan terhubung dengan node pertama dalam linked list.
- **head1 = newnode;**

  Terakhir, head1 diubah untuk menunjuk ke node baru. Dengan melakukan ini, node baru sekarang menjadi node pertama dalam linked list pemesanan. Jadi, -> Setiap kali fungsi masukpemesanan() dipanggil, sebuah node baru akan dibuat dan disisipkan di awal linked list pemesanan dengan mengubah head1 untuk menunjuk ke node baru. Hal ini memungkinkan kita untuk membangun linked list pemesanan dengan menambahkan pemesanan baru ke depan linked list setiap kali fungsi dipanggil.

```
void tambahpemesanan(){
    system("cls"); system("color 06");
    int kodpes,kodbarpes, jumpes, tgl,bln, thn,tujuan,asal;
    char nampes[50],nambar[40];
    fflush(stdin);
    printf("\t=============================================\n");
    printf("\t            Buat Pemesanan Barang            \n");
    printf("\t=============================================\n");
```

```
    printf("\tKode Pesanan  : ");
    scanf ("%d",&kodpes); fflush(stdin);
    printf("\tNama Pemesan  : ");
    scanf("%s", &nampes);   fflush(stdin);
    printf("\tKode Barang   : ");
    scanf ("%d",&kodbarpes); fflush(stdin);
    printf("\tNama Barang   : ");
    scanf("%s", &nambar);   fflush(stdin);
    printf("\tJumlah Pesanan    : ");
    scanf("%d", &jumpes);   fflush(stdin);
    printf("\tTanggal       : ");
    scanf("%d", &tgl);  fflush(stdin);
    printf("\tBulan         : ");
    scanf("%d", &bln);  fflush(stdin);
    printf("\tTahun         : ");
    scanf("%d", &thn);  fflush(stdin);
    printf("\tDaerah Asal   : ");
    printf("\n\t0. Denpasar");
    printf("\n\t1. Badung");
    printf("\n\t2. Gianyar");
    printf("\n\t3. Tabanan");
    printf("\n\t4. Klungkung");
    printf("\n\t5. Karangasem");
    printf("\n\t6. Bangli");
    printf("\n\t7. Buleleng ");
    printf("\n\t8. Jembbrana");
    printf("\n\tPilih Daerah Asal : ");
    scanf("%d", &asal);
    printf("\tDaerah Tujuan :");
    printf("\n\t0. Denpasar");
    printf("\n\t1. Badung");
    printf("\n\t2. Gianyar");
    printf("\n\t3. Tabanan");
    printf("\n\t4. Klungkung");
    printf("\n\t5. Karangasem");
    printf("\n\t6. Bangli");
    printf("\n\t7. Buleleng ");
    printf("\n\t8. Jembbrana");
    printf("\n\tPilih Daerah Tujuan : ");
    scanf("%d", &tujuan);

    masukpemesanan(kodpes,kodbarpes, jumpes, tgl,bln,
thn,nampes,nambar,tujuan,asal);
    getch();system("cls");
}
```

- Pada baris-baris ini, fungsi **tambahpemesanan()** dimulai. **system("cls")** digunakan untuk membersihkan layar konsol, sedangkan **system("color 06")** mengatur warna tampilan teks menjadi kuning muda.
- Kemudian, variabel-variabel lokal dideklarasikan. Variabel-variabel ini akan digunakan untuk menyimpan input pengguna seperti **kodpes** (kode pesanan), **kodbarpes** (kode barang pesanan), **jumpes** (jumlah pesanan), **tgl** (tanggal), **bln** (bulan), **thn** (tahun), **tujuan** (kota tujuan), **asal** (kota asal), **nampes** (nama pemesan), dan **nambar** (nama barang).
- **fflush(stdin)** digunakan untuk membersihkan input buffer sebelum menerima input dari pengguna.
- Bagian ini mengambil input dari pengguna untuk setiap variabel yang dideklarasikan sebelumnya. Pengguna akan diminta untuk memasukkan nilai-nilai seperti kode pesanan (kodpes), nama pemesan (nampes), kode barang pesanan (kodbarpes), nama barang (nambar), jumlah pesanan (jumpes), tanggal (tgl), bulan (bln), tahun (thn), kota asal (asal), dan kota tujuan (tujuan). scanf digunakan untuk membaca input dari pengguna dan menyimpannya dalam variabel yang sesuai. fflush(stdin) digunakan lagi setelah setiap scanf untuk membersihkan buffer input sebelum menerima input berikutnya. Selain itu, juga terdapat tampilan pilihan daerah asal dan daerah tujuan yang akan ditampilkan kepada pengguna sebelum menerima input. Pengguna dapat memilih kode sesuai dengan daerah yang diinginkan.

```
void lihatpemesanan(int graph[V][V]){
    struct Pemesanan* lihat;
    system("cls"); system("color 06");

    if (head1 == NULL){
        printf (" Tidak Ada Proses Pemesanan Barang ");
    }
    else{
        printf("\n\t===========================================\n"
);
        printf("\t         Lihat Proses Pemesanan Barang       \n");
        printf("\t===========================================");
        lihat = head1;
        while (lihat!= NULL){
```

```
            printf("\n\n\tKode Pesanan  : %d", lihat->kodepesanan);
            printf("\n\tNama Pemesan    : %s", lihat->nama);
            printf("\n\tKode Barang     : %d", lihat-
>kodebarangpesanan);
            printf("\n\tNama Barang : %s", lihat->namabarang);
            printf("\n\tJumlah Barang   : %d", lihat-
>jumlahpesanan);
            printf("\n\tTanggal Pesan   : %d", lihat->tanggal);
            printf("\n\tBulan           : %d", lihat->bulan);
            printf("\n\tTahun           : %d", lihat->tahun);
printf("\n");
            dijkstra(graph,10,lihat->kota_asal,lihat->kota_tujuan);
            lihat = lihat->next;
        }
    }
```

- Fungsi **lihatpemesanan()** dimulai dengan membersihkan layar menggunakan **system("cls")** dan mengatur warna tampilan teks menjadi kuning muda dengan **system("color 06")**.

