

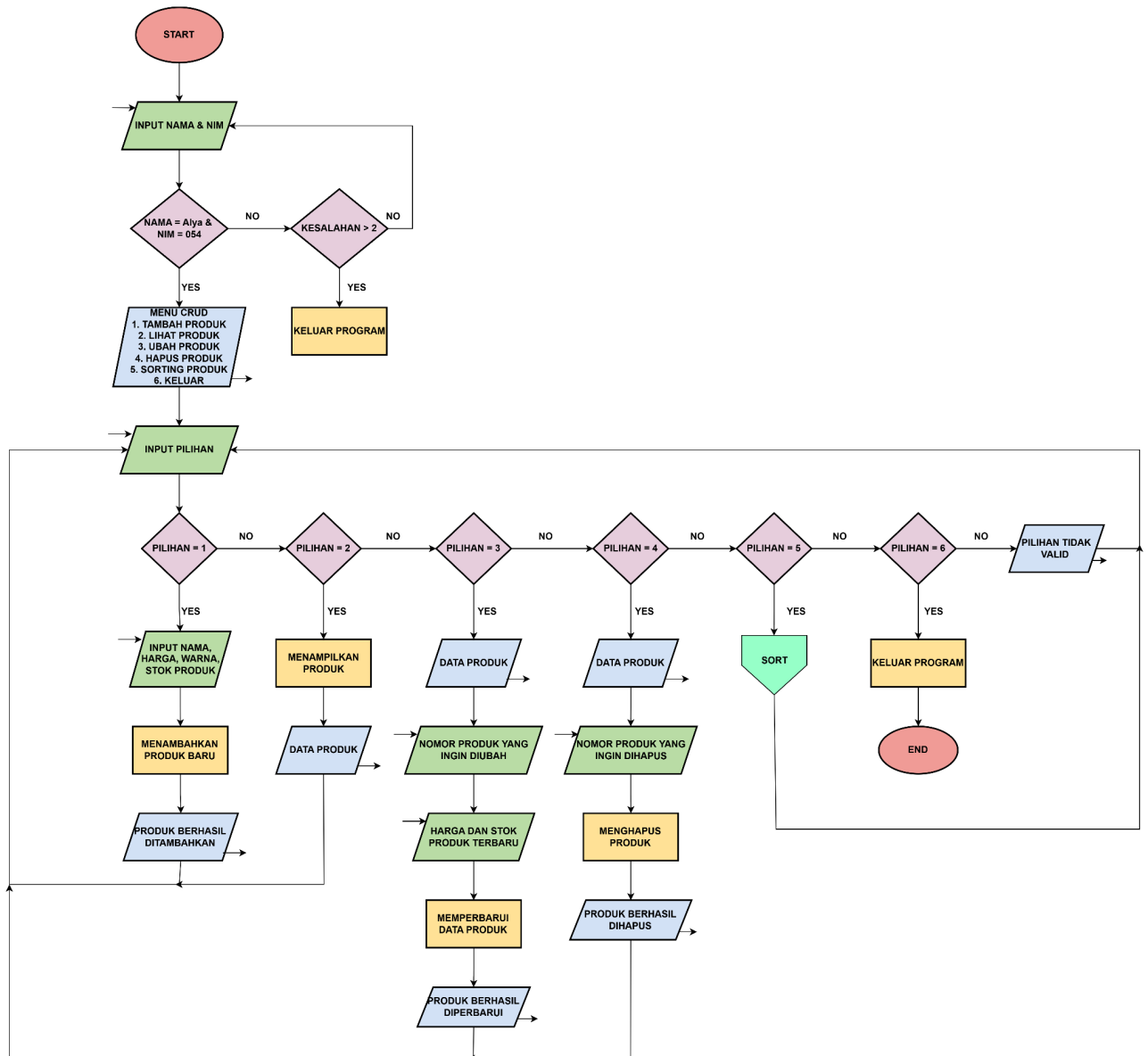
LAPORAN PRAKTIKUM
POSTTEST 6
ALGORITMA PEMROGRAMAN LANJUT



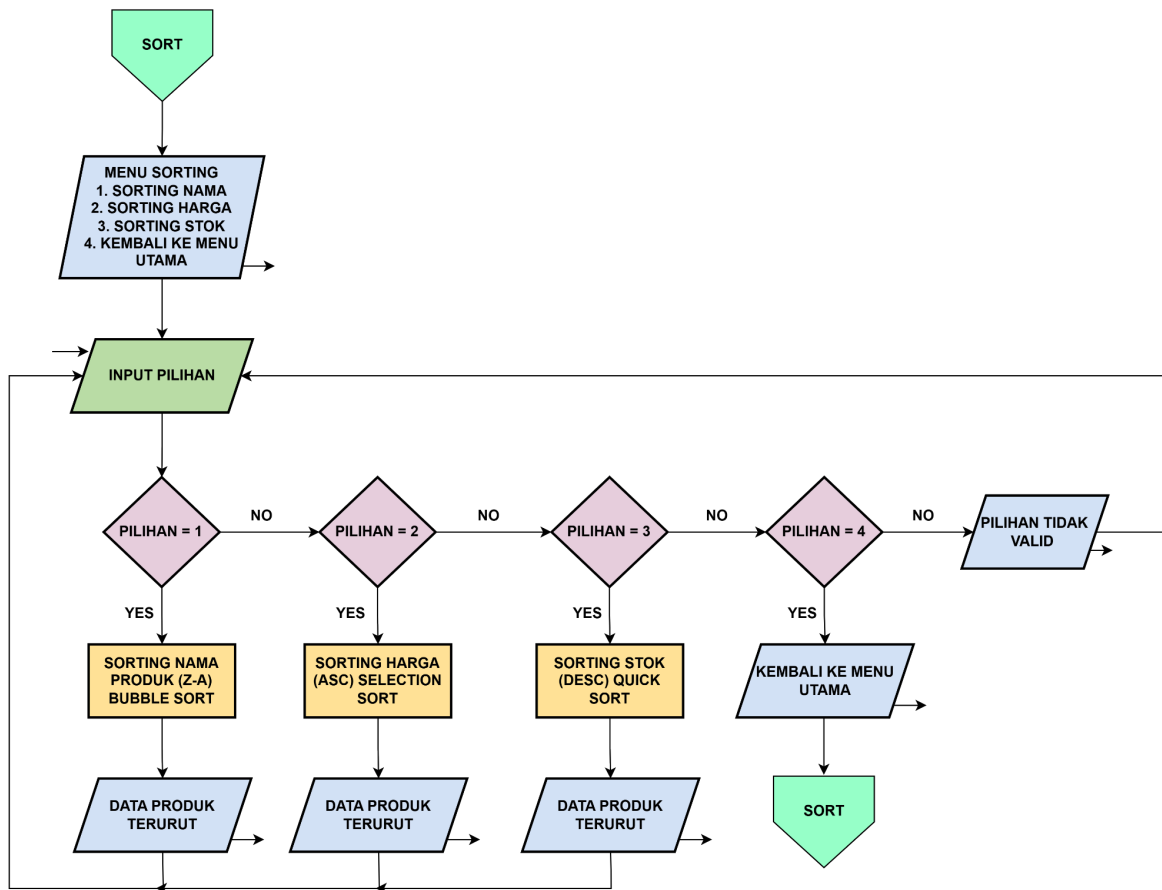
Disusun oleh:
Alya Mayasha (2409106054)
Kelas (B1'24)

PROGRAM STUDI INFORMATIKA
UNIVERSITAS MULAWARMAN
SAMARINDA
2025

1. Flowchart



Gambar 1.1 Flowchart Program Manajemen Tas Rajut



Gambar 1.2 Flowchart Menu Sorting Produk

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. Program ini juga menggunakan pointer untuk efisiensi, di mana operator *&* (*address-of*) dipakai untuk mengirimkan alamat variabel ke fungsi, dan operator *** (*dereference*) digunakan untuk mengakses atau mengubah nilai dari alamat tersebut. Dengan cara ini, perubahan yang dilakukan di dalam fungsi akan langsung mempengaruhi data asli.

Program ini juga menyediakan fitur sorting produk untuk membantu pengguna mengurutkan data berdasarkan kebutuhan. Sorting nama produk dilakukan menggunakan Bubble Sort, yaitu dengan membandingkan dua produk bertetangga dan menukarnya jika urutannya tidak sesuai, sehingga nama produk akan terurut dari Z ke A. Sorting harga produk menggunakan Selection Sort, dengan cara mencari harga termurah di sisa daftar dan menukarnya ke posisi yang benar, sehingga produk akan terurut dari harga termurah ke termahal. Sedangkan sorting stok produk menggunakan Quick Sort, yaitu algoritma rekursif yang membagi data berdasarkan pivot, sehingga stok akan diurutkan dari jumlah terbanyak ke jumlah tersedikit. Penggunaan tiga jenis algoritma ini membuat pengelolaan data produk menjadi lebih fleksibel dan efektif.

