

LAPORAN PRAKTIKUM
POSTTEST 5
ALGORITMA PEMROGRAMAN LANJUT

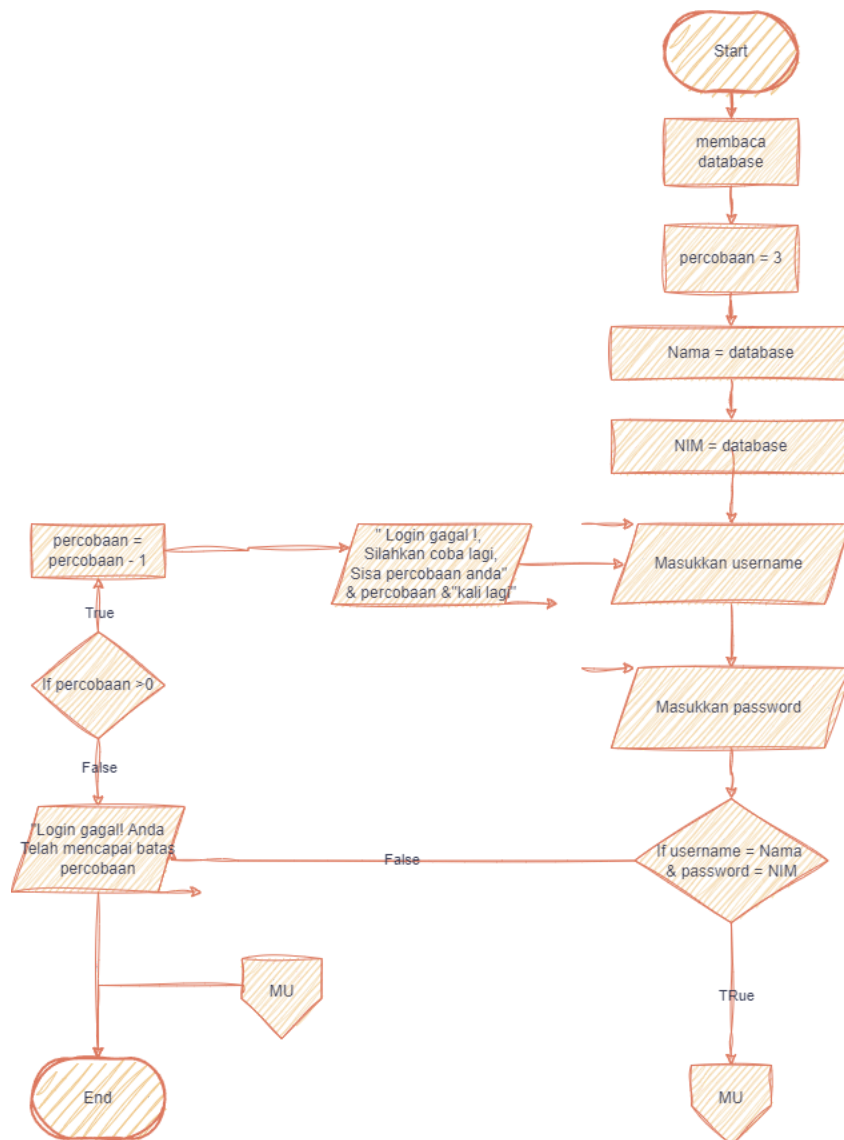


Disusun oleh:
IKHWAN HARIYANTO (2409106082)
Kelas (B2'24)