- Selanjutnya, dilakukan pengecekan apakah **head1** (pointer ke node pertama dalam linked list pemesanan) bernilai NULL. Jika iya, maka tidak ada proses pemesanan barang yang tersedia, dan pesan "Tidak Ada Proses Pemesanan Barang" akan dicetak. Jika tidak, maka proses pemesanan barang akan ditampilkan.

- Pointer lihat ditetapkan sebagai head1 untuk mengiterasi melalui setiap node dalam linked list pemesanan.
- Selama lihat tidak NULL, data pemesanan seperti kode pesanan (kodepesanan), nama pemesan (nama), kode barang pesanan (kodebarangpesanan), nama barang (namabarang), jumlah barang (jumlahpesanan), tanggal pesan (tanggal), bulan (bulan), dan tahun (tahun) akan dicetak menggunakan printf().
- Setelah mencetak informasi pemesanan, fungsi dijkstra() akan dipanggil dengan argumen graph, 10 (jumlah vertex dalam graf), kota_asal, dan kota_tujuan. Fungsi ini digunakan untuk melakukan algoritma Dijkstra pada graf untuk mencari jalur terpendek antara dua kota.
- Terakhir, pointer lihat akan diperbarui dengan lihat->next untuk melanjutkan iterasi ke node pemesanan berikutnya dalam linked list.

```
void dijkstra(int G[V][V],int n, int startnode, int targetnode)
{
    struct Pemesanan* lihat;
    int cost[V][V],distance[V],pred[V];
    int visited[V],count,mindistance,nextnode,i,j;
    char kota[25][12] =
{"Denpasar","Badung","Gianyar","Tabanan","Klungkung","Karangasem","B
angli","Buleleng", "Jembrana", };

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            if(G[i][j]==0)
                cost[i][j]=INFINITY;
            else
                cost[i][j]=G[i][j];

    for(i=0;i<n;i++){
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
    }
```

- Fungsi **dijkstra()** mengimplementasikan algoritma Dijkstra untuk mencari jalur terpendek dalam graf.
- Parameter **G** adalah matriks dua dimensi yang mewakili graf dengan **n** node.
- **startnode** adalah node awal dari jalur yang akan dicari.
- **targetnode** adalah node tujuan yang dituju dalam pencarian jalur terpendek.
- Pointer **lihat** bertipe **struct Pemesanan*** digunakan untuk mengiterasi melalui linked list pemesanan.
- Matriks **cost** digunakan untuk menyimpan biaya/jarak antara setiap pasangan node dalam graf.
- Array **distance** menyimpan jarak terpendek dari **startnode** ke setiap node.
- Array **pred** menyimpan prediksi node sebelumnya dalam jalur terpendek.
- Array **visited** menandai node yang sudah dikunjungi selama pencarian jalur terpendek.

- Array **kota** berisi daftar nama kota yang sesuai dengan indeks node dalam graf.

```
for (i = 0; i < n; i++) {
  distance[i] = cost[startnode][i];
  pred[i] = startnode;
  visited[i] = 0;
}
```

- Melalui loop **for**, **distance**, **pred**, dan **visited** diinisialisasi dengan nilai awal.
- Setiap **distance[i]** diatur dengan nilai biaya/jarak dari **startnode** ke node **i**.
- Setiap **pred[i]** diatur dengan **startnode** sebagai prediksi node sebelumnya dalam jalur terpendek.
- Setiap **visited[i]** diatur sebagai 0 untuk menandai bahwa node belum dikunjungi.

```
while (count < n - 1) {
    mindistance = INFINITY;
    for (i = 0; i < n; i++)
      if (distance[i] < mindistance && !visited[i]) {
        mindistance = distance[i];
        nextnode = i;
      }
    visited[nextnode] = 1;
    for (i = 0; i < n; i++)
      if (!visited[i] && (mindistance + cost[nextnode][i] < distance[i])) {
```

```
            distance[i] = mindistance + cost[nextnode][i];
            pred[i] = nextnode;
        }
    count++;
}
```

- Loop **while** digunakan untuk melakukan iterasi hingga **count** mencapai **n-1**, di mana **n** adalah jumlah total node dalam graf.
- Tujuan dari loop ini adalah untuk menemukan jalur terpendek dari **startnode** ke semua node lain dalam graf.
- Pada setiap iterasi loop, mencari node berikutnya yang memiliki jarak terpendek dari **startnode**.
- Pertama, **mindistance** diatur ke **INFINITY** sebagai nilai awal untuk mencari jarak terpendek.
- Loop **for** berikutnya digunakan untuk memeriksa setiap node dalam graf.
- Jika jarak **distance[i]** lebih kecil dari **mindistance** dan node **i** belum dikunjungi (**!visited[i]**), maka **mindistance** diperbarui dengan **distance[i]** dan **nextnode** diatur sebagai **i**.
- Setelah menemukan node dengan jarak terpendek yang belum dikunjungi, node tersebut ditandai sebagai telah dikunjungi dengan mengatur **visited[nextnode] = 1**.
- Selanjutnya, dilakukan loop **for** untuk memperbarui jarak terpendek dari **startnode** ke node lain yang belum dikunjungi.
- Jika node **i** belum dikunjungi dan jumlah dari **mindistance** dengan biaya dari **nextnode** ke **i** lebih kecil dari **distance[i]**, maka **distance[i]** diperbarui dengan jumlah tersebut dan **pred[i]** diatur sebagai **nextnode**.
- Setelah loop ini selesai, jalur terpendek dari **startnode** ke setiap node telah ditemukan dan disimpan dalam **distance** dan **pred**.

