

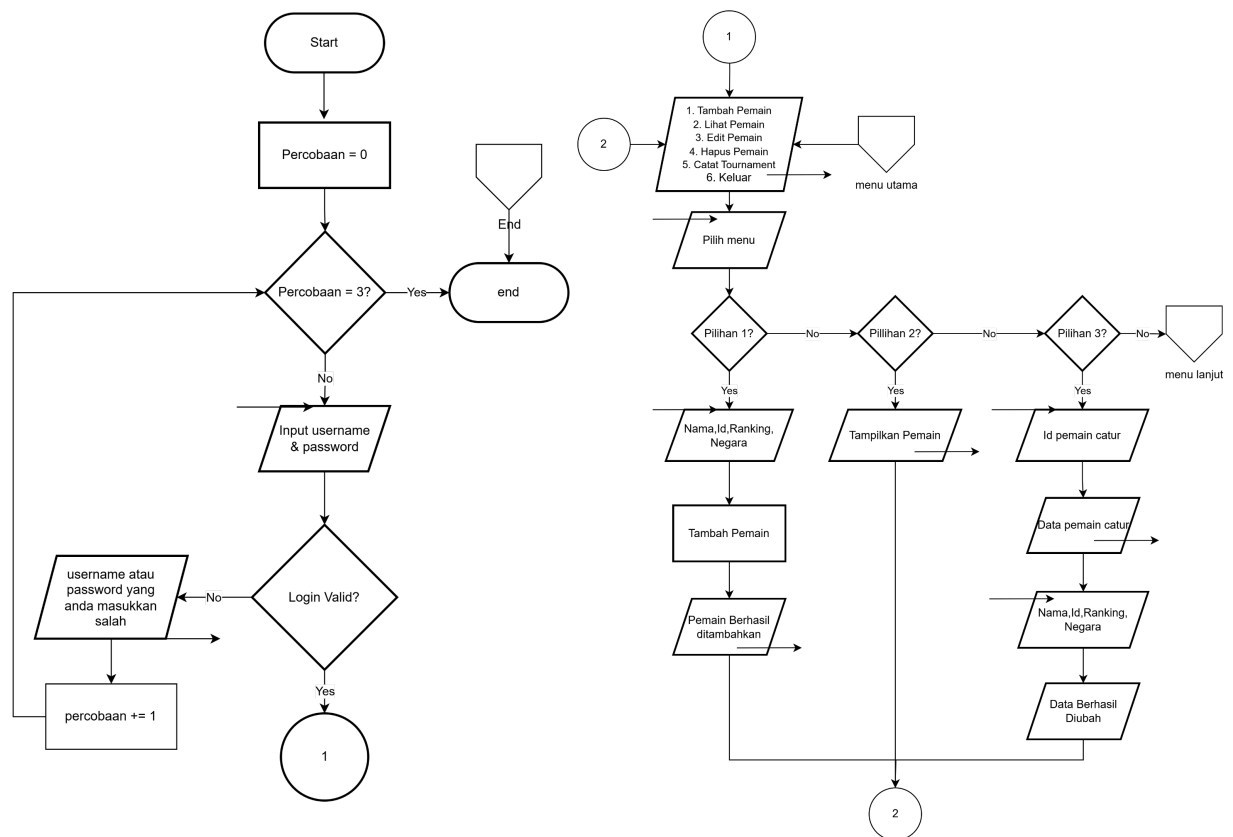
**LAPORAN PRAKTIKUM**  
**POSTTEST 2**  
**ALGORITMA PEMROGRAMAN LANJUT**



**Disusun oleh:**  
**Muhammad Nabil Rahmatullah (2409106046)**  
**Kelas (B1 '24)**

**PROGRAM STUDI INFORMATIKA**  
**UNIVERSITAS MULAWARMAN**  
**SAMARINDA**  
**2025**

## 1. Flowchart

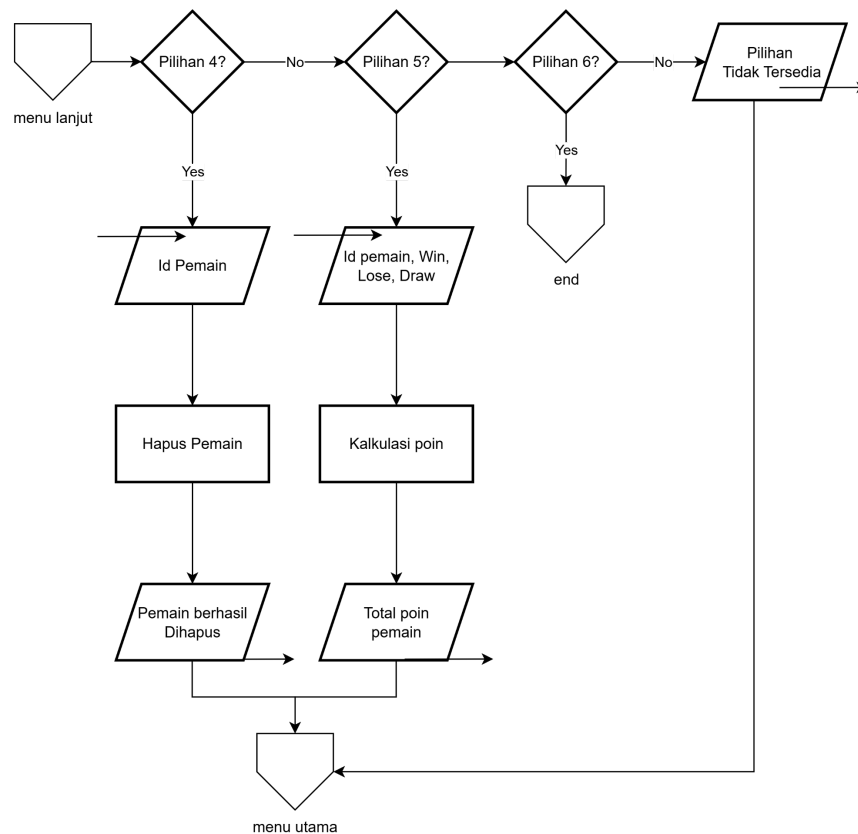


Gambar 1.1 Flowchart Login dan menu

Pada Langkah pertama program akan melakukan pengecekan apakah percobaan sama dengan 3 atau tidak. Jika tidak sama maka program akan meminta *user* untuk melakukan *input username* dan *password* untuk melakukan validasi. Jika *login* tersebut dinyatakan valid, maka program akan melanjutkan ke menu utama. Namun jika *login* tersebut tidak valid maka program akan memunculkan *output* “username atau password yang anda masukkan salah” dan akan menambah percobaan sebanyak 1 kali, dan program akan kembali ke atas untuk melakukan pengecekan apakah program sama dengan 3? Jika iya maka program akan otomatis terhenti.

Kemudian jika *user* telah berhasil melakukan *login* maka program akan melanjutkan ke menu utama. Pada menu utama, *user* akan diminta untuk memilih menu mana yang ia inginkan. Jika *user* memilih menu pertama, maka *user* akan diminta melakukan *input* Nama, Id (*unique id*), *Ranking*, dan *Negara*. Setelah itu program akan memberi *output* “Pemain Berhasil Ditambahkan”