PROGRAM STUDI INFORMATIKA
UNIVERSITAS MULAWARMAN
SAMARINDA
2025

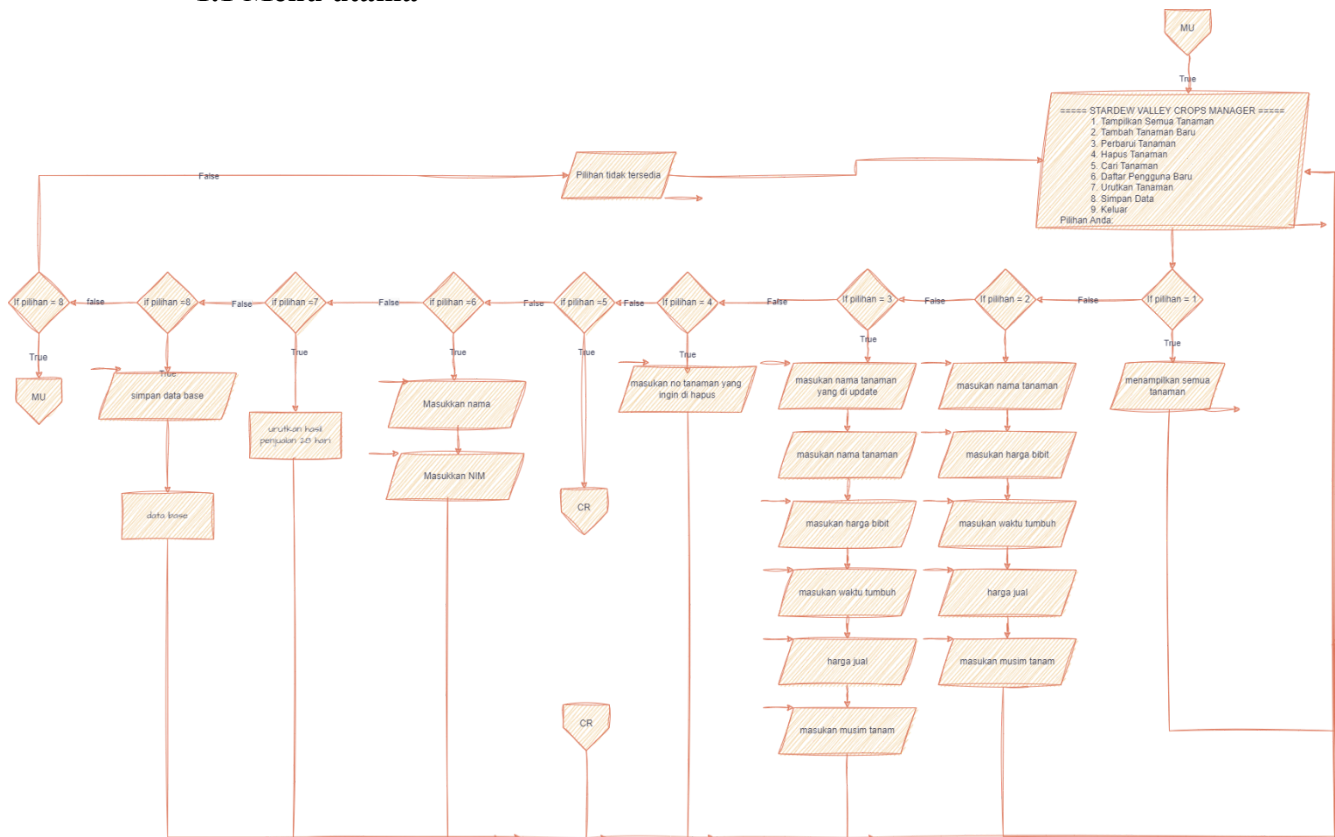
1. Flowchart

1.1 Menu Login



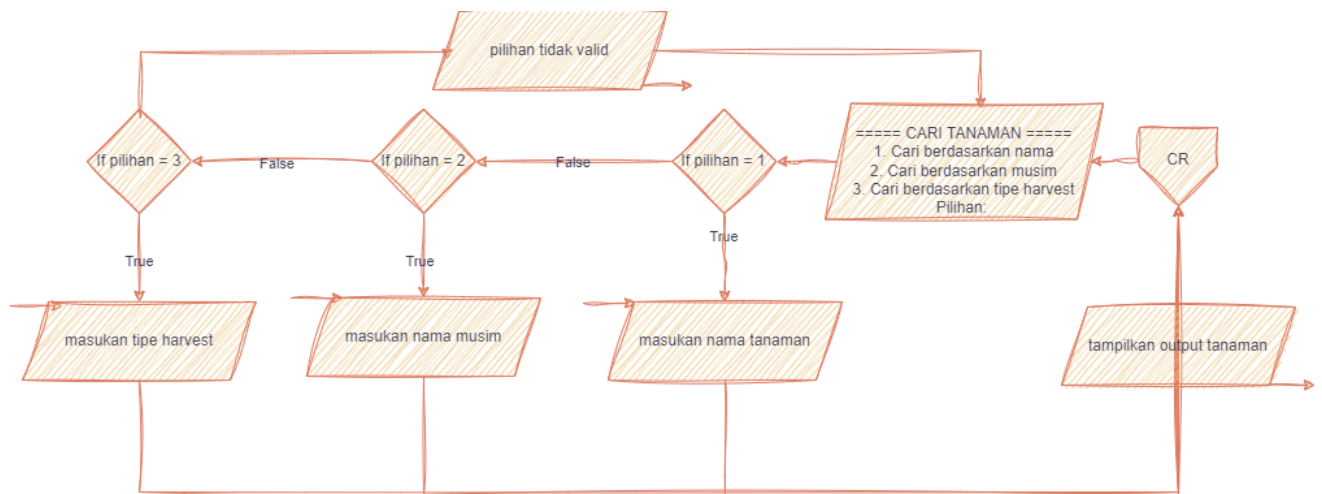
Gambar 1.1 Flowchat Menu Login

1.1 Menu utama



Gambar 1.2 Flowchart Menu utama

1.2 Menu pencarian



Gambar 1.3 Flowchat Menu pencarian

2 Analisis Program

2.1 Deskripsi Singkat Program

Program ini adalah aplikasi manajemen tanaman Stardew Valley yang membantu pemain untuk melihat harga beli dan harga jual dari tanaman yang ada dalam game tersebut. Aplikasi memungkinkan pengguna untuk menambahkan, memperbaiki, menghapus, dan mencari informasi tanaman.

Dibandingkan versi sebelumnya, Posttest 5 menambahkan implementasi pointer dan manipulasi memory, function overloading, dan recursive function untuk meningkatkan fungsionalitas program.

2.2 Penjelasan Alur & Algoritma

Inisialisasi Program

Program dimulai dengan `main()` yang membersihkan layar konsol dan menampilkan header program konversi. Pengguna diarahkan untuk login terlebih dahulu sebelum dapat mengakses fitur utama.

2.1. Inisialisasi Program

Program dimulai dengan:

1. Mengimpor library yang diperlukan (iostream, string, vector, iomanip, limits, fstream, sstream)
2. Mendefinisikan struktur data untuk tanaman (Crop) dan pengguna (User)
