**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN**

**ĐỀ TÀI: QUẢN LÝ RẠP CHIẾU PHIM**

Giảng viên hướng dẫn: **Th.S TRẦN THỊ DUNG**

Sinh viên thực hiện: **NGUYỄN THỊ KIM LIÊN**

Lớp: **CÔNG NGHỆ THÔNG TIN**

Khóa: **62**

Thành phố Hồ Chí Minh, ngày 22 tháng 11 năm 2022

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN**

**ĐỀ TÀI: QUẢN LÝ RẠP CHIẾU PHIM**

Giảng viên hướng dẫn: **Th.S TRẦN THỊ DUNG**

Sinh viên thực hiện: **NGUYỄN THỊ KIM LIÊN**

Lớp: **CÔNG NGHỆ THÔNG TIN**

Khóa: **62**

Thành phố Hồ Chí Minh, ngày 22 tháng 11 năm 2022

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI  **PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH** | CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  Độc lập – Tự do – Hạnh phúc |

**BÀI TẬP LỚN**

**MÔN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

-----\*\*\*\*\*-----

|  |  |
| --- | --- |
| **MSSV: 6251071052** | **Họ và tên: NGUYỄN THỊ KIM LIÊN** |
| **Lớp: CQ.62.CNTT** | **Khóa: 62** |
| **MSSV: 6251071060** | **Họ và tên: NGUYỄN THÀNH LUÂN** |
| **Lớp: CQ.62.CNTT** | **Khóa: 62** |
| **MSSV: 6251071035** | **Họ và tên: VÕ GIA HUY** |
| **Lớp: CQ.62.CNTT** | **Khóa: 62** |
| **MSSV: 6251071098** | **Họ và tên: ĐINH THỊ KIM THỎA** |
| **Lớp: CQ.62.CNTT** | **Khóa: 62** |

1. **Tên đề tài: QUẢN LÝ RẠP CHIẾU PHIM**
2. **Mục đích, yêu cầu:**- Mục đích: Tạo ra chương trình “Quản lý rạp chiếu phim” dựa trên những nhu cầu thực tế của khách hàng và nhà quản lý nhằm giải quyết những khó khăn gặp phải, giảm thiểu rủi ro trong quá trình quản lý rạp. Đối tượng được hướng tới là khách hàng và những nhân viên quản lý trong rạp.  
   - Yêu cầu:   
    + Giao diện thân thiện.  
    + Dễ dàng sử dụng.  
    + Đủ các chức năng phục vụ cho việc quản lý.  
    + Dễ bảo trì.  
    + Dễ cập nhật công nghệ mới.
3. **Nội dung và phạm vi đề tài:**  
   - Nội dung đề tài: Trong các rạp chiếu phim hiện nay, việc quản lý và tổ chức chiếu phim, bán vé luôn là vấn đề được quan tâm. Người quản lý luôn gặp khó khăn trong vấn đề quản lý quá trình hoạt động của từng bộ phận như lịch chiếu, phim, trang thiết bị…. và đặc biệt là quá trình bán vé. Chính vì lý do đó mà số lượng khách hàng đi xem phim ít hơn, thu nhập của rạp cũng bị ảnh hưởng. Vì vậy, tạo ra một chương trình quản lý rạp chiếu phim là một nhu cầu tất yếu.  
   - Phạm vi đề tài: Công cụ quản lý hóa đơn có thể được sử dụng cho hầu hết tất cả các rạp phim hoặc mở rộng hơn trong nhiều lĩnh vực khác. Công cụ tập trung chính vào việc lưu trữ, quản lý các đơn hàng được bán ra, người mua vé, số lượng vé, tên phim,…
4. **Công cụ, công nghệ và ngôn ngữ lập trình:**  
   - Sử dụng phần mềm DevC++ và Vscode lập trình trên ngôn ngữ lập trình C++.  
   - Sử dụng Github để lưu trữ, quản lý source code.  
   - Sử dụng Access để lưu dữ liệu.
5. **Các kết quả chính dự kiến sẽ ứng dụng và đạt được:**- Tạo ra được một công cụ quản lý hóa đơn hoàn hảo, dễ dùng, dung lượng không quá nặng, hỗ trợ 100% việc lưu trữ xuất - nhập , giảm sức người và tăng hiệu quả đến mức tối đa.
6. **Giáo viên và cán bộ hướng dẫn:**

Họ tên: **TRẦN THỊ DUNG**

Đơn vị công tác: Bộ môn Công nghệ thông tin – Trường Đại học Giao thông Vận tải Phân hiệu tại thành phố Hồ Chí Minh.

Điện thoại:

Email:

|  |  |
| --- | --- |
| **Ngày … tháng … năm 2022** |  |
| **Trưởng Bộ môn Công nghệ thông tin** | **Giáo viên hướng dẫn** |
|  |  |
| **Th.S Trần Phong Nhã** | **Th.S Trần Thị Dung** |

# LỜI CẢM ƠN

Trước tiên với tình cảm sâu sắc và chân thành nhất, cho phép các thành viên của nhóm chúng em được bày tỏ sự biết ơn đến cô đã dành thời gian đọc bài báo cáo bài tập lớn này.

Bài tập này được cả nhóm hoàn thành dựa trên các kiến thức được học tại trường với sự hướng dẫn của cô Trần Thị Dung. Nhóm xin gửi lời cảm ơn đến cô vì đã tận tình giúp đỡ, giải đáp các thắc mắc để nhóm có thể hoàn thành tốt nhất đạt đúng mục tiêu của nhóm đề ra lúc đầu cho bài tập lần này.

Những bài tập này có ý nghĩa rất lớn đối với sinh viên cụ thể là đối với nhóm chúng em, nhờ có những bài tập như này mà chúng em có cơ hội được làm quen với teamwork, được học hỏi thêm nhiều kiến thức mới từ thầy cô, bạn bè, nâng cao trình độ chuyên ngành,.. Chúng em rất vui khi được cô giúp đỡ trong quá trình hoàn thành bài này. Một lần nữa nhóm chúng em xin cảm ơn cô Trần Thị Dung.

# NHẬN XÉT CỦA GIẢNG VIÊN

**Thành phố Hồ Chí Minh, ngày 22 tháng 11 năm 2022**

|  |
| --- |
| **Giảng viên hướng dẫn** |
| **Th.S TRẦN THỊ DUNG** |

# MỤC LỤC

[LỜI CẢM ƠN i](#_Toc119839154)

[NHẬN XÉT CỦA GIẢNG VIÊN ii](#_Toc119839155)

[MỤC LỤC iii](#_Toc119839156)

[DANH MỤC CHỮ VIẾT TẮT v](#_Toc119839157)

[DANH MỤC HÌNH ẢNH vi](#_Toc119839158)

[CHƯƠNG 1: DỊCH SÁCH 1](#_Toc119839159)

[1.1 CLASS 1](#_Toc119839160)

[*1.1.1 Giới thiệu và cơ bản về class:* 1](#_Toc119839161)

[1.1.2 Lớp và cấu trúc 2](#_Toc119839162)

[1.1.3 Hàm thành viên 2](#_Toc119839163)

[1.1.4 [static]Thành viên 4](#_Toc119839164)

[1.1.5 Nạp chồng toán tử 5](#_Toc119839165)

[1.2 HÀM KHỞI TẠO, XÓA, SAO CHÉP VÀ DI CHUYỂN 5](#_Toc119839166)

[1.2.1 Giới thiệu 5](#_Toc119839167)

[1.2.2 Hàm tạo và hàm hủy 5](#_Toc119839168)

[1.2.3 Sao chép và Di chuyển 7](#_Toc119839169)

[1.2.4 Các chức năng deleted 7](#_Toc119839170)

[CHƯƠNG 2: MỞ ĐẦU 8](#_Toc119839171)

[2.1. Mục đích: 8](#_Toc119839172)

[2.2. Mục tiêu và nhiệm vụ nghiên cứu: 8](#_Toc119839173)

[2.3. Đối tượng và phạm vi nghiên cứu: 8](#_Toc119839174)

[2.4. Cách tiếp cận và phương pháp nghiên cứu: 8](#_Toc119839175)

[CHƯƠNG 3: CƠ SỞ LÝ THUYẾT 9](#_Toc119839176)

[3.1. Tổng quan ngôn ngữ lập trình C++ 9](#_Toc119839177)

[*3.1.1. Khái niệm:* 9](#_Toc119839178)

[*3.1.2. Các đặc điểm nổi bật* 9](#_Toc119839179)

[*3.1.3. Lịch sử phát triển:* 9](#_Toc119839180)

[*3.1.4. Ưu nhược điểm của ngôn ngữ c++* 10](#_Toc119839181)

[3.2. Tổng quan GITHUB 11](#_Toc119839182)

[*3.2.1. Khái niệm:* 11](#_Toc119839183)

[*3.2.2. Lịch sử:* 11](#_Toc119839184)

[*3.2.3. Các đặc điểm nổi bật* 11](#_Toc119839185)

[CHƯƠNG 4: TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG 14](#_Toc119839186)

[4.1. Lập trình hướng đối tượng là gì? 14](#_Toc119839187)

[4.2. Các nguyên lý cơ bản của OOP 14](#_Toc119839188)

[*4.2.1 Tính đóng gói (Encapsulation)* 14](#_Toc119839189)

[*4.2.2. Tính kế thừa (Inheritance)* 14](#_Toc119839190)

[*4.2.3. Tính đa hình (Polymorphism)* 14](#_Toc119839191)

[*4.2.4. Tính trừu tượng (Abstraction)* 14](#_Toc119839192)

[4.3. Các ưu điểm của lập trình hướng đối tượng 14](#_Toc119839193)

[CHƯƠNG 5: PHÂN TÍCH HỆ THỐNG 16](#_Toc119839194)

[5.1. Thư viện 16](#_Toc119839195)

[5.2. Cơ sở dữ liệu 16](#_Toc119839196)

[5.3. Ứng dụng lập trình hướng đối tượng 16](#_Toc119839197)

[*5.3.1. Tính đóng gói* 16](#_Toc119839198)

[*5.3.2. Tính kế thừa* 16](#_Toc119839199)

[5.4. Phân tích hệ thống 17](#_Toc119839200)

[*5.4.1. Đăng nhập* 17](#_Toc119839201)

[*5.4.2. Form main* 18](#_Toc119839202)

[*5.4.3. Quản lý phim* 18](#_Toc119839203)

[*5.4.4. Quản lý lịch chiếu* 20](#_Toc119839204)

[*5.4.5. Đăng ký vé* 23](#_Toc119839205)

[*5.4.6. Quản lý hóa đơn* 27](#_Toc119839206)

[*5.4.7. Doanh thu* 28](#_Toc119839207)

[*5.4.8. Đăng xuất* 29](#_Toc119839208)

[*5.4.9. Nút Thoát* 29](#_Toc119839209)

[TÀI LIỆU THAM KHẢO 30](#_Toc119839210)

[LINK GITHUB 31](#_Toc119839211)

# DANH MỤC CHỮ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Ý nghĩa** | **Mô tả** | **Ghi chú** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# DANH MỤC HÌNH ẢNH

[Hình 2.1 Repository 12](#_Toc119790525)

[Hình 2.2 Branch 12](#_Toc119790526)

[Hình 2.3 Merge 13](#_Toc119790527)

[Hình 5.1 Tính đóng gói 3](#_Toc119837700)

[Hình 5.2 Tính kế thừa 4](#_Toc119837701)

[Hình 5.3 Login 4](#_Toc119837702)

[Hình 5.4 Login ở Access 4](#_Toc119837703)

[Hình 5.5 Đăng nhập sai 5](#_Toc119837704)

[Hình 5.6 Form Main 5](#_Toc119837705)

[Hình 5.7 Quản lý phim 6](#_Toc119837706)

[Hình 5.8 Quản lý phim ở access 6](#_Toc119837707)

[Hình 5.9 Thêm phim mới 7](#_Toc119837708)

[Hình 5.10 Sửa phim 7](#_Toc119837709)

[Hình 5.11 Lịch Phim 8](#_Toc119837710)

[Hình 5.12 Lịch phim ở Access 8](#_Toc119837711)

[Hình 5.13 Thêm lịch chiếu 9](#_Toc119837712)

[Hình 5.14 Thông báo thiếu thông tin lịch chiếu 9](#_Toc119837713)

[Hình 5.15 Sửa lịch chiếu 10](#_Toc119837714)

[Hình 5.16 Xác nhận xóa 1 lịch phim 10](#_Toc119837715)

[Hình 5.17 Đăng ký vé 11](#_Toc119837716)

[Hình 5.18 Sơ đồ chỗ ngồi 11](#_Toc119837717)

[Hình 5.19 Thông tin đăng ký vé 12](#_Toc119837718)

[Hình 5.20 Thông báo nhập sai định dạng số điện thoại 12](#_Toc119837719)

[Hình 5.21 Hóa đơn mua vé 13](#_Toc119837720)

[Hình 5.22 Vị trí ghế đã mua 13](#_Toc119837721)

[Hình 5.23 Quản lý hóa đơn 14](#_Toc119837722)

[Hình 5.24 Tìm kiếm thông tin hóa đơn 14](#_Toc119837723)

[Hình 5.25 Thông báo nhập sai thời gian 15](#_Toc119837724)

[Hình 5.26 In lại hóa đơn 15](#_Toc119837725)

[Hình 5.27 Thống kê doanh thu 16](#_Toc119837726)

# 

# CHƯƠNG 1: DỊCH SÁCH

## 1.1 CLASS

### *1.1.1 Giới thiệu và cơ bản về class:*

Các class C ++ là một công cụ để tạo các kiểu dữ liệu mới có thể được sử dụng thuận tiện như các kiểu dữ liệu tích hợp sẵn.Dưới đây là một bản tóm tắt ngắn gọn về class:

• Một lớp là một kiểu dữ liệu do người dùng định nghĩa.

• Một lớp bao gồm một tập hợp các thành viên. Các loại thành viên phổ biến nhất là thành phần dữ liệu và hàm thành viên.

• Các hàm thành viên có thể xác định ý nghĩa của việc khởi tạo (tạo), sao chép, di chuyển và dọn dẹp ( tiêu hủy)

• Các thành viên được truy cập bằng cách sử dụng (dấu chấm) cho các đối tượng và > (mũi tên) cho con trỏ.

• Các toán tử chẳng hạn như + , ! , Và [], có thể được định nghĩa cho một lớp.

• Một lớp là một không gian chứa các thành viên của nó.

• Các thành phần công khai cung cấp giao diện của lớp và các thành phần riêng cung cấp chi tiết triển khai.

