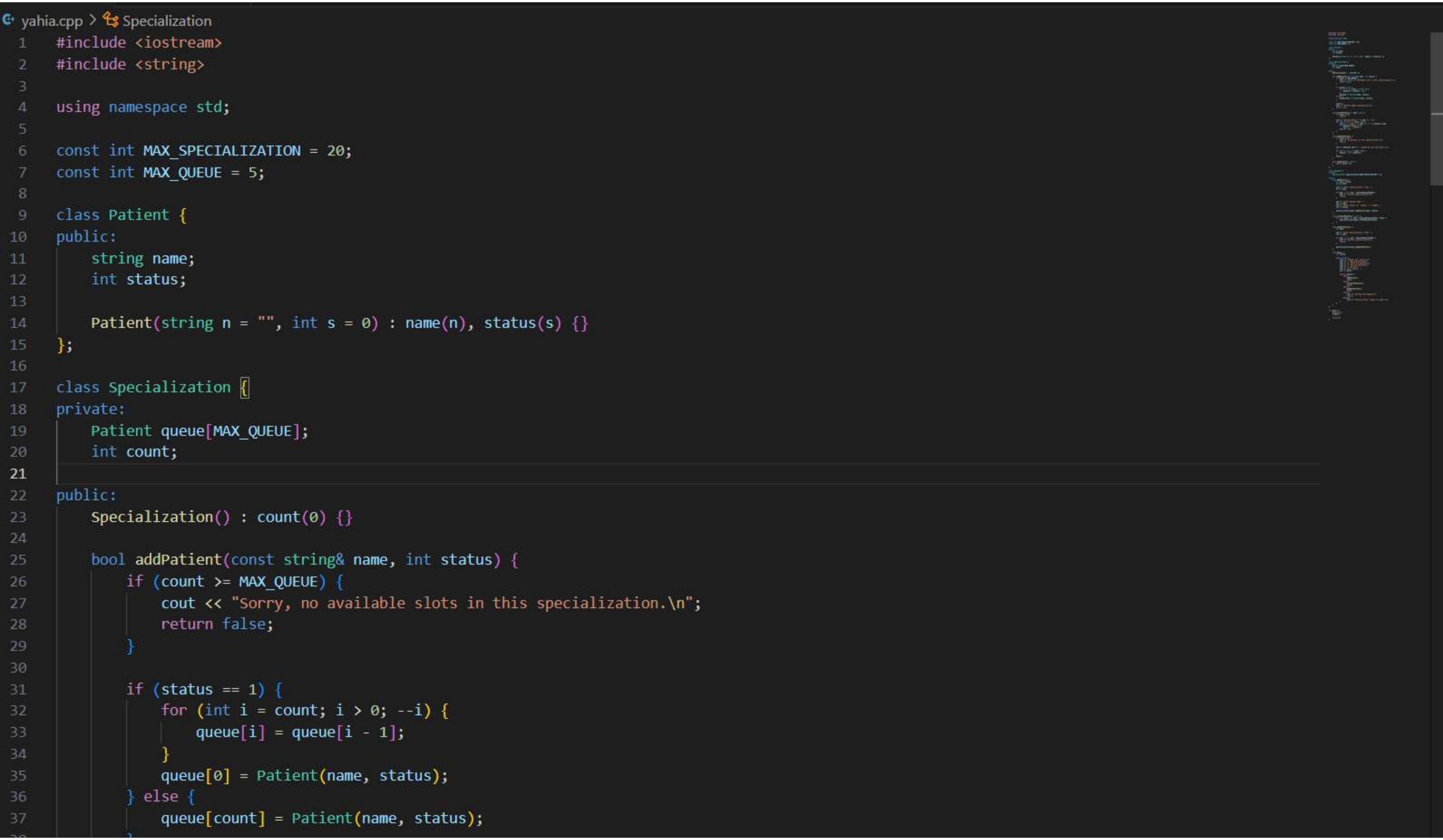
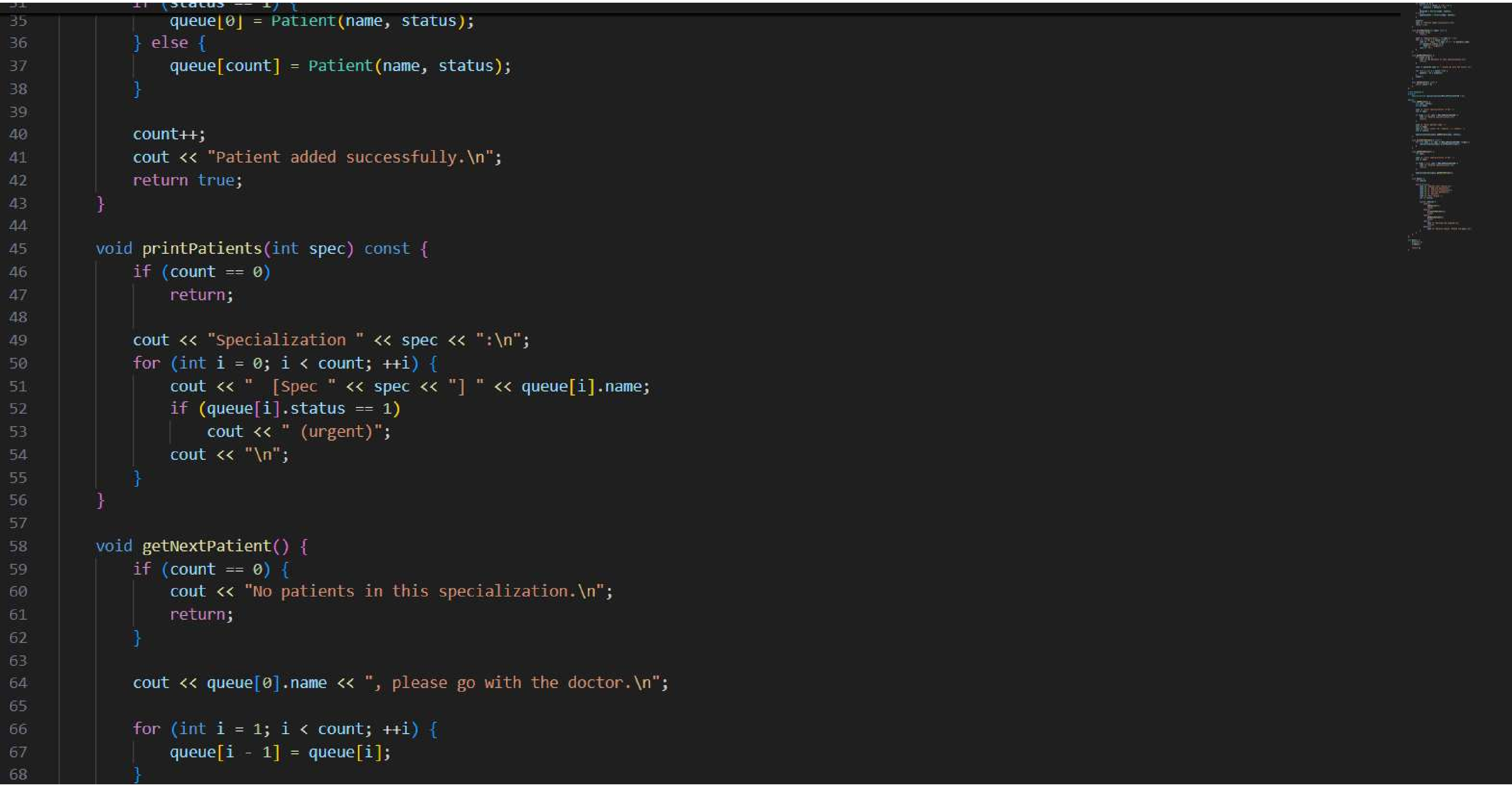
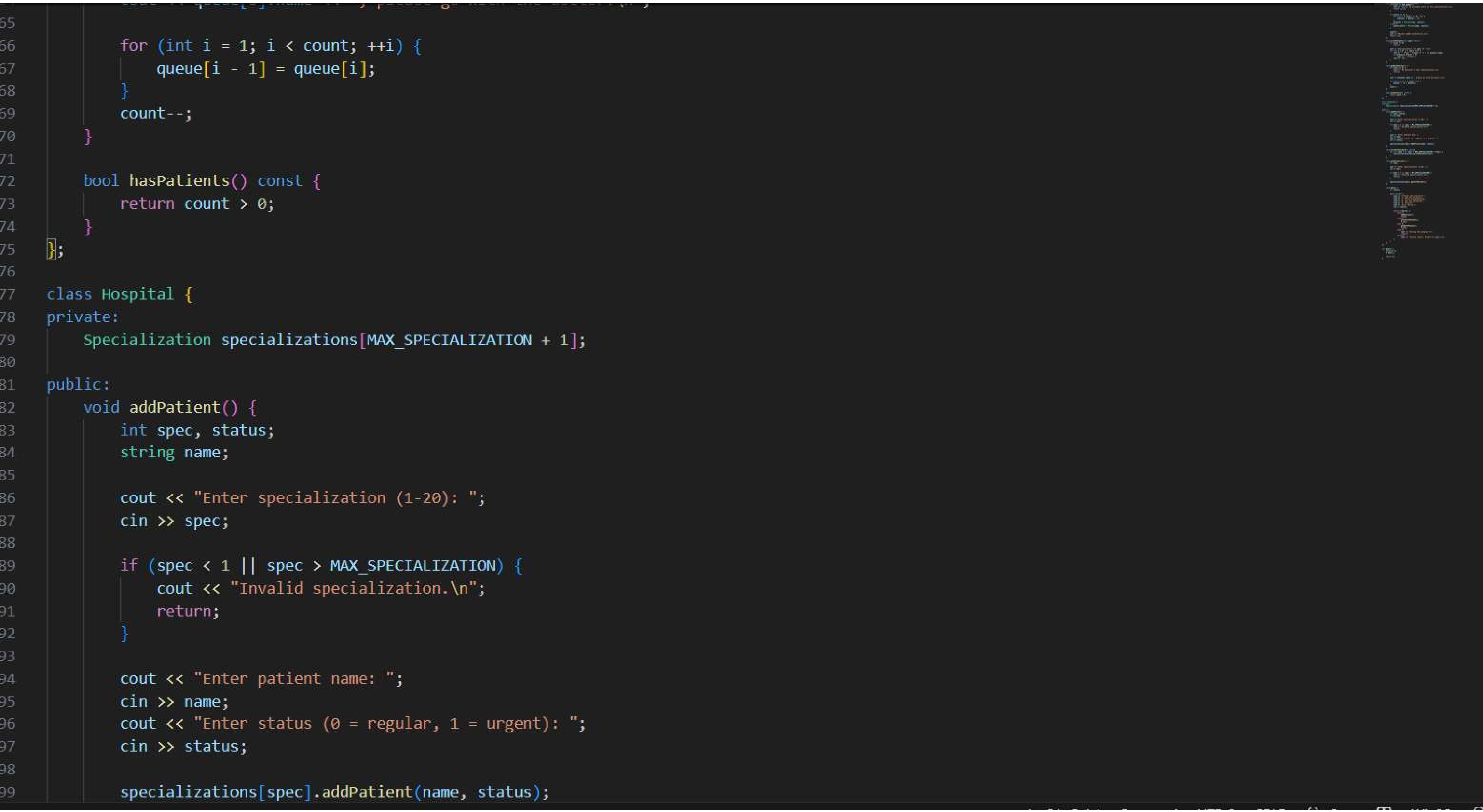


