LAPORAN PRAKTIKUM

POSTTEST 5

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Dimas Firjatullah Islamay (2409106057)

Kelas (B1 ‘24)

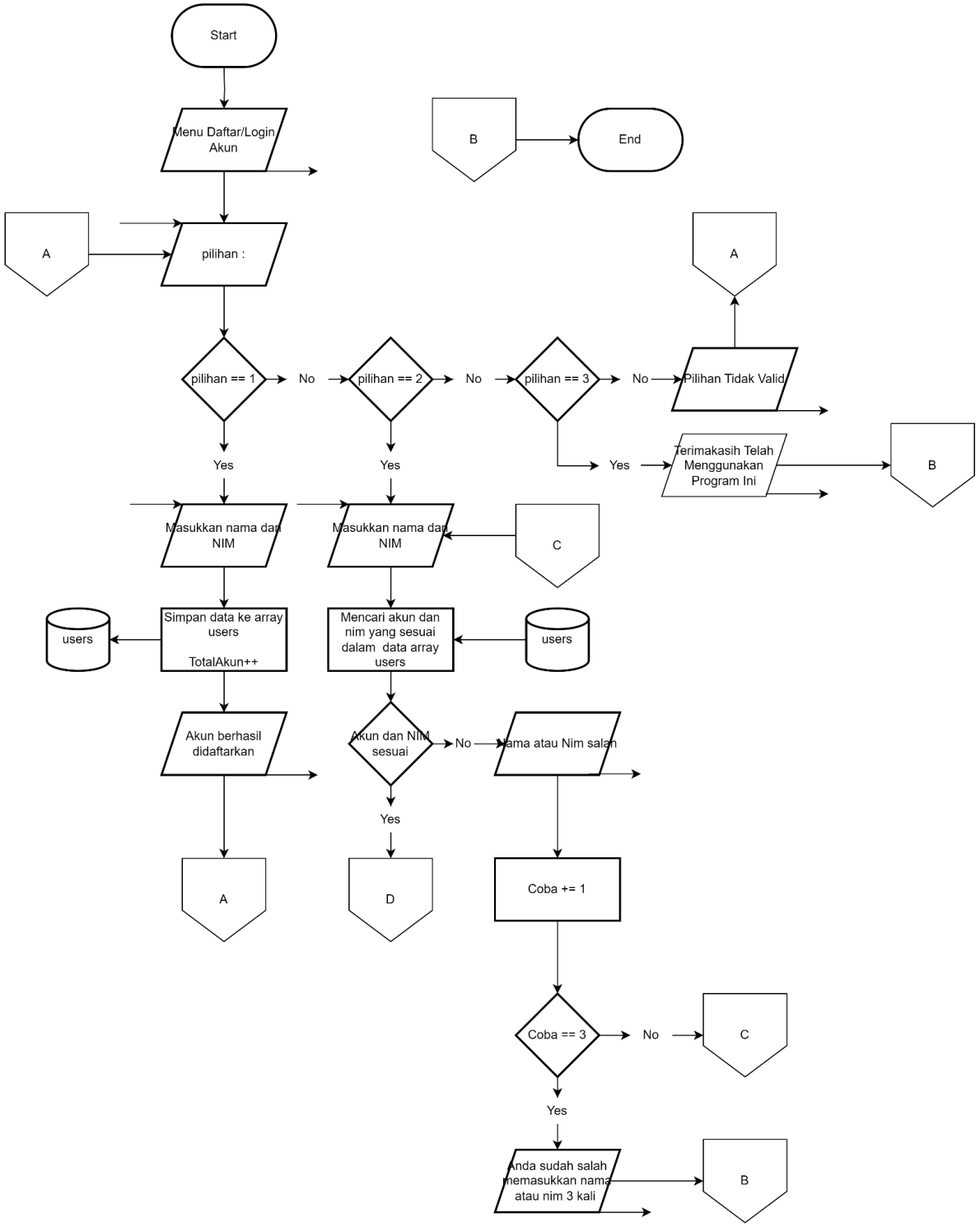
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