Jika *user* memilih menu ke dua, maka *program* akan menampilkan daftar pemain dalam bentuk tabel. Namun jika *user* memilih menu ke tiga maka *user* akan diminta melakukan *input* “*id user*” dan setelah itu program akan memberikan data dari pemain tersebut. Dan *user* diminta untuk melakukan input seperti pada menu pertama, jika user tidak melakukan *input* maka nilai yang dipakai adalah nilai sebelumnya.



Jika *user* memilih menu ke empat maka *user* diminta melakukan *input* “Id Pemain” dan program akan mencari id pemain tersebut dan akan menghapusnya, jika *user* memilih menu ke lima, maka *user* diminta untuk memilih *user* mana yang akan dipilih melalui “*unique id*” yang di input oleh *user*. Setelah itu *user* diminta menulis angka sebanyak tiga kali untuk mengisi poin “Win, Lose, Draw” dari pemain yang dipilih, setelah itu program akan mengkalkulasikan poin dan akan menampilkan total poin dari pemain tersebut.

Jika user melakukan *input* selain angka 1-6 maka akan muncul *output* “Pilihan tidak tersedia” dan *user* akan kembali ke menu utama. Namun jika *user* memilih menu ke enam, maka *user* akan keluar dari program.

## 2. Analisis Program

### 2.1 Deskripsi Singkat Program

Tujuan dari program ini adalah untuk mempermudah pengelolaan data pemain catur agar tersusun dengan rapi, mudah diakses, dan dapat diperbarui dengan efisien. Program ini memungkinkan pengguna untuk menambahkan, mengedit, menghapus, serta melihat data pemain dengan sistem yang terstruktur, sehingga memudahkan dalam pencatatan riwayat pertandingan, peringkat, serta informasi penting lainnya.

