LAPORAN PRAKTIKUM
POSTTEST 5
ALGORITMA PEMROGRAMAN LANJUT

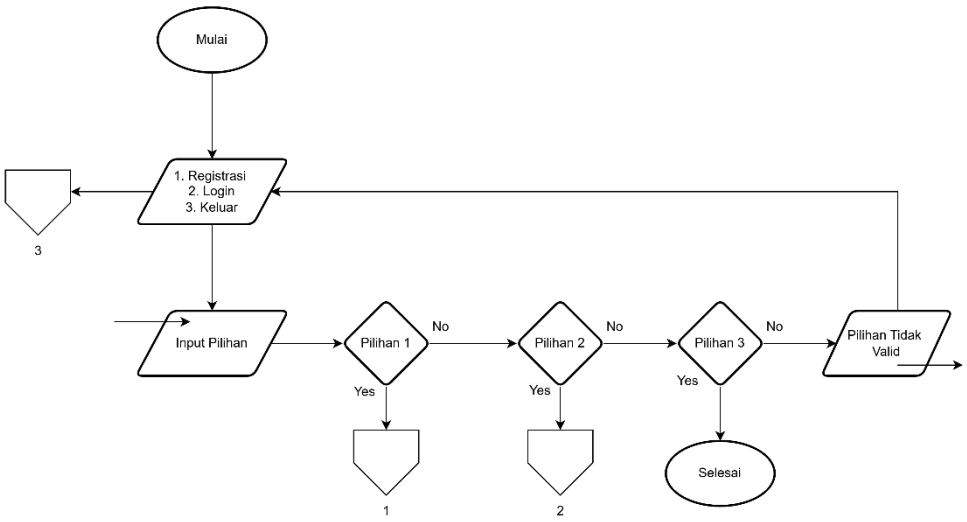
Disusun oleh:

Renaya Putri Alika (2409106002)

