LAPORAN PRAKTIKUM

POSTTEST 4

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Renaya Putri Alika (2409106002)

Kelas A1’24

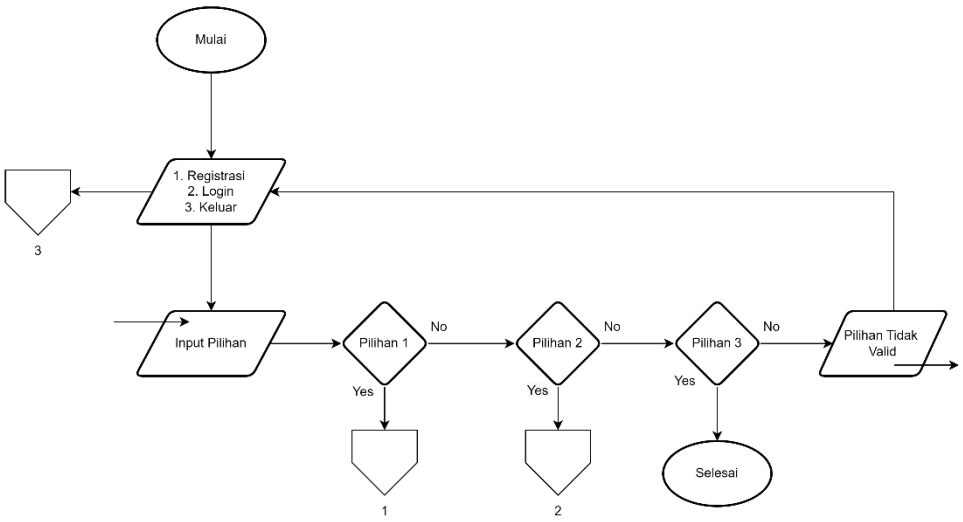
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

