Федеральное государственное бюджетное образовательное учреждение

высшего образования

«**КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**»

Кафедра систем управления и вычислительной техники

|  |  |
| --- | --- |
| Курсовая работа  допущена к защите  руководитель………………… | Курсовая работа защищена  с оценкой …………………  руководитель………………… |

Справочная система «Киностудия»

Курсовая работа по дисциплине

“Разработка и стандартизация программных средств и информационных технологий”

Пояснительная записка

55.ВТ.12

Нормоконтроллер: Проект выполнил:

доцент, студент гр. 18-ВТ2

Высоцкий Л.Г. Калистый Никита

Калининград

2020

**Содержание**

[Программная система «Киностудия» Вариант 39 3](#_Toc61347293)

[1. Обоснование выбора языка Python для курсовой работы 4](#_Toc61347294)

[2. Техническое задание 5](#_Toc61347295)

[2.1.Основание для разработки 5](#_Toc61347296)

[2.2. Описание задания 6](#_Toc61347297)

[2.3. Постановка задачи 6](#_Toc61347298)

[2.4. Требования к программе 7](#_Toc61347299)

[2.4.1. Требования к функциональным характеристикам 7](#_Toc61347300)

[3. Дерево функций проекта 7](#_Toc61347301)

[4. Дерево модулей программной системы 8](#_Toc61347302)

[5. Cтруктуры используемых данных 9](#_Toc61347303)

[6. Распределение функций системы по модулям 11](#_Toc61347304)

[7. Видео-формы пользовательского интерфейса 13](#_Toc61347305)

[8. Описание интерфейса. 23](#_Toc61347306)

[9. Блок-схемы процедур 26](#_Toc61347307)

[10. Листинг программы 35](#_Toc61347308)

[11. Руководство системного программиста 145](#_Toc61347309)

[11.1. Требования к информационной и программной совместимости 145](#_Toc61347310)

[11.2. Требования к составу и параметрам технических средств 145](#_Toc61347311)

[11.3. Установка компонентов для корректной работы программы 145](#_Toc61347312)

[11.4 Безопасность ПС «Киностудия» 147](#_Toc61347313)

[12. Руководство пользователя 148](#_Toc61347314)

[13. Вывод 174](#_Toc61347315)

[14. Список литературы 175](#_Toc61347316)

# Программная система «Киностудия» Вариант 39

Функции (сервисы) системы:

1. Просмотр информации о фильмах
2. Добавление фильмов
3. Редактирование данных о фильмах
4. Добавление актеров
5. Редактирование данных об актерах
6. Вход в режим администратора
7. Выход из режима администратора
8. Показ сведений о разработчике ПС

Ограничения на сервисы

1. Система предусматривает два режима работы: пользователь и администратор. У администратора есть все доступные функции системы, у пользователя только три функции: просмотр информации о фильмах, просмотр информации об актерах и вход в режим администратора.
2. ПС защищена паролем, который необходимо менять каждые 10 дней.

Руководитель КР

доц. Высоцкий Л.Г.

Студент гр. 18-ВТ-1

Калистый Н.С

11 декабря 2020

1. Обоснование выбора языка Python для курсовой работы

Для разработки программной системы «ПС киностудия» был выбран высокоуровневый язык написания программного кода Python в связи с тем, что этот язык программирования не требует платы за использование, обладает большим количеством библиотек, код на этом языке стабильно и безопасно выполняется, синтаксис ядра Python минималистичен. Согласно «Дзену Python**»**, этот язык нацелен на повышение производительности разработчика и читаемости кода.

В то же время стандартная библиотека включает большой объём полезных функций». К дополнительным преимуществам относятся: поддержка структурного, объектно-ориентированного, функционального, императивного и аспектно-ориентированного программирования. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты. Большая библиотека поддержки предоставляет массу возможностей, востребованных в прикладных программах, а качество программного обеспечения делает данный язык простым в изучении, понимании и использовании.

Так как Python предусматривает наличие стандартной библиотеки tkinter, которая используется для написания графических интерфейсов. В работе также используется библиотека pyodbc, служащая для взаимодействия ЯП Python с СУБД MS Access, в которой находятся данные информационной системы. Также используется билиотека tkcalendar для упрощения пользовательского ввода дат, библиотека datetime для работы с датами, библиотека os для подключения файла базы данных.

После представления и согласования требований к программной системе, была начата работа над ее реализацией. В курсовой работе «ПС Киностудия» предпологалось создание программной системы киностудии, в которой можно получить информацию о фильмах, которые были сняты в ней сняты, а также актерах, которые участвовали в съемках. В качестве исходных данных использовалась киностудия Warner Bros. (Warner Bros. Pictures). «ПС Киностудия» можно разделить на следующие логические блоки:

1) Просмотр пользователем списка фильмов и актеров

2) Просмотр пользователем информации о разработчике ПС

3) Режим Администратора:

3.1) Добавление информации о фильме

3.2) Редактирование (удаление) информации о фильме

3.3) Добавление информации об актере

3.4) Редактирование (удаление) информации об актере

Пункты 3.1-3.4 выполняются после входа в Режим администратора

Так был сделан выбор языка, а также разделена поэтапная структура ее работы.

# Техническое задание

## 2.1.Основание для разработки

Программа должна соответствовать заданию на курсовое проектирование, которое было выдано доцентом кафедры «Систем управления и вычслительной техники» Высоцким Л. Г. Данное задание было опубликовано в виде методических указаний для выполнения курсовой работы в электронной информационной образовательной среде (ЭИОС).

## 2.2. Описание задания

ПС «Киностудия»: данные актера (имя, дата рождения, пол, фильмы, роль в фильме), сведения о фильмах, снятых в студии (название, год выпуска, длительность, жанр, возрастные ограничения, стоимость, награды).

Запросы: актер, фильм, фильмы одного жанра, фильмы до/после года.

Два уровня доступа: администратор с паролем и пользователь.

Два режима доступа:

администратор с возможностью добавлять, просматривать, редактировать и удалять информацию о фильмах, актерах,

обычный пользователь с возможностью просматривать информацию о фильмах, актерах.

## 2.3. Постановка задачи

Данная программа должна:

* Выводить окно с фамилией и группой студента, чтобы подтвердить, что программа была создана самостоятельно.
* Корректно собирать и отображать данные в окне ПС, чтобы пользователь мог сделать выводы, о правильном вводе и размещении их внутри базы данных.
* Исполнять Запросы: актер, фильм, фильмы одного жанра, фильмы до/после года.
* Поддерживать 2 режима: режим администратора с возможностью добавлять, редактировать и просматривать информацию о фильмах, и пользователь с возможностью просматривать информацию о фильмах, актерах.

## 2.4. Требования к программе

## 2.4.1. Требования к функциональным характеристикам

Программа должна:

* Реагировать на действия пользователя в графическом интерфейсе ПС и выводить соответствующую информацию.
* Собирать и обрабатывать входящие от пользователя данные, проверять их на допустимость, не допуская попадания в базу данных некорректных данных, сообщать об ошибках ввода.
* Содержать информацию о студенте.
* Обеспечивать защиту ПС и базы данных от несанкционированного доступа к данным в базе данных с помощью парольной защиты

# Дерево функций проекта

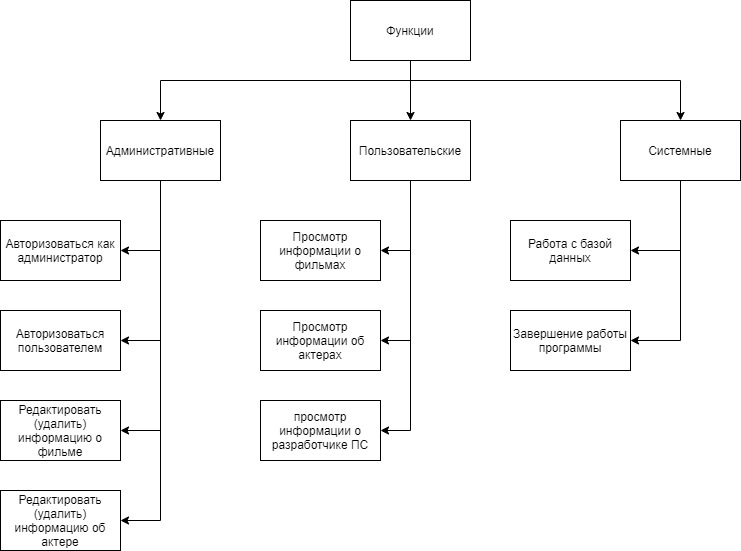


Рис. 1. Дерево функций проекта

# Дерево модулей программной системы

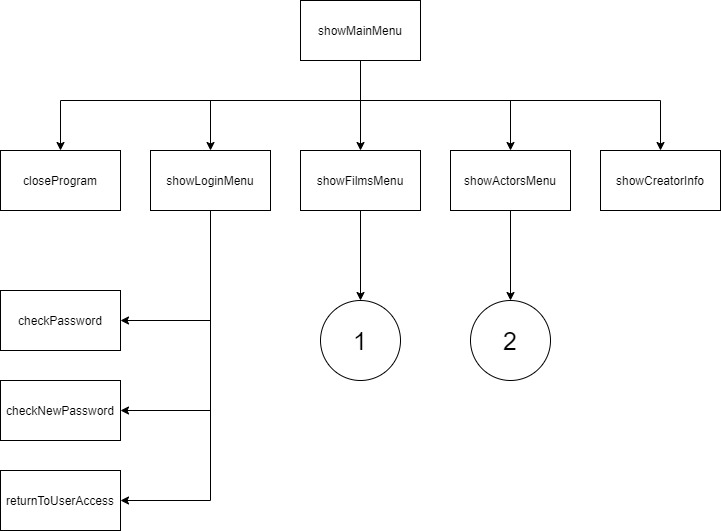


Рис.2. Дерево модулей программной системы.

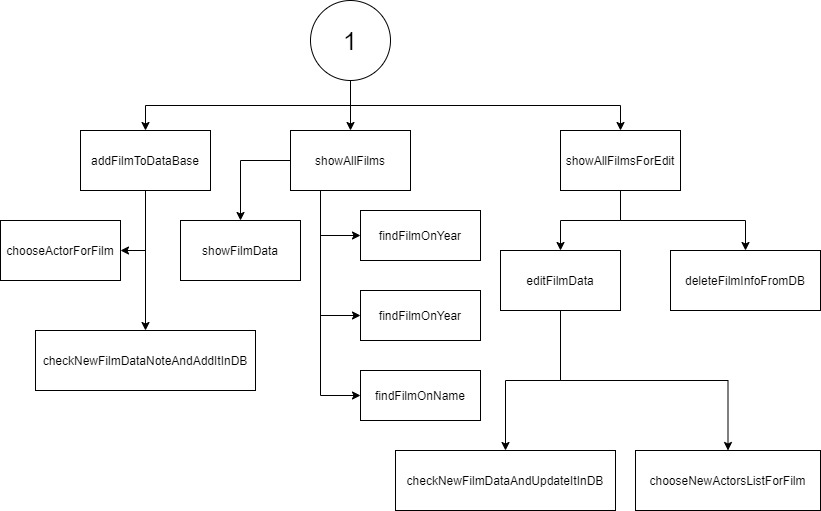
**

Рис.2. Дерево модулей программной системы. Продолжение.

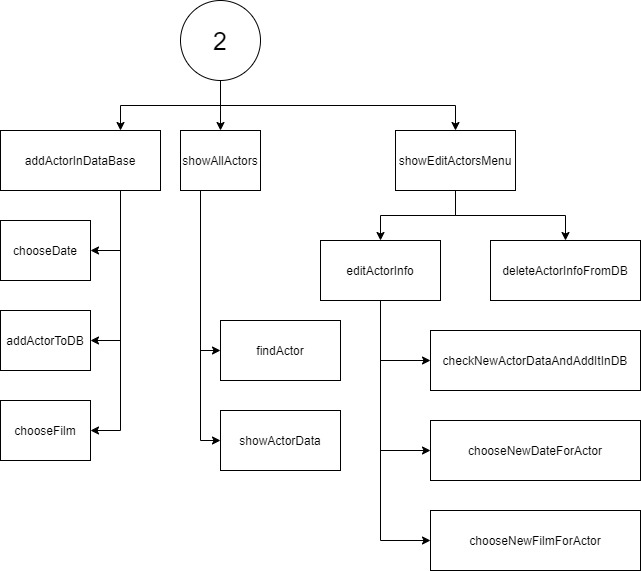
**

Рис.2. Дерево модулей программной системы. Продолжение.

# Cтруктуры используемых данных

База данных состоит из трех основных таблиц и приведина на рис. 3.

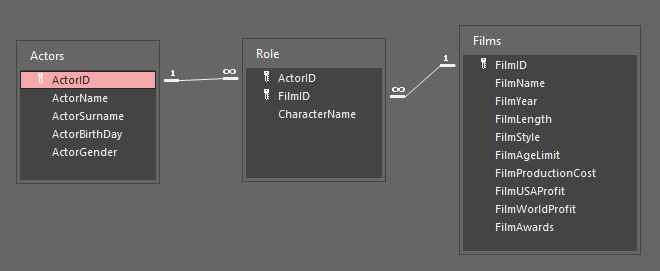


Рис. 3. Схема данных базы данных.

Описание таблиц базы данных представлено ниже.

Таблица «Actors» (таблица 1) содержит в себе сведения об актерах.

Таблица «Films» (таблица 2) содержит в себе сведения о фильмах.

Таблица «Role» (таблица 3) – это вспомогательная таблица между таблицей «Actors» и «Films», имееет составной ключ из «ActorID» и «FilmID», она используется для того, чтобы обеспечить связь между фильмами и актерами, ее необходимость обусловлена тем, что, по физической природе предметной области, каждый фильм может иметь в актерском составе несоклько актеров, а каждый из актеров в базе данных может сниматься в нескольких фильмах. Чтобы избавиться от отношения «многие ко многим» между таблицами «Actors» и «Films», введена таблица «Role».

Табл.1. - Таблица «Actors»

|  |  |  |  |
| --- | --- | --- | --- |
| Данные актера (Actors) | | | |
| Название | Название в БД | Тип данных | Обязательность |
| ID Актера | ActorID | Int, 0.. 2147483647 | Обязательное |
| Имя актера | ActorName | String, 63 | Обязательное |
| Фамилия актера | ActorSurname | String, 63 | Обязательное |
| Дата рождения | ActorBirthDay | String, 10 | Обязательное |
| Пол | ActorGender | Char | Обязательное |

Табл.2. – Таблица «Films»

|  |  |  |  |
| --- | --- | --- | --- |
| Сведения о фильме (Films) | | | |
| Название | Название в БД | Тип данных | Обязательность |
| ID фильма | FilmID | Int, 0.. 2147483647 | Обязательное |
| Название фильма | FilmName | String, 0.. 255 | Обязательное |
| Год выпуска | FilmYear | Int, 1880.. 2021 | Обязательное |
| Длительность в минутах | FilmLenght | Int, 1.. 2147483647 | Обязательное |
| Жанр | FilmStyle | String, 255 | Обязательное |
| Возрастное ограничение | FilmAgeLimit | Int, 0.. 18 | Обязательное |
| Стоимость производства | FilmProductionCost | Int, 1.. 2147483647 | Обязательное |
| Сборы в США | FilmUSAProfit | Int, 0.. 2147483647 | Обязательное |
| Сборы в мире | FilmWorldProfit | Int, 0.. 2147483647 | Обязательное |
| Награды | FilmAwards | String, 0.. 3000 | Обязательное |

Табл.3. – Таблица «Role»

|  |  |  |  |
| --- | --- | --- | --- |
| Роли (Role) | | | |
| Название | Название в БД | Тип данных | Обязательность |
| ID Актера | ActorID | Int, 0.. 2147483647 | Обязательное |
| ID фильма | FilmID | Int, 0.. 2147483647 | Обязательное |
| Персонаж | CharacterName | String, 255 | Обязательное |

# Распределение функций системы по модулям

Блок1 – Запуск главного окна приложения.

Блок2 – запуск меню информации о разработчике ПС.

Блок3 – запуск меню информации о фильмах.

Блок4 – запуск меню информации об актерах.

Блок5 – запуск меню авторизации.

Блок6 – отрисовка окна просмотра всех фильмов

Блок7 – переход к просмотру конкретного фильма

Блок8 – переход от конкретного фильма к конкретному актеру, который в этом фильме снимался

Блок9 – переход от конкретного актера к списку всех актеров

Блок10 – отрисовка окна поиска фильма по названию

* Переход к фильму с заданным названием

Блок11 отрисовка окна поиска фильма по жанру

* Переход к фильму с заданным жанром

Блок12 - отрисовка окна поиска фильма по году

* Переход к фильму с заданным годом

Блок14 отрисовка окна добавления фильма.

* Проверка всех полей информации о фильме
* Проверка роли каждого добавляемого актера
* Добавление фильма в БД

Блок15 отрисовка окна редактирования фильма.

* Проверка всех полей информации о фильме
* Проверка роли каждого добавляемого актера
* Сохрание изменений и занесение их в БД

Блок16 – Отрисовка окна просмотра всех актеров.

Блок17 – Просмотр информации выбранного актера.

Блок18 – Отрисовка окна поиска актера по имени и фамилии.

* Поиск актера по имени и фамилии

Блок19 – Вызов окна добавления актера в ПС.

* Проверка всех полей информации об актере
* Проверка роли актера в каждом фильме
* Сохрание изменений и занесение их в БД

Блок20 – Вызов окна редактирования информации об актере.

