LAPORAN PRAKTIKUM

POSTTEST 4

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Alya Mayasha (2409106054)

Kelas (B1‘24)

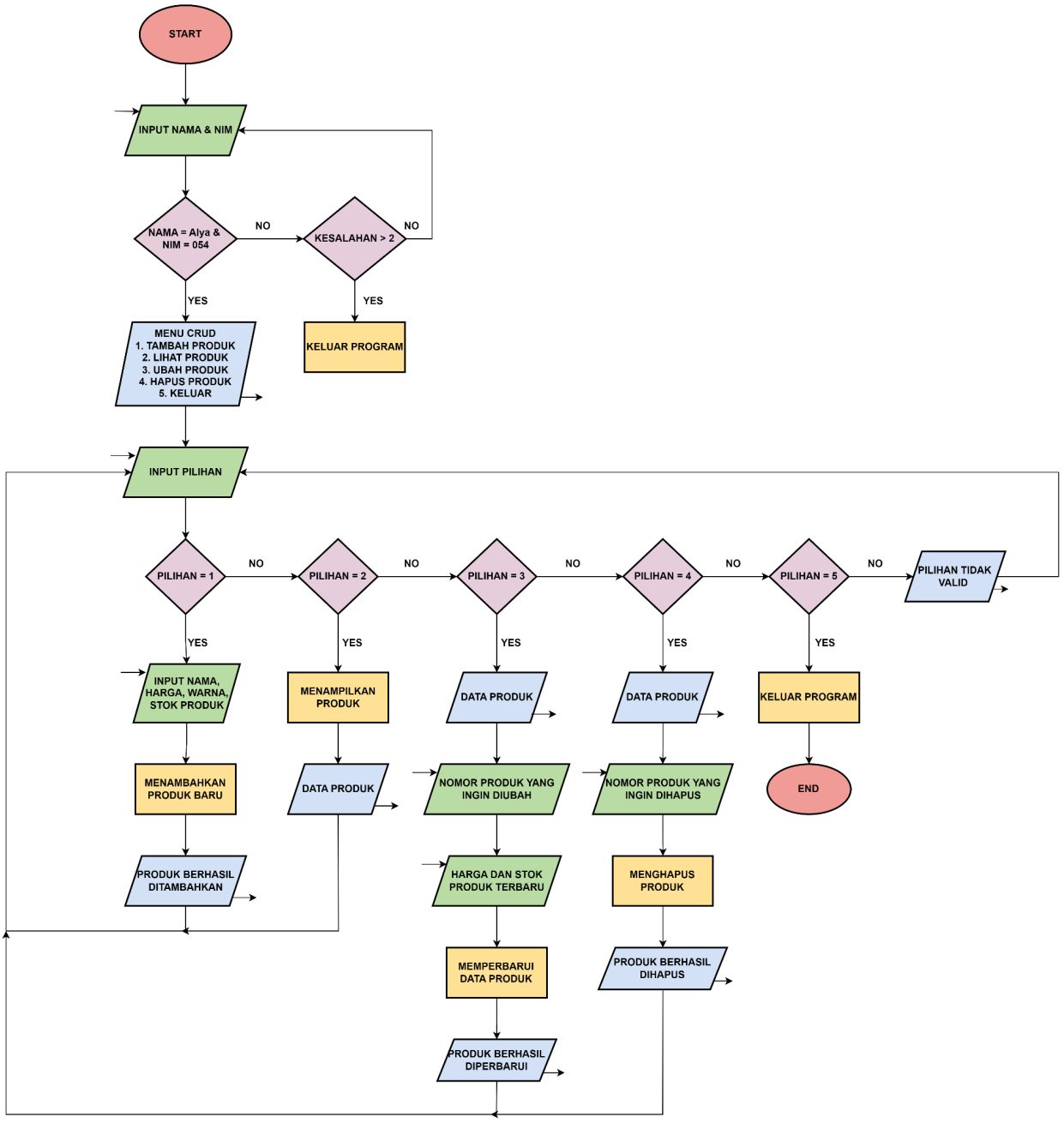
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

SAMARINDA

2025

1. Flowchart



Gambar 1.1 Flowchart Program Manajemen Tas Rajut

2. Analisis Program

Program ini adalah sebuah program manajemen produk tas rajut berbasis C++. Program memungkinkan pengguna untuk mengelola data produk dengan berbagai fitur, seperti *Create* pengguna dapat menambahkan produk baru ke dalam daftar dengan memasukkan nama, harga, warna, dan stok produk. Kemudian *Read*, pengguna dapat melihat daftar produk dalam bentuk tabel dengan informasi lengkap. Selanjutnya *Update*, pengguna dapat mengedit harga dan stok produk. Dan *Delete*, pengguna dapat menghapus produk dari daftar berdasarkan nomor yang dipilih. Pada program ini pengguna diberikan kesempatan tiga kali untuk login dengan memasukkan nama (Alya) dan nim (054). Jika berhasil pengguna dapat mengakses menu CRUD yang tersedia. Program akan terus berjalan hingga pengguna memilih opsi keluar. Jika pengguna gagal login sebanyak tiga kali, maka program akan otomatis berhenti.

