به نام خدا

پروژه دانشجویی ساختمان داده:

Library Management System

سیستم مدیریت کتابخانه

مدیریت کتاب‌ها، امانت‌ها و تاریخچه درخواست‌ها

فاطمه نبی‌قنبری

1404/10

مقدمه:

این پروژه یک سیستم کامل مدیریت کتابخانه است.

عملیات اصلی:

1. ذخیره سازی کتاب (افزودن، حذف، جستجو بر اساس کد)
2. مرتب‌سازی کتاب (بر اساس کد کتاب و سال انتشار)
3. مدیریت امانات کتاب (درخواست امانت کتاب به نوبت، پردازش درخواست‌ها به ترتیب ورود، اجرای درخواست بعدی در صورت بازگشت کتاب)
4. تاریخچه عملیات (امکان نمایش آخرین عملیات انجام شده)

اهداف:

1. مدیریت دقیق کتاب‌ها و اطلاعات آن‌ها (کد، عنوان، نویسنده، سال و وضعیت).
2. پیاده‌سازی صف درخواست‌ها برای کتاب‌های امانت داده شده.
3. فراهم کردن امکان Undo و Redo برای همه عملیات اصلی.
4. ثبت تاریخچه عملیات ((History و گزارش کامل وضعیت کتابخانه.
5. ارائه آمار دقیق کتاب‌ها، موجودی و صف‌ها.

مدل‌سازی کتابخانه:

ساختار کتاب (Book) :

- کد (code)، عنوان (title)، نویسنده (author)، سال انتشار (year)، موجود بودن (available)
- صف درخواست‌ها (Request Queue) : مدیریت کاربران منتظر، FIFO
- متدهای کمکی:
- isQueueEmpty() – بررسی خالی بودن صف
- isQueueFull() – بررسی پر بودن صف
- addToQueue(name) – افزودن کاربر به صف
- getFromQueue() – برداشتن کاربر از صف

عملیات اصلی کتابخانه:

- Add / Remove / Borrow / Return / Sort
- Undo و Redo برای برگشت یا اعمال مجدد عملیات‌ها

ساختار کتابخانه (Library) :

- آرایه کتاب‌ها (Book[]) + bookCount
- Undo / Redo Stack برای پیگیری تغییرات
- History برای ذخیره آخرین عملیات‌ها
- آمار کتابخانه: تعداد کل، کتاب‌های موجود، امانت داده شده، تعداد درخواست‌ها، ظرفیت باقیمانده

توضیحات کد مدیریت کتابخانه:

ساختار داده

```
struct Book {
    int code;
    string title;
    string author;
    int year;
    bool available;

    string requestQueue[20];
    int front = 0;
    int rear = -1;

    bool isQueueEmpty() { return front > rear; }
    bool isQueueFull() { return rear >= 19; }
    void addToQueue(string name) {
        if(!isQueueFull()) requestQueue[++rear] = name;
    }
    string getFromQueue() {
        return !isQueueEmpty() ? requestQueue[front++] : "";
    }
    int queueSize() {
        return rear - front + 1;
    }
};

enum ActionType {
    ADD_BOOK,
    DELETE_BOOK,
    BORROW_BOOK,
    RETURN_BOOK,
    SORT_BOOKS
};

struct Action {
    ActionType type;
    Book book;
    string description;
};
```

این بخش ساختار اصلی داده‌ها را تعریف می‌کند.

Book اطلاعات کتاب، وضعیت موجود بودن و صف درخواست‌ها را نگهداری می‌کند.

Action و ActionType برای پیاده‌سازی Undo/Redo و ذخیره تاریخچه عملیات استفاده می‌شوند.

متدهای داخل Book کمک می‌کنند صف درخواست‌ها را به صورت FIFO مدیریت کنیم.