- Dengan melakukan ini, kita secara bertahap memperbarui jarak terpendek dan node sebelumnya dalam jalur terpendek untuk setiap node yang belum dikunjungi.
- Loop **while** terus berjalan hingga semua node dalam graf dikunjungi dan jalur terpendek dari **startnode** ke semua node lainnya ditemukan.
- Setiap iterasi loop, **count** ditingkatkan hingga mencapai **n-1**, menandakan bahwa iterasi telah selesai.

```
    printf("\n\n\tJalur Pengiriman : %s", kota[targetnode]);
    printf(" <- ");
    j = targetnode;
    while (j != startnode) {
        j = pred[j];
        printf("%s", kota[j]);
        if (j != startnode)
            printf(" <- ");
    }
    printf("\n\tJarak Terpendek : %d KM\n", distance[targetnode]);
}
```

- Setelah iterasi selesai, jalur terpendek dan jarak terpendek dari startnode ke targetnode telah ditemukan.
- Melalui loop while, jalur terpendek dicetak dengan mengikuti nilai pred dari targetnode hingga mencapai startnode.
- Setiap node dalam jalur dicetak menggunakan kota[node], yang mengambil nama kota dari array kota berdasarkan indeks node.
- Pemisah "<-" digunakan untuk memisahkan setiap node dalam jalur.

- Setelah mencetak jalur, jarak terpendek dari startnode ke targetnode dicetak menggunakan distance[targetnode].

```
distance[startnode]=0;
visited[startnode]=1;
count=1;

while(count<n-1){
    mindistance=INFINITY;

    for(i=0;i<n;i++)
        if(distance[i]<mindistance&&!visited[i]){
            mindistance=distance[i];
            nextnode=i;
        }

        visited[nextnode]=1;
        for(i=0;i<n;i++)
            if(!visited[i])
                if(mindistance+cost[nextnode][i]<distance[i]){
                    distance[i]=mindistance+cost[nextnode][i];
                    pred[i]=nextnode;
                }
    count++;
}
```

- Pada awalnya, kita menginisialisasi jarak **distance** dari **startnode** ke semua node lain dengan nilai tak terhingga (**INFINITY**). Ini mengindikasikan bahwa belum ada jalur yang diketahui dari **startnode** ke node lain. kecuali **distance[startnode]** yang diatur menjadi 0, karena jarak dari **startnode** ke dirinya sendiri adalah 0.
- Array **visited** digunakan untuk melacak status kunjungan pada setiap node dalam proses pencarian jalur terpendek. Semua elemen **visited** diatur sebagai 0 untuk menandakan bahwa semua node belum dikunjungi.
- **count** diatur sebagai 1 karena kita telah mengunjungi **startnode**.
- Selama **count** kurang dari **n - 1** (jumlah total node dikurangi 1), yang berarti masih ada node yang belum dikunjungi, dilakukan iterasi untuk mencari node berikutnya dengan jarak terpendek yang belum dikunjungi.

- Pada awal setiap iterasi, **mindistance** diatur kembali sebagai nilai tak terhingga. Hal ini dilakukan untuk mencari jarak terpendek baru dalam setiap iterasi.
- Loop **for** digunakan untuk memeriksa setiap node dalam graf.
- Jika jarak **distance[i]** lebih kecil dari **mindistance** dan node **i** belum dikunjungi (**!visited[i]**), maka **mindistance** diperbarui dengan **distance[i]**, dan **nextnode** diatur sebagai **i**. Dengan melakukan ini, kita mencari node dengan jarak terpendek yang belum dikunjungi.
- Setelah menemukan node dengan jarak terpendek yang belum dikunjungi, node tersebut ditandai sebagai sudah dikunjungi dengan mengatur **visited[nextnode] = 1**.
- Selanjutnya, dilakukan loop **for** untuk memperbarui jarak terpendek dari **startnode** ke node lain yang belum dikunjungi.
- Jika node **i** belum dikunjungi, maka dilakukan pengecekan apakah jumlah dari **mindistance** dengan biaya dari **nextnode** ke **i** lebih kecil dari **distance[i]**. Jika ya, berarti kita telah menemukan jalur yang lebih pendek dari **startnode** ke node **i**.
- Dalam hal ini, **distance[i]** diperbarui dengan jumlah tersebut, dan **pred[i]** diatur sebagai **nextnode**, yang berarti node **nextnode** adalah node sebelumnya dalam jalur terpendek menuju node **i**.
- Dengan melakukan ini, kita secara bertahap memperbarui jarak terpendek dan node sebelumnya dalam jalur terpendek.
- Setelah selesai memperbarui semua node yang belum dikunjungi, **count** ditingkatkan dan iterasi dilanjutkan hingga semua node dikunjungi.
- Loop **while** berakhir saat **count** mencapai **n - 1**, yang berarti semua node telah dikunjungi dan kita telah menemukan jalur terpendek dari **startnode** ke semua node lain dalam graf.

