

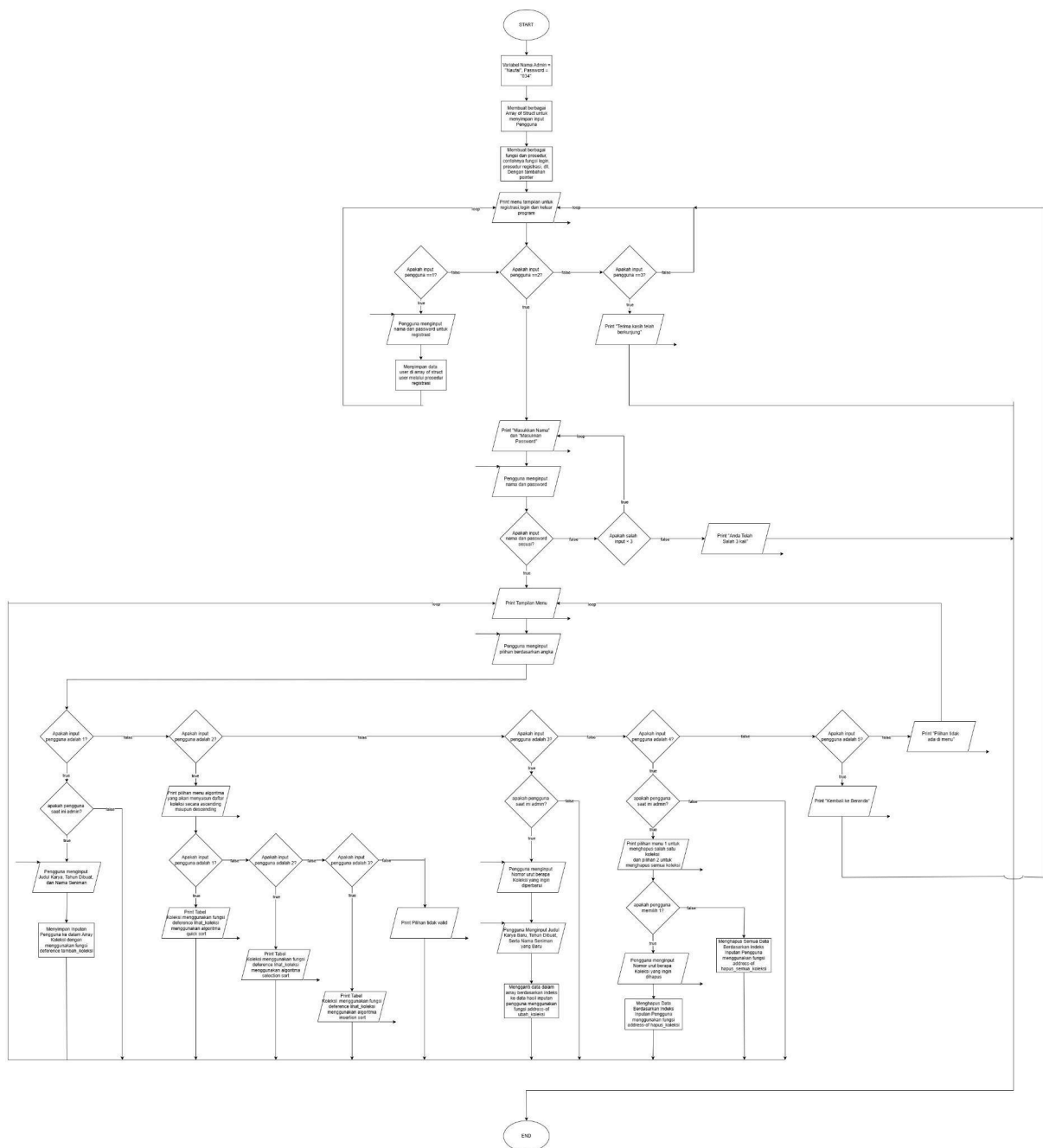
LAPORAN PRAKTIKUM
POSTTEST 6
ALGORITMA PEMROGRAMAN LANJUT



Disusun oleh:
Muhammad Naufal Rifyan Ilham (2409106034)
Kelas (A2 '24)

