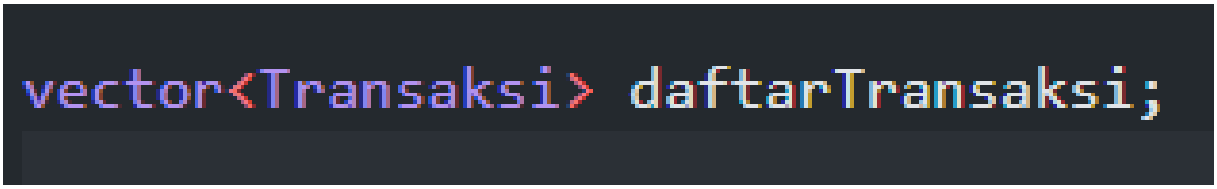
```
struct Transaksi {
    int idTransaksi;
    string jenisLapangan;
    int hargaPerJam;
    int durasiSewa;  // dalam jam
    int totalHarga;
    string waktuSelesai; // waktu selesai booking
};
```

"Pada bagian ini, saya menggunakan struktur data Transaksi untuk menyimpan informasi terkait setiap transaksi penyewaan lapangan futsal. Struktur data ini memiliki beberapa atribut, yaitu:"

1. **idTransaksi**: *"Ini adalah ID unik untuk setiap transaksi, yang digunakan untuk membedakan setiap transaksi yang terjadi. ID ini akan bertambah secara otomatis setiap kali ada transaksi baru."*
2. **jenisLapangan**: *"Atribut ini menyimpan jenis lapangan yang disewa, apakah lapangan standar atau lapangan VIP. Ini akan menentukan harga sewa per jam."*
3. **hargaPerJam**: *"Harga per jam untuk lapangan yang disewa. Lapangan standar memiliki harga Rp 100.000 per jam, sementara lapangan VIP memiliki harga Rp 150.000 per jam."*
4. **durasiSewa**: *"Ini adalah durasi sewa dalam jam. Pengguna akan memilih berapa lama mereka ingin menyewa lapangan, dan informasi ini akan digunakan untuk menghitung total harga."*
5. **totalHarga**: *"Atribut ini menyimpan total harga yang harus dibayar oleh penyewa. Total harga dihitung berdasarkan harga per jam dikalikan dengan durasi sewa."*
6. **waktuSelesai**: *"Ini adalah waktu yang menunjukkan kapan durasi sewa berakhir, yang dihitung berdasarkan waktu saat transaksi dibuat ditambah dengan durasi sewa yang dipilih. Ini membantu untuk mengingatkan kapan lapangan selesai disewa."*

"Dengan menggunakan struct Transaksi ini, kita bisa dengan mudah menyimpan, mengelola, dan memproses data terkait transaksi yang terjadi, termasuk harga, durasi, dan jenis lapangan yang disewa."



"Di sini, saya menggunakan vector<Transaksi> daftarTransaksi sebagai kontainer dinamis untuk menyimpan daftar transaksi yang terjadi dalam aplikasi ini. Secara spesifik, vector adalah struktur data dari bahasa C++ yang memungkinkan kita menyimpan elemen-elemen yang dapat diakses secara efisien dan otomatis menyesuaikan ukuran berdasarkan jumlah elemen yang ada."

1. **Kontainer Dinamis:** *"Karena menggunakan vector, kita tidak perlu khawatir tentang ukuran awalnya. vector akan secara otomatis menambah kapasitasnya ketika elemen baru ditambahkan, sehingga memudahkan kita untuk mengelola daftar transaksi tanpa perlu menentukan kapasitas yang tetap."*
2. **Menyimpan Data Transaksi:** *"Setiap elemen dalam vector ini adalah sebuah objek Transaksi, yang berisi informasi tentang ID transaksi, jenis lapangan, harga per jam, durasi sewa, total harga, dan waktu selesai."*
3. **Keuntungan Penggunaan vector:** *"Salah satu keuntungan menggunakan vector adalah kemudahan dalam menambahkan data baru. Ketika transaksi baru terjadi, kita cukup menambahkan objek Transaksi ke dalam vector, dan data akan secara otomatis disimpan."*
4. **Akses yang Cepat dan Efisien:** *"Selain itu, vector juga memberikan kemudahan dalam mengakses elemen-elemen yang ada menggunakan indeks, sehingga kita dapat dengan cepat melihat transaksi berdasarkan ID atau melakukan operasi lain seperti menghapus transaksi tertentu."*

"Dengan menggunakan vector<Transaksi> daftarTransaksi, kita dapat dengan mudah mengelola data transaksi yang dinamis, memastikan aplikasi dapat menangani penambahan dan penghapusan transaksi tanpa kesulitan."

```
// Fungsi untuk mengatur warna teks
void setColor(int textColor, int bgColor) {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    int colorAttribute = textColor + (bgColor * 16);
    SetConsoleTextAttribute(hConsole, colorAttribute);
}
```

"Fungsi setColor di sini digunakan untuk mengubah warna teks dan latar belakang pada konsol, yang akan membuat tampilan program lebih menarik dan memudahkan pengguna untuk membedakan bagian-bagian yang penting."