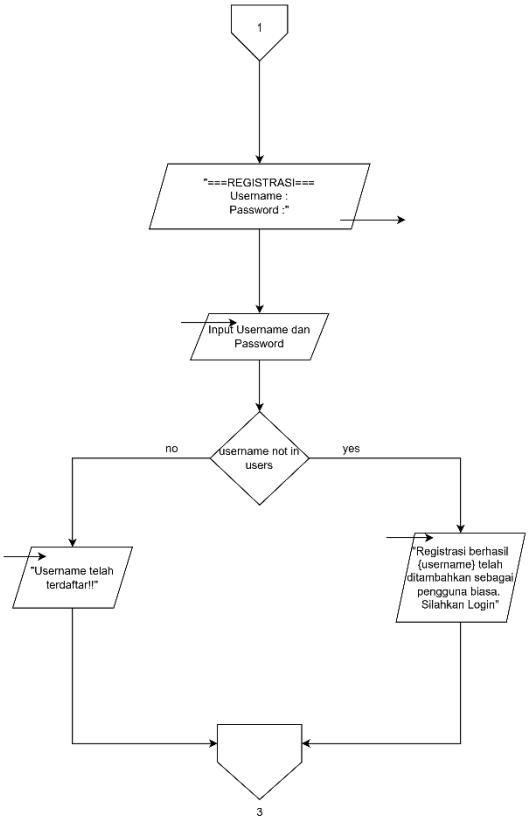
Kelas A1’24

PROGRAM STUDI INFORMATIKA
UNIVERSITAS MULAWARMAN SAMARINDA
2025

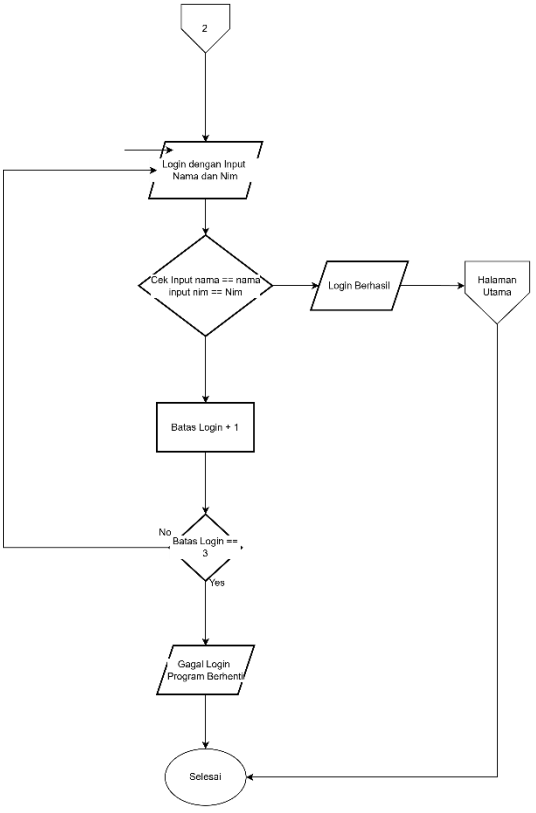
1. Flowchart



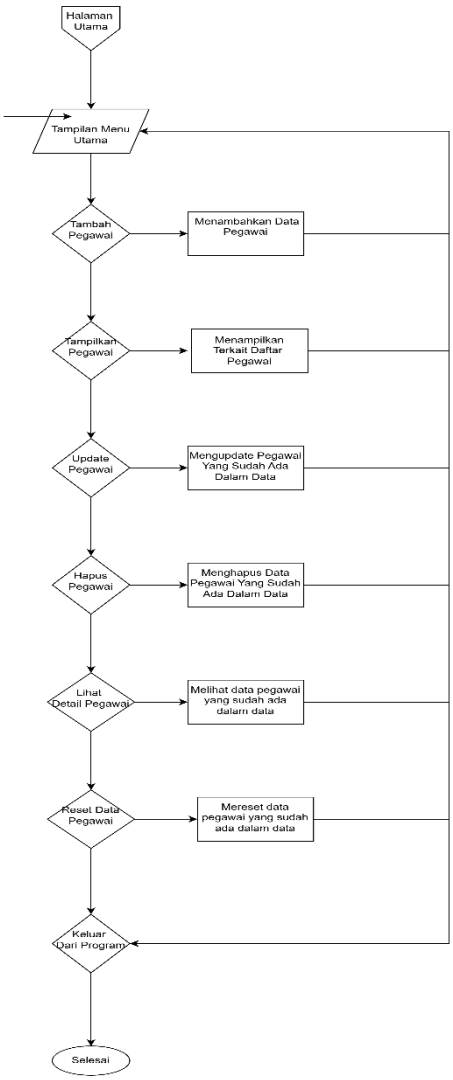
Flowchart 1.1 Pilihan Menu



Flowchart 1.2 Registrasi



Flowchart 1.3 Login



Flowchart 1.4 Halaman Utama

2. Analisis Program

2.1 Deskripsi Singkat Program

Program ini merupakan aplikasi manajemen pegawai sederhana berbasis teks yang dibuat dengan bahasa C++. Program diawali dengan proses registrasi dan login oleh pengguna menggunakan nama dan NIM. Setelah berhasil login, pengguna diarahkan ke menu utama yang menyediakan berbagai fitur untuk mengelola data pegawai. Fitur-fitur yang tersedia meliputi penambahan data pegawai, penampilan data (dengan dua versi overloading: semua pegawai dan berdasarkan nama), pembaruan data, penghapusan data, melihat detail data pegawai dalam bentuk tabel, dan mereset seluruh data. Penampilan data juga melibatkan fungsi rekursif untuk latihan konsep pemrograman rekursi. Sementara itu, fitur reset data menggunakan pointer untuk mengakses dan

memodifikasi nilai variabel secara langsung. Program ini menggabungkan konsep-konsep penting dalam C++ seperti struct, array, function, overloading, rekursi, dan pointer untuk membentuk sistem manajemen data yang fungsional.

