**DOKUMENTASI PROGRAM LAST & FINAL PROJECT**

****

Dosen Pembimbing:

Dr. Ricky Eka Putra, S.Kom., M.Kom.

Disusun oleh:

Kelompok 7

Adriano Emmanuel - 23051204082

Cornelius Louis Nathan - 230512304085

Yakobus Octavianus Putra - 23051204332

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**UNIVERSITAS NEGERI SURABAYA**

**2024**

**KATA PENGANTAR**

Puji syukur kami haturkan kepada Tuhan Yang Maha Esa sehingga atas rahmat-Nya. Kami dapat menyelesaikan laporan dokumentasi tugas struktur data yang berjudul “Dokumentasi Program *Last* & *Final Project*” sebagai tugas struktur data tepat pada waktunya.

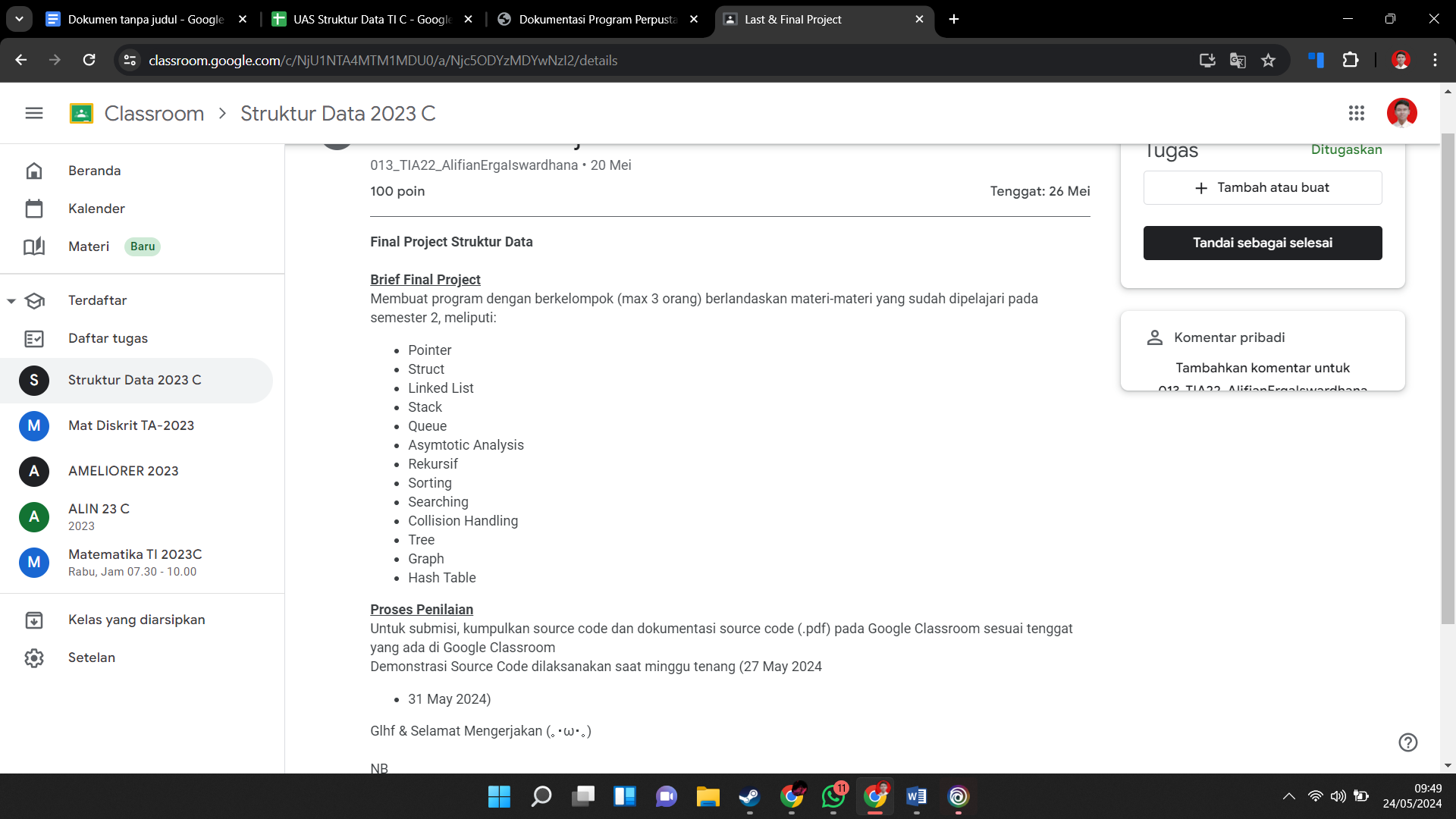
Adapun tujuan penulisan laporan ini adalah untuk mengetahui bagaimana penerapan dan pengimplementasian dari materi-materi struktur data yang diinginkan. Pembuatan laporan ini sekaligus menjadi tugas yang mengisi nilai dari Mata Kuliah Struktur Data.

Penyusunan laporan ini tidak lepas dari bantuan dan dukungan. Sehingga, kami menyampaikan terima kasih kepada pihak yang telah mendukung pembuatan laporan ini. Terutama kepada Bapak Ricky Eka Putra, S.Kom., M.Kom., selaku Dosen Pembimbing Mata Kuliah Struktur Data, atas petunjuk, didikan, dan arahan yang sangat membantu dalam penyusunan laporan ini.

Kami menyadari bahwasanya penulisan dokumentasi ini jauh dari kata sempurna, sehingga kami menerima saran dan kritik yang membangun dari pembaca guna untuk perbaikan dan perkembangan di masa mendatang. Semoga penyusunan laporan ini juga berguna bagi para pembaca dalam memahami bagaimana penerapan dan pengimplementasian dari materi-materi struktur data yang diinginkan.

Surabaya, 22 Mei 2024

Penulis

****

Untuk mendukung pengaplikasian 13 materi tersebut, program kami hanya menggunakkan 2 library, yaitu



**#include <iostream>** : Library yang memungkinkan penggunaan objek untuk operasi input dan output.

**#include <string>** : Library yang memungkinkan kita untuk memanipulasi string.

**using namespace std** : agar kita bisa menggunakan semua elemen dalam namespace std tanpa perlu menulis std:: sebelum setiap elemen (misalnya, cout daripada std::cout).

Program kami juga memiliki 5 buah struktur data yaitu, *linked list, tree, stack, graph* dan *table hashing*.

* *Linked list* digunakan untuk menyimpan data kamera dan data lensa
* *Tree* digunakan untuk menyimpan nama *member* yang ingin melakukan penyewaan kamera atau lensa
* *Stack* digunakan untuk menyimpan data fotografer yang dapat disewa jasanya
* *Graph* digunakan untuk menyimpan data tempat spot foto yang dapat diakses di Surabaya dan menghubungkan antara satu tempat dengan tempat lain yang berdekatan
* *Table hashing* digunakan untuk menyimpan data peminjaman kamera atau lensa

*Linked List* memiliki anggota *struct* sebagai berikut:

struct kamera {

string merek;

string tipe;

int harga;

struct kamera \*next;

struct kamera \*prev;

};

struct lensa {

string merek;

string tipe;

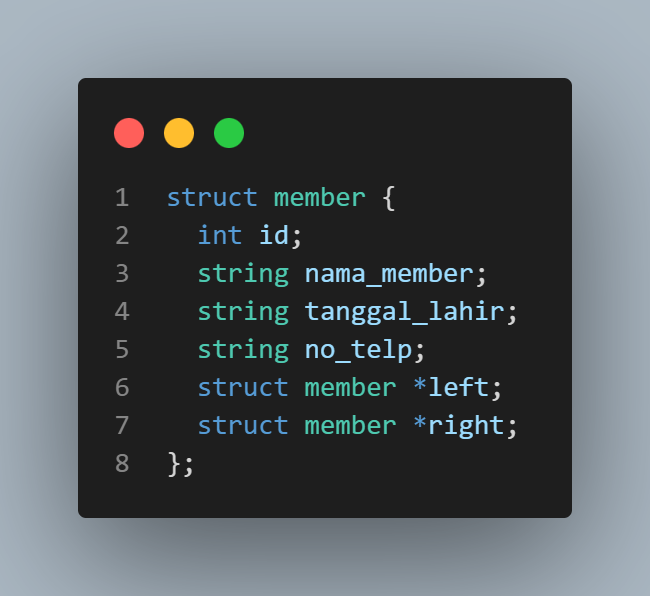
int harga;

struct lensa \*next;

struct lensa \*prev;

};

*Tree* memiliki anggota *struct* sebagai berikut:



*Stack* memiliki anggota *struct* sebagai berikut:

// Definisi struktur fotografer

struct fotografer {

string nama;

string umur;

char kelamin;

string pengalaman;

int harga;

fotografer \*left;

fotografer \*right;

};

// Definisi struktur Node Stack

struct StackNode {

fotografer data;

StackNode \*next;

};

// Definisi struktur Stack

struct Stack {

StackNode \*top;

};

*Graph* memiliki anggota *struct* sebagai berikut:

struct destinasi {

string tempat;

destinasi\* next;

};

struct graph {

int banyak\_tempat;

string tempat[10];

destinasi\* adj[10];

};

*Table hashing* memiliki anggota *struct* sebagai berikut:

struct penyewaan\_barang{

int id;

string nama;

kamera \*kamera\_yang\_disewa;

lensa \*lensa\_yang\_disewa;

int harga\_total;

int hari;

penyewaan\_barang \*next;

};

Terdapat data kamera dan lensa sebagai berikut

input\_awal\_kamera("Canon", "EOS M10", 80000);

input\_awal\_kamera("Fujifilm", "X A10", 90000);

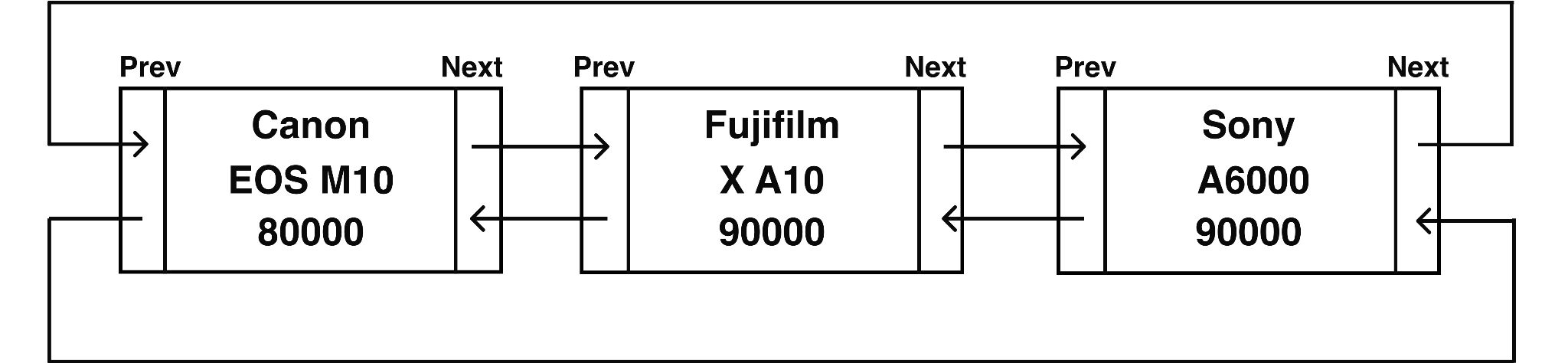
input\_awal\_kamera("Sony", "A6000", 120000);

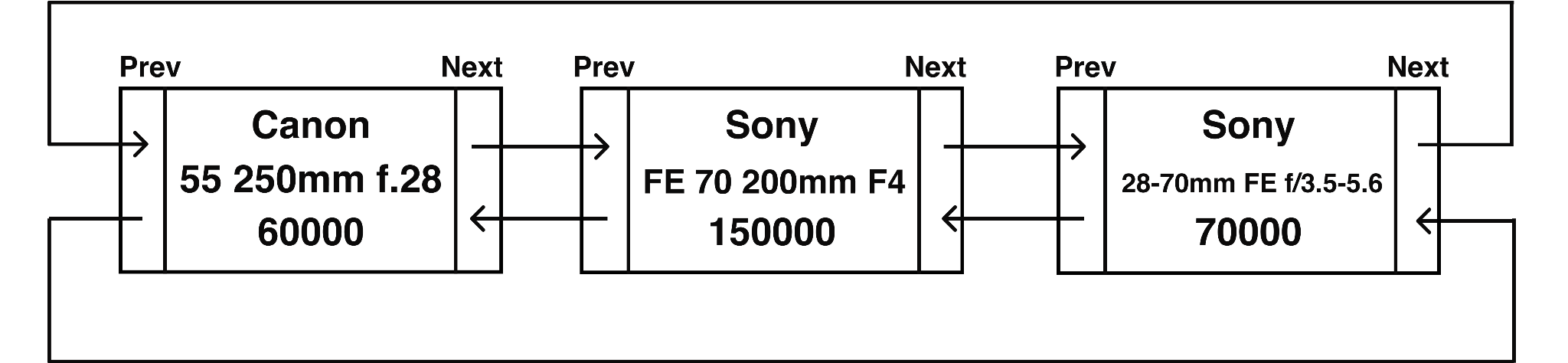
input\_awal\_lensa("Canon", "55 250mm f.28", 60000);

input\_awal\_lensa("Sony", "FE 70 200mm F4", 150000);

input\_awal\_lensa("Sony", "28-70mm FE f/3.5-5.6", 70000);

1. Berikut adalah ilustrasi *linked list* kamera dan lensa





Terdapat data member sebagai berikut:

Id Member : 133299

Nama : Muhammad Hasan

Tanggal Lahir : 04-07-1999

No Telp : 085246559899

Id Member : 193251

Nama : Steven Michael

Tanggal Lahir : 09-05-2000

No Telp : 084798215051

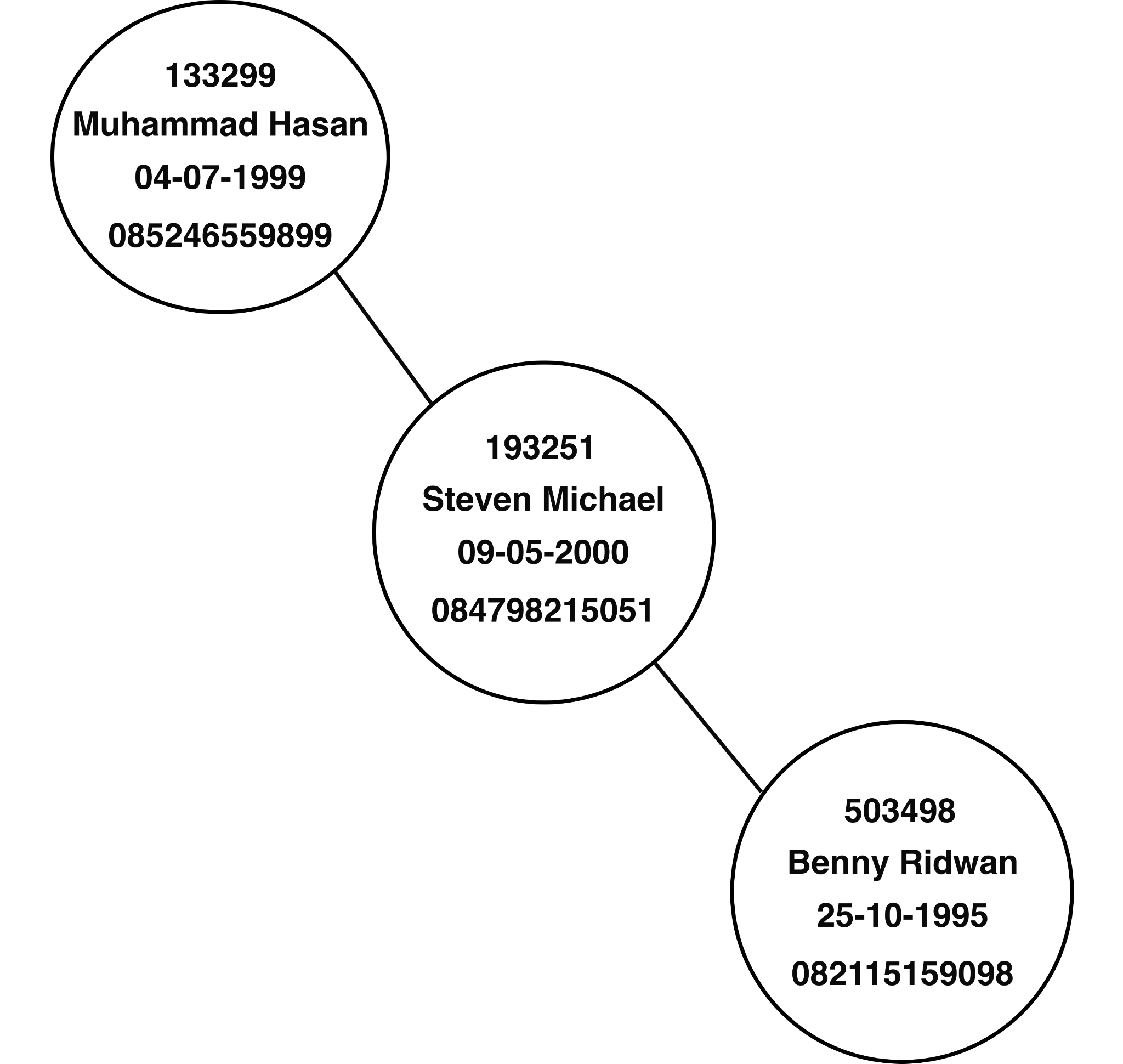
Id Member : 503498

Nama : Benny Ridwan

Tanggal Lahir : 25-10-1995

No Telp : 082115159098

1. Berikut adalah ilustrasi *tree* member (posisi *root tree* dipengaruhi oleh kode member)



Terdapat data fotografer sebagai berikut:

input\_awal\_fotografer(stack, "Yakobus", "19", 'L', "5 tahun", 200000);

input\_awal\_fotografer(stack, "Louis", "20", 'L', "3 tahun", 150000);

input\_awal\_fotografer(stack, "Adriano", "21", 'L', "8 tahun", 200000);

