LAPORAN PRAKTIKUM

POSTTEST 4

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

IKHWAN HARIYANTO (2409106082)

Kelas (B2‘24)

PROGRAM STUDI INFORMATIKA

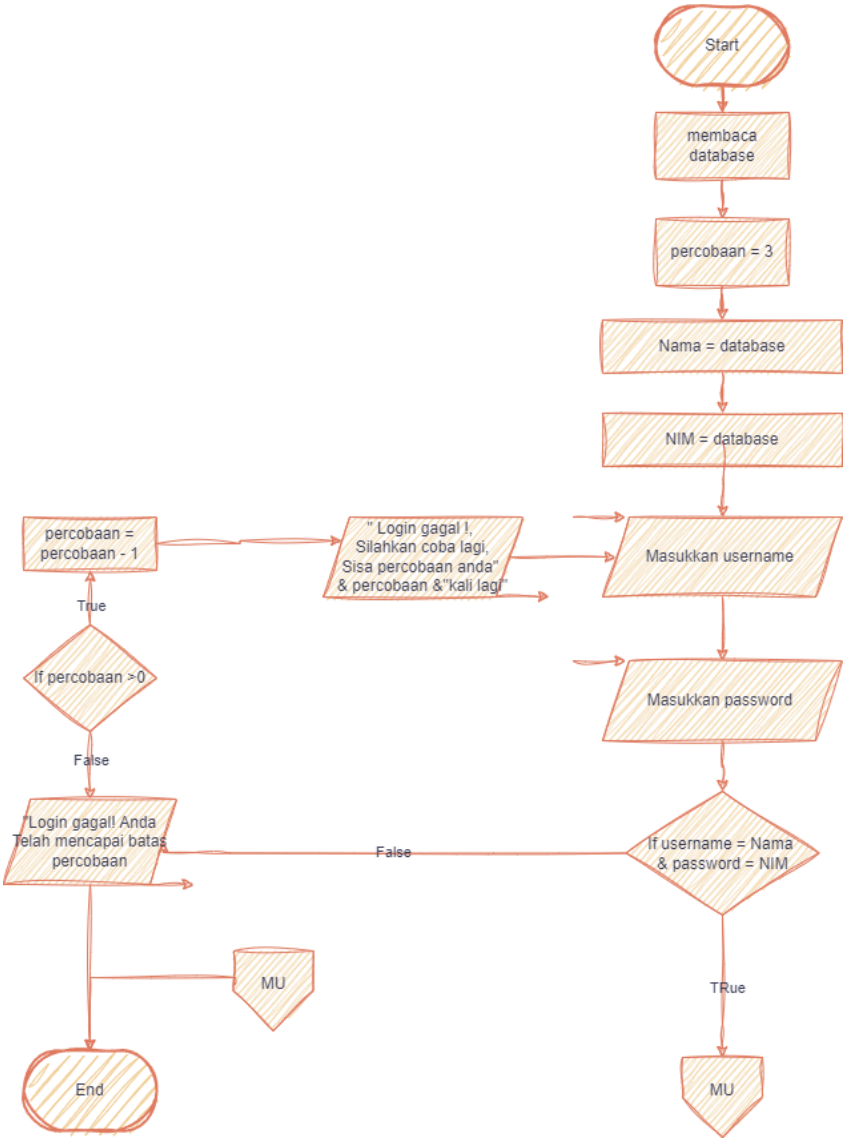
UNIVERSITAS MULAWARMAN

SAMARINDA

2025

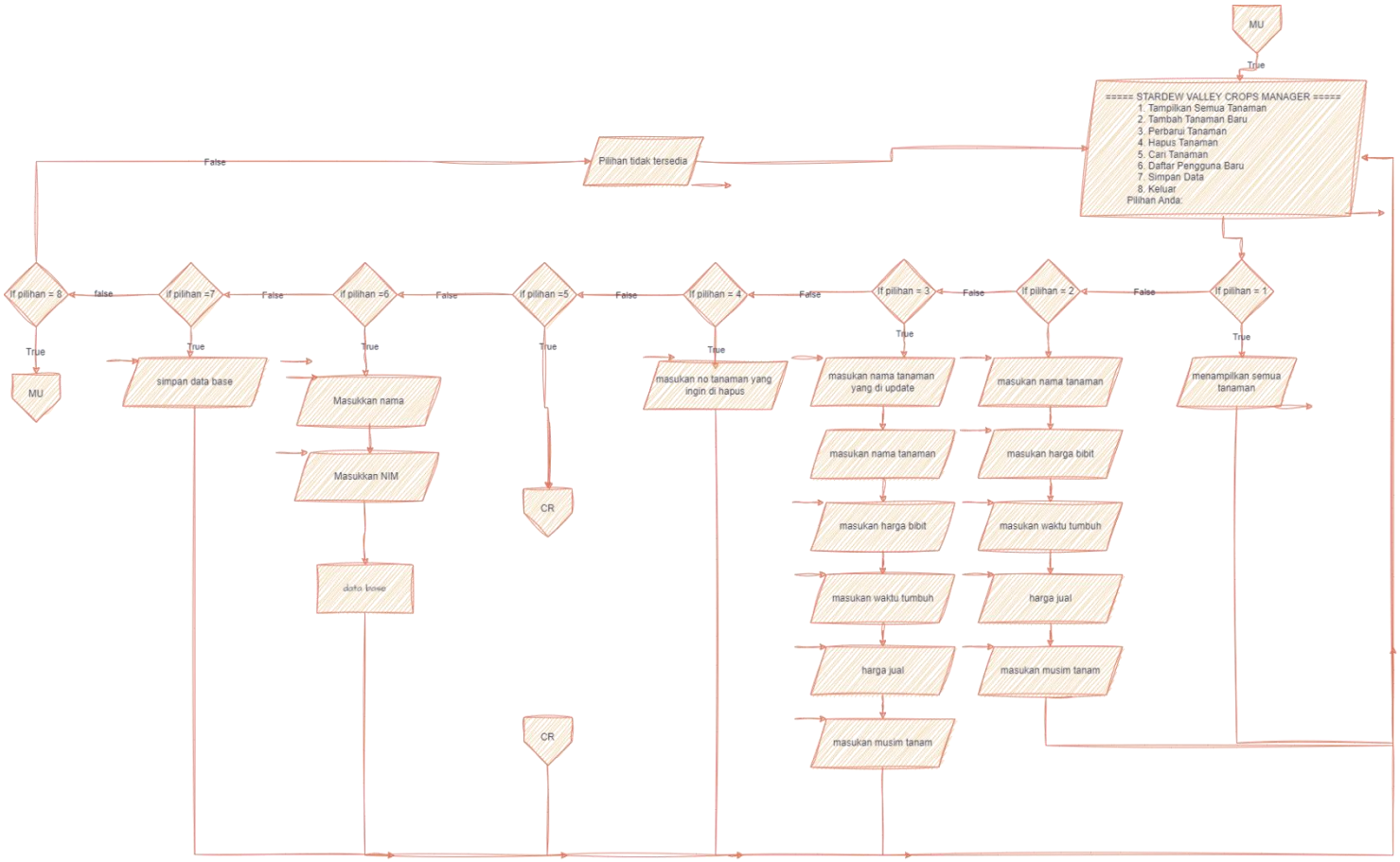
1. Flowchart

1.1 Menu Login



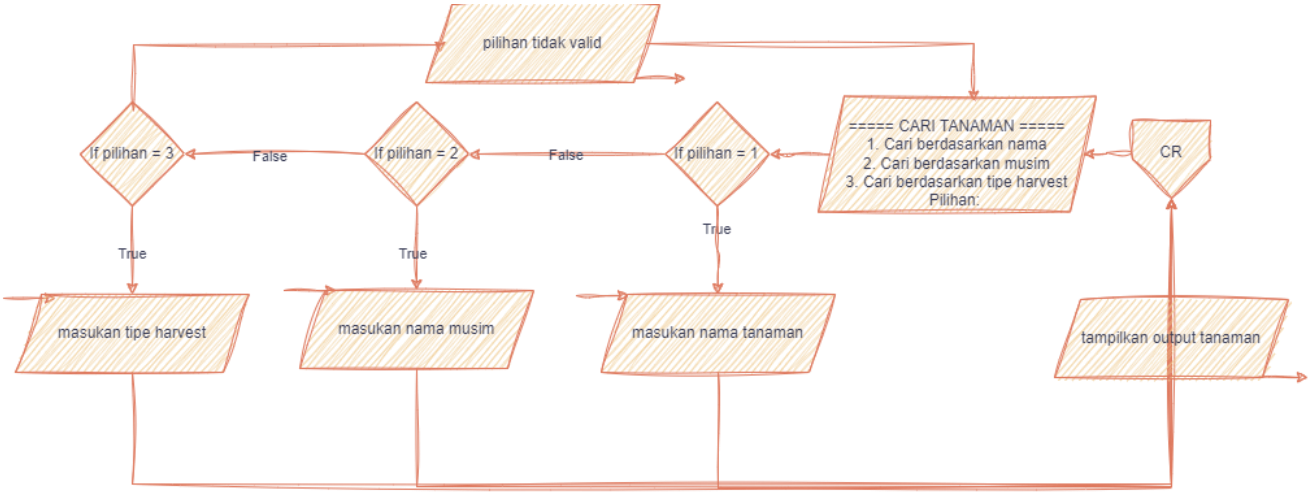
1.1 Flowchat Menu Login

1.2 Menu utama



1.2 Flowchart Menu utama

1.3 Menu Login



1.3 Flowchat Menu Login

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, memperbarui, menghapus, dan mencari informasi tanaman. Versi terbaru menambahkan fitur multi-user dan sistem penyimpanan data persisten ke dalam file.

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 Registrasi

Alur registrasi:

1. Meminta input nama dan NIM untuk pengguna baru

2. Memeriksa apakah pengguna sudah terdaftar

3. Jika belum, menambahkan pengguna baru ke dalam sistem

4. Menyimpan data pengguna ke dalam file

2.4. 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)

2.5. 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. Simpan Data
8. Keluar

2.6. Fungsi-fungsi Utama

2.6.1 Tampilkan Semua Tanaman (Opsi 1)

Membersihkan layar

Menampilkan daftar semua tanaman dalam format tabel

Menampilkan statistik jumlah tanaman per musim

2.6.2 Tambah Tanaman Baru (Opsi 2)

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

2.6.3 Perbarui Tanaman (Opsi 3)

- Menampilkan daftar tanaman untuk referensi
- Meminta nomor tanaman yang akan diperbarui
- Memperbarui data tanaman yang dipilih
- Memperbarui statistik jumlah tanaman per musim

2.6.4 Hapus Tanaman (Opsi 4)

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

2.6.5 Cari Tanaman (Opsi 5)

Menyediakan opsi pencarian berdasarkan:

1. Nama tanaman
2. Musim tanam
3. Tipe panen

Menampilkan hasil pencarian dalam format tabel

2.6.6 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

2.6.7 Simpan Data (Opsi 7)

- Menyimpan data pengguna dan tanaman ke file
- Memastikan data tersimpan dengan aman

2.6.8 Simpan Data (Opsi 8)

- 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)
- **crops.txt**: Menyimpan data tanaman dan statistik musim

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(): Menampilkan statistik dengan atau tanpa judul
- displayCrop(): Menampilkan data tanaman dengan atau tanpa nomor indeks