3. Mendeklarasikan variabel global dan prototipe fungsi
4. Memuat data pengguna dan tanaman dari file (jika ada)

2.2. Sistem Login

Alur login:

1. Menampilkan sistem login
2. Meminta input nama dan NIM
3. Memeriksa apakah input sesuai dengan kredensial yang tersimpan
4. Jika benar, login berhasil
5. Jika salah, pengguna diberi kesempatan maksimal 3 kali percobaan
6. Jika gagal setelah 3 percobaan, program berhenti

2.3 Inisialisasi Data Tanaman

Setelah login berhasil, program:

1. Memuat data tanaman dari file (jika ada)
2. Jika tidak ada file data, menginisialisasi data tanaman default
3. Menyimpan informasi seperti nama, harga bibit, waktu tumbuh, harga jual, musim tanam, dan tipe panen
4. Menghitung jumlah tanaman per musim (Spring, Summer, Fall)

1.4. Menu Utama

Program menampilkan menu utama dengan opsi:

1. Tampilkan Semua Tanaman
2. Tambah Tanaman Baru
3. Perbarui Tanaman
4. Hapus Tanaman
5. Cari Tanaman
6. Daftar Pengguna Baru
7. Urutkan Tanaman
8. Simpan Data
9. keluar

1.5. Fungsi-fungsi Utama

1.5.1 Tampilkan Semua Tanaman (Opsi 1)

- Membersihkan layar
- Menampilkan daftar semua tanaman dalam format tabel
- Menampilkan statistik jumlah tanaman per musim
- ◆ Menggunakan fungsi pointer `displayCropByPointer` untuk menampilkan tanaman