1. Berikut adalah ilustrasi *stack* fotografer (Data paling atas diletakkan paling bawah, karena berbentuk *stack*)



Terdapat data spot foto yang recommended di Surabaya

cout << "Spot foto yang recommended di Surabaya\n";

cout << "1. Kampung Arab , Surabaya Utara\n";

cout << "2. Kampung Bulak , Surabaya Utara\n";

cout << "3. Klenteng Sanggar Agung , Surabaya Utara\n";

cout << "4. Patung Buddha 4 Rupa , Surabaya Utara\n";

cout << "5. Museum House of Sampoerna, Surabaya Utara\n";

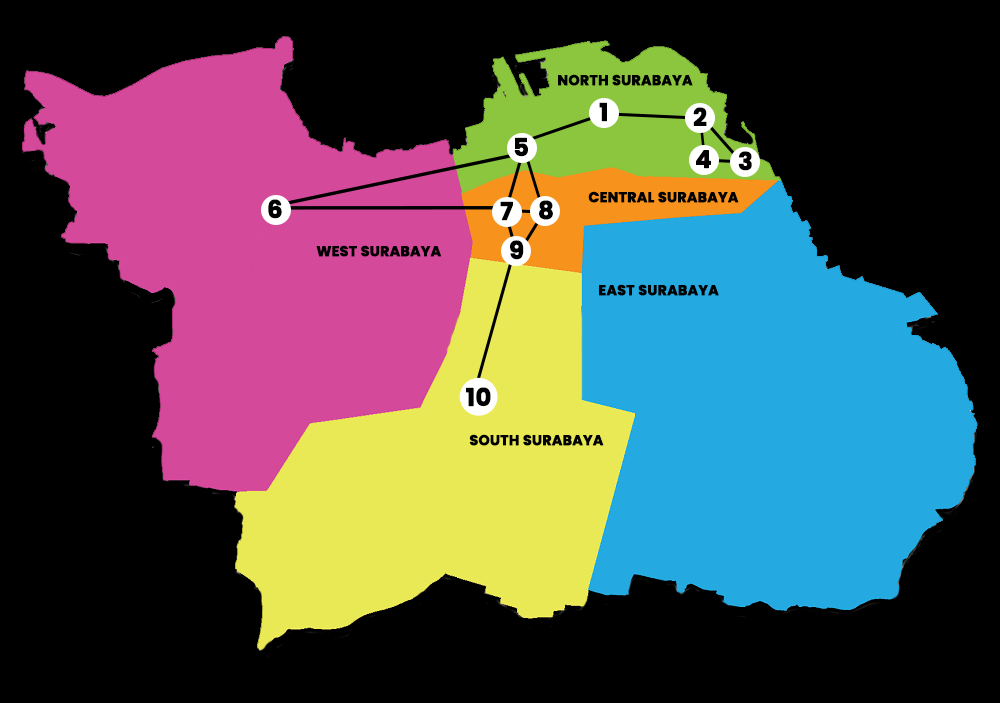
cout << "6. G-Walk Citraland , Surabaya Barat\n";

cout << "7. Tugu Pahlawan , Surabaya Pusat\n";

cout << "8. Museum Pahlawan , Surabaya Pusat\n";

cout << "9. Jalan Tunjungan, Surabaya Pusat\n";

cout << "10. Kebun Binatang Surabaya, Surabaya Selatan\n\n";

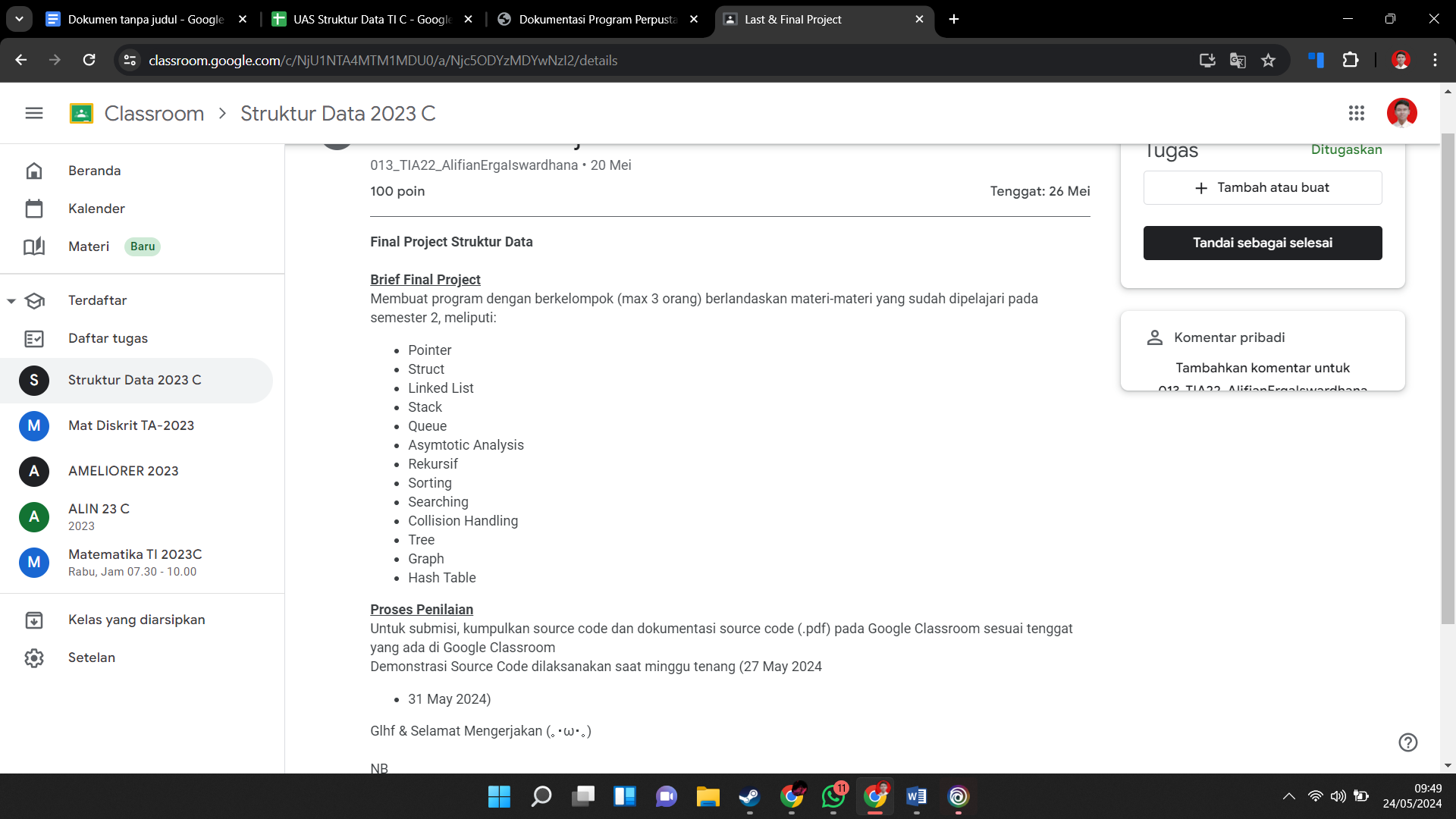
1. Berikut adalah ilustrasi *graph* antara 1 tempat dengan tempat yang lain (penomoran disesuaikan dengan urutan lokasi pada daftar diatas)
2. Berikut adalah ilustrasi *table hash* yang digunakan untuk menyimpan data peminjaman kamera dan lensa

| No | ID Member | Nama | Kamera/Lensa yang disewa | Lama Sewa | Harga Total |
| --- | --- | --- | --- | --- | --- |
| 0 | … |  |  |  |  |
| 1 | 193251 | Steven Michael | Canon 55 250mm f.28 | 2 | Rp 120.000 |
| … |  |  |  |  |  |
| 8 | 503498 | Benny Ridwan | Sony A6000 | 3 | Rp 270.000 |
| 9 | 133299 | Muhammad Hasan | Sony FE 70 200mm F4 | 2 | Rp 300.000 |

\*193251 % 10 = 1

\*\*503498 % 10 = 8

\*\*\*133299 % 10 = 9

****

Pengaplikasian 13 Materi :

1. *Pointer*

*Pointer* adalah variabel yang digunakan untuk menyimpan atau menunjukkan alamat dari suatu memori. Banyak fungsi dari program kami yang menggunakkan *pointer*.

1. Fungsi BFS (*Breadth First Search*) bertujuan untuk melakukan penelusuran *vertex*-*vertex* (untuk penerapan *graph*) menggunakkan pointer

void BFS(graph &tour, string start) {

1. Untuk mengakses daftar *adjacency* (ketetanggaan antar vertex) menggunakkan pointer

destinasi \*adj\_list = tour.adj[index\_sekarang];

1. Setiap *vertex* dalam *graph* memiliki *pointer Next* untuk menunjuk *vertex* lain

adj\_list = adj\_list->next;

1. *Pointer* digunakan untuk mengakses anggota *struct*

if (tour.tempat[i] == adj\_list->tempat) {

adj\_index = i;

break;

}

1. Fungsi admin\_input\_kamera bertujuan untuk menginputkan data kamera ke dalam *linked list* juga menggunakkan pointer

void admin\_input\_kamera() {

1. Untuk membuat *node* baru, maka dibuatlah *pointer* dinamis yang memiliki anggota sama seperti *struct* kamera.

kamera \*newKamera = new kamera;

1. Untuk menyimpan data kamera baru, maka pengguna akan menginputkan data mengenai kamera tersebut melalui *pointer* yang akan disimpan dalam *struct*.

string merekkamera;

kamera \*newKamera = new kamera;

cout << "Masukkan merek kamera\t: ";

getline(cin, merekkamera);

cout << "Masukkan tipe kamera\t: ";

getline(cin, newKamera->tipe);

cout << "Masukkan harga kamera\t: ";

cin >> newKamera->harga;

toUpperCase(merekkamera);

newKamera->merek = merekkamera;

cin.ignore();

newKamera->next = newKamera;

newKamera->prev = newKamera;

1. Untuk menghubungkan *node* baru dengan *node* yang sudah ada di dalam *linked list*, maka diaturlah *pointer* penghubung antar *node*

newKamera->next = newKamera;

newKamera->prev = newKamera;

if (head\_kamera == nullptr) {

head\_kamera = newKamera;

tail\_kamera = newKamera;

head\_kamera->next = head\_kamera;

head\_kamera->prev = head\_kamera;

tail\_kamera->next = tail\_kamera;

tail\_kamera->prev = tail\_kamera;

} else {

newKamera->next = head\_kamera;

head\_kamera->prev = newKamera;

head\_kamera = newKamera;

head\_kamera->prev = tail\_kamera;

tail\_kamera->next = head\_kamera;

}

Fungsi admin\_input\_lensa juga menggunakkan penerapan yang sama, hanya saja berbeda *linked list*.

1. Fungsi *inorder* bertujuan untuk menampilkan semua data *member* dalam bentuk *tree* juga menggunakkan *pointer*.

void inorder(member \*root) {

if (root != nullptr) {

inorder(root->left);

cout << "\nID Member\t: " << root->id << endl;

cout << "Nama\t\t: " << root->nama\_member << endl;

cout << "Tanggal Lahir\t: " << root->tanggal\_lahir << endl;

cout << "No Telp\t\t: " << root->no\_telp << endl;

inorder(root->right);

}

}

1. Parameter dari fungsi *inorder* adalah *pointer* dari *tree member*
2. Pengecekan *pointer* untuk menentukan rekursif
3. Pengaksesan data anggota seperti ID member, nama, tanggal lahir, dan nomor telepon melalui *pointer*
4. Fungsi user\_cari\_kamera bertujuan agar pengguna dapat mencari tipe kamera yang diinginkan juga menggunakkan *pointer*

void user\_cari\_kamera() {

1. Pengecekan *head\_kamera* untuk memastikan apakah terdapat *linked list* kosong atau tidak

if (head\_kamera == nullptr)

1. Deklarasi *temp* untuk melakukan *sequential search* terhadap kamera yang diinginkan

kamera \*temp = head\_kamera;

while (temp != nullptr && temp->merek != merek) {

temp = temp->next;

}

1. Pengecekan *temp* untuk menentukan apakah kamera yang dicari terdapat dalam *linked list* atau tidak

if (temp == nullptr) {

cout << "Kamera tidak ditemukan\n";

} else {

cout << "Kamera ditemukan: " << temp->merek << ", " << temp->tipe << ", "

<< temp->harga << endl;

1. Fungsi initStack bertujuan untuk melakukan inisiasi *stack* juga menggunakkan *pointer*

void initStack(Stack &stack) { stack.top = nullptr; }

Masih banyak lagi fungsi-fungsi lain yang menggunakkan penerapan *pointer*.