• Một cấu trúc là một lớp mà các thành viên được mặc định là công khai.

|  |
| --- |
| class X {  private: // Đại diên (thực hiện ) là riêng tư  int m;  public: // Giao diện người dùng là công khai  X(int i =0) :m{i} { } // Hàm tạo (khởi tạo dữ liệu cho m)  int mf(int i) // Hàm thành viên  {  int old = m;  m = i; // đặt một giá trị mới  return old; // trả về giá trị cũ  }  };  X var {7}; // một biến kiểu X, được khởi tạo bằng 7  int user(X var, X∗ ptr)  {  int x = var.mf(7); // truy cập bằng “.”  int y = ptr−>mf(9); // truy cập bằng “->”  int z = var.m; //Lỗi : không thể truy cập thành viên trong riêng tư  } |

### 1.1.2 Lớp và cấu trúc

Khởi tạo class X {...}; được gọi là định nghĩa lớp; nó xác định 1 kiểu gọi là X. Ngày xưa, định nghĩa lớp thường được gọi là khai báo lớp. Khai báo thì cũng không có định nghĩa, định nghĩa lớp có thể được sao chép trong các tệp nguồn khác nhau sử dụng #include mà không vi phạm quy tắc 1 định nghĩa. Theo định nghĩa, một cấu trúc là một lớp trong đó các thành viên được mặc định là public. Ví dụ: struct S {/ \* ... \* /}; chỉ đơn giản là viết tắt của class S {public: / \* ... \* /}; Hai định nghĩa này của S có thể thay thế cho nhau, mặc dù thông thường bạn nên theo một kiểu nhất định. Phong cách bạn sử dụng tùy thuộc vào hoàn cảnh và sở thích. Tôi có xu hướng sử dụng cấu trúc cho các lớp mà tôi nghĩ là "" chỉ là cấu trúc dữ liệu đơn giản. "" Nếu tôi nghĩ về một lớp là "" một loại thích hợp với một bất biến, "tôi sử dụng lớp. Các hàm tạo và hàm truy cập có thể khá hữu ích ngay cả đối với các cấu trúc, nhưng như một cách viết tắt chứ không phải là bảo đảm bất biến. Theo mặc định, các thành viên của một lớp là riêng tư. Không bắt buộc phải khai báo dữ liệu đầu tiên trong một lớp. Trên thực tế, việc đặt các thành phần dữ liệu cuối cùng thường nhấn mạnh các chức năng cung cấp giao diện người dùng công khai. Các chỉ định truy cập có thể được sử dụng nhiều lần trong một khai báo lớp duy nhất.

### 1.1.3 Hàm thành viên

Xem xét việc triển khai khái niệm ngày bằng cách sử dụng struct để xác định biểu diễn của Date và một tập hợp các hàm để thao tác với các biến kiểu này:

|  |
| --- |
| struct Date{  int d,m,y;  };  void init\_date(Date& d,int,int,int); //hàm khởi tạo  void add\_year(Date& d,int n); // hàm thêm n năm vào d |

Không có kết nối rõ ràng nào giữa kiểu dữ liệu, Date và các hàm này. Một kết nối kiểu này có thể được thiết lập bằng cách khai báo các chức năng là thành viên:

|  |
| --- |
| struct Date{  int d,m,y;  void init\_date(Date& d,int,int,int); //hàm khởi tạo  void add\_year(Date& d,int n); // hàm thêm n năm vào d }; |

Các hàm được khai báo trong định nghĩa lớp được gọi là các hàm thành viên và chỉ có thể được gọi cho một biến cụ thể thuộc loại thích hợp bằng cách sử dụng tiêu chuẩn cú pháp để truy cập thành viên cấu trúc. Các cấu trúc khác nhau có thể có nhiều hàm thành viên cùng tên. chúng ta phải phân biệt tên cấu trúc khi xác định hàm thành viên Trong 1 hàm thành viên, tên thành viên có thể được sử dụng mà không cần có sự tham khảo rõ ràng đến đối tượng. Trong trường hợp đó, tên có thể đề cập đến thành viên đó của đối tượng mà hàm được xác định. Ví dụ Date::init() xác định cho ngày.

1.1.3.1 Hàm khởi tạo

Việc sử dụng các hàm như init () để cung cấp khởi tạo cho các lớp là không phù hợp

và dễ xảy ra lỗi. Bởi vì không có chỗ nào khẳng định rằng một đối tượng phải được

khởi tạo, một lập trình viên có thể quên làm như vậy - hoặc làm như vậy hai lần

(thường cho kết quả tệ như nhau). Một cách tiếp cận tốt hơn là cho phép lập trình

viên khai báo một hàm với mục đích rõ ràng là khởi tạo các đối tượng. Bởi vì như

vậy một hàm xây dựng các giá trị của một kiểu nhất định, nó được gọi là một

constructor. Một phương thức khởi tạo được công nhận khi có cùng tên với chính

lớp đó.

|  |
| --- |
| class Date {  int d, m, y;  public:  Date(int dd, int mm, int yy); // constructor  // ...  }; |

Khi 1 lớp có phương thức khởi tạo, tất cả các đối tượng của lớp sẽ được khởi tạo bởi lệnh của phương thức. nếu constructor yêu cầu các đối số, các đối số này phải được cung cấp. Bằng cách cung cấp một số hàm tạo, chúng tôi có thể cung cấp nhiều cách khác nhau để khởi tạo các đối tượng của một kiểu. Ví dụ:

|  |
| --- |
| class Date {  int d, m, y;  public:  // ...  Date(int, int, int); // day, month, year  Date(int, int); // day, month, today’s year  Date(int); // day, today’s month and year  Date(); // default Date: today  Date(const char∗); // date in string representation  }; |

Các hàm tạo tuân theo các quy tắc chồng giống như các hàm thông thường. Miễn là các hàm tạo có đủ khác biệt về kiểu đối số của chúng, trình biên dịch có thể chọn đúng để sử dụng: Sự gia tăng của các hàm tạo trong ví dụ Date là điển hình. Khi thiết kế một lớp, một lập trình viên luôn bị kích thích để thêm các tính năng chỉ vì ai đó có thể muốn chúng. Nó khiến cho việc suy nghĩ nhiều hơn để quyết định cẩn thận những tính năng nào thực sự cần thiết và chỉ bao gồm những tính năng đó. Tuy vậy, suy nghĩ thêm thường dẫn đến các chương trình nhỏ hơn và dễ hiểu hơn. Một cách để giảm số lượng các hàm liên quan là sử dụng các đối số mặc định . Đối với Date, mỗi đối số có thể được cung cấp một giá trị mặc định được hiểu là '' chọn giá trị mặc định: hôm nay.'' Lưu ý rằng bằng cách đảm bảo khởi tạo các đối tượng đúng cách, các hàm tạo sẽ đơn giản hóa rất nhiều thực hiện các chức năng thành viên. Đã cho các hàm tạo, các hàm thành viên khác không còn để đối phó với khả năng dữ liệu chưa được khởi tạo.

1.1.3.2 Các hàm thành viên không đổi

Date xác định cung cấp các hàm thành viên để gán một giá trị cho Date. Rất tiếc, chúng tôi không cung cấp cách kiểm tra giá trị của Ngày. Sự cố này có thể dễ dàng được khắc phục bằng cách thêm các chức năng đọc ngày, tháng và năm.

Hằng số sau danh sách đối số (trống) trong khai báo hàm chỉ ra rằng các hàm này không sửa đổi trạng thái của Date. Đương nhiên, trình biên dịch sẽ bắt gặp những lỗi. Ví dụ:

|  |
| --- |
| int Date::year() const{  return ++y; // error : cố gắng thay đổi giá trị thành viên trong hàm const  } |

Khi một hàm thành viên const được định nghĩa bên ngoài lớp của nó, thì hậu tố const là bắt buộc.

Nói cách khác, const là một phần của kiểu Date :: day (), Date :: month () và Date ::year ().

Một hàm thành viên const có thể được gọi cho cả các đối tượng const và không phải

const, trong khi một hàm thành viên không phải const chỉ có thể được gọi cho các

đối tượng không phải const.

### 1.1.4 [static]Thành viên

Sự tiện lợi của giá trị mặc định cho Date được mua bằng cái giá của một vấn đề ẩn. Lớp Date trở nên phụ thuộc vào biến toàn cục Today. Lớp Date này chỉ có thể được sử dụng Today được định nghĩa và sử dụng chính xác bởi mọi đoạn mã. Đây là loại ràng buộc khiến một lớp trở nên vô nghĩa. Người dùng không hài lòng khi cố gắng sử dụng các lớp phụ thuộc ngữ cảnh như vậy và việc bảo trì trở nên lộn xộn. Có thể ‘‘ chỉ một biến toàn cục nhỏ ’’ không quá khó quản lý, nhưng kiểu đó dẫn đến mã không thể sử dụng ngoại trừ lập trình ban đầu của nó. nên tránh trường hợp này. chúng ta có thể thuận lợi khi không có sự cản trở của một biến toàn cầu có thể truy cập công khai. Một biến là một phần của một lớp, không phải là một phần của đối tượng lớp đó, được gọi là một thành viên STATIC. Có bản sao chính xác của thành viên STATIC thay vì một bản sao cho mỗi đối tượng, như đối với các thành viên NON STATIC thông thường. Tương tự, một hàm cần quyền truy cập vào các thành viên của một lớp, nhưng không cần được gọi cho một đối tượng cụ thể, được gọi là hàm thành viên STATIC. Đây là một thiết kế bảo toàn nghĩa của các giá trị phương thức khởi tạo mặc định cho Date mà không gặp các vấn đề bắt nguồn từ việc phụ thuộc khác

|  |
| --- |
| class Date {  int d, m, y;  static Date default\_date;  public:  Date(int dd =0, int mm =0, int yy =0);  // ...  static void set\_default(int dd, int mm, int yy); // set default\_date to Date(dd,m  m,yy)  }; |

Bây giờ chúng ta có thể xác định hàm tạo DATE để sử dụng default\_date

1.1.4.1 Các loại thành viên

Các kiểu và các kiểu bí danh kiểu có thể là thành viên của một lớp. Ví dụ: template<typename T>

Một lớp thành viên (thường được gọi là lớp lồng nhau) có thể tham chiếu đến các kiểu và các thành viên static của lớp bao quanh nó. Nó chỉ có thể tham chiếu đến các thành viên non static khi nó được cung cấp một đối tượng của lớp bao quanh để tham chiếu đến.

Các lớp thành viên là một sự tiện lợi về mặt ký hiệu hơn là một tính năng có tầm quan trọng cơ bản. Mặt khác, bí danh thành viên rất quan trọng như là cơ sở của kỹ thuật lập trình chung dựa trên các kiểu liên kết. Enum thành viên thường là một sự thay thế cho các lớp enum.

### 1.1.5 Nạp chồng toán tử

|  |
| --- |
| bool operator!=(Date, Date); // toán tử khác  bool operator<(Date, Date); // toán tử bé hơn  bool operator>(Date, Date); // toán tử lớn hơn  // ...  Date& operator++(Date& d) { return d.add\_day(1); } // toán tử tăng Date lên 1 ngà  y  Date& operator−−(Date& d) { return d.add\_day(−1); } // toán tử giảm Date xuống 1  ngày  Date& operator+=(Date& d, int n) { return d.add\_day(n); } // cộng thêm n ngày  Date& operator−=(Date& d, int n) { return d.add\_day(−n); } // trừ n ngày  Date operator+(Date d, int n) { return d+=n; } // cộng n ngày  Date operator−(Date d, int n) { return d+=n; } // trừ n ngày  ostream& operator<<(ostream&, Date d); // output  istream& operator>>(istream&, Date& d); // input |

Các toán tử này được định nghĩa để tránh quá tải và hưởng lợi từ tra cứu phụ thuộc vào đối số. Đối với Date, những toán tử này có thể được coi là những tiện ích đơn thuần. Tuy nhiên, đối với nhiều loại - chẳng hạn như số phức, vectơ, và các đối tượng giống hàm - việc sử dụng các toán tử thông thường khiến mọi người cho rằng định nghĩa của họ gần như là bắt buộc. Đối với Date, tôi đã muốn cung cấp + = và - = dưới dạng các hàm thành viên thay vì add\_day ().Lưu ý rằng việc gán và khởi tạo sao chép được cung cấp theo mặc định.

## 1.2 HÀM KHỞI TẠO, XÓA, SAO CHÉP VÀ DI CHUYỂN

### 1.2.1 Giới thiệu

Chương này tập trung vào các khía cạnh kỹ thuật ‘‘ vòng đời ’’ của đối tượng: Làm cách nào để chúng ta tạo một đối tượng, cách chúng ta sao chép nó, cách chúng ta di chuyển nó và làm cách nào để dọn dẹp nó sau khi nó biến mất? Định nghĩa thích hợp của '' copy '' và '' move '' là gì?

### 1.2.2 Hàm tạo và hàm hủy

Chúng ta có thể chỉ định cách khởi tạo một đối tượng của một lớp bằng cách định nghĩa một phương thức khởi tạo. Để bổ sung cho các hàm tạo, chúng ta có thể xác định một hàm hủy để đảm bảo '' dọn dẹp '' tại điểm phá hủyđối tượng (ví dụ: khi nó vượt ra khỏi phạm vi). Một số thao tác hiệu quả nhất để quản lý tài nguyên trong C++ dựa trên các cặp hàm tạo / hủy. Vì vậy, các kỹ thuật khác dựa trên một cặp hành động, chẳng hạn như thực hiện / hoàn tác, bắt đầu / dừng, trước / sau, v.v.

Ví dụ:

|  |
| --- |
| struct Tracer {  string mess;  Tracer(const string& s) :mess{s} { cout << mess; }  ~Tracer() {cout << "˜" << mess; }  };  void f(const vector<int>& v){  Tracer tr {"in f()\n"};  for (auto x : v) {  Tracer tr {string{"v loop "}+to<string>(x)+'\n'};  // ...  }  } |

Chúng ta có thể thực hiện lệnh f({2,3,5});

Điều này sẽ hiện ra: in\_f() v loop 2 ˜v loop 2 v loop 3 ˜v loop 3 v loop 5 ˜v loop 5 ˜in\_f()

1.2.2.1 Hàm hủy và tài nguyên

Một hàm tạo khởi tạo một đối tượng. Nói cách khác, nó tạo ra môi trường mà các chức năng thành viên hoạt động. Đôi khi, việc tạo ra môi trường đó liên quan đến việc có được một tài nguyên - chẳng hạn như tệp, khóa hoặc một số bộ nhớ - phải được giải phóng sau khi sử dụng. Do đó, một số lớp cần một hàm để đảm bảo sẽ được khai báo khi một đối tượng bị hủy theo cách tương tự như cách một phương thức khởi tạo đảm bảo sẽ được gọi khi một đối tượng được tạo. Không thể tránh khỏi, một hàm như vậy được gọi là hàm hủy. Tên của hàm hủy là ˜ theo sau là tên lớp, ví dụ ~Vector (). Một nghĩa của ~ là ‘‘ bổ sung ’’ và hàm hủy cho một lớp bổ sung cho các hàm tạo của nó. Một hàm hủy không nhận đối số và một lớp chỉ có thể có một hàm hủy. Bộ hủy được gọi ngầm khi một biến tự động vượt ra khỏi phạm vi, một đối tượng trên cửa hàng miễn phí bị xóa, v.v. Chỉ trong những trường hợp rất hiếm, người dùng mới cần gọi hàm hủy một cách rõ ràng. Hàm hủy thường dọn dẹp và giải phóng tài nguyên.

Ví dụ:

|  |
| --- |
| class Vector {  public:  Vector(int s) :elem{new double[s]}, sz{s} { }; // constructor: acquire memory  ˜Vector() { delete[] elem; } // destructor: release memory  // ...  private:  double∗ elem; // elem points to an array of sz doubles  int sz; // sz is non-negative  }; |

Tại đây, Vectơ v1 bị hủy khi thoát khỏi f (). Ngoài ra, Vector tạo trên class bởi f() sử dụng new sẽ bị hủy bởi lệnh xóa. Trong cả hai trường hợp, hàm hủy của Vector được gọi để giải phóng (phân bổ) bộ nhớ được cấp phát bởi hàm tạo. Điều gì sẽ xảy ra nếu hàm tạo không có đủ bộ nhớ?