* Редактирование информации об актере, проверка вводимых данных и их добавление в ПС
* Удаление всей информации об актере из ПС

Блок21 – Вызов окна авторизации

Блок22 – Проверка введенного пароля на соотвествие, проверка не просрочен ли пароль.

Блок23 – Смена пароля.

# Видео-формы пользовательского интерфейса

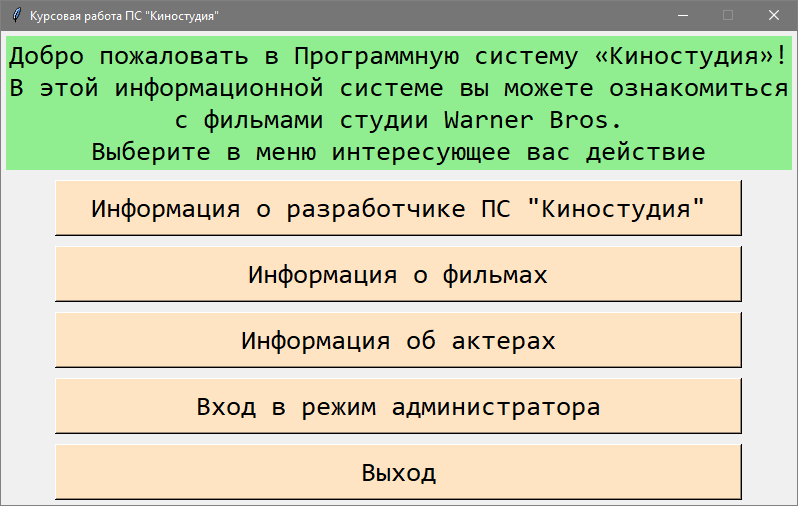


Рис. 4. Главное меню программы в режиме пользователя и администратора

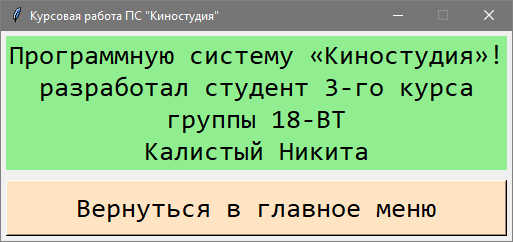


Рис. 5. Информация о разработчике ПС.

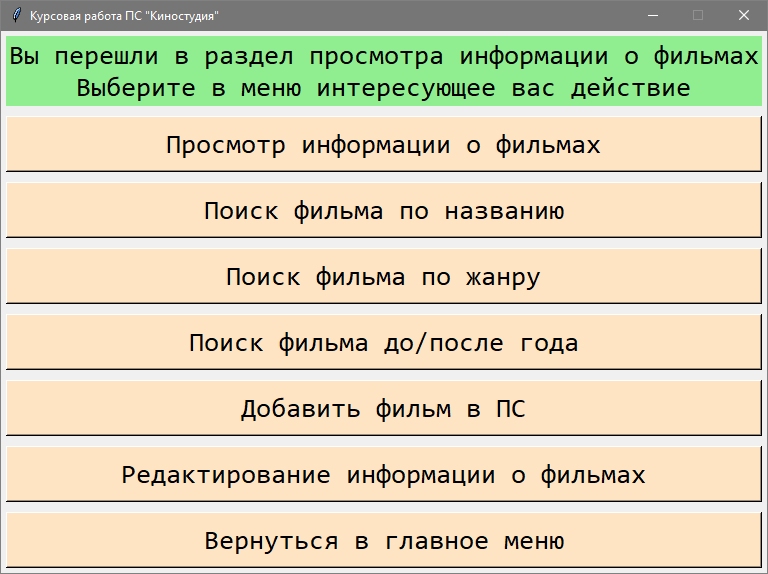


Рис. 6. Меню информации о фильмах в режиме администратора.

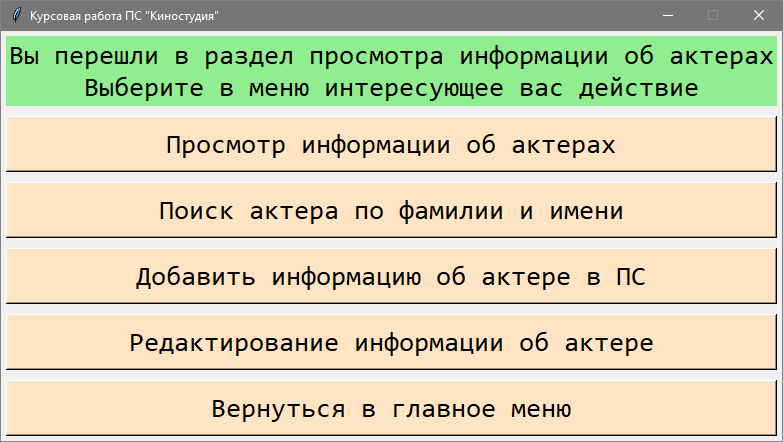


Рис. 7. Меню информации об актерах в режиме администратора.

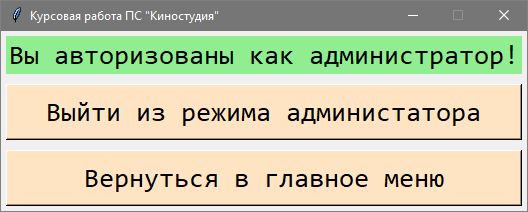


Рис. 8. Меню авторизации в режиме администратора.

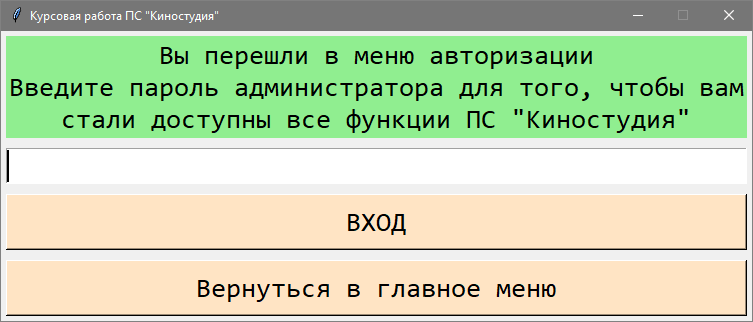


Рис. 9. Меню авторизации в режиме пользователя.

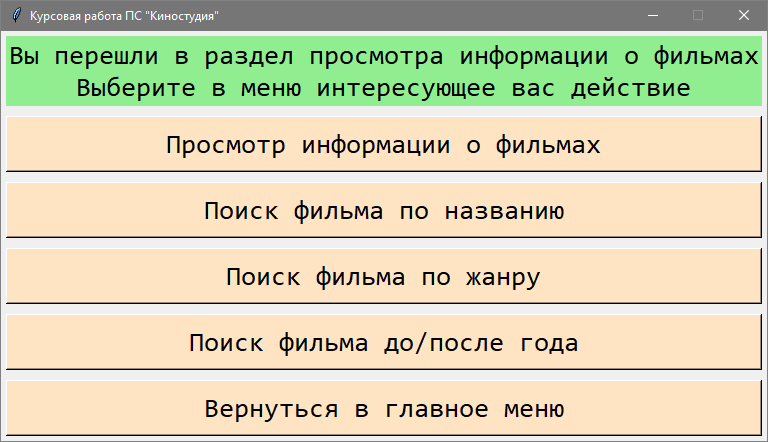


Рис. 10. Меню информации о фильмах в режиме пользователя

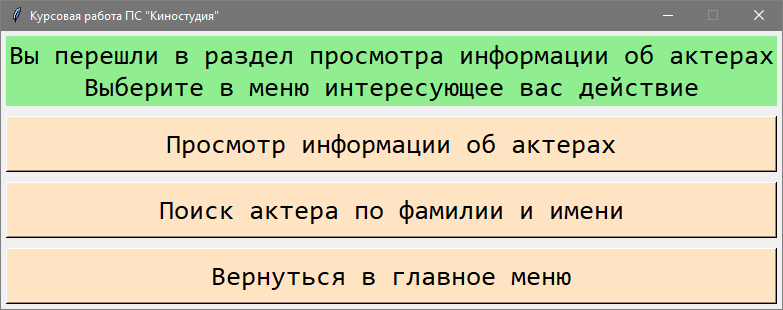


Рис. 11. Меню информации об актерах в режиме пользователя

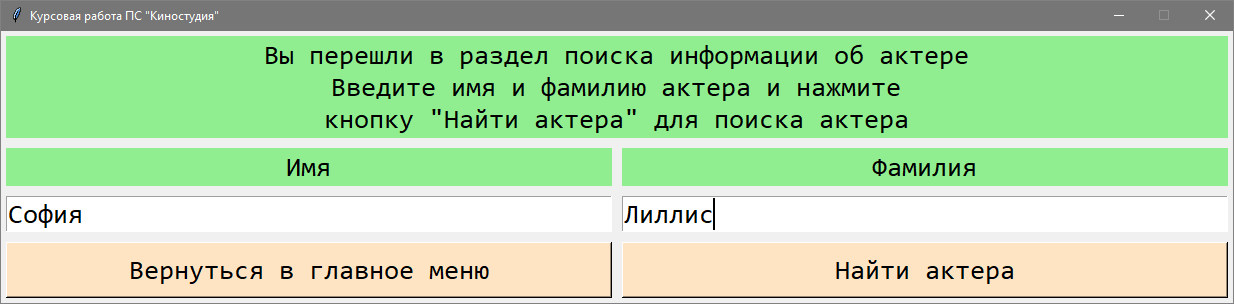


Рис. 12. Окно поиска актера по имени и фамилии

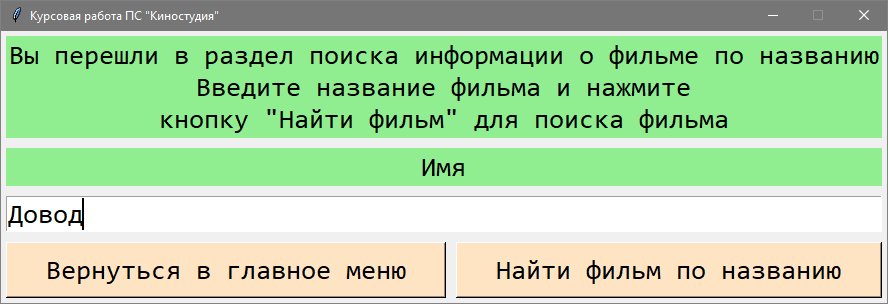


Рис. 13. Окно поиска фильма по названию.



