PP

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class LibraryItem {

protected:

string title;

string author;

public:

LibraryItem(string t, string a) : title(t), author(a) {}

// Virtual function for display (polymorphism)

virtual void display() const = 0; // Pure virtual function

};

// Derived class: Book

class Book : public LibraryItem {

int pages;

public:

Book(string t, string a, int p) : LibraryItem(t, a), pages(p) {}

void display() const override {

cout << "Book Title: " << title << ", Author: " << author << ", Pages: " << pages << endl;

}

};

// Derived class: Magazine

class Magazine : public LibraryItem {

int issueNumber;

public:

Magazine(string t, string a, int issue) : LibraryItem(t, a), issueNumber(issue) {}

void display() const override {

cout << "Magazine Title: " << title << ", Author: " << author << ", Issue Number: " << issueNumber << endl;

}

};

// Derived class: DVD

class DVD : public LibraryItem {

int duration;

public:

DVD(string t, string a, int d) : LibraryItem(t, a), duration(d) {}

void display() const override {

cout << "DVD Title: " << title << ", Director: " << author << ", Duration: " << duration << " minutes" << endl;

}

};

// Class to manage the Library

class Library {

vector<LibraryItem\*> items;

public:

// Function to add a new LibraryItem (Book, Magazine, or DVD)

void addItem() {

int choice;

cout << "\nSelect the type of item to add:\n";

cout << "1. Book\n2. Magazine\n3. DVD\n";

cout << "Enter choice: ";

cin >> choice;

string title, author;

cout << "Enter title: ";

cin.ignore();

getline(cin, title);

cout << "Enter author/director: ";

getline(cin, author);

if (choice == 1) {

int pages;

cout << "Enter number of pages: ";

cin >> pages;

items.push\_back(new Book(title, author, pages));

} else if (choice == 2) {

int issue;

cout << "Enter issue number: ";

cin >> issue;

items.push\_back(new Magazine(title, author, issue));

} else if (choice == 3) {

int duration;

cout << "Enter duration (in minutes): ";

cin >> duration;

items.push\_back(new DVD(title, author, duration));

} else {

cout << "Invalid choice. Please try again.\n";

}

}

// Function to display all items

void displayItems() const {

if (items.empty()) {

cout << "\nNo items in the library.\n";

return;

}

cout << "\nLibrary Items:\n";

for (const auto& item : items) {

item->display();

}

}

// Destructor to free allocated memory

~Library() {

for (auto& item : items) {

delete item;

}

}

};

int main() {

Library library;

int choice;

while (true) {

cout << "\nLibrary Management System\n";

cout << "1. Add Item\n2. Display All Items\n3. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

library.addItem();

break;

case 2:

library.displayItems();

break;

case 3:

cout << "Exiting...\n";

return 0;

default:

cout << "Invalid choice. Please try again.\n";

}

}

}