3. Source Code

3.1 Registrasi Pengguna

Menambahkan user baru ke dalam array users selama belum mencapai batas maksimum.

```
void registerUser(User users[], int &jumlah_user) {
    if (jumlah_user < MAX_USERS) {
        cout << "\n=== Register Akun ===\nMasukkan Nama: ";
        getline(cin, users[jumlah_user].username);
        cout << "Masukkan NIM: ";
        getline(cin, users[jumlah_user].nim);
        jumlah_user++;
        cout << "Registrasi berhasil! Silakan login.\n";
    } else {
        cout << "Jumlah user maksimal telah tercapai!\n";
    }
}
```

3.2 Login dengan Validasi dan Batas Percobaan

Memverifikasi identitas pengguna dan menghentikan program jika gagal login 3 kali.

```
bool loginUser(User users[], int jumlah_user, string &loggedInUser) {
    string username, nim;
    int attempt = 0;
    while (attempt < 3) {
        cout << "\n=== Login ===\nMasukkan Nama: ";
        getline(cin, username);
        cout << "Masukkan NIM: ";
        getline(cin, nim);
        for (int i = 0; i < jumlah_user; i++) {
            if (users[i].username == username && users[i].nim == nim) {
                loggedInUser = username;
                return true;
            }
        }
        cout << "Login gagal!\n";
        attempt++;
    }
    cout << "Anda gagal login 3 kali. Program berhenti.\n";
    exit(0);
}
```

3.3 Menu Manajemen Pegawai

Menampilkan pilihan-pilihan untuk mengelola data pegawai setelah login berhasil.

```
void menuPegawai(Pegawai pegawai[], int &jumlah_pegawai) {
    int pilihan;
    do {
        cout << "\n=== Menu Manajemen Pegawai ===\n";
        cout << "1. Tambah\n2. Tampilkan\n3. Update\n4. Hapus\n";
        cout << "5. Lihat Detail Pegawai\n6. Reset Data\n7. Logout\nPilihan:
";
        cin >> pilihan;
        cin.ignore();
        switch (pilihan) {
            case 1:
                tambahPegawai(pegawai, jumlah_pegawai);
                break;
                case 2:
                int subPilihan;
                cout << "\n1. Tampilkan Semua Pegawai\n2. Cari Pegawai
Berdasarkan Nama\nPilihan: ";
                cin >> subPilihan;
                cin.ignore();
                if (subPilihan == 1) {
```

```
                    tampilkanPegawai(pegawai, jumlah_pegawai); // overloading
1
                } else if (subPilihan == 2) {
                    string namaCari;
                    cout << "Masukkan nama pegawai yang dicari: ";
                    getline(cin, namaCari);
                    tampilkanPegawai(pegawai, jumlah_pegawai, namaCari); //
overloading 2
                }
                 else {
                    cout << "Pilihan tidak valid.\n";
                }
                break;
            case 3:
                updatePegawai(pegawai, jumlah_pegawai);
                break;
            case 4:
                hapusPegawai(pegawai, jumlah_pegawai);
                break;
            case 5:
                cariDanTampilkanPegawai(pegawai, jumlah_pegawai);
                break;
            case 6:
                resetDataPegawai(pegawai, &jumlah_pegawai);
                break;
        }
    } while (pilihan != 7);
}
```

3.4 Menampilkan Pegawai

Menampilkan data pegawai satu per satu secara rekursif

```
void tampilkanPegawaiRekursif(Pegawai pegawai[], int index, int
jumlah_pegawai) {
    if (index == 0) {
        cout << left << setw(5) << "No"
             << setw(20) << "Nama"
             << setw(15) << "Waktu Masuk"
             << setw(15) << "Waktu Keluar"
             << setw(20) << "Hasil Kerja" << endl;
        cout << string(75, '-') << endl;
    }

    if (index >= jumlah_pegawai) return;

    cout << left << setw(5) << index + 1
         << setw(20) << pegawai[index].nama
         << setw(15) << pegawai[index].waktu_masuk
         << setw(15) << pegawai[index].waktu_keluar
```

```
        << setw(20) << pegawai[index].hasil_kerja << endl;

    tampilkanPegawaiRekursif(pegawai, index + 1, jumlah_pegawai);
}
```

