Code Documentation
Sistem Manajemen Perpustakaan

- **Library yang digunakan:**
    1. #include <iostream>
    2. #include <string>
    3. #include <unordered_map>
    4. #include <vector>
    5. #include <algorithm>
    6. #include <fstream>
    7. #include <cstdlib>
- **Header tambahan untuk fungsionalitas OS:**
    1. Windows: <windows.h> dan <conio.h>
    2. Unix-like: <unistd.h> dan <termios.h>
- **Fungsi Khusus OS**
    1. Delay Function

```
#ifdef _WIN32 // Memeriksa apakah sistem operasi yang digunakan adalah Windows
#include <windows.h>
void delay(int milliseconds) { // Fungsi untuk memberikan delay dalam milidetik
    Sleep(milliseconds);
}
#else
#include <unistd.h>
void delay(int milliseconds) {
    usleep(milliseconds * 1000);
}
#endif
```

Fungsi Delay digunakan untuk memberi delay pada program sebelum melanjutkan ke perintah selanjutnya.

2. Clear Screen Function

```
#ifdef _WIN32 // Memeriksa apakah sistem operasi yang digunakan adalah Windows
#include <cstdlib>
void clearScreen() { // Fungsi untuk membersihkan layar konsol
    system("cls");
}
#else
#include <cstdio>
void clearScreen() {
    system("clear");
}
#endif
```

Fungsi Clear Screen digunakan untuk membersihkan layer terminal atau cmd user.

3. Getch (Get Character) Function

```
#ifdef _WIN32 // Memeriksa apakah sistem operasi yang digunakan adalah Windows
#include <conio.h>
#else
#include <termios.h>
#include <unistd.h>

char getch() {
    char buf = 0;
    struct termios old;
    fflush(stdout);
    if (tcgetattr(0, &old) < 0)
        perror("tcsetattr()");
    old.c_lflag &= ~ICANON;
    old.c_lflag &= ~ECHO;
    old.c_cc[VMIN] = 1;
    old.c_cc[VTIME] = 0;
    if (tcsetattr(0, TCSANOW, &old) < 0)
        perror("tcsetattr ICANON");
    if (read(0, &buf, 1) < 0)
        perror("read()");
    old.c_lflag |= ICANON;
    old.c_lflag |= ECHO;
    if (tcsetattr(0, TCSADRAIN, &old) < 0)
        perror("tcsetattr ~ICANON");
    return buf;
}
#endif
```

Fungsi getch digunakan untuk membaca satu karakter dari konsol tanpa mengharuskan pengguna menekan Enter.

- **Struktur data yang digunakan:**

```
// Struktur untuk menyimpan informasi buku
struct Book {
    int id;
    string title;
    int volume;
    string author;
    string publisher;
    string category;
    int year;
};

// Struktur untuk menyimpan informasi peminjaman buku
struct BookLoan {
    int bookId;
    string borrowerName;
    int durationDays;
};
```

- **Konstanta dan variable global yang digunakan:**

```
const int MAX_LOANS = 100; // Jumlah maksimum peminjaman buku yang dapat disimpan
BookLoan loans[MAX_LOANS]; // Array untuk menyimpan informasi peminjaman buku
int loanCount = 0; // Jumlah peminjaman buku saat ini
const int MAX_BOOKS = 100; // Jumlah maksimum buku yang dapat disimpan
const int BOOKS_PER_PAGE = 10; // Jumlah maksimum buku yang ditampilkan per halaman
const int LOANS_PER_PAGE = 5; // Jumlah maksimum peminjaman yang ditampilkan per halaman
```

1. MAX_LOANS: Jumlah maksimum pinjaman buku yang dapat disimpan.
2. BookLoan loans: Sebuah array untuk menyimpan informasi peminjaman buku.
3. loanCount: Jumlah peminjaman buku saat ini.
4. MAX_BOOKS: Jumlah maksimum buku yang dapat disimpan.
5. BOOKS_PER_PAGE: Jumlah maksimum buku yang ditampilkan per halaman.
6. LOANS_PER_PAGE: Jumlah maksimum pinjaman buku yang ditampilkan per halaman.