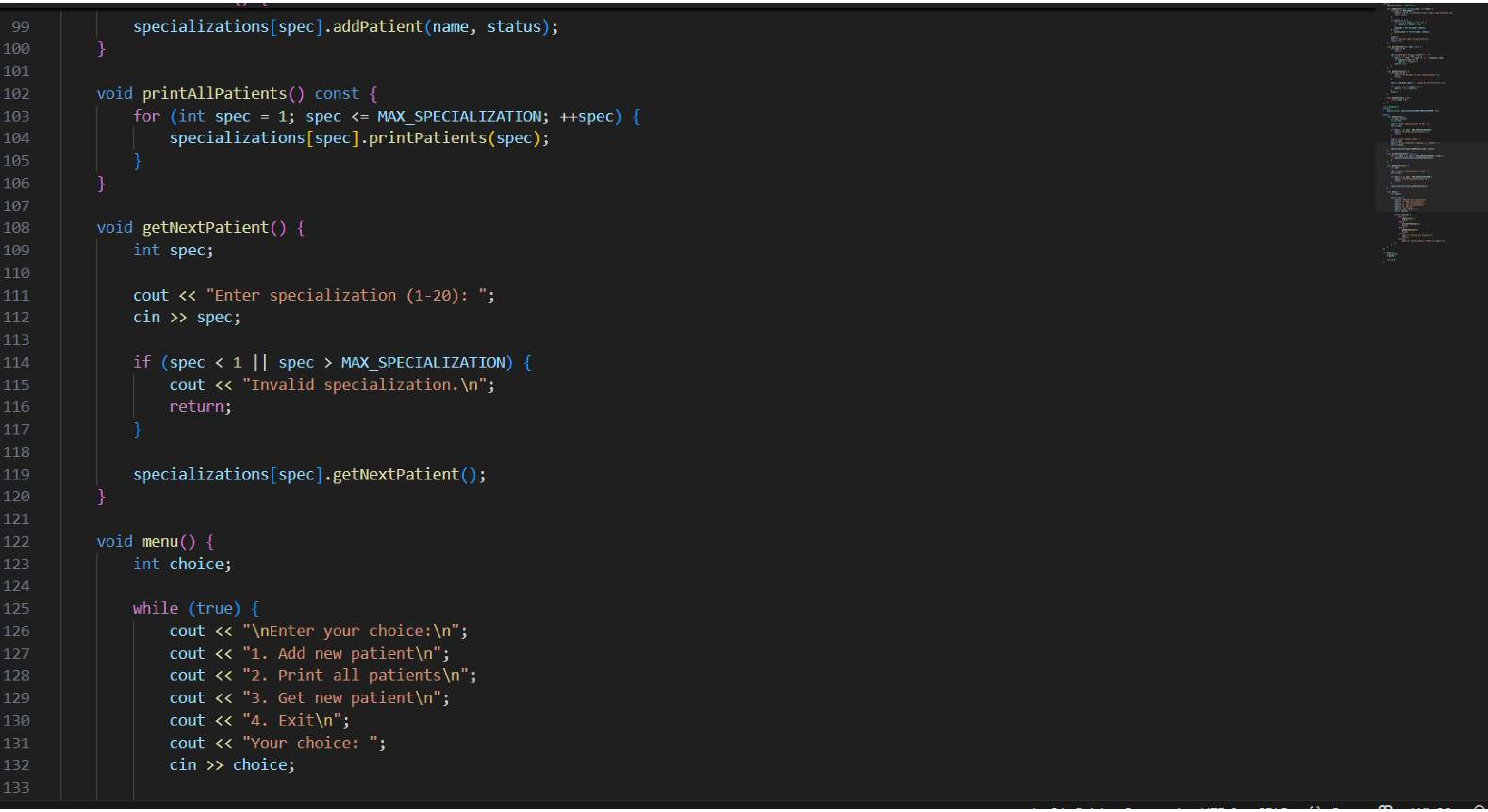
```cpp
#include <iostream>
#include <string>

using namespace std;

const int MAX_SPECIALIZATION = 20;
const int MAX_QUEUE = 5;

class Patient {
public:
    string name;
    int status;

    Patient(string n = "", int s = 0) : name(n), status(s) {}
};

class Specialization {
private:
    Patient queue[MAX_QUEUE];
    int count;

public:
    Specialization() : count(0) {}

    bool addPatient(const string& name, int status) {
        if (count >= MAX_QUEUE) {
            cout << "Sorry, no available slots in this specialization.\n";
            return false;
        }

        if (status == 1) {
            for (int i = count; i > 0; --i) {
                queue[i] = queue[i - 1];
            }
            queue[0] = Patient(name, status);
        } else {
            queue[count] = Patient(name, status);
```



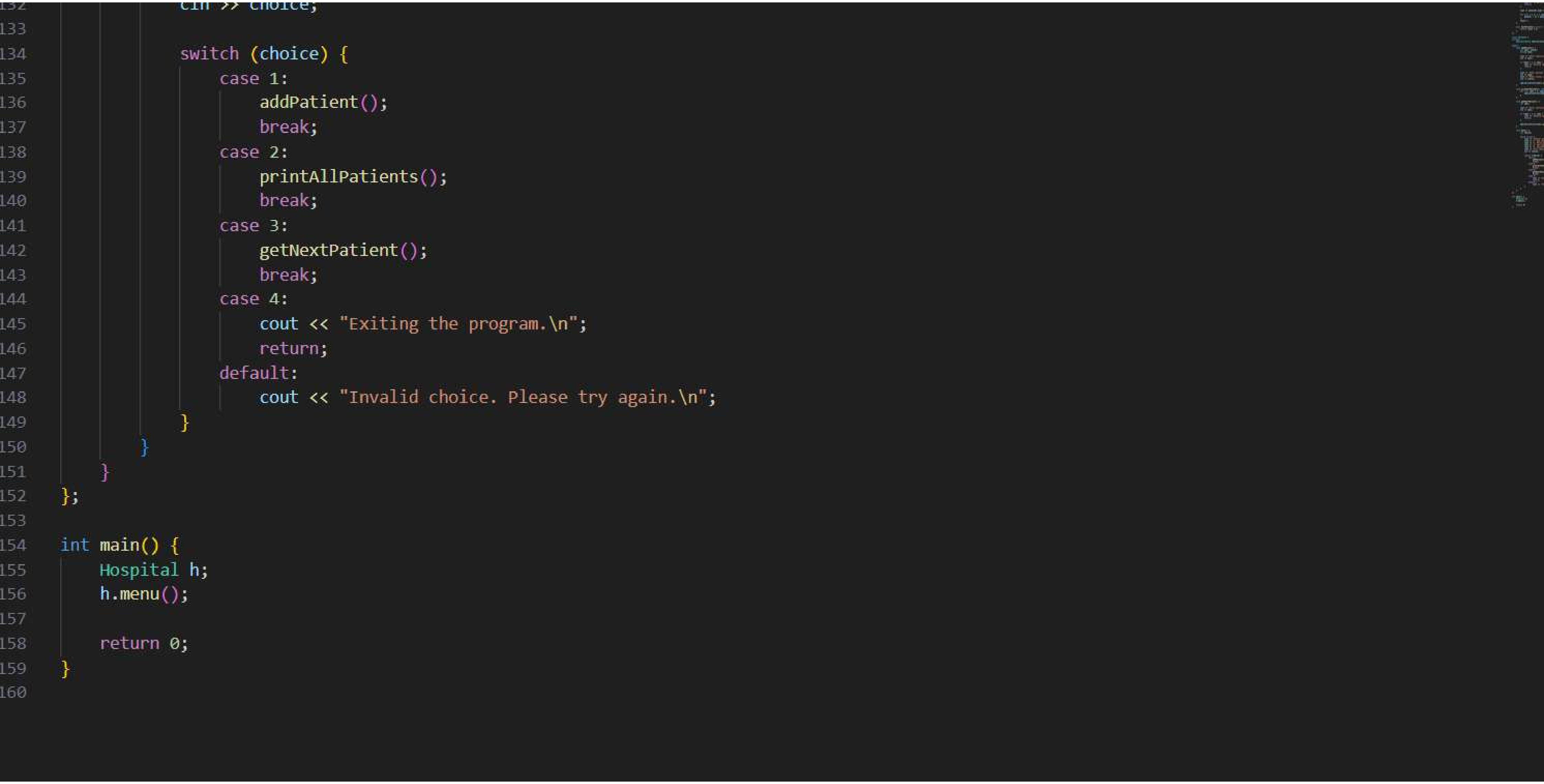
```cpp
        if (status == 1) {
            queue[0] = Patient(name, status);
        } else {
            queue[count] = Patient(name, status);
        }

        count++;
        cout << "Patient added successfully.\n";
        return true;
    }

    void printPatients(int spec) const {
        if (count == 0)
            return;

        cout << "Specialization " << spec << ":\n";
        for (int i = 0; i < count; ++i) {
            cout << "  [Spec " << spec << "] " << queue[i].name;
            if (queue[i].status == 1)
                cout << " (urgent)";
            cout << "\n";
        }
    }

    void getNextPatient() {
        if (count == 0) {
            cout << "No patients in this specialization.\n";
            return;
        }

        cout << queue[0].name << ", please go with the doctor.\n";

        for (int i = 1; i < count; ++i) {
            queue[i - 1] = queue[i];
        }
```