3.5 Fungsi Overloading

Fungsi ini memiliki dua versi :

- Menampilkan semua pegawai
- Menampilkan hanya pegawai tertentu berdasarkan nama yang dicari.

```
void tampilkanPegawai(Pegawai pegawai[], int jumlah_pegawai); // overloading 1
void tampilkanPegawai(Pegawai pegawai[], int jumlah_pegawai, string nama); //
overloading 2
```

3.6 Update dan Hapus Data Pegawai

Memungkinkan pengguna untuk memperbarui atau menghapus data pegawai dengan memilih berdasarkan nomor urut.

```
void updatePegawai(Pegawai pegawai[], int jumlah_pegawai);
void hapusPegawai(Pegawai pegawai[], int &jumlah_pegawai);
```

4 Uji Coba dan Hasil Output

```
=== Sistem Manajemen Pegawai ===
1. Register
2. Login
3. Keluar
Pilihan: 1

=== Register Akun ===
Masukkan Nama: Nayul
Masukkan NIM: 002
Registrasi berhasil! Silakan login.

=== Sistem Manajemen Pegawai ===
1. Register
2. Login
3. Keluar
Pilihan: 2

=== Login ===
Masukkan Nama: Nayul
Masukkan NIM: 002
```

Output 4.1 Register & Login

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 1
Nama Pegawai: nana
Waktu Masuk: 6.20
Waktu Keluar: 19.00
Hasil Kerja: 700.000
Pegawai ditambahkan.

=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 1
Nama Pegawai: dewi
Waktu Masuk: 8.15
Waktu Keluar: 21.00
Hasil Kerja: 800.000
Pegawai ditambahkan.
```

Output 4.2 Tambah Pegawai (Create)

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 2

1. Tampilkan Semua Pegawai
2. Cari Pegawai Berdasarkan Nama
Pilihan: 1
1. nana
2. dewi
3. angel
4. ayu
```

Output 4.3 Tampilkan Semua Pegawai

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 2

1. Tampilkan Semua Pegawai
2. Cari Pegawai Berdasarkan Nama
Pilihan: 2
Masukkan nama pegawai yang dicari: angel
Mencari pegawai bernama: angel
3. angel, Masuk: 9.30, Keluar: 23.00, Hasil: 1.000.000
```

Output 4.4 Cari Pegawai Berdasarkan Nama

```
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 3
1. nana
2. dewi
3. angel
4. ayu
Masukkan nomor pegawai: 3
Nama Baru: nilu
Waktu Masuk Baru: 6.00
Waktu Keluar Baru: 19.20
Hasil Kerja Baru: 700.000
Data diperbarui.

=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 2

1. Tampilkan Semua Pegawai
2. Cari Pegawai Berdasarkan Nama
Pilihan: 1
1. nana
2. dewi
3. nilu
4. ayu
```

Output 4.5 Update Pegawai

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 4
1. nana
2. dewi
3. nilu
4. ayu
Masukkan nomor pegawai: 3
Data berhasil dihapus.
```

Output 4.6 Delete Pegawai

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 5

=== Detail Data Pegawai ===
No   Nama               Waktu Masuk    Waktu Keluar   Hasil Kerja
-------------------------------------------------------------------------
1    nana               6.20           19.00          700.000
2    dewi               8.15           21.00          800.000
3    ayu                9.30           23.30          1.050.000
```

Output 4.7 Lihat Detail Pegawai

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 6
Apakah Anda yakin ingin mereset semua data pegawai? (y/n): y
Seluruh data pegawai berhasil di-reset.
```

Output 4.8 Reset Data Pegawai

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Lihat Detail Pegawai
6. Reset Data
7. Logout
Pilihan: 7

=== Sistem Manajemen Pegawai ===
1. Register
2. Login
3. Keluar
Pilihan: 3
Program berhenti.
```

Output 4.9 Keluar Program & Berhenti

5. GIT

1. Git Init adalah perintah Git yang digunakan untuk menginisialisasi repository Git baru dalam sebuah folder. Cukup ketik "git init" pada terminal VSCode.

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git init
Reinitialized existing Git repository in C:/Users/LENOVO/Downloads/Praktikum-apl/.git/
```

Gambar 5.1 Git Init

2. Git Add adalah perintah Git untuk menandai semua perubahan (file baru, diubah, atau dihapus) di folder saat ini agar siap disimpan ke dalam versi berikutnya di Git.