### 2.2 Penjelasan Alur & Algoritma

Pada awal program *user* diberi kesempatan sebanyak tiga kali dengan cara membuat variabel “percobaan = 0” yang akan bertambah sebanyak satu setiap kali *user* gagal *login*.

Lalu program akan meminta *user* memasukkan *username* dan *password*, autentikasi dilakukan dengan membandingkan *username* dan *password* yang dimasukkan dengan *username* dan *password* yang benar.

Jika *user* gagal login sebanyak tiga kali, maka program akan terhenti. Setelah *login* berhasil, program menampilkan menu utama dalam *loop while*. Program menggunakan struktur *if-else* untuk menangani berbagai jenis menu. Navigasi menu menggunakan tombol panah (*arrow*) atas/bawah dan enter untuk memilih. Menu yang terpilih akan ditampilkan dengan *highlight*.

#### Algoritma Penambahan Pemain

1. Meminta input nama pemain dengan validasi panjang maksimal 11 karakter
2. Meminta ID pemain dengan validasi untuk memastikan ID unik
3. Meminta ranking dengan validasi:
4. Memastikan input berupa angka
5. Memastikan ranking belum digunakan oleh pemain lain
6. Meminta input negara pemain
7. Menginisialisasi data turnamen (menang, seri, kalah) dan total poin ke 0
8. Menambah jumlah pemain dan menyimpan data ke dalam array

#### Algoritma Tampilan Pemain

1. Memeriksa apakah ada pemain yang terdaftar
2. Menghitung total poin untuk setiap pemain menggunakan formula:
3. Total poin = (jumlah kemenangan x 3) + (jumlah seri x 1) - (jumlah kalah x 2)
4. Menampilkan data dalam format tabel

### **Algoritma Edit Pemain**

1. Meminta ID pemain yang akan diedit
2. Mencari pemain dengan ID yang sesuai
3. Menampilkan data pemain saat ini
4. Meminta input untuk data baru:
  - a. Nama
  - b. Ranking
  - c. Negara
5. Jika input kosong, data lama dipertahankan
6. Menyimpan data yang diperbarui

### **Algoritma Hapus Pemain**

1. Meminta ID pemain yang akan dihapus
2. Mencari pemain dengan ID yang sesuai
3. Menghapus data dengan menggeser data pemain lain untuk mengisi posisi yang dihapus
4. Mengurangi jumlah pemain

### **Algoritma Catat Tournament**

1. Meminta ID pemain
2. Mencari pemain dengan ID yang sesuai
3. Meminta input jumlah kemenangan, seri, dan kekalahan
4. Menghitung total poin menggunakan formula:
  - a.  $\text{Total poin} = (\text{jumlah kemenangan} \times 3) + (\text{jumlah seri} \times 1) - (\text{jumlah kekalahan} \times 2)$
5. Menyimpan data turnamen dan total poin yang baru

### **Fitur Pada Program**

1. **Masking password saat login**
2. **Validasi panjang nama pemain**
3. **Validasi keunikan ID pemain**
4. **Validasi input ranking (harus berupa angka dan unik)**
5. **Penanganan input kosong pada saat edit data**
6. **Bubble sort untuk sistem ranking**

### 3. Source Code

#### A. Fitur Clear Terminal

Fitur ini digunakan untuk membersihkan terminal agar pada saat program berjalan akan terlihat lebih rapi.

**Source code:**

```
system("cls");
```

*Gambar 3.1 Code clear*

#### B. Fitur Sistem Keamanan Login

Fitur ini berfungsi untuk menghindari aksi pembobolan oleh oknum yang tidak bertanggung jawab. Pada fitur ini saya memberi kesempatan untuk *login* sebanyak tiga kali. Jika *user* mencoba *login* lebih dari tiga kali, maka akun akan terblokir secara otomatis.

Dan juga saya menggunakan “*getch()*” agar pada saat *user* melakukan *input* PIN di terminal hanya terlihat simbol “\*” yang bertujuan untuk mencegah aksi pembobolan.