Ví dụ: s∗sizeof(double) or (s+s)∗sizeof(double) có thể lớn hơn dung lượng bộ nhớ khả dụng (tính bằng byte). Trong trường hợp đó, một ngoại lệ std :: bad\_alloc bị vứt bởi new và cơ chế xử lý ngoại lệ mở ra các hàm hủy thích hợp để lấy và giải phóng tất cả bộ nhớ . Phương thức quản lý tài nguyên dựa trên phương thức khởi tạo / hủy này được gọi là Khởi tạo Tài nguyên hay đơn giản là RAII. Một cặp hàm tạo / hàm hủy phù hợp là cơ chế thông thường để thực hiện khái niệm về một đối tượng có kích thước thay đổi trong C ++. Các vùng chứa thư viện tiêu chuẩn, chẳng hạn như vectơ và map không có thứ tự, sử dụng các biến thể của kỹ thuật này để cung cấp lưu trữ cho các phần tử của chúng. không có trình hủy được khai báo, chẳng hạn như kiểu tích hợp, được coi là có trình hủy không làm gì cả. Lập trình viên khai báo hàm hủy cho một lớp cũng phải quyết định xem đối tượng của lớp đó có thể được sao chép hoặc di chuyển hay không.

### 1.2.3 Sao chép và Di chuyển

Khi chúng ta cần chuyển một giá trị từ a sang b, chúng ta thường có hai lựa chọn khác nhau về mặt logic:

• Sao chép là ý nghĩa quy ước của x = y; nghĩa là, giá trị của x và y đều bằng giá trị của y trước khi gán.

+ Sao chép cho một lớp X được xác định bằng hai phép toán:

• Sao chép hàm tạo: X (const X &)

• Sao chép phép gán: X & operator = (const X &)

• Di chuyển các x với giá trị cũ của y và y với một số trạng thái đã chuyển. Đối với các trường hợp, vùng chứa, trạng thái được chuyển từ đó là ‘‘ trống ’’. Sự phân biệt này bị nhầm lẫn và thực tế là chúng ta sử dụng cùng một ký hiệu cho cả di chuyển và sao chép. Thông thường, không thể xóa đi bước di chuyển, trong khi một bản sao có thể (vì nó có thể cần lấy tài nguyên), và di chuyển thường hiệu quả hơn một bản sao. Khi bạn thực hiện một thao tác di chuyển, bạn nên để đối tượng nguồn ở trạng thái hợp lệ nhưng không xác định vì cuối cùng nó sẽ bị hủy và trình hủy không thể hủy đối tượng còn lại ở trạng thái không hợp lệ. Ngoài ra, các thuật toán thư viện tiêu chuẩn dựa vào việc có thể gán cho (sử dụng di chuyển hoặc sao chép) một đối tượng được chuyển đến. Vì vậy, hãy thiết kế các bước di chuyển của bạn để các đối tượng nguồn cho phép hủy và chuyển nhượng. Để tránh lặp lại, hãy sao chép và di chuyển các định nghĩa mặc định. Ví dụ:

|  |
| --- |
| template<class T>  void swap(T& a, T& b)  {  const T tmp = a; // put a copy of a into tmp  a = b; // put a copy of b into a  b = tmp; // put a copy of tmp into b  }; |

### 1.2.4 Các chức năng deleted

Chúng ta có thể ‘‘ xóa ’’ một hàm; nghĩa là, chúng ta có thể nói rằng một hàm không tồn tại và không thể sử dụng nó. chức năng rõ ràng nhất là loại bỏ các chức năng mặc định khác. Ví dụ, người ta thường muốn ngăn chặn việc sao chép các lớp được sử dụng làm cơ sở vì việc sao chép như vậy dễ dẫn đến việc cắt lớp.

|  |
| --- |
| class Base {  // ...  Base& operator=(const Base&) = delete;// disallow copying  Base(const Base&) = delete;  Base& operator=(Base&&) = delete; // disallow moving  Base(Base&&) = delete;  };  Base x1;  Base x2 {x1}; // error : no copy constructor |

Một chức năng nữa là kiểm soát nơi có thể cấp phát một lớp

Lưu ý sự khác biệt giữa một hàm = deleted và một hàm chỉ đơn giản là chưa được khai báo.

# CHƯƠNG 2: MỞ ĐẦU

## 2.1. Mục đích:

Trong thời đại công nghệ 4.0 có xu hướng trong việc tự động hóa và trao đổi dữ liệu trong công nghiệp sản xuất. Vì thế, việc ứng dụng các thành tựu khoa học - kỹ thuật vào đời sống càng ngày càng phổ biến. Điều đó làm cho kinh tế của đất nước phát triển dẫn đến nhu cầu của con người ngày càng cao về mọi mặt. Trong các rạp chiếu phim hiện nay, việc quản lý và tổ chức chiếu phim, bán vé luôn là vấn đề được quan tâm. Người quản lý luôn gặp khó khăn trong vấn đề quản lý quá trình hoạt động của từng bộ phận. Chính vì lý do đó mà số lượng khách hàng đi xem phim ít hơn, thu nhập của rạp cũng bị ảnh hưởng. Vì vậy, để đáp ứng được nhu cầu của người dùng nhóm đã tạo ra một chương trình quản lý rạp chiếu phim giúp các rạp phim có thể giải quyết vấn đề trên.

## 2.2. Mục tiêu và nhiệm vụ nghiên cứu:

Với yêu cầu hiện nay của các rạp chiếu phim cần tạo ra được một công cụ quản lý quá trình hoạt động của từng bộ phận có thể ứng dụng rộng rãi cho mọi rạp phim, dễ dùng, dung lượng không quá nặng, hỗ trợ 100% việc quản lí quá trình hoạt động. Sự cạnh tranh giữa các phần mềm quản lý cũng diễn ra rất sôi động vì thế việc tạo ra một phần mềm quản lý hoàn hảo đáp ứng tất cả các nhu cầu là cần thiết.

* 1. Tạo ra một chương trình dùng để quản lý quá trình hoạt động: số lượng vé bán ra, người mua hàng, hỗ trợ các cửa hàng quản lý tốt nhất việc trao đổi mua bán.
  2. Tiết kiệm được nhiều thời gian và chi phí cho việc phải cần quá nhiều nhân lực để quản lý.
  3. Dễ dàng kiểm rạp phim.

Nhiệm vụ:

* 1. Tính tiền tốc độ nhanh, độ chính xác cao
  2. Có thể in hóa đơn từ máy in liên kết phần mềm
  3. Chỉnh sửa sai sót trong hóa đơn dễ dàng

## 2.3. Đối tượng và phạm vi nghiên cứu:

- Các rạp phim, dịch vụ

- Nhu cầu quản lý quá trình hoạt động, hóa đơn, thông tin khách hàng.

## 2.4. Cách tiếp cận và phương pháp nghiên cứu:

- Dựa trên nhu cầu của bản thân và người tiêu dùng.

- Tham khảo thông tin trên Internet, các trang mạng xã hội.

# CHƯƠNG 3: CƠ SỞ LÝ THUYẾT

## 3.1. Tổng quan ngôn ngữ lập trình C++

### *3.1.1. Khái niệm:*

C ++ là một ngôn ngữ lập trình được phát triển bởi Bjarne Stroustrup vào năm 1979 tại Bell Labs. C ++ được coi là ngôn ngữ bậc trung (middle-level) như một phần mở rộng của ngôn ngữ lập trình C, hoặc “C với các lớp Class” vì nó bao gồm sự kết hợp của cả các tính năng của ngôn ngữ cấp cao và cấp thấp.

### *3.1.2. Các đặc điểm nổi bật*

- Ngôn ngữ lập trình bậc trung:

Có thể sử dụng C++ để phát triển những chương trình bậc thấp, kể cả bậc cao đều hoạt động tốt trên hầu hết phần cứng.

- Đơn giản và hiệu quả:

Cú pháp của C++ rất dễ hiểu và khá giống tiếng Anh. Thậm chí có thể dùng C++ để thiết kế những ứng dụng dù từ trước đó đã được viết bằng hợp ngữ ( Assembly language

- Hỗ trợ đa nền tảng:

C++ được hỗ trợ trên bất kỳ nền tảng, hệ điều hành nào. Ta có thể viết một chương trình C++ trên Linux, biên dịch nó trên Windows và chạy trên Mac OS.

- Lập trình hướng đối tượng:

C++ phổ biến là do nó hỗ trợ vô cùng hiệu quả việc lập trình hướng đối tượng, bao gồm việc trừu tượng hóa dữ liệu, đóng gói dữ liệu, kế thừa và đa hình.

- Con trỏ:

Ngôn ngữ C++ cung cấp cho chúng ta một công cụ vô cùng tiện lợi là con trỏ, một con trỏ có nhiệm vụ lưu địa chỉ bộ nhớ làm giá trị của nó, đồng thời lưu trữ và truy cập dữ liệu từ bộ nhớ.

### *3.1.3. Lịch sử phát triển:*

C++ là ngôn ngữ lập trình đa năng được tạo ra bởi Bjarne Stroustrup như một phần mở rộng ngôn ngữ lập trình C, hoặc “C với các lớp Class”. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ thương mại ưa thích và phổ biến của lập trình viên. C++ được thiết kế hướng tới lập trình hệ thống máy tính và phần nhúng trên các vi mạch xử lý, bao gồm cả hệ thống có tài nguyên hạn chế và tài nguyên khổng lồ, với ưu điểm vượt trội về hiệu suất, hiệu quả và tính linh hoạt cao. C ++ có thể tìm thấy ở mọi nơi, với những điểm mạnh là cơ sở hạ tầng phần mềm và các ứng dụng bị hạn chế tài nguyên. bao gồm: phần mềm ứng dụng máy tính cá nhân, trò chơi điện tử, các hệ thống máy chủ (ví dụ: phần mềm thương mại điện tử, cỗ máy tìm kiếm trên web hoặc máy chủ SQL) và các ứng dụng ưu tiên về hiệu suất (ví dụ: tổng đài thông tin liên lạc hoặc thiết bị thăm dò không gian). C++ hầu hết được thực thi dưới dạng là một ngôn ngữ biên dịch, có thể chạy trên nhiều nền tảng khác nhau như Windows, Mac OS, Linux, các phiên bản Unix. Nhiều nhà cung cấp cung cấp các trình biên dịch C ++, bao gồm Tổ chức Phần mềm Tự do, Microsoft, Intel và IBM.

### *3.1.4. Ưu nhược điểm của ngôn ngữ c++*

Ưu điểm:

- Là một ngôn ngữ lập trình hướng đối tượng

Đây là một trong những lí do Developer có thể dùng C++ để học lập trình hướng đối tượng mặc dù không được khuyến khích. Nhưng hướng đối tượng là một tính chất mà một ngôn ngữ lập trình hiện đại nên có, và C++ cung cấp đủ những tính chất của hướng đối tượng như tính đóng gói, tính đa hình, tính trừu tượng và tính kế thừa.

- C++ cho phép thoải mái quản lý vùng nhớ

Trong c++ chúng ta không có Garbage Collectors để dọn dẹp vùng nhớ như một số ngôn ngữ bậc cao khác. Một vùng nhớ có thể sử dụng lại nhiều lần, điều này giúp cải thiện hiệu suất chương trình C++.

- C++ là một ngôn ngữ low-level dễ dàng giao tiếp với phần cứng

Một chương trình phần mềm dùng để giao tiếp với phần cứng hay một Embedded System đều được ưu tiên dùng C++.

C++ có một hiệu suất cao cùng khả năng tiêu tốn ít tài nguyên phần cứng khiến chương trình chạy nhanh hơn. Cũng vì hiệu suất cao nên C++ cũng được dùng để phát triển game. Nói chung, một chương trình đòi hỏi hiệu suất cao thì phần core lúc nào cũng sẽ được ưu tiên viết bằng C++.

Nhược điểm

- Tràn vùng nhớ

Cho phép bạn tự quản lý vùng nhớ, giúp cho việc tái sử dụng vùng nhớ một cách hiệu quả nhưng nếu quản lý không tốt việc tràn vùng nhớ sẽ xảy ra.

- OOP trong C++ khá phức tạp và khó hiểu

Trong C++ không có khái niệm Interface và có thêm khái niệm hàm thuần ảo, hàm bạn, lớp bạn, hàm hủy, đa kế thừa,… Do vậy, OOP trong C++ khá phức tạp và khó hiểu.

- Con trỏ rất khó

Con trỏ là một biến mang địa chỉ của một vùng nhớ mà biến đó trỏ tới. Những ngôn ngữ bậc cao đã lược bỏ khái niệm này để làm ngôn ngữ thân thiện và dễ hiểu hơn. Nhưng con trỏ sinh ra để giúp cho việc thao tác trực tiếp với vùng nhớ, tăng tốc độ thực thi của chương trình nên đôi khi khó học.

## 3.2. Tổng quan GITHUB

### *3.2.1. Khái niệm:*

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS), nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. Git cung cấp cho mỗi lập trình viên kho lưu trữ (repository) riêng chứa toàn bộ lịch sử thay đổi.

Git có khả năng chạy trên nhiều hệ điều hành khác nhau như: Linux, Windows, Mac OSX…

### *3.2.2. Lịch sử:*

Cũng như nhiều thứ tuyệt vời khác trong cuộc sống, Git ra đời từ một chút của sự huỷ diệt/phá sản/kết thúc có tính sáng tạo và sự tranh cãi nảy lửa. Nhân của Linux là một dự án phần mềm mã nguồn mở của một phạm vi khá lớn. Trong phần lớn thời gian bảo trì của nhân Linux (1991-2002), các thay đổi của phần mềm được truyền đi dưới dạng các bản vá và các tập tin lưu trữ. Vào năm 2002, dự án nhân Linux bắt đầu sử dụng một DVCS độc quyền có tên là BitKeeper. Vào năm 2005, sự hợp tác giữa cộng đồng phát triển nhân Linux và công ty thương mại phát triển BitKeeper bị phá vỡ, và công cụ đó không còn được cung cấp miễn phí nữa. Chính điều này đã thúc đẩy cộng đồng phát triển Linux (chính xác hơn là Linus Torvalds, người sáng lập ra Linux). phát triển công cụ của riêng họ dựa trên những bài học từ việc sử dụng BitKeeper. Một số mục tiêu của hệ thống mới được vạch ra như sau:

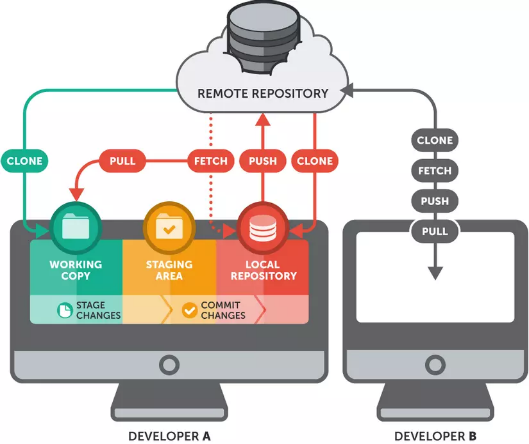
* Nhanh.
* Thiết kế đơn giản.
* Hỗ trợ tốt cho "phát triển phi tuyến tính" (non-linear development)  (hàng ngàn nhánh song song).
* Phân tán toàn diện.
* Có khả năng xử lý các dự án lớn giống như nhân Linux một cách hiệu quả (về mặt tốc độ và khối lượng dữ liệu).