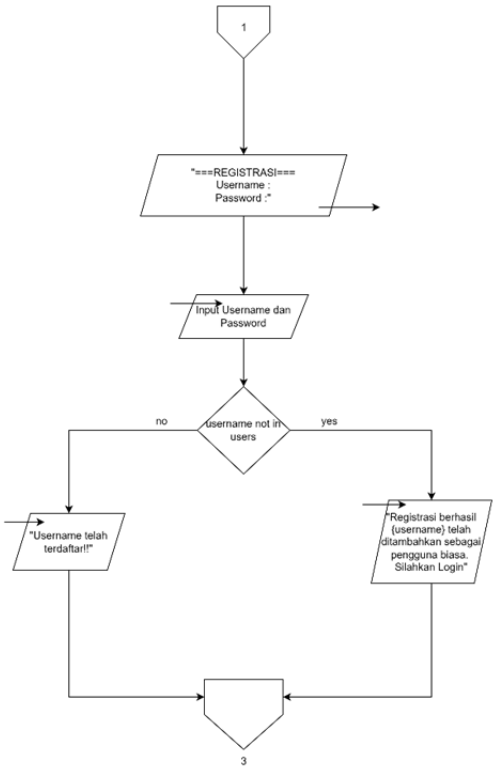
SAMARINDA

2025

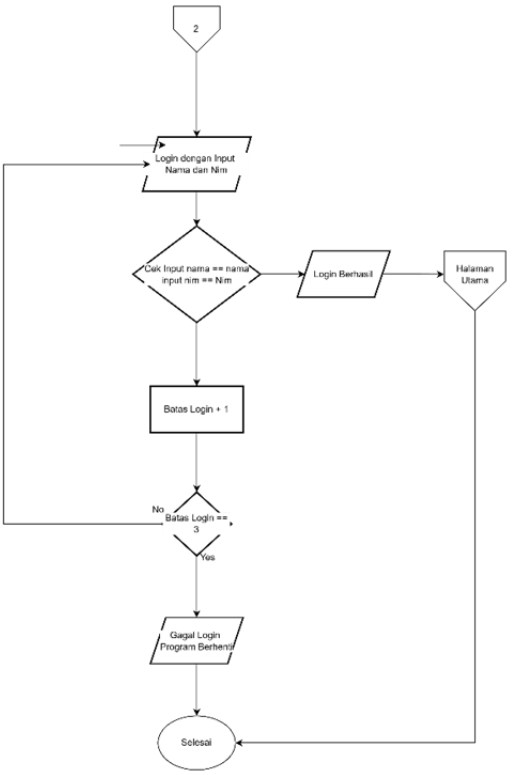
1. Flowchart



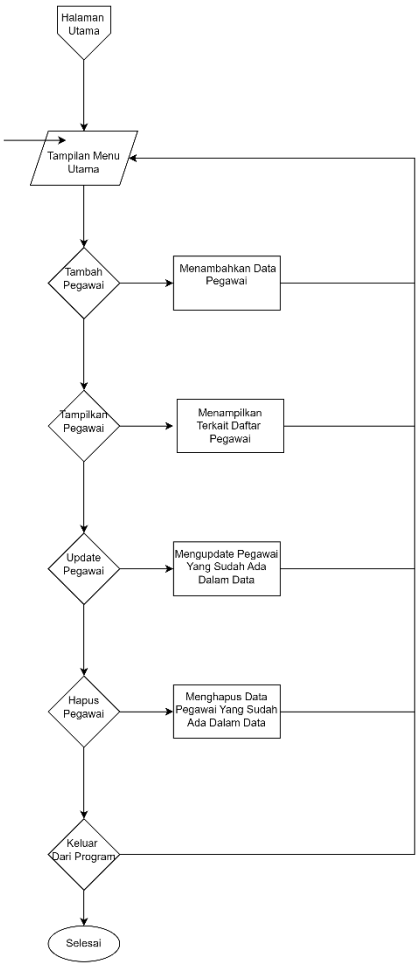
Flowchart 1.1 Pilihan Menu



Flowchart 1.2 Register



Flowchart 1.3 Login



Flowchart 1.4 Halaman Utama

2. Analisis Program

2.1 Deskripsi Singkat Program

Program ini merupakan aplikasi untuk manajemen data pegawai yang ditulis dalam bahasa C++. Program menyediakan fitur registrasi dan login untuk pengguna dengan membatasi jumlah maksimum user. Setelah berhasil login, pengguna dapat mengakses menu manajemen pegawai yang memungkinkan mereka untuk menambah, menampilkan, memperbarui, dan menghapus data pegawai. Data pegawai disimpan dalam array struct yang berisi informasi seperti nama, waktu masuk, waktu keluar, dan hasil kerja.

Tampilan data pegawai menggunakan fungsi rekursif, dan terdapat fungsi overloading yang memungkinkan pengguna untuk menampilkan seluruh data pegawai atau hanya data pegawai tertentu berdasarkan nama. Struktur program dibangun secara modular menggunakan fungsi dan prosedur, sehingga lebih terorganisir dan mudah untuk dikembangkan lebih lanjut.

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";
login_attempts++;
            }             if (login_attempts == 3) {                 cout <<
"Anda telah gagal login sebanyak 3 kali. Program berhenti.\n";
return 0;
            }             do {
```

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\n5.
Logout\nPilihan: ";
        cin >> pilihan;
        cin.ignore();
        switch (pilihan) {
            case 1:
                tambahPegawai(pegawai, jumlah_pegawai);
                break;
            case 2:
                tampilkanPegawaiRekursif(pegawai, 0, jumlah_pegawai);
                break;
            case 3:
                updatePegawai(pegawai, jumlah_pegawai);
                break;
            case 4:
                hapusPegawai(pegawai, jumlah_pegawai);
                break;
        }
    } while (pilihan != 5);
}
```

3.4 Menampilkan Pegawai

Menampilkan data pegawai satu per satu secara rekursif

```
void tampilkanPegawaiRekursif(Pegawai pegawai[], int index, int jumlah_pegawai)
{
    if (index >= jumlah_pegawai) return;
    cout << index + 1 << ". " << pegawai[index].nama << ", Masuk: " <<
pegawai[index].waktu_masuk
         << ", Keluar: " << pegawai[index].waktu_keluar
         << ", Hasil: " << 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); // 1
void tampilkanPegawai(Pegawai pegawai[], int jumlah_pegawai, string nama); // 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. Logout
Pilihan: 1
Nama Pegawai: Captain Marvel
Waktu Masuk: 7.20
Waktu Keluar: 18.40
Hasil Kerja: 500.00
Pegawai ditambahkan.

=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Logout
Pilihan: 1
Nama Pegawai: Nebula
Waktu Masuk: 8.15
Waktu Keluar: 16.05
Hasil Kerja: 450.000
Pegawai ditambahkan.

=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Logout
Pilihan: 1
Nama Pegawai: Black Widow
Waktu Masuk: 9.10
Waktu Keluar: 16.10
Hasil Kerja: 450.000
Pegawai ditambahkan.
```

Output 4.2 Tambah Pegawai (Create)

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Logout
Pilihan: 2
No   Nama              Waktu Masuk    Waktu Keluar   Hasil Kerja
-------------------------------------------------------------------------
1    Captain Marvel    7.20           18.40          500.00
2    Nebula            8.15           16.05          450.000
3    Black Widow       9.10           16.10          450.000
```

Output 4.3 Menampilkan Pegawai (Read)

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Logout
Pilihan: 3
1. Captain Marvel
2. Nebula
3. Black Widow
Masukkan nomor pegawai: 2
Nama Baru: Valkyrie
Waktu Masuk Baru: 9.10
Waktu Keluar Baru: 17.15
Hasil Kerja Baru: 455.000
Data diperbarui.

=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Logout
Pilihan: 2
No   Nama               Waktu Masuk    Waktu Keluar   Hasil Kerja
-------------------------------------------------------------------------
1    Captain Marvel     7.20           18.40          500.00
2    Valkyrie           9.10           17.15          455.000
3    Black Widow        9.10           16.10          450.000
```