PROGRAM STUDI INFORMATIKA
UNIVERSITAS MULAWARMAN
SAMARINDA
2025

1. Flowchart



Gambar 1 Flowchart

2. Analisis Program

2.1 Deskripsi Singkat Program

Program ini bertujuan untuk Memanajemen berbagai data yang berkaitan dengan Koleksi/Karya Seni yang ada di Museum Louvre, dimana pengguna dapat Menambahkan, Melihat, Mengupdate, Serta Menghapus berbagai data Koleksi yang ada di Museum Louvre. Sebagai tambahan, dalam program ini juga bersifat multiuser, dimana terdapat Admin yang memiliki seluruh akses dan User yang hanya memiliki akses yang terbatas. Lalu, berbagai data input dari pengguna akan disimpan ke dalam array of struct, juga fungsi dan prosedur yang akan menjadi sub program untuk diaplikasikan ke main program hingga berbagai reference dan pointer yang juga akan diterapkan ke dalam fungsi. Terakhir, ditambahkan beberapa algoritma agar dapat membantu penyusunan didalam fungsi program

3. Source Code

A. Fitur Login (Fungsi Deference)

Fitur ini digunakan untuk melakukan login agar pengguna dapat menggunakan program. Fitur ini mengharuskan pengguna untuk menginput nama dan password yang sesuai

Source Code:

```
bool login_user(khusus Admin[], int jumlah_admin, registrasi
regis[], int jumlah_regis, bool *user){
    string input_nama, input_password;
    int salah = 0;
    bool login_sukses = false;
    do {
        cout << "Masukkan Nama anda : ";
        cin.ignore();
        getline(cin, input_nama);
        cout << "Masukkan Password anda : ";
        getline(cin, input_password);

        for (int i = 0; i < jumlah_admin; i++) {
            if (input_nama == Admin[i].nama_admin &&
input_password == Admin[i].pass_admin) {
                cout << "Berhasil Login Sebagai Admin!" << endl;
                *user = true;
                login_sukses = true;
            }
        }

        for (int i = 0; i < jumlah_regis; i++) {
            if (input_nama == regis[i].nama_user && input_password
== regis[i].pass_user) {
                cout << "Berhasil Login Sebagai User!" << endl;
                *user = true;
                login_sukses = true;
            }
        }

        if(!login_sukses) {
            salah++;
        }
    } while (salah < 3);
}
```

```

        cout << "Nama atau Password salah! Percobaan tersisa "
<< 3 - salah << endl;
    }

    } while (!login_sukses && salah < 3);

    return login_sukses;
}

```

Gambar 3A Fitur Login

B. Fitur Registrasi (Prosedur)

Fitur berfungsi untuk melakukan registrasi sebagai user yang akan disimpan di array of struct

Source Code:

```

void registrasi_user(registrasi regis[], int &jumlah_regis){
    cout << "Silahkan isi prosedur berikut!" << endl;
    cout << "Masukkan nama anda : ";
    cin.ignore();
    getline(cin, regis[jumlah_regis].nama_user);
    cout << "Masukkan password anda : ";
    getline(cin, regis[jumlah_regis].pass_user);
    jumlah_regis++;
    cout << "Registrasi Berhasil!" << endl;
}

```

Gambar 3B Fitur Registrasi

C. Fitur Menu

Fitur ini akan menampilkan menu pertama yang berisi registrasi, login, dan keluar dari program serta menu kedua yang berisi pilihan CRUD untuk manajemen koleksi di Museum Louvre

Source Code:

```

void menu_awal(string pilihan_menu){
    if (pilihan_menu == "menu_pertama"){
        cout << R"(

```

```

=====
                Selamat datang di Museum Louvre
=====
1. Registrasi User Baru
2. Login ke Museum Louvre
3. Keluar dari Program
=====
)" << endl;
}

else{
    cout << R"(
=====
                Selamat datang di Museum Louvre
=====
1. Menambah Koleksi di Museum Louvre
2. Lihat Koleksi di Museum Louvre
3. Memperbarui Koleksi di Museum Louvre
4. Menghapus Koleksi di Museum Louvre
5. Kembali ke menu sebelumnya
=====
)" << endl;
}
}
}

```

Gambar 3C Fitur Menu

D. Fitur Menambah data (Fungsi Deference)

Fitur ini berfungsi untuk menambahkan data kedalam array Koleksi.