**Source code:**

```
password = "";  
char ch;  
while ((ch = _getch()) != 13) {  
    if (ch == 8) {  
        if (!password.empty()) {  
            cout << "\b \b";  
            password.pop_back();  
        }  
    } else {  
        cout << '*';  
        password += ch;  
    }  
}  
cout << endl;
```

*Gambar 3.2 Fitur masking*

```

        password = "";
        char ch;
        while ((ch = _getch()) != 13) {
            if (ch == 8) {
                if (!password.empty()) {
                    cout << "\b \b";
                    password.pop_back();
                }
            } else {
                cout << '*';
                password += ch;
            }
        }
        cout << endl; -" << percobaan << "\n";
    }

    if (!loginSukses) {
        cout << "Kesempatan habis, coba lagi nanti.\n";
        return 0;
    }
}

```

### C. Fitur Validasi Input

Fitur ini bertujuan agar *user* tidak melakukan *input* diluar dari yang kita inginkan, contohnya pada saat *user* ingin melakukan *input* “id pemain” maka program akan melakukan pengecekan apakah *Id* tersebut sudah terdaftar atau belum, sehingga tidak terjadi data yang terduplikat.

**Source code:**

```

while (!idValid) {
    cout << "Masukkan ID Pemain: ";
    cin >> inputID;

    idValid = true;

    for (int i = 0; i < jumlahPemain; i++) {
        if (pemain[i][1] == inputID) {
            idValid = false;
            cout << "ID " << inputID << " sudah digunakan!\n";
        }
    }
}

```

Gambar 3.3 Fitur Validasi

## D. Fitur Highlight

Fitur ini berfungsi untuk memilih menu “arrow” pada *keyboard* kita. Pada baris ke dua terlihat “`i == highlightArr`” yang berarti menu yang ditunjuk oleh “`i`” akan terhighlight sebagai petunjuk navigasi. Case 72 digunakan untuk panah atas dan case 80 untuk panah bawah, lalu “`else-if (key==13)`” untuk tombol enter dan terakhir kita akan konversi indeks pilihan ke nomor pada menu.

### Source code:

```
for (int i = 0; i < 6; i++) {
    if (i == highlightArr) {
        cout << WHITE_BG << RED << menuI[i] << RESET << endl;
    } else {
        cout << menuI[i] << endl;
    }
}
int key = _getch();

if (key == 224) {
    key = _getch();

    switch (key) {
        case 72:
            highlightArr = (highlightArr == 0) ? 5 : highlightArr - 1;
            break;
        case 80:
            highlightArr = (highlightArr == 5) ? 0 : highlightArr + 1;
            break;
    }
} else if (key == 13) {
    pilihan = highlightArr + 1;
}
```

Gambar 3.4 Fitur Highlight



## E. Fitur Create

Fitur ini berfungsi untuk menambahkan daftar pemain kedalam *array* yang telah disediakan, dengan cara meminta *user* melakukan *input* sesuai kebutuhan program, dan akan divalidasi setiap *inputnya* apakah sesuai dengan kriteria yang dibutuhkan program atau tidak. Jika sesuai dengan kriteria maka data pemain tersebut akan ditambahkan kedalam *array*.

### Source code:

```
if (pilihan == 1) {
    if (jumlahPemain < maxPemain) {
        string inputNama;
        bool namaValid = false;

        while (!namaValid) {
            cout << "Masukkan Nama Pemain (maksimal 11 karakter): ";
            cin >> inputNama;

            if (inputNama.length() <= 11) {
                namaValid = true;
                pemain[jumlahPemain][0] = inputNama;
            } else
                cout << "Nama tidak boleh lebih dari 11 karakter!";
        }

        out << "Masukkan Negara: ";
        cin >> pemain[jumlahPemain][3];
        turnamen[jumlahPemain][0] = 0;
        turnamen[jumlahPemain][1] = 0;
        turnamen[jumlahPemain][2] = 0;
        totalPoin[jumlahPemain] = 0;
        jumlahPemain++;
        cout << "Pemain berhasil ditambahkan!\n";
    }
}
```