Рис. 14. Окно просмотра всех фильмов

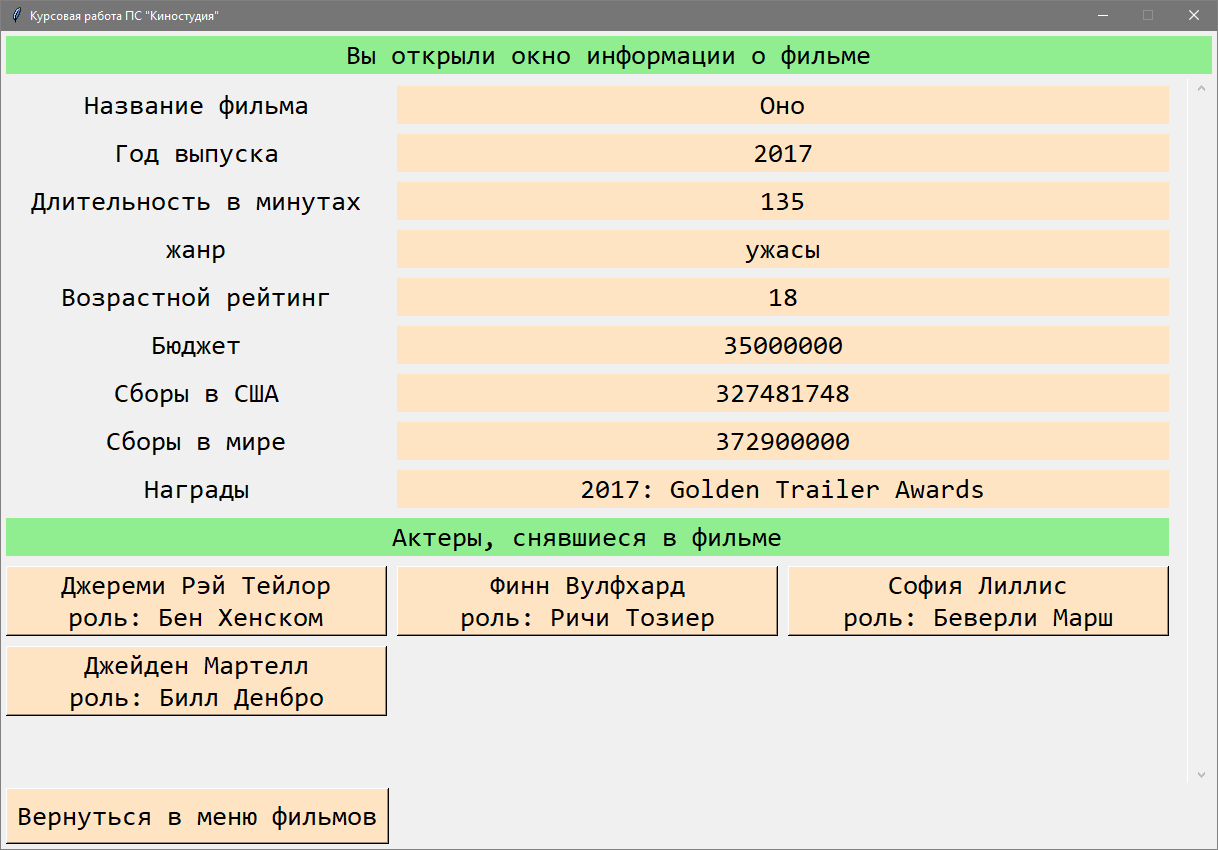


Рис. 15. Окно просмотра информации выбранного фильма

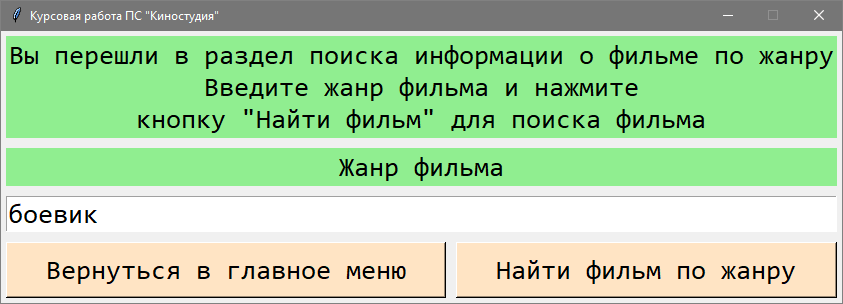


Рис. 16. Окно поиска фильма по жанру

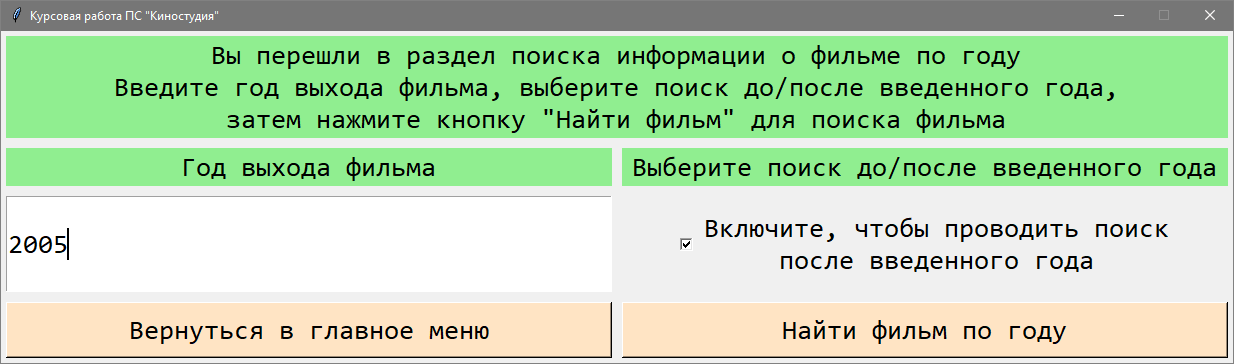


Рис. 17. Окно поиска фильма по году выпуска

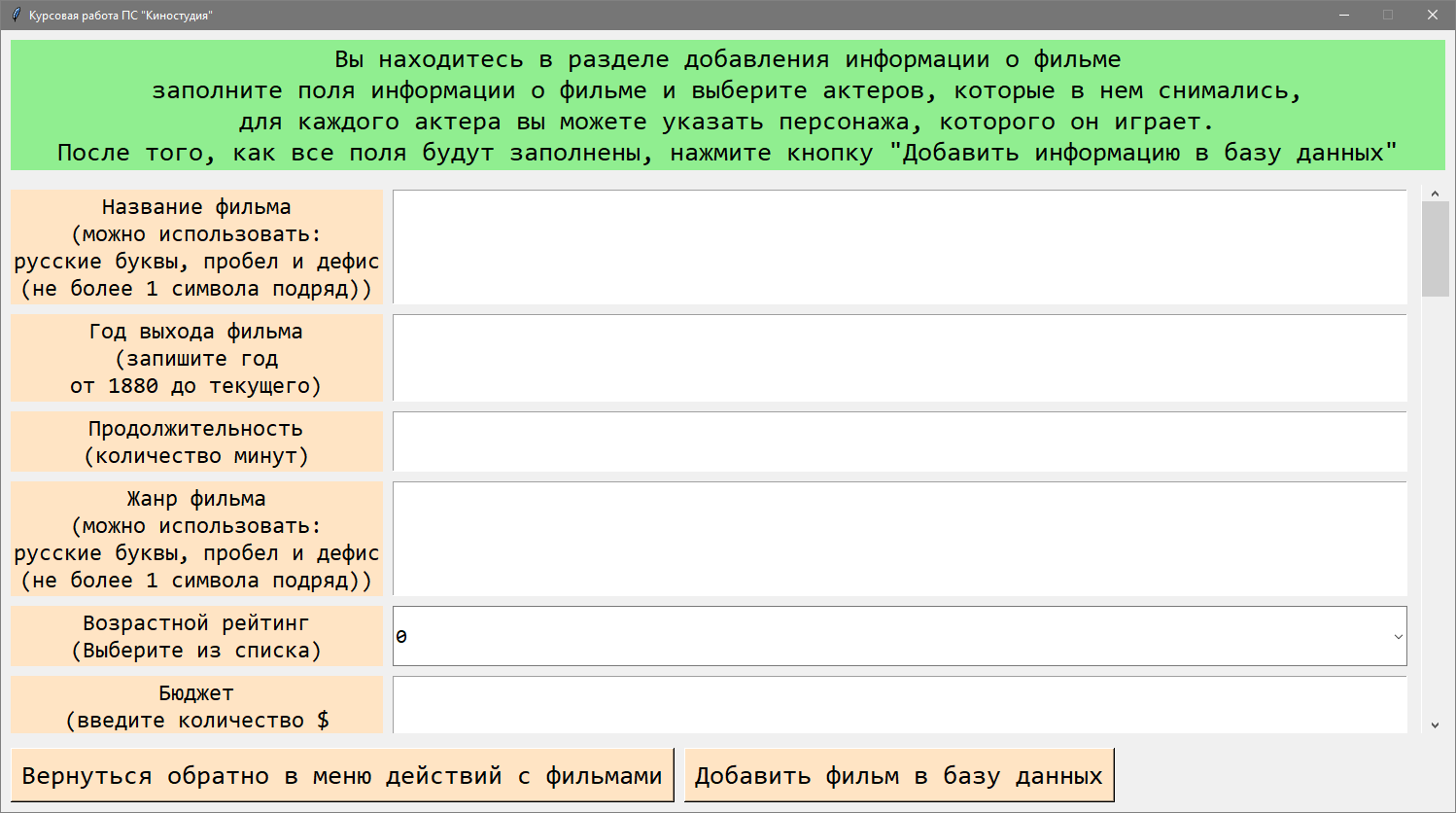


Рис. 18. Окно добавления фильма в базу данных



Рис. 19. Окно со всеми фильмами для редактирования

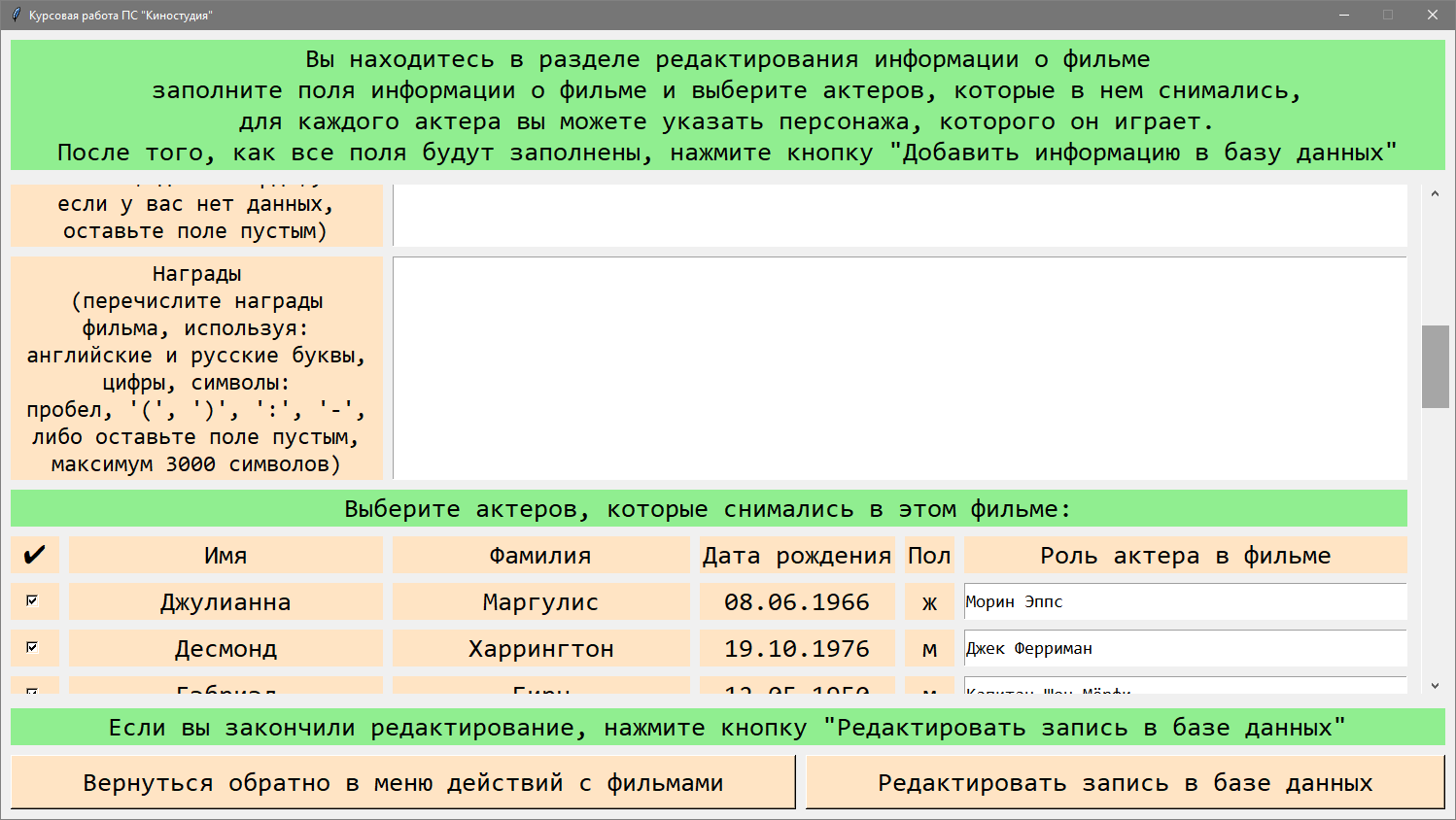


Рис. 20. Окно редактирования данных фильма

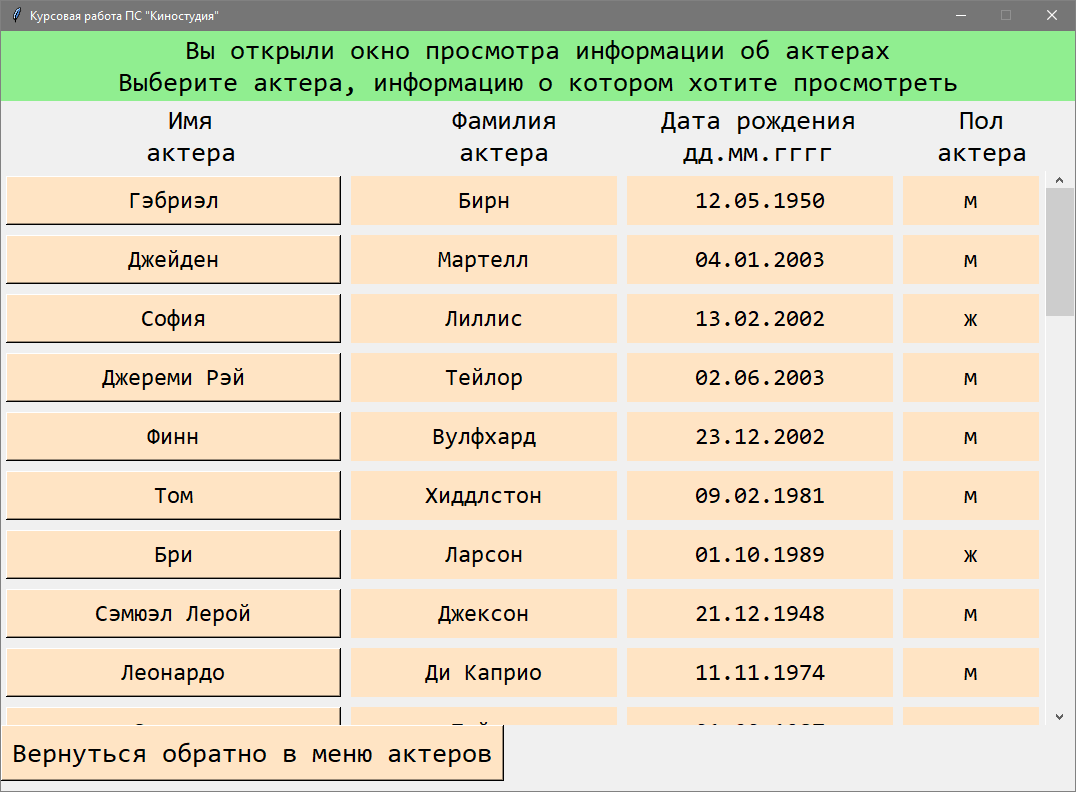


Рис. 21. Окно просмотра информации о всех актерах

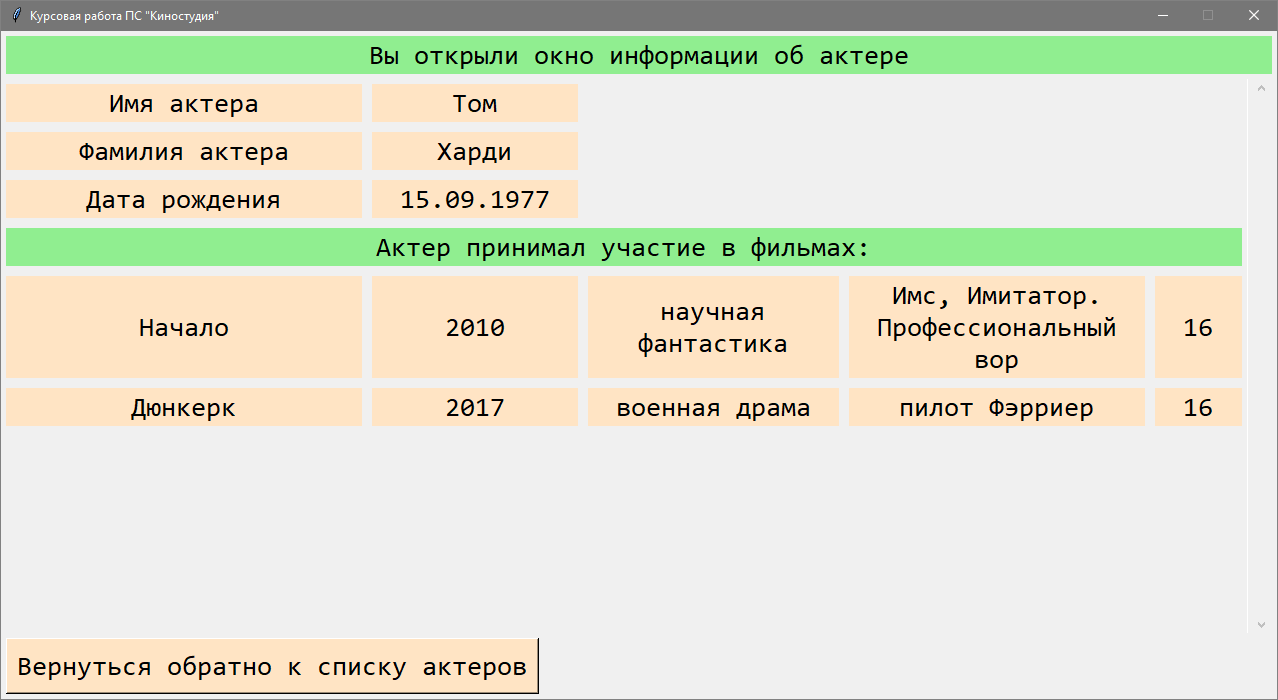


Рис. 22. Окно просмотра информации об актере

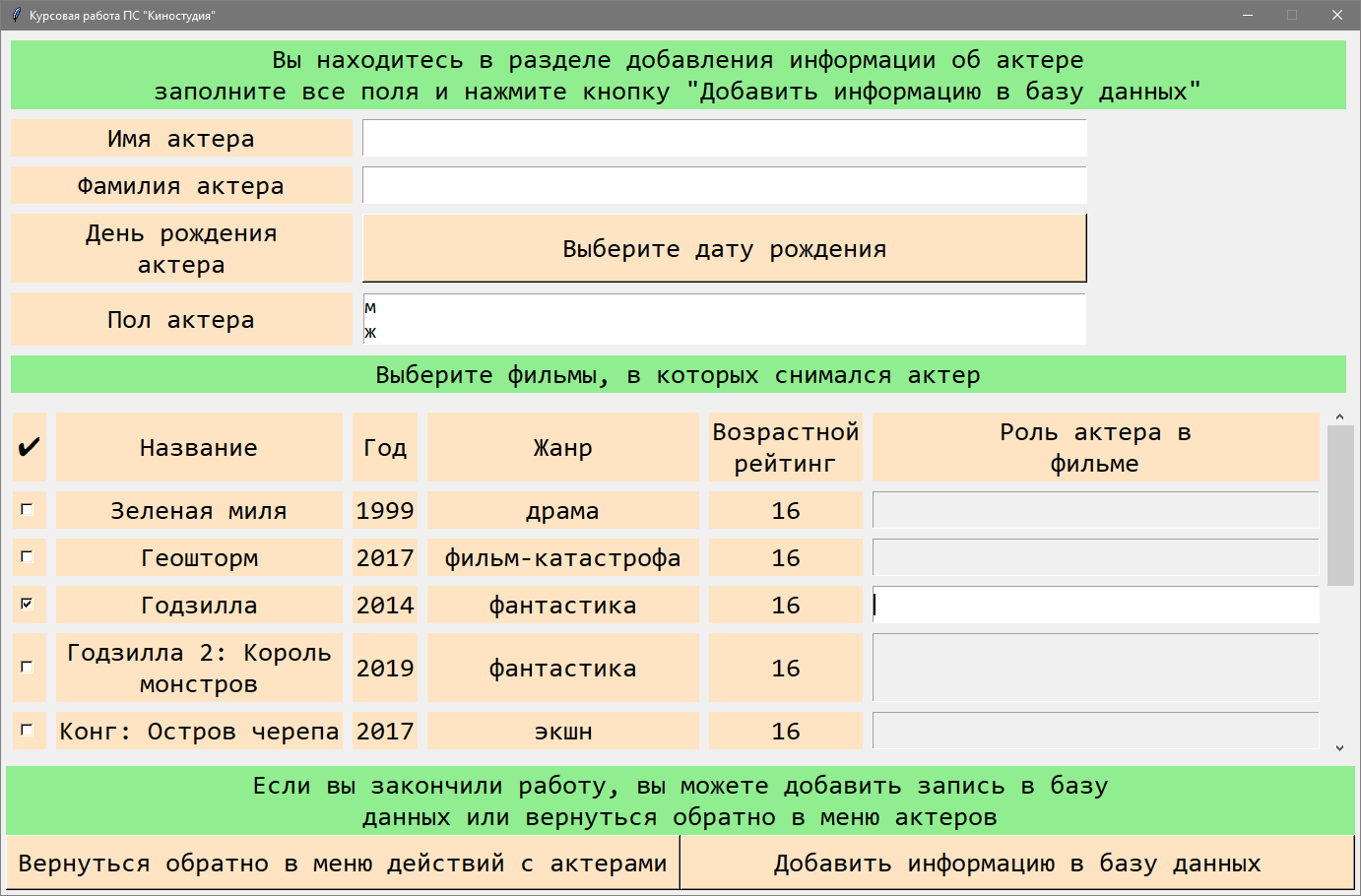


Рис. 23. Окно добавления информации об актере

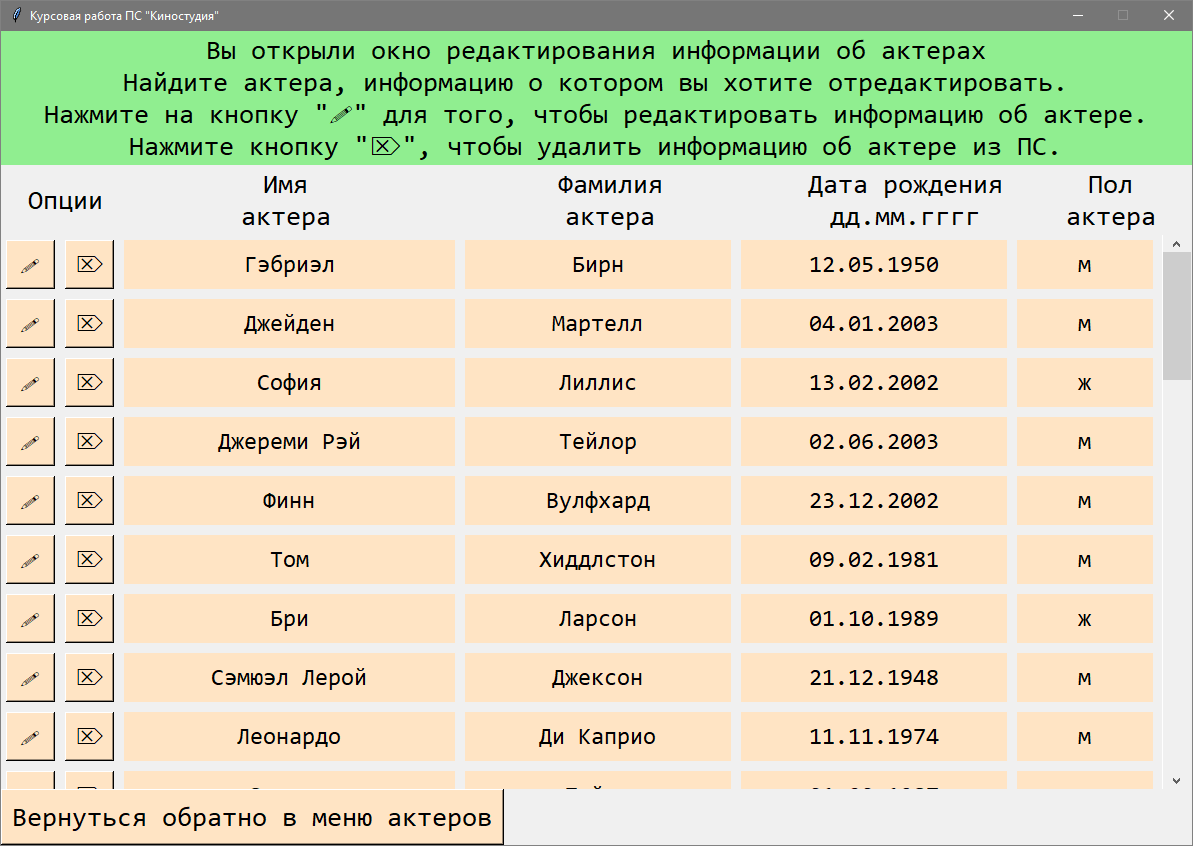


Рис. 24. Окно списка актеров для редактирования

1. Описание интерфейса.