2.7. Validasi Input

Program melakukan validasi input untuk: Memastikan input berupa angka saat memilih menuMemastikan 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 Menu Login

```
bool loginSystem() {
    string inputName, inputNim;
    int loginAttempts = 0;
    bool isLoggedIn = false;

    cout << "===== LOGIN SYSTEM =====" << endl;

    while (loginAttempts < 3 && !isLoggedIn) {
        cout << "Masukkan Nama: ";
        cin >> inputName;
        cout << "Masukkan NIM: ";
        cin >> inputNim;

        // Check login against all users
        for (const User& user : users) {
            if (inputName == user.name && inputNim == user.nim) {
                cout << "Login berhasil!" << endl;
                isLoggedIn = true;
                break;
            }
        }

        if (!isLoggedIn) {
            loginAttempts++;
```

```
            clearScreen();

            cout << "Login gagal! Percobaan ke-" << loginAttempts << " dari 3" << endl;
        }
    }

    return isLoggedIn;
}
```

gambar 3.2 menu login

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

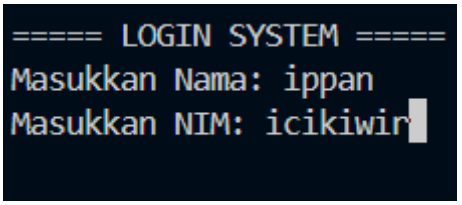
```
// Function overloading for displaying statistics
void displayStats(const vector<int>& stats, const vector<string>& labels) {
    for (size_t i = 0; i < stats.size(); i++) {
        cout << labels[i] << ": " << stats[i] << " tanaman" << endl;
    }
}

// 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

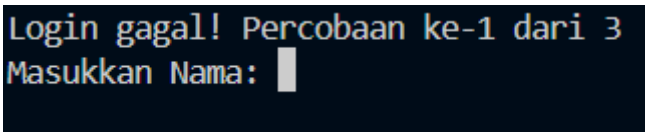
4.1 Uji Coba

(Jelaskan skenario yang digunakan untuk menguji program, misalnya dengan berbagai jenis input.)

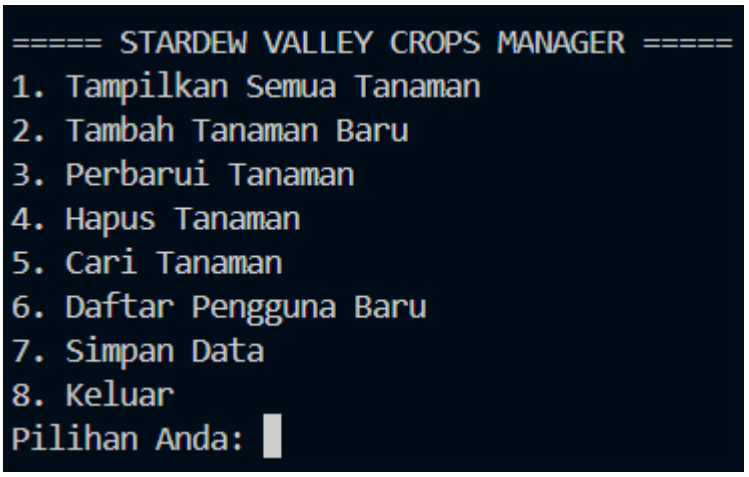
4.2 Hasil Output



gambar 4.2 menu login salah



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!
```

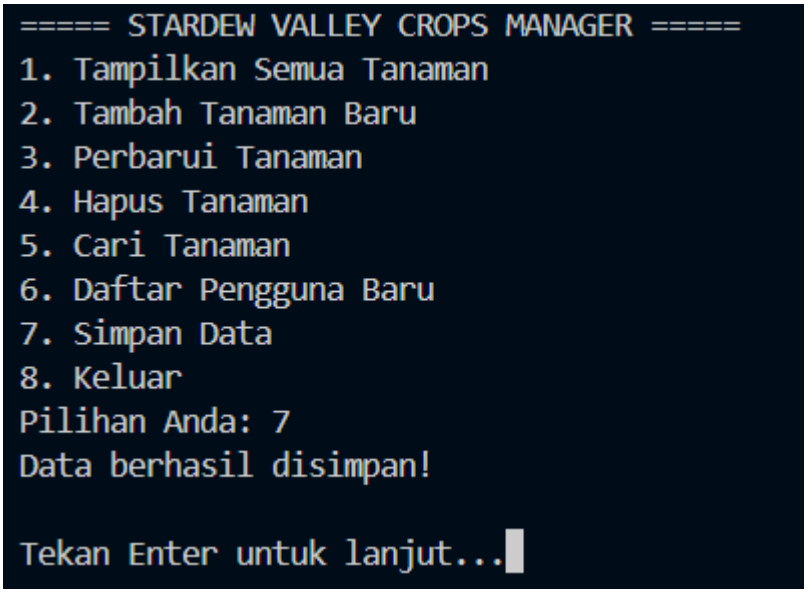
gambar 4.6 penghapusan tanaman

```
===== 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
```