- **Fungsi Hash**

```
// Fungsi hash untuk mengonversi string menjadi nilai integer
size_t hashString(const string& str) {
    size_t hash = 0;
    for (char c : str) {
        hash = hash * 31 + c;
    }
    return hash;
}
```

Fungsi ini digunakan untuk mengubah string menjadi nilai numerik, yang dapat digunakan untuk penyimpanan dan pencarian yang efisien dalam Hash Table.

- **Fungsi `inputBook`**

```
// Fungsi untuk menginput informasi buku dan memperbarui hash table
void inputBook(Book books[], int& bookCount, unordered_map<string, vector<int>>& titleIndex, unordered_map<int, int>& idIndex) {
    Book newBook;
    newBook.id = bookCount + 1; // Mengatur ID buku sesuai dengan bookCount
    cout << "Masukkan judul buku: ";
    getline(cin, newBook.title);
    cout << "Masukkan volume buku: ";
    cin >> newBook.volume;
    cin.ignore();
    cout << "Masukkan nama pengarang: ";
    getline(cin, newBook.author);
    cout << "Masukkan nama publisher: ";
    getline(cin, newBook.publisher);
    cout << "Masukkan kategori buku: ";
    getline(cin, newBook.category);
    cout << "Masukkan tahun terbit: ";
    cin >> newBook.year;
    cin.ignore();

    if (bookCount < MAX_BOOKS) {
        books[bookCount] = newBook;
        titleIndex[newBook.title].push_back(bookCount);
        idIndex[newBook.id] = bookCount; // Menyimpan indeks buku berdasarkan ID
        bookCount++;
        clearScreen();
        cout << "\nBuku dengan judul \"" << newBook.title << " Volume " << newBook.volume << "\" berhasil ditambahkan ke dalam database.\n";
        delay(4000);
        clearScreen();
    } else {
        cout << "Kapasitas penyimpanan buku sudah penuh.\n";
        delay(3000);
        clearScreen();
    }
}
```

Fungsi ini memungkinkan pengguna untuk memasukkan informasi buku baru dan memperbarui array buku dan tabel hash yang sesuai. Fungsi ini juga mengecek apakah jumlah buku sudah melebihi dari 100 atau tidak.

- **Fungsi `displayBook`**

```
// Fungsi untuk menampilkan informasi buku
void displayBook(Book book) {
    cout << "\nInformasi Buku:\n";
    cout << "ID Buku: " << book.id << endl;
    cout << "Judul: " << book.title << endl;
    cout << "Volume: " << book.volume << endl;
    cout << "Pengarang: " << book.author << endl;
    cout << "Publisher: " << book.publisher << endl;
    cout << "Kategori: " << book.category << endl;
    cout << "Tahun Terbit: " << book.year << endl;

    // Memeriksa status peminjaman buku
    bool isOnLoan = false;
    for (int i = 0; i < loanCount; i++) {
        if (loans[i].bookId == book.id) {
            isOnLoan = true;
            cout << "Status: Sedang Dipinjam oleh " << loans[i].borrowerName << " (" << loans[i].durationDays << " hari)\n";
            break;
        }
    }

    if (!isOnLoan) {
        cout << "Status: Tersedia\n";
    }
}
```

Fungsi ini digunakan menampilkan informasi dari satu buku saja. Dan juga fungsi ini digunakan oleh fungsi fungsi lain juga seperti `displayBookList` dan `searchBooks`.