Output 4.4 Update Pegawai

```
=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Logout
Pilihan: 4
1. Captain Marvel
2. Valkyrie
3. Black Widow
Masukkan nomor pegawai: 3
Data berhasil dihapus.

=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Logout
Pilihan: 2
No   Nama               Waktu Masuk    Waktu Keluar   Hasil Kerja
-------------------------------------------------------------------------
1    Captain Marvel     7.20           18.40          500.00
2    Valkyrie           9.10           17.15          455.000

=== Menu Manajemen Pegawai ===
1. Tambah
2. Tampilkan
3. Update
4. Hapus
5. Logout
Pilihan: 5
```

Output 4.5 Delete Pegawai

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

Output 4.6 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 commit Git 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 Post Test 3"
[detached HEAD 021384a] Finish Post Test 3
 11 files changed, 410 insertions(+), 2 deletions(-)
 create mode 100644 kelas/pertemuan-3/pertemuan3.cpp
 delete mode 160000 post-test
 delete mode 160000 post-test-1
 create mode 100644 post-test/post-test1/2409106002-RenayaPutriAlika-PT-1.cpp
 create mode 100644 post-test/post-test1/2409106002-RenayaPutriAlika-PT-1.exe
 create mode 100644 post-test/post-test1/2409106002-RenayaPutriAlika-PT-1.pdf
 rename {post-test-2 => post-test/post-test2}/2409106002-RenayaPutriAlika-PT-2.cpp (100%)
 rename {post-test-2 => post-test/post-test2}/2409106002-RenayaPutriAlika-PT-2.exe (100%)
 rename {post-test-2 => post-test/post-test2}/2409106002-RenayaPutriAlika-PT-2.pdf (100%)
 create mode 100644 post-test/post-test3/2409106002-RenayaPutriAlika-PT-3.cpp
 create mode 100644 post-test/post-test3/2409106002-RenayaPutriAlika-PT-3.exe
```

Gambar 5.2 Git Commit

3. 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
```

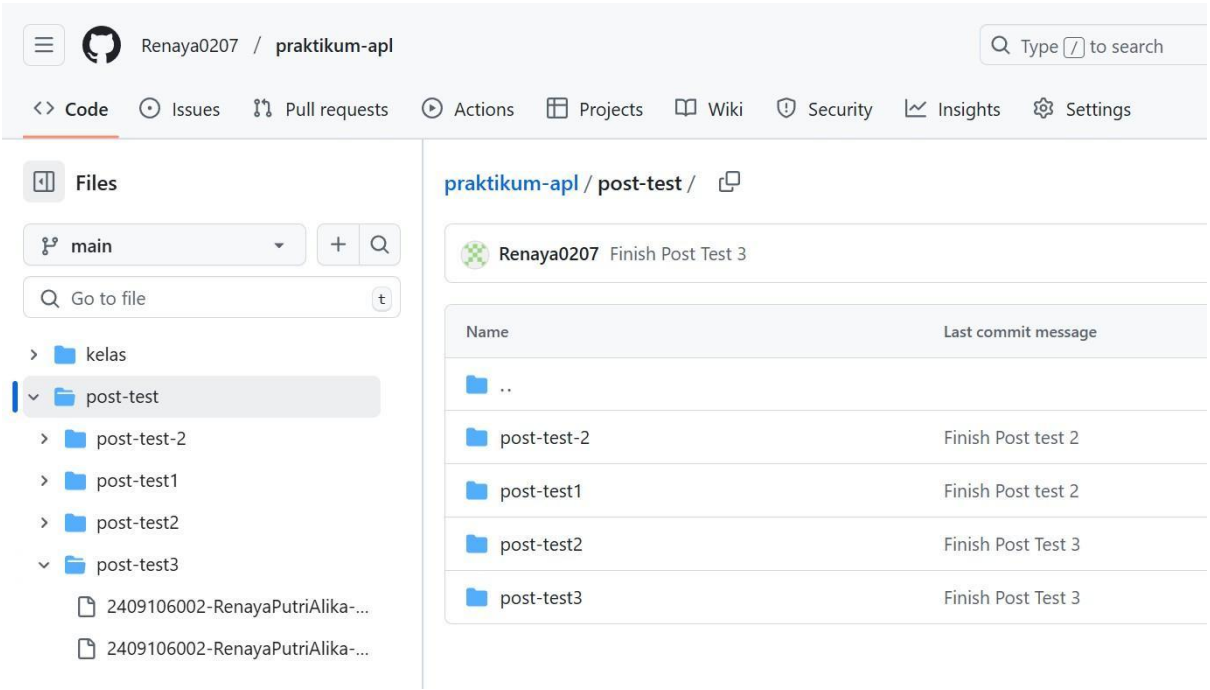
Gambar 5.3 Git Remote

4. 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), 677.85 KiB | 3.85 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Renaya0207/praktikum-apl.git
   280ff4f..828add4  main -> main
PS C:\Users\LENOVO\Downloads\Praktikum-apl>
```

Gambar 5.4 Git Push

5. Berhasil di Up



Gambar 5.5 Berhasil