- Pada akhirnya, kita memiliki array distance yang berisi jarak terpendek dari startnode ke setiap node, dan array pred yang menyimpan node sebelumnya dalam jalur terpendek menuju setiap node.

```
int a;
    int u;
    char tes[20][8] =
{"Senin","Selasa","Rabu","Kamis","Jumat","Sabtu","Minggu"};
    for(i=0;i<n;i++)
        if(i==targetnode){
            printf("\tDaerah Asal \t: %s", kota[startnode]);
            printf("\n\tDaerah Tujuan \t: %s", kota[i]);
            printf("\n\tJarak Tujuan \t: %d KM", distance[i]);
            printf("\n\tJalur \t\t: %s", kota[i]);

            j=i;
            do{
                j=pred[j];
                printf(" <- %s",kota[j]);
            }while(j!=startnode);
            a=distance[i];
            if (a<=30){
                u = 1;
                printf("\n\tPengiriman\t: SENIN");
                printf("\n\tWaktu Pengiriman: 1 Hari");
                printf("\n\tWaktu Sampai\t: %s\n",tes[u]);
            }
            else if (a<=120){
                u = 3;
                printf("\n\tPengiriman\t: SENIN");
                printf("\n\tWaktu Pengiriman: 3 Hari");
                printf("\n\tWaktu Sampai\t: %s\n",tes[u]);
            }
            else if (a<=180){
                u = 5;
                printf("\n\tPengiriman\t: SENIN");
                printf("\n\tWaktu Pengiriman: 5 Hari");
                printf("\n\tWaktu Sampai\t: %s\n",tes[u]);
            }
            else if (a>=390){
                u = 7;
                printf("\n\tPengiriman\t: SENIN");
                printf("\n\tWaktu Pengiriman: 7 Hari");
                printf("\n\tWaktu Sampai\t: %s\n",tes[u]);
            }
    }
```

```
}
```

- Variabel **a** digunakan untuk menyimpan jarak terpendek dari **startnode** ke **targetnode**.
- Variabel **u** digunakan untuk menyimpan indeks yang akan digunakan untuk mengakses array **tes**.
- Array **tes** merupakan array 2 dimensi yang berisi nama-nama hari dalam bahasa Inggris ("Senin", "Selasa", dst.). Array ini akan digunakan untuk menampilkan informasi waktu sampai berdasarkan jarak yang telah dihitung.
- Dilakukan loop **for** untuk memeriksa setiap node dalam graf.
- Pada setiap iterasi, kode melakukan pengecekan apakah **i** (node saat ini) sama dengan **targetnode**. Jika ya, maka informasi terkait pemesanan dan rute pengiriman akan ditampilkan.
- Pertama, dicetak kota asal (**startnode**) dan kota tujuan (**i**) menggunakan **printf()**.
- Kemudian, dicetak jarak tujuan **distance[i]** (jarak terpendek dari **startnode** ke **i**) dalam satuan kilometer.
- Selanjutnya, dicetak jalur yang diikuti dari **startnode** ke **i**. Hal ini dilakukan dengan menggunakan variabel **j** yang awalnya diisi dengan nilai **i**, kemudian iterasi dilakukan dengan **do-while** untuk mencetak jalur mundur dari **i** ke **startnode** dengan menggunakan array **kota**.
- Nilai jarak terpendek (**a**) kemudian digunakan untuk menentukan waktu pengiriman dan waktu sampai.
- Pada kondisi **if** dan **else if**, dilakukan perbandingan jarak (**a**) dengan ambang batas tertentu untuk menentukan waktu pengiriman dan waktu sampai. Jika jarak berada dalam kisaran tertentu, maka indeks **u** akan disesuaikan untuk mengakses array **tes** dan mencetak waktu pengiriman dan waktu sampai yang sesuai.
- Informasi pengiriman dan waktu sampai ditampilkan di layar.

```
void masukbarang(int kodbar,char nambar[],int stock1){
```

```
    struct Barang* newnode = (struct Barang*)malloc(sizeof(struct
Barang));
    newnode->kodebarang = kodbar;
    strcpy(newnode->namabarang, nambar);
    newnode->stock = stock1;
    newnode->next = head;
    head = newnode;
}
```

- Fungsi **masukbarang** digunakan untuk menambahkan sebuah barang baru ke dalam linked list **Barang**.
- Fungsi ini menerima tiga parameter:
    - **kodbar**: Kode barang yang akan ditambahkan.
    - **nambar**: Nama barang yang akan ditambahkan.
    - **stock1**: Jumlah stok barang yang tersedia.
- Pertama, sebuah node baru (**newnode**) dibuat dengan menggunakan fungsi **malloc**. **sizeof(struct Barang)** digunakan untuk mengalokasikan memori yang cukup untuk menyimpan informasi tentang barang dalam struktur **Barang**.
- Nilai dari parameter **kodbar** disimpan dalam variabel **kodebarang** dari **newnode** untuk menyimpan kode barang yang baru ditambahkan.
- Nama barang (**nambar**) disalin ke dalam variabel **namabarang** dari **newnode** menggunakan fungsi **strcpy**. Ini memastikan bahwa data nama barang yang diberikan tidak akan konflik dengan memori saat disimpan di **newnode**.
- Jumlah stok barang (**stock1**) disimpan dalam variabel **stock** dari **newnode**. Ini memungkinkan untuk melacak jumlah stok barang yang tersedia.
- Selanjutnya, **next** dari **newnode** diatur untuk menunjuk ke **head**. Ini dilakukan untuk menyambungkan **newnode** dengan linked list yang sudah ada. Dalam hal ini, **newnode** akan menjadi node pertama dalam linked list.
- Terakhir, **head** diupdate untuk menunjuk ke **newnode**. Ini dilakukan agar **newnode** menjadi node pertama dalam linked list, sehingga barang baru berhasil ditambahkan di depan linked list **Barang**.
- Dengan demikian, fungsi **masukbarang** bertanggung jawab untuk mengalokasikan memori untuk node baru, mengisi nilainya dengan informasi tentang barang yang akan ditambahkan, dan menyambungkannya