- **Fungsi `displayBookList`**

```
// Fungsi untuk menampilkan daftar buku
void displayBookList(Book books[], int bookCount, int& currentPage) {
    int startIndex = (currentPage - 1) * BOOKS_PER_PAGE;
    int endIndex = min(startIndex + BOOKS_PER_PAGE, bookCount);

    cout << "\n\n==========================================\n";
    cout << "Daftar Buku (Halaman " << currentPage << "):\n";
    for (int i = startIndex; i < endIndex; i++) {
        cout << "\n" << (i + 1) << ". ";
        displayBook(books[i]);
    }

    char choice;
    do {
        cout << "\n\n==========================================\n";
        cout << "Tekan 'n' untuk halaman selanjutnya, 'p' untuk halaman sebelumnya, atau 'q' untuk keluar: ";
        choice = getch();

        if (choice == 'n') {
            if (endIndex == bookCount) {
                clearScreen();
                cout << "\nAnda sudah berada di halaman terakhir.\n";
            } else {
                clearScreen();
                currentPage++;
                return displayBookList(books, bookCount, currentPage);
            }
        } else if (choice == 'p') {
            if (currentPage > 1) {
                clearScreen();
                currentPage--;
                return displayBookList(books, bookCount, currentPage);
            } else {
                clearScreen();
                cout << "\nAnda sudah berada di halaman pertama.\n";
            }
        } else if (choice != 'q') {
            cout << "Pilihan tidak valid. Silakan coba lagi.\n";
        }
    } while (choice != 'q');
    clearScreen();
}
```

Fungsi ini digunakan untuk menampilkan daftar buku dengan maksimal 10 buku setiap halaman, jika ada lebih dari 10 buku maka user harus menekan tombol *n* untuk next dan *p* untuk previous dan *q* untuk keluar dari menu. Lalu jika user sudah mencapai halaman terakhir atau awal halaman maka program akan memberi peringatan.

- **Fungsi `searchBooks`**

```
// Fungsi untuk mencari buku berdasarkan judul (tidak case-sensitive)
void searchBooks(Book books[], unordered_map<string, vector<int>>& titleIndex) {
    string searchString;
    cout << "Masukkan judul buku: ";
    getline(cin, searchString);

    // Mengonversi searchString menjadi huruf kecil
    transform(searchString.begin(), searchString.end(), searchString.begin(), ::tolower);

    vector<int> foundIndices;
    for (const auto& pair : titleIndex) {
        string lowercaseTitle = pair.first;
        transform(lowercaseTitle.begin(), lowercaseTitle.end(), lowercaseTitle.begin(), ::tolower);

        if (lowercaseTitle.find(searchString) != string::npos) {
            for (int index : pair.second) {
                foundIndices.push_back(index);
            }
        }
    }

    if (foundIndices.empty()) {
        clearScreen();
        cout << "Tidak ada buku yang ditemukan dengan judul tersebut.\n";
        delay(3000);
        clearScreen();
    } else {
        int currentPage = 1;
        int totalPages = (foundIndices.size() + BOOKS_PER_PAGE - 1) / BOOKS_PER_PAGE;

        char choice;
        do {
            int startIndex = (currentPage - 1) * BOOKS_PER_PAGE;
            int endIndex = min(startIndex + BOOKS_PER_PAGE, static_cast<int>(foundIndices.size()));

            cout << "\nHasil Pencarian (Halaman " << currentPage << " dari " << totalPages << "):\n";
            for (int i = startIndex; i < endIndex; i++) {
                cout << "\n" << (i + 1) << ". ";
                displayBook(books[foundIndices[i]]);
            }
```

Fungsi ini digunakan untuk mencari buku berdasarkan judulnya dan menampilkan hasil pencariannya. Fungsi ini mengambil input string yang diberi oleh user dan mengubahnya menjadi huruf kecil semua dan mencari judul tersebut di `titleIndex` unordered map, jika buku dengan judul yang sama ditemukan maka daftar buku ditampilkan, namun jika tidak ada buku yang memiliki judul yang sama maka program akan mengeluarkan peringatan jika buku dengan judul tersebut tidak ada.