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git add .
```

Gambar 5.2 Git Add

3. Git commit adalah perintah Git yang digunakan untuk menyimpan perubahan yang telah ditambahkan ke staging area ke dalam repository. Setiap commit akan memiliki hash unik, pesan commit, dan menyimpan snapshot dari perubahan yang dilakukan.

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git commit -m "Finish PostTest 5"
[main 6b8b555] Finish PostTest 5
 3 files changed, 313 insertions(+)
```

Gambar 5.3 Git Commit

\

4. Git remote adalah perintah Git yang digunakan untuk menghubungkan repository lokal dengan repository remote (misalnya di GitHub, GitLab, atau Bitbucket). Dengan perintah ini, kita bisa mengelola koneksi ke repository jarak jauh, memungkinkan push, pull, dan fetch dari repository tersebut.

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git remote set-url origin https://github.com/Renaya0207/praktikum-apl.git
```

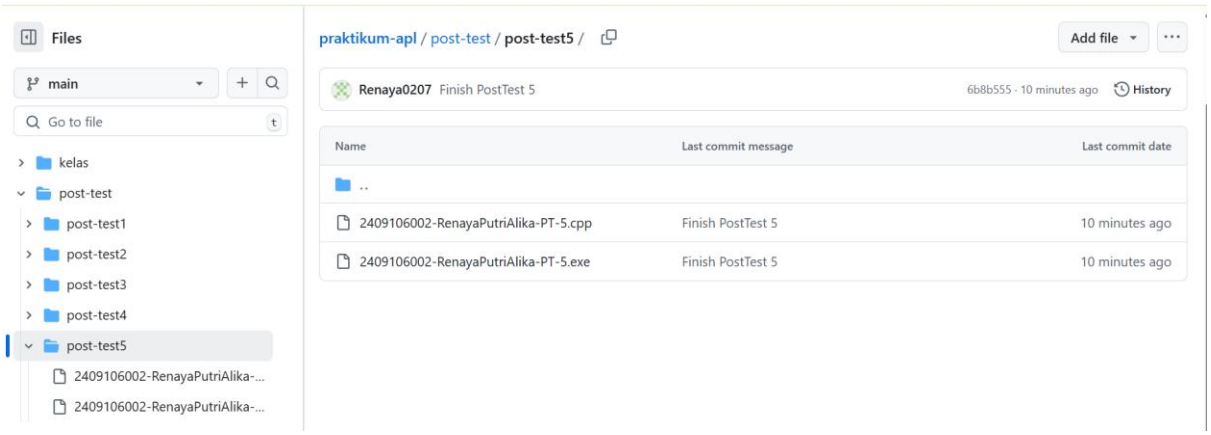
Gambar 5.4 Git Remote

5. Git Push adalah perintah Git yang digunakan untuk mengirim (mengunggah) perubahan dari repository lokal ke repository remote (seperti GitHub, GitLab, atau Bitbucket). Perintah ini memastikan bahwa perubahan yang sudah dikomit di lokal tersedia di repository jarak jauh sehingga bisa diakses oleh orang lain atau untuk cadangan.

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git push origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 679.90 KiB | 3.43 MiB/s, done.
Total 9 (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/Renaya0207/praktikum-apl.git
   292c5eb..6b8b555  main -> main
PS C:\Users\LENOVO\Downloads\Praktikum-apl>
```

Gambar 5.5 Git Push

6. Berhasil di Up



Gambar 5.6 Berhasil