dengan linked list **Barang** yang sudah ada. Hal ini memungkinkan pengguna untuk menambahkan barang baru ke linked list dengan melakukan operasi prepend.

```
void tambahbarang(){
    system("cls");
    int kodbar, stock1;
    char nambar[50];

    fflush(stdin);
    printf("\t===============================================\n");
    printf("\t              Tambah Barang                    \n");
    printf("\t===============================================\n");
    printf("\tKode Barang       : ");
    scanf("%d", &kodbar);  fflush(stdin);
    printf("\tNama Barang       : ");
    scanf("%s", &nambar);   fflush(stdin);
    printf("\tStock Barang      : ");
    scanf("%d",&stock1);    fflush(stdin);
    masukbarang(kodbar,nambar,stock1);
    getch();system("cls");
}
```

- Fungsi **tambahbarang** digunakan untuk menambahkan barang baru ke dalam linked list **Barang**.
- Pertama, layar dikosongkan menggunakan **system("cls")** untuk membersihkan tampilan konsol sebelum menampilkan pesan dan menerima input dari pengguna.
- Variabel **kodbar** dideklarasikan untuk menyimpan kode barang yang akan ditambahkan. Ini adalah bilangan bulat.
- Array **nambar** dideklarasikan untuk menyimpan nama barang yang akan ditambahkan. Array ini memiliki panjang maksimal 50 karakter.
- Pengguna diminta untuk memasukkan nilai **kodbar** menggunakan **scanf** dengan format **%d**. **%d** digunakan untuk membaca bilangan bulat. **&kodbar** digunakan untuk menyimpan nilai yang dimasukkan oleh pengguna ke dalam variabel **kodbar**.
- Pengguna juga diminta untuk memasukkan nilai **nambar** menggunakan **scanf** dengan format **%s**. **%s** digunakan untuk membaca string (kata atau

kalimat). **&nambar** digunakan untuk menyimpan string yang dimasukkan oleh pengguna ke dalam array **nambar**.

- Pengguna diminta untuk memasukkan nilai **stock1** menggunakan **scanf** dengan format **%d**. **&stock1** digunakan untuk menyimpan nilai yang dimasukkan oleh pengguna ke dalam variabel **stock1**.
- Setelah selesai menerima input dari pengguna, fungsi **masukbarang** dipanggil dengan argumen **kodbar**, **nambar**, dan **stock1**. Ini akan membuat node baru dalam linked list **Barang** dengan menggunakan nilai-nilai yang dimasukkan oleh pengguna.
- Setelah itu, getch() digunakan untuk menunggu pengguna menekan tombol sebelum melanjutkan, dan layar dikosongkan lagi menggunakan system("cls") untuk membersihkan tampilan konsol sebelum kembali ke menu utama.

```
void updatebarang(){
    struct Barang *current;
    char namabarang1[50];
    int stock1, num;
    system("cls");

    printf("\t=============================================\n");
    printf("\t              Update Barang                  \n");
    printf("\t=============================================\n");

    printf("\tMasukkan Nilai Yang ingin di ubah : ");
    scanf("%d", &num);
    fflush(stdin);

    if(head==NULL){
        printf("\tList Saat ini Masih Kosong"); getch();
system("cls"); return;
    }

    current = head;
    while(current!=NULL){
        if(num==current->kodebarang){
            printf("\t=============================================\
n");
            printf("\tMasukkan Nama Barang  : ");
            scanf("%s", &namabarang1);
```

```
            strcpy(current->namabarang, namabarang1);
            printf("\tMasukkan Stock Baru   : ");
            scanf("%d", &stock1);
            current->stock = stock1;
            return;
        }
        current=current->next;
    }
    printf("\tKode Barang %d Tidak Berada di dalam List", num);
    getch(); system("cls");
}
```

- Fungsi **updatebarang** digunakan untuk mengubah informasi barang yang ada dalam linked list **Barang**.
- Pertama, layar dikosongkan menggunakan **system("cls")** untuk membersihkan tampilan konsol sebelum menampilkan pesan dan menerima input dari pengguna.
- Variabel **num** dideklarasikan untuk menyimpan nilai yang ingin diubah. Ini adalah bilangan bulat.
- Array **namabarang1** dideklarasikan untuk menyimpan nama barang yang baru setelah diubah. Array ini memiliki panjang maksimal 50 karakter.
- Pengguna diminta untuk memasukkan nilai **num** menggunakan **scanf** dengan format **%d**. **%d** digunakan untuk membaca bilangan bulat. **&num** digunakan untuk menyimpan nilai yang dimasukkan oleh pengguna ke dalam variabel **num**.
- Setelah itu, dilakukan pengecekan apakah linked list **Barang** kosong atau tidak. Jika kosong, pesan "List Saat ini Masih Kosong" ditampilkan kepada pengguna, dan fungsi segera mengembalikan kontrol ke menu utama menggunakan **getch()** dan **system("cls")**.
- Jika linked list **Barang** tidak kosong, dilakukan penelusuran dari awal linked list sampai akhir menggunakan pointer **current**. Pada setiap langkah penelusuran, dicek apakah **num** sama dengan **current->kodebarang**. Jika iya, artinya ditemukan barang dengan kode yang sesuai yang akan diubah.
- Pengguna diminta untuk memasukkan nilai baru untuk nama barang menggunakan **scanf** dengan format **%s**. **%s** digunakan untuk membaca string (kata atau kalimat). **&namabarang1** digunakan untuk menyimpan string yang dimasukkan oleh pengguna ke dalam array **namabarang1**.

