

yahia.cpp > Specialization

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  const int MAX_SPECIALIZATION = 20;
7  const int MAX_QUEUE = 5;
8
9  class Patient {
10 public:
11     string name;
12     int status;
13
14     Patient(string n = "", int s = 0) : name(n), status(s) {}
15 };
16
17 class Specialization {
18 private:
19     Patient queue[MAX_QUEUE];
20     int count;
21
22 public:
23     Specialization() : count(0) {}
24
25     bool addPatient(const string& name, int status) {
26         if (count >= MAX_QUEUE) {
27             cout << "Sorry, no available slots in this specialization.\n";
28             return false;
29         }
30
31         if (status == 1) {
32             for (int i = count; i > 0; --i) {
33                 queue[i] = queue[i - 1];
34             }
35             queue[0] = Patient(name, status);
36         } else {
37             queue[count] = Patient(name, status);
38         }
39     }
40 }
```

```
1 // yahia.cpp
2 #include <iostream>
3 #include <string>
4
5 using namespace std;
6
7 const int MAX_SPECIALIZATION = 20;
8 const int MAX_QUEUE = 5;
9
10 class Patient {
11 public:
12     string name;
13     int status;
14
15     Patient(string n = "", int s = 0) : name(n), status(s) {}
16 };
17
18 class Specialization {
19 private:
20     Patient queue[MAX_QUEUE];
21     int count;
22
23 public:
24     Specialization() : count(0) {}
25
26     bool addPatient(const string& name, int status) {
27         if (count >= MAX_QUEUE) {
28             cout << "Sorry, no available slots in this specialization.\n";
29             return false;
30         }
31
32         if (status == 1) {
33             for (int i = count; i > 0; --i) {
34                 queue[i] = queue[i - 1];
35             }
36             queue[0] = Patient(name, status);
37         } else {
38             queue[count] = Patient(name, status);
39         }
40     }
41 }
```

```

34     if (status == 1) {
35         queue[0] = Patient(name, status);
36     } else {
37         queue[count] = Patient(name, status);
38     }
39
40     count++;
41     cout << "Patient added successfully.\n";
42     return true;
43 }
44
45 void printPatients(int spec) const {
46     if (count == 0)
47         return;
48
49     cout << "Specialization " << spec << ":\n";
50     for (int i = 0; i < count; ++i) {
51         cout << "    [Spec " << spec << "] " << queue[i].name;
52         if (queue[i].status == 1)
53             cout << " (urgent)";
54         cout << "\n";
55     }
56 }
57
58 void getNextPatient() {
59     if (count == 0) {
60         cout << "No patients in this specialization.\n";
61         return;
62     }
63
64     cout << queue[0].name << ", please go with the doctor.\n";
65
66     for (int i = 1; i < count; ++i) {
67         queue[i - 1] = queue[i];
68     }

```

```

65
66     for (int i = 1; i < count; ++i) {
67         queue[i - 1] = queue[i];
68     }
69     count--;
70 }
71
72 bool hasPatients() const {
73     return count > 0;
74 }
75 };
76
77 class Hospital {
78 private:
79     Specialization specializations[MAX_SPECIALIZATION + 1];
80
81 public:
82     void addPatient() {
83         int spec, status;
84         string name;
85
86         cout << "Enter specialization (1-20): ";
87         cin >> spec;
88
89         if (spec < 1 || spec > MAX_SPECIALIZATION) {
90             cout << "Invalid specialization.\n";
91             return;
92         }
93
94         cout << "Enter patient name: ";
95         cin >> name;
96         cout << "Enter status (0 = regular, 1 = urgent): ";
97         cin >> status;
98
99         specializations[spec].addPatient(name, status);

```