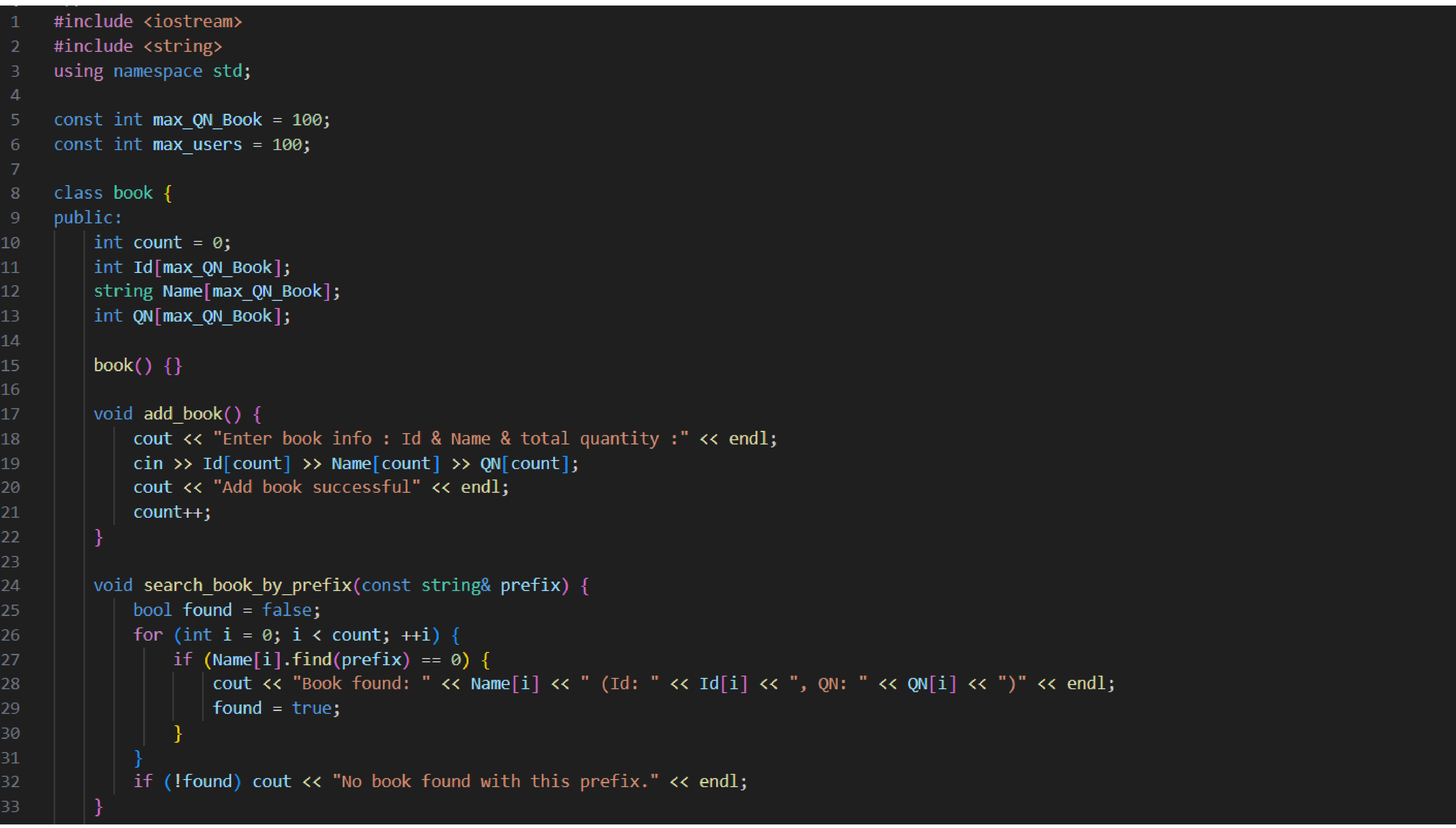
```cpp

        for (int i = 1; i < count; ++i) {
            queue[i - 1] = queue[i];
        }
        count--;
    }

    bool hasPatients() const {
        return count > 0;
    }
};

class Hospital {
private:
    Specialization specializations[MAX_SPECIALIZATION + 1];

public:
    void addPatient() {
        int spec, status;
        string name;

        cout << "Enter specialization (1-20): ";
        cin >> spec;

        if (spec < 1 || spec > MAX_SPECIALIZATION) {
            cout << "Invalid specialization.\n";
            return;
        }

        cout << "Enter patient name: ";
        cin >> name;
        cout << "Enter status (0 = regular, 1 = urgent): ";
        cin >> status;

        specializations[spec].addPatient(name, status);
```

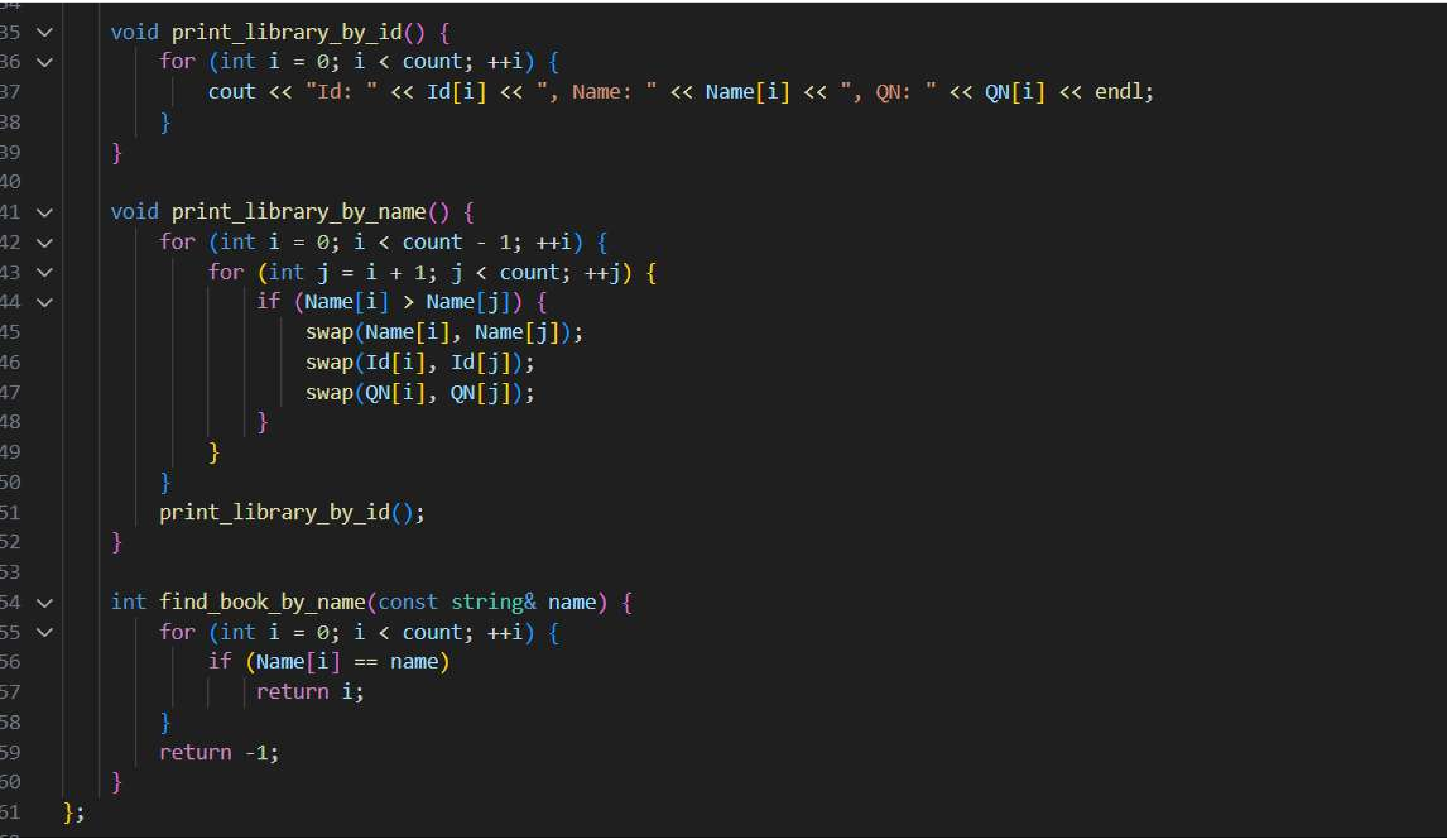


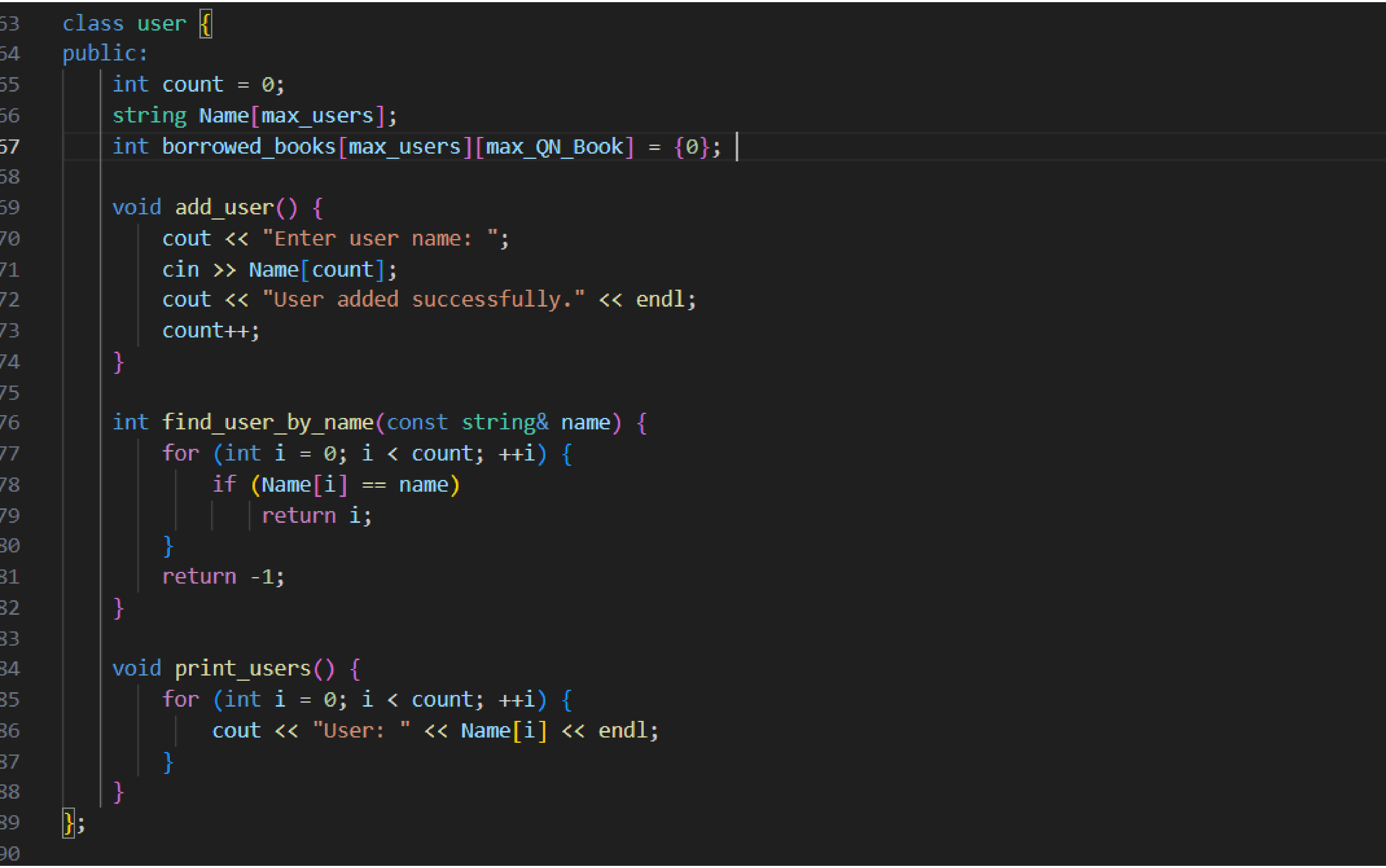
```
        specializations[spec].addPatient(name, status);
    }

    void printAllPatients() const {
        for (int spec = 1; spec <= MAX_SPECIALIZATION; ++spec) {
            specializations[spec].printPatients(spec);
        }
    }

    void getNextPatient() {
        int spec;

        cout << "Enter specialization (1-20): ";
        cin >> spec;

        if (spec < 1 || spec > MAX_SPECIALIZATION) {
            cout << "Invalid specialization.\n";
            return;
        }

        specializations[spec].getNextPatient();
    }

    void menu() {
        int choice;

        while (true) {
            cout << "\nEnter your choice:\n";
            cout << "1. Add new patient\n";
            cout << "2. Print all patients\n";
            cout << "3. Get new patient\n";
            cout << "4. Exit\n";
            cout << "Your choice: ";
            cin >> choice;