Gambar 3.5 FiturCreate

## F. Fitur Read

Fitur ini berfungsi untuk membaca dan memunculkan data dalam bentuk tabel yang berwarna. Disini program menggunakan “setw( )” untuk mengatur lebar teks.

```
cout << "\nDaftar Pemain Catur:\n";
cout << BLUE <<
"+-----+-----+-----+-----+-----+-----+-----+-----+\n"
;
cout << "| No | Nama | ID | Ranking| Negara |
W | D | L | Poin |\n";
cout <<
"+-----+-----+-----+-----+-----+-----+-----+-----+"
;
```

```

<< RESET << endl;
    for (int i = 0; i < jumlahPemain; i++) {
        cout << " | " << setw(3) << i + 1 << " | "
            << GREEN << setw(10) << left << pemain[i][0] <<
RESET << " | "
" | "
            << YELLOW << setw(4) << pemain[i][1] << RESET <<
            << setw(6) << right << pemain[i][2] << " | "
            << setw(7) << left << pemain[i][3] << " | "
            << RED << setw(3) << right << turnamen[i][0] <<
RESET << " | "
" | "
            << RED << setw(3) << turnamen[i][1] << RESET <<
            << RED << setw(3) << turnamen[i][2] << RESET <<
            << YELLOW << setw(5) << totalPoin[i] << RESET <<
" | \n";
    }
    cout << BLUE <<
"+-----+-----+-----+-----+-----+-----+-----+-----+"
<< RESET << endl;

```

Gambar 3.6 Fitur Read

## G. Fitur Update

Fitur ini berfungsi untuk melakukan perubahan pada data pemain yang ingin diubah. Dengan menggunakan *looping for* untuk mencari id pemain yang ingin diubah datanya, jika ditemukan maka program akan menampilkan data terkini dari pemain tersebut dan akan memberi opsi kepada *user* apakah ingin mengubahnya atau tetap mempertahankan nilai yang lama dengan cara memberi *input* kosong.

```

for (int i = 0; i < jumlahPemain; i++) {
    if (pemain[i][1] == id) {
        string newName, newRanking, newNegara;

        cout << "Data saat ini: " << endl;
        cout << "Nama: " << pemain[i][0] << endl;
        cout << "Ranking: " << pemain[i][2] << endl;
        cout << "Negara: " << pemain[i][3] << endl <<
endl;

        cout << "Masukkan Negara Baru (kosongkan untuk
tetap): ";
    }
}

```

```

        getline(cin, newNegara);
        if (!newNegara.empty()) {
            pemain[i][3] = newNegara;
        }

        cout << "Data pemain berhasil diperbarui!\n";
        ditemukan = true;

```

Gambar 3.7 Fitur Update

## H. Fitur Delete

Fitur ini berfungsi untuk menghapus data pemain yang ada pada *array* sebelumnya dan menggeser data yang berada disebelahnya agar menempati posisi yang kosong karena telah dihapus.

```

cout << "Masukkan ID pemain yang ingin dihapus: ";
cin >> id;
bool ditemukan = false;
for (int i = 0; i < jumlahPemain; i++) {
    if (pemain[i][1] == id) {
        for (int j = i; j < jumlahPemain - 1; j++) {
            pemain[j][0] = pemain[j + 1][0];
            pemain[j][1] = pemain[j + 1][1];
            pemain[j][2] = pemain[j + 1][2];
            pemain[j][3] = pemain[j + 1][3];
            turnamen[j][0] = turnamen[j + 1][0];
            turnamen[j][1] = turnamen[j + 1][1];
            turnamen[j][2] = turnamen[j + 1][2];
            totalPoin[j] = totalPoin[j + 1];
        }
        jumlahPemain--;
        cout << "Pemain berhasil dihapus!\n";
        ditemukan = true;
        break;
    }
}
if (!ditemukan) {
    cout << "Pemain dengan ID tersebut tidak ditemukan.\n";
}

```

Gambar 3.8 Fitur Delete

## I. Fitur Poin

Fitur ini berfungsi untuk menyimpan data statistik dari pemain yang telah mengikuti turnamen, dengan cara menambah poin sebanyak 3 jika menang, 1 jika draw, dan mengurangi poin sebanyak 2 jika kalah.

```

cout << "Masukkan ID pemain: ";
    cin >> id;
    bool ditemukan = false;
    for (int i = 0; i < jumlahPemain; i++) {
        if (pemain[i][1] == id) {
            cout << "Masukkan jumlah kemenangan: ";
            cin >> turnamen[i][0];
            cout << "Masukkan jumlah seri: ";
            cin >> turnamen[i][1];
            cout << "Masukkan jumlah kekalahan: ";
            cin >> turnamen[i][2];

            totalPoin[i] = (turnamen[i][0] * 3) +
(turnamen[i][1] * 1) - (turnamen[i][2] * 2);

            cout << "Data turnamen diperbarui!\n";
            cout << "Total poin pemain sekarang: " <<
totalPoin[i] << "\n";

            ditemukan = true;
            break;
        }
    }
    if (!ditemukan) {
        cout << "Pemain dengan ID tersebut tidak ditemukan.\n";
    }
}

```

*Gambar 3.9 Fitur Poin*

## J. Fitur Sorting Ranking

Fitur ini berfungsi untuk menempatkan posisi pemain sesuai dengan poin tertinggi hingga terendah, sehingga *user* tidak perlu memeriksa secara manual.