```

99     specializations[spec].addPatient(name, status);
100 }
101
102 void printAllPatients() const {
103     for (int spec = 1; spec <= MAX_SPECIALIZATION; ++spec) {
104         specializations[spec].printPatients(spec);
105     }
106 }
107
108 void getNextPatient() {
109     int spec;
110
111     cout << "Enter specialization (1-20): ";
112     cin >> spec;
113
114     if (spec < 1 || spec > MAX_SPECIALIZATION) {
115         cout << "Invalid specialization.\n";
116         return;
117     }
118
119     specializations[spec].getNextPatient();
120 }
121
122 void menu() {
123     int choice;
124
125     while (true) {
126         cout << "\nEnter your choice:\n";
127         cout << "1. Add new patient\n";
128         cout << "2. Print all patients\n";
129         cout << "3. Get new patient\n";
130         cout << "4. Exit\n";
131         cout << "Your choice: ";
132         cin >> choice;
133

```

```

132     cin >> choice;
133
134     switch (choice) {
135     case 1:
136         addPatient();
137         break;
138     case 2:
139         printAllPatients();
140         break;
141     case 3:
142         getNextPatient();
143         break;
144     case 4:
145         cout << "Exiting the program.\n";
146         return;
147     default:
148         cout << "Invalid choice. Please try again.\n";
149     }
150 }
151 }
152 };
153
154 int main() {
155     Hospital h;
156     h.menu();
157
158     return 0;
159 }
160

```

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  const int max_QN_Book = 100;
6  const int max_users = 100;
7
8  class book {
9  public:
10     int count = 0;
11     int Id[max_QN_Book];
12     string Name[max_QN_Book];
13     int QN[max_QN_Book];
14
15     book() {}
16
17     void add_book() {
18         cout << "Enter book info : Id & Name & total quantity :" << endl;
19         cin >> Id[count] >> Name[count] >> QN[count];
20         cout << "Add book successful" << endl;
21         count++;
22     }
23
24     void search_book_by_prefix(const string& prefix) {
25         bool found = false;
26         for (int i = 0; i < count; ++i) {
27             if (Name[i].find(prefix) == 0) {
28                 cout << "Book found: " << Name[i] << " (Id: " << Id[i] << ", QN: " << QN[i] << ")" << endl;
29                 found = true;
30             }
31         }
32         if (!found) cout << "No book found with this prefix." << endl;
33     }

```

```

34
35 void print_library_by_id() {
36     for (int i = 0; i < count; ++i) {
37         cout << "Id: " << Id[i] << ", Name: " << Name[i] << ", QN: " << QN[i] << endl;
38     }
39 }
40
41 void print_library_by_name() {
42     for (int i = 0; i < count - 1; ++i) {
43         for (int j = i + 1; j < count; ++j) {
44             if (Name[i] > Name[j]) {
45                 swap(Name[i], Name[j]);
46                 swap(Id[i], Id[j]);
47                 swap(QN[i], QN[j]);
48             }
49         }
50     }
51     print_library_by_id();
52 }
53
54 int find_book_by_name(const string& name) {
55     for (int i = 0; i < count; ++i) {
56         if (Name[i] == name)
57             return i;
58     }
59     return -1;
60 }
61 };