1. **Parameter Fungsi:** *"Fungsi ini menerima dua parameter: textColor dan bgColor, yang masing-masing digunakan untuk menentukan warna teks dan latar belakang. Kedua parameter ini berupa angka yang merepresentasikan warna yang diinginkan."*
2. **Mendapatkan Handle Konsol:** *"Di dalam fungsi ini, kita pertama-tama mendapatkan handle untuk konsol dengan menggunakan GetStdHandle(STD_OUTPUT_HANDLE), yang memungkinkan kita untuk melakukan manipulasi output teks pada konsol."*
3. **Menghitung Atribut Warna:** *"Kemudian, kita menghitung nilai atribut warna dengan mengalikan nilai bgColor (warna latar belakang) dengan 16 dan menambahkannya dengan textColor (warna teks). Ini adalah cara untuk mengkombinasikan kedua warna dalam satu nilai yang diterima oleh fungsi SetConsoleTextAttribute."*
4. **Mengatur Warna Teks dan Latar Belakang:** *"Fungsi SetConsoleTextAttribute(hConsole, colorAttribute) digunakan untuk mengubah atribut warna teks dan latar belakang berdasarkan nilai colorAttribute yang telah dihitung."*
5. **Contoh Penggunaan:** *"Sebagai contoh, jika kita ingin teks berwarna merah dengan latar belakang hitam, kita bisa memanggil setColor(RED, BLACK). Ini akan mengubah warna teks menjadi merah dan latar belakang menjadi hitam pada konsol."*

"Dengan menggunakan fungsi ini, kita dapat memperbaiki pengalaman pengguna dengan memberikan warna yang kontras, membuat informasi yang penting lebih menonjol, serta meningkatkan interaksi visual dalam aplikasi."

```
// Fungsi untuk mendapatkan waktu sekarang dan menghitung waktu selesai
string hitungWaktuSelesai(int durasiSewa) {
    time_t now = time(0);
    tm *ltm = localtime(&now);

    ltm->tm_hour += durasiSewa; // Tambahkan durasi sewa ke jam sekarang
    mktime(ltm); // Perbarui waktu jika terjadi overflow (misalnya lebih dari 24 jam)

    char buffer[80];
    strftime(buffer, 80, "%H:%M", ltm); // Format waktu menjadi HH:MM
    return string(buffer);
}
```

"Fungsi hitungWaktuSelesai digunakan untuk menghitung waktu selesai sewa lapangan berdasarkan durasi yang dimasukkan oleh pengguna. Fungsi ini memanfaatkan waktu saat ini dan menambahkan durasi sewa untuk menghasilkan waktu selesai."

1. **Mendapatkan Waktu Saat Ini:** *"Fungsi ini dimulai dengan mendapatkan waktu saat ini menggunakan time(0), yang akan memberikan waktu dalam bentuk epoch time (jumlah detik sejak 1 Januari 1970)."*
2. **Mengkonversi Waktu ke Format Lokal:** *"Kemudian, kita mengkonversi epoch time tersebut menjadi waktu lokal dengan localtime(&now), yang menghasilkan struktur tm yang berisi berbagai komponen waktu seperti tahun, bulan, hari, jam, menit, dan detik."*
3. **Menambahkan Durasi Sewa:** *"Selanjutnya, kita menambahkan durasi sewa (dalam jam) ke jam saat ini dengan ltm->tm_hour += durasiSewa. Ini mengupdate waktu lokal dengan durasi sewa yang dimasukkan oleh pengguna."*
4. **Menyesuaikan Waktu jika Terjadi Overflow:** *"Karena waktu memiliki batasan (misalnya, jam tidak bisa lebih dari 23), kita menggunakan mktime(ltm) untuk menyesuaikan waktu jika terjadi overflow, seperti jika hasil jam melebihi 24 jam. Fungsi ini akan memperbarui waktu secara otomatis."*
5. **Format Waktu ke Bentuk yang Dapat Dibaca:** *"Setelah itu, kita menggunakan strftime(buffer, 80, "%H:%M", ltm) untuk memformat waktu selesai dalam format HH:MM (jam dan menit), yang lebih mudah dibaca oleh pengguna."*
6. **Mengembalikan Waktu Selesai:** *"Akhirnya, fungsi ini mengembalikan waktu selesai dalam bentuk string yang dapat ditampilkan di konsol atau digunakan dalam proses lainnya."*

"Dengan cara ini, kita bisa menghitung waktu selesai sewa secara otomatis dan menampilkannya kepada pengguna agar mereka tahu kapan waktu sewa mereka berakhir."

```
bool loginAdmin() {
    string username, password;
    const string adminUsername = "byte";  // Username admin
    const string adminPassword = "mantap";  // Password admin

    setColor(YELLOW, BLACK);
    cout << "Masukkan username admin: ";
    cin >> username;
    cout << "Masukkan password admin: ";
    cin >> password;

    setColor(WHITE, BLACK);
    return (username == adminUsername && password == adminPassword);
}
```

"Fungsi loginAdmin bertanggung jawab untuk memverifikasi identitas admin yang mencoba mengakses bagian admin dari aplikasi ini. Fungsi ini akan meminta pengguna untuk memasukkan username dan password, kemudian memeriksa apakah keduanya cocok dengan kredensial yang telah ditentukan."