1. *Struct*

*Struct* adalah pengelompokan variabel-variabel yang bernaung dalam satu tipe data yang sama. Biasanya untuk menyimpan suatu data yang tidak bisa berdiri sendiri. Pada program kami, kami menggunakkan 9 *struct* untuk menyimpan data-data yang akan membantu proses pengimplementasian struktur data yang lain

struct kamera { // untuk menyimpan data kamera dalam penerapan linked list

string merek; // merek kamera

string tipe; // tipe kamera

int harga; // harga sewa per hari

struct kamera \*next; // pointer next linked list

struct kamera \*prev; // pointer prev linked list

};

struct lensa { // untuk menyimpan data lensa dalam penerapan linked list

string merek; // merek kamera

string tipe; // tipe kamera

int harga; // harga sewa per hari

struct lensa \*next; // pointer next linked list

struct lensa \*prev; // pointer prev linked list

};

struct member { // untuk menyimpan data anggota dalam penerapan tree

int id; // id member

string nama\_member; // nama member

string tanggal\_lahir; // tanggal lahir member

string no\_telp; // nomor telepon member

struct member \*left; // pointer left tree

struct member \*right; // pointer right tree

};

// Definisi struktur fotografer yang akan disimpan dalam penerapan stack

struct fotografer {

string nama; // nama fotografer

string umur; // umur fotografer

char kelamin; // jenis kelamin fotografer

string pengalaman; // lama pengalaman fotografer

int harga; // harga sewa jasa per sesi

fotografer \*left; // pointer left untuk pengecekan fotografer atas

fotografer \*right; // pointer right untuk pengecekan fotografer bawah

};

// Definisi struktur Node Stack

struct StackNode {

fotografer data; // data fotografer

StackNode \*next; // next

};

// Definisi struktur Stack

struct Stack {

StackNode \*top; // data fotografer paling atas

};

struct destinasi { // untuk menyimpan data vertex dalam penerapan graph

string tempat; // nama tempat atau spot foto

destinasi\* next; // pointer next

};

struct graph { // untuk menyimpan data graph

int banyak\_tempat; // banyaknya spot foto

string tempat[10]; // nama tempat

destinasi\* adj[10]; // banyaknya hubungan antara 1 spot foto dengan lainnya

};

struct penyewaan\_barang{ // untuk menyimpan data penyewaan barang dalam penerapan table hash

int id; // id member

string nama; // nama member

kamera \*kamera\_yang\_disewa; // data kamera yang disewa (apabila ada)

lensa \*lensa\_yang\_disewa; // lensa kamera yang disewa (apabila ada)

int harga\_total; // harga total dari penyewaan kamera atau lensa

int hari; // lama penyewaan kamera atau lensa

penyewaan\_barang \*next; // pointer next untuk chaining table hash

};

1. *Linked List*

*Linked list* adalah struktur data yang terdiri atas kumpulan node yang tersusun secara sekuensial dan saling terhubung satu dengan yang lain. Pada program kami, *linked list* digunakan untuk menyimpan data kamera dan data lensa.

struct kamera { // untuk menyimpan data kamera dalam penerapan linked list

string merek; // merek kamera

string tipe; // tipe kamera

int harga; // harga sewa per hari

struct kamera \*next; // pointer next linked list

struct kamera \*prev; // pointer prev linked list

};

struct lensa { // untuk menyimpan data lensa dalam penerapan linked list

string merek; // merek kamera

string tipe; // tipe kamera

int harga; // harga sewa per hari

struct lensa \*next; // pointer next linked list

struct lensa \*prev; // pointer prev linked list

};

Kami menciptakan 10 fungsi untuk menambah, menghapus, mencari data dan mengakses *linked list*. Fungsi yang digunakan banyak karena terdapat 2 *linked list* yaitu untuk menyimpan data kamera dan data lensa secara terpisah, selain itu fungsi yang digunakan juga masih dibedakan berdasarkan admin dan member.

1. Fungsi void toUpperCase untuk merubah sebuah kalimat menjadi huruf kapital semua.
2. Fungsi *void* admin\_input\_kamera untuk menambahkan data kamera dari dalam *linked list.*
3. Fungsi *void* admin\_input\_lensa untuk menambahkan data lensa dari dalam *linked list.*
4. Fungsi *void* admin\_hapus\_kamera untuk menghapus data kamera dari dalam *linked list.*
5. Fungsi *void* admin\_hapus\_lensa untuk menghapus data lensa dari dalam *linked list.*
6. Fungsi *void* list\_kamera untuk menampilkan data kamera dari dalam *linked list.*
7. Fungsi *void* list\_lensa untuk menampilkan data lensa dari dalam *linked list.*
8. Fungsi *void* user\_cari\_kamera untuk mencari data kamera dari dalam *linked list.*
9. Fungsi *void* user\_cari\_lensa untuk mencari data lensa dari dalam *linked list.*
10. Fungsi *void* input\_awal\_kamera untuk menambahkan data kamera ke dalam *linked list* secara otomatis pada saat awal program berjalan.
11. Fungsi *void* input\_awal\_lensa untuk menambahkan data kamera ke dalam *linked list* secara otomatis pada saat awal program berjalan.

void toUpperCase(string &str) {

for (char &c : str) {

c = toupper(static\_cast<unsigned char>(c));

}

}

// Fungsi untuk menambahkan kamera ke linked list

void admin\_input\_kamera() {

clearScreen();

kamera\* current = head\_kamera;

kamera \*newKamera = new kamera();

cout << "Masukkan merek kamera\t: ";

getline(cin, newKamera->merek);

cout << "Masukkan tipe kamera\t: ";

getline(cin, newKamera->tipe);

cout << "Masukkan harga kamera\t: ";

cin >> newKamera->harga;

cin.ignore();

newKamera->next = newKamera;

newKamera->prev = newKamera;

if (head\_kamera == nullptr) {

head\_kamera = newKamera;

tail\_kamera = newKamera;

head\_kamera->next = head\_kamera;

head\_kamera->prev = head\_kamera;

tail\_kamera->next = tail\_kamera;

tail\_kamera->prev = tail\_kamera;

} else {

newKamera->next = head\_kamera;

head\_kamera->prev = newKamera;

head\_kamera = newKamera;

head\_kamera->prev = tail\_kamera;

tail\_kamera->next = head\_kamera;

}

cout << "\nKamera berhasil ditambahkan\n";

}

// Fungsi untuk menambahkan lensa ke linked list

void admin\_input\_lensa() {

clearScreen();

string mereklensa;

lensa \*newLensa = new lensa();

cout << "Masukkan merek lensa\t: ";

getline(cin, mereklensa);

cout << "Masukkan tipe lensa\t: ";

getline(cin, newLensa->tipe);

cout << "Masukkan harga lensa\t: ";

cin >> newLensa->harga;

toUpperCase(mereklensa);

newLensa->merek = mereklensa;

cin.ignore();

newLensa->next = newLensa;

newLensa->prev = newLensa;

if (head\_lensa == nullptr) {

head\_lensa = newLensa;

tail\_lensa = newLensa;

head\_lensa->next = head\_lensa;

head\_lensa->prev = head\_lensa;

tail\_lensa->next = tail\_lensa;

tail\_lensa->prev = tail\_lensa;

} else {

newLensa->next = head\_lensa;

head\_lensa->prev = newLensa;

head\_lensa = newLensa;

head\_lensa->prev = tail\_lensa;

tail\_lensa->next = head\_lensa;

}

cout << "\nLensa berhasil ditambahkan\n";

}

// Fungsi untuk menghapus kamera dari linked list

void admin\_hapus\_kamera() {

clearScreen();

if (head\_kamera == nullptr) {

cout << "\nTidak ada kamera untuk dihapus\n";

return;

}

string tipe;

cout << "Masukkan tipe kamera yang ingin dihapus: ";

getline(cin,tipe);

bool ketemu = false;

kamera \*temp = head\_kamera;

kamera \*prev\_node = tail\_kamera;

do{

if(temp->tipe == tipe){

ketemu = true;

if (temp == head\_kamera) {

head\_kamera = temp->next;

tail\_kamera->next = head\_kamera;

delete temp;

}

else if (temp == tail\_kamera) {

tail\_kamera = prev\_node;

tail\_kamera->next = head\_kamera;

delete temp;

}

else {

prev\_node->next = temp->next;

delete temp;

}

}

prev\_node = temp;

temp = temp->next;

} while (temp != head\_kamera);

if(ketemu == true){

cout << "Kamera berhasil dihapus dari database\n";

} else {

cout << "Lensa dengan tipe " << tipe << " tidak ditemukan\n";

}

}

// Fungsi untuk menghapus lensa dari linked list

void admin\_hapus\_lensa() {

clearScreen();

if (head\_lensa == nullptr) {

cout << "Tidak ada lensa untuk dihapus\n";

return;

}

string tipe;

cout << "Masukkan tipe lensa yang ingin dihapus: ";

getline(cin,tipe);

bool ketemu = false;

lensa \*temp = head\_lensa;

lensa \*prev\_node = tail\_lensa;

do{

if(temp->tipe == tipe){

ketemu = true;

if (temp == head\_lensa) {

head\_lensa = temp->next;

tail\_lensa->next = head\_lensa;

delete temp;

}

else if (temp == tail\_lensa) {

tail\_lensa = prev\_node;

tail\_lensa->next = head\_lensa;

delete temp;

}

else {

prev\_node->next = temp->next;

delete temp;

}

}

prev\_node = temp;

temp = temp->next;

} while (temp != head\_lensa);

if(ketemu == true){

cout << "Lensa berhasil dihapus dari database\n";

} else {

cout << "Lensa dengan tipe " << tipe << " tidak ditemukan\n";

}

}

// Fungsi untuk menampilkan daftar kamera

void list\_kamera() {

if (head\_kamera == nullptr) {

cout << "\nTidak ada kamera yang tersedia\n\n";

return;

}

int pilih;

kamera \*temp = head\_kamera;

do {

clearScreen();

welcome();

cout << "Merek Kamera\t\t: " << temp->merek << endl;

cout << "Tipe Kamera\t\t: " << temp->tipe << endl;

cout << "Harga Sewa (per hari)\t: " << temp->harga << endl;

cout << "1. Lihat Kamera Selanjutnya\n";

cout << "2. Lihat Kamera Sebelumnya\n";

cout << "0. Exit\n> ";

cin >> pilih;

switch (pilih) {

case 1: {

temp = temp->next;

} break;

case 2: {

temp = temp->prev;

} break;

case 0:

break;

default:

cout << "Pilihan Tidak Valid!";

break;

}

cout << "3. Kembali\n";

} while (pilih != 0);

}

// Fungsi untuk menampilkan daftar lensa

void list\_lensa() {

if (head\_lensa == nullptr) {

cout << "\nTidak ada lensa yang tersedia\n\n";

return;

}

int pilih;

lensa \*temp = head\_lensa;

do {

clearScreen();

welcome();

cout << "Merek Lensa\t\t: " << temp->merek << endl;

cout << "Tipe Lensa\t\t: " << temp->tipe << endl;

cout << "Harga Sewa (per hari)\t: " << temp->harga << endl;

cout << "1. Lihat Lensa Selanjutnya\n";

cout << "2. Lihat Lensa Sebelumnya\n";

cout << "0. Exit\n> ";

cin >> pilih;

switch (pilih) {

case 1:{

temp = temp->next;

} break;

case 2:{

temp = temp->prev;

} break;

case 0:

break;

default:

cout << "Pilihan Tidak Valid!";

break;

}

} while (pilih != 0);

}

// Fungsi untuk mencari kamera

void user\_cari\_kamera() {

if (head\_kamera == nullptr) {

cout << "Tidak ada kamera yang tersedia\n";

return;

} else {

clearScreen();

string tipe\_yang\_dicari;

cout << "Masukkan tipe kamera yang ingin dicari: ";

getline(cin,tipe\_yang\_dicari);

kamera \*temp = head\_kamera;

while (temp->next != head\_kamera && temp->tipe != tipe\_yang\_dicari) {

temp = temp->next;

}

if (temp->next == head\_kamera) {

cout << "\nKamera dengan tipe " << tipe\_yang\_dicari << "tidak ditemukan\n";

} else {

cout << "\nKamera ditemukan: " << temp->merek << " " << temp->tipe << " , dengan harga sewa per hari "

<< temp->harga << endl << endl;

}

}

}

// Fungsi untuk mencari lensa

void user\_cari\_lensa() {

if (head\_lensa == nullptr) {

cout << "Tidak ada lensa yang tersedia\n";

return;

} else {

clearScreen();

string tipe\_yang\_dicari;

cout << "Masukkan tipe lensa yang ingin dicari: ";

getline(cin,tipe\_yang\_dicari);

lensa \*temp = head\_lensa;

while (temp->next != head\_lensa && temp->tipe != tipe\_yang\_dicari) {

temp = temp->next;

}

if (temp->next == head\_lensa) {

cout << "\nLensa dengan tipe " << tipe\_yang\_dicari << "tidak ditemukan\n";

} else {

cout << "\nLensa ditemukan: " << temp->merek << " " << temp->tipe << " , dengan harga sewa per hari "

<< temp->harga << endl << endl;

}

}

}

// Fungsi untuk memasukkan data kamera pada awal program dijalankan

void input\_awal\_kamera(string brand, string type, int price){

kamera \*newKamera = new kamera;

newKamera->merek = brand;

newKamera->tipe = type;

newKamera->harga = price;

newKamera->next = newKamera;

newKamera->prev = newKamera;

if (head\_kamera == nullptr) {

head\_kamera = newKamera;

tail\_kamera = newKamera;

head\_kamera->next = head\_kamera;

head\_kamera->prev = head\_kamera;

tail\_kamera->next = tail\_kamera;

tail\_kamera->prev = tail\_kamera;

} else {

newKamera->next = head\_kamera;

head\_kamera->prev = newKamera;

head\_kamera = newKamera;

head\_kamera->prev = tail\_kamera;

tail\_kamera->next = head\_kamera;

}

}

// Fungsi untuk memasukkan data lensa pada awal program dijalankan

void input\_awal\_lensa(string brand, string type, int price){

lensa \*newLensa = new lensa();

newLensa->merek = brand;

newLensa->tipe = type;

newLensa->harga = price;

newLensa->next = newLensa;

newLensa->prev = newLensa;

if (head\_lensa == nullptr) {

head\_lensa = newLensa;

tail\_lensa = newLensa;

head\_lensa->next = head\_lensa;

head\_lensa->prev = head\_lensa;

tail\_lensa->next = tail\_lensa;

tail\_lensa->prev = tail\_lensa;

} else {

newLensa->next = head\_lensa;

head\_lensa->prev = newLensa;

head\_lensa = newLensa;

head\_lensa->prev = tail\_lensa;

tail\_lensa->next = head\_lensa;

}

}

Berikut adalah contoh *output* yang diberikan fungsi list\_kamera yang dapat diakses oleh admin

================================

Welcome to Cam Store

================================

Merek Kamera : Sony

Tipe Kamera : A6000

Harga Sewa (per hari) : 120000

1. Lihat Kamera Selanjutnya

2. Lihat Kamera Sebelumnya

0. Exit

>

Berikut adalah contoh *output* yang diberikan oleh fungsi user\_pesan\_kamera yang dapat diakses oleh user

================================

User

================================

Merek Kamera : Fujifilm

Tipe Kamera : X A10

Harga Sewa (per hari) : 90000

1. Lihat Kamera Selanjutnya

2. Lihat Kamera Sebelumnya

3. Pesan Kamera

0. Exit

>

Berikut adalah contoh *output* yang diberikan oleh fungsi user\_cari\_lensa yang dapat diakses oleh user

Masukkan tipe lensa yang ingin dicari: FE 70 200mm F4

Lensa ditemukan: Sony FE 70 200mm F4 , dengan harga sewa per hari 150000

Klik enter untuk melanjutkan

dan apabila lensa yang dicari tidak ditemukan, maka berikut adalah *output* nya

Masukkan tipe lensa yang ingin dicari: EF 14mm f/ 2.8 L II USM

Lensa dengan tipe EF 14mm f/ 2.8 L II USM tidak ditemukan

Klik enter untuk melanjutkan

1. *Stack*

*Stack* adalah struktur data bersusun yang menerapkan prinsip LIFO (*Last In, First Out*), yang artinya elemen terakhir yang dimasukkan adalah elemen pertama yang keluar. Pada program kami, *stack* digunakan untuk menyimpan data fotografer. Sehingga apabila seseorang ingin menyewa jasa fotografer, maka *stack* fotografer paling atas yang akan diambil dahulu.

struct member { // untuk menyimpan data anggota dalam penerapan tree

int id; // id member

string nama\_member; // nama member

string tanggal\_lahir; // tanggal lahir member

string no\_telp; // nomor telepon member

struct member \*left; // pointer left tree

struct member \*right; // pointer right tree

};

Kami menciptakan 8 fungsi untuk menginisiasi *stack,* memasukkan data fotografer (*push*) ke dalam *stack*, mengambil data fotografer *(pop*) dari dalam *stack* dan menampilkan semua data fotografer.

1. Fungsi *void* initStack untuk melakukan inisiasi terhadap *stack*.
2. Fungsi *void push* untuk memasukkan data fotografer baru ke dalam *stack.*
3. Fungsi *fotografer pop* untuk mengambil data fotografer dari dalam *stack.*
4. Fungsi *void* *deleteByName* untuk menghapus data fotografer dari dalam *stack.*
5. Fungsi *void* displayFotografer untuk menampilkan data fotografer dari dalam *stack.*
6. Fungsi *void* displayStackInteractive untuk menampilkan data fotografer dari dalam *stack* oleh admin.
7. Fungsi *void* pesan\_fotografer untuk melakukan sewa jasa fotografer dan melakukan proses *pop* dari dalam *stack*.
8. Fungsi *void* input\_awal\_fotografer untuk menambahkan data kamera ke dalam *linked list* secara otomatis pada saat awal program berjalan.