```

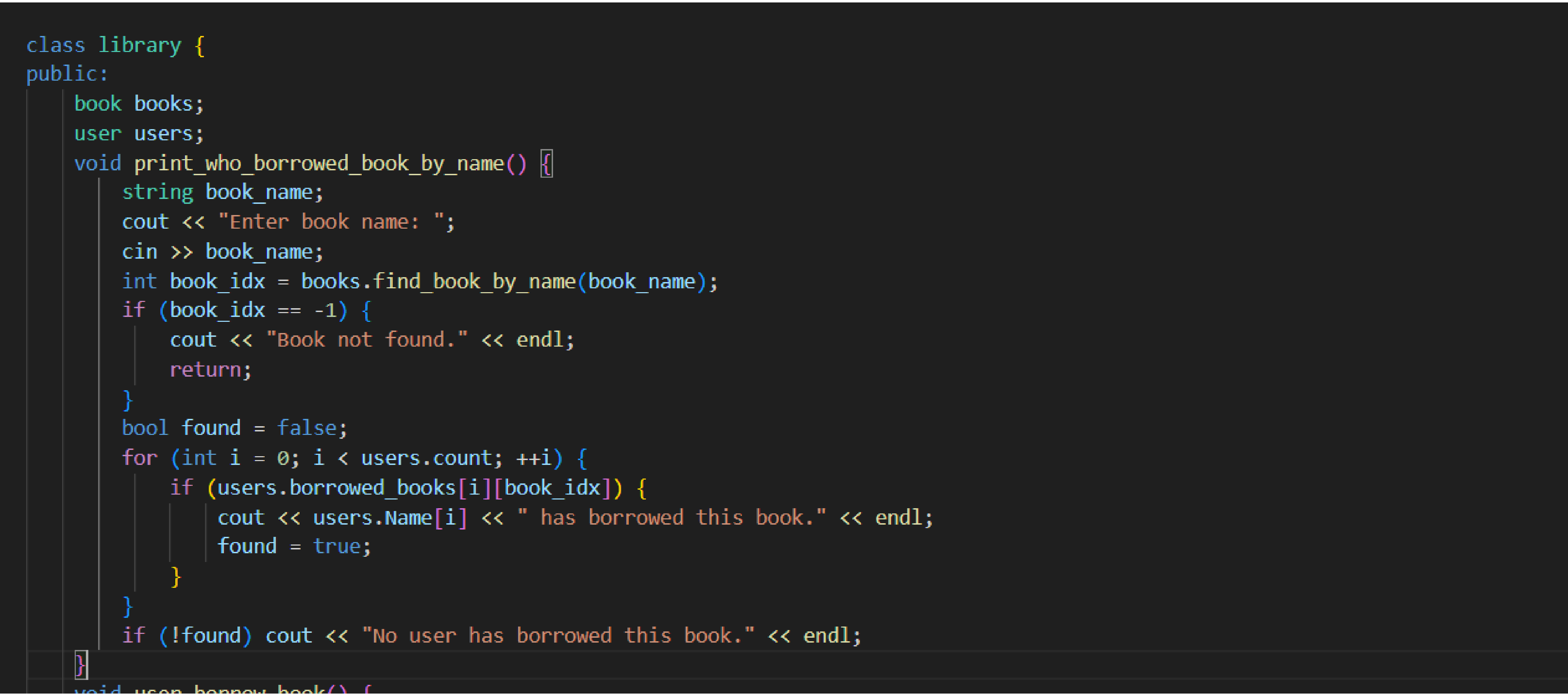


```cpp
            cin >> choice;

            switch (choice) {
                case 1:
                    addPatient();
                    break;
                case 2:
                    printAllPatients();
                    break;
                case 3:
                    getNextPatient();
                    break;
                case 4:
                    cout << "Exiting the program.\n";
                    return;
                default:
                    cout << "Invalid choice. Please try again.\n";
            }
        }
    }
};

int main() {
    Hospital h;
    h.menu();

    return 0;
}
```

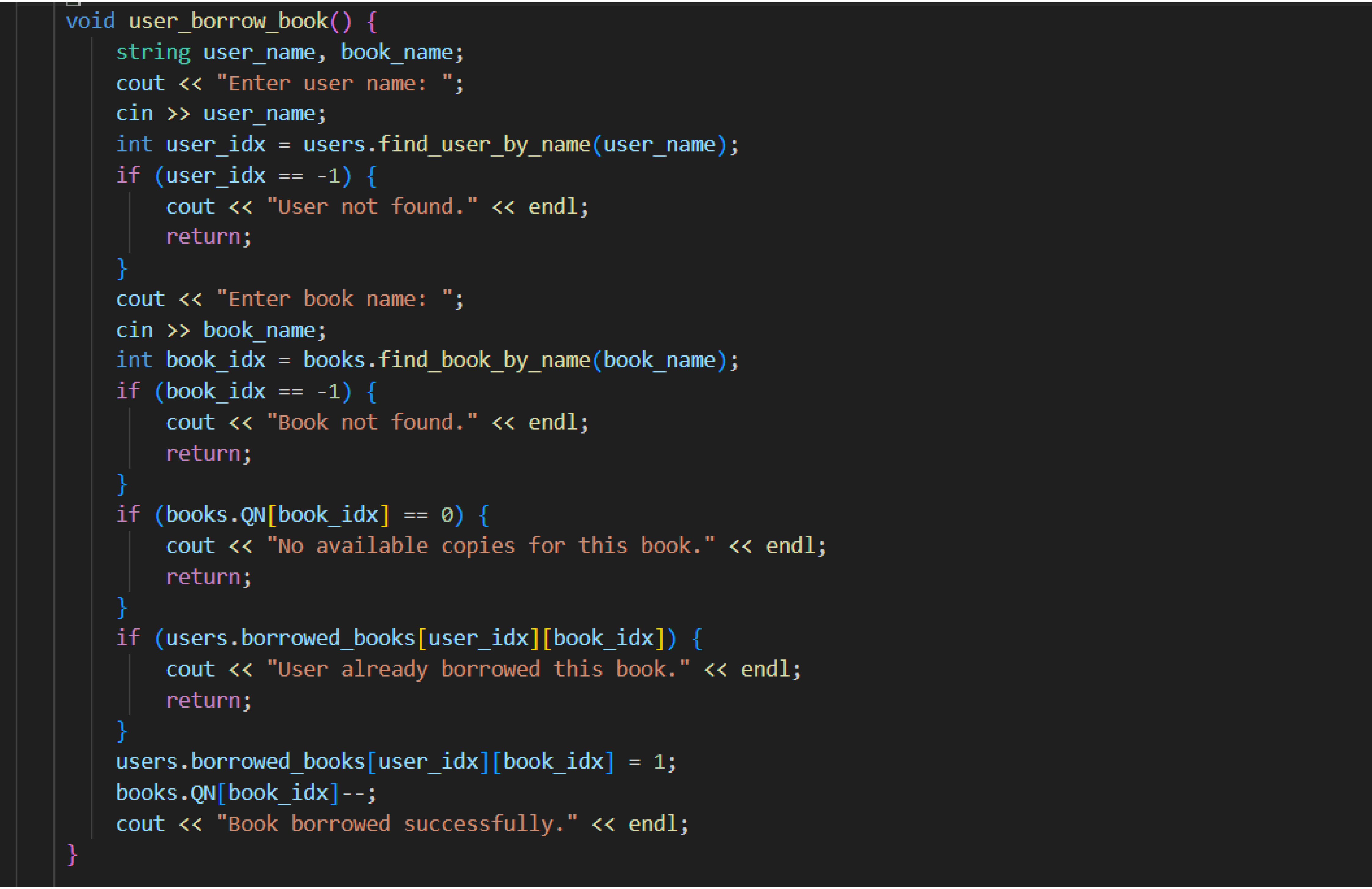

```cpp
#include <iostream>
#include <string>
using namespace std;

const int max_QN_Book = 100;
const int max_users = 100;

class book {
public:
    int count = 0;
    int Id[max_QN_Book];
    string Name[max_QN_Book];
    int QN[max_QN_Book];

    book() {}

    void add_book() {
        cout << "Enter book info : Id & Name & total quantity :" << endl;
        cin >> Id[count] >> Name[count] >> QN[count];
        cout << "Add book successful" << endl;
        count++;
    }

    void search_book_by_prefix(const string& prefix) {
        bool found = false;
        for (int i = 0; i < count; ++i) {
            if (Name[i].find(prefix) == 0) {
                cout << "Book found: " << Name[i] << " (Id: " << Id[i] << ", QN: " << QN[i] << ")" << endl;
                found = true;
            }
        }
        if (!found) cout << "No book found with this prefix." << endl;
    }
```


```
    void print_library_by_id() {
        for (int i = 0; i < count; ++i) {
            cout << "Id: " << Id[i] << ", Name: " << Name[i] << ", QN: " << QN[i] << endl;
        }
    }

    void print_library_by_name() {
        for (int i = 0; i < count - 1; ++i) {
            for (int j = i + 1; j < count; ++j) {
                if (Name[i] > Name[j]) {
                    swap(Name[i], Name[j]);
                    swap(Id[i], Id[j]);
                    swap(QN[i], QN[j]);
                }
            }
        }
        print_library_by_id();
    }

    int find_book_by_name(const string& name) {
        for (int i = 0; i < count; ++i) {
            if (Name[i] == name)
                return i;
        }
        return -1;
    }
};
```