На рис. 4 изображено главное меню программы, оно общее для всех пользователей, с его помощью вызываются все функции ПС.

На рис. 5 изображено окно с выведенной информацией о разработчике ПС.

На рис.6 изображено меню информации о фильмах в режимах администратора. С помощью этого меню производится просмотр информации о фильмах, выборка нужного фильма по запросу, а также выполняется редактирование и добавление данных о фильмах.

На рис. 7 изображено меню просмотра информации обо всех актерах. С помощью этого меню выполняются все манипуляции с актерами: просмотр списка актеров, поиск актера по имени и фамилии, добавление новых актеров в пс и редактирование информации об актерах.

На рис. 8 изображено меню авторизации в режиме администратора. С помощью этого меню происходит смена прав пользователя.

На рис. 9 изображено меню авторизации в режиме пользователя, с помощью этого меню происходит авторизация в ПС и открывается доступ к редактированию и удалению данных из БД.

На рис. 10 изображено меню информации о фильмах в режиме пользователя. Среди кнопок нет кнопок редактирования и добавления фильмов.

На рис. 11 изображено меню информации об актерах. С его помощью осуществляются действия по просмотру информации об актерах, выборка актера по его имени и фамилии, добавление актеров в базу данных и их редактирование.

На рис. 12 изображено окно поиска актера по имени и фамилии.

На рис. 13 изображено окно с запросом фильма по его названию.

На рис. 14 изображено окно просмотра списка всех фильмов. С его помощью можно выбрать фильм и просмотреть детальную информацию о нем, а также список актеров, которые в нем снимались.

На рис. 15 изображено окно прсомотра информации выбранного фильма.

На рис. 16 изображено окно, реализующее запрос фильма по жанру.

На рис. 17 изображено окно, реализующее запрос фильма по году выхода.

На рис. 18 представлено окно добавления фильма в базу данных. Администратор должен заполнить все информационные поля фильма, а также может выбрать актеров, которые снимались в этом фильме, уазать какие роли они играли.

На рис. 19 изображено окно со списком фильмов для редактирования. В этом окне пользователь может редактировать информацию о фильме, нажав на карандаш, либо удалить фильм из базы данных, нажав на символ удаления.

На рис. 20 представлено непосредственно окно редактирования фильма. Даные в поля информации о фильме будут подставлены автоматически, после чего администратор может внести правки в данные о фильме, изменить список актеров, играющих в фильме или их роли, и сохранить изменения в базе данных.

На рис. 21 представлено окно просмотра информации о всех актерах. В этом окне пользователь может просмотреть список актеров в ПС, а также перейти к конкретному актеру и просмотреть его полные данные, роли в фильмах.

На рис. 22 представлено окно просмотра информации об актере, роли в которых он снимался.

На рис. 23 представлено окно добавления информации об актере в ПС. Администратор должен заполнить все поля актера, и может выбрать фильмы и написать роли, которые добавляемый актер играл в этих фильмах.

На рис. 24 представлено окно списка актеров для редактирования. С помощью этого окно выполняется редактирование информации актеров, удаление их из базы данных.