Kể từ khi ra đời năm 2005, Git đã tiến hoá và phát triển toàn diện để dễ dàng sử dụng hơn, tuy thế các tiêu chí ban đầu vẫn được đảm bảo. Nó nhanh một cách đáng kinh ngạc, vô cùng hiệu quả với các dự án lớn, và một hệ thống phân nhánh không thể tin được cho phát triển phi tuyến tính

### *3.2.3. Các đặc điểm nổi bật*

*3.2.3.1. Repository*

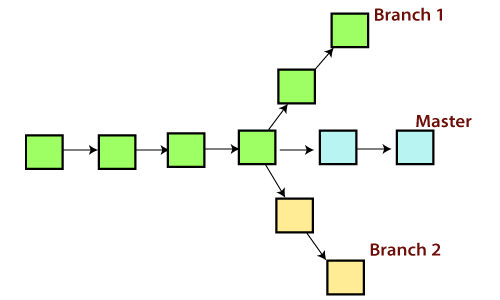
Repository là nơi quản lý, lưu trữ tất cả mọi thông tin cần thiết cũng như các sửa đổi và lịch sử sửa đổi của toàn bộ dự án. Có hai loại repository gồm local repository - là repository nằm trên chính máy tính của chúng ta và remote repository - là repository nằm trên một máy chủ từ xa được cung cấp bởi các nhà phân phối như Github, Gitlab hay Bitbucket, …



Hình 2.1 Repository

*3.2.3.2. Branch*

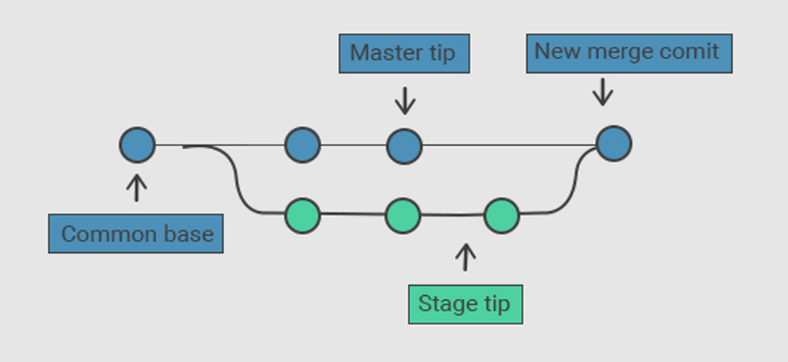
Nhánh là khái niệm rất hay trong Git với nhánh và bạn có thể tách riêng được các tính năng của dự án rồi thử nghiệm những tính năng mới dễ dàng hơn. Hoặc bạn cũng có thể sử dụng nhánh để có thể khắc phục và hoàn chỉnh lỗi bất kỳ nào đó của dự án.



Hình 2.2 Branch

*3.2.3.3. Merge*

Git Merge là một lệnh dùng để hợp nhất các chi nhánh độc lập thành một nhánh duy nhất trong Git



Hình 2.3 Merge

Chú ý:

Kiểm tra branch hiện đang làm việc trước khi merge

Phải đẩy tất cả những thay đổi dưới máy local lên Git trước khi merge

Trước khi merge phải lấy hết những thay đổi mới nhất của các branch khác, hay ít nhất là branch cần merge về máy

Merge thành công thì nên đẩy source lên lại lên server

Nên merge bằng GUI tool

*3.2.3.4. Commit*

Là thao tác báo cho hệ thống biết bạn có muốn lưu lại trạng thái hiện hành hay không rồi ghi nhận lại lịch sử các xử lý đã thực hiện như: xóa, cập nhật, thêm các file hoặc thư mục nào đó trên repository.

*3.2.3.5. Push*

Lệnh Push thường được sử dụng để đưa nội dung kho lưu trữ cục bộ lên server và nó cũng là cách bạn chuyển giao các commit từ kho lưu trữ cục bộ lên server.

*3.2.3.6. Pull*

Lệnh này sẽ lấy toàn bộ dữ liệu ở Repository trên server và gộp vào cái branch hiện tại mà bạn đang làm việc.

# CHƯƠNG 4: TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

## 4.1. Lập trình hướng đối tượng là gì?

Lập trình hướng đối tượng (Object Oriented Programming – OOP) là một trong những kỹ thuật lập trình rất quan trọng và sử dụng nhiều hiện nay. Hầu hết các ngôn ngữ lập trình hiện nay như Java, PHP, .NET, Ruby, Python… đều hỗ trợ **OOP**.

Lập trình hướng đối tượng (OOP) là một kỹ thuật lập trình cho phép lập trình viên tạo ra các đối tượng trong code trừu tượng hóa các đối tượng.

## 4.2. Các nguyên lý cơ bản của OOP

### *4.2.1 Tính đóng gói (Encapsulation)*

Các thuộc tính và phương thức của đối tượng cần cho việc giải quyết bài toán đã được chọn ra sẽ được đóng gói vào một kiểu dữ liệu gọi là class. Nhằm che dấu thông tin khỏi các đối tượng bên ngoài, thuộc tính nào thì phương thức nấy. Tức là code bên ngoài không trực tiếp nhìn thấy và thay đổi trạng thái của đối tượng đó.

### *4.2.2. Tính kế thừa (Inheritance)*

Nó cho phép xây dựng một lớp mới dựa trên các định nghĩa của lớp đã có. Có nghĩa là lớp cha có thể chia sẽ dữ liệu và phương thức cho các lớp con. Các lớp con khỏi phải định nghĩa lại, ngoài ra có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới. Tái sử dụng mã nguồn 1 cách tối ưu, tận dụng được mã nguồn. Một số loại kế loại kế thừa thường gặp: đơn kế thừa, đa kế thừa, kế thừa đa cấp, kế thừa thứ bậc.

### *4.2.3. Tính đa hình (Polymorphism)*

Tính đa hình là một hành động có thể được thực hiện bằng nhiều cách khác nhau. Đây lại là một tính chất có thể nói là chứa đựng hầu hết sức mạnh của lập trình hướng đối tượng.

Hiểu một cách đơn giản hơn: Đa hình là khái niệm mà hai hoặc nhiều lớp có những phương thức giống nhau nhưng có thể thực thi theo những cách thức khác nhau.

### *4.2.4. Tính trừu tượng (Abstraction)*

Trừu tượng có nghĩ là tổng quát hóa một cái gì đó lên, không cần chú ý chi tiết bên trong. Nó không màng đến chi tiết bên trong là gì và người ta vẫn hiểu nó mỗi khi nghe về nó.

Ở đây trong lập trình OOP, tính trừu tượng nghĩa là chọn ra các thuộc tính, phương thức của đối tượng cần cho việc giải quyết bài toán đang lập trình. Vì một đối tượng có rất nhiều thuộc tính phương thức, nhưng với bài toán cụ thể không nhất thiết phải chọn tất cả.

## 4.3. Các ưu điểm của lập trình hướng đối tượng

Dựa trên nguyên lý kế thừa, trong quá trình mô tả các lớp có thể loại bỏ những chương trình bị lặp, dư. Và có thể mở rộng khả năng sử dụng các lớp mà không cần thực hiện lại. Tối ưu và tái sử dụng code hiệu quả.

Đảm bảo rút ngắn thời gian xây dựng hệ thống và tăng năng suất thực hiện.

Sự xuất hiện của 2 khái niệm mới là lớp và đối tượng chính là đặc trưng của phương pháp lập trình hướng đối tượng. Nó đã giải quyết được các khuyết điểm của phương pháp lập trình hướng cấu trúc để lại. Ngoài ra 2 khái niệm này đã giúp biểu diễn tốt hơn thế giới thực trên máy tính.

# CHƯƠNG 5: PHÂN TÍCH HỆ THỐNG

## 5.1. Thư viện

Sử dụng thư viện hỗ trợ giao diện Winform thiết kế giao diện người dùng. Ưu điểm của Winform là dễ xây dựng các nút thao tác câu lệnh.

## 5.2. Cơ sở dữ liệu

Sử dụng Access lưu trữ tất cả thông tin của ứng dụng. Những thông tin về phim, khách hàng, vé được lưu trữ hoàn toàn trên Access.

Được sử dụng miễn phí, tốc độ xử lý nhanh, dễ dàng thao tác là ưu điểm mà Access mang lại.

## 5.3. Ứng dụng lập trình hướng đối tượng

Lập trình với kỹ thuật lập trình hướng đối tượng là mẫu lập trình phổ biến hiện nay.

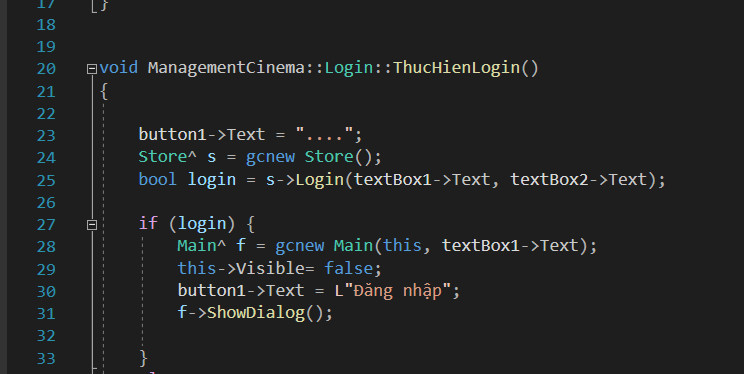
Lập trình theo hướng đối tượng giúp tối ưu code và tái sử dụng code hiệu quả, rút ngắn thời gian xây dựng hệ thống và tăng năng năng suất thực hiện

### *5.3.1. Tính đóng gói*



Hình 5.1 Tính đóng gói

### *5.3.2. Tính kế thừa*

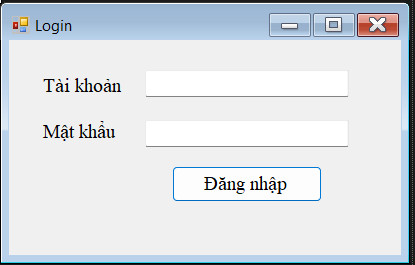


Hình 5.2 Tính kế thừa

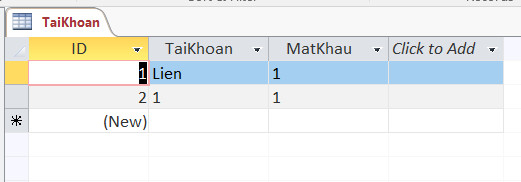
## 5.4. Phân tích hệ thống

### *5.4.1. Đăng nhập*

Khi khởi động chương trình, hệ thống sẽ hiện form yêu cầu người dùng tiến hành đăng nhập để thực hiện các thao tác.

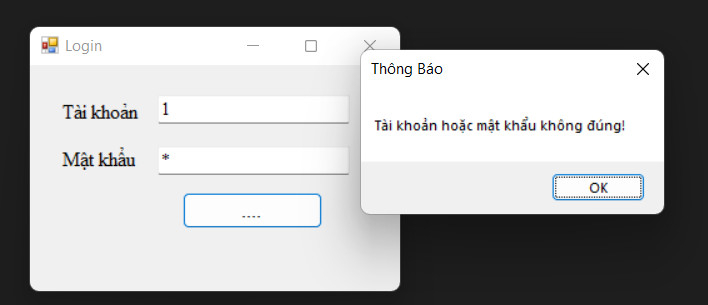


Hình 5.3 Login



Hình 5.4 Login ở Access

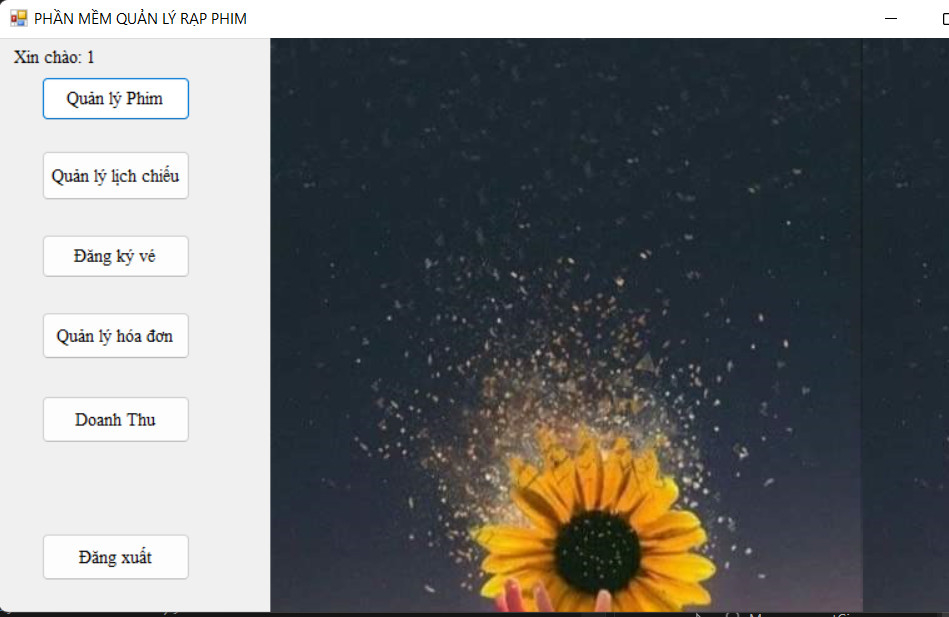
Chương trình hiển thị form Login yêu cầu đăng nhập. Dữ liệu tài khoản đăng nhập được lấy từ Access, nếu người dùng nhập đúng với dữ liệu trong bảng Tài khoản thì form main sẽ hiện lên. Nếu sai tài khoản hoặc mật khẩu, hệ thống sẽ thông báo cho người dùng để tiến hành đăng nhập lại.