- Nama barang di dalam node **current** diubah menggunakan fungsi **strcpy** dengan parameter **current->namabarang** dan **namabarang1**.
- Pengguna juga diminta untuk memasukkan nilai baru untuk stok barang menggunakan **scanf** dengan format **%d**. **&stock1** digunakan untuk menyimpan nilai yang dimasukkan oleh pengguna ke dalam variabel **stock1**.
- Stok barang di dalam node **current** diubah menjadi nilai baru **stock1**.
- Jika kode barang yang dicari tidak ditemukan dalam linked list, pesan "Kode Barang [num] Tidak Berada di dalam List" ditampilkan kepada pengguna.
- Setelah pengubahan selesai atau pesan kesalahan ditampilkan, fungsi kembali ke menu utama setelah menunggu pengguna menekan tombol menggunakan getch() dan layar dikosongkan menggunakan system("cls").

```
void lihatbarang(){
struct Barang* lihat;
system("cls"); printf("\n");

    if(head==NULL){
        printf("\n\tData Barang Kosong\n");
    }
    else{
    printf("\t============================================\n");
    printf("\t=              Lihat Barang                =\n");
    printf("\t============================================");
    lihat = head;

        while(lihat != NULL){
            printf("\n\tKode Barang     : %d", lihat->kodebarang);
            printf("\n\tNama Barang     : %s", lihat->namabarang);
            printf("\n\tStock Barang    : %d", lihat->stock);
            lihat = lihat->next;
        }
    }
    getch(); system("cls");
}
```

- Fungsi **lihatbarang** digunakan untuk menampilkan informasi tentang semua barang yang ada dalam linked list **Barang**.

- Pertama, layar dikosongkan menggunakan **system("cls")** untuk membersihkan tampilan konsol sebelum menampilkan informasi barang.
- Jika linked list **Barang** kosong (yaitu **head** bernilai NULL), pesan "Data Barang Kosong" ditampilkan kepada pengguna.
- Jika linked list **Barang** tidak kosong, tampilan informasi barang dimulai dengan judul "Lihat Barang" yang diapit oleh garis pembatas.
- Pointer **lihat** diberikan nilai awal dari **head**, dan kemudian dilakukan penelusuran linked list menggunakan loop **while** selama **lihat** tidak NULL.
- Pada setiap iterasi loop, informasi barang seperti kode barang (**lihat->kodebarang**), nama barang (**lihat->namabarang**), dan stok barang (**lihat->stock**) ditampilkan kepada pengguna.
- Setelah semua barang ditampilkan, fungsi menunggu pengguna menekan tombol menggunakan getch() dan layar dikosongkan menggunakan system("cls").

```
void hapusbarang(){
system("cls");
struct Barang *temp = NULL;
int num;

    printf("\t============================================\n");
    printf("\t              HAPUS Barang                  \n");
    printf("\t============================================\n");
    printf("\tMasukkan List Barang Yang ingin di Hapus : ");
    scanf("%d", &num);

    if(head==NULL){
        printf("\n\tList Saat ini Masih Kosong");
        getch(); system("cls"); return;
    }

    if(num==head->kodebarang){  //Mengecek list pertama
        temp = head;
        head = head->next;
        free(temp);
        return;
    }

    p = head;
    while(p->next!=NULL){   //Mengecek list berikutnya
```

```
        if(p->next->kodebarang==num){
            temp = p->next;
            p->next = temp->next;
            free(temp);
            getch(); system("CLS");
            return;
        }
        p = p->next;
    }
    printf("\n\tKode Barang %d Tidak Berada di dalam List", num);
    getch(); system("cls");
}
```

- Fungsi **hapusbarang** digunakan untuk menghapus barang dari linked list **Barang** berdasarkan kode barang yang diinput oleh pengguna.
- Pertama, layar dikosongkan menggunakan **system("cls")** untuk membersihkan tampilan konsol sebelum melakukan operasi hapus barang.
- Pengguna diminta untuk memasukkan kode barang yang ingin dihapus melalui **scanf("%d", &num)**.
- Pengecekan awal dilakukan untuk memastikan apakah linked list **Barang** kosong atau tidak. Jika **head** bernilai NULL, berarti linked list kosong, dan pesan "List Saat ini Masih Kosong" ditampilkan kepada pengguna. Selanjutnya, **getch()** digunakan untuk menunggu pengguna menekan tombol sebelum melanjutkan, dan **system("cls")** digunakan untuk membersihkan tampilan konsol.
- Jika kode barang yang ingin dihapus sama dengan kode barang pada elemen pertama (**num == head->kodebarang**), maka elemen pertama tersebut dihapus dari linked list. Untuk melakukan hal ini, **temp** diatur untuk menunjuk ke elemen pertama, **head** diubah untuk menunjuk ke elemen berikutnya, dan **free(temp)** digunakan untuk membebaskan memori yang dialokasikan untuk elemen yang dihapus. Setelah itu, fungsi selesai dengan pernyataan **return**.
- Jika kode barang yang ingin dihapus tidak sama dengan kode barang pada elemen pertama, maka dilakukan penelusuran linked list dengan menggunakan variabel penunjuk **p**. Pengecekan dilakukan pada setiap

elemen berikutnya (**p->next**) untuk memeriksa apakah kode barang pada elemen tersebut sama dengan **num**.