- **Fungsi `saveBookData`**

```cpp
// Fungsi untuk menyimpan data buku ke dalam file CSV
void saveBookData(Book books[], int bookCount) {
    ofstream outputFile("book_data.csv"); // Membuka file dalam mode tulis

    if (outputFile.is_open()) {
        // Menulis header file
        outputFile << "ID,Judul,Volume,Pengarang,Publisher,Kategori,Tahun\n";

        // Menulis data setiap buku ke dalam file
        for (int i = 0; i < bookCount; i++) {
            outputFile << books[i].id << "," << books[i].title << "," << books[i].volume << "," << books[i].author << "," << books[i].publisher << "," << books[i].category << "," << books[i].year << "\n";
        }

        outputFile.close(); // Menutup file
        cout << "Data buku berhasil disimpan ke dalam file book_data.csv.\n";
    } else {
        cout << "Gagal membuka file book_data.csv untuk menyimpan data buku.\n";
    }
}
```

Fungsi ini digunakan untuk menyimpan data buku ke dalam *.csv* file.

- **Fungsi `saveLoanData`**

```cpp
// Fungsi untuk menyimpan data peminjaman ke dalam file CSV
void saveLoanData() {
    ofstream outputFile("loan_data.csv"); // Membuka file dalam mode tulis

    if (outputFile.is_open()) {
        // Menulis header file
        outputFile << "ID Buku,Nama Peminjam,Durasi (hari)\n";

        // Menulis data setiap peminjaman ke dalam file
        for (int i = 0; i < loanCount; i++) {
            outputFile << loans[i].bookId << "," << loans[i].borrowerName << "," << loans[i].durationDays << "\n";
        }

        outputFile.close(); // Menutup file
        cout << "Data peminjaman berhasil disimpan ke dalam file loan_data.csv.\n";
    } else {
        cout << "Gagal membuka file loan_data.csv untuk menyimpan data peminjaman.\n";
    }
}
```

Fungsi ini digunakan untuk menyimpan data peminjam buku ke dalam *.csv* file.

- **Fungsi `loadBookData`**

```cpp
// Fungsi untuk membaca data buku dari file CSV
void loadBookData(Book books[], int& bookCount, unordered_map<string, vector<int>>& titleIndex, unordered_map<int, int>& idIndex) {
    ifstream inputFile("book_data.csv"); // Membuka file dalam mode baca

    if (inputFile.is_open()) {
        string line;
        getline(inputFile, line); // Mengabaikan baris header

        bookCount = 0;
        while (getline(inputFile, line)) {
            if (bookCount >= MAX_BOOKS) {
                cout << "Kapasitas penyimpanan buku sudah penuh. Data tidak dapat dimuat seluruhnya.\n";
                break;
            }

            // Memisahkan setiap nilai dalam baris menggunakan koma sebagai pemisah
            size_t pos = 0;
            string token;
            vector<string> values;
            while ((pos = line.find(',')) != string::npos) {
                token = line.substr(0, pos);
                values.push_back(token);
                line.erase(0, pos + 1);
            }
            values.push_back(line); // Menambahkan nilai terakhir

            // Mengisi data buku ke dalam array
            books[bookCount].id = stoi(values[0]); // Mengisi nilai id dari file
            books[bookCount].title = values[1];
            books[bookCount].volume = stoi(values[2]);
            books[bookCount].author = values[3];
            books[bookCount].publisher = values[4];
            books[bookCount].category = values[5];
            books[bookCount].year = stoi(values[6]);

            // Memperbarui hash table
            titleIndex[books[bookCount].title].push_back(bookCount);
            idIndex[books[bookCount].id] = bookCount;

            bookCount++;
        }
```

Fungsi ini digunakan untuk memuat data buku dari file *.csv* ke dalam program.