1.5.2 Tambah Tanaman Baru (Opsi 2)

- Meminta input untuk detail tanaman baru
- Menambahkan data ke dalam vector tanaman
- Memperbarui statistik jumlah tanaman per musim dengan fungsi pointer
- Menghitung perkiraan keuntungan menggunakan fungsi pointer

1.5.3 Perbarui Tanaman (Opsi 3)

- Menampilkan daftar tanaman untuk referensi
- Meminta nomor tanaman yang akan diperbarui
- Validasi input menggunakan fungsi pointer
- Memperbarui data tanaman yang dipilih menggunakan fungsi pointer
- Memperbarui statistik jumlah tanaman per musim

2.5.4 Hapus Tanaman (Opsi 4)

- Menampilkan daftar tanaman untuk referensi
- Meminta nomor tanaman yang akan dihapus
- Validasi input menggunakan fungsi pointer
- Meminta konfirmasi penghapusan
- Jika dikonfirmasi, menghapus tanaman dari vector
- Memperbarui statistik jumlah tanaman per musim dengan fungsi pointer

2.5.5 Cari Tanaman (Opsi 5)

- Menyediakan opsi pencarian berdasarkan:
 1. Nama tanaman
 2. Musim tanam
 3. Tipe panen
- Menampilkan hasil pencarian dalam format tabel menggunakan fungsi pointer
- Jika pencarian berdasarkan musim, menggunakan fungsi pointer untuk menampilkan statistik

2.5.6 Daftar Pengguna Baru (Opsi 6)

- Meminta input nama dan NIM untuk pengguna baru
- Memeriksa apakah pengguna sudah terdaftar
- Jika belum, mendaftarkan pengguna baru dan menyimpan ke file

2.5.7 Urutkan Tanaman (Opsi 7) - Fitur Baru

- Mengurutkan tanaman berdasarkan keuntungan menggunakan algoritma bubble sort
- Menggunakan fungsi pointer untuk menghitung keuntungan setiap tanaman
- Menggunakan fungsi pointer untuk menukar tanaman dalam array

2.5.8 Simpan Data (Opsi 8)

- Menyimpan data pengguna dan tanaman ke file
- Memastikan data tersimpan dengan menggunakan file I/O function

2.5.9 Simpan Data (Opsi 9)

- Menyimpan data sebelum keluar
- Menampilkan pesan terima kasih
- Mengakhiri program

Fitur Penyimpanan Data Persisten

Program menggunakan file untuk menyimpan data:

- **users.txt**: Menyimpan data pengguna (nama, NIM, status aktif)

Fungsi penyimpanan dan pemuatan:

- `simpanPengguna()`: Menyimpan data pengguna ke file
- `muatPengguna()`: Memuat data pengguna dari file
- `simpanCrops()`: Memuat data tanaman dari file
- `muatCrops()`: Memuat data tanaman dari file

Fungsi Rekursif

Program menggunakan fungsi rekursif **calculateProfit()** untuk menghitung keuntungan potensial dari penanaman dalam jumlah hari tertentu, mempertimbangkan:

- Jenis tanaman (single/multiple harvest)
- Waktu tumbuh
- Hargajual
- Harga bibit
- Jumlah hari

Function Overloading

Program mengimplementasikan function overloading untuk:

- `displayStats(const vector<int>& stats, const vector<string>& labels)`

Menampilkan statistik dengan atau tanpa judul

- `displayStats(const string& title, const vector<int>& stats, const vector<string>& labels)`

: Menampilkan data tanaman dengan atau tanpa nomor indeks

- `displayCrop(const Crop& crop)`
: menampilkan data tanaman detail

Fungsi dengan parameter pointer

- `processCropInput(int* valuePtr)`
: validasi input menggunakan pointer
- `validateIndex(int* indexPtr, int maxSize)`
: validasi index menggunakan pointer
- `updateCropByPointer(Crop* cropPtr)`
: update data tanaman menggunakan pointer
- `displayCropByPointer(const Crop* cropPtr, int index)`
: menampilkan tanaman menggunakan pointer
- `displayCropByPointer(const Crop* cropPtr, int index)`
: Overloading fungsi dengan parameter tambahan
- `calculateProfitByPtr(const Crop* cropPtr, int days, int plantCount)`
: menghitung profit menggunakan pointer