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 Login

Fungsi login digunakan untuk memverifikasi pengguna dengan memasukkan nama dan NIM. Pengguna diberikan tiga kali kesempatan untuk memasukkan data dengan benar. Jika gagal, program akan berhenti.

Source Code:

```
bool login(string username, string nim) {
    string inputnama, inputnim;
    int percobaan = 3;
    while (percobaan > 0) {
        cout << "Masukkan Nama: ";
        cin >> inputnama;
        cout << "Masukkan NIM: ";
        cin >> inputnim;

        if (inputnama == username && inputnim == nim) {
            return true;
        } else {
            percobaan--;
            cout << "Login gagal! Sisa percobaan: " << percobaan << endl;
        }
    }
    return false;
}
```

```

    }
}
return false;
}

```

C. Fitur Tambah Produk

Fitur ini memungkinkan pengguna untuk menambahkan produk baru ke dalam daftar jika kapasitas belum penuh.

Source Code:

```

int tambahProduk(Produk *produk, int *jumlahproduk) {
    if (*jumlahproduk < max_produk) {
        cin.ignore();
        cout << "Masukkan nama produk: ";
        getline(cin, (produk + *jumlahproduk)->nama);
        cout << "Masukkan harga produk: ";
        cin >> (produk + *jumlahproduk)->harga;
        cin.ignore();
        cout << "Masukkan warna produk: ";
        getline(cin, (produk + *jumlahproduk)->detail.warna);
        cout << "Masukkan stok produk: ";
        cin >> (produk + *jumlahproduk)->detail.stok;
        cout << "Produk berhasil ditambahkan!\n";
        (*jumlahproduk)++;
        tampilkanProduk(produk, jumlahproduk);
    } else {
        cout << "Data penuh! Tidak bisa menambah produk lagi.\n";
    }
    return *jumlahproduk;
}

```

D. Fitur Tampilkan Produk

Menampilkan daftar produk dalam bentuk tabel yang rapi.

Source Code:

```

void tampilkanProduk(Produk *produk, int *jumlahproduk) {
    if (*jumlahproduk == 0) {
        cout << "Belum ada produk yang tersedia.\n";
    } else {
        cout <<
"\n+---+-----+-----+-----+-----+\n";
        cout << " | No | Nama Produk          | Harga      | Stok | Warna\n";
        cout <<

```

```

"+-----+-----+-----+-----+-----+-----+\n";
    for (int i = 0; i < *jumlahproduk; i++) {
        cout << " | " << setw(2) << i + 1
            << " | " << left << setw(18) << (produk + i)->nama
            << " | " << right << setw(10) << (produk + i)->harga
            << " | " << setw(4) << (produk + i)->detail.stok
            << " | " << setw(10) << left << (produk + i)->detail.warna << "
        |\n";
    }
    cout <<
"+-----+-----+-----+-----+-----+-----+\n";
}
}

```

E. Fitur Ubah Produk

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

Source Code:

```

void ubahProduk(Produk &produk) {
    cout << "Masukkan harga baru: ";
    cin >> produk.harga;
    cout << "Masukkan stok baru: ";
    cin >> produk.detail.stok;
    cout << "Produk berhasil diperbarui!\n";
}

```

F. Fitur Hapus Produk

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

Source Code:

```

int hapusProduk(Produk *produk, int &jumlahproduk) {
    if (jumlahproduk == 0) {
        cout << "Tidak ada produk yang bisa dihapus.\n";
        return jumlahproduk;
    }

    tampilkanProduk(produk, &jumlahproduk);
    int index;
    cout << "Masukkan nomor produk yang ingin dihapus: ";
    cin >> index;
    if (index > 0 && index <= jumlahproduk) {
        index--;
    }
}

```

```

        for (int i = index; i < jumlahproduk - 1; i++) {
            *(produk + i) = *(produk + i + 1);
        }
        jumlahproduk--;
        cout << "Produk berhasil dihapus!\n";
    } else {
        cout << "Nomor produk tidak valid!\n";
    }
    return jumlahproduk;
}

```

G. Fitur Sorting Produk

Fitur ini mempermudah pengguna mengurutkan daftar produk berdasarkan kriteria tertentu: nama produk, harga produk, dan stok produk. Program menyediakan tiga algoritma sorting yang berbeda sesuai kebutuhan.