### Source code:

```

for (int a = 0; a < jumlahPemain - 1; a++) {
    for (int b = 0; b < jumlahPemain - a - 1; b++) {
        if (totalPoin[b] < totalPoin[b + 1]) {
            //swap total poin
            int tempPoin = totalPoin[b];
            totalPoin[b] = totalPoin[b + 1];
            totalPoin[b + 1] = tempPoin;

            //tukar pemain
            string tempNama = pemain[b][0];
            string tempID = pemain[b][1];
            string tempRanking = pemain[b][2];
            string tempNegara = pemain[b][3];

```

```

    pemain[b][0] = pemain[b + 1][0];
    pemain[b][1] = pemain[b + 1][1];
    pemain[b][2] = pemain[b + 1][2];
    pemain[b][3] = pemain[b + 1][3];

    pemain[b + 1][0] = tempNama;
    pemain[b + 1][1] = tempID;
    pemain[b + 1][2] = tempRanking;
    pemain[b + 1][3] = tempNegara;

    int tempWin = turnamen[b][0];
    int tempDraw = turnamen[b][1];
    int tempLoss = turnamen[b][2];

    turnamen[b][0] = turnamen[b + 1][0];
    turnamen[b][1] = turnamen[b + 1][1];
    turnamen[b][2] = turnamen[b + 1][2];

    turnamen[b + 1][0] = tempWin;
    turnamen[b + 1][1] = tempDraw;
    turnamen[b + 1][2] = tempLoss;

```

Gambar 3.10 Fitur Sorting

## 4. Uji Coba dan Hasil Output

### 4.1 Uji Coba

Untuk memastikan program berjalan dengan baik dan sesuai dengan keinginan kita, dilakukan beberapa skenario pengujian dengan berbagai jenis input. Berikut adalah beberapa skenario pengujian yang diterapkan:

#### Skenario 1: Pengujian Login

- **Input:**
  - Memasukkan *username* dan *password* yang benar
  - Memasukkan *username* atau *password* yang salah hingga batas maksimum percobaan.
- **Ekspektasi Hasil:**
  - Jika *username* dan *password* benar, pengguna dapat mengakses menu utama.
  - Jika *username* atau *password* salah hingga 3 kali, program akan berhenti.

#### Skenario 2: Pengujian Menu Create

- **Input:**
  - Memasukkan nama lebih dari 11 karakter
  - Memasukkan id yang telah ada
  - Memasukkan rangking yang telah ada
  - Memasukkan sesuai dengan kriteria
- **Ekspektasi Hasil:**
  - Error : nama tidak boleh lebih dari 11 karakter
  - Error : id sudah terdaftar
  - Error : rangking sudah ada
  - Pemain bertambah

### **Skenario 3: Pengujian Tampilkan pemain**

- **Input:**
  - Menampilkan pemain pada saat data kosong
  - Menampilkan pemain pada saat data terisi
- **Ekspektasi Hasil:**
  - Terdapat *output* “Belum ada pemain yang terdaftar.”
  - Menampilkan data pemain dalam bentuk tabel.

### **Skenario 4: Pengujian Edit Pemain**

- **Input:**
  - Id pemain yang ada dan melakukan perubahan sesuai kriteria
  - Id pemain yang tidak terdaftar
- **Ekspektasi Hasil:**
  - Mengubah data pemain sesuai dengan *input* dari *user*
  - Menampilkan “Pemain dengan ID tersebut tidak ditemukan”

### **Skenario 5: Pengujian Hapus Pemain**

- **Input:**
  - Memasukkan ID pemain yang terdaftar
  - Memasukkan ID pemain yang tidak terdaftar.
- **Ekspektasi Hasil:**
  - Menampilkan “Pemain berhasil dihapus”

- Menampilkan “Pemain dengan ID tersebut tidak ditemukan”

#### Skenario 6: Pengujian Catat Turnamen

- **Input:**
  - Memasukkan *input* menang 1, kalah 1, draw 1
  - Memasukkan *input* berupa *non-numerik*
- **Ekspektasi Hasil:**
  - Program menampilkan “Poin 2”
  - Program menampilkan “Error : *input* harus berupa angka”

#### Skenario 7: Pengujian Sorting Rangking

- **Input:**
  - Memasukkan *input* menang 1, kalah 0, draw 1 pada *user a*
  - Memasukkan *input* menang 5, kalah 0, draw 0 pada *user b*
- **Ekspektasi Hasil:**
  - Program mengurutkan dari *user b*.