Data produk direpresentasikan dalam struct Produk yang memuat nama dan harga, serta nested struct DetailProduk untuk menyimpan warna dan stok. Seluruh fitur program dipisahkan ke dalam fungsi tersendiri. Fungsi tambahProduk() memungkinkan pengguna menambahkan data ke array dan menampilkannya langsung setelah ditambahkan. Fungsi tampilkanProduk() menampilkan daftar produk dalam bentuk tabel, dengan proses pencetakan baris dilakukan secara rekursif melalui fungsi tampilkanRekursif()—contoh penggunaan prosedur rekursif yang efisien. Fungsi ubahProduk() memungkinkan pengguna mengedit harga dan stok produk yang dipilih, sementara hapusProduk() menangani penghapusan data dengan cara menggeser elemen array. Pemisahan fungsi ini memudahkan pemeliharaan dan pengembangan program karena setiap fitur dikelola secara independen dengan parameter yang sesuai.

3. Source Code

A. Struct dan Nested Struct

Struct *DetailProduk* digunakan untuk menyimpan detail produk, seperti warna dan jumlah stok. Sedangkan Struct *Produk* merepresentasikan data utama produk, seperti nama dan harga. Struct ini juga memiliki anggota berupa *DetailProduk* sebagai nested struct untuk menyimpan informasi tambahan. Dalam penggunaan nested struct, data diakses melalui notasi titik (dot notation). Misalnya, produk[0].detail.warna mengakses warna produk pertama, dan produk[0].detail.stok mengakses stok produk pertama. Hal ini membuat data terorganisir secara hierarkis.

Source Code:

```
struct DetailProduk {
    string warna;
    int stok;
};

struct Produk {
    string nama;
    int harga;
    DetailProduk detail;
};
```

B. Fungsi Rekursif

Fungsi tampilkanRekursif() digunakan untuk menampilkan isi array produk[] secara rekursif, satu per satu, hingga jumlah produk tercapai.

Source Code:

```
void tampilkanRekursif(Produk data[], int indeks, int jumlah) {
    if (indeks >= jumlah) return;
    cout << "| " << setw(2) << indeks + 1
         << " | " << left << setw(18) << data[indeks].nama
         << " | " << right << setw(10) << data[indeks].harga
         << " | " << setw(4) << data[indeks].detail.stok
         << " | " << setw(10) << left << data[indeks].detail.warna << " |\n";
    tampilkanRekursif(data, indeks + 1, jumlah);
}
```

C. Prosedur (Fungsi Non-Rekursif / CRUD)

Fungsi tampilkanRekursif() digunakan untuk menampilkan isi array produk[] secara rekursif, satu per satu, hingga jumlah produk tercapai.

1. **Menampilkan Produk**

Menampilkan tabel produk dengan memanggil fungsi rekursif.

Source Code:

```
void tampilkanProduk(Produk data[], int jumlah) {
    if (jumlah == 0) {
        cout << "Belum ada produk yang tersedia.\n";
        return;
    }
    cout << "\n+----+--------------------+------------+------+------------+\n";
    cout << "| No | Nama Produk        | Harga      | Stok | Warna      |\n";
    cout << "+----+--------------------+------------+------+------------+\n";
    tampilkanRekursif(data, 0, jumlah);
    cout << "+----+--------------------+------------+------+------------+\n";
}
```

2. **Menambah Produk**

Fitur ini meminta input dari pengguna untuk menambahkan data produk baru ke dalam daftar jika kapasitas belum penuh.

Source Code:

```
void tambahProduk(Produk data[], int &jumlah) {
    if (jumlah >= max_produk) {
        cout << "Data penuh! Tidak bisa menambah produk lagi.\n";
        return;
    }

    cin.ignore();
    cout << "Masukkan nama produk: ";
    getline(cin, data[jumlah].nama);
    cout << "Masukkan harga produk: ";
    cin >> data[jumlah].harga;
    cin.ignore();
    cout << "Masukkan warna produk: ";
    getline(cin, data[jumlah].detail.warna);
    cout << "Masukkan stok produk: ";
    cin >> data[jumlah].detail.stok;

    jumlah++;
    cout << "Produk berhasil ditambahkan!\n";
}
```

3. **Mengubah Produk**

Fitur ini memungkinkan pengguna untuk mengedit harga dan stok produk berdasarkan nomor urutnya.