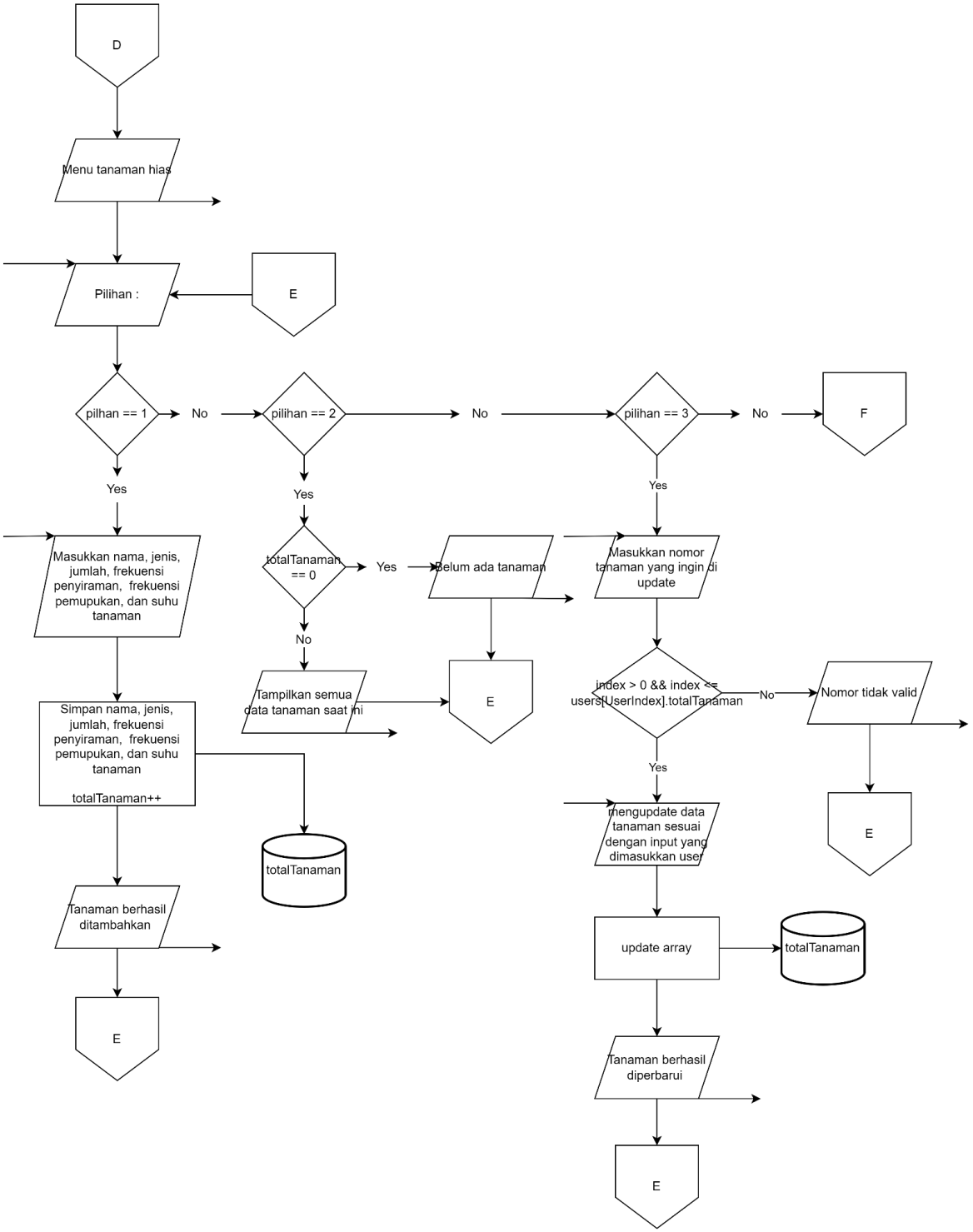
SAMARINDA

2025

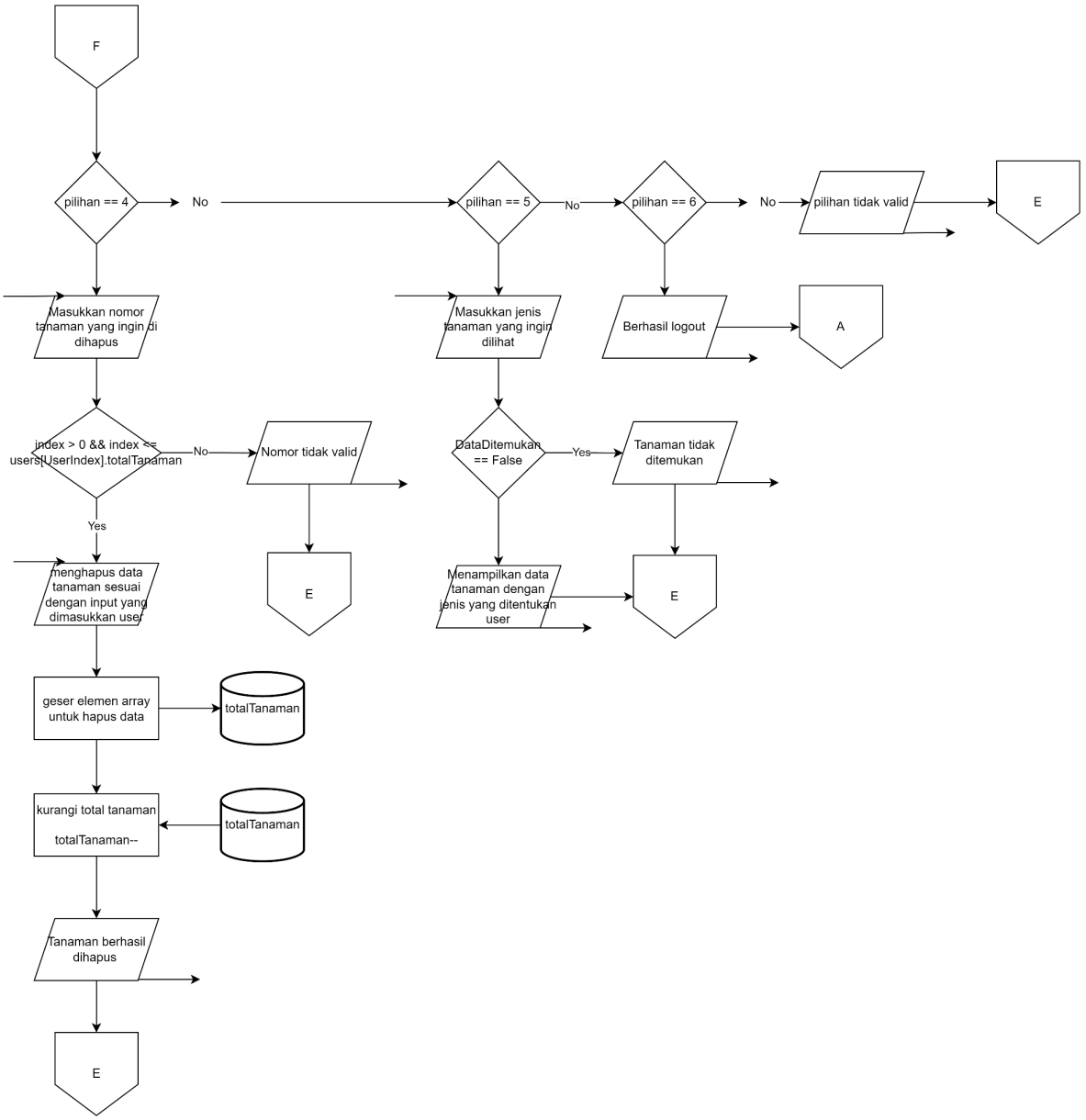
1. Flowchart



Gambar 1.1 Fitur Daftar, Login Akun dan Keluar Program



Gambar 1. 2 Fitur Tambah, Tampilkan, Dan Update Tanaman



Gambar 1.3 Fitur Hapus Tanaman, Cari Tanaman Berdasarkan Jenis dan Logout

2. Analisis Program

2.1 Deskripsi Singkat Program

Program ini merupakan program manajemen perawatan tanaman hias yang berfungsi untuk mendata nama, jenis, jumlah, frekuensi penyiraman, frekuensi pemupukan, dan suhu ideal tanaman dari seseorang yang suka mengoleksi tanaman hias. Program ini menerapkan beberapa fitur diantaranya seperti fitur daftar akun dan login akun yang dimana user harus registrasi akun dengan nama sebagai username dan NIM sebagai password lalu mencocokkan username dan password yang sudah diatur, fitur CRUD yang dilengkapi index, terdapat array of struct atau nested struct sebagai tempat untuk mendata tanaman, terdapat penggunaan beberapa fungsi dengan parameter dan prosedur tiap fitur fitur yang ada di dalam program, memakai satu fungsi overloading untuk mencari tanaman hanya berdasarkan jenisnya, memakai pointer dan terdapat beberapa fungsi dengan parameter address of dan parameter dereference dan terdapat fitur looping yang dimana program akan mengulang terus kecuali user salah memasukkan username/password sebanyak 3 kali atau user memilih logout kemudian memilih keluar program.

3. Source Code

A. Daftar Akun

Merupakan blok kode yang digunakan untuk menambah akun baru ke dalam data array user dengan cara menginput nama dan NIM dari user lalu disimpan di dalam TotalAkun, kemudian TotalAkun akan bertambah satu karena kita sudah mendaftar satu akun, setelah selesai akan muncul output akun berhasil ditambahkan.