Source Code:

```

bool tambah_koleksi(Koleksi koleksi[], int *jumlah_elemen){
    if (*jumlah_elemen >= MAX_JUMLAH) {
        return false;
    }

    else{

        cout << "Masukkan Judul Karya: ";

```

```

        cin.ignore();
        getline(cin, koleksi[*jumlah_elemen].Judul_karya);
        cout << "Masukkan Tanggal Dibuat (Angka): ";
        cin >> koleksi[*jumlah_elemen].detail.hari;
        cout << "Masukkan Bulan Dibuat (Angka): ";
        cin >> koleksi[*jumlah_elemen].detail.bulan;
        cout << "Masukkan Tahun Dibuat : ";
        cin >> koleksi[*jumlah_elemen].detail.tahun;
        cout << "Masukkan Nama Seniman: ";
        cin.ignore();
        getline(cin, koleksi[jumlah_elemen].Nama_seniman);
        (*jumlah_elemen)++;
        return true;
    }
}

```

Gambar 3D Fitur Menambah data

E. Fitur Menampilkan Data Berbentuk Tabel (Fungsi Deference)

Fitur ini berfungsi untuk menampilkan data didalam array Koleksi yang berbentuk tabel

Source Code:

```

bool lihat_koleksi(Koleksi koleksi[], int *jumlah_elemen, bool lengkap){
    if(*jumlah_elemen > 0){
        cout <<
        "===== " <<
        endl;
        cout << "| No | " << left << setw(26) << "Judul Karya" << " | " <<
        setw(7) << "Waktu Dibuat" << " | " << setw(18) << "Seniman" << " |" << endl;
        cout <<
        "===== " <<
        endl;

        for (int i = 0; i < *jumlah_elemen; i++){
            cout << "| " << setw(2) << i+1 << " | " << left << setw(26)
            << koleksi[i].Judul_karya << " | "
            << setw(2) << koleksi[i].detail.hari << "/"
            << setw(2) << koleksi[i].detail.bulan << "/"
            << setw(5) << koleksi[i].detail.tahun << " | "
            << setw(18) << koleksi[i].Nama_seniman << " |" << endl;
        }

        cout <<
        "===== " <<
    }
}

```

```
endl;
    return true;
} else{
    return false;
}
}
```

Gambar 3E Fitur Menampilkan

F. Fitur Menampilkan Data Berdasarkan Nama Lukisan Saja (Fungsi Deference)

Fitur ini berfungsi untuk menampilkan data koleksi berdasarkan nama lukisan saja tanpa keterangan lain

Source Code:

```
bool lihat_koleksi(Koleksi koleksi[], int *jumlah_elemen){
    if (*jumlah_elemen > 0){
        cout << "Daftar Koleksi" << endl;
        for (int i = 0; i < *jumlah_elemen; i++) {
            cout << i+1 << ". " << koleksi[i].Judul_karya << endl;
        }
        return true;
    } else{
        return false;
    }
}
```

Gambar 3F Fitur Menampilkan Singkat

G. Fitur Mengupdate/Mengubah data (Fungsi Address-of)

Fitur ini berfungsi untuk mengubah data yang ada didalam array

Source Code:

```
bool ubah_koleksi(Koleksi koleksi[], int &jumlah_elemen){
    int index;
```



```

if (!lihat_koleksi(koleksi, &jumlah_elemen)){
    cout << "Tidak ada koleksi di Museum" << endl;
}

cout << "Masukkan nomor koleksi yang ingin diubah/perbarui :
";
cin >> index;

if (index < 1 || index > jumlah_elemen) {
    return false;
}

else {
    index--;

    cout << "Masukkan Judul Karya Baru: ";
    cin.ignore();
    getline(cin, koleksi[index].Judul_karya);

    cout << "Masukkan Tahun Dibuat yang Baru!" << endl;
    cout << "Masukkan Tanggal Dibuat yang baru (Angka): ";
    cin >> koleksi[index].detail.hari;
    cout << "Masukkan Bulan Dibuat yang baru (Angka): ";
    cin >> koleksi[index].detail.bulan;
    cout << "Masukkan Tahun Dibuat yang baru : ";
    cin >> koleksi[index].detail.tahun;

    cout << "Masukkan Nama Seniman Baru: ";
    cin.ignore();
    getline(cin, koleksi[index].Nama_seniman);
    return true;
}
}

```

Gambar 3G Fitur Mengubah data

H. Fitur Menghapus data (Fungsi Address-of)

Fitur ini berfungsi untuk menghapus data yang ada didalam array berdasarkan inputan pengguna