Source Code:

```

void sortNamaProduk(Produk *produk, int jumlahproduk) {
    for (int i = 0; i < jumlahproduk - 1; i++) {
        for (int j = 0; j < jumlahproduk - i - 1; j++) {
            if ((produk + j)->nama < (produk + j + 1)->nama) {
                Produk temp = *(produk + j);
                *(produk + j) = *(produk + j + 1);
                *(produk + j + 1) = temp;
            }
        }
    }
    cout << "Produk berhasil disorting berdasarkan nama (Z-A) dengan Bubble Sort!\n";
}

void sortHarga(Produk *produk, int jumlahproduk) {
    for (int i = 0; i < jumlahproduk - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < jumlahproduk; j++) {
            if ((produk + j)->harga < (produk + minIndex)->harga) {
                minIndex = j;
            }
        }
        if (minIndex != i) {
            Produk temp = *(produk + i);
            *(produk + i) = *(produk + minIndex);
            *(produk + minIndex) = temp;
        }
    }
    cout << "Produk berhasil disorting berdasarkan harga (termurah-termahal) dengan Selection Sort!\n";
}

```



```

}

int partition(Produk *produk, int low, int high) {
    int pivot = (produk + high)->detail.stok;
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if ((produk + j)->detail.stok > pivot) {
            i++;
            Produk temp = *(produk + i);
            *(produk + i) = *(produk + j);
            *(produk + j) = temp;
        }
    }
    Produk temp = *(produk + i + 1);
    *(produk + i + 1) = *(produk + high);
    *(produk + high) = temp;
    return i + 1;
}

void quickSortStok(Produk *produk, int low, int high) {
    if (low < high) {
        int pi = partition(produk, low, high);
        quickSortStok(produk, low, pi - 1);
        quickSortStok(produk, pi + 1, high);
    }
}

void sortStokDesc(Produk *produk, int jumlahproduk) {
    quickSortStok(produk, 0, jumlahproduk - 1);
    cout << "Produk berhasil disorting berdasarkan stok (banyak ke sedikit)
dengan Quick Sort!\n";
}

```

H. Fungsi Utama (main)

Fungsi main() merupakan pusat kendali program, mulai dari login hingga navigasi ke berbagai fitur dalam sistem manajemen produk.

Source Code:

```

int main() {
    string username = "Alya";
    string nim = "054";

    Produk produk[max_produk];
    int jumlahproduk = 5;

    produk[0] = {"Backpack Rajut", 300000, {"Pink", 20}};

```

```

produk[1] = {"Tote Bag Rajut", 125000, {"Baby Blue", 45}};
produk[2] = {"Sling Bag Rajut", 95000, {"Hitam", 30}};
produk[3] = {"Hand Bag Rajut", 80000, {"Cream", 15}};
produk[4] = {"Pouch Rajut", 65000, {"Maroon", 50}};

if (!login(username, nim)) {
    cout << "Mohon maaf percobaan login gagal! Program berhenti.\n";
    return 0;
}

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. Sorting Produk\n";
    cout << "6. Keluar\n";
    cout << "Pilihan: ";
    cin >> pilihan;

    switch (pilihan) {
        case 1:
            jumlahproduk = tambahProduk(produk, &jumlahproduk);
            break;
        case 2:
            tampilkanProduk(produk, &jumlahproduk);
            break;
        case 3:
            if (jumlahproduk == 0) {
                cout << "Tidak ada produk untuk diubah.\n";
            } else {
                tampilkanProduk(produk, &jumlahproduk);
                int index;
                cout << "Masukkan nomor produk yang ingin diubah: ";
                cin >> index;
                if (index > 0 && index <= jumlahproduk) {
                    ubahProduk(*(produk + index - 1));
                } else {
                    cout << "Nomor produk tidak valid!\n";
                }
            }
            break;
        case 4:
            jumlahproduk = hapusProduk(produk, jumlahproduk);
            break;
        case 5: {
            int opsiSort;
            do {
                cout << "\n=== Menu Sorting ===\n";