// Fungsi untuk inisialisasi stack

void initStack(Stack &stack) { stack.top = nullptr; }

// Fungsi untuk push elemen ke stack dengan input dari pengguna

void push(Stack &stack) {

clearScreen();

StackNode \*newNode = new StackNode();

cout << "Masukkan nama fotografer: ";

cin.ignore();

getline(cin, newNode->data.nama);

cout << "Masukkan umur fotografer: ";

getline(cin, newNode->data.umur);

cout << "Masukkan jenis kelamin fotografer (L/P): ";

cin >> newNode->data.kelamin;

cin.ignore(); // Ignore newline character left in buffer

cout << "Masukkan pengalaman fotografer (.. tahun/... bulan): ";

getline(cin, newNode->data.pengalaman);

cout << "Masukkan harga jasa (per sesi): ";

cin >> newNode->data.harga;

cin.ignore();

newNode->next = stack.top;

stack.top = newNode;

cout << "Fotografer " << newNode->data.nama << " berhasil ditambahkan ke stack\n";

}

// Fungsi untuk pop elemen dari stack

fotografer pop(Stack &stack) {

if (stack.top == nullptr) {

cout << "Stack kosong\n";

return fotografer();

}

StackNode \*temp = stack.top;

stack.top = stack.top->next;

fotografer item = temp->data;

delete temp;

return item;

}

// Fungsi untuk menghapus fotografer berdasarkan nama

void deleteByName(Stack &stack, const string &name) {

StackNode \*current = stack.top;

StackNode \*previous = nullptr;

while (current != nullptr) {

if (current->data.nama == name) {

if (previous == nullptr) {

stack.top = current->next;

} else {

previous->next = current->next;

}

delete current;

cout << "Fotografer dengan nama " << name << " telah dihapus\n";

return;

}

previous = current;

current = current->next;

}

cout << "\nFotografer dengan nama " << name << " tidak ditemukan\n";

}

// Fungsi untuk menampilkan fotografer satu per satu

void displayFotografer(StackNode \*node) {

clearScreen();

if (node != nullptr) {

cout << "Nama\t: " << node->data.nama << endl

<< "Umur\t: " << node->data.umur << endl

<< "Kelamin\t: " << node->data.kelamin << endl

<< "Pengalaman: " << node->data.pengalaman << endl

<< "Harga jasa (per sesi): " << node->data.harga << endl;

}

}

void displayStackInteractive(Stack &stack) {

StackNode \*current = stack.top;

if (current == nullptr) {

cout << "Stack kosong\n";

return;

}

// Menghitung total node untuk memudahkan navigasi circular

int totalNode = 0;

StackNode \*tempNode = stack.top;

while (tempNode != nullptr) {

totalNode++;

tempNode = tempNode->next;

}

int currentIndex = 0;

int choice;

do {

clearScreen();

displayFotografer(current);

cout << "1. Lihat Fotografer Selanjutnya\n";

cout << "2. Lihat Fotografer Sebelumnya\n";

cout << "0. Exit\n> ";

cin >> choice;

if (choice == 1) {

if (current->next != nullptr) {

current = current->next;

currentIndex++;

} else {

// Kembali ke elemen pertama jika mencapai akhir

current = stack.top;

currentIndex = 0;

}

} else if (choice == 2) {

if (currentIndex == 0) {

// Pergi ke elemen terakhir jika di elemen pertama

StackNode \*lastNode = stack.top;

while (lastNode->next != nullptr) {

lastNode = lastNode->next;

}

current = lastNode;

currentIndex = totalNode - 1;

} else {

// Pergi ke elemen sebelumnya

StackNode \*prevNode = stack.top;

for (int i = 0; i < currentIndex - 1; i++) {

prevNode = prevNode->next;

}

current = prevNode;

currentIndex--;

}

}

} while (choice != 0);

clearScreen();

}

void pesan\_fotografer (Stack &stack) { / /pesan fotografer

StackNode \*current = stack.top;

StackNode \*previous = nullptr;

if (current == nullptr) {

cout << "Stack kosong\n";

return;

}

int choice;

do {

displayFotografer(current);

cout << "1. Lihat Fotografer Selanjutnya\n";

cout << "2. Lihat Fotografer Sebelumnya\n";

cout << "3. Pesan Fotografer\n";

cout << "0. Exit\n> ";

cin >> choice;

if (choice == 1) {

if (current->next != nullptr) {

previous = current;

current = current->next;

} else {

cout << "Sudah di fotografer terakhir\n";

cin.get();

}

clearScreen();

} else if (choice == 2) {

if (previous != nullptr) {

current = previous;

previous = nullptr; // Reset previous untuk pencarian ulang

StackNode \*temp = stack.top;

while (temp != current && temp != nullptr) {

previous = temp;

temp = temp->next;

}

} else {

cout << "Sudah di fotografer pertama\n";

}

clearScreen();

} else if (choice == 3) {

cout << "Fotografer telah dipesan oleh anda\n";

pop(stack);

return;

} else {

cout << "Item tidak ditemukan\n";

}

} while (choice != 0);

clearScreen();

}

// Fungsi untuk memasukkan fotografer awal ke stack

void input\_awal\_fotografer(Stack &stack, string nama, string umur, char kelamin, string pengalaman, int harga) {

StackNode \*newNode = new StackNode();

newNode->data.nama = nama;

newNode->data.umur = umur;

newNode->data.kelamin = kelamin;

newNode->data.pengalaman = pengalaman;

newNode->data.harga = harga;

newNode->next = stack.top;

stack.top = newNode;

}

Berikut adalah *output* pada fungsi *pop* yang dapat diakses oleh admin

Masukkan nama fotografer: Marcos Alonso

Masukkan umur fotografer: 30

Masukkan jenis kelamin fotografer (L/P): L

Masukkan pengalaman fotografer (.. tahun/... bulan): 7 tahun

Masukkan harga jasa (per sesi): 1250000

Fotografer Marcos Alonso berhasil ditambahkan ke stack

Klik enter untuk melanjutkan

Berikut adalah *output* pada fungsi displayStackInteractive yang dapat diakses oleh admin

Nama : Adriano

Umur : 21

Kelamin : L

Pengalaman: 25 tahun

Harga jasa (per sesi): 200000

1. Lihat Fotografer Selanjutnya

2. Lihat Fotografer Sebelumnya

0. Exit

>

Berikut adalah output pada fungsi pesan\_fotografer yang dapat diakses oleh *user*

Nama : Marcos Alonso

Umur : 30

Kelamin : L

Pengalaman: 7 tahun

Harga jasa (per sesi): 1250000

1. Lihat Fotografer Selanjutnya

2. Lihat Fotografer Sebelumnya

3. Pesan Fotografer

0. Exit

> 3

Fotografer telah dipesan oleh Anda

Klik enter untuk melanjutkan

1. *Queue*

*Queue* adalah struktur data bersusun yang menerapkan prinsip FIFO (*First In, First Out*), yang artinya elemen pertama yang dimasukkan adalah elemen pertama yang keluar. Pada program kami, *queue* secara implisit digunakan untuk membantu penerapan proses *graph BFS*.

1. Dilakukan inisiasi *queue*

string queue\_tour[10];

int front = 0, rear = 0;

1. Menemukan *index start* untuk memulai proses penelusuran *graph*

int start\_index = -1;

for (int i = 0; i < tour.banyak\_tempat; i++) {

if (tour.tempat[i] == start) {

start\_index = i;

break;

}

}

1. Menandai *start* dan ditambahkan ke dalam *queue*

visited[start\_index] = true;

queue\_tour[rear++] = start;

1. Memproses *queue* dan menampilkan *output* berupa tempat yang dikunjungi

while (front < rear) {

string start\_tour = queue\_tour[front++];

cout << start\_tour << " ---> ";

1. Mengakses daftar adjacency spot foto yang dikunjungi

int index\_sekarang = -1;

for (int i = 0; i < tour.banyak\_tempat; i++) {

if (tour.tempat[i] == start\_tour) {

index\_sekarang = i;

break;

}

}

1. Menambahkan spot foto yang belum diakses ke dalam *queue* (*enqueue*)

while (adj\_list != nullptr) {

int adj\_index = -1;

for (int i = 0; i < tour.banyak\_tempat; i++) {

if (tour.tempat[i] == adj\_list->tempat) {

adj\_index = i;

break;

}

}

if (!visited[adj\_index]) {

visited[adj\_index] = true;

queue\_tour[rear++] = adj\_list->tempat;

}

adj\_list = adj\_list->next;

}

Sehingga, *queue* bekerja dengan menambahkan spot-spot foto yang terhubung dan belum dikunjungi ke dalam *queue*. Sehingga, BFS akan melakukan eksplorasi secara melebar (*Breadth*) sebelum melakukan eksplorasi ke spot selanjutnya.

1. *Asymptotic Analysis*

*Asymptotic Analysis* adalah cara untuk menghitung kinerja algoritma komputer dalam penggunaan memori dan efisiensi waktu saat suatu data diinputkan. Secara implisit, program kami menerapkan penggunaan *asymptotic analysis*. Sebagai contoh, proses *searching* kami menggunakkan metode *sequential search*, dan bukan menggunakkan *binary search*.

void user\_cari\_kamera() {

if (head\_kamera == nullptr) {

cout << "Tidak ada kamera yang tersedia\n";

return;

} else {

clearScreen();

string tipe\_yang\_dicari;

cout << "Masukkan tipe kamera yang ingin dicari: ";

getline(cin,tipe\_yang\_dicari);

kamera \*temp = head\_kamera;

while (temp->next != head\_kamera && temp->tipe != tipe\_yang\_dicari) {

temp = temp->next;

}

if (temp->next == head\_kamera) {

cout << "\nKamera dengan tipe " << tipe\_yang\_dicari << "tidak ditemukan\n";

} else {

cout << "\nKamera ditemukan: " << temp->merek << " " << temp->tipe << " , dengan harga sewa per hari "

<< temp->harga << endl << endl;

}

}

}

Secara perhitungan, *sequential search* atau *linear search* memiliki kompleksitas waktu sebanyak O(n), karena waktu yang dibutuhkan untuk melakukan pencarian bergantung pada banyaknya data dalam suatu struktur data. Sedangkan *binary search* memiliki kompleksitas waktu sebanyak O(log n), karena *binary search* membagi pencarian menjadi dua bagian yang sama besar dan ditetapkan suatu titik tengah.

Tetapi kami tetap menggunakkan *sequential search* karena secara pengimplementasian lebih sederhana dan ukuran data juga relatif kecil. Selain itu, *sequential search* tidak perlu mengurutkan data terlebih dahulu. Hal ini juga kami implementasikan terhadap fungsi *void user\_cari\_lensa* juga.

Sedangkan pada *sorting*…

1. Rekursif

Rekursif adalah fungsi yang memanggil dirinya sendiri. Program kami juga menggunakkan fungsi rekursif, terutama pada fungsi *void inorder* untuk menampilkan *root member* secara urut. Fungsi *void inorder* adalah fungsi *direct tail recursion*.

void inorder(member \*root) {

if (root != nullptr) {

inorder(root->left);

cout << "\nID Member\t: " << root->id << endl;

cout << "Nama\t\t: " << root->nama\_member << endl;

cout << "Tanggal Lahir\t: " << root->tanggal\_lahir << endl;

cout << "No Telp\t\t: " << root->no\_telp << endl;

inorder(root->right);

}

}

1. *Sorting*

*Sorting* adalah metode yang digunakan untuk melakukan pengurutan data berdasarkan kriteria tertentu. Pada program saya, fungsi *sorting* digunakan untuk mengurutkan data *input* kamera dan juga lensa. Ketika admin melakukan *input* kamera dan lensa, maka *linked list* akan langsung melakukan *sorting* dan memasukkan data berdasarkan harga, dari yang terkecil ke terbesar.

void admin\_input\_kamera() {

clearScreen();

string merekkamera;

kamera \*newKamera = new kamera;

cout << "Masukkan merek kamera\t: ";

getline(cin, merekkamera);

cout << "Masukkan tipe kamera\t: ";

getline(cin, newKamera->tipe);

cout << "Masukkan harga kamera\t: ";

cin >> newKamera->harga;

toUpperCase(merekkamera);

newKamera->merek = merekkamera;

cin.ignore();

newKamera->next = newKamera;

newKamera->prev = newKamera;

if (head\_kamera == nullptr) {

head\_kamera = newKamera;

tail\_kamera = newKamera;

head\_kamera->next = head\_kamera;

head\_kamera->prev = head\_kamera;

tail\_kamera->next = tail\_kamera;

tail\_kamera->prev = tail\_kamera;

} else {

newKamera->next = head\_kamera;

head\_kamera->prev = newKamera;

head\_kamera = newKamera;

head\_kamera->prev = tail\_kamera;

tail\_kamera->next = head\_kamera;

}

cout << "\nKamera berhasil ditambahkan\n";

}

void admin\_input\_lensa() {

clearScreen();

string mereklensa;

lensa \*newLensa = new lensa();

cout << "Masukkan merek lensa\t: ";

getline(cin, mereklensa);

cout << "Masukkan tipe lensa\t: ";

getline(cin, newLensa->tipe);

cout << "Masukkan harga lensa\t: ";

cin >> newLensa->harga;

toUpperCase(mereklensa);

newLensa->merek = mereklensa;

cin.ignore();

newLensa->next = newLensa;

newLensa->prev = newLensa;

if (head\_lensa == nullptr) {

head\_lensa = newLensa;

tail\_lensa = newLensa;

head\_lensa->next = head\_lensa;

head\_lensa->prev = head\_lensa;

tail\_lensa->next = tail\_lensa;

tail\_lensa->prev = tail\_lensa;

} else {

newLensa->next = head\_lensa;

head\_lensa->prev = newLensa;

head\_lensa = newLensa;

head\_lensa->prev = tail\_lensa;

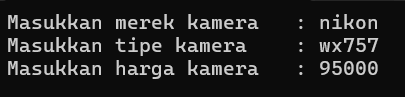
tail\_lensa->next = head\_lensa;

}

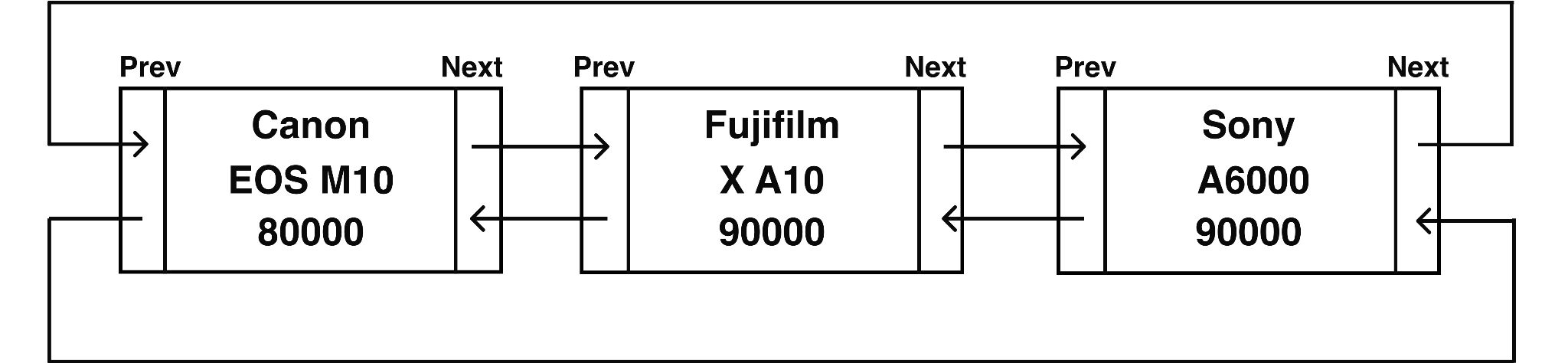
cout << "\nLensa berhasil ditambahkan\n";