Source Code:

```
bool hapus_koleksi(Koleksi koleksi[], int &jumlah_elemen){
    int index;
    if (!lihat_koleksi(koleksi, &jumlah_elemen)){
        cout << "Tidak ada koleksi di Museum" << endl;
    }

    cout << "Masukkan nomor koleksi yang ingin dihapus : ";
    cin >> index;

    if (index < 1 || index > jumlah_elemen) {
        return false;
    }

    else {
        index--;

        for (int i = index; i < jumlah_elemen - 1; i++) {
            koleksi[i].Judul_karya = koleksi[i+1].Judul_karya;
            koleksi[i].detail.hari = koleksi[i+1].detail.hari;
            koleksi[i].detail.bulan = koleksi[i+1].detail.bulan;
            koleksi[i].detail.tahun = koleksi[i+1].detail.tahun;
            koleksi[i].Nama_seniman = koleksi[i+1].Nama_seniman;
        }
        jumlah_elemen--;
        return true;
    }
}
```

Gambar 3H Fitur Menghapus data

I. Fitur Menghapus Semua data (Fungsi Rekursif & Fungsi Address-of)

Fitur ini berfungsi untuk menghapus seluruh koleksi yang ada di array

Source Code:

```
bool hapus_semua_koleksi(Koleksi koleksi[], int &jumlah_elemen){
    if (jumlah_elemen > 0){
```

```

        jumlah_elemen--;
        hapus_semua_koleksi(koleksi, jumlah_elemen);
        return true;
    } else{
        return false;
    }
}

```

Gambar 3I Fitur Hapus Semua data

J. Prosedur Algoritma Quick Sort (Descending)

Prosedur ini berfungsi untuk mengurutkan daftar koleksi berdasarkan Nama Koleksi yang akan diterapkan di fungsi lihat koleksi tanggal. Prosedur ini menggunakan algoritma Quick Sort.

Source Code:

```

void QuickSortKoleksi(Koleksi koleksi[], int kiri, int kanan) {
    if (kiri >= kanan) return;

    int tengah = kiri + (kanan - kiri) / 2;
    string pivot = koleksi[tengah].Judul_karya;
    int i = kiri, j = kanan;

    while (i <= j) {

        while (koleksi[i].Judul_karya > pivot) {
            i++;
        }

        while (koleksi[j].Judul_karya < pivot) {
            j--;
        }

        if (i <= j) {
            swap(koleksi[i], koleksi[j]);
            i++;
            j--;
        }
    }
}

```

```

    }
}

if (kiri < j) {
    QuickSortKoleksi(koleksi, kiri, j);
}
if (i < kanan) {
    QuickSortKoleksi(koleksi, i, kanan);
}
}

```

Gambar 3J Prosedur Algoritma Quick Sort

K. Prosedur Algoritma Selection Sort (Ascending)

Prosedur ini berfungsi untuk mengurutkan daftar koleksi berdasarkan tanggal pembuatan yang akan diterapkan di fungsi lihat koleksi tanggal. Prosedur ini menggunakan algoritma Selection Sort.

Source Code:

```

void SelectionSortTanggal(Koleksi koleksi[], int panjang) {
    for (int i = 0; i < panjang - 1; i++) {
        int min = i;
        for (int j = i + 1; j < panjang; j++) {
            string tanggal_j1 = koleksi[j].detail.tahun +
koleksi[j].detail.bulan + koleksi[j].detail.hari;
            string tanggal_min = koleksi[min].detail.tahun +
koleksi[min].detail.bulan + koleksi[min].detail.hari;

            if (tanggal_j1 < tanggal_min) {
                min = j;
            }
        }

        if (min != i) {
            swap(koleksi[i], koleksi[min]);
        }
    }
}

```

Gambar 3K Prosedur Algoritma Selection Sort

L. Prosedur Algoritma Insertion Sort (Ascending)

Prosedur ini berfungsi untuk mengurutkan daftar koleksi berdasarkan nama seniman yang akan diterapkan di fungsi lihat koleksi tanggal. Prosedur ini menggunakan algoritma Selection Sort.

Source Code:

```
void InsertionSortSeniman(Koleksi koleksi[], int panjang) {
    for (int i = 1; i < panjang; i++) {
        Koleksi nilai = koleksi[i];
        int j = i - 1;

        while (j >= 0 && koleksi[j].Nama_seniman >
nilai.Nama_seniman) {
            koleksi[j + 1] = koleksi[j];
            j = j - 1;
        }
        koleksi[j + 1] = nilai;
    }
}
```

Gambar 3L Prosedur Algoritma Insertion Sort

M. Fungsi Melihat Koleksi Berdasarkan Nama Koleksi (Descending)

Fitur ini berfungsi untuk melihat daftar koleksi berdasarkan nama koleksi yang diurutkan secara descending menggunakan algoritma Quick Sort.