- Jika kode barang ditemukan pada elemen berikutnya (**p->next->kodebarang == num**), maka elemen tersebut dihapus dari linked list. Untuk melakukan hal ini, **temp** diatur untuk menunjuk ke elemen yang akan dihapus (**p->next**), **p->next** diubah untuk melewati elemen yang dihapus (**p->next = temp->next**), dan **free(temp)** digunakan untuk membebaskan memori yang dialokasikan untuk elemen yang dihapus. Setelah itu, **getch()** digunakan untuk menunggu pengguna menekan tombol sebelum melanjutkan, dan **system("cls")** digunakan untuk membersihkan tampilan konsol.
- Jika kode barang yang ingin dihapus tidak ditemukan pada linked list, pesan "Kode Barang {num} Tidak Berada di dalam List" ditampilkan kepada pengguna. Setelah itu, **getch()** digunakan untuk menunggu pengguna menekan tombol sebelum melanjutkan, dan **system("cls")** digunakan untuk membersihkan tampilan konsol.
- Terakhir, layar dikosongkan menggunakan system("cls") untuk membersihkan tampilan konsol sebelum fungsi selesai.

```
void detailbarang(){
system("cls"); printf("\n");
int search_detail;
int kembali;
struct Barang* detail;
struct Barang* letak;
struct Pemesanan* lihat;

    if(head==NULL){
        printf("\n\tDetail Barang Kosong\n");
    }
    else{
        printf("\t===============================================\n");
        printf("\t=            Lihat Detail Barang              =\n");
        printf("\t===============================================\n");
        detail = head;

        while(detail != NULL){
            printf("\n\t[%d]\t[%s] \n", detail->kodebarang, detail-
>namabarang);
```

```
            detail = detail->next;
        }
        printf("\tSilahkan Masukkan kode barang yang ingin dilihat :
");
        scanf("%d", &search_detail);

        letak = head;
        lihat = head1;
        while(letak!=NULL){
            if(search_detail==letak->kodebarang){
                printf("\n\tKode Barang     : %d", letak-
>kodebarang);
                printf("\n\tNama Barang     : %s", letak-
>namabarang);
                printf("\n\tStock Barang        : %d", letak-
>stock);
                while (lihat != NULL){
                    if(letak->kodebarang == lihat-
>kodebarangpesanan){
                        printf("\n\tNama Pemesan        : %s \n",
lihat->nama);
                        lihat = lihat->next;
                        kembali =1;
                    }
                    else{
                        lihat = lihat->next;
                        kembali =0;
                    }
                }
                if (kembali == 0){
                    printf("\n\tBarang belum di pesan");
                }
            }
            letak = letak->next;
        }
    }
    getch();
    system("cls");
}
```

- Fungsi **detailbarang** digunakan untuk menampilkan detail barang dan informasi pemesanan yang terkait dengan barang tersebut.
- Pertama, layar dikosongkan menggunakan **system("cls")** untuk membersihkan tampilan konsol sebelum menampilkan detail barang.