1. **Mendeklarasikan Username dan Password:** *"Di dalam fungsi ini, kita mendeklarasikan dua variabel string, yaitu username dan password, yang akan digunakan untuk menyimpan input dari pengguna. Kami juga telah mendefinisikan dua variabel const string yaitu adminUsername dan adminPassword, yang berisi kredensial default admin: 'byte' untuk username dan 'mantap' untuk password."*
2. **Minta Input Username dan Password:** *"Fungsi ini kemudian meminta pengguna untuk memasukkan username dan password menggunakan cin. Teks yang dimasukkan oleh pengguna akan disimpan dalam variabel username dan password."*
3. **Mengubah Warna Teks:** *"Sebelum menampilkan pesan input, kita mengubah warna teks menggunakan fungsi setColor untuk memberi visualisasi yang lebih menarik dan membedakan bagian login dari bagian lainnya. Di sini, warna teks diubah menjadi kuning dengan latar belakang hitam."*
4. **Verifikasi Kredensial:** *"Setelah menerima input, fungsi ini membandingkan nilai username dan password yang dimasukkan oleh pengguna dengan kredensial yang telah didefinisikan, yaitu adminUsername dan adminPassword."*
5. **Mengembalikan Nilai:** *"Fungsi ini mengembalikan nilai boolean true jika username dan password yang dimasukkan cocok dengan kredensial yang benar. Jika salah satu atau keduanya tidak cocok, fungsi ini akan mengembalikan false."*
6. **Visual Feedback:** *"Setelah proses login selesai, kita mengubah kembali warna teks menjadi putih dengan latar belakang hitam menggunakan setColor untuk memastikan tampilan aplikasi tetap konsisten."*

"Dengan menggunakan fungsi ini, kita dapat memastikan hanya pengguna yang memiliki kredensial yang benar (seperti admin) yang dapat mengakses fitur-fitur tertentu dalam aplikasi, seperti menambah atau menghapus transaksi."

```
// Berikut adalah fungsi untuk menulis log ke file
void writeLog(const string &message) {
    ofstream logFile;
    logFile.open("log futsal.txt", ios::app);  // Membuka file l
    if (logFile.is_open()) {
        logFile << fixed << setprecision(2) << message << endl;
        logFile.close();
    } else {
        setColor(RED, BLACK);
        cout << "Error: Tidak bisa membuka file log.\n";
        setColor(WHITE, BLACK);
    }
}
```

"Fungsi writeLog digunakan untuk mencatat log transaksi atau event penting dalam aplikasi ke dalam file. Hal ini berguna untuk menyimpan riwayat transaksi atau tindakan tertentu yang terjadi dalam sistem."

1. **Membuka File Log:** *"Fungsi ini pertama-tama membuka file log futsal.txt menggunakan ofstream dengan mode ios::app, yang berarti file akan dibuka untuk ditambah (append) tanpa menghapus data yang sudah ada di dalamnya. Ini memungkinkan kita untuk menyimpan riwayat log tanpa kehilangan informasi sebelumnya."*
2. **Menulis Pesan ke File:** *"Jika file berhasil dibuka (artinya file dapat diakses dengan benar), kita menggunakan logFile << fixed << setprecision(2) << message << endl; untuk menulis pesan log ke dalam file. Di sini, fixed dan setprecision(2) memastikan bahwa pesan yang ditulis diformat dengan dua angka desimal, memberikan konsistensi dalam tampilan angka seperti harga atau durasi yang dicatat."*
3. **Menutup File:** *"Setelah pesan log ditulis, kita menutup file log menggunakan logFile.close(). Ini penting untuk memastikan bahwa file tersebut ditutup dengan benar setelah operasi selesai dan untuk menghindari kebocoran sumber daya."*
4. **Menangani Error:** *"Jika file tidak dapat dibuka, kita menampilkan pesan error ke konsol dengan warna teks merah menggunakan fungsi setColor agar pesan error terlihat jelas dan menonjol. Ini membantu pengguna atau administrator untuk mengetahui bahwa ada masalah dengan akses file log."*

"Dengan menggunakan writeLog, kita dapat melacak berbagai aktivitas dalam aplikasi, seperti transaksi yang dilakukan atau tindakan admin, yang berguna untuk analisis atau troubleshooting di kemudian hari."

```
void addTransaksi(Transaksi &transaksi, int id) {
    transaksi.idTransaksi = id;

    int pilihanLapangan;
    setColor(CYAN, BLACK);
    cout << "Pilih Jenis Lapangan:\n";
    cout << "1. Lapangan Standar (Rp 100.000/jam)\n";
    cout << "2. Lapangan VIP (Rp 150.000/jam)\n";
    cout << "Pilih (1/2): ";
    cin >> pilihanLapangan;
```

"Bagian pertama dari fungsi addTransaksi adalah meminta pengguna untuk memilih jenis lapangan yang ingin disewa. Pilihan ini terdiri dari dua opsi: Lapangan Standar dan Lapangan VIP."