- **Fungsi `loadLoanData`**

```cpp
// Fungsi untuk membaca data peminjaman dari file CSV
void loadLoanData(unordered_map<int, int>& idIndex) {
    ifstream inputFile("loan_data.csv"); // Membuka file dalam mode baca

    if (inputFile.is_open()) {
        string line;
        getline(inputFile, line); // Mengabaikan baris header

        loanCount = 0;
        while (getline(inputFile, line)) {
            if (loanCount >= MAX_LOANS) {
                cout << "Kapasitas penyimpanan peminjaman sudah penuh. Data tidak dapat dimuat seluruhnya.\n";
                break;
            }

            // Memisahkan setiap nilai dalam baris menggunakan koma sebagai pemisah
            size_t pos = 0;
            string token;
            vector<string> values;
            while ((pos = line.find(',')) != string::npos) {
                token = line.substr(0, pos);
                values.push_back(token);
                line.erase(0, pos + 1);
            }
            values.push_back(line); // Menambahkan nilai terakhir

            int bookId = stoi(values[0]);
            string borrowerName = values[1];
            int durationDays = stoi(values[2]);

            // Memeriksa apakah buku dengan ID tersebut ada dalam database
            if (idIndex.count(bookId) > 0) {
                BookLoan newLoan = { bookId, borrowerName, durationDays };
                loans[loanCount++] = newLoan;
            } else {
                cout << "Buku dengan ID " << bookId << " tidak ditemukan dalam database. Data peminjaman tidak dapat dimuat.\n";
            }
        }

        inputFile.close(); // Menutup file
        cout << "Data peminjaman berhasil dimuat dari file loan_data.csv.\n";
    } else {
        cout << "File loan_data.csv tidak ditemukan. Data peminjaman tidak dapat dimuat.\n";
    }
}
```

Fungsi ini digunakan untuk memuat data peminjam buku dari file *.csv* ke dalam program.

- **Fungsi `insertionSortLoans`**

```cpp
// Fungsi untuk mengurutkan data peminjaman menggunakan algoritma insertion sort
void insertionSortLoans(BookLoan loans[], int n) {
    if (n <= 1) {
        return;  // Basis: Array dengan 0 atau 1 elemen sudah terurut
    }

    // Urutkan elemen pertama dari loans[1...n]
    insertionSortLoans(loans, n - 1);

    // Masukkan loans[n] ke dalam posisi yang tepat di antara loans[0...n-1]
    BookLoan lastElement = loans[n - 1];
    int j = n - 2;

    // Geser elemen yang lebih besar dari lastElement ke kanan
    while (j >= 0 && loans[j].durationDays > lastElement.durationDays) {
        loans[j + 1] = loans[j];
        j--;
    }

    loans[j + 1] = lastElement;
}
```

Fungsi ini digunakan untuk mengurutkan array peminjam buku dalam urutan durasi peminjaman yang paling rendah menggunakan algoritma *insertion sort*. Dan fungsi ini menggunakan sistem rekursif.

- **Fungsi `borrowBook`**

```
// Fungsi untuk meminjam buku
void borrowBook(Book books[], unordered_map<int, int>& idIndex) {
    int bookId;
    cout << "Masukkan ID buku yang ingin dipinjam: ";
    cin >> bookId;
    cin.ignore();

    if (idIndex.count(bookId) == 0) {
        cout << "Buku dengan ID tersebut tidak ditemukan.\n";
        delay(3000);
        clearScreen();
        return;
    }

    int bookIndex = idIndex[bookId];

    // Memeriksa apakah buku sedang dipinjam
    bool isOnLoan = false;
    for (int i = 0; i < loanCount; i++) {
        if (loans[i].bookId == bookId) {
            isOnLoan = true;
            break;
        }
    }

    if (isOnLoan) {
        cout << "Tidak dapat meminjam buku \"" << books[bookIndex].title << " Volume " << books[bookIndex].volume << "\" karena sedang dipinjam oleh orang lain.\n";
        delay(3000);
        clearScreen();
        return;
    }
```

Fungsi ini digunakan untuk user yang mau meminjam buku, dan program akan memperbarui database peminjaman buku. Fungsi ini juga akan mengecek apakah ID buku yang dimasukkan user dan buku itu sedang dipinjam oleh user lain atau tidak. Dan fungsi ini menggunakan fungsi `insertionSortLoans` untuk pengurutan data peminjam buku. Lalu mengubah status buku di daftar buku menjadi "Sedang Dipinjam".