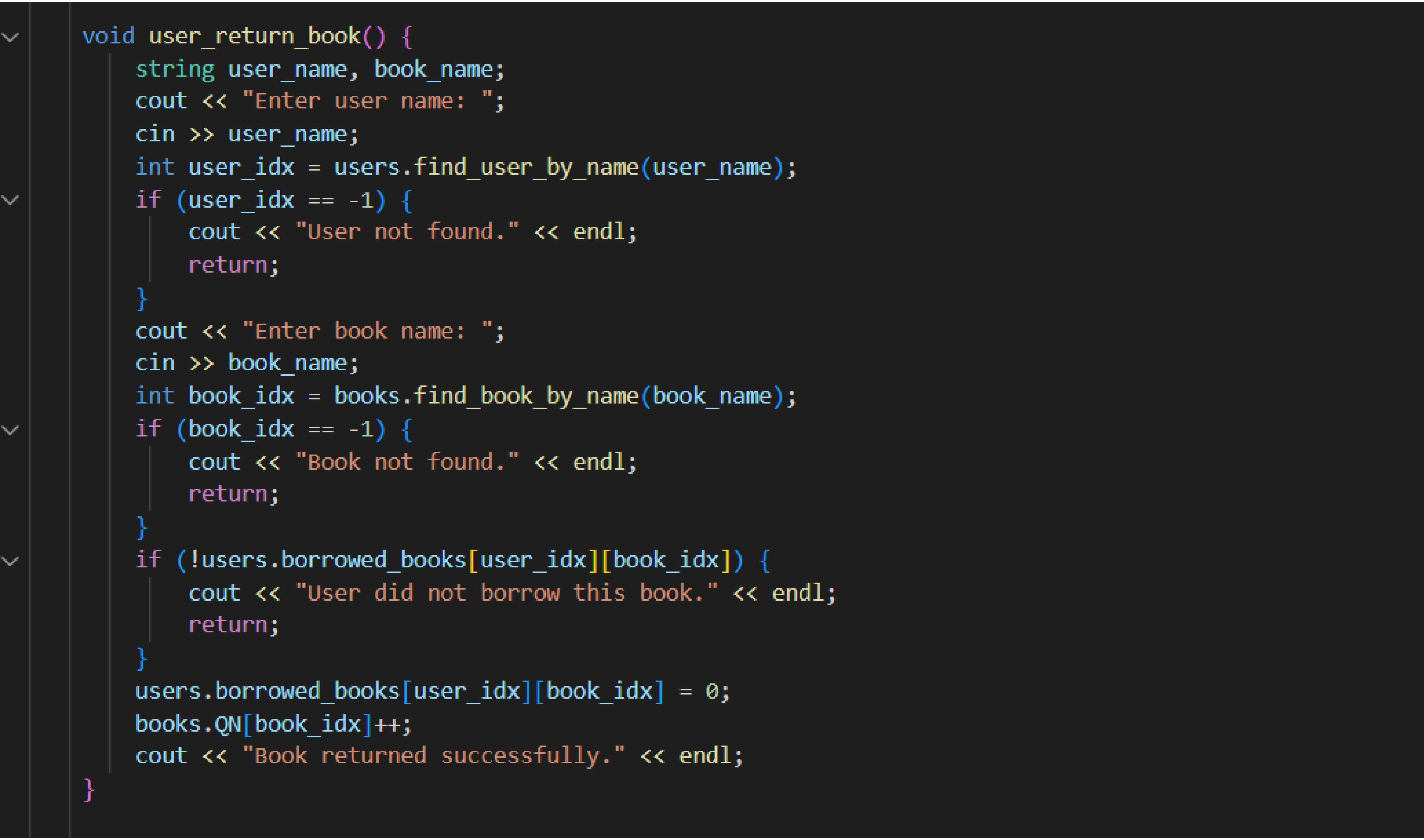

```
class user {
public:
    int count = 0;
    string Name[max_users];
    int borrowed_books[max_users][max_QN_Book] = {0};

    void add_user() {
        cout << "Enter user name: ";
        cin >> Name[count];
        cout << "User added successfully." << endl;
        count++;
    }

    int find_user_by_name(const string& name) {
        for (int i = 0; i < count; ++i) {
            if (Name[i] == name)
                return i;
        }
        return -1;
    }

    void print_users() {
        for (int i = 0; i < count; ++i) {
            cout << "User: " << Name[i] << endl;
        }
    }
};
```

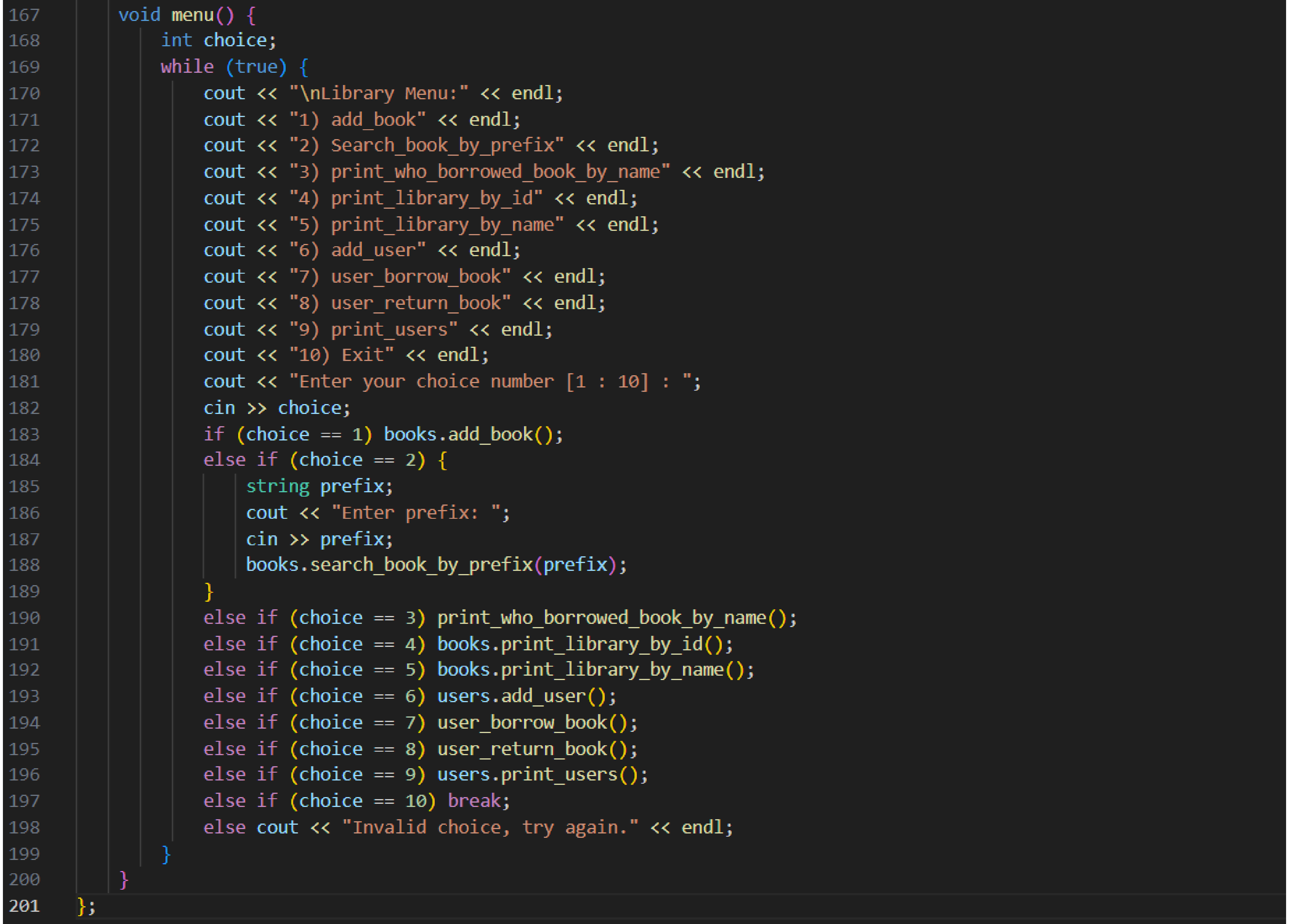


```cpp
class library {
public:
    book books;
    user users;
    void print_who_borrowed_book_by_name() {
        string book_name;
        cout << "Enter book name: ";
        cin >> book_name;
        int book_idx = books.find_book_by_name(book_name);
        if (book_idx == -1) {
            cout << "Book not found." << endl;
            return;
        }
        bool found = false;
        for (int i = 0; i < users.count; ++i) {
            if (users.borrowed_books[i][book_idx]) {
                cout << users.Name[i] << " has borrowed this book." << endl;
                found = true;
            }
        }
        if (!found) cout << "No user has borrowed this book." << endl;
    }
```

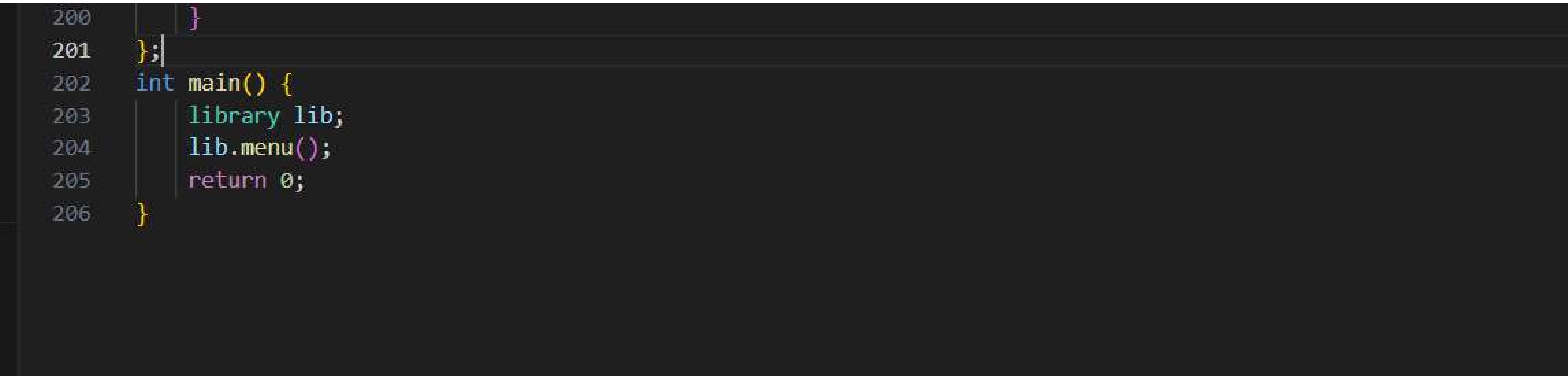


```cpp
void user_borrow_book() {
    string user_name, book_name;
    cout << "Enter user name: ";
    cin >> user_name;
    int user_idx = users.find_user_by_name(user_name);
    if (user_idx == -1) {
        cout << "User not found." << endl;
        return;
    }
    cout << "Enter book name: ";
    cin >> book_name;
    int book_idx = books.find_book_by_name(book_name);
    if (book_idx == -1) {
        cout << "Book not found." << endl;
        return;
    }
    if (books.QN[book_idx] == 0) {
        cout << "No available copies for this book." << endl;
        return;
    }
    if (users.borrowed_books[user_idx][book_idx]) {
        cout << "User already borrowed this book." << endl;
        return;
    }
    users.borrowed_books[user_idx][book_idx] = 1;
    books.QN[book_idx]--;
    cout << "Book borrowed successfully." << endl;
}
```