Source Code:

```
bool lihat_koleksi_descending(Koleksi koleksi[], int
*jumlah_elemen, bool lengkap){
    if(*jumlah_elemen > 0){
        QuickSortKoleksi(koleksi, 0, *jumlah_elemen - 1);
        cout <<
```

```

"=====
=====" << endl;
        cout << "| No | " << left << setw(26) << "Judul Karya" <<
" | " << setw(7) << "Waktu Dibuat" << " | " << setw(18) <<
"Seniman" << " |" << endl;
        cout <<
"=====
=====" << endl;

        for (int i = 0; i < *jumlah_elemen; i++){
            cout << "| " << setw(2) << i+1 << " | " << left <<
setw(26)
            << koleksi[i].Judul_karya << " | "
            << setw(2) << koleksi[i].detail.hari << "/"
            << setw(2) << koleksi[i].detail.bulan << "/"
            << setw(5) << koleksi[i].detail.tahun << " | "
            << setw(18) << koleksi[i].Nama_seniman << " |" <<
endl;
        }

        cout <<
"=====
=====" << endl;
        return true;
    } else{
        return false;
    }
}

```

Gambar 3M Fungsi Lihat Koleksi Berdasarkan Nama Koleksi

N. Fungsi Melihat Koleksi Berdasarkan Tanggal Pembuatan (Ascending)

Fitur ini berfungsi untuk melihat daftar koleksi berdasarkan tanggal pembuatan yang diurutkan secara descending menggunakan algoritma Selection Sort.

Source Code:

```

bool lihat_koleksi_tanggal(Koleksi koleksi[], int *jumlah_elemen,
bool lengkap){
    if(*jumlah_elemen > 0){

```

```

        SelectionSortTanggal(koleksi, *jumlah_elemen);
        cout <<
        "=====
===== " << endl;
        cout << "| No | " << left << setw(26) << "Judul Karya" <<
        " | " << setw(7) << "Waktu Dibuat" << " | " << setw(18) <<
        "Seniman" << " |" << endl;
        cout <<
        "=====
===== " << endl;

        for (int i = 0; i < *jumlah_elemen; i++){
            cout << "| " << setw(2) << i+1 << " | " << left <<
            setw(26)
            << koleksi[i].Judul_karya << " | "
            << setw(2) << koleksi[i].detail.hari << "/"
            << setw(2) << koleksi[i].detail.bulan << "/"
            << setw(5) << koleksi[i].detail.tahun << " | "
            << setw(18) << koleksi[i].Nama_seniman << " |" <<
            endl;
        }

        cout <<
        "=====
===== " << endl;
        return true;
    } else{
        return false;
    }
}

```

Gambar 3N Fungsi Lihat Koleksi Berdasarkan Tanggal Pembuatan

O. Fungsi Melihat Koleksi Berdasarkan Nama Seniman (Ascending)

Fitur ini berfungsi untuk melihat daftar koleksi berdasarkan nama seniman yang diurutkan secara descending menggunakan algoritma Insertion Sort.

Source Code:

```

bool lihat_koleksi_seniman(Koleksi koleksi[], int *jumlah_elemen,

```

```

bool lengkap){
    if(*jumlah_elemen > 0){
        InsertionSortSeniman(koleksi, *jumlah_elemen);
        cout <<
        "=====
===== " << endl;
        cout << "| No | " << left << setw(26) << "Judul Karya" <<
        " | " << setw(7) << "Waktu Dibuat" << " | " << setw(18) <<
        "Seniman" << " |" << endl;
        cout <<
        "=====
===== " << endl;

        for (int i = 0; i < *jumlah_elemen; i++){
            cout << "| " << setw(2) << i+1 << " | " << left <<
            setw(26)
            << koleksi[i].Judul_karya << " | "
            << setw(2) << koleksi[i].detail.hari << "/"
            << setw(2) << koleksi[i].detail.bulan << "/"
            << setw(5) << koleksi[i].detail.tahun << " | "
            << setw(18) << koleksi[i].Nama_seniman << " |" <<
            endl;
        }

        cout <<
        "=====
===== " << endl;
        return true;
    } else{
        return false;
    }
}