2.6. Validasi Input

Program melakukan validasi input untuk: Memastikan input berupa angka saat memilih menu
Memastikan indeks tanaman valid saat memperbarui atau menghapus
Memastikan format data yang dimasukkan sesuai

3 Source Code

3.1 Struktur Data

```
// Struct to store crop data
struct Crop {
    string name;
    int seedPrice;
    int growthTime;
    int sellingPrice;
    string growSeason;
    string harvestType;
};

// Struct for user credentials (for multi-user support)
struct User {
    string name;
    string nim;
    bool isActive = false;
};
```

gambar 3.1 struktur data

3.2 Fungsi pointer

```
// New function to validate input using pointer
void processCropInput(int* valuePtr) {
    if (!(cin >> *valuePtr)) {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Input tidak valid! Masukkan angka." << endl;
        *valuePtr = 0; // Set default value
    }
}

// New function to validate index using pointer (returns boolean)
bool validateIndex(int* indexPtr, int maxSize) {
    if (*indexPtr < 1 || *indexPtr > maxSize) {
        cout << "Nomor tanaman tidak valid!" << endl;
        return false;
    }

    // Adjust index to 0-based
    (*indexPtr)--;
    return true;
}
```

```

// New function to display crop with index using pointer (parameter dereference)
void displayCropByPointer(const Crop* cropPtr, int index) {
    int profit = calculateProfitByPtr(cropPtr, 28);
    cout << left << setw(5) << index
        << setw(20) << cropPtr->name
        << setw(15) << cropPtr->seedPrice
        << setw(15) << cropPtr->growthTime
        << setw(15) << cropPtr->sellingPrice
        << setw(15) << cropPtr->growSeason
        << setw(10) << cropPtr->harvestType
        << setw(15) << profit << endl;
}

```

gambar 3.2 fungsi pointer

3.3 fungsi registrasi

```

void registerUser() {
    User newUser;

    cin.ignore();
    cout << "\n==== DAFTAR PENGGUNA BARU =====> << endl;
    cout << "Masukkan Nama: ";
    getline(cin, newUser.name);

    cout << "Masukkan NIM: ";
    getline(cin, newUser.nim);

    // Check if user already exists
    bool userExists = false;
    for (const User& user : users) {
        if (user.name == newUser.name && user.nim == newUser.nim) {
            userExists = true;
            break;
        }
    }

    if (userExists) {
        cout << "Pengguna sudah terdaftar!" << endl;
    } else {
        newUser.isActive = true;
        users.push_back(newUser);

        // Save user data immediately after registration
        simpanPengguna();

        cout << "Pengguna berhasil terdaftar!" << endl;
    }
}

```

gambar 3.3 registrasi

3.4 Fungsi penyimpanan data

```
// Save user data to file
void simpanPengguna() {
    ofstream outFile("users.txt");

    if (!outFile) {
        cout << "Error: Tidak bisa membuka file users.txt untuk menyimpan data!" <<
endl;
        return;
    }

    for (const User& user : users) {
        outFile << user.name << "," << user.nim << "," << (user.isActive ? "1" : "0") <<
endl;
    }

    outFile.close();
}

// Load user data from file
void muatPengguna() {
    ifstream inFile("users.txt");

    if (!inFile) {
        cout << "File users.txt tidak ditemukan. Menggunakan pengguna default." << endl;
        return;
    }

    users.clear();
    string line;

    while (getline(inFile, line)) {
        stringstream ss(line);
        string name, nim, active;

        getline(ss, name, ',');
        getline(ss, nim, ',');
        getline(ss, active, ',');

        User user;
        user.name = name;
        user.nim = nim;
        user.isActive = (active == "1");

        users.push_back(user);
    }

    inFile.close();
}
```