Hình 5.5 Đăng nhập sai

Sau khi đăng nhập thành công hệ thống sẽ chuyển đến form main như phần 4.4.2

### *5.4.2. Form main*

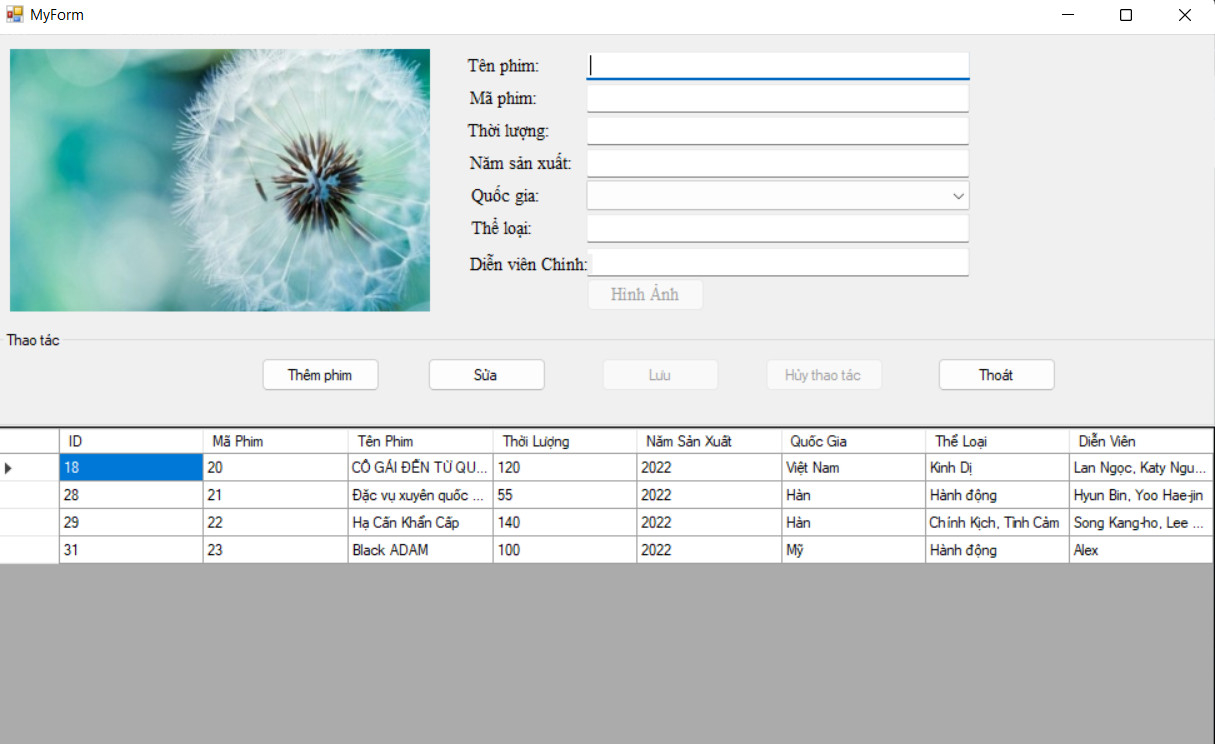


Hình 5.6 Form Main

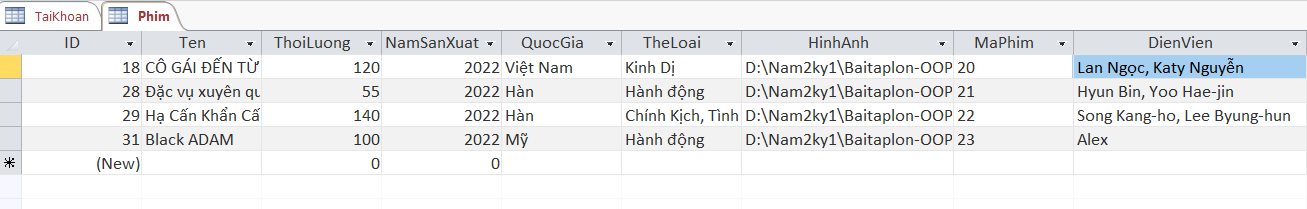
Ở đây sẽ hiển thị 5 chức năng chính gồm có: Quản lý phim, Quản lý lịch chiếu, Đăng ký vé, Quản lý hóa đơn và Doanh thu.

### *5.4.3. Quản lý phim*

Người dùng có thể xem thông tin của những phim hiện tại đang có ở bảng chi tiết phim phía dưới. Hình ảnh minh họa bên dưới đã được cập nhật 4 bộ phim với dữ liệu được lưu và truy xuất ở Access. Thông tin phim mà người dùng có thể xem được gồm có: Poster phim, tên phim, mã phim, thời lượng phim, năm sản xuất, quốc gia, thể loại, diễn viên chính.



Hình 5.7 Quản lý phim



Hình 5.8 Quản lý phim ở access

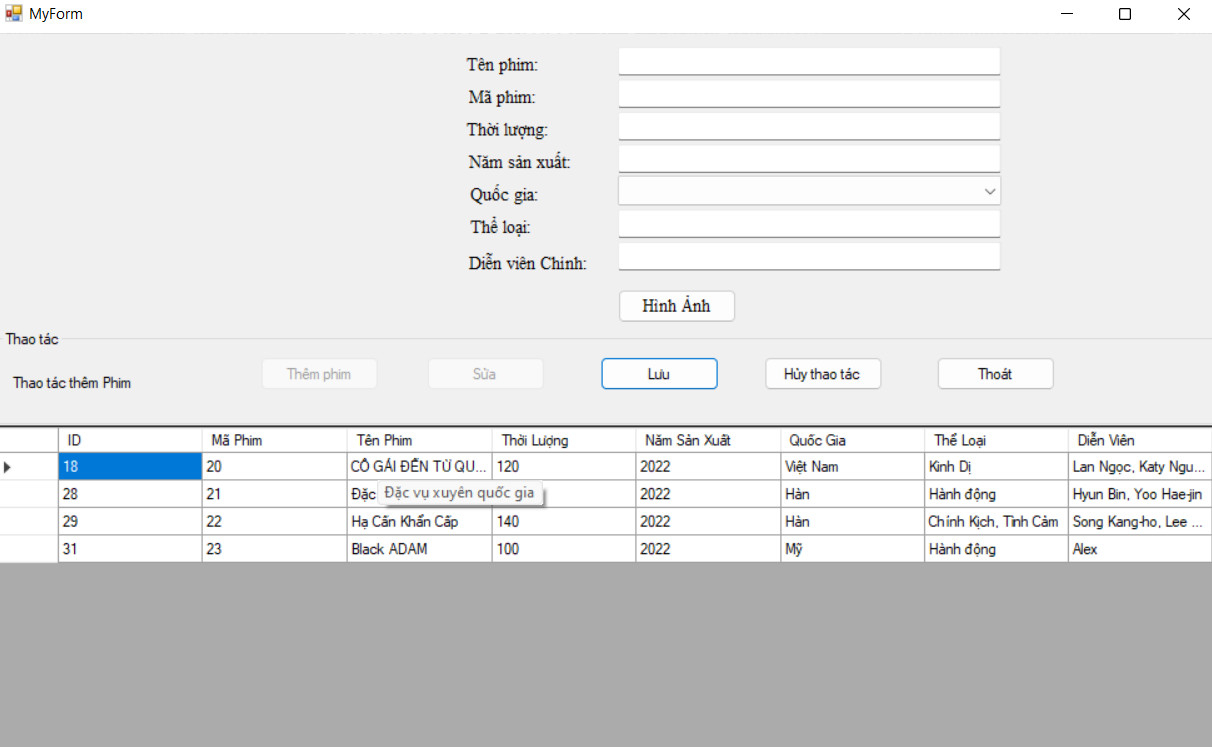
Ở chức năng quản lý phim người dùng có 2 phần chính để thao tác là thêm phim mới và chỉnh sửa thông tin phim.

*5.4.3.1. Thêm phim*

Khi bấm thêm phim nút thêm phim và sửa phim sẽ bị ẩn đi.

Phần thêm phim sẽ cho phép người dùng thêm phim mới vào danh sách quản lý phim. Hệ thống sẽ yêu cầu nhập đầy đủ thông tin cơ bản trước khi ấn nút lưu. Hoặc khi chúng ta không muốn thực hiện thao tác thêm phim, nút hủy thao tác sẽ giúp chúng ta thực hiện việc đó. Những thông tin trước đó sẽ không được lưu lại.

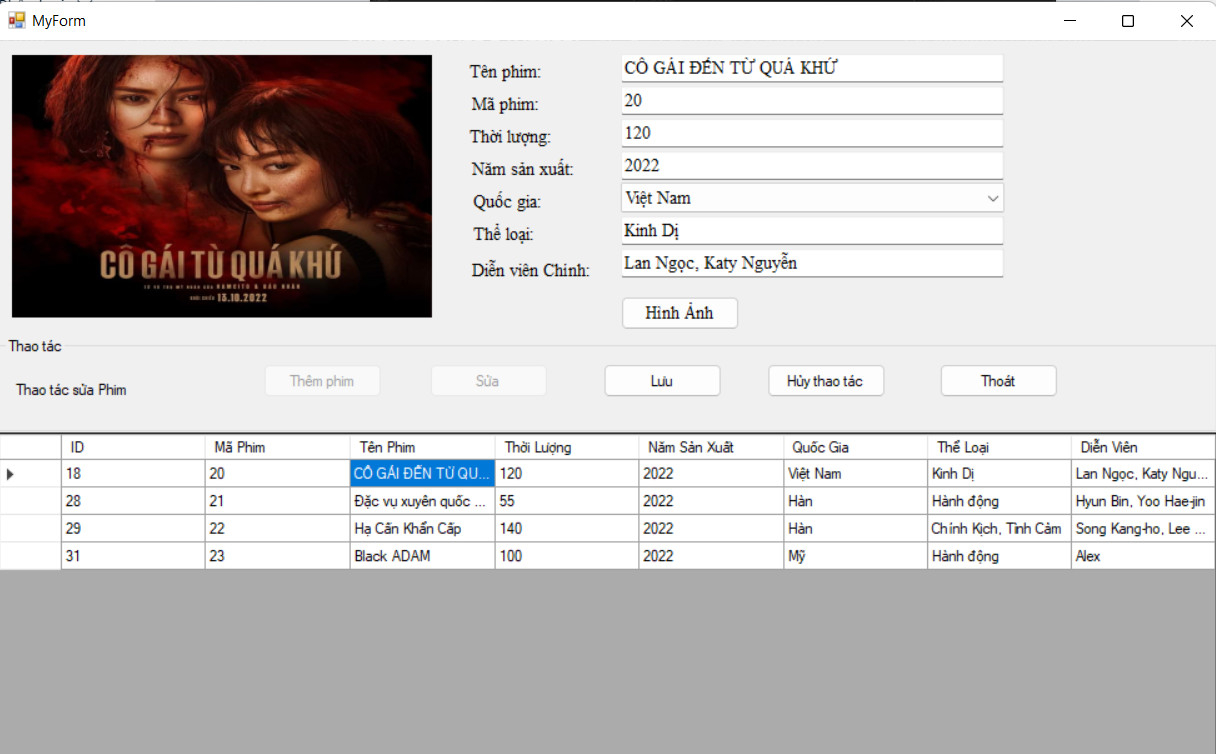
Mã phim không được nhập trùng với những bộ phim đã có trong danh sách phim và ID phim được hệ thống tạo tự động không trùng với những bộ phim khác.



Hình 5.9 Thêm phim mới

*5.4.3.2. Sửa thông tin phim*

Sửa thông tin phim cho phép người dùng điều chỉnh lại thông tin của phim nếu thông tin nhập sai hoặc thông tin được cập nhật mới hay cập nhật một chiếc Poster thật xịn xò khi chúng ta thêm chưa kịp cập nhật chiếc ảnh đó. Tất cả thông tin đều có thể thực hiện chỉnh sửa ngoại trừ ID phim hệ thống tự tạo.

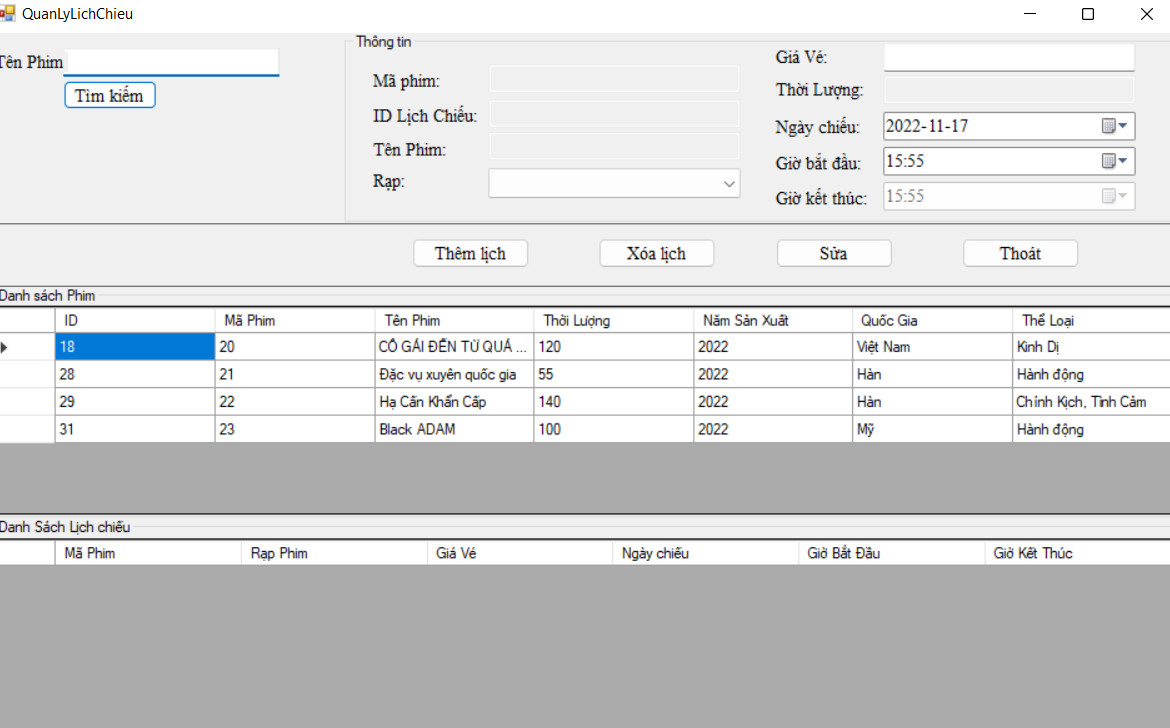


Hình 5.10 Sửa phim

### *5.4.4. Quản lý lịch chiếu*

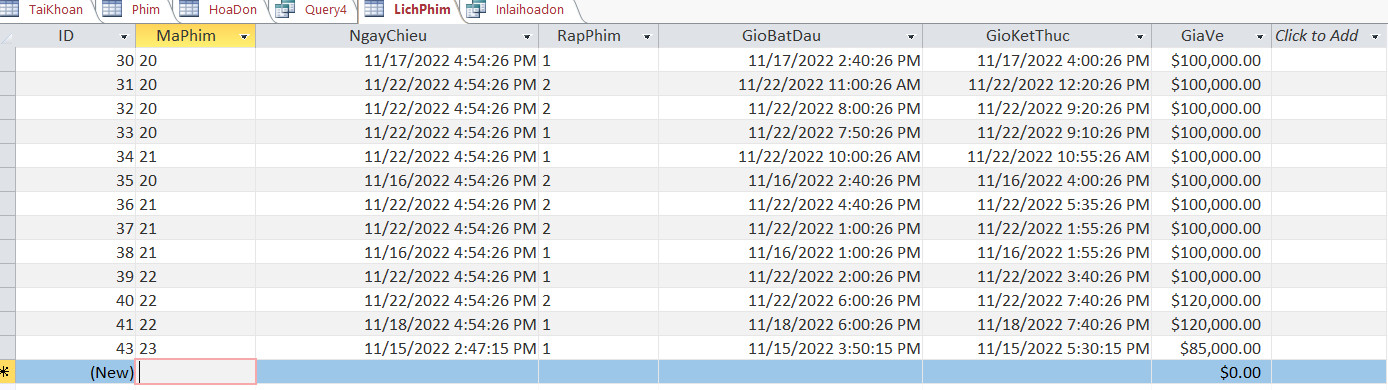
Ở phần này ta sẽ làm việc với lịch chiếu phim gồm các thao tác: thêm lịch chiếu, sửa lịch chiếu, xóa lịch chiếu. Trên form giao diện thông tin các bộ phim sẽ hiện ra đầy đủ giúp người dùng dễ dàng quan sát và kiểm tra thông tin.