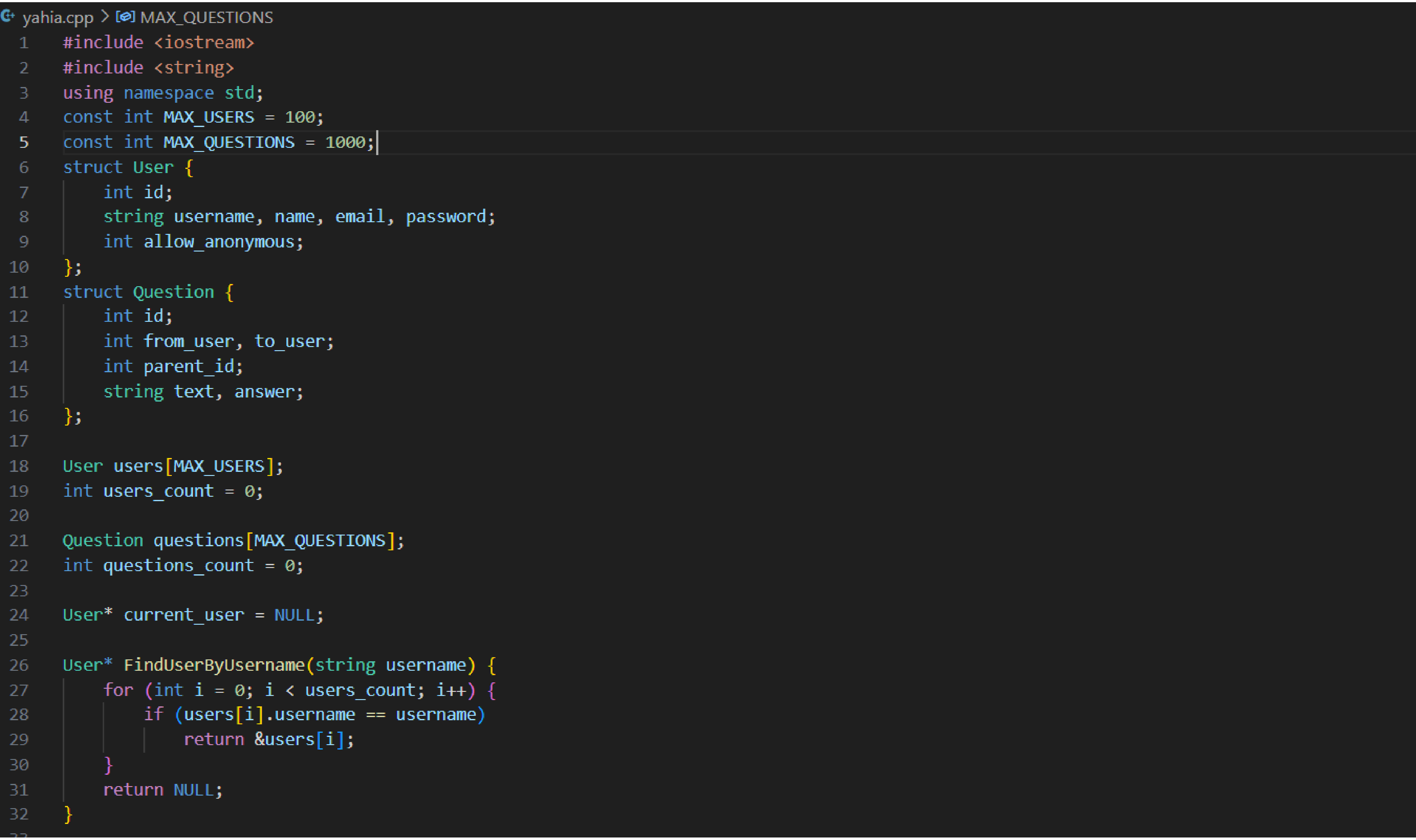

```cpp
void user_return_book() {
    string user_name, book_name;
    cout << "Enter user name: ";
    cin >> user_name;
    int user_idx = users.find_user_by_name(user_name);
    if (user_idx == -1) {
        cout << "User not found." << endl;
        return;
    }
    cout << "Enter book name: ";
    cin >> book_name;
    int book_idx = books.find_book_by_name(book_name);
    if (book_idx == -1) {
        cout << "Book not found." << endl;
        return;
    }
    if (!users.borrowed_books[user_idx][book_idx]) {
        cout << "User did not borrow this book." << endl;
        return;
    }
    users.borrowed_books[user_idx][book_idx] = 0;
    books.QN[book_idx]++;
    cout << "Book returned successfully." << endl;
}
```



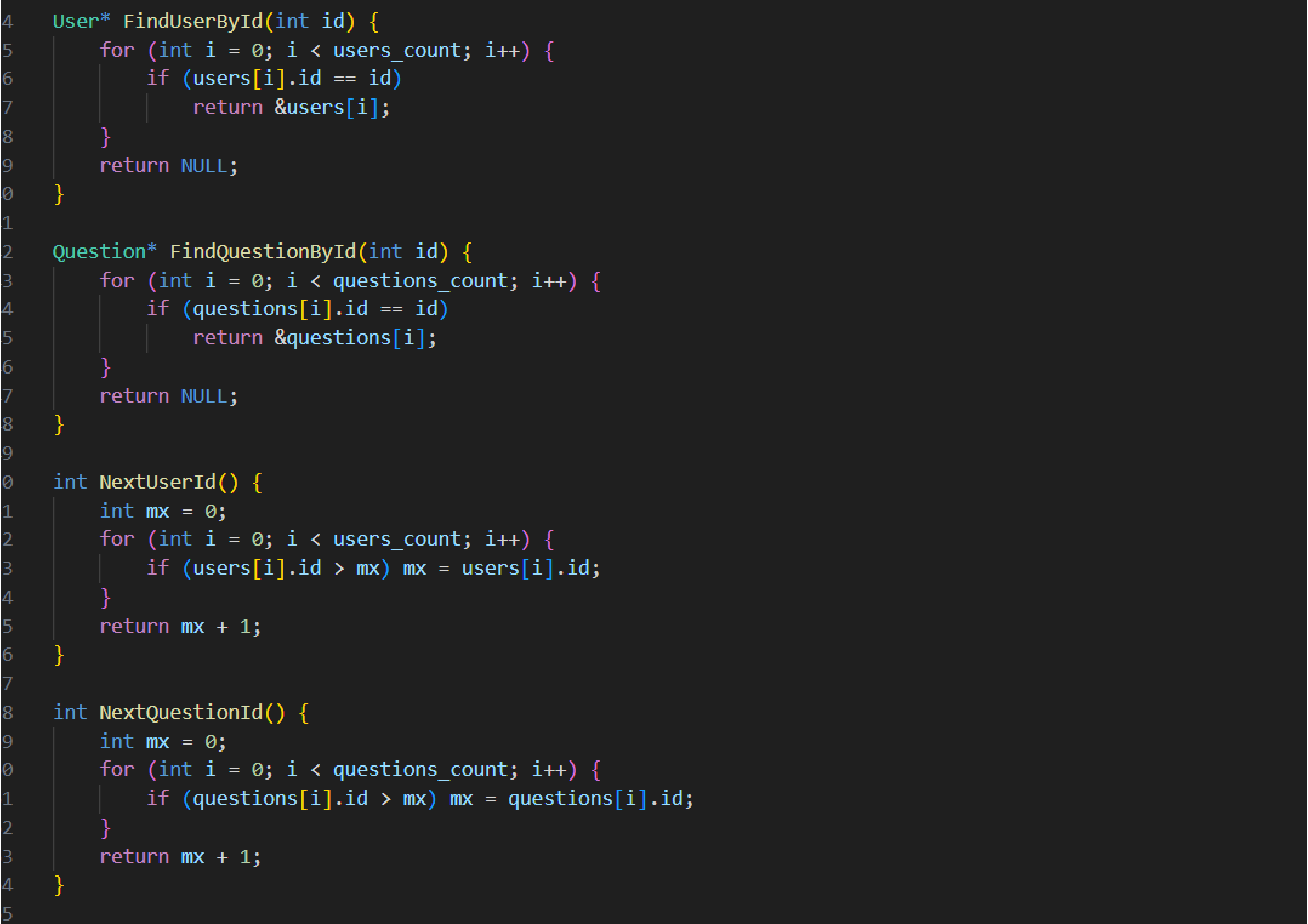
```cpp
    void menu() {
        int choice;
        while (true) {
            cout << "\nLibrary Menu:" << endl;
            cout << "1) add_book" << endl;
            cout << "2) Search_book_by_prefix" << endl;
            cout << "3) print_who_borrowed_book_by_name" << endl;
            cout << "4) print_library_by_id" << endl;
            cout << "5) print_library_by_name" << endl;
            cout << "6) add_user" << endl;
            cout << "7) user_borrow_book" << endl;
            cout << "8) user_return_book" << endl;
            cout << "9) print_users" << endl;
            cout << "10) Exit" << endl;
            cout << "Enter your choice number [1 : 10] : ";
            cin >> choice;
            if (choice == 1) books.add_book();
            else if (choice == 2) {
                string prefix;
                cout << "Enter prefix: ";
                cin >> prefix;
                books.search_book_by_prefix(prefix);
            }
            else if (choice == 3) print_who_borrowed_book_by_name();
            else if (choice == 4) books.print_library_by_id();
            else if (choice == 5) books.print_library_by_name();
            else if (choice == 6) users.add_user();
            else if (choice == 7) user_borrow_book();
            else if (choice == 8) user_return_book();
            else if (choice == 9) users.print_users();
            else if (choice == 10) break;
            else cout << "Invalid choice, try again." << endl;
        }
    }
};
```



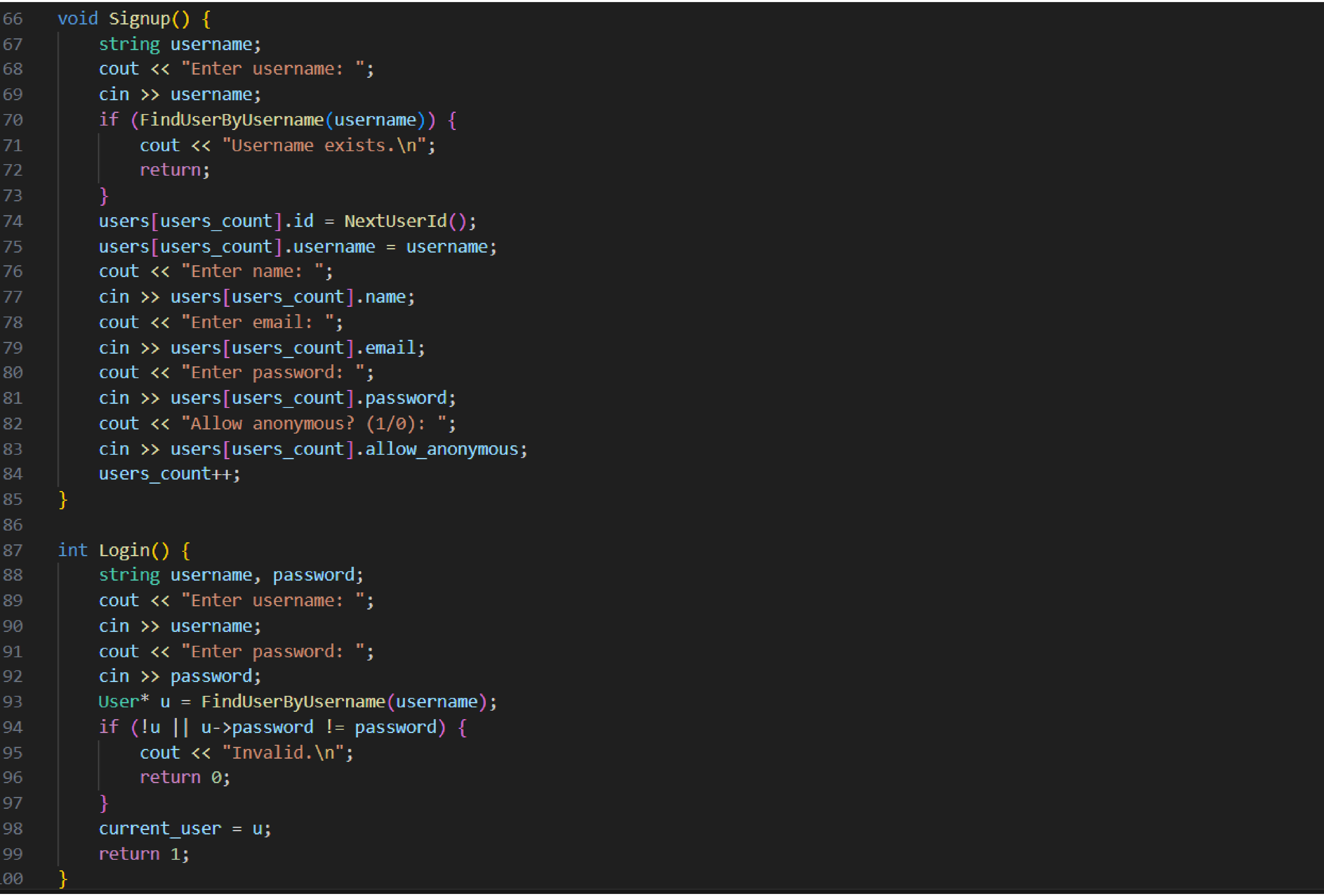
```
    }
};
int main() {
    library lib;
    lib.menu();
    return 0;
}
```