gambar 3.4 penyimpanan data

3.5 Fungsi rekursif

```
// Recursive function to calculate potential profit over days
int calculateProfit(const Crop& crop, int days, int plantCount) {
    // Base case: no more days left
    if (days <= 0) {
        return 0;
    }

    // If growth time is more than remaining days, no profit
    if (crop.growthTime > days) {
        return 0;
    }

    int profit = 0;

    if (crop.harvestType == "single") {
        // Single harvest crops: calculate how many harvests we can get in the days
        int numHarvests = days / crop.growthTime;
        profit = numHarvests * (crop.sellingPrice - crop.seedPrice) * plantCount;
    } else if (crop.harvestType == "multiple") {
        // Multiple harvest crops: get one harvest every 2-3 days after initial growth
        int harvestInterval = 3; // Average regrow time

        // Initial harvest after growth time
        profit = crop.sellingPrice * plantCount;

        // Remaining days for additional harvests
        int remainingDays = days - crop.growthTime;

        // Additional harvests
        profit += (remainingDays / harvestInterval) * crop.sellingPrice * plantCount;

        // Subtract seed cost once per plant
        profit -= crop.seedPrice * plantCount;
    }

    return profit;
}
```

gambar 3.5 fungsi rekursif

3.6 menu overloading

```
// Function overloading for displaying statistics
void displayStats(const vector<int>& stats, const vector<string>& labels) {
    for (size_t i = 0; i < stats.size(); i++) {
        cout << labels[i] << ": " << stats[i] << " tanaman" << endl;
    }
}

// Function overloading with custom title
void displayStats(const string& title, const vector<int>& stats, const vector<string>& labels) {
    cout << "\n" << title << endl;
    for (size_t i = 0; i < stats.size(); i++) {
        cout << labels[i] << ": " << stats[i] << " tanaman" << endl;
    }
}

// Function overloading for displaying crops
void displayCrop(const Crop& crop) {
    cout << left << "Nama: " << crop.name << endl
        << "Harga Bibit: " << crop.seedPrice << endl
        << "Waktu Tumbuh: " << crop.growthTime << " hari" << endl
        << "Harga Jual: " << crop.sellingPrice << endl
        << "Musim Tanam: " << crop.growSeason << endl
        << "Tipe Harvest: " << crop.harvestType << endl;
}

// Function overloading with index parameter
void displayCrop(const Crop& crop, int index) {
    cout << left << setw(5) << index
        << setw(20) << crop.name
        << setw(15) << crop.seedPrice
        << setw(15) << crop.growthTime
        << setw(15) << crop.sellingPrice
        << setw(15) << crop.growSeason
        << setw(10) << crop.harvestType << endl;
}
```

gambar 3.6 fungsi overloading

3.7 Pengurutan tanaman

```
// Sort crops by profit (using pointers)
cout << "\n==== URUTKAN TANAMAN =====> endl;
cout << "Mengurutkan tanaman berdasarkan keuntungan..." << endl;

// Simple bubble sort using pointers
for (size_t i = 0; i < crops.size() - 1; i++) {
    for (size_t j = 0; j < crops.size() - i - 1; j++) {
        int profit1 = calculateProfitByPtr(&crops[j], 28);
        int profit2 = calculateProfitByPtr(&crops[j + 1], 28);

        if (profit1 < profit2) {
            swapCrops(&crops[j], &crops[j + 1]);
        }
    }
}

cout << "Tanaman berhasil diurutkan berdasarkan keuntungan!" <<
endl;

cout << "Gunakan pilihan '1' untuk melihat hasil pengurutan." <<
endl;
```

gambar 3.7 pengurutan tanaman

4.1 Uji Coba

Program diuji dengan beberapa skenario berikut:

- 1. Login dengan kredensial yang benar dan salah*
- 2. Menampilkan semua tanaman*
- 3. Menambahkan tanaman baru dengan berbagai jenis data*
- 4. Memperbarui data tanaman yang sudah ada*
- 5. Menghapus tanaman*
- 6. Mencari tanaman berdasarkan nama, musim, dan tipe panen*
- 7. Mendaftarkan pengguna baru*
- 8. Mengurutkan tanaman berdasarkan keuntungan (fitur baru)*
- 9. Menyimpan data dan memuat kembali*