Công cụ có hỗ trợ các tính năng như tìm kiếm phim dựa trên các ký tự có xuất hiện ở tên phim. Sau khi tìm thấy được lịch phim mình mong muốn người dùng sẽ tiến hành các thao tác thêm, sửa, xóa.



Hình 5.11 Lịch Phim

Danh sách lịch chiếu phim được lưu trữ toàn bộ ở Access và sẽ được truy xuất dựa trên kết quả tìm kiếm người dùng nhập vào.

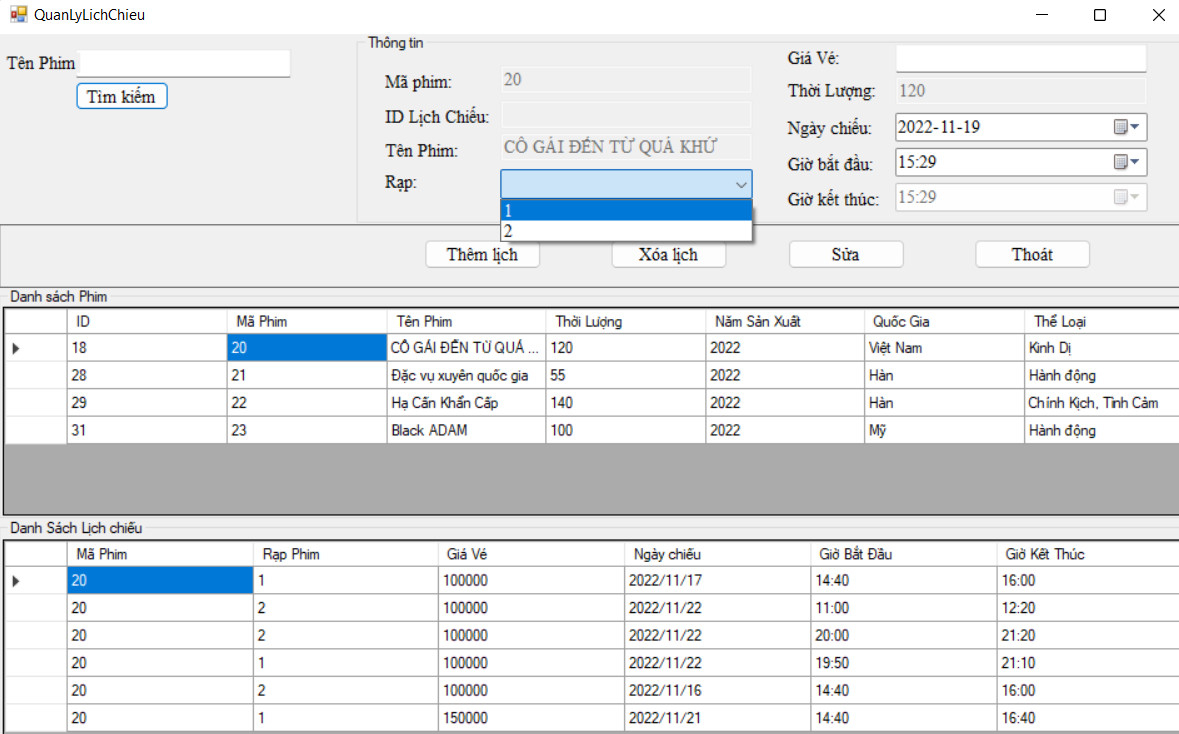


Hình 5.12 Lịch phim ở Access

*5.4.4.1. Thêm lịch chiếu*

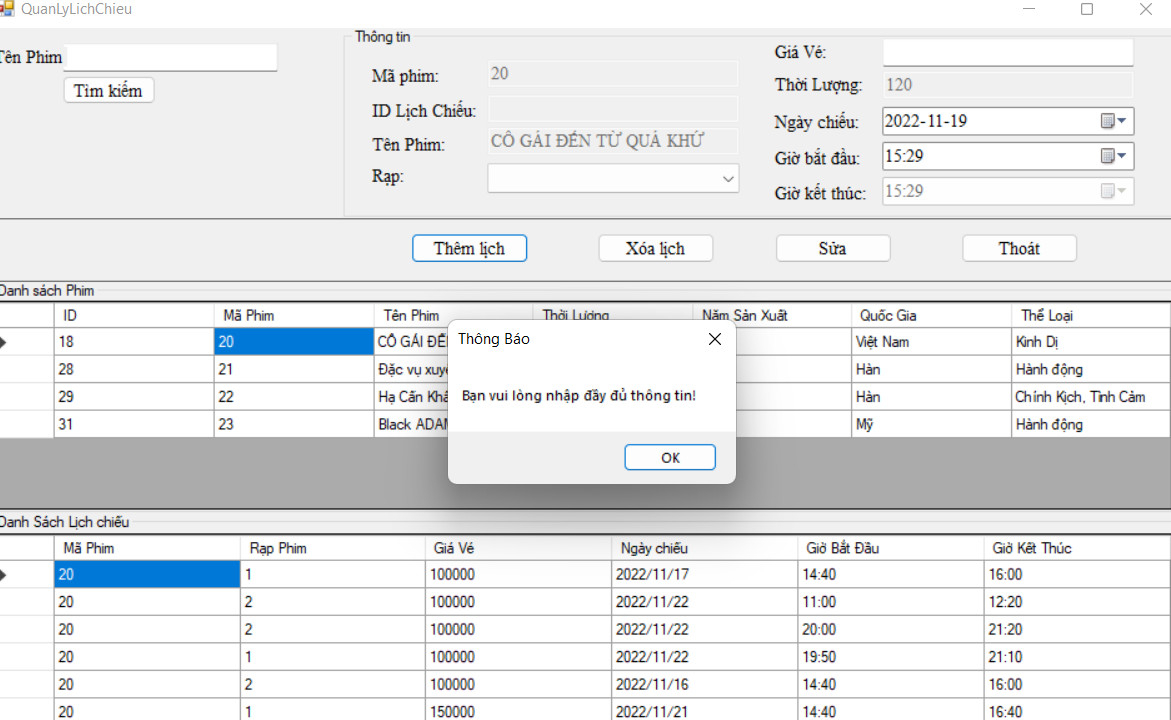
Trước tiên người dùng cần chọn phim muốn cài đặt thêm lịch chiếu ở bảng Danh sách phim. Trên bảng thông tin sẽ lấy dữ liệu của bộ phim đó và cho phép chúng ta thêm các thông tin như rạp, giá vé, thời gian chiếu. Những thông tin được cố định không được thay đổi gồm có ID lịch chiếu, mã phim, thời lượng. ID lịch chiếu sẽ được tạo tự động.

Khi nhập thời gian bắt đầu, hệ thống sẽ tự động cập nhật thời gian kết thúc của bộ phim dựa trên thời lượng của phim đó.



Hình 5.13 Thêm lịch chiếu

Nếu trong quá trình nhập thông tin mà thông tin chưa đầy đủ thì khi nhấn nút “Thêm lịch” hệ thống sẽ thông báo “Vui lòng nhập đầy đủ thông tin”. Khi đó người dùng sẽ phải nhập đầy đủ thông tin mới có thể thêm lịch.

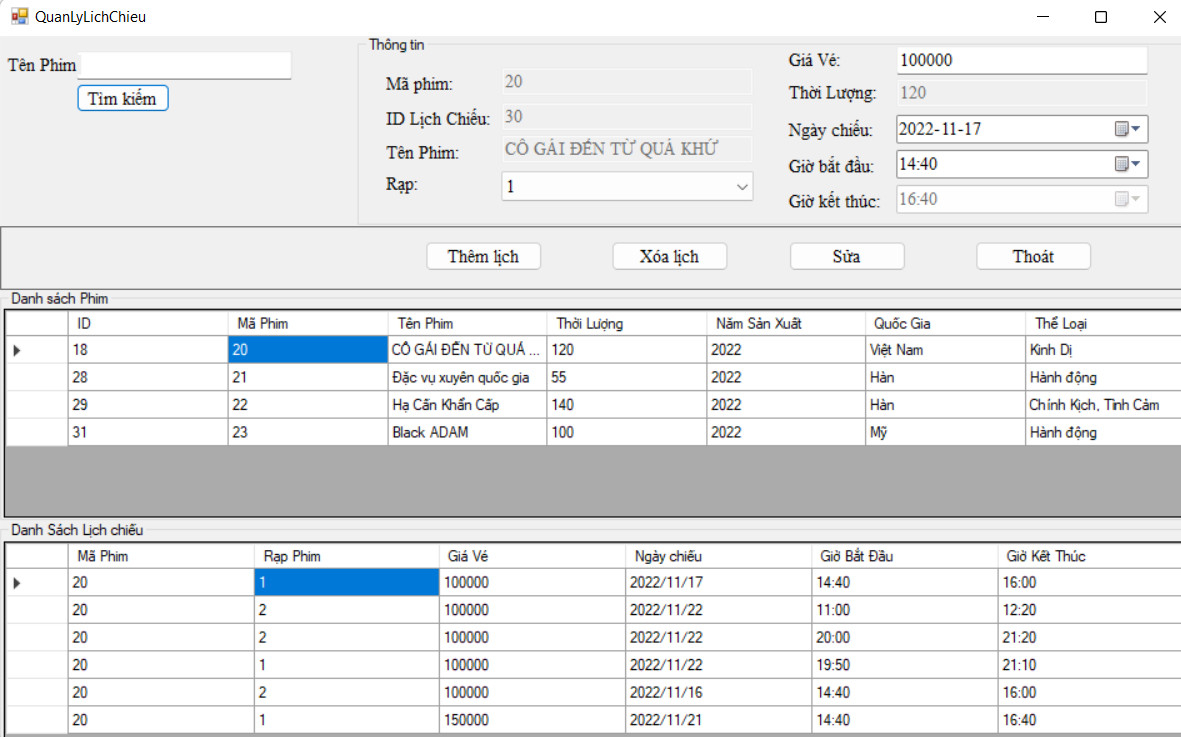


Hình 5.14 Thông báo thiếu thông tin lịch chiếu

*5.4.4.2. Sửa lịch chiếu*

Khi muốn sửa thông tin chiếu phim, người dùng phải nhấn chọn phim muốn sửa sau đó hệ thống sẽ hiện danh sách lịch chiếu ở bảng “Danh sách lịch chiếu”. Sau đó người dùng nhấn chọn vào lịch chiếu và tiến hành sửa.

Những thông tin có thể sửa gồm có: Rạp, giá vé, ngày chiếu và giờ chiếu.



Hình 5.15 Sửa lịch chiếu

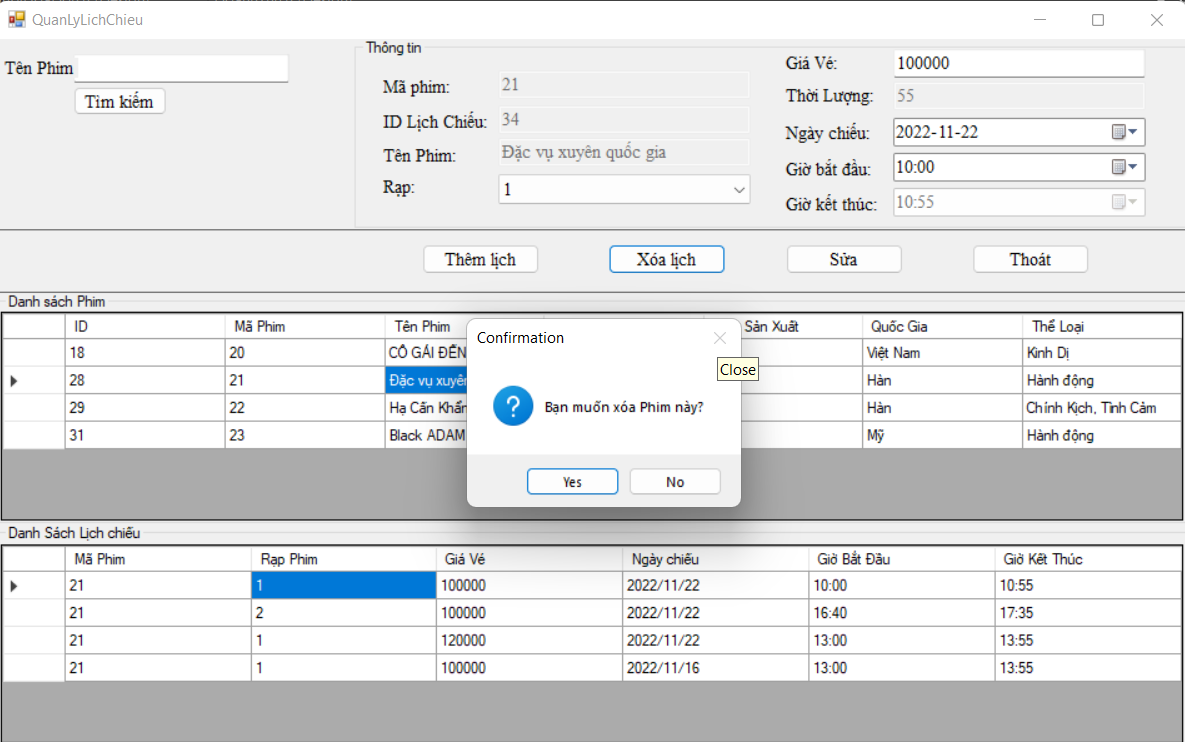
Thông tin sau khi được chỉnh sửa sẽ cập nhật lên Access.

*5.4.4.3. Xóa lịch chiếu*

Khi muốn xóa thông tin lịch chiếu của một bộ phim. Chúng ta thao tác như sửa một lịch phim.

Chúng ta sẽ nhấp vào phim muốn xóa. Thông tin các lịch phim của bộ phim đó sẽ hiện ở bảng danh sách lịch chiếu. Chúng ta tiếp tục nhấp vào lịch phim muốn xóa và xóa một cách đơn giản.

Tới đây sẽ có thông báo cho chúng ta xác nhận 1 lần nữa có muốn xóa lịch phim đó hay không.