- **Fungsi `returnBook`**

```
// Fungsi untuk mengembalikan buku
void returnBook(Book books[], unordered_map<int, int>& idIndex) {
    int bookId;
    cout << "Masukkan ID buku yang ingin dikembalikan: ";
    cin >> bookId;
    cin.ignore();

    if (idIndex.count(bookId) == 0) {
        cout << "Buku dengan ID tersebut tidak ditemukan.\n";
        delay(3000);
        clearScreen();
        return;
    }

    int bookIndex = idIndex[bookId];
    bool isOnLoan = false;
    int loanIndex = -1;

    for (int i = 0; i < loanCount; i++) {
        if (loans[i].bookId == bookId) {
            isOnLoan = true;
            loanIndex = i;
            break;
        }
    }

    if (isOnLoan) {
        string borrowerName = loans[loanIndex].borrowerName;
        cout << "Buku dengan judul \"" << books[bookIndex].title << " Volume " << books[bookIndex].volume << "\" berhasil dikembalikan oleh " << borrowerName << ".\n";
        delay(4000);
        clearScreen();

        // Menghapus informasi peminjaman dari array loans
        for (int i = loanIndex; i < loanCount - 1; i++) {
            loans[i] = loans[i + 1];
        }
        loanCount--;
    } else {
        cout << "Buku dengan ID tersebut tidak sedang dipinjam.\n";
        delay(3000);
        clearScreen();
    }
}
```

Fungsi ini digunakan untuk user yang mau mengembalikan buku yang sedang dipinjam, dan memperbarui database peminjaman buku. Fungsi ini akan mengembil ID buku yang diberikan oleh user dan mengecek daftar buku hingga menemukan ID buku yang sama. Jika memang buku tersebut sedang dipinjam maka program akan mengembalikan status buku menjadi tersedia, akan tetapi jika ID buku tidak ditemukan atau ID buku yang dimasukkan salah maka program akan mengeluarkan peringatan.

- **Fungsi `displayLoanList`**

```
538  // Fungsi untuk menampilkan daftar peminjam buku
539  void displayLoanList(unordered_map<int, int>& idIndex, Book books[]) {
540      if (loanCount == 0) {
541          cout << "Tidak ada buku yang sedang dipinjam.\n";
542          delay(3000);
543          clearScreen();
544          return;
545      }
546
547      int currentPage = 1;
548      char choice;
549
550      do {
551          int startIndex = (currentPage - 1) * LOANS_PER_PAGE;
552          int endIndex = min(startIndex + LOANS_PER_PAGE, loanCount);
553
554          cout << "\n\n==============================================\n";
555          cout << "\nDaftar Peminjam Buku (Halaman " << currentPage << "):\n";
556          for (int i = startIndex; i < endIndex; i++) {
557              int bookIndex = idIndex[loans[i].bookId];
558              Book book = books[bookIndex];
559
560              cout << "ID Buku: " << book.id << "\n";
561              cout << "Peminjam: " << loans[i].borrowerName << "\n";
562              cout << "Judul Buku: " << book.title << " Volume " << book.volume << "\n";
563              cout << "Durasi Peminjaman: " << loans[i].durationDays << " hari\n\n";
564          }
565
566          if (endIndex < loanCount) {
567              cout << "==============================================\n";
568              cout << "Tekan 'n' untuk halaman selanjutnya atau 'q' untuk keluar: ";
569              choice = getch();
570
571              if (choice == 'n') {
572                  clearScreen();
573                  currentPage++;
574              } else if (choice != 'q') {
575                  cout << "Pilihan tidak valid. Silakan coba lagi.\n";
576              }
577          } else {
578              cout << "==============================================\n";
579              cout << "Tekan 'q' untuk keluar: ";
580              choice = getch();
581          }
582      } while (choice != 'q');
583      clearScreen();
584  }
```