```

```

        cout << "1. Sorting Nama (Z-A) [Bubble Sort]\n";
        cout << "2. Sorting Harga (Ascending) [Selection Sort]\n";
        cout << "3. Sorting Stok (Descending) [Quick Sort]\n";
        cout << "4. Kembali ke Menu Utama\n";
        cout << "Pilihan: ";
        cin >> opsiSort;

        switch (opsiSort) {
            case 1:
                sortNamaProduk(produk, jumlahproduk);
                tampilkanProduk(produk, &jumlahproduk);
                break;
            case 2:
                sortHarga(produk, jumlahproduk);
                tampilkanProduk(produk, &jumlahproduk);
                break;
            case 3:
                sortStokDesc(produk, jumlahproduk);
                tampilkanProduk(produk, &jumlahproduk);
                break;
            case 4:
                cout << "Kembali ke menu utama...\n";
                break;
            default:
                cout << "Pilihan tidak valid! Coba lagi.\n";
        }
    } while (opsiSort != 4);
    break;
}

case 6:
    cout << "Terima kasih telah menggunakan program ini. Have a  
great day >_<!\n";
    break;
default:
    cout << "Pilihan tidak valid! Coba lagi.\n";
}
} while (pilihan != 6);

return 0;
}

```

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\coding>
```

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. Sorting Produk
6. 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. Sorting Produk
6. 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. Sorting Produk
6. 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: 6
Masukkan harga baru: 155000
Masukkan stok baru: 5
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. Sorting Produk
6. 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      | 65000    | 50   | Maroon   |
| 6  | Goodie Bag Rajut | 155000   | 5    | Army     |
+---+-----+-----+-----+-----+

Masukkan nomor produk yang ingin dihapus: 6
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. Sorting Produk
6. Keluar
Pilihan: 5

=== Menu Sorting ===
1. Sorting Nama (Z-A) [Bubble Sort]
2. Sorting Harga (Ascending) [Selection Sort]
3. Sorting Stok (Descending) [Quick Sort]
4. Kembali ke Menu Utama
Pilihan: 1
Produk berhasil disorting berdasarkan nama (Z-A) dengan Bubble Sort!

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

Gambar 4.6 Output Ketika Pengguna Memilih Menu Sorting (Pilihan 1)

```
=== Menu Sorting ===
1. Sorting Nama (Z-A) [Bubble Sort]
2. Sorting Harga (Ascending) [Selection Sort]
3. Sorting Stok (Descending) [Quick Sort]
4. Kembali ke Menu Utama
Pilihan: 2
Produk berhasil disorting berdasarkan harga (termurah-termahal) dengan Selection Sort!

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

Gambar 4.7 Output Ketika Pengguna Memilih Menu Sorting (Pilihan 2)

```
=== Menu Sorting ===
1. Sorting Nama (Z-A) [Bubble Sort]
2. Sorting Harga (Ascending) [Selection Sort]
3. Sorting Stok (Descending) [Quick Sort]
4. Kembali ke Menu Utama
Pilihan: 3
Produk berhasil disorting berdasarkan stok (banyak ke sedikit) dengan Quick Sort!

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

Gambar 4.8 Output Ketika Pengguna Memilih Menu Sorting (Pilihan 3)

```
=== Menu Sorting ===
1. Sorting Nama (Z-A) [Bubble Sort]
2. Sorting Harga (Ascending) [Selection Sort]
3. Sorting Stok (Descending) [Quick Sort]
4. Kembali ke Menu Utama
Pilihan: 4
Kembali ke menu utama...

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

Gambar 4.9 Output Ketika Pengguna Memilih Menu Sorting (Pilihan 4)

```
=== Manajemen Produk Tas Rajut ===  
1. Tambah Produk  
2. Tampilkan Produk  
3. Ubah Produk  
4. Hapus Produk  
5. Sorting Produk  
6. Keluar  
Pilihan: 6  
Terima kasih telah menggunakan program ini. Have a great day >_<!  
PS C:\Users\alyam\coding>
```

Gambar 4.10 Output Ketika Pengguna Memilih Pilihan 6

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-N8AD402 MINGW64 ~/desktop/praktikum-apl/post-test/post-test-6 (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-N8AD402 MINGW64 ~/desktop/praktikum-apl/post-test/post-test-6 (main)
$ git commit -m "file cpp dan exe done"
[main 98f9cf6] file cpp dan exe done
2 files changed, 268 insertions(+)
create mode 100644 post-test/post-test-6/2409106054-AlyaMayasha-PT-6.cpp
create mode 100644 post-test/post-test-6/2409106054-AlyaMayasha-PT-6.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-N8AD402 MINGW64 ~/desktop/praktikum-apl/post-test/post-test-6 (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), 681.01 KiB | 4.28 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
458b929..98f9cf6 main -> main
```

Gambar 5.3 Git Push Origin Main