Source Code:

```
void ubahProduk(Produk data[], int jumlah) {
    int index;
    cout << "Masukkan nomor produk yang ingin diubah: ";
    cin >> index;

    if (index > 0 && index <= jumlah) {
        index--;
        cout << "Masukkan harga baru: ";
        cin >> data[index].harga;
        cout << "Masukkan stok baru: ";
        cin >> data[index].detail.stok;
        cout << "Produk berhasil diperbarui!\n";
    } else {
        cout << "Nomor produk tidak valid!\n";
    }
}
```

4. **Menghapus Produk**

Fitur ini digunakan untuk menghapus produk dari array berdasarkan nomor produk yang dipilih.

Source Code:

```
void hapusProduk(Produk data[], int &jumlah) {
    int index;
    cout << "Masukkan nomor produk yang ingin dihapus: ";
    cin >> index;

    if (index > 0 && index <= jumlah) {
        index--;
        for (int i = index; i < jumlah - 1; i++) {
            data[i] = data[i + 1];
        }
        jumlah--;
        cout << "Produk berhasil dihapus!\n";
    } else {
        cout << "Nomor produk tidak valid!\n";
    }
}
```

D. Pemanggilan Fungsi dalam Menu Utama

Bagian ini menggunakan struktur switch-case untuk memanggil prosedur sesuai pilihan pengguna.

Source Code:

```
int pilihan;
do {
    cout << "\n=== Manajemen Produk Tas Rajut ===\n";
    cout << "1. Tambah Produk\n";
    cout << "2. Tampilkan Produk\n";
    cout << "3. Ubah Produk\n";
    cout << "4. Hapus Produk\n";
    cout << "5. Keluar\n";
    cout << "Pilihan: ";
    cin >> pilihan;

    switch (pilihan) {
        case 1:
            tambahProduk(produk, jumlahProduk);
            break;

        case 2:
            tampilkanProduk(produk, jumlahProduk);
            break;

        case 3:
            ubahProduk(produk, jumlahProduk);
            break;

        case 4:
            hapusProduk(produk, jumlahProduk);
            break;

        case 5:
            cout << "Terima kasih telah menggunakan program ini. Have a great
day >_<!\n";
            break;

        default:
            cout << "Pilihan tidak valid! Coba lagi.\n";
    }
} while (pilihan != 5);
```

4. Hasil Output

```
Masukkan Nama: Alya
Masukkan NIM: 041
Login gagal! Sisa percobaan: 2
Masukkan Nama: Milly
Masukkan NIM: 054
Login gagal! Sisa percobaan: 1
Masukkan Nama: Ssulki
Masukkan NIM: 013
Login gagal! Sisa percobaan: 0
Mohon maaf percobaan login gagal! Program berhenti.
PS C:\Users\alyam\koding>
```

Gambar 4.1 Output Ketika Pengguna Gagal Login

```
=== Manajemen Produk Tas Rajut ===
1. Tambah Produk
2. Tampilkan Produk
3. Ubah Produk
4. Hapus Produk
5. Keluar
Pilihan: 2

+----+--------------------+------------+------+------------+
| No | Nama Produk        | Harga      | Stok | Warna      |
+----+--------------------+------------+------+------------+
| 1  | Backpack Rajut     |     300000 |   20 | Pink       |
| 2  | Tote Bag Rajut     |     125000 |   45 | Baby Blue  |
| 3  | Sling Bag Rajut    |      95000 |   30 | Hitam      |
| 4  | Hand Bag Rajut     |      80000 |   15 | Cream      |
| 5  | Pouch Rajut        |      65000 |   50 | Maroon     |
+----+--------------------+------------+------+------------+
```

Gambar 4.2 Output Ketika Pengguna Memilih Pilihan 2

```
=== Manajemen Produk Tas Rajut ===
1. Tambah Produk
2. Tampilkan Produk
3. Ubah Produk
4. Hapus Produk
5. Keluar
Pilihan: 1
Masukkan nama produk: Goodie Bag Rajut
Masukkan harga produk: 175000
Masukkan warna produk: Army
Masukkan stok produk: 7
Produk berhasil ditambahkan!

+----+--------------------+------------+------+------------+
| No | Nama Produk        | Harga      | Stok | Warna      |
+----+--------------------+------------+------+------------+
| 1  | Backpack Rajut     |     300000 |   20 | Pink       |
| 2  | Tote Bag Rajut     |     125000 |   45 | Baby Blue  |
| 3  | Sling Bag Rajut    |      95000 |   30 | Hitam      |
| 4  | Hand Bag Rajut     |      80000 |   15 | Cream      |
| 5  | Pouch Rajut        |      65000 |   50 | Maroon     |
| 6  | Goodie Bag Rajut   |     175000 |    7 | Army       |
+----+--------------------+------------+------+------------+
```