# Блок-схемы процедур

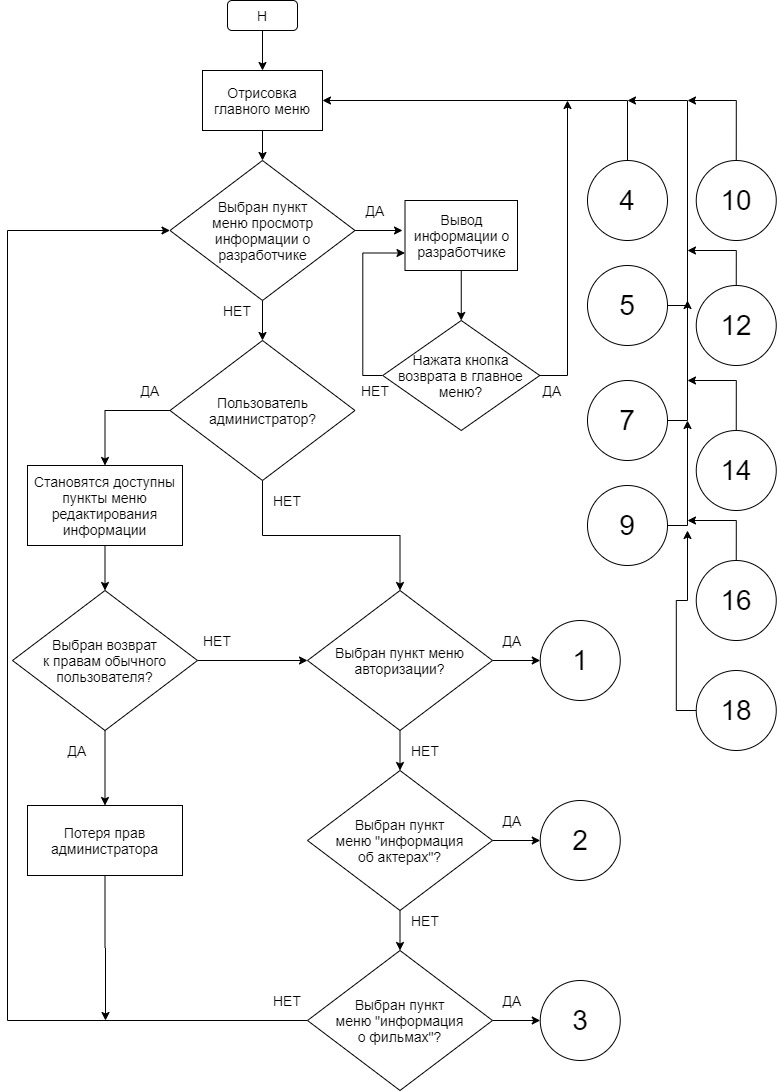
**

Рис.25. Блок-схема программы

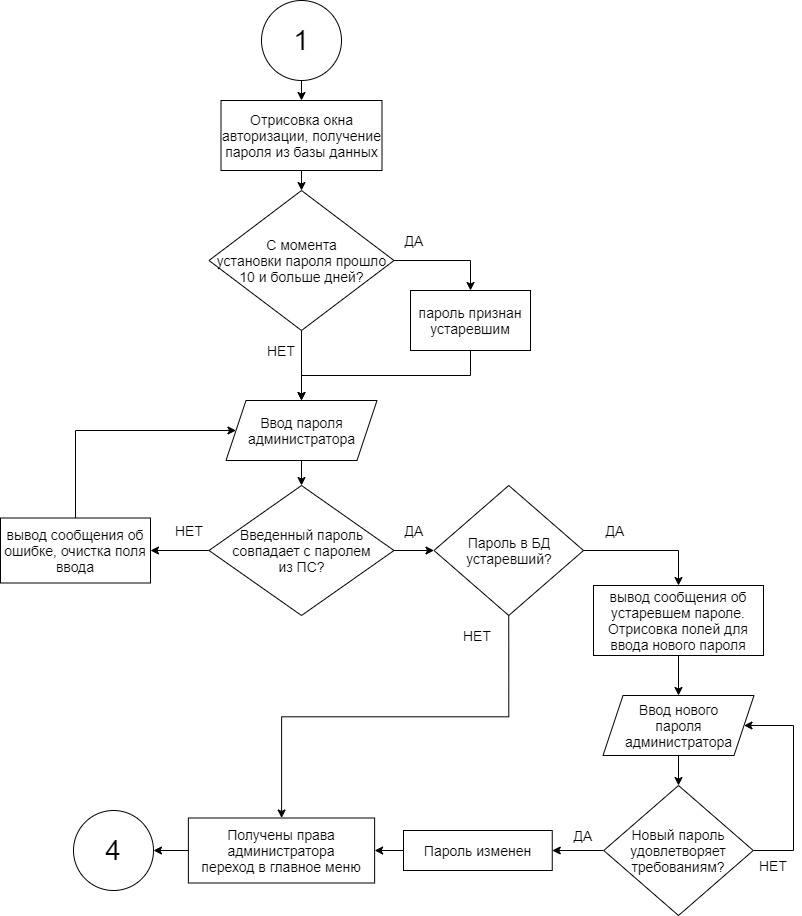


Рис.26. Блок-схема программы. Продолжение

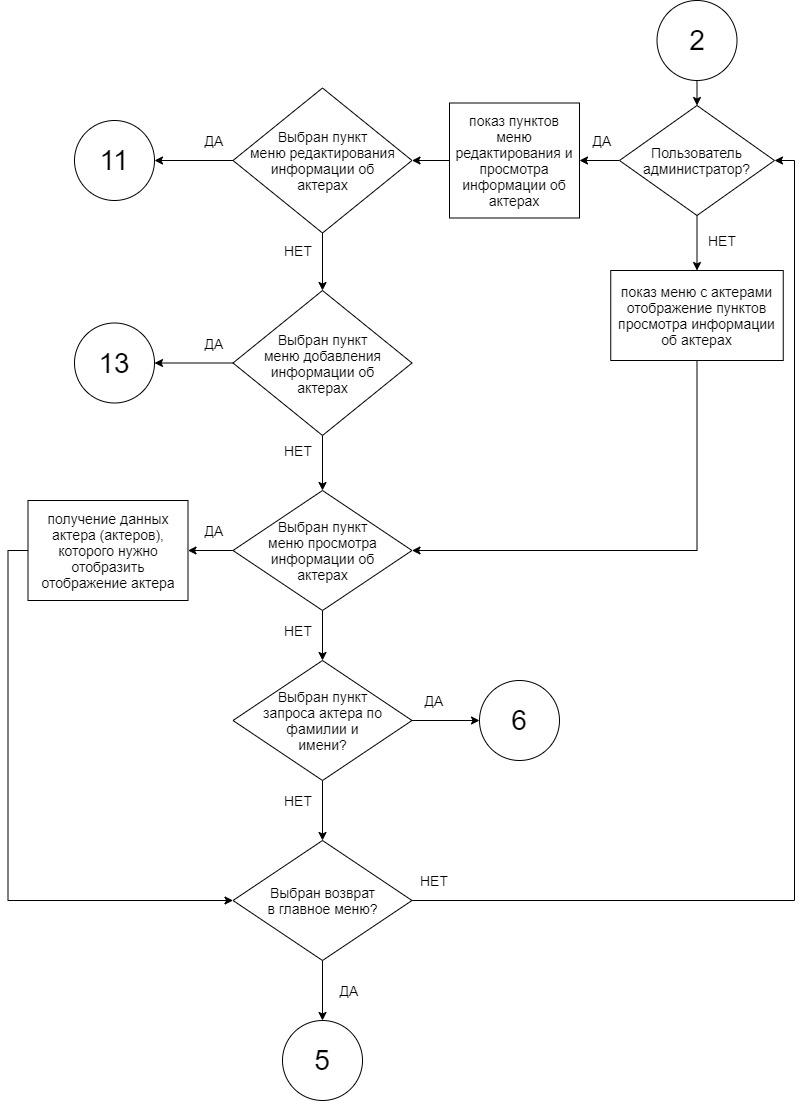


Рис.27. Блок-схема программы. Продолжение

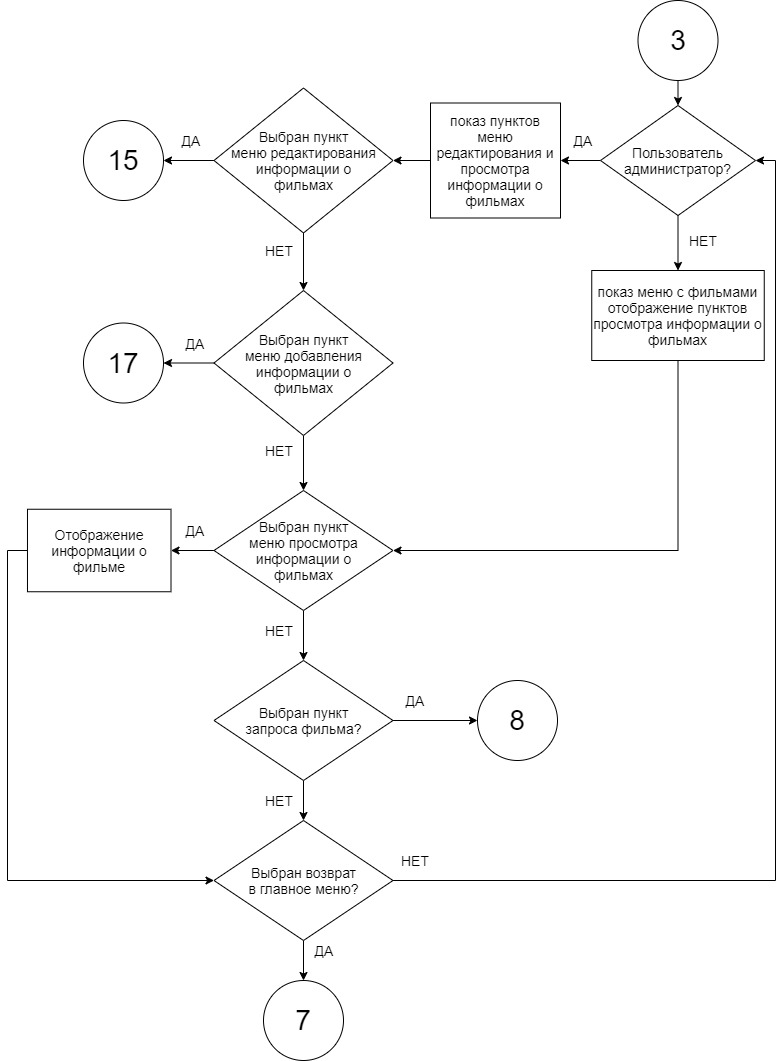


Рис.28. Блок-схема программы. Продолжение

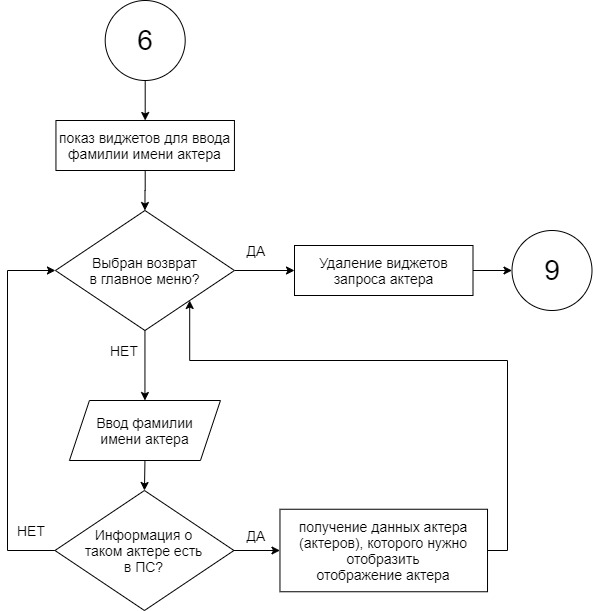


Рис.29. Блок-схема программы. Продолжение

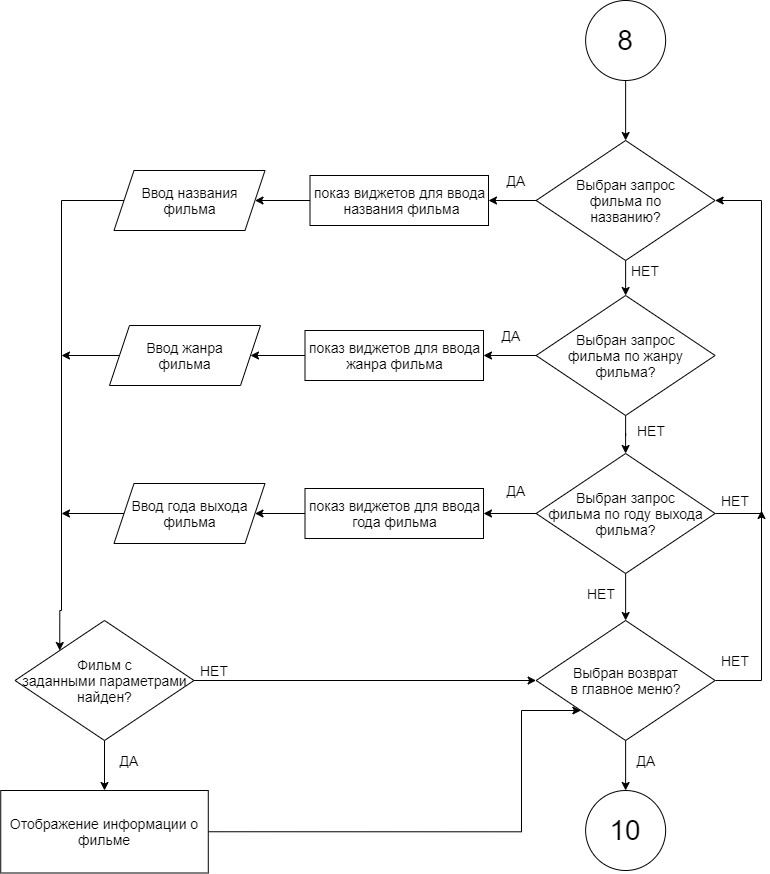


Рис.30. Блок-схема программы. Продолжение

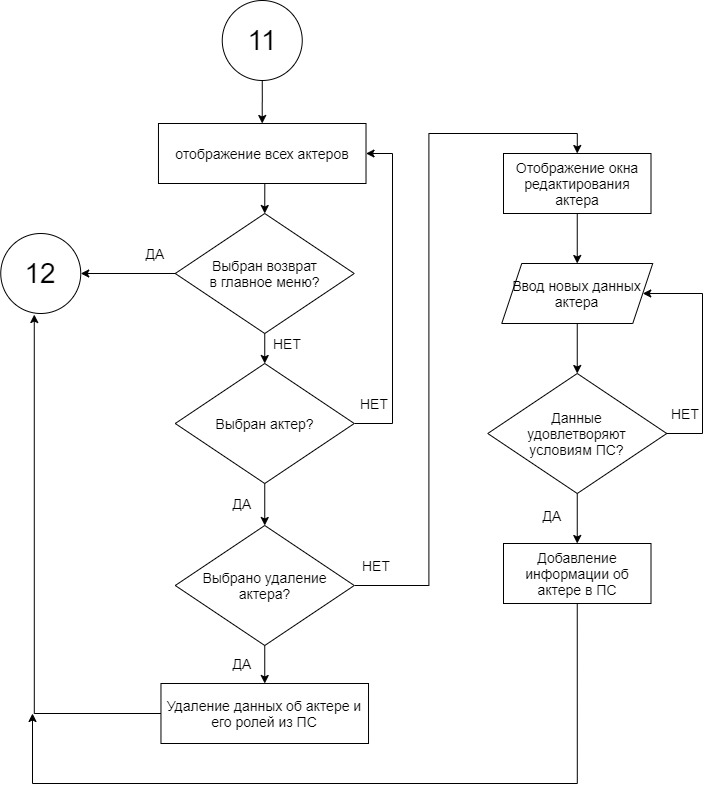


Рис.31. Блок-схема программы. Продолжение

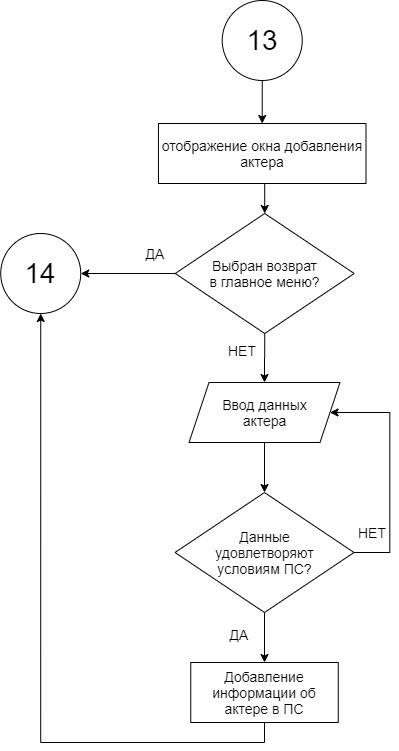


Рис.32. Блок-схема программы. Продолжение

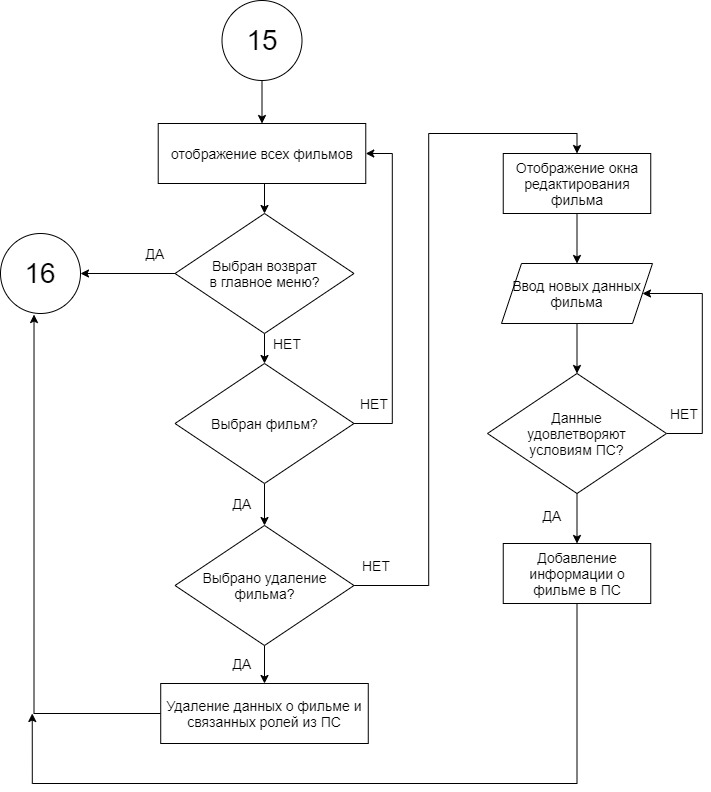


Рис.33. Блок-схема программы. Продолжение

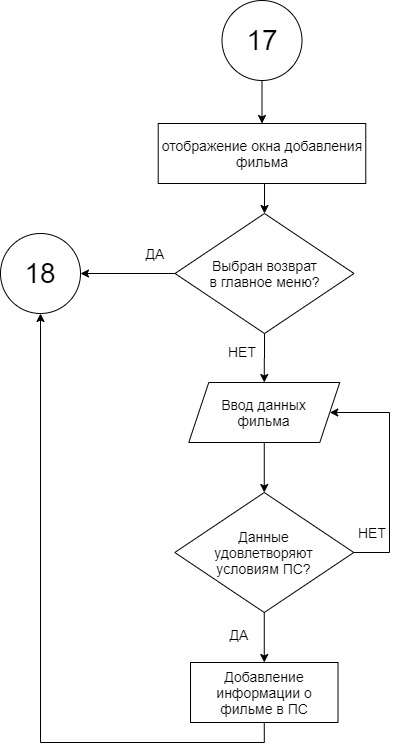


Рис.34. Блок-схема программы. Продолжение

# Листинг программы

import pyodbc

import os

from tkinter import \*

import datetime

from tkcalendar import DateEntry, Calendar

from tkinter import messagebox

import tkinter as tk

from tkinter import ttk

def stopper():

print("f")

'''класс Scrollable реализует фрейм с бесконечной прокруткой, на котором

можно разместить неограниченное количество элементов

'''

class Scrollable(tk.Frame):

"""

Make a frame scrollable with scrollbar on the right.

After adding or removing widgets to the scrollable frame,

call the update() method to refresh the scrollable area.

"""

def \_\_init\_\_(self, frame, width=16, windowHeight = 550):

self.scrollbar = tk.Scrollbar(frame, width=width)

self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y, expand=False)

self.canvas = tk.Canvas(frame, yscrollcommand=self.scrollbar.set, height = windowHeight)

self.canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

self.scrollbar.config(command=self.canvas.yview)

self.canvas.bind('<Configure>', self.\_\_fill\_canvas)

# base class initialization

tk.Frame.\_\_init\_\_(self, frame)

# assign this obj (the inner frame) to the windows item of the canvas

self.windows\_item = self.canvas.create\_window(0,0, window=self, anchor=tk.NW)

def \_\_fill\_canvas(self, event):

"Enlarge the windows item to the canvas width"

canvas\_width = event.width

self.canvas.itemconfig(self.windows\_item, width = canvas\_width)

def update(self):

"Update the canvas and the scrollregion"

self.update\_idletasks()

self.canvas.config(scrollregion=self.canvas.bbox(self.windows\_item))

def destroyAll(self):

"Delete the canvas and the scrollregion"

self.scrollbar.destroy()

self.canvas.destroy()

'''функция showMainMenu отображает главное меню программы на экране

'''

def showMainMenu():

deleteAllFromThisWindow()

programmDescription.grid(row = 0, column = 0, columnspan = 3, sticky=N+S+W+E, padx = 5, pady = 5)

creatorInfoButton.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

filmInformationButton.grid(row = 2, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

actorInformationButton.grid(row = 3, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

administratorLoginButton.grid(row = 4, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

exitButton.grid(row = 5, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

'''функция closeProgram завершает работу программы

'''

def closeProgram():

print("работа программы завершена")

exit()

'''функция showCreatorInfo отображает на экране информацию о разработчике ПО

'''

def showCreatorInfo():

deleteAllFromThisWindow()

creatorInfoLabel.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

'''функция showFilmsMenu отображает на экране меню действий для фильмов

'''

def showFilmsMenu():

global fromAllFilms

deleteAllFromThisWindow()

fromAllFilms = False

filmsMenuDescription.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

showFilmsButton.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmsNameButton.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmsStyleButton.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmsYearButton.grid(row = 4, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

if accessLevel == "Admin":

addFilmButton.grid(row = 5, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmButton.grid(row = 6, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 7, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

else:

backToMainMenu.grid(row = 5, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

'''функция showActorsMenu отображает на экране меню действий для актеров

'''

def showActorsMenu():

global fromAllActors

#top.destroy()

deleteAllFromThisWindow()

fromAllActors = False

actorsMenuDescription.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

showActorsButton.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

findActorButton.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

if accessLevel == "Admin":

addActorButton.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editActorButton.grid(row = 4, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 5, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

else:

backToMainMenu.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

'''функция showLoginMenu отображает на экране меню ввода пароля администратора

'''

def showLoginMenu():

deleteAllFromThisWindow()

if accessLevel == "Admin":

youAreAdminLabel.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

toBeUser.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

else:

loginDescriptionLabel.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

getPassword = conn.cursor()

getPassword.execute('SELECT \* FROM Access')

passwordData = []

for row in getPassword:

passwordData.append(row)

if len(passwordData) != 1:

messagebox.showinfo( "Критическая ошибка" , "Обнаружены критические изменения базы данных, переход в режим администратора невозможен, обратитесь к разработчику ПС.")

else:

passwordData = passwordData[0]

if passwordData[0] == None or passwordData[1] == None or passwordData[2] == None:

messagebox.showerror( "Критическая ошибка" , "Обнаружены критические изменения базы данных, переход в режим администратора невозможен, обратитесь к разработчику ПС.")

else:

print("Получена информация о пароле: ", passwordData)

today = datetime.date.today()

print("сегодня: ", today.strftime("%d.%m.%Y"))

if today.strftime("%d.%m.%Y") > passwordData[2].strftime("%d.%m.%Y"):

messagebox.showerror( "Предупреждение" , "Пароль устарел {}. Чтобы пользоваться правами администратора, вам необходимо ввести текущий пароль, а затем установить новый пароль администратора".format(passwordData[2].strftime("%d.%m.%Y")))

passwordEntry.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

checkPasswordButton.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

'''функция checkPassword читает данные о пароле из базы данных и сравнивает ввод пользователя с имеющимся паролем

'''

def checkPassword():

global accessLevel

getPassword = conn.cursor()

getPassword.execute('SELECT \* FROM Access')

passwordData = []

for row in getPassword:

passwordData.append(row)

passwordData = passwordData[0]

systemPassword = passwordData[0]

print("Получена информация о пароле: ", passwordData)

today = datetime.date.today()

print("сегодня: ", today.strftime("%d.%m.%Y"))

hasAccess = False

userPassword = passwordEntry.get()

passwordEntry.delete(0, END)

print("введен пароль: ", userPassword)

if systemPassword == userPassword:

hasAccess = True

if hasAccess == True:

passwordEnded = False

if today.strftime("%d.%m.%Y") > passwordData[2].strftime("%d.%m.%Y"):

passwordEnded = True

loginDescriptionLabel.grid\_forget()

editPasswordLabel.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editPasswordEntry.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editPasswordButton.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

messagebox.showinfo("Предупреждение", "Пароль устарел {}. Чтобы пользоваться правами администратора, установите новый пароль администратора.".format(passwordData[2].strftime("%d.%m.%Y")))

if passwordEnded == False:

accessLevel = "Admin"

loginDescriptionLabel.grid\_forget()

passwordEntry.grid\_forget()

checkPasswordButton.grid\_forget()

backToMainMenu.grid\_forget()

youAreAdminLabel.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

toBeUser.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

messagebox.showinfo( "Пароль принят!" , "Пароль администратора принят! Вам доступны все функции администратора!")

else:

messagebox.showerror("Ошибка ввода", "Пароль неверен, введите пароль снова!")

def checkNewPassword():

global accessLevel

hasError = False

newPassword = editPasswordEntry.get()

enUpperLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

enLowerLetters = "abcdefghijklmnopqrstuvwxyz"

if len(newPassword) >=4 and len(newPassword)<=16:

for i in newPassword:

if (i not in ["\_"]) and (i.isdigit() == False) and (i not in enUpperLetters) and (i not in enLowerLetters):

hasError = True

else:

hasError = True

if hasError == True:

messagebox.showerror("Ошибка ввода", "Пароль задан неверно, в пароле вы можете использовать строчные и заглавные английские буквы, цифры и знак подчеркивания, пароль не можетбыть меньше 4 и более 16 символов!")

#hasError = True

if hasError == False:

accessLevel = "Admin"

#обновление записи в таблице Access

changePasswordInDB = conn.cursor()

changePasswordInDB.execute("{CALL changePassword(?, ?, ?)}", newPassword, datetime.date.today(), datetime.date.today() + datetime.timedelta(days=10))

conn.commit()

editPasswordLabel.grid\_forget()

editPasswordEntry.grid\_forget()

editPasswordButton.grid\_forget()

youAreAdminLabel.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

toBeUser.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

messagebox.showinfo( "Пароль успешно изменен!" , "Пароль администратора успешно изменен! Вам доступны все функции администратора!")

showMainMenu()

def returnToUserAccess():

global accessLevel

accessLevel = "User"

youAreAdminLabel.grid\_forget()

toBeUser.grid\_forget()

showLoginMenu()

'''функция showAllFilms отображает на экране информацию о всех фильмах в ИС

'''

def showAllFilms():

global filmInfoDataList, actorsInFilmDataFramesList, actorDataLabelsList, dataList, fromFilm, fromAllFilms

global iFindFilmOnName, iFoundThisFilmData, iFindFilmOnStyle, iFoundThisFilmDataStyle

global iFindFilmOnYearBefore, iFindFilmOnYearAfter, iFoundThisFilmDataYear

filmsMenuDescription.grid\_forget()

showFilmsButton.grid\_forget()

addFilmButton.grid\_forget()

editFilmButton.grid\_forget()

backToMainMenu.grid\_forget()

#если ранее был какой-то фильм

filmInformationDescription.grid\_forget()

filmInfoName.grid\_forget()

filmInfoYear.grid\_forget()

filmInfoDuration.grid\_forget()

filmInfoStyle.grid\_forget()

filmInfoAge.grid\_forget()

filmInfoProdactionCost.grid\_forget()

filmInfoUSAProfit.grid\_forget()

filmInfoWorldProfit.grid\_forget()

filmInfoAwards.grid\_forget()

backToFilmsFromFilm.grid\_forget()

oneFilmheader.grid\_forget()

oneFilmbody.grid\_forget()

oneFilmfooter.grid\_forget()

if len(filmInfoDataList) != 0:

for i in filmInfoDataList:

i.destroy()

for i in actorsInFilmDataFramesList:

i.grid\_forget()

actorsInFilmDataFramesList.clear()

for i in actorDataLabelsList:

i.grid\_forget()

actorDataLabelsList.clear()

actorsDataList.clear()

fromFilm = False

fromAllFilms = True

#по названию

#iFindFilmOnName, iFoundThisFilmData

#по жанру

#iFindFilmOnStyle, iFoundThisFilmDataStyle

#по году

#iFindFilmOnYearBefore, iFindFilmOnYearAfter, iFoundThisFilmDataYear

if iFindFilmOnName == True:

dataList = iFoundThisFilmData

iFindFilmOnName = False

elif iFindFilmOnStyle == True:

dataList = iFoundThisFilmDataStyle

iFindFilmOnStyle = False

elif iFindFilmOnYearBefore == True or iFindFilmOnYearAfter == True:

dataList = iFoundThisFilmDataYear

iFindFilmOnYearBefore = False

iFindFilmOnYearAfter = False

else:

cursor = conn.cursor()

cursor.execute('select \* from Films')

dataList = []

for row in cursor:

dataList.append(row)

deleteAllFromThisWindow()

allFilmsHeader.grid(row = 0, sticky=N+S+W+E)

allFilmsBody.grid(row = 1, sticky=N+S+W+E)

allFilmsFooter.grid(row = 2, sticky=N+S+W+E)