1. **Menampilkan Pilihan Jenis Lapangan:** *"Fungsi ini menampilkan dua pilihan jenis lapangan kepada pengguna, yaitu Lapangan Standar dengan harga Rp 100.000 per jam dan Lapangan VIP dengan harga Rp 150.000 per jam. Informasi harga ini jelas ditampilkan untuk membantu pengguna membuat keputusan."*
2. **Memasukkan Pilihan Pengguna:** *"Pengguna diminta untuk memasukkan pilihan mereka, apakah mereka ingin memilih Lapangan Standar (pilihan 1) atau Lapangan VIP (pilihan 2). Pilihan ini disimpan dalam variabel pilihanLapangan."*
3. **Menggunakan Warna Teks untuk Tampilan:** *"Untuk meningkatkan interaksi dengan pengguna, tampilan pilihan lapangan diberi warna cyan untuk teks, yang memberikan kesan menarik dan mudah dibaca. Penggunaan warna ini mengoptimalkan tampilan antarmuka pada konsol."*

"Pada tahap ini, fungsi hanya menerima input pilihan lapangan dan menampilkan opsi yang tersedia. Selanjutnya, validasi pilihan dilakukan dan harga per jam akan ditentukan berdasarkan pilihan pengguna."

```
// memvalidasi pilihan lapangan dan menghitung total harga meng
switch (pilihanLapangan) {
    case 1:
        transaksi.jenisLapangan = "Lapangan Standar";
        transaksi.hargaPerJam = 100000;
        break;
    case 2:
        transaksi.jenisLapangan = "Lapangan VIP";
        transaksi.hargaPerJam = 150000;
        break;
    default:
        setColor(RED, BLACK);
        cout << "Pilihan tidak valid.\n";
        setColor(WHITE, BLACK);
        return;
}
```

"Bagian ini bertanggung jawab untuk memvalidasi pilihan lapangan yang dimasukkan oleh pengguna dan menetapkan harga per jam berdasarkan pilihan tersebut."

1. **Validasi Pilihan Lapangan:** *"Setelah pengguna memasukkan pilihan lapangan (1 atau 2), kode ini menggunakan struktur switch-case untuk memeriksa nilai pilihanLapangan yang dimasukkan oleh pengguna."*
2. **Pilihan Lapangan Standar:** *"Jika pengguna memilih '1' untuk Lapangan Standar, maka program akan menetapkan variabel jenisLapangan dengan nilai 'Lapangan Standar' dan harga per jam (hargaPerJam) akan diset menjadi Rp 100.000."*
3. **Pilihan Lapangan VIP:** *"Jika pengguna memilih '2' untuk Lapangan VIP, maka program akan menetapkan variabel jenisLapangan dengan nilai 'Lapangan VIP' dan harga per jam (hargaPerJam) akan diset menjadi Rp 150.000."*
4. **Menangani Pilihan Tidak Valid:** *"Jika pengguna memasukkan pilihan selain 1 atau 2, maka blok default akan dijalankan. Pada kondisi ini, fungsi akan menampilkan pesan error berwarna merah yang memberi tahu pengguna bahwa pilihan mereka tidak valid." "Setelah menampilkan pesan error, program akan keluar dari fungsi addTransaksi dengan return;, sehingga transaksi tidak akan dilanjutkan jika pilihan lapangan tidak valid."*
5. **Warna Teks untuk Menarik Perhatian:** *"Untuk memastikan pesan error terlihat jelas, kita menggunakan fungsi setColor untuk mengubah warna teks menjadi merah, yang memberikan efek visual yang kuat dan mudah dilihat oleh pengguna."*

"Dengan menggunakan switch-case, kita memastikan bahwa hanya pilihan yang valid yang diterima, sementara pilihan yang tidak valid akan segera ditangani dengan pesan error yang sesuai."

```
// setelah itu, user diminta untuk memasukkan durasi sewa
cout << "Masukkan durasi sewa (dalam jam): ";
cin >> transaksi.durasiSewa;
if (transaksi.durasiSewa <= 0) { // durasi sewa harus lebih dari 1 jam
    setColor(RED, BLACK);
    cout << "Durasi sewa tidak valid.\n";
    setColor(WHITE, BLACK);
    return;
}
```

"Bagian ini meminta pengguna untuk memasukkan durasi sewa lapangan dan memvalidasi input tersebut."

1. **Meminta Durasi Sewa:** *"Setelah memilih jenis lapangan, pengguna diminta untuk memasukkan durasi sewa dalam satuan jam. Input ini akan disimpan dalam variabel durasiSewa pada objek transaksi." "Pesan yang muncul di layar meminta pengguna untuk memasukkan durasi sewa yang diinginkan."*
2. **Validasi Durasi Sewa:** *"Setelah pengguna memasukkan durasi sewa, sistem akan memeriksa apakah durasi tersebut valid. Pada bagian ini, durasi sewa harus lebih besar dari 0 jam, karena durasi sewa yang kurang dari atau sama dengan 0 jam tidak mungkin diterima."*
3. **Menangani Input Tidak Valid:** *"Jika durasi sewa yang dimasukkan kurang dari atau sama dengan 0, maka program akan menampilkan pesan kesalahan dengan teks merah, memberi tahu pengguna bahwa durasi sewa tidak valid." "Fungsi setColor(RED, BLACK) digunakan untuk mengganti warna teks menjadi merah, memberikan efek visual yang jelas pada kesalahan yang terjadi." "Setelah itu, program akan keluar dari fungsi addTransaksi dengan perintah return;, sehingga transaksi tidak akan dilanjutkan jika durasi sewa tidak valid."*
4. **Warna Teks untuk Error:** *"Seperti pada bagian sebelumnya, kita menggunakan warna merah untuk pesan kesalahan, sehingga kesalahan tersebut dapat langsung dilihat oleh pengguna."*