Source Code:

```
void DaftarAkun(User *users, int *TotalAkun)
{
    cout << "Masukkan Nama: ";
    cin >> users[*TotalAkun].nama;
    cout << "Masukkan NIM: ";
    cin >> users[*TotalAkun].nim;
    (*TotalAkun)++;
    cout << "Akun Berhasil Didaftarkan" << endl;
}
```

Gambar 3.1

B. Login Akun

Merupakan blok kode yang digunakan untuk memverifikasi akun user berdasarkan nama dan NIM yang telah diinputkan di blok kode Daftar Akun dengan maksimal 3 kali percobaan. User memasukkan nama dan NIM terlebih dahulu lalu sistem akan mencocokkan inputan user dengan data dalam array users. Jika ternyata cocok maka login berhasil dan program akan lanjut ke Menu Tanaman, jika ternyata tidak cocok maka user bisa mencoba lagi, jika sudah 3 kali percobaan dan tetap salah, maka login gagal dan fungsi berhenti.

Source Code:

```
int Login(User *users, int TotalAkun)
{
    string Nama, NIM;
    int Coba = 0;
    while (Coba < 3)
    {
        cout << "=== Login ===" << endl;
        cout << "Masukkan Nama: ";
        cin >> Nama;
        cout << "Masukkan NIM: ";
```

```
        cin >> NIM;
        for (int i = 0; i < TotalAkun; i++)
        {
            if (users[i].nama == Nama && users[i].nim == NIM)
            {
                cout << "Login Berhasil" << endl;
                return i;
            }
        }
        cout << "Nama Atau NIM Salah" << endl;
        Coba++;
    }
    cout << "Anda Salah Memasukkan Nama Atau NIM 3 Kali" << endl;
    return -1;
}
```

Gambar 3.2

C. Tambah Tanaman

Merupakan blok kode yang digunakan untuk menambahkan data tanaman yang berupa nama, jenis, jumlah, frekuensi penyiraman, frekuensi pemupukan, dan suhu ideal tanaman, lalu data akan disimpan ke dalam array tanaman milik user.

Source Code:

```
void TambahTanaman(User *user)
{
    int index = user->totalTanaman;
    cout << "Masukkan Nama Tanaman: ";
    cin >> user->tanaman[index].nama;
    cout << "Masukkan Jenis Tanaman: ";
    cin >> user->tanaman[index].jenis;
    cout << "Masukkan Jumlah Tanaman: ";
    cin >> user->tanaman[index].jumlah;
    cout << "Masukkan Frekuensi Penyiraman Per Minggu: ";
    cin >> user->tanaman[index].siram;
    cout << "Masukkan Frekuensi Pemupukan Per Bulan: ";
    cin >> user->tanaman[index].pupuk;
    cout << "Masukkan Suhu Tanaman (Celcius): ";
    cin >> user->tanaman[index].suhu;
    user->totalTanaman++;
    cout << "Tanaman Berhasil Ditambahkan" << endl;
}
```

Gambar 3.3

D. Tampilkan Tanaman

Merupakan blok kode yang digunakan untuk menampilkan daftar tanaman user. Jika ternyata tanaman belum ada maka akan menampilkan output belum ada tanaman. Jika ada maka fungsi akan mencetak daftar data tanaman satu per satu yang terdiri dari nama, jenis, jumlah, frekuensi penyiraman, frekuensi pemupukan, dan suhu ideal.

Source Code:

```
void LihatTanaman(const User *user)
{
    if (user->totalTanaman == 0)
    {
        cout << "Belum Ada Tanaman" << endl;
        return;
    }

    cout << "=== DAFTAR TANAMAN ===" << endl;
    for (int i = 0; i < user->totalTanaman; i++)
    {
        cout << i + 1 << ". " << user->tanaman[i].nama << " | "
             << user->tanaman[i].jenis << " | "
             << user->tanaman[i].jumlah << " Tanaman | "
             << user->tanaman[i].siram << " Kali Siram/Minggu | "
             << user->tanaman[i].pupuk << " Kali Pupuk/Bulan | "
             << user->tanaman[i].suhu << " Derajat Celcius" << endl;
    }
}
```

Gambar 3.4

E. Tampilkan Tanaman Berdasarkan Jenis

Merupakan blok kode dengan parameter tambahan jenisfilter yang berfungsi untuk menampilkan hanya tanaman yang sesuai dengan jenis tertentu yang diinputkan oleh user. Jika ternyata jenis tanaman yang dimaksud tidak ada maka akan menampilkan output tanaman tidak ditemukan. Jika ada maka fungsi akan mencetak daftar data tanaman yang memiliki jenis yang sama yang telah diinputkan user.