}

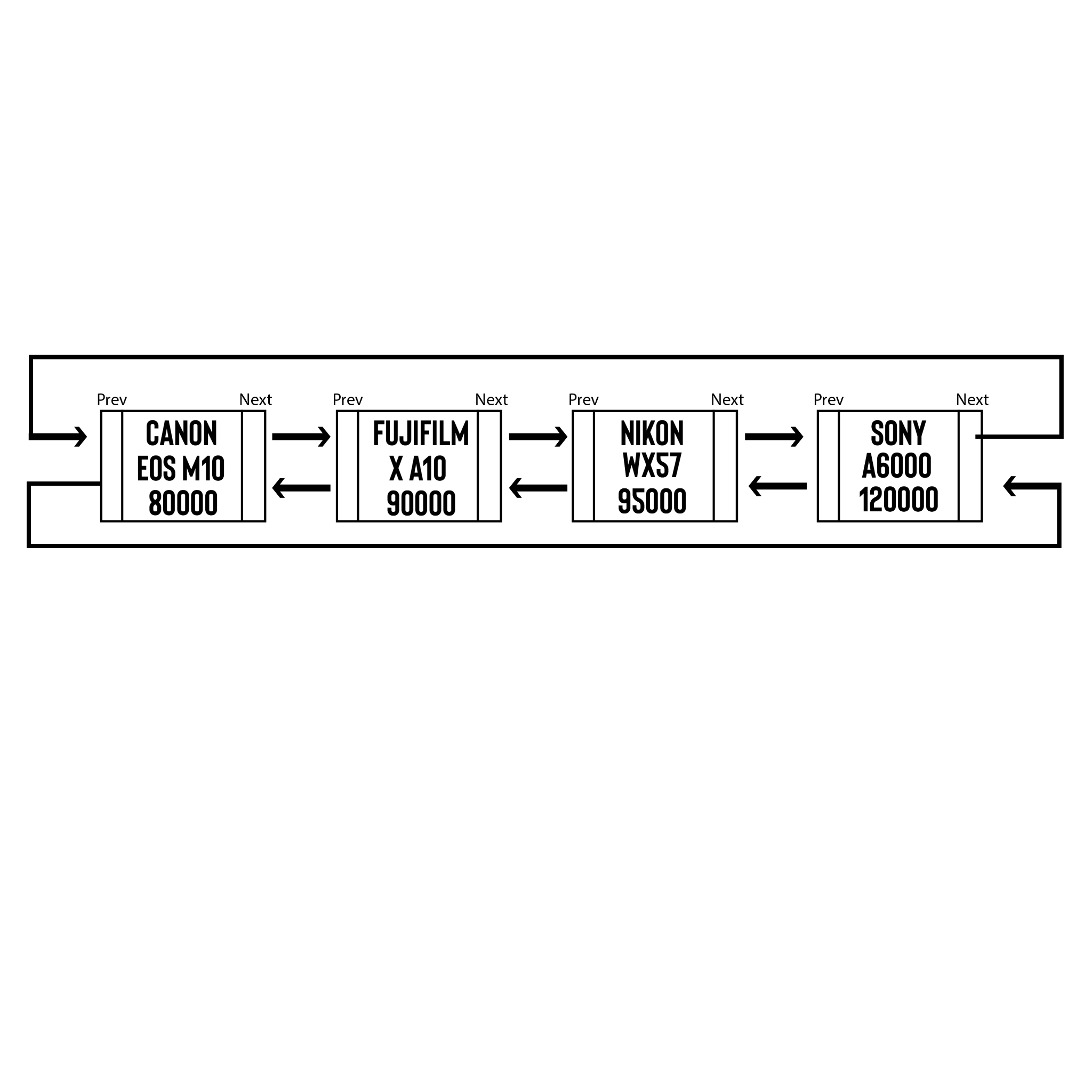
Berikut adalah contoh ketika admin memasukkan data kamera ke dalam linked list. Ketika admin memasukkan kamera, maka akan dicari hargayang lebih besar dan kecil dari harga yang akan dimasukkan. Kemudian akan dimasukkan ke antara data tersebut.



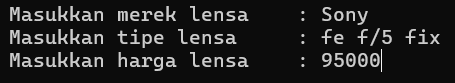
Ini merupakan ilustrasi awal linked list kamera dari data yang sebelumnya sudah dimasukkan.



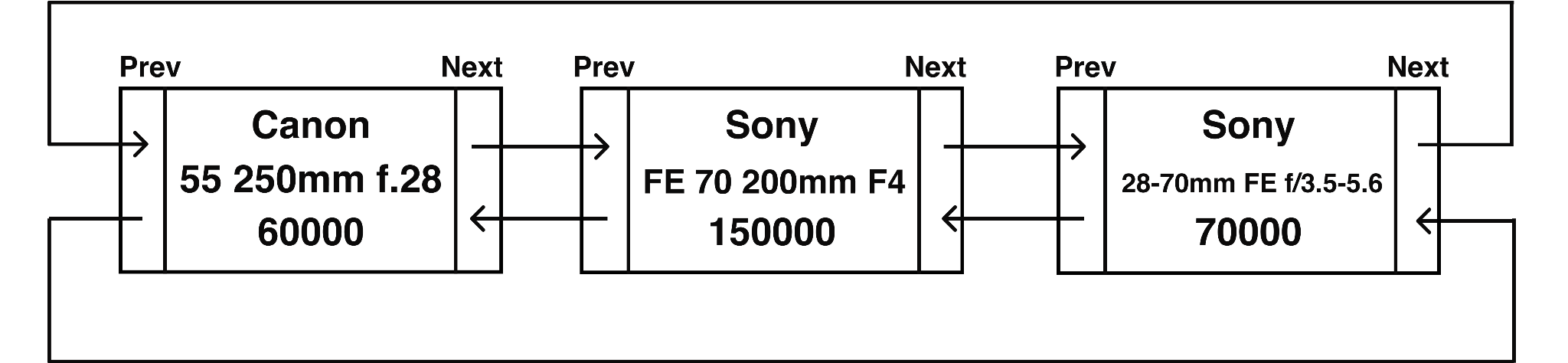
Ini merupakan ilustrasi setelah admin melakukan *input* data kamera, lokasi akan disesuaikan berdasarkan harga.



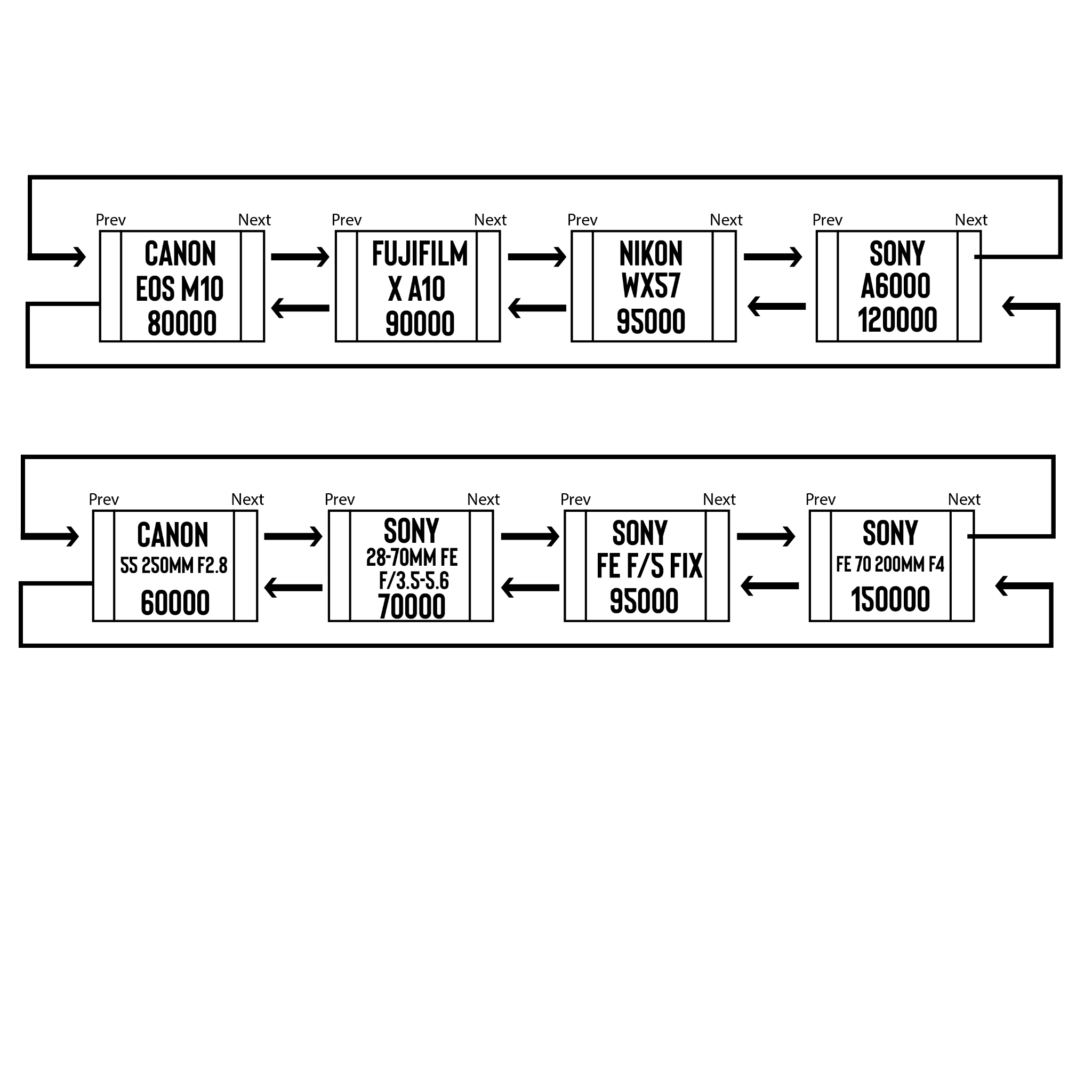
Ini merupakan contoh ketika admin memasukkan data lensa. Cara input sama seperti ketika melakukan input kamera. Lensa akan diurutkan berdasarkan harga.



Ini merupakan ilustrasi awal linked list kamera dari data yang sebelumnya sudah dimasukkan.



Ini merupakan ilustrasi ketika admin melakukan *input* lensa. Menggunakan sistem yang sama seperti *input* pada kamera, data baru akan disisipkan berdasarkan harga sehingga data akan urut dari harga termurah ke termahal.



1. *Searching*

*Searching* adalah metode yang digunakan untuk melakukan proses pencarian data yang diinginkan. Pada program kami, fungsi *searching* digunakan untuk mencari tipe kamera dan lensa yang diinginkan. Fungsi *searching* yang kami gunakan adalah *sequential search*.

*Sequential search* adalah metode penelusuran yang menelusuri semua elemen dari awal hingga akhir secara satu persatu. Kelebihan dari *sequential search* adalah data tidak perlu diurutkan terlebih dahulu. Walaupun kompleksitas waktu *sequential search* tidak sebaik *binary search*, pengimplemenatasian *sequential search* cocok dengan data kamera dan lensa yang kami miliki, karena jumlah data kamera dan lensa yang dimiliki tidak banyak dan tidak perlu diurutkan juga.

// Fungsi untuk mencari kamera

void user\_cari\_kamera() {

if (head\_kamera == nullptr) {

cout << "Tidak ada kamera yang tersedia\n";

return;

} else {

clearScreen();

string tipe\_yang\_dicari;

cout << "Masukkan tipe kamera yang ingin dicari: ";

getline(cin,tipe\_yang\_dicari);

kamera \*temp = head\_kamera;

while (temp->next != head\_kamera && temp->tipe != tipe\_yang\_dicari) {

temp = temp->next;

}

if (temp->next == head\_kamera) {

cout << "\nKamera dengan tipe " << tipe\_yang\_dicari << "tidak ditemukan\n";

} else {

cout << "\nKamera ditemukan: " << temp->merek << " " << temp->tipe << " , dengan harga sewa per hari "

<< temp->harga << endl << endl;

}

}

}

// Fungsi untuk mencari lensa

void user\_cari\_lensa() {

if (head\_lensa == nullptr) {

cout << "Tidak ada lensa yang tersedia\n";

return;

} else {

clearScreen();

string tipe\_yang\_dicari;

cout << "Masukkan tipe lensa yang ingin dicari: ";

getline(cin,tipe\_yang\_dicari);

lensa \*temp = head\_lensa;

while (temp->next != head\_lensa && temp->tipe != tipe\_yang\_dicari) {

temp = temp->next;

}

if (temp->next == head\_lensa) {

cout << "\nLensa dengan tipe " << tipe\_yang\_dicari << "tidak ditemukan\n";

} else {

cout << "\nLensa ditemukan: " << temp->merek << " " << temp->tipe << " , dengan harga sewa per hari "

<< temp->harga << endl << endl;

}

}

}

Berikut adalah contoh *output* yang diberikan oleh fungsi user\_cari\_kamera yang dapat diakses oleh user

Masukkan tipe kamera yang ingin dicari: A6000

Lensa ditemukan: Sony A6000 , dengan harga sewa per hari 120000

Klik enter untuk melanjutkan

dan apabila kamera yang dicari tidak ditemukan, maka berikut adalah *output* nya

Masukkan tipe kamera yang ingin dicari: EOS M10

Kamera dengan tipe EOS M10 tidak ditemukan

Berikut adalah contoh *output* yang diberikan oleh fungsi user\_cari\_lensa yang dapat diakses oleh user

Masukkan tipe lensa yang ingin dicari: FE 70 200mm F4

Lensa ditemukan: Sony FE 70 200mm F4 , dengan harga sewa per hari 150000

Klik enter untuk melanjutkan

dan apabila lensa yang dicari tidak ditemukan, maka berikut adalah *output* nya

Masukkan tipe lensa yang ingin dicari: EF 14mm f/ 2.8 L II USM

Lensa dengan tipe EF 14mm f/ 2.8 L II USM tidak ditemukan

Klik enter untuk melanjutkan

1. *Table Hash* dan *Collision Handling*

*Collision Handling* adalah pengaturan sistematis di mana dua data yang berbeda tetapi memiliki hasil indeks *hash* yang sama. Dimana data dapat dimasukkan ke dalam index *table* yang sama tetapi tetap dikaitkan dengan data lain yang memiliki hasil *index* sama (*chaining*), atau data dapat diletakkan pada *table* berikutnya (*linear probing*).

Pada program kami, *Table Hash* digunakan untuk menyimpan data penyewaan kamera atau lensa. ID member di *hash* ing untuk mengetahui indeks mana data penyewaan dapat disimpan. *Collision Handling* yang saya gunakan adalah teknik *chaining*.

Kami menciptakan 10 fungsi untuk menambah, menghapus, menampilkan data pada *table hash*.