```

Gambar 30 Fungsi Lihat Koleksi Berdasarkan Nama Seniman

4. Uji Coba dan Hasil Output

Berikut beberapa screenshot hasil dari berbagai skenario pada program:

4.1 Hasil Output

```
Masukkan Nama anda : Naufal
Masukkan Password anda : 034
Berhasil Login Sebagai Admin!
```

Gambar 4.1 Login Sebagai Admin

```
Silahkan isi prosedur berikut!
Masukkan nama anda : Castorice
Masukkan password anda : Karbit123
Registrasi Berhasil!
```

Gambar 4.1 Registrasi

```
Masukkan Nama anda : Castorice
Masukkan Password anda : Karbit123
Berhasil Login Sebagai User!
```

Gambar 4.1 Login Sebagai User

```
Fitur ini hanya bisa digunakan oleh Admin
```

Gambar 4.1 Fitur Khusus Admin

```
Masukkan Nama anda : adsa
Masukkan Password anda : asd
Masukkan Nama anda : ad
Masukkan Password anda : fadf
Masukkan Nama anda : fafe
Masukkan Password anda : adf
Anda telah salah 3 kali
PS C:\praktikum-apl\post-test\post-test-2> █
```

Gambar 4.1 Salah saat Login

```
Masukkan Judul Karya: Mona Lisa
Masukkan Tanggal Dibuat (Angka): 13
Masukkan Bulan Dibuat (Angka): 06
Masukkan Tahun Dibuat : 1503
Masukkan Nama Seniman: Leonardo Da Vinci
Karya berhasil ditambahkan
```

Gambar 4.1 Menambahkan Karya

No	Judul Karya	Waktu Dibuat	Seniman
1	Mona Lisa	13/06/1503	Leonardo Da Vinci

Gambar 4.1 Hasil Tabel

```
Daftar Koleksi
1. Mona Lisa
Masukkan nomor koleksi yang ingin diubah/perbarui : 1
Masukkan Judul Karya Baru: Virgin of the Rocks
Masukkan Tahun Dibuat yang Baru!
Masukkan Tanggal Dibuat yang baru (Angka): 01
Masukkan Bulan Dibuat yang baru (Angka): 08
Masukkan Tahun Dibuat yang baru : 1487
Masukkan Nama Seniman Baru: Leonardo Da Vinci
Koleksi berhasil diperbarui
```

Gambar 4.1 Perbarui karya

No	Judul Karya	Waktu Dibuat	Seniman
1	Virgin of the Rocks	13/06/1503	Leonardo Da Vinci

Gambar 4.1 Hasil Tabel (Setelah diubah)

```
Daftar Koleksi
1. The Starry Night
Masukkan nomor koleksi yang ingin dihapus : 1
Koleksi Berhasil di Hapus
```

Gambar 4.1 Hapus Karya

```
Terima Kasih telah berkunjung  
PS C:\praktikum-apl\post-test\post-test-2> █
```

Gambar 4.1 Exit

```
Pilihan tidak ada di menu  
  
=====
```

Selamat datang di Museum Louvre

```
=====
```

1. Menambah Koleksi di Museum Louvre
2. Lihat Koleksi di Museum Louvre
3. Memperbarui Koleksi di Museum Louvre
4. Menghapus Koleksi di Museum Louvre
5. Keluar dari program

```
=====
```

Masukkan Pilihan : █

Gambar 4.1 Pilihan Invalid

```
Daftar Koleksi  
1. Mona Lisa  
Masukkan nomor koleksi yang ingin diubah/perbarui : 3  
Nomor tidak valid
```

Gambar 4.1 Nomor tidak Valid

```
Semua koleksi Berhasil di Hapus
```

Gambar 4.1 Hapus Semua Koleksi

```
1. Urutkan Berdasarkan Descending (Huruf)  
2. Urutkan Berdasarkan Tanggal Waktu Pembuatan (Ascending)  
3. Urutkan Berdasarkan Nama Seniman (Ascending)  
  
Masukkan Pilihan : █
```

Gambar 4.1 Pilihan Menu Lihat Koleksi

No	Judul Karya	Waktu Dibuat	Seniman
1	The Starry Night	24/06/1889	Vincent Van Gogh
2	The Lacemaker	01/05/1669	Johannes Vermeer
3	Mona Lisa	13/06/1503	Leonardo Da Vinci