Fungsi ini menampilkan daftar buku yang sedang dipinjam dengan menampilkan nama buku yang sedang dipinjam, nama user yang sedang meminjam, dan durasi peminjaman buku. Maksimal daftar peminjam yang dapat ditampilkan adalah 5.

- **Main program**

```
586 int main() {
587     int choice;
588     Book books[MAX_BOOKS]; // Array untuk menyimpan kumpulan buku
589     int bookCount = 0; // Jumlah buku yang telah ditambahkan
590     int currentPage = 1; // Indeks buku yang sedang ditampilkan
591 
592     unordered_map<string, vector<int>> titleIndex; // Hash table untuk menyimpan indeks buku berdasarkan judul
593     unordered_map<int, int> idIndex; // Hash table untuk menyimpan indeks buku berdasarkan ID
594 
595     loadBookData(books, bookCount, titleIndex, idIndex); // Membaca data buku dari file CSV
596     loadLoanData(idIndex); // Membaca data peminjaman dari file CSV
597 
598     do {
599         cout << "\n===== APLIKASI MANAJEMEN PERPUSTAKAAN =====\n";
600         cout << "Jumlah Buku: " << bookCount << endl << "Jumlah buku yang dipinjam: " << loanCount << endl << endl;
601         cout << "1. Tambah Buku Baru\n";
602         cout << "2. Cari Buku\n";
603         cout << "3. Daftar Buku\n";
604         cout << "4. Peminjaman Buku\n";
605         cout << "5. Pengembalian Buku\n";
606         cout << "6. Daftar Peminjam Buku\n";
607         cout << "7. Simpan Data Buku\n";
608         cout << "8. Keluar\n";
609         cout << "Pilihan Anda: ";
610         choice = getch();
611 
612         if (choice >= '1' && choice <= '8') { // Memeriksa apakah input berupa angka 1-8
613             choice = choice - '0'; // Konversi dari nilai ASCII ke nilai numerik
614         }
615         cout << choice << endl;
```

1. `books[MAX_BOOKS]`: Array yang digunakan untuk menyimpan Kumpulan buku.
2. `unordered_map<string, vector<int>> titleIndex`: hash table yang menggunakan library unordered_map untuk penyimpanan indeks judul buku.
3. `unordered_map<int, int> idIndex`: hash table yang menggunakan library unordered_map untuk penyimpanan indeks ID buku yang dipinjam.
4. `loadBookData(books, bookCount, titleIndex, idIndex)`: digunakan untuk meload data buku dari file *.csv*.
5. `loadLoanData(idIndex)`: digunakan untuk meload data peminjam buku dari file *.csv*.

- **Catatan**
    1. Q: Mengapa saya menggunakan library <unordered_map> untuk membuat hash table

       A: Karena dengan menggunakan library tersebut penggunaan hash table lebih efisien dengan kompleksitas waktu O(1), lalu data yang bisa disimpan sangat banyak (sebagai contoh, buku komik *Hanako Si Arwah Penasaran* tidak hanya tersedia dalam 1 volume saja akan tetapi memiliki banyak volume), jika menggunakan library ini, penggunaan hash table akan menjadi lebih fleksibel dan dapat menyimpan banyak sekali buku walaupun memiliki judul yang sama, asalkan ada pembeda satu buku dengan buku yang lain.
    2. Q: Materi yang sudah diimplementasikan apa aja?

       A: Rekursif, sorting, searching, collison handling & hash table (untuk tree dan graph saya bingung bagaimana penerapannya di dalam program ini)