ذخیره‌سازی و تاریخچه

```
45    Book books[200];
46    int bookCount = 0;
47
48    Action undoStack[10];
49    Action redoStack[10];
50    int undoTop = -1;
51    int redoTop = -1;
52
53    string history[10];
54    int historyCount = 0;
55
```

آرایه []books کتاب‌ها را نگه می‌دارد و bookCount تعداد کل کتاب‌ها را مشخص می‌کند.

undoStack و redoStack برای برگشت یا تکرار تغییرات کتابخانه استفاده می‌شوند.

history برای ثبت آخرین عملیات‌ها و ارائه گزارش کوتاه کاربرد دارد

متدهای کمکی برای تاریخچه

```
56  void saveToHistory(string desc)

66  void saveForUndo(ActionType type, Book b, string desc)

80  int findBookByCode(int code)

86  void printBookInfo(Book b)

86  void printBookInfo(Book b)
```

متدها مدیریت تاریخچه عملیات و Undo/Redo را ساده و قابل استفاده می‌کنند.

findBookByCode برای جستجوی سریع کتاب با کد منحصر به فرد استفاده می‌شود.

printBookInfo نمایش حرفه‌ای اطلاعات کتاب است.

افزودن کتاب

```
98  void addBook() { .... }
```

اضافه کردن کتاب جدید با بررسی Duplicate .

به محض افزودن، عملیات ذخیره Undo انجام می‌شود تا امکان برگشت وجود داشته باشد.

حذف کتاب

```
131    void removeBook() { ... }
```

حذف کتاب با ذخیره عملیات در Undo Stack .

امکان بازگشت حذف با Undo وجود دارد.

جستجوی کتاب

```
153    void searchBook() { ... }
```

جستجوی کتاب با کد.

نمایش اطلاعات کامل کتاب و تعداد افراد در صف.

ذخیره عملیات در History برای گزارش کوتاه.

امانت کتاب

```
169    void borrowBook() { ... }
```

مدیریت امانت کتاب با بررسی موجود بودن.

اگر کتاب موجود نباشد، کاربر در صف انتظار قرار می‌گیرد.

ذخیره عملیات در Undo/Redo و History انجام می‌شود.

بازگشت کتاب

```
203    void returnBook() { ... }
```

اگر صف انتظار وجود داشته باشد، کتاب به نفر بعدی داده می‌شود.

در غیر این صورت، وضعیت کتاب به موجود بودن تغییر می‌کند.

امکان Undo و ذخیره عملیات در History وجود دارد.

مرتب‌سازی کتاب‌ها

```
227    void sortByCode() { ... }
247    void sortByYear() { ... }
```

جستجوی کتاب با کد.

نمایش اطلاعات کامل کتاب و تعداد افراد در صف.

ذخیره عملیات در History برای گزارش کوتاه.

نمایش کتاب‌ها، تاریخچه و آمار

```
267    void displayAllBooks() { ... }
340    void showHistory() { ... }
354    void showStatistics() { ... }
```

displayAllBooks : نمایش کامل کتاب‌ها با وضعیت و صف‌ها.

showHistory : نمایش آخرین عملیات‌ها با توضیح کوتاه.

showStatistics : تعداد کل، موجود، امانت داده شده و صف‌ها.