"Dengan validasi ini, kita memastikan bahwa durasi sewa yang dimasukkan sesuai dengan aturan yang ada dan bahwa transaksi hanya dilanjutkan jika input durasi sewa valid."

```
// setelah itu, total harga dihitung berdasarkan harga per jam dan durasi sewa
transaksi.totalHarga = transaksi.hargaPerJam * transaksi.durasiSewa;
transaksi.waktuSelesai = hitungWaktuSelesai(transaksi.durasiSewa); // Hitung wa
daftarTransaksi.push_back(transaksi);  // menambahkan transaksi ke daftar trans
```

"Bagian ini bertanggung jawab untuk menghitung total harga transaksi, menentukan waktu selesai, menambahkan transaksi ke dalam daftar transaksi, dan mencatat log transaksi baru."

1. **Menghitung Total Harga:** *"Setelah durasi sewa dan harga per jam ditentukan, total harga dihitung dengan cara mengalikan hargaPerJam dengan durasiSewa. Hasilnya disimpan dalam variabel totalHarga pada objek transaksi." "Total harga ini mencerminkan jumlah uang yang harus dibayar pelanggan untuk menyewa lapangan selama durasi yang telah ditentukan."*
2. **Menghitung Waktu Selesai:** *"Setelah itu, fungsi hitungWaktuSelesai dipanggil untuk menentukan waktu selesai dari transaksi. Fungsi ini menerima durasi sewa (dalam jam) sebagai input dan menghitung waktu selesai berdasarkan jam saat ini." "Waktu selesai ini dihitung dengan menambahkan durasi sewa ke jam saat ini, dan hasilnya disimpan dalam variabel waktuSelesai."*
3. **Menambahkan Transaksi ke Daftar Transaksi:** *"Selanjutnya, transaksi yang baru saja dihitung ditambahkan ke dalam daftar transaksi yang ada menggunakan fungsi push_back() pada vektor daftarTransaksi. Ini memastikan bahwa transaksi baru disimpan dan dapat diakses untuk keperluan selanjutnya."*
4. **Mencatat Log Transaksi Baru:** *"Setelah transaksi berhasil dihitung dan ditambahkan, informasi transaksi ini dicatat dalam file log menggunakan fungsi writeLog." "Pesan log ini berisi detail transaksi seperti ID transaksi, jenis lapangan, durasi sewa, total harga, dan waktu selesai. Log ini ditulis dalam format yang mudah dibaca dan memiliki presisi dua angka desimal."*
5. **Pesan Konfirmasi Transaksi Berhasil:** *"Setelah transaksi berhasil ditambahkan dan log tercatat, program akan menampilkan pesan konfirmasi dengan teks berwarna hijau yang memberi tahu pengguna bahwa transaksi telah berhasil ditambahkan." "Pesan ini memastikan pengguna mendapatkan feedback langsung setelah mereka berhasil menambahkan transaksi baru."*

"Dengan langkah-langkah ini, program memastikan bahwa setiap transaksi dihitung dengan benar, disimpan dengan rapi, dan dicatat dalam file log untuk referensi di masa depan."

```
// setelah itu, total harga dihitung berdasarkan harga per jam dan durasi sewa
transaksi.totalHarga = transaksi.hargaPerJam * transaksi.durasiSewa;
transaksi.waktuSelesai = hitungWaktuSelesai(transaksi.durasiSewa); // Hitung w
daftarTransaksi.push_back(transaksi);  // menambahkan transaksi ke daftar tra

// menuliskan log transaksi baru ke log file
string logMessage = "Transaksi Baru: ID "
                  + to_string(transaksi.idTransaksi)
                  + ", Lapangan: " + transaksi.jenisLapangan
                  + ", Durasi: " + to_string(transaksi.durasiSewa)
                  + " jam, Total: Rp " + to_string(transaksi.totalHarga)
                  + ", Waktu Selesai: " + transaksi.waktuSelesai;
writeLog(logMessage);

setColor(GREEN, BLACK);
cout << "Transaksi berhasil ditambahkan!\n";
setColor(WHITE, BLACK);
```

"Fungsi ini bertugas untuk menampilkan struk pembayaran yang mencantumkan detail transaksi yang telah dilakukan."