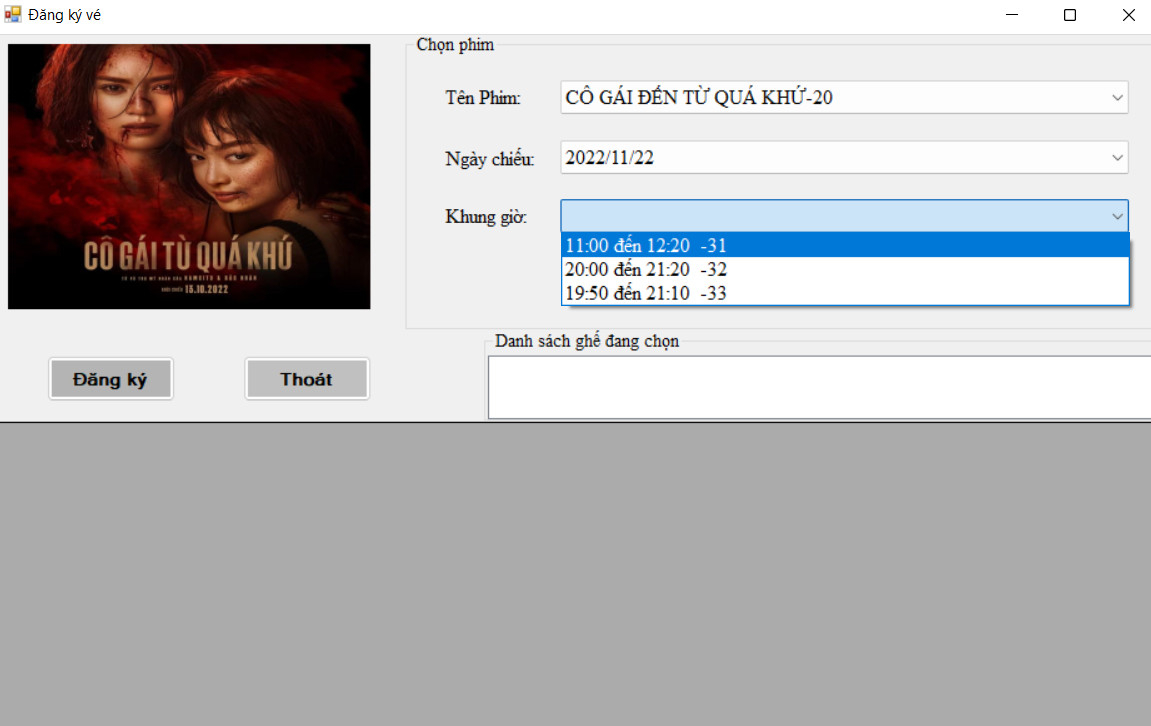
Hình 5.16 Xác nhận xóa 1 lịch phim

### *5.4.5. Đăng ký vé*

Ở phần mua vé đầu tiên người dùng sẽ chọn bộ phim và thời gian chiếu phim mình mong muốn sau đó là tiến hành chọn ghế ngồi và cuối cùng thanh toán hóa đơn.

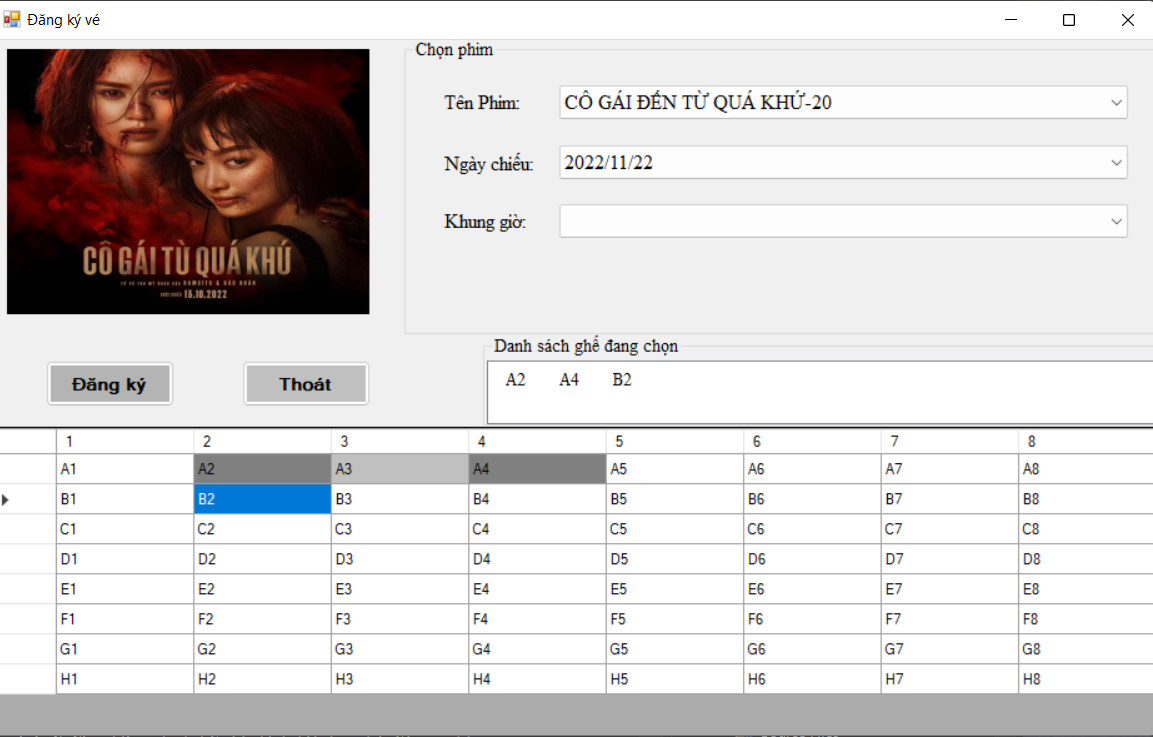
Sau khi nhấn chọn bộ phim mình mong muốn poster phim sẽ hiện ra hỗ trợ người dùng kiểm tra chắc chắn là bộ phim mình chọn.

Tiếp đó khách hàng chọn thời gian chiếu phim dựa trên những lịch chiếu được cài đặt lúc trước.



Hình 5.17 Đăng ký vé

Sau khi chọn đầy đủ các thông tin ở trên sơ đồ rạp phim sẽ xuất hiện, khách hàng sẽ tiến hành chọn ghế ngồi dựa trên sơ đồ rạp phim hiển thị trên màn hình.

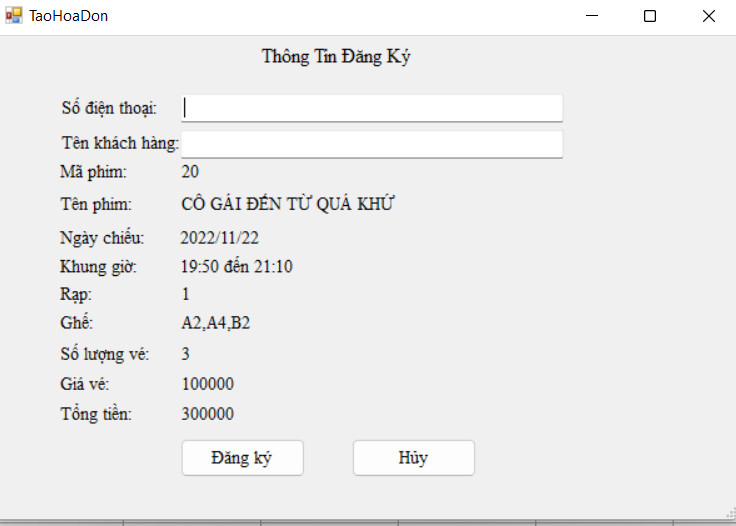


Hình 5.18 Sơ đồ chỗ ngồi

Nhấn vào vị trí ghế để chọn mua vé, sau khi nhấn vào ghế sẽ chuyển màu kèm theo hiển thị ghế chọn lên màn Danh sách ghế chọn. Nếu muốn bỏ chọn ghế thì nhấn vào ghế đó một lần nữa màu ghế sẽ thay đổi và màn hình Danh sách ghế chọn sẽ mất đi vị trí ghế đó.

Ở hình 4.18 màu xám đậm đang được lựa chọn và xuất hiện lưu tạm thời vào 1 danh sách trên bảng danh sách ghế đang chọn. Ghế A3 đã được hủy bỏ nên có màu nhạt hơn giúp cho người dung dễ nhận biết ghế nào vừa bỏ phân biệt được với ghế chưa từng chọn.

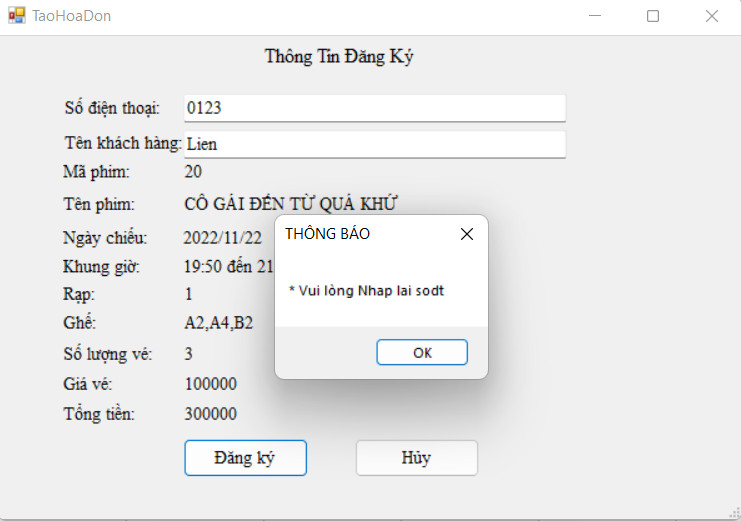
Sau khi hoàn thành chọn và nhấn Đăng Ký, hệ thống sẽ hiện form yêu cầu nhập thông tin khách hàng.



Hình 5.19 Thông tin đăng ký vé

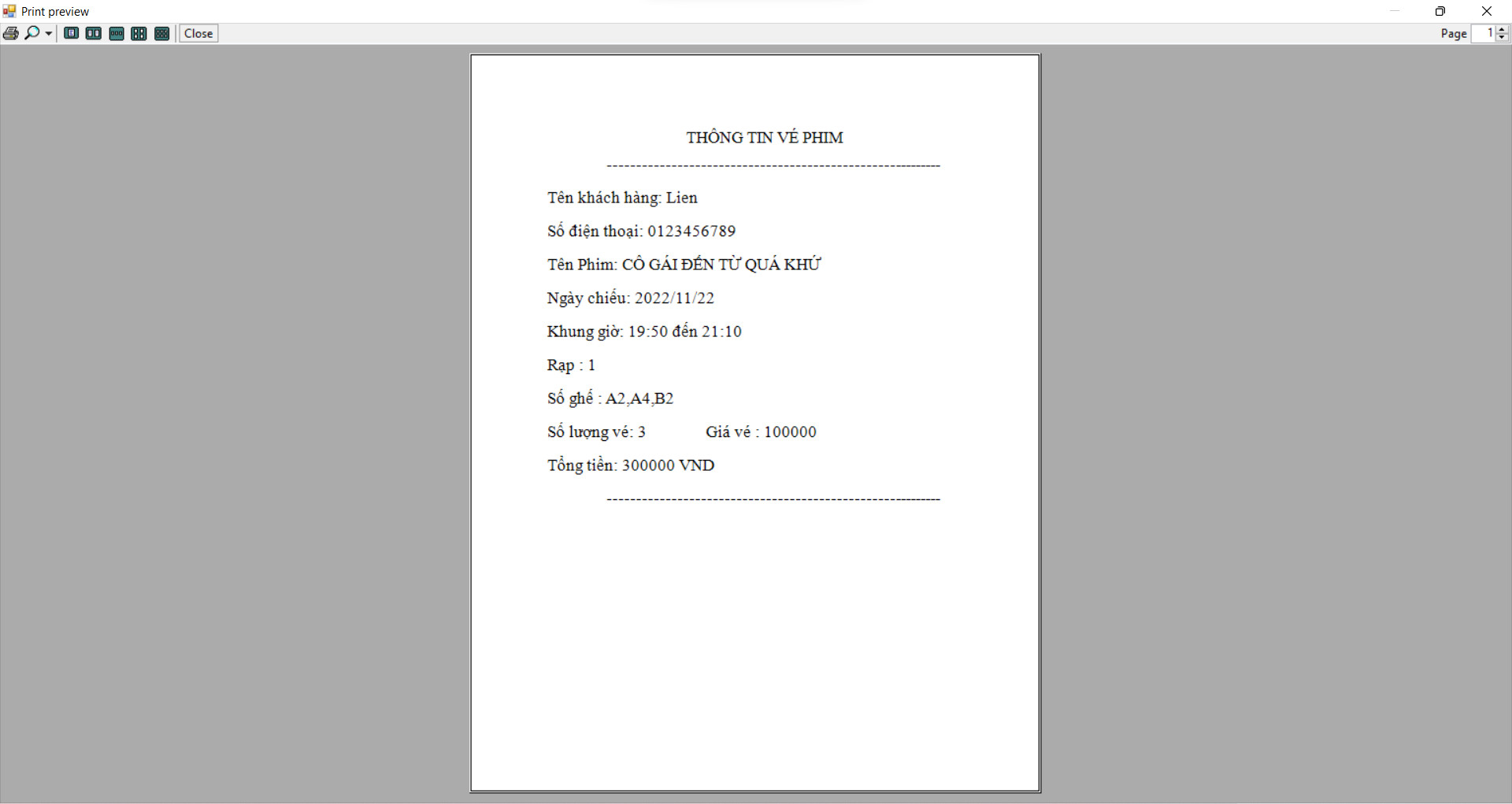
Hệ thống yêu cầu khách hàng nhập số điện thoại trước để kiểm tra nếu khách hàng đã từng mua vé thì hệ thống sẽ sử dụng thông tin cũ - Tên của khách hàng.

Số điện thoại được yêu cầu nhập 10-11 số, hệ thống sẽ báo lỗi nếu định dạng số điện thoại không đúng với yêu cầu. Sau khi nhập thông tin thành công hóa đơn sẽ được in ra.



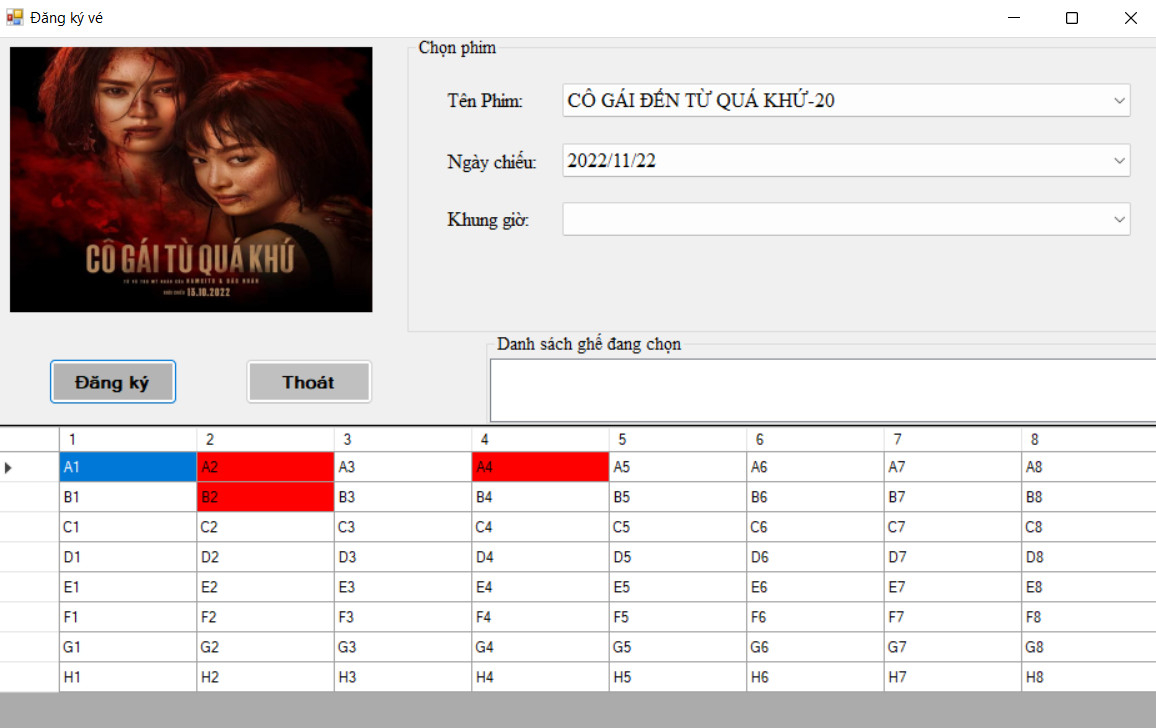
Hình 5.20 Thông báo nhập sai định dạng số điện thoại

Trên hóa đơn có tất cả thông tin gồm thông tin khách hàng: tên khách hàng, số điện thoại; thông tin phim gồm có: tên phim, ngày chiếu, giờ chiếu, số rạp, vị trí ghế, số lượng vé, giá vé và tổng tiền hóa đơn.