Gambar 4.1 Lihat Koleksi Berdasarkan Nama Koleksi

No	Judul Karya	Waktu Dibuat	Seniman
1	Mona Lisa	13/06/1503	Leonardo Da Vinci
2	The Lacemaker	01/05/1669	Johannes Vermeer
3	The Starry Night	24/06/1889	Vincent Van Gogh

Gambar 4.1 Lihat Koleksi Berdasarkan Tanggal Pembuatan

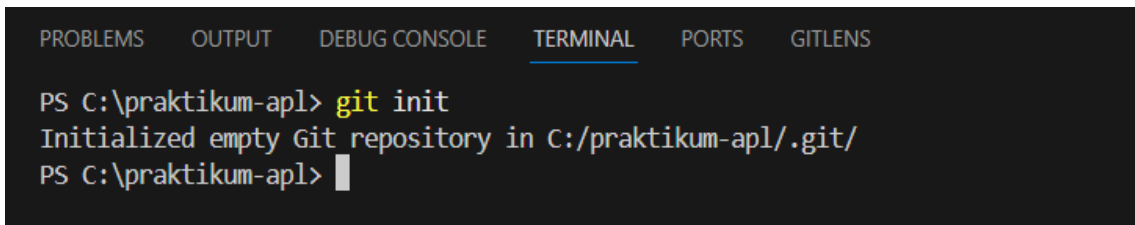
No	Judul Karya	Waktu Dibuat	Seniman
1	The Lacemaker	01/05/1669	Johannes Vermeer
2	Mona Lisa	13/06/1503	Leonardo Da Vinci
3	The Starry Night	24/06/1889	Vincent Van Gogh

Gambar 4.1 Lihat Koleksi Berdasarkan Nama Seniman

5. Langkah Langkah GIT

1. Git Init

Git init merupakan langkah pertama untuk memulai git, Perintah git init digunakan untuk membuat sebuah direktori bernama .git di dalam proyek kita.

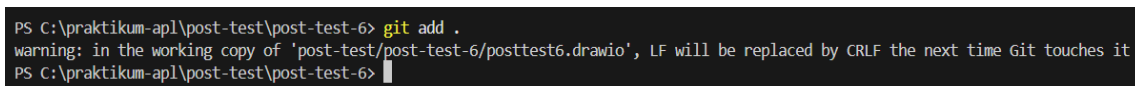


```
PS C:\praktikum-apl> git init
Initialized empty Git repository in C:/praktikum-apl/.git/
PS C:\praktikum-apl> █
```

Gambar 5.1 Git Init

2. Git Add

Selanjutnya adalah Git Add, Perintah ini digunakan untuk menambahkan file baru pada repository yang akan dipilih. Gambar dibawah ini menggunakan perintah “Git Add .” untuk menambah semua file ke dalam repository. Selain itu, kita juga dapat menggunakan “Git Add <nama file>” untuk menambahkan file tertentu saja.

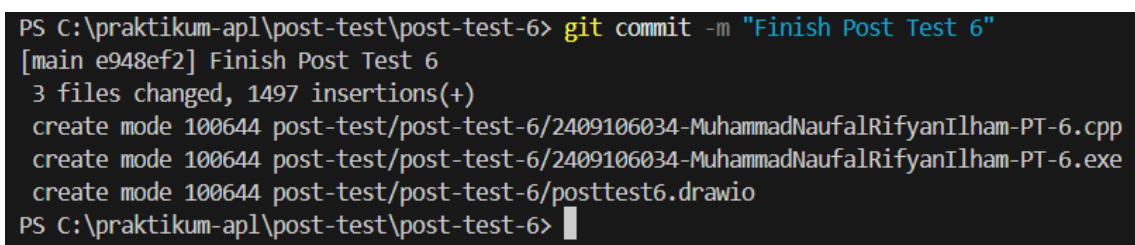


```
PS C:\praktikum-apl\post-test\post-test-6> git add .
warning: in the working copy of 'post-test/post-test-6/posttest6.drawio', LF will be replaced by CRLF the next time Git touches it
PS C:\praktikum-apl\post-test\post-test-6> █
```

Gambar 5.2 Git Add

3. Git Commit

Setelah menambah file pada repository, maka langkah selanjutnya adalah Git Commit. Git Commit adalah perintah Git untuk menyimpan perubahan versi revisi pada repository Git. Jadi, setiap kita melakukan “commit”, Git akan membuat dan menyimpan history revisi pada repository proyek kita.