#allFilmsHeader

allFilmsInformationLabel.grid(row = 0, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

filmNameLabel.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmYearLabel.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

filmLengthLabel.grid(row = 1, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

filmStyleLabel.grid(row = 1, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

filmsLinksButtonsList = []

param = "дебаг"

#allFilmsBody

for i in range(len(dataList)):

filmsLinksButtonsList.append(Button(scrollableBodyAllFilms, text="%s"%(dataList[i][1]), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 300, width = 25))

filmsLinksButtonsList[i].grid(row = i, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmsLinksButtonsList[i].bind('<Button-1>', lambda event: showFilmData(event, param))

Label(scrollableBodyAllFilms, text="%s"%(dataList[i][2]), bg="#FFE4C4", font = ("Consolas", 18), width = 13).grid(row = i, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

Label(scrollableBodyAllFilms, text="%s"%(dataList[i][3]), bg="#FFE4C4", font = ("Consolas", 18), width = 13).grid(row = i, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

Label(scrollableBodyAllFilms, text="%s"%(dataList[i][4]), bg="#FFE4C4", font = ("Consolas", 18), width = 22).grid(row = i, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

#allFilmsFooter

footerAllInformationMenu.grid(row = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenuFromFilms.grid(row = 1, sticky=N+S+W+E, padx = 5, pady = 5)

scrollableBodyAllFilms.update()

'''функция showFilmData отображает на экране информацию о выбранном фильме

'''

def showFilmData(event, param):

global filmInfoDataList, actorsInFilmDataFramesList, fromThisFilmIDToActor, startRowForActors, actorsDataList, actorDataLabelsList, actorsInOneRowCount, dataList, fromAllActors, actorRolesLabelsList#, returnToFilmID

global fromFilm, fromActor, filmID, fromAllFilms, actorRolesDataList, startRowForFilmsInActorInfo

#если раньше были открыты все фильмы

allFilmsInformationLabel.grid\_forget()

filmNameLabel.grid\_forget()

filmYearLabel.grid\_forget()

filmLengthLabel.grid\_forget()

filmStyleLabel.grid\_forget()

footerAllInformationMenu.grid\_forget()

backToMainMenuFromFilms.grid\_forget()

allFilmsHeader.grid\_forget()

allFilmsBody.grid\_forget()

allFilmsFooter.grid\_forget()

#Если раньше был открыт актер

oneActorheader.grid\_forget()

oneActorbody.grid\_forget()

oneActorfooter.grid\_forget()

actorInformationDescription.grid\_forget()

actorInfoName.grid\_forget()

actorInfoSurname.grid\_forget()

actorBirthDay.grid\_forget()

backToFilmFromActor.grid\_forget()

actorRolesFilmName.grid\_forget()

actorRolesFilmYear.grid\_forget()

actorRolesStyle.grid\_forget()

actorRolesCharacter.grid\_forget()

actorRolesAge.grid\_forget()

actorRolesLabelsList.clear()

for i in actorDataLabelsList:

i.grid\_forget()

i.grid\_remove()

i.destroy()

actorDataLabelsList.clear()

for i in actorRolesLabelsList:

i.grid\_forget()

i.grid\_remove()

i.destroy()

actorRolesLabelsList.clear()

print("===============================================\nparam = ", param)

oneFilmheader.grid(row = 0, sticky=N+S+W+E)

oneFilmbody.grid(row = 1, sticky=N+S+W+E)

oneFilmfooter.grid(row = 2, sticky=N+S+W+E)

filmInformationDescription.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoName.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoYear.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoDuration.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoStyle.grid(row = 4, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoAge.grid(row = 5, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoProdactionCost.grid(row = 6, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoUSAProfit.grid(row = 7, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoWorldProfit.grid(row = 8, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoAwards.grid(row = 9, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToFilmsFromFilm.grid(row = 10, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

print("info: ", event.widget.grid\_info())

filmData = []

if fromAllActors == True:

print("3333")

fromAllFilms = False

filmInfo = []

number = event.widget.grid\_info()["row"]

print("number = ", number)

filmInfo = actorRolesDataList[number - startRowForFilmsInActorInfo] #потому что надо получить строку, предшествующую этой

filmID = filmInfo[6]

print("filmInfo после всех актеров", filmInfo)

print("filmID = ", filmID)

getFilmOnID = """\

{CALL getFilmOnID(?)}

"""

queryGetFilmOnID = conn.cursor()

queryGetFilmOnID.execute(getFilmOnID, filmID)

filmInfo = []

for note in queryGetFilmOnID:

filmInfo.append(note)

filmData = filmInfo[0]

print("filmData после актера", filmData)

fromThisFilmIDToActor = filmID

elif fromActor == True:

print("\nВозврат от актера к фильму\n22222")

fromActor = False

getFilmOnID = """\

{CALL getFilmOnID(?)}

"""

print("\nactorRolesLabelsList\n", actorRolesLabelsList)

filmID = int(fromThisFilmIDToActor)

# number = event.widget.grid\_info()["row"]

# if number == 0:

# filmID = int(fromThisFilmIDToActor)

# else:

filmInfo = []

#print("number = ", number)

# filmInfo = actorRolesDataList[number - startRowForFilmsInActorInfo] #потому что надо получить строку, предшествующую этой

# filmID = filmInfo[6]

# print("filmInfo после всех актеров", filmInfo)

print("filmID = ", filmID)

queryGetFilmOnID = conn.cursor()

queryGetFilmOnID.execute(getFilmOnID, filmID)

for note in queryGetFilmOnID:

filmData.append(note)

filmData = filmData[0]

print("filmData после актера", filmData)

fromThisFilmIDToActor = filmID

elif fromAllFilms == True:

print("1111")

fromAllFilms = False

row = event.widget.grid\_info()["row"]

filmData = dataList[row]

filmID = filmData[0]

print("filmData после всех фильмов", filmData)

fromThisFilmIDToActor = filmID

print("actorRolesLabelsList = ", actorRolesLabelsList)

for i in actorRolesLabelsList:

for j in range(len(i)):

i[j].grid\_forget()

actorRolesLabelsList.clear()

filmInfoDataList = []

i = 0

for i in range(len(filmData)):

if i == 0:

continue

filmInfoDataList.append(Label(scrollableBodyOneFilm, text = filmData[i], bg="#FFE4C4", font = ("Consolas", 20), wraplength = 800))

filmInfoDataList[i-1].grid(row = i, column = 1, columnspan = actorsInOneRowCount - 1, sticky=N+S+W+E, padx = 5, pady = 5)

actorsInFilmDescriptionLabel.grid(row = i + 1, column = 0, columnspan = actorsInOneRowCount, sticky=N+S+W+E, padx = 5, pady = 5)

actorsDataList = []

queryGetActorsForFilm = """\

{CALL getActorsOnFilmID(?)}

"""

getActorsFromFilmOnFilmID = conn.cursor()

getActorsFromFilmOnFilmID.execute(queryGetActorsForFilm, filmID)

for row in getActorsFromFilmOnFilmID:

actorsDataList.append(row)

print("открыт фильм ", filmData)

print("актеры: ", actorsDataList)

print(fromThisFilmIDToActor)

startRowForActors = i + 2

lastActorRow = startRowForActors

lastActorColumn = 0

for j in range(len(actorsDataList)):

if j % actorsInOneRowCount == 0:

lastActorRow += 1

lastActorColumn = 0

textField = Label(master = scrollableBodyOneFilm, text = "%s %s\nроль: %s"%(actorsDataList[j][1], actorsDataList[j][2], actorsDataList[j][3]), relief='raised', font = ("Consolas", 20), width = filmInformationLabelWidth, wraplength = 380, bg="#FFE4C4")

textField.grid(row = lastActorRow, column = lastActorColumn, sticky=N+S+W+E, padx = 5, pady = 5)

textField.bind('<Button-1>', showActorData)

actorsInFilmDataFramesList.append(textField)

lastActorColumn += 1

fromAllActors = False

fromFilm = True

fromActor = False

scrollableBodyOneFilm.update()

actorRolesDataList.clear()

actorRolesDataList = []

print("\n\nactorRolesDataList = \n\n\n", actorRolesDataList)

def showAllActors():

global fromAllActors, actorRolesLabelsList, allActorsDataList, fromActor, actorsInFilmDataFramesList, iFindActor, iFoundThisActorData

filmsMenuDescription.grid\_forget()

showFilmsButton.grid\_forget()

addFilmButton.grid\_forget()

editFilmButton.grid\_forget()

backToMainMenu.grid\_forget()

for i in actorsInFilmDataFramesList:

i.grid\_forget()

actorsInFilmDataFramesList.clear()

#Если раньше был открыт актер

fromActor = False

oneActorheader.grid\_forget()

oneActorbody.grid\_forget()

oneActorfooter.grid\_forget()

actorInformationDescription.grid\_forget()

actorInfoName.grid\_forget()

actorInfoSurname.grid\_forget()

actorBirthDay.grid\_forget()

backToFilmFromActor.grid\_forget()

actorRolesDescriptionLabel.grid\_forget()

actorRolesFilmName.grid\_forget()

actorRolesFilmYear.grid\_forget()

actorRolesStyle.grid\_forget()

actorRolesCharacter.grid\_forget()

actorRolesAge.grid\_forget()

#если ранее было открыто меню актеров

actorsMenuDescription.grid\_forget()

showActorsButton.grid\_forget()

addActorButton.grid\_forget()

editActorButton.grid\_forget()

backToMainMenu.grid\_forget()

for i in actorDataLabelsList:

i.grid\_remove()

i.destroy()

actorDataLabelsList.clear()

backToMainMenu.grid\_forget()

findActorDescriptionLabel.grid\_forget()

findActorNameLabel.grid\_forget()

findActorSurnameLabel.grid\_forget()

findActorNameEntry.grid\_forget()

findActorSurnameEntry.grid\_forget()

startFindActorButton.grid\_forget()

#print("actorRolesLabelsList = ", actorRolesLabelsList)

for i in actorRolesLabelsList:

for j in range(len(i)):

i[j].grid\_forget()

actorRolesLabelsList.clear()

fromAllActors = True

allActorsDataList = []

if iFindActor == True:

allActorsDataList = iFoundThisActorData

iFindActor = False

else:

queryGetAllActors = conn.cursor()

queryGetAllActors.execute('select \* from Actors')

for row in queryGetAllActors:

allActorsDataList.append(row)

allActorsheader.grid(row = 0, sticky=N+S+W+E)

allActorsbody.grid(row = 1, sticky=N+S+W+E)

allActorsfooter.grid(row = 2, sticky=N+S+W+E)

#allFilmsHeader

allActorsInformationDescription.grid(row = 0, column = 0, columnspan = 4, sticky=N+S+W+E)

allActorsInfoName.grid(row = 1, column = 0, sticky=N+S+W+E)

allActorsInfoSurname.grid(row = 1, column = 1, sticky=N+S+W+E)

allActorsBirthDay.grid(row = 1, column = 2, sticky=N+S+W+E)

allActorsGender.grid(row = 1, column = 3, sticky=N+S+W+E)

filmsLinksButtonsList = []

#allFilmsBody

for i in range(len(allActorsDataList)):

filmsLinksButtonsList.append(Button(scrollableBodyallActors, text="%s"%(allActorsDataList[i][1]), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 250, width = 25))

filmsLinksButtonsList[i].grid(row = i, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmsLinksButtonsList[i].bind('<Button-1>', showActorData)

Label(scrollableBodyallActors, text="%s"%(allActorsDataList[i][2]), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 250, width = 20).grid(row = i, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

Label(scrollableBodyallActors, text="%s"%(allActorsDataList[i][3].strftime("%d.%m.%Y")), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 250, width = 20).grid(row = i, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

Label(scrollableBodyallActors, text="%s"%(allActorsDataList[i][4]), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 250, width = 10).grid(row = i, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

#allFilmsFooter

backToActorsMenuFromActors.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E)

scrollableBodyallActors.update()

'''функция showActorData отображает на экране информацию о выбранном актере

'''

def showActorData(event):

global fromThisFilmIDToActor, startRowForActors, actorsDataList, actorDataLabelsList, actorsInFilmDataFramesList, actorsInOneRowCount, allActorsDataList, fromAllActors, fromFilm, actorRolesLabelsList#, returnToFilmID

global fromActor, actorRolesDataList, startRowForFilmsInActorInfo, scrollableBodyOneActor, filmInfoDataList

print("===================================\nвы в меню информации об актере\n=====================================")

#если ранее был какой-то фильм

filmInformationDescription.grid\_forget()

filmInfoName.grid\_forget()

filmInfoYear.grid\_forget()

filmInfoDuration.grid\_forget()

filmInfoStyle.grid\_forget()

filmInfoAge.grid\_forget()

filmInfoProdactionCost.grid\_forget()

filmInfoUSAProfit.grid\_forget()

filmInfoWorldProfit.grid\_forget()

filmInfoAwards.grid\_forget()

backToFilmsFromFilm.grid\_forget()

actorsInFilmDescriptionLabel.grid\_forget()

oneFilmheader.grid\_forget()

oneFilmbody.grid\_forget()

oneFilmfooter.grid\_forget()

#если ранее были открыты все актеры

allActorsheader.grid\_forget()

allActorsbody.grid\_forget()

allActorsfooter.grid\_forget()

if len(filmInfoDataList) != 0:

for lab in filmInfoDataList:

lab.destroy()

print("actorRolesLabelsList ПРИ ПОКАЗЕ АКТЕРА= ", actorRolesLabelsList)

for i in actorRolesLabelsList:

for j in range(len(i)):

i[j].destroy()

actorRolesLabelsList.clear()

scrollableBodyOneActor.destroyAll()

oneActorheader.grid(row = 0, sticky=N+S+W+E)

oneActorbody.grid(row = 1, sticky=N+S+W+E)

oneActorfooter.grid(row = 2, sticky=N+S+W+E)

scrollableBodyOneActor = Scrollable(oneActorbody, width=30)

actorInformationDescription = Label(oneActorheader, text='Вы открыли окно информации об актере', bg ="lightgreen", font = ("Consolas", 20), width = 84)

actorInfoName = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Имя актера",bg ="#FFE4C4", width = 20)

actorInfoSurname = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Фамилия актера",bg ="#FFE4C4", width = 20)

actorBirthDay = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Дата рождения",bg ="#FFE4C4", width = 20)

actorRolesDescriptionLabel = Label(scrollableBodyOneActor, width = 82, font = ("Consolas", 20), text="Актер принимал участие в фильмах:", bg ="lightgreen")

actorInformationDescription.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

actorInfoName.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

actorInfoSurname.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

actorBirthDay.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

row = event.widget.grid\_info()["row"]

column = event.widget.grid\_info()["column"]

actorData = []

if fromFilm == True:

fromFilm = False

fromActor = True

actorID = actorsDataList[(row - startRowForActors - 1) \* actorsInOneRowCount + column][4]

backToFilmFromActor.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

fromThisFilmIDToActor = actorsDataList[(row - startRowForActors - 1) \* actorsInOneRowCount + column][0]

if fromAllActors == True:

actorID = allActorsDataList[row][0]

fromThisFilmIDToActor = -5

getActorOnActorID = """\

{CALL getActorOnActorID(?)}

"""

queryGetActorOnActorID = conn.cursor()

queryGetActorOnActorID.execute(getActorOnActorID, actorID)

for note in queryGetActorOnActorID:

actorData.append(note)

print("actorData = ", actorData)

print("actorID = ", actorID)

backToActorsFromActor.grid(row = 0, column = 2, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

actorDataLabelsList = []

for j in range(1, 4):

if j == 3:

actorDataTextLabel = Label(scrollableBodyOneActor, text = actorData[0][j].strftime("%d.%m.%Y"), font = ("Consolas", 20), wraplength = 380, bg="#FFE4C4")#.strftime("%d.%m.%Y")

actorDataTextLabel.grid(row = j - 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

actorDataLabelsList.append(actorDataTextLabel)

else:

actorDataTextLabel = Label(scrollableBodyOneActor, text = actorData[0][j], font = ("Consolas", 20), wraplength = 380, bg="#FFE4C4")

actorDataTextLabel.grid(row = j - 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

actorDataLabelsList.append(actorDataTextLabel)

#получаем все фильмы где снимался актер

actorRolesDataList.clear()

actorRolesDataList = []

print("\n\nactorRolesDataList = \n\n\n", actorRolesDataList)

getAllFilmsForActorOnActorID = """\

{CALL getAllFilmForActor(?)}

"""

queryGetAllFilmsForActorOnActorID = conn.cursor()

queryGetAllFilmsForActorOnActorID.execute(getAllFilmsForActorOnActorID, actorID)

for note in queryGetAllFilmsForActorOnActorID:

actorRolesDataList.append(note)

#actorRolesLabelsList = []

for i in actorRolesLabelsList:

i.destroy()

actorRolesLabelsList.clear()

#param = "дебаг"

print("actorRolesDataList", actorRolesDataList)

if len(actorRolesDataList) != 0:

actorRolesDescriptionLabel.grid(row = 3, column = 0, columnspan = 5, sticky=N+S+W+E, padx = 5, pady = 5)

actorRolesFilmName.grid(row = 4, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

actorRolesFilmYear.grid(row = 4, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

actorRolesStyle.grid(row = 4, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

actorRolesCharacter.grid(row = 4, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

actorRolesAge.grid(row = 4, column = 4, sticky=N+S+W+E, padx = 5, pady = 5)

startRowForFilmsInActorInfo = 5

for i in range(len(actorRolesDataList)):

actorRolesLabelsList.append([])

for j in range(len(actorRolesDataList[i]) - 2):

if j == 0:

actorRolesLabelsList[i].append(Label(scrollableBodyOneActor, text = actorRolesDataList[i][j], font = ("Consolas", 20), wraplength = 250, bg="#FFE4C4"))

actorRolesLabelsList[i][j].grid(row = i + 5, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

#actorRolesLabelsList[i][j].bind('<Button-1>', lambda event, param = event.widget.grid\_info()["row"]: showFilmData(event, param))

#print("i = %d, j = %d, info: "%(i, j), actorRolesLabelsList[i][j].grid\_info())

else:

actorRolesLabelsList[i].append(Label(scrollableBodyOneActor, text = actorRolesDataList[i][j], font = ("Consolas", 20), wraplength = 250, bg="#FFE4C4"))

actorRolesLabelsList[i][j].grid(row = i + 5, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

print("actorRolesLabelsList in ShowActorData", actorRolesLabelsList)

else:

messagebox.showinfo("Внимание!", "По выбранному вами актеру нет информации об участии в фильмах!")