1. **Pengaturan Warna Teks:** *"Fungsi setColor(LIGHTBLUE, BLACK) digunakan untuk mengubah warna teks menjadi biru terang dengan latar belakang hitam. Hal ini bertujuan untuk memberi penekanan pada bagian struk pembayaran agar mudah dibaca oleh pengguna."*
2. **Menampilkan Header Struk Pembayaran:** *"Di bagian awal, ditampilkan header struk yang berisi judul 'STRUK PEMBAYARAN FUTSAL BYTE' dalam format yang menarik. Hal ini memberikan konteks kepada pengguna bahwa ini adalah struk transaksi futsal."*
3. **Menampilkan Detail Transaksi:** *"Selanjutnya, informasi detail transaksi ditampilkan satu per satu:*
    - **ID Transaksi:** ID unik yang menunjukkan transaksi tersebut.
    - **Jenis Lapangan:** Lapangan yang disewa, apakah lapangan standar atau VIP.
    - **Durasi Sewa:** Lama waktu penyewaan dalam jam.
    - **Harga Per Jam:** Harga sewa per jam untuk lapangan yang dipilih.
    - **Total Harga:** Total biaya yang harus dibayar oleh pelanggan.
    - **Waktu Selesai:** Waktu yang diperkirakan untuk selesai berdasarkan durasi sewa."*
4. **Format Uang dengan Presisi:** *"Harga per jam dan total harga ditampilkan menggunakan format uang dengan 3 angka desimal. Fungsi setprecision(3) digunakan untuk memastikan angka harga ditampilkan dengan presisi yang tepat."*
5. **Footer Struk:** *"Bagian akhir struk menampilkan garis pemisah untuk memberikan kesan bahwa struk ini telah selesai dan mudah untuk dipahami."*
6. **Pengaturan Kembali Warna Teks:** *"Setelah struk ditampilkan, warna teks dikembalikan ke warna semula menggunakan setColor(WHITE, BLACK)."*

"Dengan cara ini, pengguna akan mendapatkan struk yang jelas dan mudah dipahami mengenai rincian biaya sewa lapangan futsal, serta informasi terkait transaksi mereka."

```cpp
// Berikut adalah fungsi untuk menampilkan semua transaksi
void printStruk(const Transaksi &transaksi) {
    setColor(LIGHTBLUE, BLACK);
    cout << "\n\n======== STRUK PEMBAYARAN FUTSAL BYTE ========\n";
    cout << "ID Transaksi: " << transaksi.idTransaksi << endl;
    cout << "Jenis Lapangan: " << transaksi.jenisLapangan << endl;
    cout << "Durasi Sewa: " << transaksi.durasiSewa << " jam\n";
    cout << "Harga Per Jam: Rp " << fixed << setprecision(3) << transaksi.hargaPerJam << endl
    cout << "Total Harga: Rp " << fixed << setprecision(3) << transaksi.totalHarga << endl;
    cout << "Waktu Selesai: " << transaksi.waktuSelesai << endl;
    cout << "==============================================\n";
    setColor(WHITE, BLACK);
}
```

Untuk bagian fungsi tampilkanDaftarTransaksi(), berikut adalah penjelasan yang dapat digunakan untuk presentasi:

"Fungsi ini digunakan untuk menampilkan daftar semua transaksi yang telah dilakukan."

1. **Memeriksa Apakah Daftar Transaksi Kosong:** ***"Fungsi dimulai dengan memeriksa apakah daftarTransaksi kosong. Jika kosong, maka pesan 'Belum ada transaksi' akan ditampilkan dengan warna merah, menggunakan fungsi setColor(RED, BLACK)." "Setelah itu, program kembali ke tampilan awal menggunakan setColor(WHITE, BLACK) dan fungsi langsung keluar (return) jika tidak ada transaksi."***
2. **Menampilkan Daftar Transaksi:** ***"Jika ada transaksi yang tercatat, daftar transaksi akan ditampilkan satu per satu dalam format yang jelas. Di sini, warna teks diubah menjadi kuning dengan setColor(YELLOW, BLACK) untuk menarik perhatian pada daftar transaksi."***
3. **Rincian Tiap Transaksi:** ***"Untuk setiap transaksi, informasi yang ditampilkan meliputi:***
    - **ID Transaksi: ID unik dari transaksi tersebut.**
    - **Jenis Lapangan: Jenis lapangan yang disewa (Standar atau VIP).**
    - **Durasi Sewa: Durasi waktu penyewaan dalam jam.**
    - **Total Harga: Total biaya yang harus dibayar oleh pelanggan (dengan dua angka desimal).**
    - **Waktu Selesai: Waktu yang diperkirakan untuk selesai berdasarkan durasi yang dimasukkan."***
4. **Format Tampilan yang Rapi:** ***"Setiap transaksi ditampilkan dalam format yang rapi, dengan pemisah antar informasi menggunakan baris teks. setprecision(2) digunakan untuk menampilkan total harga dengan dua angka desimal, memberikan format uang yang tepat."***
5. **Footer Daftar Transaksi:** ***"Setelah semua transaksi ditampilkan, bagian akhir daftar akan diberi garis pemisah sebagai penutup, memberikan struktur yang jelas pada tampilan."***
6. **Mengembalikan Warna Teks ke Standar:** ***"Setelah menampilkan daftar transaksi, warna teks kembali ke semula dengan setColor(WHITE, BLACK)."***

"Dengan menggunakan fungsi ini, admin atau pengguna dapat melihat semua transaksi yang telah tercatat dengan informasi yang lengkap dan jelas mengenai jenis lapangan, durasi, harga, dan waktu selesai."

```cpp
// Berikut adalah fungsi untuk menampilkan semua transaksi
void tampilkanDaftarTransaksi() {
    if (daftarTransaksi.empty()) {
        setColor(RED, BLACK);
        cout << "Belum ada transaksi.\n";
        setColor(WHITE, BLACK);
        return;
    }

    setColor(YELLOW, BLACK);
    cout << "\n===== Daftar Transaksi =====\n";
    for (const auto &transaksi : daftarTransaksi) {
        cout << "ID Transaksi: " << transaksi.idTransaksi << "\n";
        cout << "Lapangan: " << transaksi.jenisLapangan << ", Durasi: "
             << transaksi.durasiSewa << " jam, Total Harga: Rp "
             << fixed << setprecision(2) << transaksi.totalHarga << ", Waktu Selesai: "
             << transaksi.waktuSelesai << endl;
    }
    cout << "============================\n";
    setColor(WHITE, BLACK);
}
```

"Fungsi ini digunakan untuk menampilkan daftar semua transaksi yang telah dilakukan."