Gambar 4.3 Output Ketika Pengguna Memilih Pilihan 1

```
=== Manajemen Produk Tas Rajut ===
1. Tambah Produk
2. Tampilkan Produk
3. Ubah Produk
4. Hapus Produk
5. Keluar
Pilihan: 3

+----+--------------------+------------+------+------------+
| No | Nama Produk        | Harga      | Stok | Warna      |
+----+--------------------+------------+------+------------+
| 1  | Backpack Rajut     |     300000 |   20 | Pink       |
| 2  | Tote Bag Rajut     |     125000 |   45 | Baby Blue  |
| 3  | Sling Bag Rajut    |      95000 |   30 | Hitam      |
| 4  | Hand Bag Rajut     |      80000 |   15 | Cream      |
| 5  | Pouch Rajut        |      65000 |   50 | Maroon     |
| 6  | Goodie Bag Rajut   |     175000 |    7 | Army       |
+----+--------------------+------------+------+------------+
Masukkan nomor produk yang ingin diubah: 5
Masukkan harga baru: 75000
Masukkan stok baru: 30
Produk berhasil diperbarui!
```

Gambar 4.4 Output Ketika Pengguna Memilih Pilihan 3

```
=== Manajemen Produk Tas Rajut ===
1. Tambah Produk
2. Tampilkan Produk
3. Ubah Produk
4. Hapus Produk
5. Keluar
Pilihan: 4

+----+--------------------+------------+------+------------+
| No | Nama Produk        | Harga      | Stok | Warna      |
+----+--------------------+------------+------+------------+
| 1  | Backpack Rajut     |     300000 |   20 | Pink       |
| 2  | Tote Bag Rajut     |     125000 |   45 | Baby Blue  |
| 3  | Sling Bag Rajut    |      95000 |   30 | Hitam      |
| 4  | Hand Bag Rajut     |      80000 |   15 | Cream      |
| 5  | Pouch Rajut        |      75000 |   30 | Maroon     |
| 6  | Goodie Bag Rajut   |     175000 |    7 | Army       |
+----+--------------------+------------+------+------------+
Masukkan nomor produk yang ingin dihapus: 3
Produk berhasil dihapus!
```

Gambar 4.5 Output Ketika Pengguna Memilih Pilihan 4

```
=== Manajemen Produk Tas Rajut ===
1. Tambah Produk
2. Tampilkan Produk
3. Ubah Produk
4. Hapus Produk
5. Keluar
Pilihan: 5
Terima kasih telah menggunakan program ini. Have a great day >_<!
PS C:\Users\alyam\koding>
```

Gambar 4.6 Output Ketika Pengguna Memilih Pilihan 5

5. Langkah-Langkah Git

5.1 Git Add

Git Add berfungsi untuk menambahkan file tertentu ke staging area agar siap untuk dikommit.

```
HP-GK@DESKTOP-N8AD4O2 MINGW64 ~/desktop/praktikum-apl/post-test/post-test-4 (main)
$ git add .
```

Gambar 5.1 Git Add

5.2 Git Commit

Git Commit berfungsi untuk menyimpan perubahan yang ada di staging area ke dalam repository disertai dengan pesan deskriptif.

```
HP-GK@DESKTOP-N8AD4O2 MINGW64 ~/desktop/praktikum-apl/post-test/post-test-4 (main)
$ git commit -m "file cpp dan exe"
[main 93d8030] file cpp dan exe
 2 files changed, 173 insertions(+)
 create mode 100644 post-test/post-test-4/2409106054-AlyaMayasha-PT-4.cpp
 create mode 100644 post-test/post-test-4/2409106054-AlyaMayasha-PT-4.exe
```

Gambar 5.2 Git Commit

5.3 Git Push Origin Main

Git Push Origin Main berfungsi untuk mengirim (mengunggah) perubahan dari branch main di repository lokal ke repository remote yang telah dikonfigurasi.

```
HP-GK@DESKTOP-N8AD4O2 MINGW64 ~/desktop/praktikum-apl/post-test/post-test-4 (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 678.41 KiB | 3.10 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/AlyaMayasha/praktikum-apl.git
   78abaf7..93d8030  main -> main
```

Gambar 5.3 Git Push Origin Main