'''функция addActorInDataBase обеспечивает окно для добавления информации об актере в базу данных

'''

def addActorInDataBase():

global labelsListWithFilmsForAddActorInDB, checkBoxIntVarListForAddActorInDB, selectedFilmsIndexesList, allFilmsDataList, selectedFilmsRowIndexesList, actorRolesEntryList

print("Вы решили добавить актера в базу данных")

deleteAllFromThisWindow()

for i in actorRolesEntryList:

i.delete(0, END)

i["state"] = DISABLED

addActorActorNameEntry.delete(0, END)

addActorActorSurnameEntry.delete(0, END)

addActorActorBirthDayEntry["text"] = "Выберите дату рождения"

actorAddNoteHeader.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

actorAddNoteBody.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

actorAddNoteFooter.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

getAllFilmsFromDataBase = """\

SELECT \* FROM getAllFilmsFromDBForAddActor

"""

queryGetAllFilmsFromDataBase = conn.cursor()

queryGetAllFilmsFromDataBase.execute(getAllFilmsFromDataBase)

allFilmsDataList = []

for note in queryGetAllFilmsFromDataBase:

allFilmsDataList.append(note)

filmsCount = len(allFilmsDataList)

print(allFilmsDataList)

addActorDescriptionLabel.grid(row = 0, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

addActorActorNameLabel.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

addActorActorSurnameLabel.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

addActorActorBirthDayLabel.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

addActorActorGenderLabel.grid(row = 4, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

addActorActorNameEntry.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

addActorActorSurnameEntry.grid(row = 2, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

addActorActorBirthDayEntry.grid(row = 3, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

addActorActorGenderEntry.grid(row = 4, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

addActorChooseRolesDescription.grid(row = 5, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

selectedFilmsIndexesList = []

selectedFilmsRowIndexesList = []

#здесь

addActorChooseRoleCheckboxLabel.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

addActorChooseRoleFilmNameLabel.grid(row = 0, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

addActorChooseRoleFilmYearLabel.grid(row = 0, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addActorChooseRoleFilmStyleLabel.grid(row = 0, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

addActorChooseRoleFilmAgeLimitLabel.grid(row = 0, column = 4, sticky=N+S+W+E, padx = 5, pady = 5)

addActorChooseRoleActorRoleLabel.grid(row = 0, column = 5, sticky=N+S+W+E, padx = 5, pady = 5)

for i in range(filmsCount):

labelsListWithFilmsForAddActorInDB.append([])

for j in range(len(allFilmsDataList[i]) + 1):

if j == 0:

checkBoxIntVarListForAddActorInDB.append(IntVar())

labelsListWithFilmsForAddActorInDB[i].append(Checkbutton(scrollableBodyChooseFilmsForActor, text='', variable=checkBoxIntVarListForAddActorInDB[i], onvalue=1, offvalue=0, bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].bind('<Button-1>', chooseFilm)#lambda event, param = event.widget.grid\_info()["row"]:

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i+1, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

elif j == 5:

actorRolesEntryList.append(Entry(scrollableBodyChooseFilmsForActor, font = ("Consolas", 14), state = DISABLED, width = 45))

actorRolesEntryList[i].grid(row = i+1, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

else:

labelsListWithFilmsForAddActorInDB[i].append(Label(scrollableBodyChooseFilmsForActor, text = "%s"%(allFilmsDataList[i][j]), wraplength = 300, font = ("Consolas", 20), bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i+1, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

scrollableBodyChooseFilmsForActor.update()

addActorFinishDescription.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E)

returnBackToActorsMenu.grid(row = 1, column = 0)

addActorNoteInDataBase.grid(row = 1, column = 1)

'''Фукнция выбора значения в чекбоксе для роли актера в фильме при занесении в БД информации об актере

'''

def chooseFilm(event):

global selectedFilmsIndexesList, allFilmsDataList, selectedFilmsRowIndexesList, actorRolesEntryList

print(event.widget.grid\_info())

rowIndex = event.widget.grid\_info()["row"] - 1

if checkBoxIntVarListForAddActorInDB[rowIndex].get() == 0:

actorRolesEntryList[rowIndex]["state"] = NORMAL

selectedFilmsIndexesList.append(allFilmsDataList[rowIndex][0])

selectedFilmsRowIndexesList.append(rowIndex)

print("выбран индекс: %d"%rowIndex)

print("индекс выбранных фильмов:", selectedFilmsIndexesList)

print("индекс выбранных строк:", selectedFilmsRowIndexesList)

else:

selectedFilmsIndexesList.remove(allFilmsDataList[rowIndex][0])

selectedFilmsRowIndexesList.remove(rowIndex)

actorRolesEntryList[rowIndex].delete(0, END)

actorRolesEntryList[rowIndex]["state"] = DISABLED

print("индекс выбранных фильмов:", selectedFilmsIndexesList)

print("индекс выбранных строк:", selectedFilmsRowIndexesList)

'''Фукнция выбора даты рождения при занесении в БД информации об актере

'''

def chooseDate():

global actorBirthDayDateTime

def print\_sel():

global actorBirthDayDateTime

print(calendarForChooseDate.selection\_get())

#calendarForChooseDate.see(datetime.date(year=2016, month=2, day=5))

actorBirthDayDateTime = calendarForChooseDate.selection\_get()

addActorActorBirthDayEntry["text"] = actorBirthDayDateTime

returnBackToActorsMenu["state"] = NORMAL

addActorNoteInDataBase["state"] = NORMAL

top.destroy()

returnBackToActorsMenu["state"] = DISABLED

addActorNoteInDataBase["state"] = DISABLED

top = Toplevel(root)

top.geometry("+300+300")

top.overrideredirect(True) #запрет на взаимодействие с панелью инструментов окна

today = datetime.date.today()

mindate = datetime.date(year = 1800, month=1, day=21)

maxdate = today# + datetime.timedelta(days=5)

calendarInfoLabel = Label(top, text = "Выберите дату рождения актера:\nгод, месяц и число.\nПосле этого нажмите\nна кнопку 'Выбрать эту дату'", font = ("Consolas", 16), bg= "lightgreen")

calendarInfoLabel.pack(fill="both", expand=True)

calendarForChooseDate = Calendar(top, font="Arial 14", selectmode='day', locale='ru\_RU', mindate=mindate, maxdate=maxdate, disabledforeground='red', cursor="hand1", year = 1975, month = 1, day = 1, date\_pattern = "y-mm-dd")

calendarForChooseDate.pack(fill="both", expand=True)

Button(top, text="Выбрать эту дату", font = ("Consolas", 16), command=print\_sel).pack()

"""Функция addActorToDB проверяет правильность введенных в форму актера данных и заносит их в БД

"""

def addActorToDB():

global actorBirthDayDateTime, selectedFilmsIndexesList, selectedFilmsRowIndexesList, actorRolesEntryList

hasError = False

newNoteForActors = []

#получение информации о всех имеющихся актерах

getAllActorsFromDataBase = """\

SELECT \* FROM getAllActors

"""

queryGetAllActorsIDFromDataBase = conn.cursor()

queryGetAllActorsIDFromDataBase.execute(getAllActorsFromDataBase)

allActorsList = []

for note in queryGetAllActorsIDFromDataBase:

allActorsList.append(note)

actorsCount = len(allActorsList)

print("Всего актеров в БД: ", allActorsList)

#получение индекса актера

maxActorIndex = -5

if actorsCount != 0:

for i in range(actorsCount):

if allActorsList[i][0] > maxActorIndex:

maxActorIndex = allActorsList[i][0]

else:

maxActorIndex = -1

print("Максимальный индекс в списке: %d"%maxActorIndex)

addActorInDBActorIndex = maxActorIndex + 1

newNoteForActors.append(addActorInDBActorIndex)

# ruUpperLetters = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

# ruLowerLetters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя -"

ruUpperLetters = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

ruLowerLetters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

enUpperLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

enLowerLetters = "abcdefghijklmnopqrstuvwxyz"

#проверка имени актера

#hasActorName = False

actorName = addActorActorNameEntry.get()

lenActorName = len(actorName)

if not actorName:

messagebox.showerror("Ошибка", "Вы не ввели имя актера! Повторите ввод!")

addActorActorNameEntry.delete(0, END)

addActorActorNameEntry.focus()

hasError = True

elif actorName[0] == " ":

messagebox.showerror("Ошибка", "Вы ввели пробел перед именем! Повторите ввод!")

addActorActorNameEntry.delete(0, END)

addActorActorNameEntry.focus()

hasError = True

elif not actorName[0] == actorName[0].upper():

messagebox.showerror("Ошибка", "Вы ввели имя актера с маленькой буквы! Повторите ввод!")

addActorActorNameEntry.focus()

hasError = True

else:

nizya = r'\~\`\!\@"\#\№\$\;\%\^\:\&\?\(\)\\_\+\=\<\>\,\.\[\]\{\}\\\/\|\\*\''

find\_nizya = re.compile('([{}])'.format(nizya))

res = find\_nizya.findall(actorName)

myString = ', '.join(res)

print(res, myString)

if res:

messagebox.showerror('ОШИБКА', 'Вы ввели имя актера с запрещёнными символами, такими как: {}.'.format(

myString) + ' ' + 'Пожалуйста, не вводите символы: ~, `, @, ", #, №, $, ;, %, ^, :, &, ?, \*, (, ), \_, +, =, [, ], {, }, |, \, /, ., , , <, >')

#CityTeamEntry.delete(0, END)

hasError = True

addActorActorNameEntry.focus()

else:

nizya1 = 'a-zA-Z'

find\_nizya1 = re.compile('([{}])'.format(nizya1))

res1 = find\_nizya1.findall(actorName)

myString1 = ', '.join(res1)

print(res1, myString1)

if res1:

hasError = True

messagebox.showerror('ОШИБКА',

'Вы ввели имя актера с помощью латинских букв, таких как: {}.'.format(

myString1) + ' ' + 'Пожалуйста, не используйте латинские буквы.')

#CityTeamEntry.delete(0, END)

addActorActorNameEntry.focus()

else:

if actorName[lenActorName - 1] == " ":

hasError = True

messagebox.showerror('ОШИБКА', 'Вы ввели символы пробела после имени актера')

addActorActorNameEntry.focus()

else:

i = 0

while actorName[i] == ' ':

actorName = actorName[1:]

while actorName[

len(actorName) - 1] == ' ':

actorName = actorName[:-1]

i = 1

while i < len(actorName) - 1:

if actorName[i] == ' ' and actorName[i + 1] == ' ':

actorName = actorName[:i + 1] + actorName[i + 2:]

else:

i += 1

hasActorName = True

addActorActorNameEntry.delete(0, END)

addActorActorNameEntry.insert(0, actorName)

newNoteForActors.append(actorName)

#проверка фамилии актера

actorSurname = addActorActorSurnameEntry.get()

defisCount = 0

spaceCount = 0

if hasError == False:

if len(actorSurname) == 0:

hasError = True

messagebox.showerror("Ошибка", "Вы не ввели фамилию актера! Вы можете использовать русские буквы, пробел и дефисы. Повторите ввод!")

else:

if (actorSurname[0] in ['-', ' '] or actorSurname[-1] in ['-', ':', ' ']):

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера не должна начинаться или оканчиваться на пробел или '-'! Введите имя актера рускими буквами!\nВы можете использовать русские буквы, пробелы и дефисы (не более 1 символа подряд).")

else:

if actorSurname[0] in ruLowerLetters:

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера написана со строчной буквы! Введите имя актера с заглавной буквы!")

else:

for i in actorSurname:

if i not in ruUpperLetters and i not in ruLowerLetters and i not in ["-", " "]:

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера введена неверно!\nВы можете использовать русские буквы, пробелы и дефисы (не более 1 символа подряд).")

break

else:

if defisCount == 0 and i == "-":

defisCount +=1

elif defisCount == 1 and i != "-":

defisCount = 0

elif defisCount == 1 and i == "-":

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера введена неверно! Вы ввели слишком много дефисов подряд! Вы можете использовать русские буквы, пробелы, дефисы (не более 1 символа подряд).")

break

if spaceCount == 0 and i == " ":

spaceCount +=1

elif spaceCount == 1 and i != " ":

spaceCount = 0

elif spaceCount == 1 and i == " ":

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера введена неверно! Вы ввели слишком много пробелов подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать русские буквы, пробелы, дефисы (не более 1 символа подряд).")

break

newNoteForActors.append(actorSurname)

#проверка даты рождения актера

if hasError == False:

if addActorActorBirthDayEntry.cget("text") != "Выберите дату рождения":

newNoteForActors.append(actorBirthDayDateTime)

else:

hasError = True

messagebox.showerror("Ошибка", "Дата рождения выбрана неверно! Повторите ввод!")

#проверка пола

actorGender = addActorActorGenderEntry.curselection()

if hasError == False:

if len(actorGender) == 0:

hasError = True

messagebox.showerror("Ошибка", "Вы не указали пол актера! Выберите пол актера в соответствующем поле!")

else:

if actorGender[0] == 0:

newNoteForActors.append("м")

else:

newNoteForActors.append("ж")

addActorInDBActorRolesDataList = []

#проверка полей с ролями актера в фильмах

if hasError == False:

forbiddenCharacters = []

for i in range(len(selectedFilmsIndexesList)):

actorCharacter = actorRolesEntryList[selectedFilmsRowIndexesList[i]].get()

if len(actorCharacter) != 0:

for s in actorCharacter:

if (s not in ruUpperLetters) and (s not in ruLowerLetters) and (s not in enUpperLetters) and (s not in enLowerLetters) and (s not in ['(', ')', ':', '-', '.', ',', '\n', ' ']) and (s.isdigit() == False):

hasError = True

if s not in forbiddenCharacters:

forbiddenCharacters.append(s)

print("Актер с индексом: ", addActorInDBActorIndex, "фильм с индексом: ", selectedFilmsIndexesList[i], "роль: ", actorCharacter, "строка номер: ", selectedFilmsRowIndexesList[i])

addActorInDBActorRolesDataList.append(tuple([int(addActorInDBActorIndex), int(selectedFilmsIndexesList[i]), str(actorCharacter)]))

if hasError == True:

messagebox.showerror("Ошибка ввода", "Вы ввели эти недопустимые символы при описании ролей актеров в фильме: {}! Вы можете указать роли актеров в фильме, используя английские и русские буквы, цифры, символы: пробел, '(', ')', ':', '-', '.', ',', либо оставить эти поля пустыми".format(forbiddenCharacters))

#добавление в БД

print("В таблицу 'Actors' БД будет добавлена запись: \n", newNoteForActors)

print("В таблицу 'Roles' БД будет добавлена запись: \n", addActorInDBActorRolesDataList)

#hasError = True

if hasError == False:

cursorAddActorInDB = conn.cursor()

cursorAddActorInDB.execute("INSERT INTO Actors VALUES (?, ?, ?, ?, ?)", tuple(newNoteForActors))

conn.commit()

for i in addActorInDBActorRolesDataList:

cursorAddActorRoleInDB = conn.cursor()

cursorAddActorRoleInDB.execute("INSERT INTO Role VALUES (?, ?, ?)", i)

conn.commit()

for i in actorRolesEntryList:

i.delete(0, END)

i["state"] = DISABLED

addActorActorNameEntry.delete(0, END)

addActorActorSurnameEntry.delete(0, END)

messagebox.showinfo("Успешно добавлен актер!", "Успешно добавлен актер %s %s, этот актер участвует в %d фильмах."%(newNoteForActors[1], newNoteForActors[2], len(addActorInDBActorRolesDataList)))

'''функция showEditActorsMenu отображает окно редактирования актеров и позволяет удалять и редактировать записи об актерах

'''

def showEditActorsMenu():

global fromAllActors, actorRolesLabelsList, fromActor, actorsInFilmDataFramesList, allActorsEditActorDataWidgetsList, editActorsDataAllActorsList

deleteAllFromThisWindow()

for i in actorDataLabelsList:

i.grid\_remove()

i.destroy()

actorDataLabelsList.clear()

#print("actorRolesLabelsList = ", actorRolesLabelsList)

for i in actorRolesLabelsList:

for j in range(len(i)):

i[j].grid\_forget()

actorRolesLabelsList.clear()

fromAllActors = True

queryGetAllActors = conn.cursor()

queryGetAllActors.execute('select \* from Actors')

editActorsDataAllActorsList = []

for row in queryGetAllActors:

editActorsDataAllActorsList.append(row)

print(editActorsDataAllActorsList)

editActorsheader.grid(row = 0, sticky=N+S+W+E)

editActorsbody.grid(row = 1, sticky=N+S+W+E)

editActorsfooter.grid(row = 2, sticky=N+S+W+E)