1. **Memeriksa Apakah Daftar Transaksi Kosong:** *"Fungsi dimulai dengan memeriksa apakah daftarTransaksi kosong. Jika kosong, maka pesan 'Belum ada transaksi' akan ditampilkan dengan warna merah, menggunakan fungsi setColor(RED, BLACK)." "Setelah itu, program kembali ke tampilan awal menggunakan setColor(WHITE, BLACK) dan fungsi langsung keluar (return) jika tidak ada transaksi."*
2. **Menampilkan Daftar Transaksi:** *"Jika ada transaksi yang tercatat, daftar transaksi akan ditampilkan satu per satu dalam format yang jelas. Di sini, warna teks diubah menjadi kuning dengan setColor(YELLOW, BLACK) untuk menarik perhatian pada daftar transaksi."*
3. **Rincian Tiap Transaksi:** *"Untuk setiap transaksi, informasi yang ditampilkan meliputi:*
    - **ID Transaksi:** ID unik dari transaksi tersebut.
    - **Jenis Lapangan:** Jenis lapangan yang disewa (Standar atau VIP).
    - **Durasi Sewa:** Durasi waktu penyewaan dalam jam.
    - **Total Harga:** Total biaya yang harus dibayar oleh pelanggan (dengan dua angka desimal).
    - **Waktu Selesai:** Waktu yang diperkirakan untuk selesai berdasarkan durasi yang dimasukkan."*
4. **Format Tampilan yang Rapi:** *"Setiap transaksi ditampilkan dalam format yang rapi, dengan pemisah antar informasi menggunakan baris teks. setprecision(2) digunakan untuk menampilkan total harga dengan dua angka desimal, memberikan format uang yang tepat."*
5. **Footer Daftar Transaksi:** *"Setelah semua transaksi ditampilkan, bagian akhir daftar akan diberi garis pemisah sebagai penutup, memberikan struktur yang jelas pada tampilan."*
6. **Mengembalikan Warna Teks ke Standar:** *"Setelah menampilkan daftar transaksi, warna teks kembali ke semula dengan setColor(WHITE, BLACK)."*

"Dengan menggunakan fungsi ini, admin atau pengguna dapat melihat semua transaksi yang telah tercatat dengan informasi yang lengkap dan jelas mengenai jenis lapangan, durasi, harga, dan waktu selesai."

```
void hapusTransaksi(int id) {
    bool ditemukan = false;
    for (auto it = daftarTransaksi.begin(); it != daftarTransaksi.end(); ++it) {
        if (it->idTransaksi == id) {
            daftarTransaksi.erase(it);
            setColor(GREEN, BLACK);
            cout << "Transaksi dengan ID " << id << " telah dihapus.\n";
            setColor(WHITE, BLACK);

            // menulis log penghapusan transaksi
            string logMessage = "Transaksi Dihapus: ID " + to_string(id);
            writeLog(logMessage);

            ditemukan = true;
            break;
        }
    }

    if (!ditemukan) {
        setColor(RED, BLACK);
        cout << "Transaksi dengan ID " << id << " tidak ditemukan.\n";
        setColor(WHITE, BLACK);
    }
```

"Fungsi hapusTransaksi() digunakan untuk menghapus transaksi yang tercatat dalam daftar berdasarkan ID yang diberikan."

1. Inisialisasi Variabel:

- **bool ditemukan = false;**
  "Variabel ditemukan digunakan untuk menandakan apakah transaksi dengan ID yang dimasukkan ditemukan dalam daftar transaksi. Awalnya diatur ke false, menandakan bahwa transaksi tersebut belum ditemukan."

2. Mencari Transaksi Berdasarkan ID:

- Fungsi menggunakan loop for untuk iterasi melalui semua transaksi dalam daftarTransaksi yang disimpan dalam vector.
- **auto it = daftarTransaksi.begin();**
  "Iterator it digunakan untuk berjalan melalui setiap elemen dalam vector daftarTransaksi."

3. Memeriksa Kesesuaian ID:

- **if (it->idTransaksi == id)**
  "Di dalam loop, fungsi memeriksa apakah ID transaksi yang sedang diperiksa sama dengan ID yang diberikan oleh pengguna. Jika ditemukan, maka transaksi akan dihapus."

4. Menghapus Transaksi:

- **daftarTransaksi.erase(it);**
  "Jika ID transaksi cocok, fungsi menggunakan erase untuk menghapus transaksi dari daftar berdasarkan iterator it."
- Setelah transaksi dihapus, pesan konfirmasi ditampilkan dengan warna hijau menggunakan setColor(GREEN, BLACK), memberitahukan bahwa transaksi berhasil dihapus.
- **Log Penghapusan:**
  "Setelah transaksi dihapus, pesan log penghapusan dicatat dengan menggunakan writeLog(), yang mencatat ID transaksi yang telah dihapus."