## 4.2 Hasil Output

```
Masukkan Username: nabil
Masukkan Password: ***
Login gagal! Percobaan ke-1
Masukkan Username: ucup
Masukkan Password: ***
Login gagal! Percobaan ke-2
Masukkan Username: tes
Masukkan Password: ****
Login gagal! Percobaan ke-3
Kesempatan habis, coba lagi nanti.
```

Gambar 4.2.1 Output jika gagal login sebanyak tiga kali

```

1. Tambah Pemain
2. Tampilkan Pemain
3. Edit Pemain
4. Hapus Pemain
5. Catat Turnamen
6. Keluar
Gunakan panah atas/bawah dan Enter untuk memilih
|

```

Gambar 4.2.2 Output jika user berhasil login

```

Masukkan Nama Pemain (maksimal 11 karakter): inilebihdarisebelaskarakterges
Error: Nama tidak boleh lebih dari 11 karakter!

```

Gambar 4.2.3 Output jika user memasukkan nama pemain lebih dari sebelas karakter

```

Masukkan Nama Pemain (maksimal 11 karakter): magnus
Masukkan ID Pemain: 1A
Masukkan Negara: India
Pemain berhasil ditambahkan!
Press any key to continue . . . |

```

Gambar 4.2.4 Output jika user melakukan input sesuai kriteria

```

Daftar Pemain Catur:
+-----+-----+-----+-----+-----+-----+-----+-----+
| No  | Nama      | ID   | Ranking| Negara  | W   | D   | L   | Poin  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1   | magnus    | 1A   | 1      | India   | 0   | 0   | 0   | 0     |
+-----+-----+-----+-----+-----+-----+-----+-----+
Press any key to continue . . . |

```

Gambar 4.2.5 Output jika memilih daftar pemain

```

Belum ada pemain yang terdaftar.
Press any key to continue . . . |

```

Gambar 4.2.6 Output jika tidak ada pemain yang terdaftar



```
Masukkan ID pemain yang ingin diedit: 3
Pemain dengan ID tersebut tidak ditemukan.
Press any key to continue . . . |
```

*Gambar 4.2.7 Output jika pemain yang di edit tidak ditemukan*

```
Masukkan ID pemain yang ingin diedit: 1A
Data saat ini:
Nama: Magnus
Ranking: 1 (otomatis diperbarui berdasarkan poin)
Negara: india

Masukkan Nama Baru (kosongkan untuk tetap, maks 11 karakter): MagnusGambit
Error: Nama tidak boleh lebih dari 11 karakter!
Nama tidak diubah.
Masukkan Negara Baru (kosongkan untuk tetap): |
```

*Gambar 4.2.8 Output jika mengedit nama pemain lebih dari 11 karakter*

```
Masukkan ID pemain yang ingin diedit: 1A
Data saat ini:
Nama: Magnus
Ranking: 1 (otomatis diperbarui berdasarkan poin)
Negara: india

Masukkan Nama Baru (kosongkan untuk tetap, maks 11 karakter): Gambit
Masukkan Negara Baru (kosongkan untuk tetap):
Data pemain berhasil diperbarui!
Press any key to continue . . . |
```

*Gambar 4.2.9 Output jika berhasil melakukan edit*

```
Masukkan ID pemain yang ingin dihapus: 3
Pemain dengan ID tersebut tidak ditemukan.
Press any key to continue . . . |
```

*Gambar 4.2.10 Output jika melakukan hapus pemain yang tidak terdaftar*

```
Masukkan ID pemain yang ingin dihapus: 1A
Pemain berhasil dihapus!
Press any key to continue . . . |
```

*Gambar 4.2.11 Output jika berhasil menghapus pemain*

```
Masukkan ID pemain: 2
Pemain dengan ID tersebut tidak ditemukan.
Press any key to continue . . . |
```

Gambar 4.2.12 Output jika tidak menemukan pemain di menu catat turnamen

```
Masukkan ID pemain: 1
Masukkan jumlah kemenangan (hanya angka): asd
Error: Input harus berupa angka!
Masukkan jumlah kemenangan (hanya angka): |
```