yahia.cpp > MAX_QUESTIONS

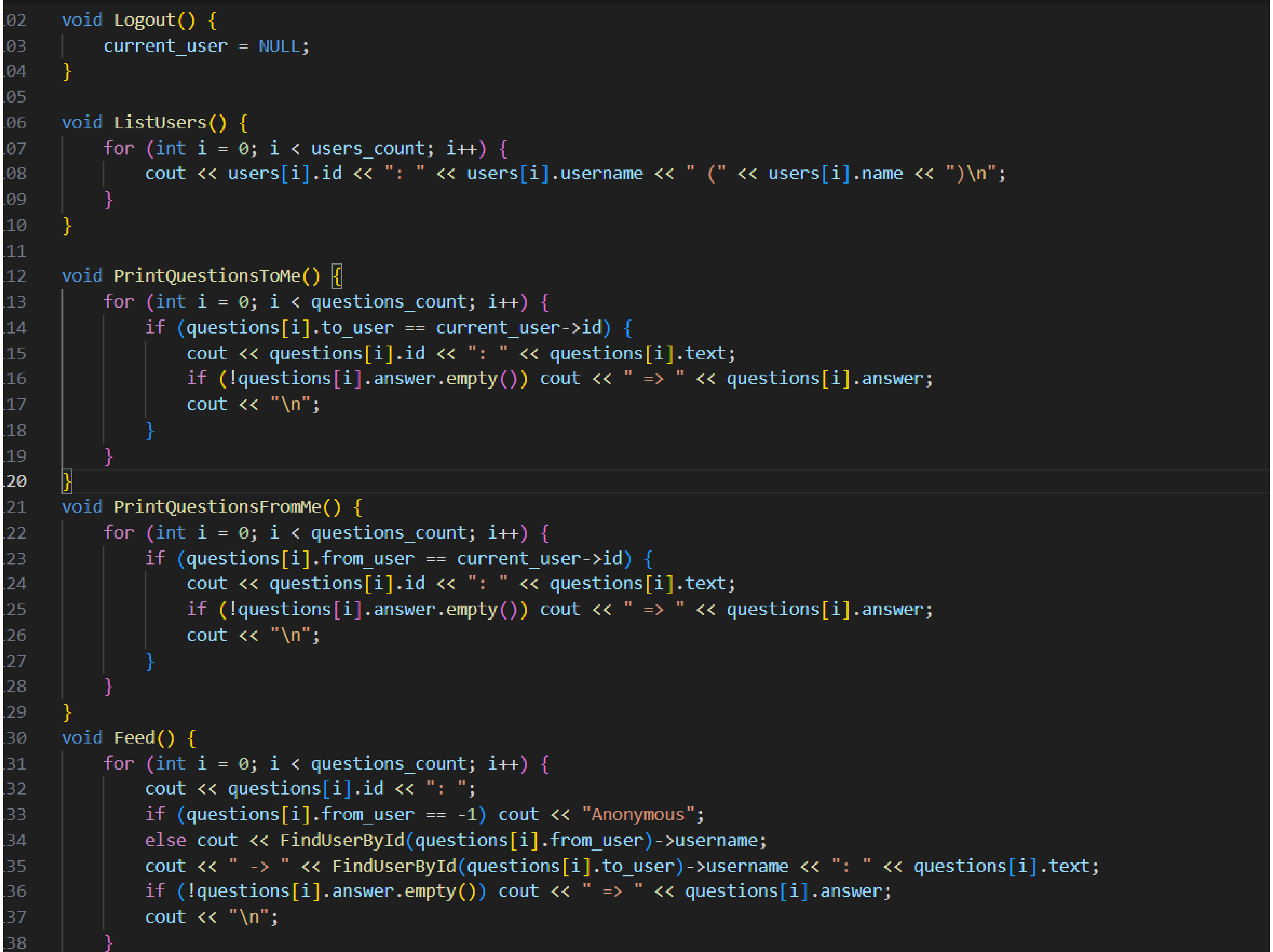
```cpp
#include <iostream>
#include <string>
using namespace std;
const int MAX_USERS = 100;
const int MAX_QUESTIONS = 1000;
struct User {
    int id;
    string username, name, email, password;
    int allow_anonymous;
};
struct Question {
    int id;
    int from_user, to_user;
    int parent_id;
    string text, answer;
};

User users[MAX_USERS];
int users_count = 0;

Question questions[MAX_QUESTIONS];
int questions_count = 0;

User* current_user = NULL;

User* FindUserByUsername(string username) {
    for (int i = 0; i < users_count; i++) {
        if (users[i].username == username)
            return &users[i];
    }
    return NULL;
}
```



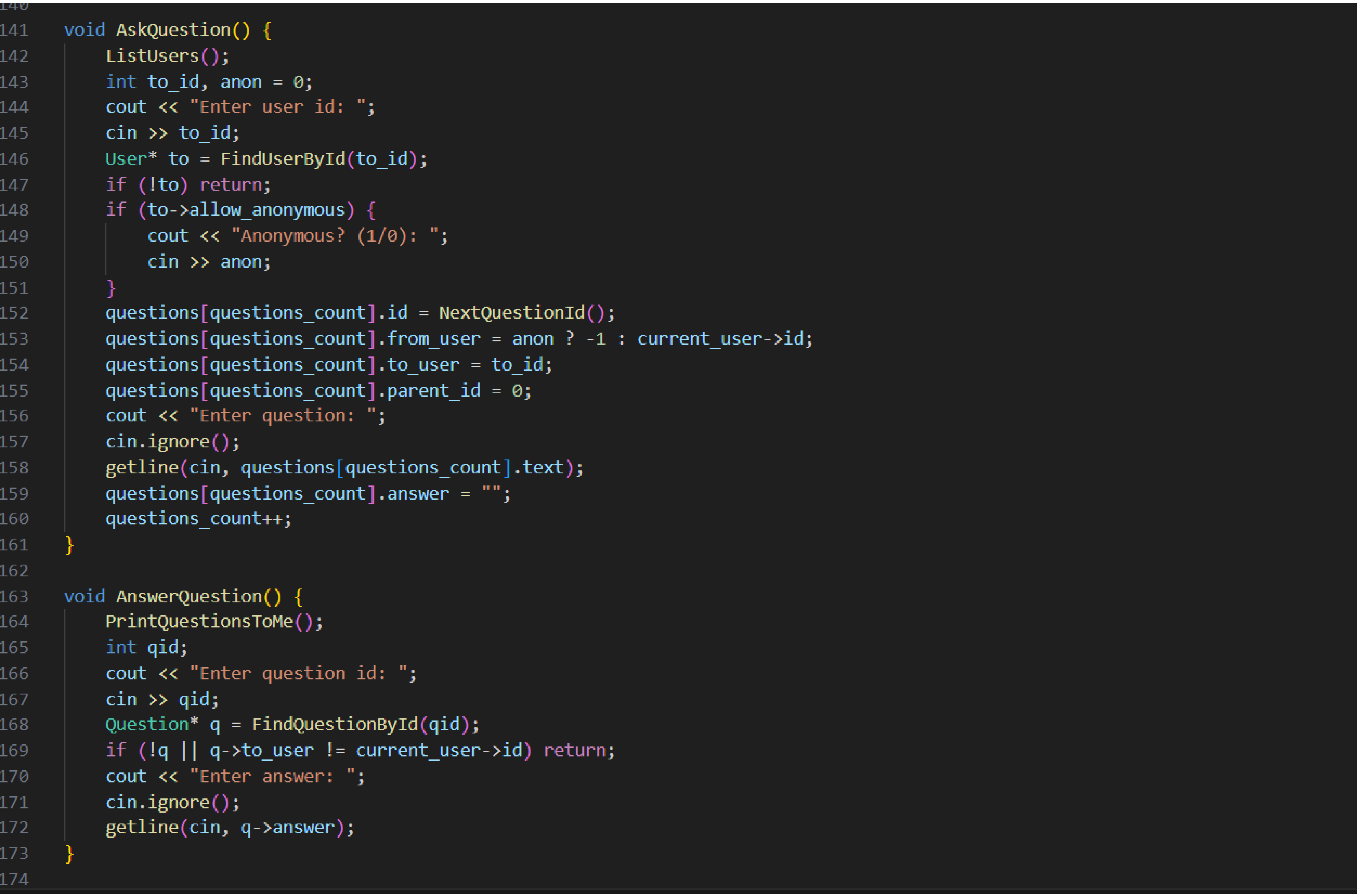
```
User* FindUserById(int id) {
    for (int i = 0; i < users_count; i++) {
        if (users[i].id == id)
            return &users[i];
    }
    return NULL;
}

Question* FindQuestionById(int id) {
    for (int i = 0; i < questions_count; i++) {
        if (questions[i].id == id)
            return &questions[i];
    }
    return NULL;
}

int NextUserId() {
    int mx = 0;
    for (int i = 0; i < users_count; i++) {
        if (users[i].id > mx) mx = users[i].id;
    }
    return mx + 1;
}

int NextQuestionId() {
    int mx = 0;
    for (int i = 0; i < questions_count; i++) {
        if (questions[i].id > mx) mx = questions[i].id;
    }
    return mx + 1;
}
```