```



```
53 class user {
54 public:
55     int count = 0;
56     string Name[max_users];
57     int borrowed_books[max_users][max_QN_Book] = {0};
58
59     void add_user() {
60         cout << "Enter user name: ";
61         cin >> Name[count];
62         cout << "User added successfully." << endl;
63         count++;
64     }
65
66     int find_user_by_name(const string& name) {
67         for (int i = 0; i < count; ++i) {
68             if (Name[i] == name)
69                 return i;
70         }
71         return -1;
72     }
73
74     void print_users() {
75         for (int i = 0; i < count; ++i) {
76             cout << "User: " << Name[i] << endl;
77         }
78     }
79 };
80
```



```
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;
    }
    void user_borrow_book() {
```

```
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;  
}
```

```
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;  
}
```

```

167 void menu() {
168     int choice;
169     while (true) {
170         cout << "\nLibrary Menu:" << endl;
171         cout << "1) add_book" << endl;
172         cout << "2) Search_book_by_prefix" << endl;
173         cout << "3) print_who_borrowed_book_by_name" << endl;
174         cout << "4) print_library_by_id" << endl;
175         cout << "5) print_library_by_name" << endl;
176         cout << "6) add_user" << endl;
177         cout << "7) user_borrow_book" << endl;
178         cout << "8) user_return_book" << endl;
179         cout << "9) print_users" << endl;
180         cout << "10) Exit" << endl;
181         cout << "Enter your choice number [1 : 10] : ";
182         cin >> choice;
183         if (choice == 1) books.add_book();
184         else if (choice == 2) {
185             string prefix;
186             cout << "Enter prefix: ";
187             cin >> prefix;
188             books.search_book_by_prefix(prefix);
189         }
190         else if (choice == 3) print_who_borrowed_book_by_name();
191         else if (choice == 4) books.print_library_by_id();
192         else if (choice == 5) books.print_library_by_name();
193         else if (choice == 6) users.add_user();
194         else if (choice == 7) user_borrow_book();
195         else if (choice == 8) user_return_book();
196         else if (choice == 9) users.print_users();
197         else if (choice == 10) break;
198         else cout << "Invalid choice, try again." << endl;
199     }
200 }
201 };

```

```
200     }
201 };|
202 int main() {
203     library lib;
204     lib.menu();
205     return 0;
206 }
```

yahia.cpp > [e] MAX_QUESTIONS

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  const int MAX_USERS = 100;
5  const int MAX_QUESTIONS = 1000;
6  struct User {
7      int id;
8      string username, name, email, password;
9      int allow_anonymous;
10 };
11 struct Question {
12     int id;
13     int from_user, to_user;
14     int parent_id;
15     string text, answer;
16 };
17
18 User users[MAX_USERS];
19 int users_count = 0;
20
21 Question questions[MAX_QUESTIONS];
22 int questions_count = 0;
23
24 User* current_user = NULL;
25
26 User* FindUserByUsername(string username) {
27     for (int i = 0; i < users_count; i++) {
28         if (users[i].username == username)
29             return &users[i];
30     }
31     return NULL;
32 }
```

```
4 User* FindUserById(int id) {
5     for (int i = 0; i < users_count; i++) {
6         if (users[i].id == id)
7             return &users[i];
8     }
9     return NULL;
10 }

1
2 Question* FindQuestionById(int id) {
3     for (int i = 0; i < questions_count; i++) {
4         if (questions[i].id == id)
5             return &questions[i];
6     }
7     return NULL;
8 }

9
10 int NextUserId() {
11     int mx = 0;
12     for (int i = 0; i < users_count; i++) {
13         if (users[i].id > mx) mx = users[i].id;
14     }
15     return mx + 1;
16 }

17
18 int NextQuestionId() {
19     int mx = 0;
20     for (int i = 0; i < questions_count; i++) {
21         if (questions[i].id > mx) mx = questions[i].id;
22     }
23     return mx + 1;
24 }

25
```



```
66 void Signup() {
67     string username;
68     cout << "Enter username: ";
69     cin >> username;
70     if (FindUserByUsername(username)) {
71         cout << "Username exists.\n";
72         return;
73     }
74     users[users_count].id = NextUserId();
75     users[users_count].username = username;
76     cout << "Enter name: ";
77     cin >> users[users_count].name;
78     cout << "Enter email: ";
79     cin >> users[users_count].email;
80     cout << "Enter password: ";
81     cin >> users[users_count].password;
82     cout << "Allow anonymous? (1/0): ";
83     cin >> users[users_count].allow_anonymous;
84     users_count++;
85 }
86
87 int Login() {
88     string username, password;
89     cout << "Enter username: ";
90     cin >> username;
91     cout << "Enter password: ";
92     cin >> password;
93     User* u = FindUserByUsername(username);
94     if (!u || u->password != password) {
95         cout << "Invalid.\n";
96         return 0;
97     }
98     current_user = u;
99     return 1;
100 }
```