1. Fungsi *int* fungsi\_hash untuk melakukan *modulo* terhadap ID member, sehingga mengetahui pada indeks mana data penyewaan dapat disimpan
2. Fungsi *void* sewa\_kamera untuk melakukan proses penyewaan kamera dan memasukkan data kedalam *table hash*
3. Fungsi *void* sewa\_lensa untuk melakukan proses penyewaan lensa dan memasukkan data kedalam *table hash*
4. Fungsi *void display\_data\_penyewaan* untuk menampilkan *output* berupa data ID member yang melakukan penyewaan, nama penyewa, tipe dan merk kamera yang disewa, lama penyewaan (hari) dan harga total yang harus dibayar oleh penyewa
5. Fungsi *void hapus\_penyewaan* untuk menghapus data penyewaan

int fungsi\_hash(int id){

return id % ukuran\_tabel\_hash;

}

void sewa\_kamera(int id, string nama, kamera\* kamera\_yang\_disewa\_sekarang, int hari, int harga\_total){

int index = fungsi\_hash(id);

penyewaan\_barang \*entry = tabel\_hash[index];

while (entry != nullptr){

if(entry->id == id){

kamera\_yang\_disewa\_sekarang->next = entry->kamera\_yang\_disewa;

if(entry->kamera\_yang\_disewa != nullptr){

entry->kamera\_yang\_disewa->prev = kamera\_yang\_disewa\_sekarang;

}

entry->kamera\_yang\_disewa = kamera\_yang\_disewa\_sekarang;

return;

}

entry = entry->next;

}

entry = new penyewaan\_barang();

entry->id = id;

entry->nama = nama;

entry->kamera\_yang\_disewa = kamera\_yang\_disewa\_sekarang;

entry->lensa\_yang\_disewa = nullptr;

entry->hari = hari;

entry->harga\_total = harga\_total;

entry->next = tabel\_hash[index];

tabel\_hash[index] = entry;

cin.ignore();

cout << "\nPermohonan penyewaan telah diajukan, silahkan datang ke tempat kami\n\n";

return;

}

void sewa\_lensa(int id,string nama, lensa\* lensa\_yang\_disewa\_sekarang, int hari, int harga\_total){

int index = fungsi\_hash(id);

penyewaan\_barang \*entry = tabel\_hash[index];

while (entry != nullptr){

if(entry->id == id){

lensa\_yang\_disewa\_sekarang->next = entry->lensa\_yang\_disewa;

if(entry->lensa\_yang\_disewa != nullptr){

entry->lensa\_yang\_disewa->prev = lensa\_yang\_disewa\_sekarang;

}

entry->lensa\_yang\_disewa = lensa\_yang\_disewa\_sekarang;

return;

}

entry = entry->next;

}

entry = new penyewaan\_barang();

entry->id = id;

entry->nama = nama;

entry->kamera\_yang\_disewa = nullptr;

entry->lensa\_yang\_disewa = lensa\_yang\_disewa\_sekarang;

entry->hari = hari;

entry->harga\_total = harga\_total;

entry->next = tabel\_hash[index];

tabel\_hash[index] = entry;

cin.ignore();

cout << "\nPermohonan penyewaan telah diajukan, silahkan datang ke tempat kami\n\n";

return;

}

void display\_data\_penyewaan() {

bool isEmpty = true;

for (int i = 0; i < ukuran\_tabel\_hash; i++) {

if (tabel\_hash[i] != nullptr) {

isEmpty = false;

break;

}

}

if (isEmpty) {

cout << "\nTidak ada data penyewaan\n\n";

return;

}

for (int i = 0; i < ukuran\_tabel\_hash; i++) {

penyewaan\_barang\* entry = tabel\_hash[i];

while (entry != nullptr) {

cout << entry->nama << " dengan " << entry->id << " meminjam \n";

if (entry->kamera\_yang\_disewa != nullptr) {

cout << "\tKamera " << entry->kamera\_yang\_disewa->merek << " " << entry->kamera\_yang\_disewa->tipe << " selama " << entry->hari << " hari, total biaya Rp " << entry->harga\_total << endl << endl;

}

if (entry->lensa\_yang\_disewa != nullptr) {

cout << "\tLensa " << entry->lensa\_yang\_disewa->merek << " " << entry->lensa\_yang\_disewa->merek << " selama " << entry->hari << " hari, total biaya Rp " << entry->harga\_total << endl << endl;

}

entry = entry->next;

}

}

return;

}

void hapus\_penyewaan(int id\_dicari) {

int index = fungsi\_hash(id\_dicari);

penyewaan\_barang\* entry = tabel\_hash[index];

penyewaan\_barang\* prev = nullptr;

while (entry != nullptr) {

if (entry->id == id\_dicari) {

if (prev == nullptr) {

tabel\_hash[index] = entry->next;

} else {

prev->next = entry->next;

}

delete entry;

cout << "\nData penyewaan untuk member " << id\_dicari << " telah dihapus.\n\n";

return;

}

prev = entry;

entry = entry->next;

}

cout << "\nData penyewaan untuk member " << id\_dicari << " tidak ditemukan.\n\n";

}

Berikut adalah contoh *output* yang diberikan oleh fungsi sewa\_kamera yang dapat diakses oleh user

Masukkan nama member : Muhammad Hasan

Masukkan tgl lahir (dd-mm-yyyy) : 17-05-1994

Masukkan no telp : 08529123643

Member berhasil dibuat

ID Member : 133343

Nama : Muhammad Hasan

Tanggal Lahir : 17-05-1994

No Telp : 08529123643

Ingin sewa kamera berapa hari?

2

Permohonan penyewaan telah diajukan, silahkan datang ke tempat kami

Klik enter untuk melanjutkan

Berikut adalah contoh *output* yang diberikan oleh fungsi sewa\_lensa yang dapat diakses oleh user

Masukkan nama member : Fajar Budi

Masukkan tgl lahir (dd-mm-yyyy) : 09-12-2000

Masukkan no telp : 082190901918

Member berhasil dibuat

ID Member : 543218

Nama : Fajar Budi

Tanggal Lahir : 09-12-2000

No Telp : 082190901918

Ingin sewa lensa berapa hari?

2

Permohonan penyewaan telah diajukan, silahkan datang ke tempat kami

Klik enter untuk melanjutkan

Berikut adalah contoh *output* yang diberikan oleh fungsi display\_data\_penyewaan yang dapat diakses oleh admin

================================

Admin

================================

1. Penyimpanan kamera dan lensa

2. Proses penyewaan

3. Pengurusan Member

4. Pengurusan Fotografer

0. Logout

> 2

Muhammad Hasan dengan 133343 meminjam

Kamera Sony A6000 selama 2 hari, total biaya Rp 240000

Fajar Budi dengan 543218 meminjam

Lensa Canon Canon selama 2 hari, total biaya Rp 120000

Hapus data penyewaan?

>

Berikut adalah contoh *output* yang diberikan oleh fungsi hapus\_penyewaan yang dapat diakses oleh admin

================================

Admin

================================

1. Penyimpanan kamera dan lensa

2. Proses penyewaan

3. Pengurusan Member

4. Pengurusan Fotografer

0. Logout

> 2

Muhammad Hasan dengan 133343 meminjam

Kamera Sony A6000 selama 2 hari, total biaya Rp 240000

Fajar Budi dengan 543218 meminjam

Lensa Canon Canon selama 2 hari, total biaya Rp 120000

Hapus data penyewaan?

>

Masukkan id member

> 133343

Data penyewaan untuk member 133343 telah dihapus

dan jika ID member tidak ditemukan, maka berikut adalah *output* yang dihasilkan

Muhammad Hasan dengan 133343 meminjam

Kamera Sony A6000 selama 2 hari, total biaya Rp 240000

Fajar Budi dengan 543218 meminjam

Lensa Canon Canon selama 2 hari, total biaya Rp 120000

Hapus data penyewaan?

>

Masukkan id member

> 12345

Data penyewaan untuk member 12345 tidak ditemukan

1. *Tree*

*Tree* adalah struktur data yang terdiri dari *root* dan *subtree* yang disusun secara berhirarki ke bawah. *Tree* pada program kami digunakan untuk menyimpan data member. Kami menciptakan 10 fungsi untuk menambah, menghapus, dan mengakses *tree*. Fungsi yang digunakan banyak karena terdapat 2 barang yang berbeda yaitu untuk kamera dan lensa, selain itu fungsi yang digunakan juga masih dibedakan berdasarkan admin dan member.

1. Fungsi *void* admin\_buat\_kamera untuk fungsi admin menciptakan data member
2. Fungsi *void inorder* untuk menampilkan daftar member secara urut berdasarkan ID member
3. Fungsi *void* admin\_lihat\_member untuk fungsi admin menampilkan daftar member
4. Fungsi *void* admin\_hapus\_member untuk menghapus data suatu member
5. Fungsi *void* user\_buat\_member\_untuk\_kamera untuk fungsi user melakukan pesanan penyewaan kamera, tetapi user harus membuat data member terlebih dahulu
6. Fungsi *void* user\_buat\_member\_untuk\_lensa untuk fungsi user melakukan pesanan penyewaan lensa, tetapi user harus membuat data member terlebih dahulu

// Fungsi untuk menambahkan member ke tree

void admin\_buat\_member() {

clearScreen();

member \*newMember = new member();

cout << "Masukkan nama member\t\t: ";

getline(cin,newMember->nama\_member);

cout << "Masukkan tgl lahir (dd-mm-yyyy)\t: ";

getline(cin,newMember->tanggal\_lahir);

cout << "Masukkan no telp\t\t: ";

cin >> newMember->no\_telp;

newMember->left = nullptr;

newMember->right = nullptr;

newMember->id = buat\_id(

newMember->nama\_member, newMember->tanggal\_lahir, newMember->no\_telp);

cin.ignore();

if (root == nullptr) {

root = newMember;

} else {

member \*current = root;

member \*parent = nullptr;

while (current != nullptr) {

parent = current;

if (newMember->nama\_member < current->nama\_member) {

current = current->left;

} else {

current = current->right;

}

}

if (newMember->nama\_member < parent->nama\_member) {

parent->left = newMember;

} else {

parent->right = newMember;

}

}

cout << "\nMember berhasil dibuat\n";

cout << "\nId Member\t: " << newMember->id << endl;

cout << "Nama\t\t: " << newMember->nama\_member << endl;

cout << "Tanggal Lahir\t: " << newMember->tanggal\_lahir << endl;

cout << "No Telp\t\t: " << newMember->no\_telp << endl;

}

// Fungsi untuk melihat daftar member (inorder traversal)

void inorder(member \*root) {

if (root != nullptr) {

inorder(root->left);

cout << "\nID Member\t: " << root->id << endl;

cout << "Nama\t\t: " << root->nama\_member << endl;

cout << "Tanggal Lahir\t: " << root->tanggal\_lahir << endl;

cout << "No Telp\t\t: " << root->no\_telp << endl;

inorder(root->right);

}

}

// Fungsi untuk melihat daftar member

void admin\_lihat\_member() {

if (root == nullptr) {

cout << "Tidak ada member\n";

return;

} else {

inorder(root);

}

}

//Fungsi untuk menghapus member

void admin\_hapus\_member() {

if (root == nullptr) {

return;

}

string memberName;

cout << "Masukkan nama member yang ingin dihapus: ";

cin >> memberName;

member \*current = root;

member \*parent = nullptr;

while (current != nullptr && current->nama\_member != memberName) {

parent = current;

if (memberName < current->nama\_member) {

current = current->left;

} else {

current = current->right;

}

}

if (current == nullptr) {

return;

}

if (current->left == nullptr) {

member \*rightChild = current->right;

if (current == root) {

root = rightChild;

} else {

if (parent->left == current) {

parent->left = rightChild;

} else {

parent->right = rightChild;

}

}

if (rightChild != nullptr) {

rightChild->parent = parent;

}

} else {

member \*leftChild = current->left;

member \*successor = leftChild;

while (successor->right != nullptr) {

successor = successor->right;

}

if (successor->left != nullptr) {

successor->left->parent = successor;

}

successor->left = leftChild;

successor->right = current->right;

if (current == root) {

root = successor;

} else {

if (parent->left == current) {

parent->left = successor;

} else {

parent->right = successor;

}

}

successor->parent = parent;

if (current->right != nullptr) {

current->right->parent = successor;

}

}

delete current;

cout << "Member berhasil di hapus!\n";

cout << "Enter untuk melanjutkan";

cin.get();

}

void user\_buat\_member\_untuk\_kamera(kamera \*kamera\_yang\_disewa) {

clearScreen();

member \*newMember = new member();

cout << "Masukkan nama member\t\t: ";

getline(cin,newMember->nama\_member);

cout << "Masukkan tgl lahir (dd-mm-yyyy)\t: ";

getline(cin,newMember->tanggal\_lahir);

cout << "Masukkan no telp\t\t: ";

cin >> newMember->no\_telp;

newMember->left = nullptr;

newMember->right = nullptr;

newMember->id = buat\_id(

newMember->nama\_member, newMember->tanggal\_lahir, newMember->no\_telp);

cin.ignore();

if (root == nullptr) {

root = newMember;

} else {

member \*current = root;

member \*parent = nullptr;

while (current != nullptr) {

parent = current;

if (newMember->nama\_member < current->nama\_member) {

current = current->left;

} else {

current = current->right;

}

}

if (newMember->nama\_member < parent->nama\_member) {

parent->left = newMember;

} else {

parent->right = newMember;

}

}

cout << "\nMember berhasil dibuat\n";

cout << "\nID Member\t: " << newMember->id << endl;

cout << "Nama\t\t: " << newMember->nama\_member << endl;

cout << "Tanggal Lahir\t: " << newMember->tanggal\_lahir << endl;

cout << "No Telp\t\t: " << newMember->no\_telp << endl;

int lama\_sewa;

cout << "\nIngin sewa kamera berapa hari?\n";

cin >> lama\_sewa;

int harga\_total = kamera\_yang\_disewa->harga \* lama\_sewa;

sewa\_kamera(newMember->id,newMember->nama\_member,kamera\_yang\_disewa,lama\_sewa,harga\_total);

}

void user\_buat\_member\_untuk\_lensa(lensa \*lensa\_yang\_disewakan) {

clearScreen();

member \*newMember = new member();

cout << "Masukkan nama member\t\t: ";

getline(cin,newMember->nama\_member);

cout << "Masukkan tgl lahir (dd-mm-yyyy)\t: ";

getline(cin,newMember->tanggal\_lahir);

cout << "Masukkan no telp\t\t: ";

cin >> newMember->no\_telp;

newMember->left = nullptr;

newMember->right = nullptr;

newMember->id = buat\_id(

newMember->nama\_member, newMember->tanggal\_lahir, newMember->no\_telp);

cin.ignore();

if (root == nullptr) {

root = newMember;

} else {

member \*current = root;

member \*parent = nullptr;

while (current != nullptr) {

parent = current;

if (newMember->nama\_member < current->nama\_member) {

current = current->left;

} else {

current = current->right;

}

}

if (newMember->nama\_member < parent->nama\_member) {

parent->left = newMember;

} else {

parent->right = newMember;

}

}

cout << "\nMember berhasil dibuat\n";

cout << "\nID Member\t: " << newMember->id << endl;

cout << "Nama\t\t: " << newMember->nama\_member << endl;

cout << "Tanggal Lahir\t: " << newMember->tanggal\_lahir << endl;

cout << "No Telp\t\t: " << newMember->no\_telp << endl;

int lama\_sewa;

cout << "\n Ingin sewa lensa berapa hari?\n";

cin >> lama\_sewa;

int harga\_total = lensa\_yang\_disewakan->harga \* lama\_sewa;

sewa\_lensa(newMember->id,newMember->nama\_member,lensa\_yang\_disewakan,lama\_sewa,harga\_total);

}

Berikut adalah contoh *output* yang diberikan oleh fungsi admin\_buat\_member yang dapat diakses oleh admin

Masukkan nama member : Muhammad Hasan

Masukkan tgl lahir (dd-mm-yyyy) : 17-05-1994

Masukkan no telp : 08529123643

Member berhasil dibuat

Id Member : 133343

Nama : Muhammad Hasan

Tanggal Lahir : 17-05-1994

No Telp : 08529123643

Klik enter untuk melanjutkan

Berikut adalah contoh *output* yang diberikan oleh fungsi *inorder* yang dapat diakses oleh admin

ID Member : 133343

Nama : Muhammad Hasan

Tanggal Lahir : 17-05-1994

No Telp : 08529123643

ID Member : 543218

Nama : Fajar Budi

Tanggal Lahir : 09-12-2000

No Telp : 082190901918

Klik enter untuk melanjutkan

Berikut adalah contoh *output* yang diberikan oleh fungsi admin\_hapus\_member yang dapat diakses oleh admin

Masukkan nama member yang ingin dihapus : Muhammad Hasan

Member berhasil dihapus

Berikut adalah contoh *output* yang diberikan oleh fungsi user\_buat\_member\_untuk\_kamera yang dapat diakses oleh user

Masukkan nama member : Muhammad Hasan

Masukkan tgl lahir (dd-mm-yyyy) : 17-05-1994

Masukkan no telp : 08529123643

Member berhasil dibuat

ID Member : 133343

Nama : Muhammad Hasan

Tanggal Lahir : 17-05-1994

No Telp : 08529123643

Ingin sewa kamera berapa hari?