Source Code:

```
void LihatTanaman(const User *user, string jenisFilter)
{
    bool ditemukan = false;
    cout << "=== Tanaman Dengan Jenis " << jenisFilter << " ===" << endl;
    for (int i = 0; i < user->totalTanaman; i++)
    {
```

```
        if (user->tanaman[i].jenis == jenisFilter)
        {
            cout << i + 1 << ". " << user->tanaman[i].nama << " | "
                 << user->tanaman[i].jenis << " | "
                 << user->tanaman[i].jumlah << " Tanaman | "
                 << user->tanaman[i].siram << " Kali Siram/Minggu | "
                 << user->tanaman[i].pupuk << " Kali Pupuk/Bulan | "
                 << user->tanaman[i].suhu << " Derajat Celcius" << endl;
            ditemukan = true;
        }
    }
    if (!ditemukan)
    {
        cout << "Tanaman Tidak Ditemukan" << endl;
    }
}
```

Gambar 3.5

F. Update Tanaman

Merupakan blok kode yang digunakan untuk mengupdate data dari tanaman yang dipilih oleh user dengan memasukkan nomor tanaman yang ingin di update datanya. Jika nomor yang dimasukkan valid maka program akan meminta inputan ulang terkait nama, jenis, jumlah, frekuensi penyiraman, frekuensi pemupukan, dan suhu untuk di update datanya, jika nomornya tidak valid maka akan muncul output nomor tidak valid.

Source Code:

```
void UpdateTanaman(User *user)
{
    int index;
    cout << "Masukkan Nomor Tanaman Yang Ingin Diupdate: ";
    cin >> index;
    if (index > 0 && index <= user->totalTanaman)
    {
        index--;
        cout << "Masukkan Nama Tanaman Baru: ";
        cin >> user->tanaman[index].nama;
        cout << "Masukkan Jenis Tanaman Baru: ";
        cin >> user->tanaman[index].jenis;
        cout << "Masukkan Jumlah Tanaman Baru: ";
        cin >> user->tanaman[index].jumlah;
        cout << "Masukkan Frekuensi Penyiraman Per Minggu Baru: ";
        cin >> user->tanaman[index].siram;
        cout << "Masukkan Frekuensi Pemupukan Per Bulan Baru: ";
        cin >> user->tanaman[index].pupuk;
        cout << "Masukkan Suhu Tanaman Baru (Celcius): ";
```

```
        cin >> user->tanaman[index].suhu;
        cout << "Tanaman Berhasil Diperbarui" << endl;
    }
    else
    {
        cout << "Nomor Tidak Valid" << endl;
    }
}
```

Gambar 3.6

G. Hapus Tanaman

Blok kode ini digunakan untuk menghapus tanaman berdasarkan nomor yang dimasukkan oleh user. Jika nomor yang dimasukkan valid maka program akan menghapus data tanaman tersebut, jika nomornya tidak valid maka akan muncul output nomor tidak valid.

Source Code:

```
void HapusTanaman(User *user)
{
    int index;
    cout << "Masukkan Nomor Tanaman Yang Ingin Dihapus: ";
    cin >> index;
    if (index > 0 && index <= user->totalTanaman)
    {
        for (int i = index - 1; i < user->totalTanaman - 1; i++)
        {
            user->tanaman[i] = user->tanaman[i + 1];
        }
        user->totalTanaman--;
        cout << "Tanaman Berhasil Dihapus" << endl;
    }
    else
    {
        cout << "Nomor Tidak Valid" << endl;
    }
}
```

Gambar 3.7

H. Menu Tanaman

Merupakan blok kode menu tanaman yang menampilkan 6 pilihan seperti tambah, lihat, update, hapus, lihat berdasarkan jenis tanaman, dan logout yang bisa dipilih user dan akan memasukkan mereka ke blok kode terkait.

Source Code:

```
void MenuTanaman(User *user)
{
    while (true)
    {
        int pilihan;
        cout << "=== MENU TANAMAN HIAS ===" << endl;
        cout << "1. Tambah Tanaman" << endl;
        cout << "2. Lihat Tanaman" << endl;
        cout << "3. Update Tanaman" << endl;
        cout << "4. Hapus Tanaman" << endl;
        cout << "5. Lihat Tanaman Berdasarkan Jenis" << endl;
        cout << "6. Logout" << endl;
        cout << "Pilih: ";
        cin >> pilihan;

        switch (pilihan)
        {
        case 1:
            TambahTanaman(user);
            break;
        case 2:
            LihatTanaman(user);
            break;
        case 3:
            UpdateTanaman(user);
            break;
        case 4:
            HapusTanaman(user);
            break;
        case 5:
        {
            string jenis;
            cout << "Masukkan Jenis Tanaman yang Ingin Dilihat: ";
            cin >> jenis;
            LihatTanaman(user, jenis);
            break;
        }
        case 6:
            cout << "Logout Berhasil" << endl;
            return;
        default:
            cout << "Pilihan Tidak Valid" << endl;
        }
    }
}
```

Gambar 3.8

I. Menu Utama

Merupakan blok kode menu utama yang menampilkan 3 pilihan yaitu daftar, login, dan keluar yang bisa dipilih user dan akan memasukkan mereka ke blok kode terkait. Jika user memilih keluar maka program akan berhenti.