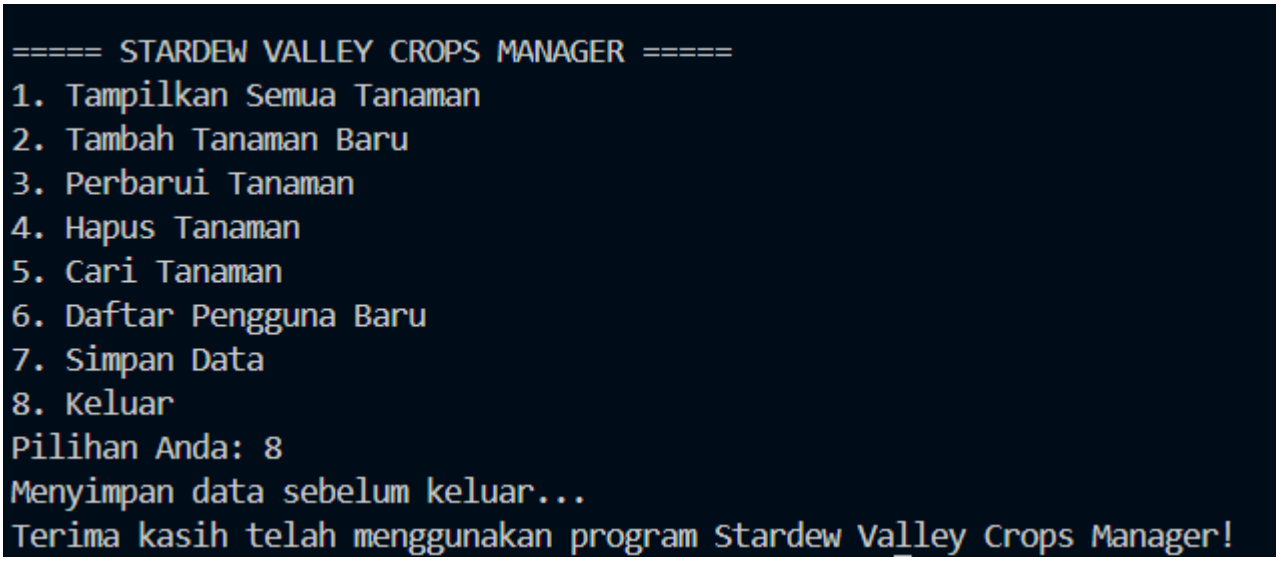
gambar 4.6 pencarian tanaman

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

Tekan Enter untuk lanjut...
```

gambar 4.7 menu daftar pengunna baru



gambar 4.8 memilih pilihan simpan data

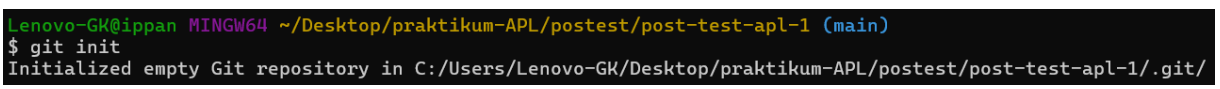


gambar 4.9 logout

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.



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/postest/post-test-apl-1 (main)
$ git add .
```

5.2 Git Add

5.3 Git Commit

Commit adalah tindakan menyimpan perubahan kode ke dalam repositori

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

5.4 Git Remote

5.5 Git Push

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

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

5.5 Git Pu