```
02 void Logout() {
03     current_user = NULL;
04 }
05
06 void ListUsers() {
07     for (int i = 0; i < users_count; i++) {
08         cout << users[i].id << ": " << users[i].username << " (" << users[i].name << ")\n";
09     }
10 }
11
12 void PrintQuestionsToMe() {
13     for (int i = 0; i < questions_count; i++) {
14         if (questions[i].to_user == current_user->id) {
15             cout << questions[i].id << ": " << questions[i].text;
16             if (!questions[i].answer.empty()) cout << " => " << questions[i].answer;
17             cout << "\n";
18         }
19     }
20 }
21 void PrintQuestionsFromMe() {
22     for (int i = 0; i < questions_count; i++) {
23         if (questions[i].from_user == current_user->id) {
24             cout << questions[i].id << ": " << questions[i].text;
25             if (!questions[i].answer.empty()) cout << " => " << questions[i].answer;
26             cout << "\n";
27         }
28     }
29 }
30 void Feed() {
31     for (int i = 0; i < questions_count; i++) {
32         cout << questions[i].id << ": ";
33         if (questions[i].from_user == -1) cout << "Anonymous";
34         else cout << FindUserById(questions[i].from_user)->username;
35         cout << " -> " << FindUserById(questions[i].to_user)->username << ": " << questions[i].text;
36         if (!questions[i].answer.empty()) cout << " => " << questions[i].answer;
37         cout << "\n";
38     }
```

```

140
141 void AskQuestion() {
142     ListUsers();
143     int to_id, anon = 0;
144     cout << "Enter user id: ";
145     cin >> to_id;
146     User* to = FindUserById(to_id);
147     if (!to) return;
148     if (to->allow_anonymous) {
149         cout << "Anonymous? (1/0): ";
150         cin >> anon;
151     }
152     questions[questions_count].id = NextQuestionId();
153     questions[questions_count].from_user = anon ? -1 : current_user->id;
154     questions[questions_count].to_user = to_id;
155     questions[questions_count].parent_id = 0;
156     cout << "Enter question: ";
157     cin.ignore();
158     getline(cin, questions[questions_count].text);
159     questions[questions_count].answer = "";
160     questions_count++;
161 }
162
163 void AnswerQuestion() {
164     PrintQuestionsToMe();
165     int qid;
166     cout << "Enter question id: ";
167     cin >> qid;
168     Question* q = FindQuestionById(qid);
169     if (!q || q->to_user != current_user->id) return;
170     cout << "Enter answer: ";
171     cin.ignore();
172     getline(cin, q->answer);
173 }
174

```

```

175 void DeleteQuestion() {
176     PrintQuestionsFromMe();
177     int qid;
178     cout << "Enter question id: ";
179     cin >> qid;
180     for (int i = 0; i < questions_count; i++) {
181         if (questions[i].id == qid && questions[i].from_user == current_user->id) {
182             for (int j = i; j < questions_count - 1; j++)
183                 questions[j] = questions[j + 1];
184             questions_count--;
185             break;
186         }
187     }
188 }
189
190 void UserMenu() {
191     while (1) {
192         cout << "\n1: To Me 2: From Me 3: Answer 4: Delete 5: Ask 6: List 7: Feed 8: Logout\n";
193         int c;
194         cin >> c;
195         if (c == 1) PrintQuestionsToMe();
196         else if (c == 2) PrintQuestionsFromMe();
197         else if (c == 3) AnswerQuestion();
198         else if (c == 4) DeleteQuestion();
199         else if (c == 5) AskQuestion();
200         else if (c == 6) ListUsers();
201         else if (c == 7) Feed();
202         else if (c == 8) { Logout(); break; }
203     }
204 }
205

```

```
206 void Run() {
207     while (1) {
208         cout << "\n1: Login 2: Signup\n";
209         int c;
210         cin >> c;
211         if (c == 1) {
212             if (Login()) UserMenu();
213         } else if (c == 2) Signup();
214     }
215 }
216
217 int main() {
218     Run();
219     return 0;
220 }
```