5. Menandai Transaksi Ditemukan:

- Setelah penghapusan, **ditemukan** diubah menjadi true, menandakan bahwa transaksi dengan ID yang diberikan telah berhasil ditemukan dan dihapus.
- Fungsi kemudian keluar dari loop dengan **break;** untuk menghentikan pencarian lebih lanjut.

6. Menangani Kasus Jika Transaksi Tidak Ditemukan:

- Jika loop selesai tanpa menemukan ID yang cocok (variabel ditemukan masih false), maka pesan kesalahan ditampilkan dengan warna merah menggunakan setColor(RED, BLACK).
- Pesan tersebut memberitahukan bahwa transaksi dengan ID yang diberikan tidak ditemukan dalam daftar transaksi.

7. Mengembalikan Warna Teks:

- Setelah menampilkan pesan, warna teks kembali ke putih dengan latar belakang hitam menggunakan setColor(WHITE, BLACK).

Ringkasan: Fungsi hapusTransaksi() bekerja dengan mencari transaksi berdasarkan ID yang diberikan. Jika ditemukan, transaksi tersebut dihapus dari daftar dan dicatat dalam log. Jika tidak ditemukan, fungsi akan menampilkan pesan kesalahan.

Dengan cara ini, admin dapat mengelola daftar transaksi secara lebih efisien dan mudah menghapus transaksi yang sudah tidak diperlukan.

```
int main(){
    int idTransaksi = 1;  // ID transaksi pertama
    Transaksi transaksi;
    int choice;

    // Menampilkan menu untuk admin
    setColor(WHITE, BLUE); // Teks putih, background biru
    cout << "=======================================\n";
    cout << "        APLIKASI KASIR FUTSAL BYTE     \n";
    cout << "=======================================\n\n";
    setColor(WHITE, BLACK);

    if (!loginAdmin()) {
        setColor(RED, BLACK);
        cout << "Login gagal. Keluar dari program.\n";
        return 0;
    }
```

Berikut adalah penjelasan untuk fungsi **main()** yang bisa digunakan dalam presentasi:

1. Inisialisasi Program:

- **int idTransaksi = 1;**
  "ID transaksi pertama dimulai dari 1, yang akan digunakan untuk memberikan ID unik pada setiap transaksi yang baru."
- **Transaksi transaksi;**
  "Deklarasi objek transaksi untuk menyimpan data transaksi yang akan diproses."

2. Menampilkan Menu Awal Aplikasi:

- Program mulai dengan menampilkan judul aplikasi menggunakan **setColor()** untuk memberikan warna teks dan latar belakang yang sesuai.
- **cout** menampilkan judul aplikasi *"APLIKASI KASIR FUTSAL BYTE"* untuk memberi tahu pengguna bahwa mereka sedang menggunakan aplikasi kasir futsal.

3. Proses Login Admin:

- **if (!loginAdmin())**
  "Setelah menampilkan menu utama, program memanggil fungsi loginAdmin() untuk meminta username dan password dari admin."
- Jika login gagal, **setColor(RED, BLACK)** digunakan untuk menampilkan pesan kesalahan berwarna merah dan program keluar menggunakan **return 0;**.
- **Jika login berhasil, admin dapat mengakses menu utama aplikasi.**

4. Menu Admin:

- Setelah berhasil login, program menampilkan menu utama dengan pilihan sebagai berikut:
  1. **Lihat Daftar Transaksi**
  2. **Hapus Transaksi**

3. **Tambah Transaksi**
4. **Keluar**

- Admin diminta untuk memilih salah satu menu dengan memasukkan angka sesuai pilihan.

5. Switch Case untuk Menangani Pilihan Menu:

- Program menggunakan **switch** untuk menanggapi pilihan admin:
  - **Case 1: Lihat Daftar Transaksi**
    - Fungsi **tampilkanDaftarTransaksi()** dipanggil untuk menampilkan semua transaksi yang ada.
  - **Case 2: Hapus Transaksi**
    - Admin diminta untuk memasukkan ID transaksi yang ingin dihapus.
    - Fungsi **hapusTransaksi(idHapus)** dipanggil untuk menghapus transaksi yang dimaksud.
  - **Case 3: Tambah Transaksi**
    - Fungsi **addTransaksi()** digunakan untuk menambahkan transaksi baru.
    - Setelah transaksi berhasil ditambahkan, **printStruk()** menampilkan struk pembayaran transaksi yang baru.
    - **ID transaksi** diperbarui (incremented) untuk transaksi berikutnya.
  - **Case 4: Keluar**
    - Jika admin memilih opsi untuk keluar, program akan menampilkan pesan *"Keluar dari program."* dan keluar dari loop.

6. Loop Program:

- Menu akan terus ditampilkan dalam **do-while loop** selama pilihan yang dimasukkan oleh admin bukan nomor 4 (Keluar).
- Ketika admin memilih untuk keluar (choice = 4), program akan berhenti.

Ringkasan:
Fungsi main() adalah pusat kontrol dari aplikasi kasir futsal ini. Setelah login berhasil, admin diberikan pilihan untuk melihat daftar transaksi, menghapus transaksi, menambah transaksi baru, atau keluar. Menu ini akan terus berulang hingga admin memilih untuk keluar dari program.