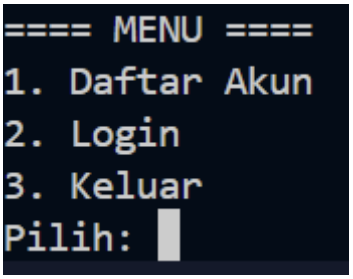
Source Code:

```
int main()
{
    while (true)
    {
        int Pilihan;
        cout << "==== MENU UTAMA ====" << endl;
        cout << "1. Daftar Akun" << endl;
        cout << "2. Login" << endl;
        cout << "3. Keluar" << endl;
        cout << "Pilih: ";
        cin >> Pilihan;

        if (Pilihan == 1)
        {
            DaftarAkun(users, &TotalAkun);
        }
        else if (Pilihan == 2)
        {
            int index = Login(users, TotalAkun);
            if (index != -1)
            {
                MenuTanaman(&users[index]);
            }
            else
            {
                return 0;
            }
        }
        else if (Pilihan == 3)
        {
            cout << "Terima kasih telah menggunakan program ini" << endl;
            break;
        }
        else
        {
            cout << "Pilihan Tidak Valid" << endl;
        }
    }

    return 0;
}
```

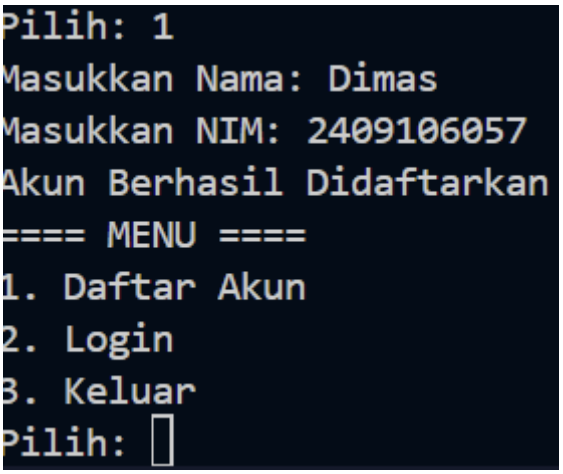
Gambar 3.9

4. Hasil Output

4.1 Hasil Output



Gambar 4.1 Menu Awal Daftar



Gambar 4.2 Daftar Akun

```
=== Login ===
Masukkan Nama: DimasKroco
Masukkan NIM: 2409106057
Nama Atau NIM Salah
=== Login ===
Masukkan Nama: Dimas
Masukkan NIM: 2323232323
Nama Atau NIM Salah
=== Login ===
Masukkan Nama: Fufufafa
Masukkan NIM: 100
Nama Atau NIM Salah
Anda Salah Memasukkan Nama Atau NIM 3 Kali
PS C:\Users\ASUS\OneDrive\Desktop\praktikum-apl\post-test\post-test-4>
```

Gambar 4.3 Login Akun Gagal

```
=== Login ===
Masukkan Nama: Dimas
Masukkan NIM: 2409106057
Login Berhasil
```

Gambar 4.4 Login Berhasil

```
=== MENU TANAMAN HIAS ===
1. Tambah Tanaman
2. Lihat Tanaman
3. Update Tanaman
4. Hapus Tanaman
5. Lihat Tanaman Berdasarkan Jenis
6. Logout
Pilih:
```

Gambar 4.5 Menu Utama Tanaman Hias

```
Pilih: 1
Masukkan Nama Tanaman: Mawar
Masukkan Jenis Tanaman: Bunga
Masukkan Jumlah Tanaman: 20
Masukkan Frekuensi Penyiraman Per Minggu: 3
Masukkan Frekuensi Pemupukan Per Bulan: 2
Masukkan Suhu Tanaman (Celcius): 25
Tanaman Berhasil Ditambahkan
=== MENU TANAMAN HIAS ===
1. Tambah Tanaman
2. Lihat Tanaman
3. Update Tanaman
4. Hapus Tanaman
5. Lihat Tanaman Berdasarkan Jenis
6. Logout
Pilih: 1
Masukkan Nama Tanaman: Teratai
Masukkan Jenis Tanaman: Air
Masukkan Jumlah Tanaman: 5
Masukkan Frekuensi Penyiraman Per Minggu: 0
Masukkan Frekuensi Pemupukan Per Bulan: 1
Masukkan Suhu Tanaman (Celcius): 25
Tanaman Berhasil Ditambahkan
=== MENU TANAMAN HIAS ===
1. Tambah Tanaman
2. Lihat Tanaman
3. Update Tanaman
4. Hapus Tanaman
5. Lihat Tanaman Berdasarkan Jenis
6. Logout
Pilih: 1
Masukkan Nama Tanaman: Melati
Masukkan Jenis Tanaman: Bunga
```