Gambar 4.2.13 Output jika melakukan input non-numerik pada catat turnamen

```
Masukkan ID pemain: 1
Masukkan jumlah kemenangan (hanya angka): 1
Masukkan jumlah seri (hanya angka): 1
Masukkan jumlah kekalahan (hanya angka): 0
Data turnamen diperbarui!
Total poin pemain sekarang: 4
Ranking diperbarui berdasarkan poin.
Press any key to continue . . . |
```

Gambar 4.2.14 Output jika berhasil melakukan catat turnamen

Daftar Pemain Catur:

No	Nama	ID	Ranking	Negara	W	D	L	Poin
1	userA	1	1	id	1	1	0	4
2	userB	2	2	sg	0	0	0	0

Press any key to continue

Gambar 4.2.15 Tabel sorting

Daftar Pemain Catur:

No	Nama	ID	Ranking	Negara	W	D	L	Poin
1	userB	2	1	sg	5	0	0	15
2	userA	1	2	id	1	1	0	4

Press any key to continue . . .

Gambar 4.2.16 Tabel sorting

Keluar dari program.  
Press any key to continue . . .

Gambar 4.2.17 Output jika keluar dari program

## 5. Langkah-Langkah GIT

### 5.1 GIT ADD

Berfungsi untuk menandai *file* mana yang ada perubahan dengan cara menulis “git add “nama file” atau gunakan “.” jika ingin menandai semua file

```
PS D:\praktikum-apl\post-test\post-test-apl-2> git add .
```

Gambar 5.1.1 Melakukan git add

### 5.2 GIT COMMIT

Berfungsi untuk melakukan perubahan pada *file* yang kita pantau dan dapat memberi keterangan apa yang kita ubah pada *file* tersebut. Caranya dengan menulis ‘git commit -m “pesan yang ingin ditulis”’. Dan juga kita bisa melakukan rename dengan cara “ren namafilelama namafilebaru”

```
PS D:\praktikum-apl\post-test\post-test-apl-2> ren tes.cpp 2409106046-MuhammadNabilRahmatullah-PT-2.cpp
PS D:\praktikum-apl\post-test\post-test-apl-2> ren a.exe 2409106046-MuhammadNabilRahmatullah-PT-2.exe
PS D:\praktikum-apl\post-test\post-test-apl-2> git status
On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    a.exe
    deleted:    tes.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    2409106046-MuhammadNabilRahmatullah-PT-2.cpp
    2409106046-MuhammadNabilRahmatullah-PT-2.exe

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\praktikum-apl\post-test\post-test-apl-2> git add .
PS D:\praktikum-apl\post-test\post-test-apl-2> git commit -m "Finish post test 2"
[main 255a52a] Finish post test 2
 2 files changed, 0 insertions(+), 0 deletions(-)
 rename tes.cpp => 2409106046-MuhammadNabilRahmatullah-PT-2.cpp (100%)
 rename a.exe => 2409106046-MuhammadNabilRahmatullah-PT-2.exe (100%)
PS D:\praktikum-apl\post-test\post-test-apl-2> |
```

Gambar 5.2.1 Melakukan git commit

### 5.3 GIT REMOTE

Berfungsi untuk menyambungkan langsung ke repository yang terdapat pada *repository remote* (*github*) kita.

Dengan cara menulis “git remote add origin link-repo”

```
PS D:\praktikum-apl\post-test\post-test-apl-1> git remote add origin https://github.com/orkvacy/praktikum-apl.git
PS D:\praktikum-apl\post-test\post-test-apl-1> git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 831 bytes | 831.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/orkvacy/praktikum-apl.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Gambar 5.3.1 Melakukan git remote

### 5.4 GIT PUSH

Berfungsi untuk melakukan *push* atau mengirimkan rubahan yang telah di *commit* ke *repository remote* (*github*). Dengan cara menulis “git push -u origin main”

```
PS D:\praktikum-apl\post-test\post-test-apl-2> git pull origin main --rebase
From https://github.com/orkvacy/praktikum-apl
 * branch      main      -> FETCH_HEAD
Successfully rebased and updated refs/heads/main.
PS D:\praktikum-apl\post-test\post-test-apl-2> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 20 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 683.99 KiB | 4.62 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/orkvacy/praktikum-apl.git
 b08be25..eafba49  main -> main
PS D:\praktikum-apl\post-test\post-test-apl-2> |
```

Gambar 5.4.1 melakukan git push