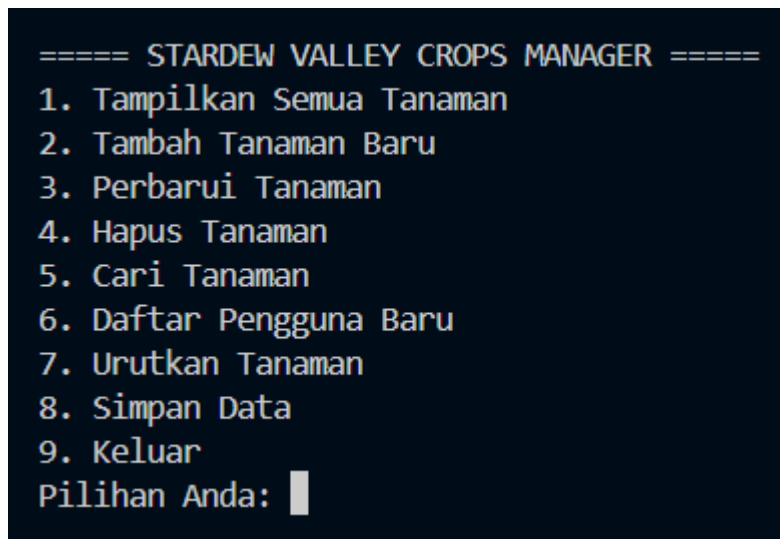
4.2 Hasil Output

```
===== LOGIN SYSTEM =====  
Masukkan Nama: ippan  
Masukkan NIM: icikiwir
```

gambar 4.2 menu login salah

```
Login gagal! Percobaan ke-1 dari 3  
Masukkan Nama: 
```

gambar 4.2 output password salah



gambar 4.3 menu utama

===== DAFTAR SEMUA TANAMAN =====						
No	Nama	Harga Bibit	Waktu Tumbuh	Harga Jual	Musim	Tipe
1	Blue Jazz	30	7	50	Spring	single
2	Cauliflower	80	12	175	Spring	single
3	Garlic	40	4	60	Spring	single
4	Kale	70	6	110	Spring	single
5	Parsnip	20	4	35	Spring	single
6	Potato	50	6	80	Spring	single
7	Rhubarb	100	13	220	Spring	single
8	Tulip	20	6	30	Spring	single
9	Unmilled Rice	40	6	30	Spring	single
10	Carrot	0	3	35	Spring	single
11	Coffee Bean	2500	10	60	Spring, Summer	multiple
12	Green Bean	60	10	40	Spring	multiple
13	Strawberry	100	8	120	Spring	multiple
14	Melon	80	12	250	Summer	single
15	Poppy	100	7	140	Summer	single
16	Radish	40	6	90	Summer	single
17	Red Cabbage	100	9	260	Summer	single
18	Starfruit	400	13	750	Summer	single
19	Summer Spangle	50	8	90	Summer	single
20	Wheat	10	4	25	Summer, Fall	single
21	Sunflower	200	8	80	Summer, Fall	single
22	Blueberry	80	13	150	Summer	multiple
23	Corn	150	14	50	Summer, Fall	multiple
24	Hops	50	11	25	Summer	multiple
25	Hot Pepper	40	5	40	Summer	multiple
26	Tomato	50	11	60	Summer	multiple
27	Summer Squash	0	6	45	Summer	multiple

gambar 4.4 menampilkan daftar tumbuhan

```

===== TAMBAH TANAMAN BARU =====
Nama Tanaman: spiderlily
Harga Bibit: 2000
Waktu Tumbuh (hari): 19
Harga Jual: 99999
Musim Tanam (Spring/Summer/Fall): fall
Tipe Harvest (single/multiple): single
Tanaman berhasil ditambahkan!

```

gambar 4.5 penambahan tanaman

```

Masukkan nomor tanaman yang ingin diperbarui: 38
Nama Tanaman [Grape]: anggur merah di tepi sungai
Harga Bibit [60]: 150
Waktu Tumbuh [10]: 11
Harga Jual [80]: 80
Musim Tanam [Fall]: fall
Tipe Harvest [multiple]: multiple
Tanaman berhasil diperbarui!

```

gambar 4.6 memperbarui tanaman

```

38. anggur merah di tepi sungai
39. spiderlily
Masukkan nomor tanaman yang ingin dihapus: 38
Tanaman anggur merah di tepi sungai akan dihapus. Lanjutkan? (y/n): y
Tanaman berhasil dihapus!