Gambar 4.6 Tambah Tanaman

```
Pilih: 2
=== DAFTAR TANAMAN ===
1. Mawar | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
2. Teratai | Air | 5 Tanaman | 0 Kali Siram/Minggu | 1 Kali Pupuk/Bulan | 25 Derajat Celcius
3. Melati | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
4. Peashooter | Polong | 10 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
5. Matahari | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
```

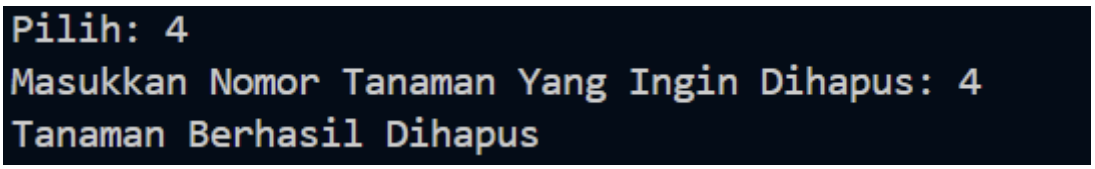
Gambar 4.7 Tampilkan Tanaman

```
Pilih: 3
Masukkan Nomor Tanaman Yang Ingin Diupdate: 4
Masukkan Nama Tanaman Baru: SnowPea
Masukkan Jenis Tanaman Baru: Polong
Masukkan Jumlah Tanaman Baru: 5
Masukkan Frekuensi Penyiraman Per Minggu Baru: 2
Masukkan Frekuensi Pemupukan Per Bulan Baru: 2
Masukkan Suhu Tanaman Baru (Celcius): 27
Tanaman Berhasil Diperbarui
```

Gambar 4.8 Update Tanaman

```
Pilih: 2
=== DAFTAR TANAMAN ===
1. Mawar | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
2. Teratai | Air | 5 Tanaman | 0 Kali Siram/Minggu | 1 Kali Pupuk/Bulan | 25 Derajat Celcius
3. Melati | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
4. SnowPea | Polong | 5 Tanaman | 2 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 27 Derajat Celcius
5. Matahari | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
```

Gambar 4.9 Tanaman Setelah Di Update



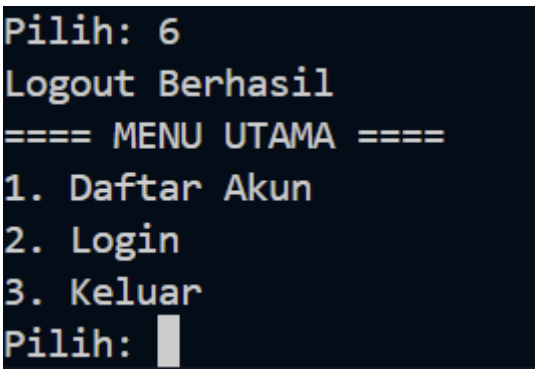
Gambar 4.10 Hapus Tanaman

```
Pilih: 2
=== DAFTAR TANAMAN ===
1. Mawar | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
2. Teratai | Air | 5 Tanaman | 0 Kali Siram/Minggu | 1 Kali Pupuk/Bulan | 25 Derajat Celcius
3. Melati | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
4. Matahari | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
```