Hình 5.21 Hóa đơn mua vé

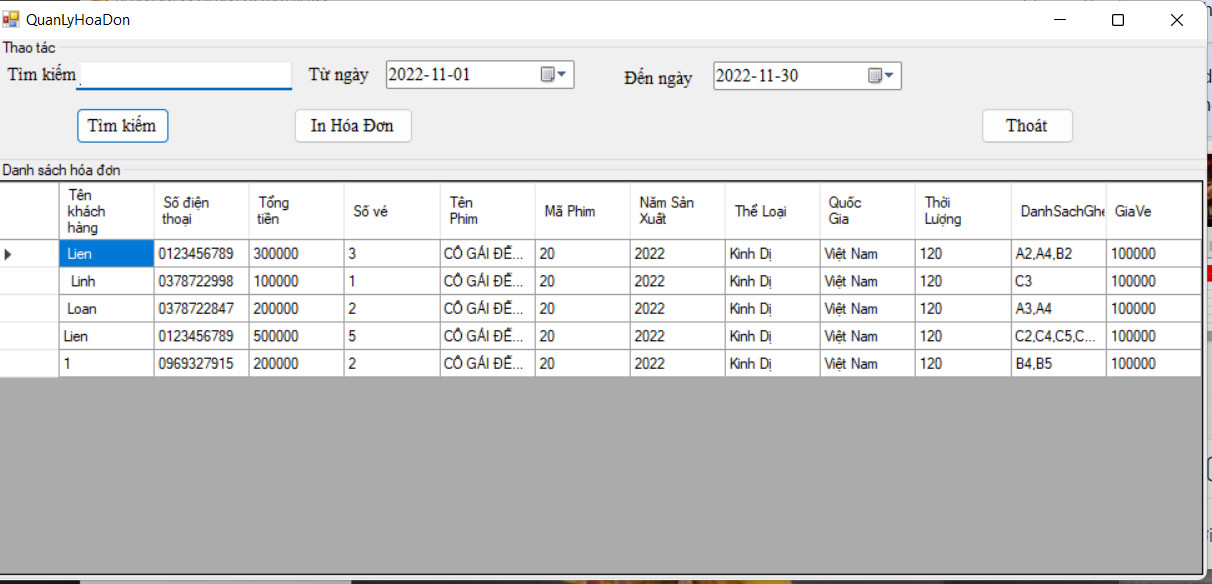
Những ghế được chọn sẽ đánh dấu đỏ để thông báo cho khách hàng rằng vị trí đó đã có người chọn, kèm theo đó nếu nhấn vào vị trí đánh dấu đỏ thì Danh sách ghế chọn không cập nhật.



Hình 5.22 Vị trí ghế đã mua

### *5.4.6. Quản lý hóa đơn*

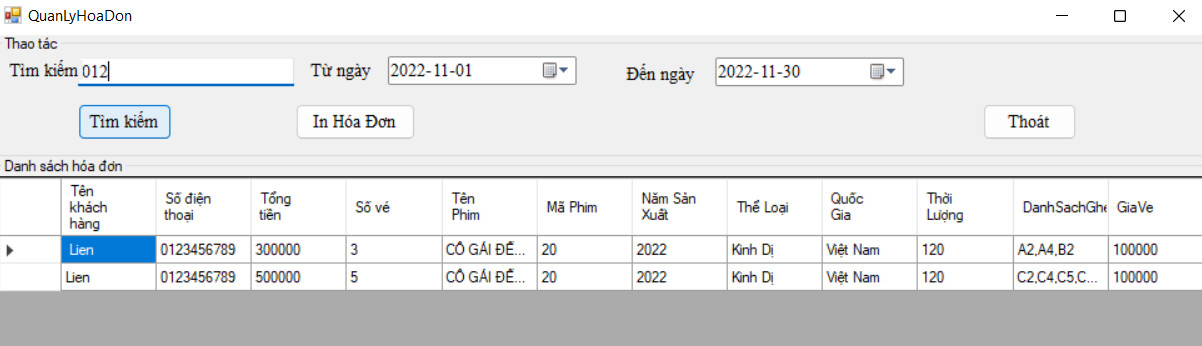
Ở mục này sẽ cho phép người dùng kiểm tra tất cả hóa đơn đã bán được, và thực hiện một số thao tác như tìm kiếm hóa đơn dựa theo thời gian và tìm kiếm hóa đơn theo số điện thoại, in lại hóa đơn mới.



Hình 5.23 Quản lý hóa đơn

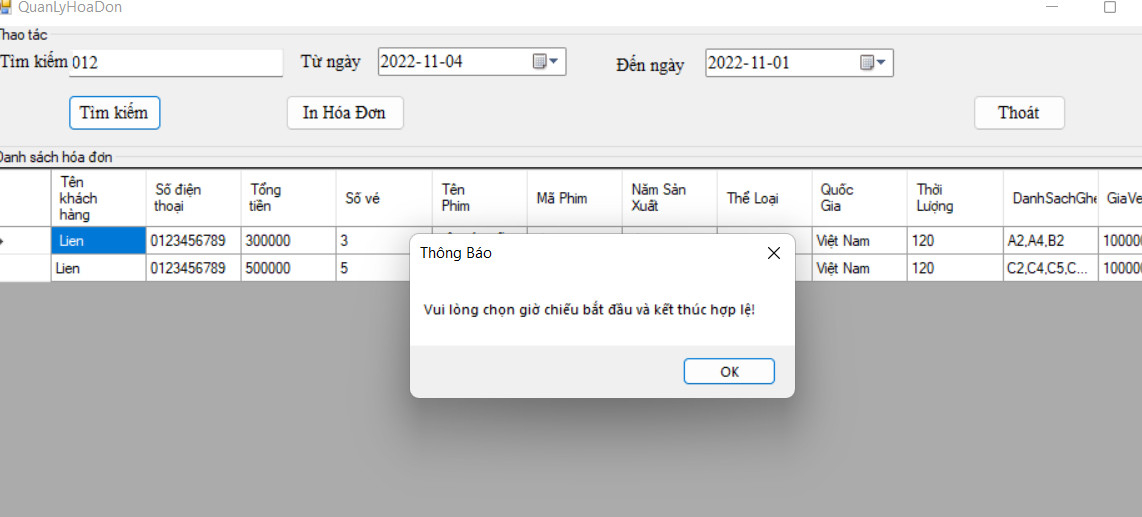
Để tối ưu thời gian tìm kiếm, ở mục tìm kiếm sẽ hiện ra những kết quả tìm được ngay lập tức dựa trên các ký tự người dùng nhập vào kể cả khi số điện thoại nhập chưa đủ.

Hình 4.23 thời gian khi load lên form sẽ được mặc định lấy theo tháng hiện tại và cập nhật từ đầu tháng đến cuối tháng. Nếu muốn tìm vào 1 khoảng thời gian ngắn hơn dựa trên số điện thoại đã sử dụng ở rạp nhiều tháng thì vẫn rất dễ tìm. Và không bị giới hạn về điều đó.



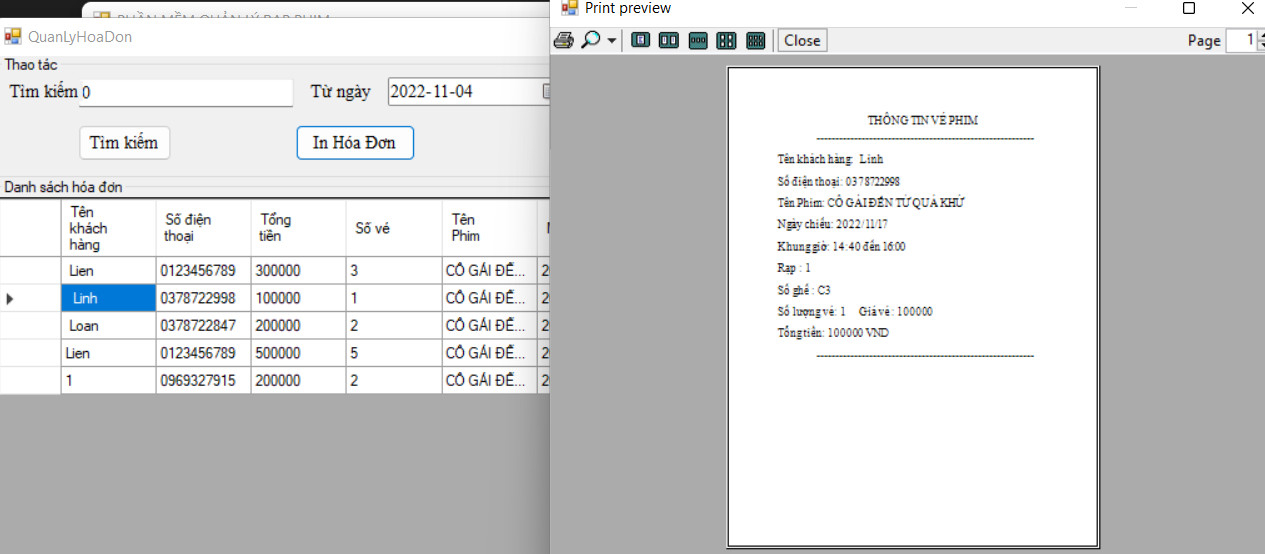
Hình 5.24 Tìm kiếm thông tin hóa đơn

Mục tìm kiếm bằng giờ chiếu cũng sẽ thông báo không hợp lệ và yêu cầu nhập lại nếu người dùng nhập thời gian kết thúc sớm hơn thời gian bắt đầu, có nghĩa là định dạng thời gian không phù hợp.



Hình 5.25 Thông báo nhập sai thời gian

Trong nhiều trường hợp khách hàng đặt vé trước nhưng đến giờ phim chiếu làm mất vé thì công cụ sẽ hỗ trợ in lại vé mới cho khách hàng. Việc này sẽ hỗ trợ tốt nhất cho khách hàng và tăng độ thân thiện giữa khách hàng và rạp phim.



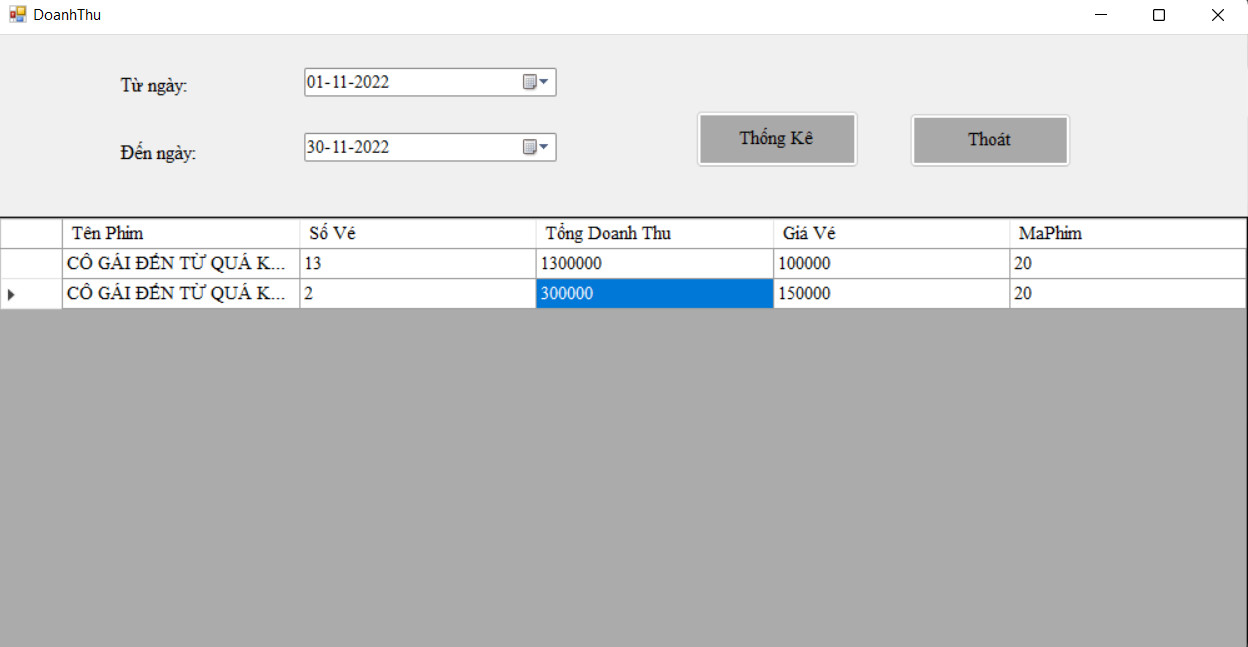
Hình 5.26 In lại hóa đơn

### *5.4.7. Doanh thu*

Chức năng quản lý doanh thu là một trong những chức năng mà bất cứ một cửa hàng nào cũng cần có.

Người dùng sẽ nhập khoảng thời gian muốn thống kê doanh thu, sau đó nhấn vào nút “Thống kê” để hệ thống tính số tiền thu được. Nếu một bộ phim mà có 2 giá vé khác nhau thì hệ thống sẽ thống kê 2 doanh thu khác nhau.

Ở khung thông tin sẽ chỉ hiện tên phim, số vé được mua, tổng doanh thu bán được, giá vé của bộ phim và mã phim.



Hình 5.27 Thống kê doanh thu

Chức năng này giúp chúng ta có thể thống kê tối đa các phim có lượng vé bán chạy hoặc thời gian nào có giá vé thấp hơn bán chạy. Có các giải pháp tối ưu để dễ dàng tiếp thị, marketing, đến gần hơn với khách hang. Tăng doanh thu, lợi nhuận cho rạp phim.

### *5.4.8. Đăng xuất*

Người dùng muốn đóng công cụ thì đơn giản chỉ cần nhấn vào nút đăng xuất ở form main. Công cụ sẽ lập tức đăng xuất tài khoản người dùng, màn hình Login sẽ hiện ra.

### *5.4.9. Nút Thoát*

Rất nhiều form cần đến nút này. Và điều dễ hiểu là nó giúp chúng ta tắt form đang thực thi một cách nhanh nhất.

# TÀI LIỆU THAM KHẢO

[1]. Sách: The C++ Programming Language, 4th Edition Always learning

tác giả: Stroustrup, nhà xuất bản: **Addison-Wesley**, 2013

[2]. Tìm hiểu về Github: <https://www.kenhgiaiphap.com/news/github-la-gi-cach-su-dung-github-chi-tiet-2ubc9q9>

[3]. Tài liệu C++: <https://www.w3schools.com/cpp/cpp_oop.asp>

[4] Video winform: <https://www.youtube.com/watch?v=gB51Tla5pPI>

# LINK GITHUB

Qr code

Description automatically generated