- Jika linked list **Barang** kosong (**head == NULL**), maka pesan "Detail Barang Kosong" ditampilkan kepada pengguna. Ini menandakan bahwa tidak ada barang yang tersedia.
- Jika linked list **Barang** tidak kosong, tampilan menu untuk melihat detail barang ditampilkan kepada pengguna.
- Variabel **detail** diinisialisasi dengan **head** untuk melakukan penelusuran pada linked list **Barang** dan menampilkan daftar barang yang tersedia. Setiap elemen **Barang** ditampilkan dengan mencetak kode barang dan nama barang.
- Selanjutnya, dilakukan penelusuran pada linked list **Barang** menggunakan variabel **detail** untuk menampilkan daftar barang yang tersedia. Setiap elemen **Barang** ditampilkan dengan mencetak kode barang dan nama barang.
- Pengguna diminta memasukkan kode barang yang ingin dilihat detailnya melalui **scanf("%d", &search_detail)**.
- Variabel **letak** dan **lihat** diinisialisasi dengan **head** dan **head1** untuk melakukan penelusuran pada linked list **Barang** dan **Pemesanan**.
- Dilakukan penelusuran pada linked list **Barang** dengan menggunakan variabel **letak**. Jika kode barang yang dicari (**search_detail**) ditemukan pada elemen **letak**, maka detail barang tersebut ditampilkan, termasuk kode barang, nama barang, dan stok barang.
- Selanjutnya, dilakukan penelusuran pada linked list **Pemesanan** dengan menggunakan variabel **lihat**. Jika kode barang pada elemen **lihat** sama dengan kode barang pada elemen **letak**, maka nama pemesan ditampilkan.
- Penggunaan variabel **kembali** berguna untuk menandai apakah ada pesanan yang terkait dengan barang tersebut.
- Jika tidak ada pesanan yang terkait dengan barang tersebut, pesan "Barang belum di pesan" ditampilkan kepada pengguna.
- Setelah penelusuran selesai, pengguna diminta menekan tombol untuk melanjutkan menggunakan getch(), dan layar dikosongkan menggunakan system("cls") untuk membersihkan tampilan konsol sebelum fungsi selesai.

```
void ubahgraph(){
```

```
    int kota_awal, kota_tujuan, jarak_baru;
    char kota[25][12] =
{"Denpasar","Bangli","Karangasem","Jembrana", ""};
    system("cls");

    printf("\t==============================================\n");
    printf("\t=             Update Data Barang             =\n");
    printf("\t==============================================\n");
    printf("\tPilih Kota Yang ingin di rubah\n");
    printf("\n\t0. Denpasar");
    printf("\n\t1. Badung");
    printf("\n\t2. Gianyar");
    printf("\n\t3. Tabanan");
    printf("\n\t4. Klungkung");
    printf("\n\t5. Karangasem");
    printf("\n\t6. Bangli");
    printf("\n\t7. Buleleng ");
    printf("\n\t8. Jembbrana");
    printf("\tMasukkan Kota : ");
    scanf("%d", &kota_awal);
    fflush(stdin);

    printf("\tKota %s Ke Kota? : \n", kota[kota_awal]);
    printf("\n\t0. Denpasar");
    printf("\n\t1. Badung");
    printf("\n\t2. Gianyar");
    printf("\n\t3. Tabanan");
    printf("\n\t4. Klungkung");
    printf("\n\t5. Karangasem");
    printf("\n\t6. Bangli");
    printf("\n\t7. Buleleng ");
    printf("\n\t8. Jembbrana");
    printf("\tMasukkan Kota : ");
    scanf("%d", &kota_tujuan);
    fflush(stdin);
    printf("\tMasukkan jarak yang Baru : ");
    scanf("%d", &jarak_baru);
    fflush(stdin);

    graph[kota_awal][kota_tujuan] = jarak_baru;
    printf("\tJarak %s ke %s yang baru adalah %d Km",
kota[kota_awal], kota[kota_tujuan], jarak_baru*100);

    getch();
    system("cls");
}
```

- Fungsi **ubahgraph()** bertujuan untuk memperbarui data jarak antar kota dalam graf.
- Variabel **kota_awal**, **kota_tujuan**, dan **jarak_baru** digunakan untuk menyimpan pilihan pengguna dan jarak baru yang dimasukkan.
- Array **kota** berisi nama-nama kota yang akan ditampilkan kepada pengguna.
- Pertama, layar dikosongkan menggunakan **system("cls")** agar tampilan pada konsol bersih sebelum menampilkan menu pengubahan data jarak.
- Selanjutnya, menu untuk memilih kota awal ditampilkan menggunakan **printf()** dengan menampilkan daftar kota yang dapat dipilih.
- Pengguna diminta untuk memasukkan pilihan kota awal menggunakan **scanf("%d", &kota_awal)**. **fflush(stdin)** digunakan untuk membersihkan *buffer* masukan sebelumnya agar tidak mempengaruhi input selanjutnya.
- Setelah memilih kota awal, menu untuk memilih kota tujuan ditampilkan dengan cara yang serupa.
- Pengguna diminta untuk memasukkan pilihan kota tujuan menggunakan **scanf("%d", &kota_tujuan)**. Kembali, **fflush(stdin)** digunakan untuk membersihkan *buffer* masukan.
- Pengguna kemudian diminta untuk memasukkan jarak baru antara kota awal dan kota tujuan menggunakan **scanf("%d", &jarak_baru)**.
- Data jarak antara kota awal dan kota tujuan diubah dengan mengassign nilai **jarak_baru** ke elemen **graph[kota_awal][kota_tujuan]**.
- Pesan yang menampilkan jarak baru yang telah diubah ditampilkan kepada pengguna.
- Pengguna diminta menekan tombol untuk melanjutkan menggunakan **getch()**, dan layar dikosongkan menggunakan **system("cls")** untuk membersihkan tampilan konsol sebelum fungsi selesai.

Penjelasan alur program, penulis sajikan dalam bentuk video di link google drive berikut: https://drive.google.com/file/d/1bGfFoPzYE6SGKkqlRn36OcamJo5yqMre/view?usp=drive_link.

BAB III
KESIMPULAN

Berdasarkan pembahasan pada bab II, dapat disimpulkan bahwa;

1. Solusi yang penulis tawarkan adalah program manajemen pemesanan dan pengiriman barang menggunaakn single linked list dan menerapkan algoritma graph yakni algoritma djikstra untuk mencari rute terpendek. Sebelum membuat implementasi program, penulis menyusun rancangan flowchart sebagai alur dari setiap proses program. Penjelasan flowchart penulis sajikan dalam link video Google Drive berikut. https://drive.google.com/file/d/1bGfFoPzYE6SGKkqlRn36OcamJo5yqMre/view?usp=drive_link.
2. Alur program yang dijelaskan pada bab 2 didasar oleh alur flowchart program yang telah dibuat sebelumnya. Untuk lebih penjelasan lebih detail dan proses running program, penulis sajikan dalam video yang dikemas pada link Google Drive berikut. https://drive.google.com/file/d/1bGfFoPzYE6SGKkqlRn36OcamJo5yqMre/view?usp=drive_link

LAMPIRAN

Berikut adalah folder Goole drive yang berisi video penjelasan, flowchart, graph yang digunakan, dan source code program dalam bahasa C.
https://drive.google.com/drive/folders/1NQ0ROpmfK8WTMxEyaR-iUZnMCgb3KS9O?usp=sharing