Gambar 4.10 Tanaman Setelah Di Hapus

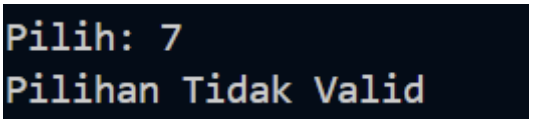
```
Pilih: 5
Masukkan Jenis Tanaman yang Ingin Dilihat: Bunga
=== Tanaman Dengan Jenis Bunga ===
1. Mawar | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
3. Melati | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
4. Matahari | Bunga | 20 Tanaman | 3 Kali Siram/Minggu | 2 Kali Pupuk/Bulan | 25 Derajat Celcius
=== MENU TANAMAN HIAS ===
1. Tambah Tanaman
2. Lihat Tanaman
3. Update Tanaman
4. Hapus Tanaman
5. Lihat Tanaman Berdasarkan Jenis
6. Logout
Pilih: 5
Masukkan Jenis Tanaman yang Ingin Dilihat: Air
=== Tanaman Dengan Jenis Air ===
2. Teratai | Air | 5 Tanaman | 0 Kali Siram/Minggu | 1 Kali Pupuk/Bulan | 25 Derajat Celcius
=== MENU TANAMAN HIAS ===
1. Tambah Tanaman
2. Lihat Tanaman
3. Update Tanaman
4. Hapus Tanaman
5. Lihat Tanaman Berdasarkan Jenis
6. Logout
Pilih: 5
Masukkan Jenis Tanaman yang Ingin Dilihat: Polong
=== Tanaman Dengan Jenis Polong ===
Tanaman Tidak Ditemukan
```

Gambar 4.10 Lihat Tanaman Berdasarkan Jenis



Gambar 4.10 Logout

4.11 Keluar Program



Gambar 4.12 Pilihan Tidak Valid

5. Langkah-langkah GIT

```
PS C:\Users\ASUS\OneDrive\Desktop\praktikum-apl\post-test\post-test-5> git add .
```

Gambar 5.1 Git add

Git add digunakan untuk menambahkan file yang ingin di commit di step selanjutnya

```
PS C:\Users\ASUS\OneDrive\Desktop\praktikum-apl\post-test\post-test-5> git commit -m "Finish Posttest 5"
[main 114ccbc] Finish Posttest 5
 1 file changed, 0 insertions(+), 0 deletions(-)
```

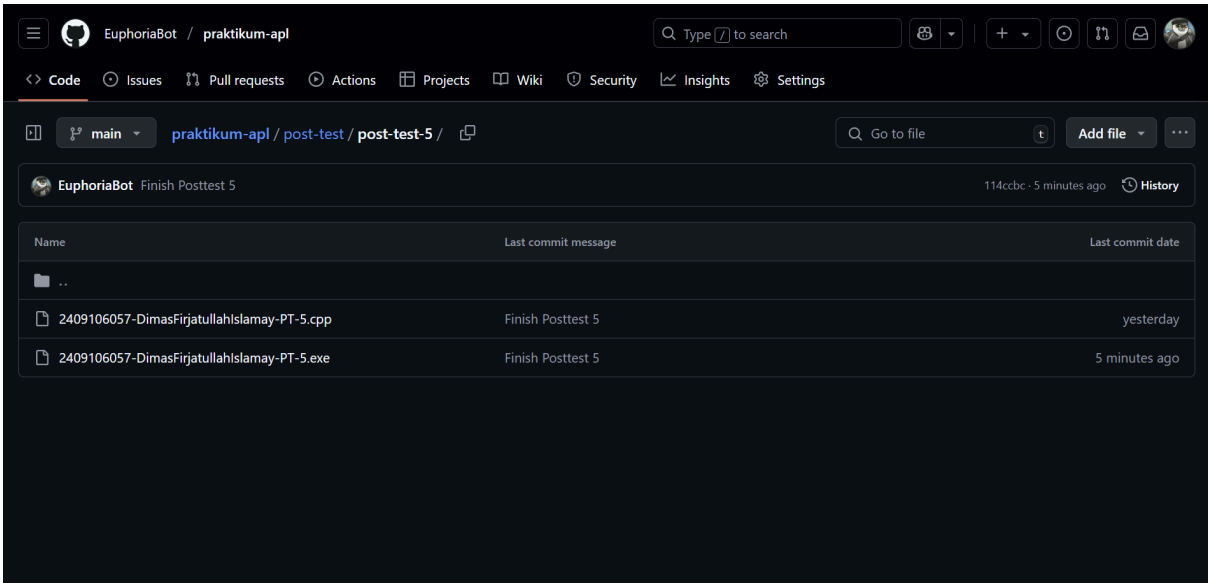
Gambar 5.2 Git commit

Git commit digunakan untuk menyimpan hasil file yang telah ditambahkan

```
PS C:\Users\ASUS\OneDrive\Desktop\praktikum-apl\post-test\post-test-5> git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 22 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 6.01 KiB | 1.00 MiB/s, done.
Total 5 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/EuphoriaBot/praktikum-apl.git
   d2fc335..114ccbc  main -> main
PS C:\Users\ASUS\OneDrive\Desktop\praktikum-apl\post-test\post-test-5>
```

Gambar 5.3 Git push

Git push digunakan untuk mengupload seluruh commit yang sudah dilakukan dalam satu repository lokal



Gambar 5.4 Github

Berikut contoh gambar jika langkah-langkah Git berhasil dilaksanakan