تکرار و برگشت عملیات

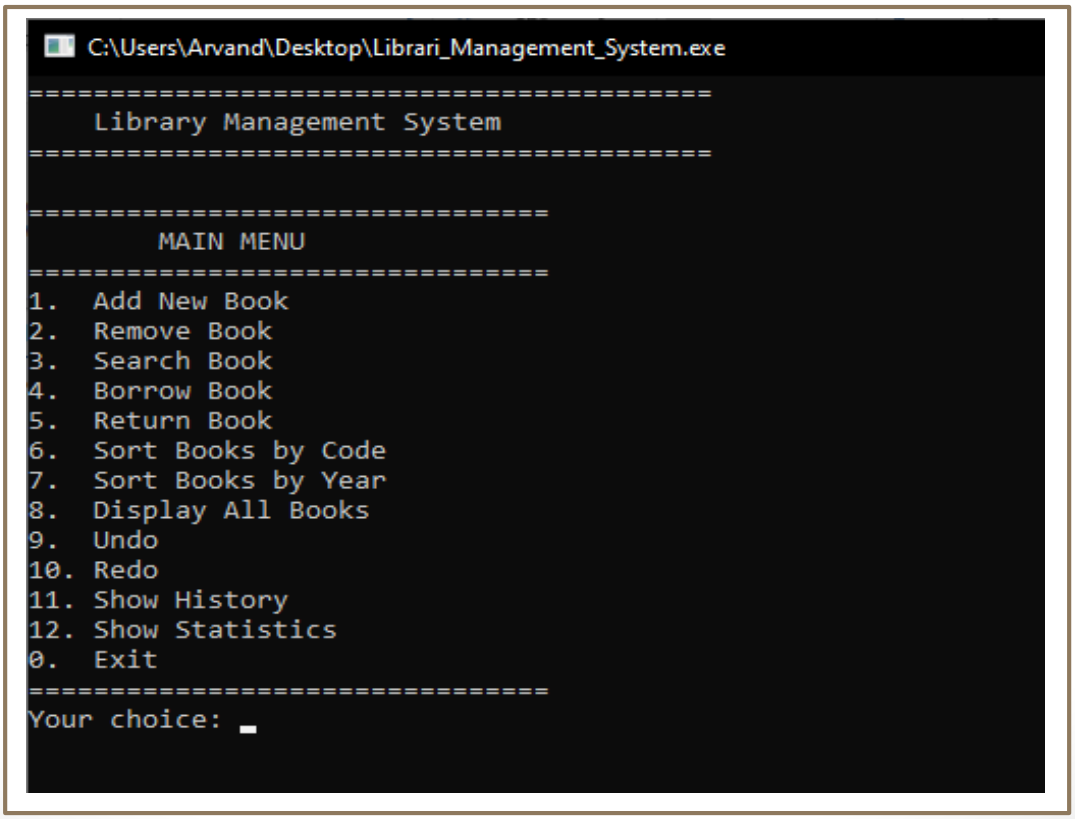
```
281  void undo() { ... }

329  void redo() { ... }
```

Undo : برگشت عملیات انجام شده (افزودن، حذف، امانت، بازگشت، مرتب‌سازی)

Redo : تکرار عملیات Undo شده

تمام تغییرات همزمان در History ثبت می‌شوند تا قابلیت ردیابی وجود داشته باشد.

خروجی نهایی



جمعبندی نهایی پروژه:

در این پروژه، یک سیستم مدیریت کتابخانه طراحی و پیاده‌سازی شده که هدف اصلی آن، استفاده عملی از مفاهیم ساختمان داده‌ها در قالب یک مسئله‌ی واقعی بوده است. تمرکز پروژه فقط روی انجام چند عملیات ساده نبوده، بلکه سعی شده مدیریت داده‌ها، وضعیت کتاب‌ها و ثبت تغییرات سیستم به شکل منطقی و اصولی انجام شود.

در طراحی این سیستم، برای نگهداری اطلاعات کتاب‌ها از آرایه استفاده شده تا دسترسی به داده‌ها ساده و سریع باشد. مدیریت درخواست‌های امانت با استفاده از صف انجام شده تا ترتیب درخواست‌ها به‌صورت عادلانه و بر اساس FIFO رعایت شود. همچنین قابلیت‌های Undo و Redo با استفاده از پشته پیاده‌سازی شده‌اند که امکان بازگشت به وضعیت‌های قبلی سیستم را فراهم می‌کند. طراحی ساختارهایی مانند Book و Action نیز باعث شده کد نهایی منظم و قابل توسعه باشد.

در مجموع، این پروژه نشان می‌دهد که مفاهیم پایه‌ای ساختمان داده‌ها می‌توانند به‌صورت عملی در یک سیستم واقعی به کار گرفته شوند و با طراحی مناسب، یک برنامه‌ی ساده اما کاربردی ایجاد شود.

این سیستم می‌تواند به‌عنوان پایه‌ای مناسب برای پیاده‌سازی نسخه‌های کامل‌تر و پیشرفته‌تر مورد استفاده قرار گیرد.