2

Berikut adalah contoh *output* yang diberikan oleh fungsi user\_buat\_member\_untuk\_lensa yang dapat diakses oleh user

Masukkan nama member : Fajar Budi

Masukkan tgl lahir (dd-mm-yyyy) : 09-12-2000

Masukkan no telp : 082190901918

Member berhasil dibuat

ID Member : 543218

Nama : Fajar Budi

Tanggal Lahir : 09-12-2000

No Telp : 082190901918

Ingin sewa lensa berapa hari?

2

1. *Graph*

*Graph* adalah representasi objek-objek diskrit dan hubungan antara objek-objek tersebut. Pada program kami, *graph* digunakan untuk menghubungkan perekomendasian antara satu spot foto dengan spot foto lainnya. Semua spot foto setidaknya terhubung dengan satu spot foto dan *graph* dibuat dengan metode BFS

struct destinasi {

string tempat;

destinasi\* next;

};

struct graph {

int banyak\_tempat;

string tempat[10];

destinasi\* adj[10];

};

void BFS(graph &tour, string start) {

bool visited[10] = {false};

string queue\_tour[10];

int front = 0, rear = 0;

int start\_index = -1;

for (int i = 0; i < tour.banyak\_tempat; i++) {

if (tour.tempat[i] == start) {

start\_index = i;

break;

}

}

if (start\_index == -1) return;

visited[start\_index] = true;

queue\_tour[rear++] = start;

while (front < rear) {

string start\_tour = queue\_tour[front++];

cout << start\_tour << " ---> ";

int index\_sekarang = -1;

for (int i = 0; i < tour.banyak\_tempat; i++) {

if (tour.tempat[i] == start\_tour) {

index\_sekarang = i;

break;

}

}

destinasi \*adj\_list = tour.adj[index\_sekarang];

while (adj\_list != nullptr) {

int adj\_index = -1;

for (int i = 0; i < tour.banyak\_tempat; i++) {

if (tour.tempat[i] == adj\_list->tempat) {

adj\_index = i;

break;

}

}

if (!visited[adj\_index]) {

visited[adj\_index] = true;

queue\_tour[rear++] = adj\_list->tempat;

}

adj\_list = adj\_list->next;

}

}

cout << "Finish\n";

}

graph destinasi\_foto;

destinasi\_foto.banyak\_tempat = 10;

string destinasi[10] = {

"Kampung Arab", "Kampung Bulak", "Klenteng Sanggar Agung", "Patung Buddha 4 Rupa",

"Museum House of Sampoerna", "G-Walk Citraland", "Tugu Pahlawan",

"Museum Pahlawan", "Jalan Tunjungan", "Kebun Binatang Surabaya"

};

for (int i = 0; i < 10; i++){

destinasi\_foto.tempat[i] = destinasi[i];

destinasi\_foto.adj[i] = nullptr;

}

tambah\_garis(destinasi\_foto, "Kampung Arab", "Museum House of Sampoerna");

tambah\_garis(destinasi\_foto, "Kampung Arab", "Kampung Bulak");

tambah\_garis(destinasi\_foto, "Kampung Bulak", "Patung Buddha 4 Rupa");

tambah\_garis(destinasi\_foto, "Kampung Bulak", "Klenteng Sanggar Agung");

tambah\_garis(destinasi\_foto, "Kampung Bulak", "Kampung Arab");

tambah\_garis(destinasi\_foto, "Klenteng Sanggar Agung", "Patung Buddha 4 Rupa");

tambah\_garis(destinasi\_foto, "Klenteng Sanggar Agung", "Kampung Bulak");

tambah\_garis(destinasi\_foto, "Patung Buddha 4 Rupa", "Klenteng Sanggar Agung");

tambah\_garis(destinasi\_foto, "Patung Buddha 4 Rupa", "Kampung Bulak");

tambah\_garis(destinasi\_foto, "Museum House of Sampoerna", "Kampung Arab");

tambah\_garis(destinasi\_foto, "Museum House of Sampoerna", "Tugu Pahlawan");

tambah\_garis(destinasi\_foto, "Museum House of Sampoerna", "Museum Pahlawan");

tambah\_garis(destinasi\_foto, "Museum House of Sampoerna", "G-Walk Citraland");

tambah\_garis(destinasi\_foto, "G-Walk Citraland", "Museum House of Sampoerna");

tambah\_garis(destinasi\_foto, "G-Walk Citraland", "Tugu Pahlawan");

tambah\_garis(destinasi\_foto, "Tugu Pahlawan", "Museum Pahlawan");

tambah\_garis(destinasi\_foto, "Tugu Pahlawan", "Museum House of Sampoerna");

tambah\_garis(destinasi\_foto, "Tugu Pahlawan", "G-Walk Citraland");

tambah\_garis(destinasi\_foto, "Tugu Pahlawan", "Jalan Tunjungan");

tambah\_garis(destinasi\_foto, "Museum Pahlawan", "Tugu Pahlawan");

tambah\_garis(destinasi\_foto, "Museum Pahlawan", "Jalan Tunjungan");

tambah\_garis(destinasi\_foto, "Museum Pahlawan", "Museum House of Sampoerna");

tambah\_garis(destinasi\_foto, "Jalan Tunjungan", "Kebun Binatang Surabaya");

tambah\_garis(destinasi\_foto, "Jalan Tunjungan", "Tugu Pahlawan");

tambah\_garis(destinasi\_foto, "Jalan Tunjungan", "Museum Pahlawan");

tambah\_garis(destinasi\_foto, "Kebun Binatang Surabaya", "Jalan Tunjungan");

Dengan daftar tempat sebagai berikut :

cout << "Spot foto yang recommended di Surabaya\n";

cout << "1. Kampung Arab , Surabaya Utara\n";

cout << "2. Kampung Bulak , Surabaya Utara\n";

cout << "3. Klenteng Sanggar Agung , Surabaya Utara\n";

cout << "4. Patung Buddha 4 Rupa , Surabaya Utara\n";

cout << "5. Museum House of Sampoerna, Surabaya Utara\n";

cout << "6. G-Walk Citraland , Surabaya Barat\n";

cout << "7. Tugu Pahlawan , Surabaya Pusat\n";

cout << "8. Museum Pahlawan , Surabaya Pusat\n";

cout << "9. Jalan Tunjungan, Surabaya Pusat\n";

cout << "10. Kebun Binatang Surabaya, Surabaya Selatan\n\n";

Berikut adalah hasil *output* yang diberikan oleh fungsi BFS yang dapat diakses oleh user

Spot foto yang recommended di Surabaya

1. Kampung Arab , Surabaya Utara

2. Kampung Bulak , Surabaya Utara

3. Klenteng Sanggar Agung , Surabaya Utara

4. Patung Buddha 4 Rupa , Surabaya Utara

5. Museum House of Sampoerna, Surabaya Utara

6. G-Walk Citraland , Surabaya Barat

7. Tugu Pahlawan , Surabaya Pusat

8. Museum Pahlawan , Surabaya Pusat

9. Jalan Tunjungan, Surabaya Pusat

10. Kebun Binatang Surabaya, Surabaya Selatan

Ketikkan nama tempat terdekat dari posisimu

> Tugu Pahlawan

Mungkin kita bisa merekomendasikan rute spot foto yang dekat dengan destinasimu

Tugu Pahlawan ---> Jalan Tunjungan ---> G-Walk Citraland ---> Museum House of Sampoerna ---> Museum Pahlawan ---> Kebun Binatang Surabaya ---> Kampung Arab ---> Kampung Bulak ---> Klenteng Sanggar Agung ---> Patung Buddha 4 Rupa ---> Finish

Klik enter untuk melanjutkan

1. *Hash Table*

*Table Hash* adalah struktur data yang menyimpan data menurut hasil *hash* (dengan melakukan *modulo* berdasarkan kode buku yang diinputkan) dan data disimpan ke dalam table berdasarkan hasil indeksnya.

Pada program kami, *Table Hash* digunakan untuk menyimpan data penyewaan kamera atau lensa. ID member di *hash* ing untuk mengetahui indeks mana data penyewaan dapat disimpan. *Collision Handling* yang saya gunakan adalah teknik *chaining*.

Kami menciptakan 10 fungsi untuk menambah, menghapus, menampilkan data pada *table hash*.

1. Fungsi *int* fungsi\_hash untuk melakukan *modulo* terhadap ID member, sehingga mengetahui pada indeks mana data penyewaan dapat disimpan
2. Fungsi *void* sewa\_kamera untuk melakukan proses penyewaan kamera dan memasukkan data kedalam *table hash*
3. Fungsi *void* sewa\_lensa untuk melakukan proses penyewaan lensa dan memasukkan data kedalam *table hash*
4. Fungsi *void display\_data\_penyewaan* untuk menampilkan *output* berupa data ID member yang melakukan penyewaan, nama penyewa, tipe dan merk kamera yang disewa, lama penyewaan (hari) dan harga total yang harus dibayar oleh penyewa
5. Fungsi *void hapus\_penyewaan* untuk menghapus data penyewaan

int fungsi\_hash(int id){

return id % ukuran\_tabel\_hash;

}

void sewa\_kamera(int id, string nama, kamera\* kamera\_yang\_disewa\_sekarang, int hari, int harga\_total){

int index = fungsi\_hash(id);

penyewaan\_barang \*entry = tabel\_hash[index];

while (entry != nullptr){

if(entry->id == id){

kamera\_yang\_disewa\_sekarang->next = entry->kamera\_yang\_disewa;

if(entry->kamera\_yang\_disewa != nullptr){

entry->kamera\_yang\_disewa->prev = kamera\_yang\_disewa\_sekarang;

}

entry->kamera\_yang\_disewa = kamera\_yang\_disewa\_sekarang;

return;

}

entry = entry->next;

}

entry = new penyewaan\_barang();

entry->id = id;

entry->nama = nama;

entry->kamera\_yang\_disewa = kamera\_yang\_disewa\_sekarang;

entry->lensa\_yang\_disewa = nullptr;

entry->hari = hari;

entry->harga\_total = harga\_total;

entry->next = tabel\_hash[index];

tabel\_hash[index] = entry;

cin.ignore();

cout << "\nPermohonan penyewaan telah diajukan, silahkan datang ke tempat kami\n\n";

return;

}

void sewa\_lensa(int id,string nama, lensa\* lensa\_yang\_disewa\_sekarang, int hari, int harga\_total){

int index = fungsi\_hash(id);

penyewaan\_barang \*entry = tabel\_hash[index];

while (entry != nullptr){

if(entry->id == id){

lensa\_yang\_disewa\_sekarang->next = entry->lensa\_yang\_disewa;

if(entry->lensa\_yang\_disewa != nullptr){

entry->lensa\_yang\_disewa->prev = lensa\_yang\_disewa\_sekarang;

}

entry->lensa\_yang\_disewa = lensa\_yang\_disewa\_sekarang;

return;

}

entry = entry->next;

}

entry = new penyewaan\_barang();

entry->id = id;

entry->nama = nama;

entry->kamera\_yang\_disewa = nullptr;

entry->lensa\_yang\_disewa = lensa\_yang\_disewa\_sekarang;

entry->hari = hari;

entry->harga\_total = harga\_total;

entry->next = tabel\_hash[index];

tabel\_hash[index] = entry;

cin.ignore();

cout << "\nPermohonan penyewaan telah diajukan, silahkan datang ke tempat kami\n\n";

return;

}

void display\_data\_penyewaan() {

bool isEmpty = true;

for (int i = 0; i < ukuran\_tabel\_hash; i++) {

if (tabel\_hash[i] != nullptr) {

isEmpty = false;

break;

}

}

if (isEmpty) {

cout << "\nTidak ada data penyewaan\n\n";

return;

}

for (int i = 0; i < ukuran\_tabel\_hash; i++) {

penyewaan\_barang\* entry = tabel\_hash[i];

while (entry != nullptr) {

cout << entry->nama << " dengan " << entry->id << " meminjam \n";

if (entry->kamera\_yang\_disewa != nullptr) {

cout << "\tKamera " << entry->kamera\_yang\_disewa->merek << " " << entry->kamera\_yang\_disewa->tipe << " selama " << entry->hari << " hari, total biaya Rp " << entry->harga\_total << endl << endl;

}

if (entry->lensa\_yang\_disewa != nullptr) {

cout << "\tLensa " << entry->lensa\_yang\_disewa->merek << " " << entry->lensa\_yang\_disewa->merek << " selama " << entry->hari << " hari, total biaya Rp " << entry->harga\_total << endl << endl;

}

entry = entry->next;

}

}

return;

}

void hapus\_penyewaan(int id\_dicari) {

int index = fungsi\_hash(id\_dicari);

penyewaan\_barang\* entry = tabel\_hash[index];

penyewaan\_barang\* prev = nullptr;

while (entry != nullptr) {

if (entry->id == id\_dicari) {

if (prev == nullptr) {

tabel\_hash[index] = entry->next;

} else {

prev->next = entry->next;

}

delete entry;

cout << "\nData penyewaan untuk member " << id\_dicari << " telah dihapus.\n\n";

return;

}

prev = entry;

entry = entry->next;

}

cout << "\nData penyewaan untuk member " << id\_dicari << " tidak ditemukan.\n\n";

}

Berikut adalah contoh *output* yang diberikan oleh fungsi sewa\_kamera yang dapat diakses oleh user

Masukkan nama member : Muhammad Hasan

Masukkan tgl lahir (dd-mm-yyyy) : 17-05-1994

Masukkan no telp : 08529123643

Member berhasil dibuat

ID Member : 133343

Nama : Muhammad Hasan

Tanggal Lahir : 17-05-1994

No Telp : 08529123643

Ingin sewa kamera berapa hari?

2

Permohonan penyewaan telah diajukan, silahkan datang ke tempat kami

Klik enter untuk melanjutkan

Berikut adalah contoh *output* yang diberikan oleh fungsi sewa\_lensa yang dapat diakses oleh user

Masukkan nama member : Fajar Budi

Masukkan tgl lahir (dd-mm-yyyy) : 09-12-2000

Masukkan no telp : 082190901918

Member berhasil dibuat

ID Member : 543218

Nama : Fajar Budi

Tanggal Lahir : 09-12-2000

No Telp : 082190901918

Ingin sewa lensa berapa hari?

2

Permohonan penyewaan telah diajukan, silahkan datang ke tempat kami

Klik enter untuk melanjutkan

Berikut adalah contoh *output* yang diberikan oleh fungsi display\_data\_penyewaan yang dapat diakses oleh admin

================================

Admin

================================

1. Penyimpanan kamera dan lensa

2. Proses penyewaan

3. Pengurusan Member

4. Pengurusan Fotografer

0. Logout

> 2

Muhammad Hasan dengan 133343 meminjam

Kamera Sony A6000 selama 2 hari, total biaya Rp 240000

Fajar Budi dengan 543218 meminjam

Lensa Canon Canon selama 2 hari, total biaya Rp 120000

Hapus data penyewaan?

>

Berikut adalah contoh *output* yang diberikan oleh fungsi hapus\_penyewaan yang dapat diakses oleh admin

================================

Admin

================================

1. Penyimpanan kamera dan lensa

2. Proses penyewaan

3. Pengurusan Member

4. Pengurusan Fotografer

0. Logout

> 2

Muhammad Hasan dengan 133343 meminjam

Kamera Sony A6000 selama 2 hari, total biaya Rp 240000

Fajar Budi dengan 543218 meminjam

Lensa Canon Canon selama 2 hari, total biaya Rp 120000

Hapus data penyewaan?

>

Masukkan id member

> 133343

Data penyewaan untuk member 133343 telah dihapus

dan jika ID member tidak ditemukan, maka berikut adalah *output* yang dihasilkan

Muhammad Hasan dengan 133343 meminjam

Kamera Sony A6000 selama 2 hari, total biaya Rp 240000

Fajar Budi dengan 543218 meminjam

Lensa Canon Canon selama 2 hari, total biaya Rp 120000

Hapus data penyewaan?