```

gambar 4.6 penghapusan tanaman

```

===== CARI TANAMAN =====
1. Cari berdasarkan nama
2. Cari berdasarkan musim
3. Cari berdasarkan tipe harvest
Pilihan: 1
Masukkan nama tanaman: Tomato

Hasil Pencarian untuk 'Tomato':

```

No	Nama	Harga Bibit	Waktu Tumbuh	Harga Jual	Musim	Tipe
1	Tomato	50	11	60	Summer	multiple

gambar 4.6 pencarian tanaman

```

===== DAFTAR PENGGUNA BARU =====
Masukkan Nama: ippaniciwir
Masukkan NIM: 24091099
Pengguna berhasil terdaftar!

Tekan Enter untuk lanjut...

```

gambar 4.7 menu daftar pengguna baru

```
===== URUTKAN TANAMAN =====  
Mengurutkan tanaman berdasarkan keuntungan...  
Tanaman berhasil diurutkan berdasarkan keuntungan!  
Gunakan pilihan '1' untuk melihat hasil pengurutan.  
  
Tekan Enter untuk lanjut...  
█
```

gambar 4.8 urutkan tanaman berdasarkan keuntungan

```
===== STARDEW VALLEY CROPS MANAGER =====  
1. Tampilkan Semua Tanaman  
2. Tambah Tanaman Baru  
3. Perbarui Tanaman  
4. Hapus Tanaman  
5. Cari Tanaman  
6. Daftar Pengguna Baru  
7. Urutkan Tanaman  
8. Simpan Data  
9. Keluar  
Pilihan Anda: 8  
Data berhasil disimpan!  
  
Tekan Enter untuk lanjut...  
█
```

gambar 4.9 simpan data pengguna

```
===== STARDEW VALLEY CROPS MANAGER =====  
1. Tampilkan Semua Tanaman  
2. Tambah Tanaman Baru  
3. Perbarui Tanaman  
4. Hapus Tanaman  
5. Cari Tanaman  
6. Daftar Pengguna Baru  
7. Simpan Data  
8. Keluar  
Pilihan Anda: 8  
Menyimpan data sebelum keluar...  
Terima kasih telah menggunakan program Stardew Valley Crops Manager!
```

gambar 4.9 logout

5. Langkah Langkah Git

5.1 Git Init

Perintah `git init` memungkinkan pengguna untuk menginisialisasi repository baru di dalam suatu direktori kerja. Setelah perintah ini dijalankan, pengguna dapat mulai melacak perubahan yang terjadi pada berbagai file.

```
Lenovo-GK@ippan MINGW64 ~/Desktop/praktikum-APL/posttest/post-test-apl-1 (main)
$ git init
Initialized empty Git repository in C:/Users/Lenovo-GK/Desktop/praktikum-APL/posttest/post-test-apl-1/.git/
```

Gambar 5.1 Git Init

5.2 Git Add

Perintah "git add" digunakan untuk menambahkan file yang ingin Anda komit.

```
Lenovo-GK@ippan MINGW64 ~/Desktop/praktikum-APL/posttest/post-test-apl-1 (main)
$ git add .
```

5.2 Git Add

5.3 Git Commit

Commit adalah tindakan menyimpan perubahan kode ke dalam repository

```
Lenovo-GK@ippan MINGW64 ~/Desktop/praktikum-APL/posttest/post-test-apl-1 (main)
$ git commit -m "ganti nama file"
```

5.3 Git Commit

5.4 Git Remote

Git Remote adalah perintah yang digunakan untuk menghubungkan repository lokal dengan repository yang ada di GitHub.

```
Lenovo-GK@ippan MINGW64 ~/Desktop/praktikum-APL/posttest/post-test-apl-1 (main)
$ git remote add origin https://github.com/ippanicikiwir/KELAS-B24/tree/main/praktikum-apl/posttest/post-test-apl-1
```

5.4 Git Remote

5.5 Git Push

Push adalah perintah yang digunakan untuk mengirim commit dari repository lokal ke repository jarak jauh (server).

```
Lenovo-GK@ippan MINGW64 ~/Desktop/praktikum-APL/posttest/post-test-apl-1 (main)
$ git push -u origin main
```

5.5 Git Push