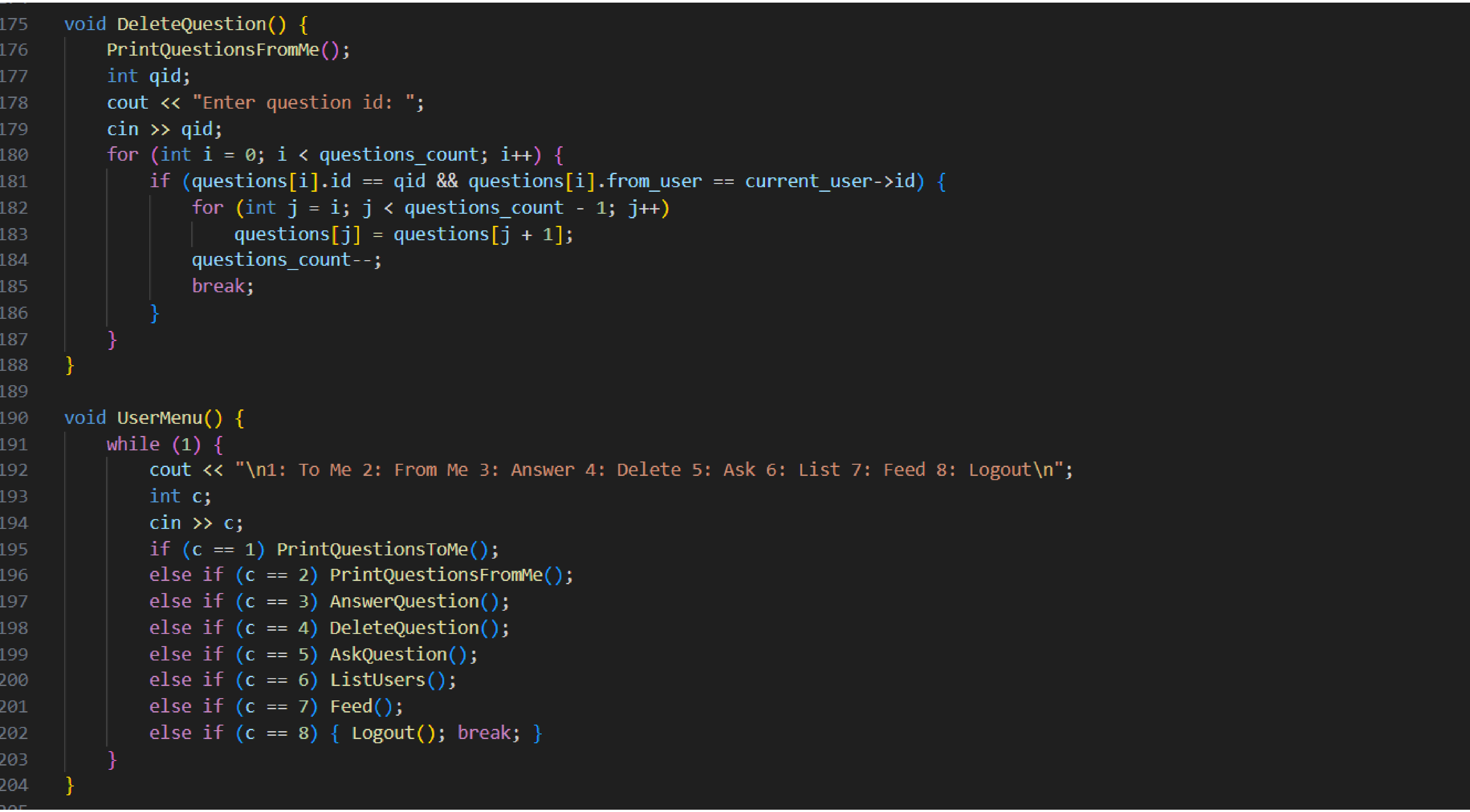
```cpp
void Signup() {
    string username;
    cout << "Enter username: ";
    cin >> username;
    if (FindUserByUsername(username)) {
        cout << "Username exists.\n";
        return;
    }
    users[users_count].id = NextUserId();
    users[users_count].username = username;
    cout << "Enter name: ";
    cin >> users[users_count].name;
    cout << "Enter email: ";
    cin >> users[users_count].email;
    cout << "Enter password: ";
    cin >> users[users_count].password;
    cout << "Allow anonymous? (1/0): ";
    cin >> users[users_count].allow_anonymous;
    users_count++;
}

int Login() {
    string username, password;
    cout << "Enter username: ";
    cin >> username;
    cout << "Enter password: ";
    cin >> password;
    User* u = FindUserByUsername(username);
    if (!u || u->password != password) {
        cout << "Invalid.\n";
        return 0;
    }
    current_user = u;
    return 1;
}
```



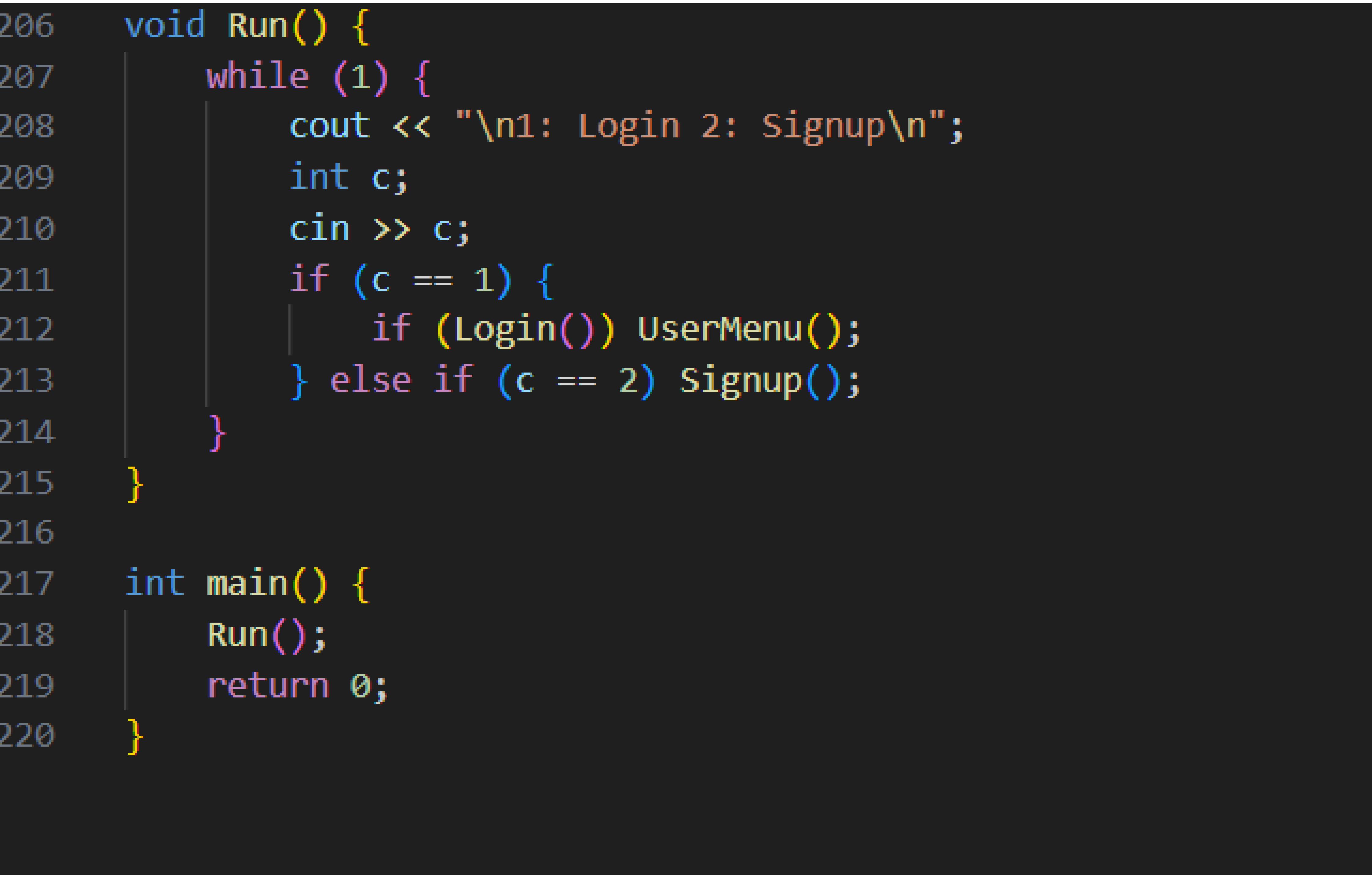
```cpp
void Logout() {
    current_user = NULL;
}

void ListUsers() {
    for (int i = 0; i < users_count; i++) {
        cout << users[i].id << ": " << users[i].username << " (" << users[i].name << ")\n";
    }
}

void PrintQuestionsToMe() {
    for (int i = 0; i < questions_count; i++) {
        if (questions[i].to_user == current_user->id) {
            cout << questions[i].id << ": " << questions[i].text;
            if (!questions[i].answer.empty()) cout << " => " << questions[i].answer;
            cout << "\n";
        }
    }
}
void PrintQuestionsFromMe() {
    for (int i = 0; i < questions_count; i++) {
        if (questions[i].from_user == current_user->id) {
            cout << questions[i].id << ": " << questions[i].text;
            if (!questions[i].answer.empty()) cout << " => " << questions[i].answer;
            cout << "\n";
        }
    }
}
void Feed() {
    for (int i = 0; i < questions_count; i++) {
        cout << questions[i].id << ": ";
        if (questions[i].from_user == -1) cout << "Anonymous";
        else cout << FindUserById(questions[i].from_user)->username;
        cout << " -> " << FindUserById(questions[i].to_user)->username << ": " << questions[i].text;
        if (!questions[i].answer.empty()) cout << " => " << questions[i].answer;
        cout << "\n";
    }
```



```cpp
void AskQuestion() {
    ListUsers();
    int to_id, anon = 0;
    cout << "Enter user id: ";
    cin >> to_id;
    User* to = FindUserById(to_id);
    if (!to) return;
    if (to->allow_anonymous) {
        cout << "Anonymous? (1/0): ";
        cin >> anon;
    }
    questions[questions_count].id = NextQuestionId();
    questions[questions_count].from_user = anon ? -1 : current_user->id;
    questions[questions_count].to_user = to_id;
    questions[questions_count].parent_id = 0;
    cout << "Enter question: ";
    cin.ignore();
    getline(cin, questions[questions_count].text);
    questions[questions_count].answer = "";
    questions_count++;
}

void AnswerQuestion() {
    PrintQuestionsToMe();
    int qid;
    cout << "Enter question id: ";
    cin >> qid;
    Question* q = FindQuestionById(qid);
    if (!q || q->to_user != current_user->id) return;
    cout << "Enter answer: ";
    cin.ignore();
    getline(cin, q->answer);
}
```



```cpp
void DeleteQuestion() {
    PrintQuestionsFromMe();
    int qid;
    cout << "Enter question id: ";
    cin >> qid;
    for (int i = 0; i < questions_count; i++) {
        if (questions[i].id == qid && questions[i].from_user == current_user->id) {
            for (int j = i; j < questions_count - 1; j++)
                questions[j] = questions[j + 1];
            questions_count--;
            break;
        }
    }
}

void UserMenu() {
    while (1) {
        cout << "\n1: To Me 2: From Me 3: Answer 4: Delete 5: Ask 6: List 7: Feed 8: Logout\n";
        int c;
        cin >> c;
        if (c == 1) PrintQuestionsToMe();
        else if (c == 2) PrintQuestionsFromMe();
        else if (c == 3) AnswerQuestion();
        else if (c == 4) DeleteQuestion();
        else if (c == 5) AskQuestion();
        else if (c == 6) ListUsers();
        else if (c == 7) Feed();
        else if (c == 8) { Logout(); break; }
    }
}
```



```cpp
void Run() {
    while (1) {
        cout << "\n1: Login 2: Signup\n";
        int c;
        cin >> c;
        if (c == 1) {
            if (Login()) UserMenu();
        } else if (c == 2) Signup();
    }
}

int main() {
    Run();
    return 0;
}
```