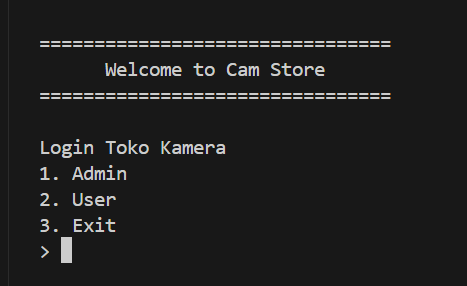
>

Masukkan id member

> 12345

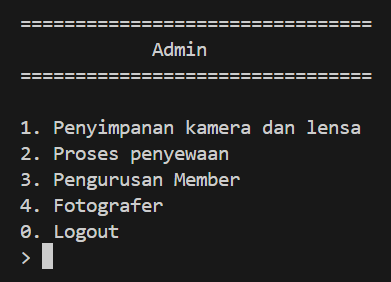
Data penyewaan untuk member 12345 tidak ditemukan

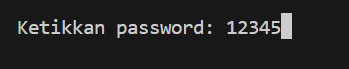
**Penjelasan Mengenai Fitur-Fitur yang Terdapat Pada Program**

Berikut ini adalah tampilan program pada saat pertama kali ditampilkan. Pada bagian ini berfungsi menampilkan opsi login yang ingin dipilih yaitu login sebagai admin, dan login sebagai user*.* Masing-masing pilihan login tentunya memiliki berbagai macam perbedaan fitur di dalam toko. Dan yang terakhir disediakan menu untuk exit sehingga kita bisa keluar dari program. Berikut ini adalah penjelasan mengenai Admin dan User. 

**Admin**

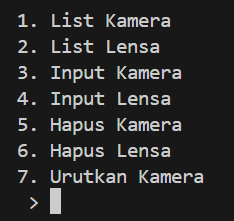
*Roles* dari seorang admin yaitu bertugas sebagai pengurus dari Cam Store, dimana memiliki beberapa wewenang untuk mengatur segala macam data yang ada pada program, memanipulasi, dan melakukan tindakan lainnya. Mengingat tugasnya yang cukup vital, untuk login sebagai seorang admin kita memerlukan password sebagai bentuk keamanan

Ketika telah memasukkan password yang sesuai, maka akan terbukalah semua fitur dari seorang admin seperti ini : 



1. Penyimpanan Kamera dan Lensa

Ini merupakan fitur yang berfungsi untuk mengatur segala jenis data yang berkaitan dengan kamera dan lensa. Jika memilih fitur ini maka akan terdapat tampilan fitur seperti ini :

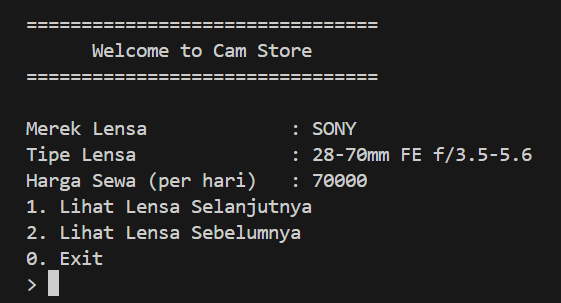


1. List Kamera

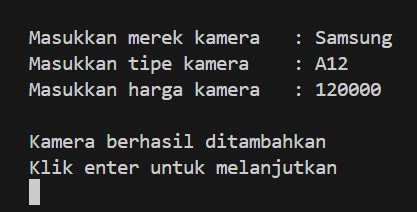
Ini merupakan fitur untuk menampilkan data dari kamera yang terdapat pada toko ini. Tidak hanya itu saja, fitur ini cukup informatif dan dilengkapi tampilan interaktif yang akan memudahkan kita dalam melihat data kamera yang ada.



1. List Lensa

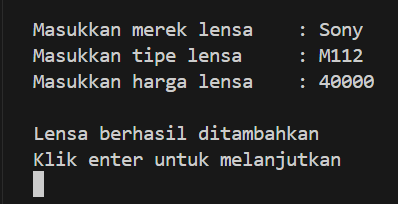
Sama seperti fitur sebelumnya, fitur ini menampilkan data dari lensa yang tersedia di toko ini. Tampilannya pun cukup informatif dan juga interaktif. 

1. Input Kamera

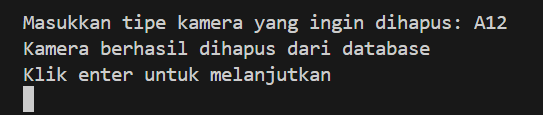
Fitur ini berfungsi untuk memasukkan data kamera baru ke dalam toko. Proses ini memerlukan data berupa merk kamera, tipe kamera, dan juga harga sewa kamera. 

1. Input Lensa

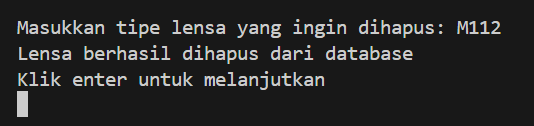
Fitur ini berfungsi untuk memasukkan data lensa baru ke dalam toko. Proses ini memerlukan data berupa merk lensa, tipe lensa, dan juga harga sewa lensa.



1. Hapus Kamera

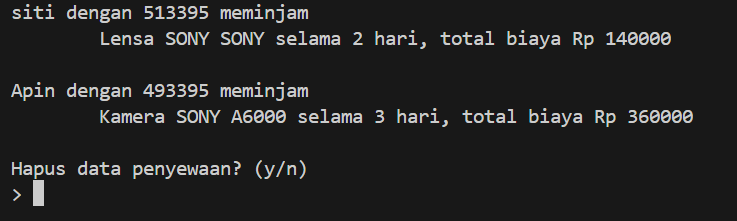
Fitur ini dapat menghapus data kamera berdasarkan tipe dari kamera yang ingin dihapus. 

1. Hapus Lensa

Fitur ini dapat menghapus data lensa berdasarkan tipe lensa yang ingin dihapus. 

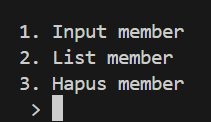
2. Proses Penyewaan

Fitur ini menampilkan segala bentuk penyewaan kamera dan lensa. Fitur ini disertai juga dengan nama penyewa, id penyewa, lama peminjaman, dan harga yang harus dibayar oleh penyewa.



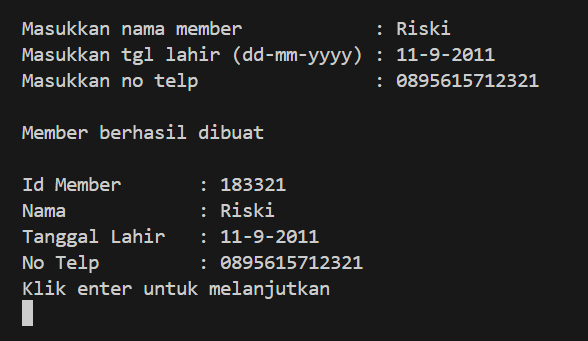
3. Pengurusan Member

Fitur berfungsi untuk mengatur segala macam data member yang ada. Dilengkapi dengan 3 fitur yaitu :



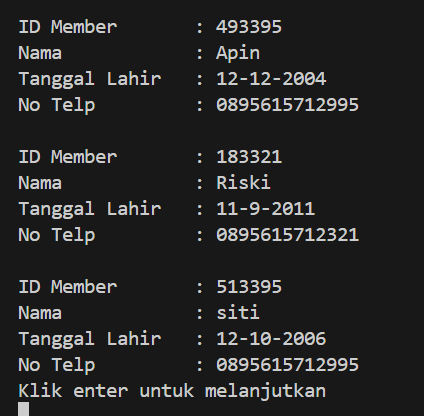
A. Input member

Fitur ini berfungsi untuk memasukkan data member baru dan memerlukan berbagai macam informasi seperti berikut :



B. List Member

Fitur ini menampilkan daftar member yang ada pada toko ini.



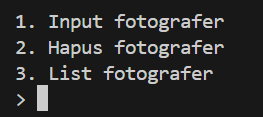
C. Hapus Member

Fitur ini berfungsi untuk menghapus member berdasarkan nama yang diinputkan



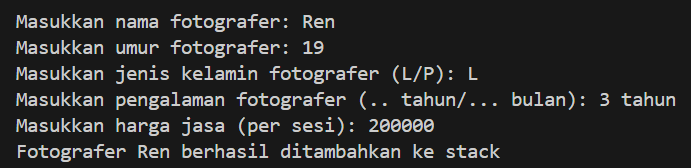
4. Fotografer

Fitur ini berfungsi untuk mengatur segala macam data fotografer yang ada. Dilengkapi dengan 3 fitur yaitu :



1. Input Fotografer

Fitur ini berfungsi untuk menambahkan data fotografer



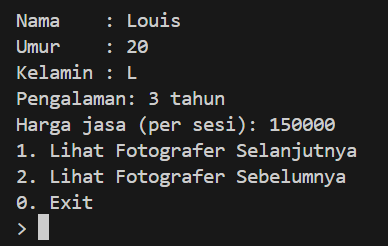
1. Hapus Fotografer

Fitur ini berfungsi menghapus fotografer berdasarkan nama yang telah diinputkan.



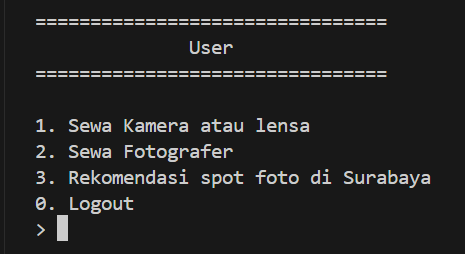
1. List Fotografer

Fitur ini berfungsi menampilkan semua data fotografer dan memiliki tampilan yang informatif serta interaktif pada waktu penggunaannya.

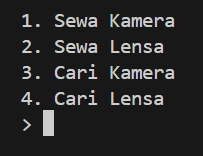


**User**

*Roles* dari seorang user adalah untuk melakukan aktivitas penyewaan kamera, lensa, dan juga fotografer. User tentunya tidak memiliki wewenang untuk melakukan manipulasi data apapun berbeda dengan admin. Ketika memilih untuk login sebagai user, maka akan muncul tampilan seperti ini :

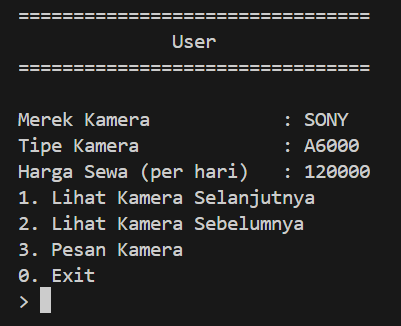


1. Sewa Kamera atau Lensa

Fitur ini terdiri dari fitur fitur lain di dalamnya yang berfungsi untuk membantu user dalam melakukan penyewaan. 

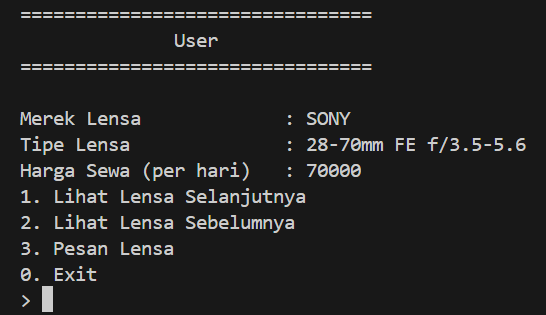
1. Sewa Kamera

Fitur ini berfungsi untuk melakukan penyewaan kamera. Disertai juga tampilan yang informatif dan interaktif yang akan mempermudah penyewaan kamera.



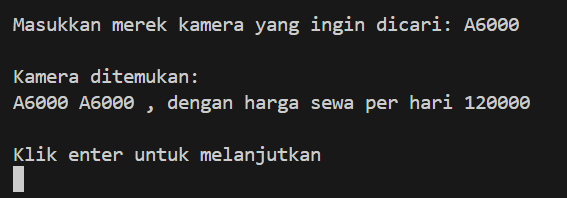
1. Sewa Lensa

Fitur ini berfungsi untuk melakukan penyewaan lensa. Disertai juga tampilan yang informatif dan interaktif yang akan mempermudah penyewaan lensa.

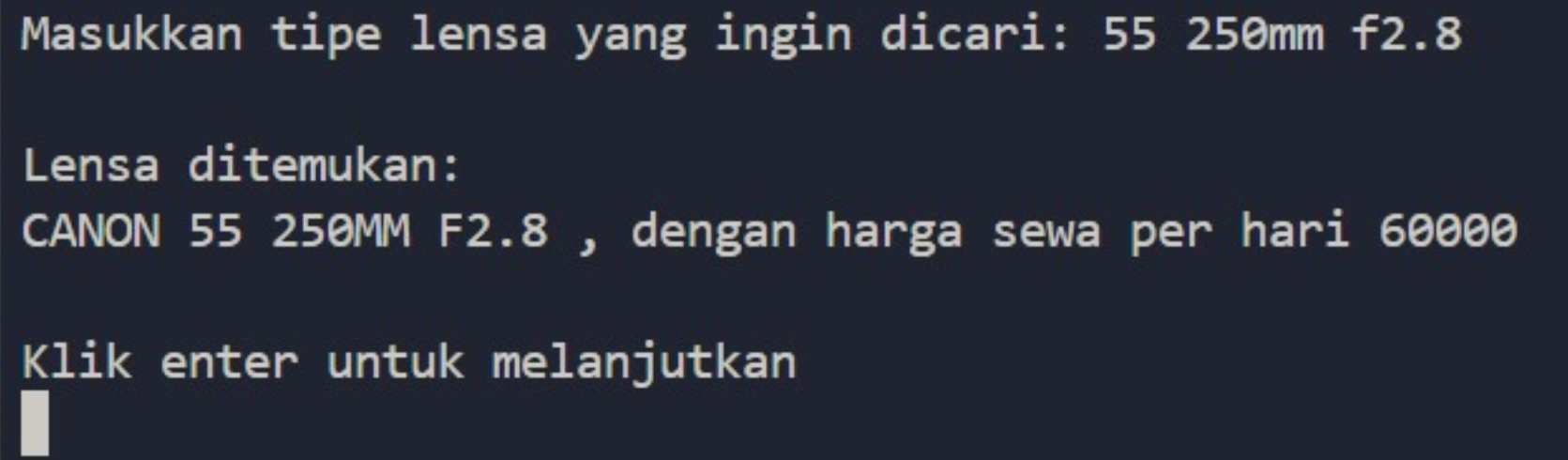


1. Cari Kamera

Fitur ini berfungsi mencari kamera berdasarkan tipe yang telah diinputkan.

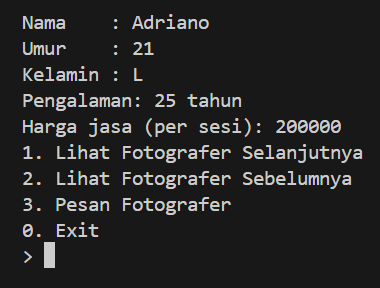


1. Cari Lensa

Fitur ini berfungsi mencari lensa berdasarkan

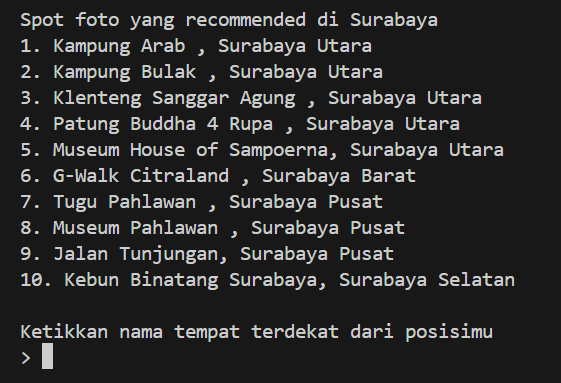
2. Sewa Fotografer

Fitur ini dapat digunakan oleh user untuk menyewa fotografer. Terdapat tampilan yang informatif serta interaktif yang dapat mempermudah user untuk melakukan penyewaan fotografer dikarenakan memiliki detail informasi masing-masing fotografer.



3. Rekomendasi Spot Foto di Surabaya

Fitur ini berfungsi untuk menampilkan rekomendasi spot foto kepada user.



Berdasarkan tampilan di atas, telah disajikan 10 rekomendasi spot foto yang terletak di berbagai wilayah di Surabaya. Kemudian program meminta user untuk memasukkan nama tempat spot foto tersebut yang terdekat dengan posisinya. Kemudian program akan menampilkan urutan rute terbaik untuk mengunjungi keseluruhan tempat di atas.