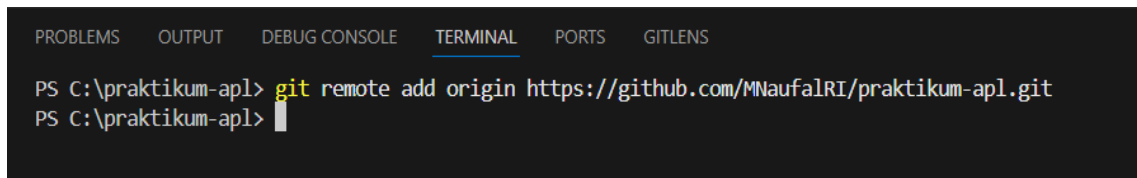


```
PS C:\praktikum-apl\post-test\post-test-6> git commit -m "Finish Post Test 6"
[main e948ef2] Finish Post Test 6
3 files changed, 1497 insertions(+)
create mode 100644 post-test/post-test-6/2409106034-MuhammadNaufalRifyanIlham-PT-6.cpp
create mode 100644 post-test/post-test-6/2409106034-MuhammadNaufalRifyanIlham-PT-6.exe
create mode 100644 post-test/post-test-6/posttest6.drawio
PS C:\praktikum-apl\post-test\post-test-6> █
```

Gambar 5.3 Git Commit

4. Git Remote

Langkah selanjutnya adalah Git Remote, git remote adalah perintah dalam Git yang digunakan untuk mengelola daftar repositori jarak jauh (remote repositories) yang terhubung dengan repositori lokal.

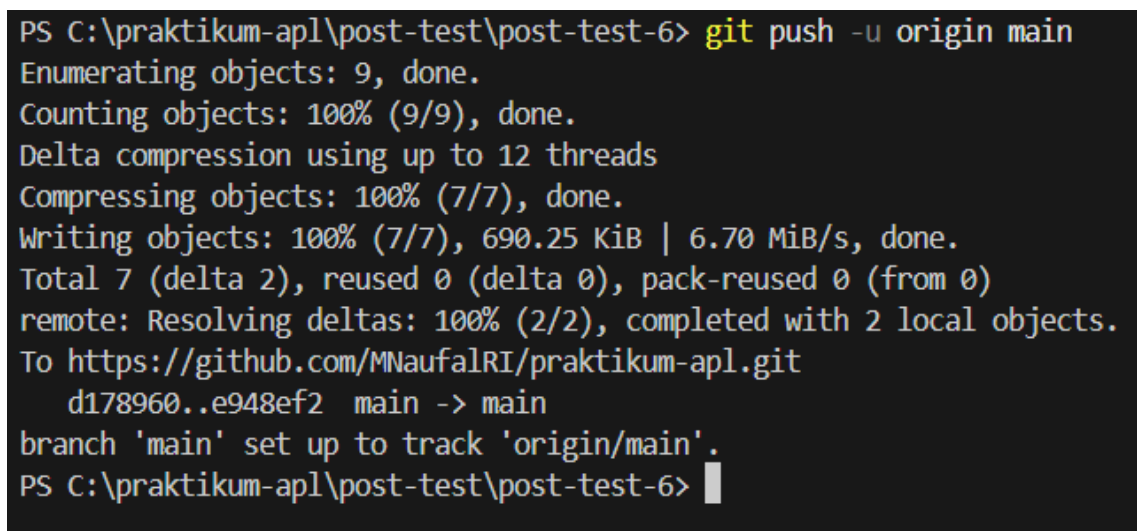


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  
PS C:\praktikum-apl> git remote add origin https://github.com/MNaufalRI/praktikum-apl.git  
PS C:\praktikum-apl>
```

Gambar 5.4 Git Remote

5. Git Push

Langkah yang terakhir adalah Git Push, dimana perintah ini akan berfungsi untuk mengirimkan perubahan ke master branch dari remote repository yang berhubungan dengan direktori kerja.



```
PS C:\praktikum-apl\post-test\post-test-6> git push -u origin main  
Enumerating objects: 9, done.  
Counting objects: 100% (9/9), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (7/7), done.  
Writing objects: 100% (7/7), 690.25 KiB | 6.70 MiB/s, done.  
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
To https://github.com/MNaufalRI/praktikum-apl.git  
d178960..e948ef2  main -> main  
branch 'main' set up to track 'origin/main'.  
PS C:\praktikum-apl\post-test\post-test-6>
```

Gambar 5.5 Git Push