#scrollableBodyEditAllActors

#allFilmsHeader

editActorsInformationDescription.grid(row = 0, column = 0, columnspan = 5, sticky=N+S+W+E)

#allActorsInformationDescription["text"] = "Вы открыли окно редактирования информации об актерах\nНайдите актера, информацию о котором вы хотите отредактировать.\nНажмите на кнопку "🖉" для того, чтобы редактировать информацию об актере.\nНажмите кнопку "⌦", чтобы удалить информацию об актере из ПС."

editActorsInfoOptions.grid(row = 1, column = 0, sticky=N+S+W+E)

editActorsInfoName.grid(row = 1, column = 1, sticky=N+S+W+E)

editActorsInfoSurname.grid(row = 1, column = 2, sticky=N+S+W+E)

editActorsBirthDay.grid(row = 1, column = 3, sticky=N+S+W+E)

editActorsGender.grid(row = 1, column = 4, sticky=N+S+W+E)

allActorsEditActorDataWidgetsList = []

#allFilmsBody 🖉 ⌦

for i in range(len(editActorsDataAllActorsList)):

allActorsEditActorDataWidgetsList.append([])

allActorsEditActorDataWidgetsList[i].append(Button(scrollableBodyEditAllActors, text="🖉", bg="#FFE4C4", font = ("Consolas", 18), width = 3))

allActorsEditActorDataWidgetsList[i][0].grid(row = i, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

allActorsEditActorDataWidgetsList[i][0].bind('<Button-1>', editActorInfo)

allActorsEditActorDataWidgetsList[i].append(Button(scrollableBodyEditAllActors, text="⌦", bg="#FFE4C4", font = ("Consolas", 18), width = 3))

allActorsEditActorDataWidgetsList[i][1].grid(row = i, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

allActorsEditActorDataWidgetsList[i][1].bind('<Button-1>', deleteActorInfoFromDB)

allActorsEditActorDataWidgetsList[i].append(Label(scrollableBodyEditAllActors, text="%s"%(editActorsDataAllActorsList[i][1]), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 250, width = 25))

allActorsEditActorDataWidgetsList[i][2].grid(row = i, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

allActorsEditActorDataWidgetsList[i].append(Label(scrollableBodyEditAllActors, text="%s"%(editActorsDataAllActorsList[i][2]), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 250, width = 20))

allActorsEditActorDataWidgetsList[i][3].grid(row = i, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

allActorsEditActorDataWidgetsList[i].append(Label(scrollableBodyEditAllActors, text="%s"%(editActorsDataAllActorsList[i][3].strftime("%d.%m.%Y")), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 250, width = 20))

allActorsEditActorDataWidgetsList[i][4].grid(row = i, column = 4, sticky=N+S+W+E, padx = 5, pady = 5)

allActorsEditActorDataWidgetsList[i].append(Label(scrollableBodyEditAllActors, text="%s"%(editActorsDataAllActorsList[i][4]), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 250, width = 10))

allActorsEditActorDataWidgetsList[i][5].grid(row = i, column = 5, sticky=N+S+W+E, padx = 5, pady = 5)

#allFilmsFooter

backToActorsMenuFromEditActors.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E)

scrollableBodyEditAllActors.update()

'''функция deleteActorInfoFromDB удаляет всю информацию об актере из базы данных

'''

def deleteActorInfoFromDB(event):

global editActorsDataAllActorsList

print("нажата кнопка в ряду: ", event.widget.grid\_info())

userChooseThisActorRow = event.widget.grid\_info()["row"]

userChooseThisActorID = editActorsDataAllActorsList[userChooseThisActorRow][0]

userChooseThisActorData = editActorsDataAllActorsList[userChooseThisActorRow]

print("выбран актер: ", userChooseThisActorData, "его айди: ", userChooseThisActorID)

#actorNewBirthDayDate = userChooseThisActorData[3]

userAnswer = messagebox.askyesno("Попытка удаления актера из базы данных", "Вы выбрали опцию удаления из базы данных для актера %s %s! Вы действительно хотите удалить всю информацию об этом актере из базы данных?"%(userChooseThisActorData[1], userChooseThisActorData[2]))

if userAnswer:

#удаление всех ролей для актера

cursorDeleteAllActorRoleFromDB = conn.cursor()

cursorDeleteAllActorRoleFromDB.execute("{CALL deleteRoleOnActorID(?)}", userChooseThisActorID)

conn.commit()

#удаление данных об актере

cursorDeleteActorInfoFromDB = conn.cursor()

cursorDeleteActorInfoFromDB.execute("{CALL deleteActorOnActorID(?)}", userChooseThisActorID)

conn.commit()

messagebox.showinfo("Операция выполнена успешно!", "Актер %s %s и все связанные с ним данные были удалены из базы данных!"%(userChooseThisActorData[1], userChooseThisActorData[2]))

deleteAllFromThisWindow()

showEditActorsMenu()

else:

messagebox.showinfo("Операция отменена!", "Актер %s %s НЕ был удален из базы данных!"%(userChooseThisActorData[1], userChooseThisActorData[2]))

'''функция editActorInfo вызывает окно, в котором пользовать может редактировать информацию выбранного актера

'''

def editActorInfo(event):

global allActorsEditActorDataWidgetsList, editActorsDataAllActorsList

global labelsListWithFilmsForAddActorInDB, checkBoxIntVarListForAddActorInDB, selectedNewFilmsIndexesList, allFilmsDataList, selectedFilmsRowIndexesList, actorRolesEntryList

global userChooseThisActorData, actorNewBirthDayDate

#получаем информацию о том, какого актера выбрали

print("нажата кнопка в ряду: ", event.widget.grid\_info())

userChooseThisActorRow = event.widget.grid\_info()["row"]

userChooseThisActorID = editActorsDataAllActorsList[userChooseThisActorRow][0]

userChooseThisActorData = editActorsDataAllActorsList[userChooseThisActorRow]

print("выбран актер: ", userChooseThisActorData, "его айди: ", userChooseThisActorID)

actorNewBirthDayDate = userChooseThisActorData[3]

#очищаем окно от всех виджетов

deleteAllFromThisWindow()

print("Вы находитесь в режиме редактирования информации об актере")

deleteAllFromThisWindow()

for i in actorRolesEntryList:

i.delete(0, END)

i["state"] = DISABLED

addActorActorNameEntry.delete(0, END)

addActorActorSurnameEntry.delete(0, END)

editorAddNoteHeader.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editorAddNoteBody.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editorAddNoteFooter.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

getAllFilmsFromDataBase = """\

SELECT \* FROM getAllFilmsFromDBForAddActor

"""

queryGetAllFilmsFromDataBase = conn.cursor()

queryGetAllFilmsFromDataBase.execute(getAllFilmsFromDataBase)

allFilmsDataList = []

for note in queryGetAllFilmsFromDataBase:

allFilmsDataList.append(note)

filmsCount = len(allFilmsDataList)

print("данные всех фильмов: ", allFilmsDataList, "\n")

editActorDescriptionLabel.grid(row = 0, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

editActorActorNameLabel.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editActorActorSurnameLabel.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editActorActorBirthDayLabel.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editActorActorGenderLabel.grid(row = 4, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editActorActorNameEntry.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

editActorActorNameEntry.insert(0, userChooseThisActorData[1])

editActorActorSurnameEntry.grid(row = 2, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

editActorActorSurnameEntry.insert(0, userChooseThisActorData[2])

editActorNewActorBirthDayEntry.grid(row = 3, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

editActorActorGenderEntry.grid(row = 4, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

editActorNewActorBirthDayEntry["text"] = userChooseThisActorData[3].strftime("%Y-%m-%d")

editActorChooseRolesDescription.grid(row = 5, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

selectedNewFilmsIndexesList = []

selectedFilmsRowIndexesList = []

#здесь

editActorChooseRoleCheckboxLabel.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editActorChooseRoleFilmNameLabel.grid(row = 0, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

editActorChooseRoleFilmYearLabel.grid(row = 0, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editActorChooseRoleFilmStyleLabel.grid(row = 0, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

editActorChooseRoleFilmAgeLimitLabel.grid(row = 0, column = 4, sticky=N+S+W+E, padx = 5, pady = 5)

editActorChooseRoleActorRoleLabel.grid(row = 0, column = 5, sticky=N+S+W+E, padx = 5, pady = 5)

#получение информации о всех имеющихся ролях для редактируемого актера

getRolesForActorFromDataBase = """\

{CALL getRolesForActor(?)}

"""

queryGetAllRoleForActor = conn.cursor()

queryGetAllRoleForActor.execute(getRolesForActorFromDataBase, userChooseThisActorID)

allActorRolesList = []

for note in queryGetAllRoleForActor:

allActorRolesList.append(note)

roleCount = len(allActorRolesList)

print("актер снимался в фильмах", allActorRolesList)

for i in range(filmsCount):

labelsListWithFilmsForAddActorInDB.append([])

isActorInThisFilm = False

#учавствовал ли актер в этом фильме

for role in allActorRolesList:

if allFilmsDataList[i][0] == role[1]:

isActorInThisFilm = True

actorCharacterInThisFilm = role[2]

for j in range(len(allFilmsDataList[i]) + 1):

if j == 0:

checkBoxIntVarListForAddActorInDB.append(IntVar())

labelsListWithFilmsForAddActorInDB[i].append(Checkbutton(scrollableBodyEditActor, text='', variable=checkBoxIntVarListForAddActorInDB[i], onvalue=1, offvalue=0, bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].bind('<Button-1>', chooseNewFilmForActor)#lambda event, param = event.widget.grid\_info()["row"]:

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i+1, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

if isActorInThisFilm == True:

checkBoxIntVarListForAddActorInDB[i].set(1)

selectedNewFilmsIndexesList.append(allFilmsDataList[i][0])

selectedFilmsRowIndexesList.append(i)

elif j == 5:

actorRolesEntryList.append(Entry(scrollableBodyEditActor, font = ("Consolas", 14), state = DISABLED, width = 45))

actorRolesEntryList[i].grid(row = i+1, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

if isActorInThisFilm == True:

actorRolesEntryList[i]["state"] = NORMAL

actorRolesEntryList[i].insert(0, actorCharacterInThisFilm)

else:

labelsListWithFilmsForAddActorInDB[i].append(Label(scrollableBodyEditActor, text = "%s"%(allFilmsDataList[i][j]), wraplength = 300, font = ("Consolas", 20), bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i+1, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

scrollableBodyEditActor.update()

editActorFinishDescription.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E)

returnBackToActorMenuFromActorEdit.grid(row = 1, column = 0, sticky=N+S+W+E)

editActorNoteInDataBase.grid(row = 1, column = 1, sticky=N+S+W+E)

'''Фукнция выбора значения в чекбоксе для роли актера в фильме при занесении в БД информации об актере

'''

def chooseNewFilmForActor(event):

global selectedNewFilmsIndexesList, allFilmsDataList, selectedFilmsRowIndexesList, actorRolesEntryList

print(event.widget.grid\_info())

rowIndex = event.widget.grid\_info()["row"] - 1

print("selectedNewFilmsIndexesList: ", selectedNewFilmsIndexesList)

print("selectedFilmsRowIndexesList: ", selectedFilmsRowIndexesList)

print('allFilmsDataList[rowIndex][0]: ', allFilmsDataList[rowIndex][0])

if checkBoxIntVarListForAddActorInDB[rowIndex].get() == 0:

actorRolesEntryList[rowIndex]["state"] = NORMAL

selectedNewFilmsIndexesList.append(allFilmsDataList[rowIndex][0])

selectedFilmsRowIndexesList.append(rowIndex)

print("выбран индекс: %d"%rowIndex)

print("индекс выбранных фильмов:", selectedNewFilmsIndexesList)

print("индекс выбранных строк:", selectedFilmsRowIndexesList)

else:

selectedNewFilmsIndexesList.remove(allFilmsDataList[rowIndex][0])

selectedFilmsRowIndexesList.remove(rowIndex)

actorRolesEntryList[rowIndex].delete(0, END)

actorRolesEntryList[rowIndex]["state"] = DISABLED

print("индекс выбранных фильмов:", selectedNewFilmsIndexesList)

print("индекс выбранных строк:", selectedFilmsRowIndexesList)

'''Фукнция chooseNewDateForActor позволяет установить новую дату рождения при редактировании в БД информации об актере

'''

def chooseNewDateForActor():

global actorNewBirthDayDate, userChooseThisActorData

def print\_sel():

global actorNewBirthDayDate

print(calendarForChooseDate.selection\_get())

#calendarForChooseDate.see(datetime.date(year=2016, month=2, day=5))

actorNewBirthDayDate = calendarForChooseDate.selection\_get()

editActorNewActorBirthDayEntry["text"] = actorNewBirthDayDate

returnBackToActorsMenu["state"] = NORMAL

editActorNoteInDataBase["state"] = NORMAL

top.destroy()

returnBackToActorsMenu["state"] = DISABLED

editActorNoteInDataBase["state"] = DISABLED

top = Toplevel(root)

top.geometry("+300+300")

top.overrideredirect(True) #запрет на взаимодействие с панелью инструментов окна

today = datetime.date.today()

mindate = datetime.date(year = 1800, month=1, day=21)

maxdate = today# + datetime.timedelta(days=5)

calendarInfoLabel = Label(top, text = "Выберите дату рождения актера:\nгод, месяц и число.\nПосле этого нажмите\nна кнопку 'Выбрать эту дату'", font = ("Consolas", 16), bg= "lightgreen")

calendarInfoLabel.pack(fill="both", expand=True)

calendarForChooseDate = Calendar(top, font="Arial 14", selectmode='day', locale='ru\_RU', mindate=mindate, maxdate=maxdate, disabledforeground='red', cursor="hand1", year = 1975, month = 1, day = 1, date\_pattern = "y-mm-dd")

calendarForChooseDate.pack(fill="both", expand=True)

calendarForChooseDate.see(userChooseThisActorData[3])

Button(top, text="Выбрать эту дату", font = ("Consolas", 16), command=print\_sel).pack()

"""Функция checkNewActorDataAndAddItInDB проверяет правильность введенных в форму актера данных и заносит их в БД

"""

def checkNewActorDataAndAddItInDB():

global actorNewBirthDayDate, selectedNewFilmsIndexesList, selectedFilmsRowIndexesList, actorRolesEntryList, userChooseThisActorData

hasError = False

newNoteForActors = []

#получение информации о всех имеющихся актерах

getAllActorsFromDataBase = """\

SELECT \* FROM getAllActors

"""

queryGetAllActorsIDFromDataBase = conn.cursor()

queryGetAllActorsIDFromDataBase.execute(getAllActorsFromDataBase)

allActorsList = []

for note in queryGetAllActorsIDFromDataBase:

allActorsList.append(note)

actorsCount = len(allActorsList)

print(allActorsList)

#получение индекса актера

# maxActorIndex = -5

# for i in range(actorsCount):

# if allActorsList[i][0] > maxActorIndex:

# maxActorIndex = allActorsList[i][0]

# print("Максимальный индекс в списке: %d"%maxActorIndex)

addActorInDBActorIndex = userChooseThisActorData[0]

newNoteForActors.append(addActorInDBActorIndex)

ruUpperLetters = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

ruLowerLetters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

enUpperLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

enLowerLetters = "abcdefghijklmnopqrstuvwxyz"

#проверка имени актера

actorName = editActorActorNameEntry.get()

lenActorName = len(actorName)

if not actorName:

messagebox.showerror("Ошибка", "Вы не ввели имя актера! Повторите ввод!")

editActorActorNameEntry.delete(0, END)

editActorActorNameEntry.focus()

hasError = True

elif actorName[0] == " ":

messagebox.showerror("Ошибка", "Вы ввели пробел перед именем! Повторите ввод!")

editActorActorNameEntry.delete(0, END)

editActorActorNameEntry.focus()

hasError = True

elif not actorName[0] == actorName[0].upper():

messagebox.showerror("Ошибка", "Вы ввели имя актера с маленькой буквы! Повторите ввод!")

editActorActorNameEntry.focus()

hasError = True

else:

nizya = r'\~\`\!\@"\#\№\$\;\%\^\:\&\?\(\)\\_\+\=\<\>\,\.\[\]\{\}\\\/\|\\*\''

find\_nizya = re.compile('([{}])'.format(nizya))

res = find\_nizya.findall(actorName)

myString = ', '.join(res)

print(res, myString)

if res:

messagebox.showerror('ОШИБКА', 'Вы ввели имя актера с запрещёнными символами, такими как: {}.'.format(

myString) + ' ' + 'Пожалуйста, не вводите символы: ~, `, @, ", #, №, $, ;, %, ^, :, &, ?, \*, (, ), \_, +, =, [, ], {, }, |, \, /, ., , , <, >')

#CityTeamEntry.delete(0, END)

hasError = True

editActorActorNameEntry.focus()

else:

nizya1 = 'a-zA-Z'

find\_nizya1 = re.compile('([{}])'.format(nizya1))

res1 = find\_nizya1.findall(actorName)

myString1 = ', '.join(res1)

print(res1, myString1)

if res1:

hasError = True

messagebox.showerror('ОШИБКА',

'Вы ввели имя актера с помощью латинских букв, таких как: {}.'.format(

myString1) + ' ' + 'Пожалуйста, не используйте латинские буквы.')

#CityTeamEntry.delete(0, END)

editActorActorNameEntry.focus()

else:

if actorName[lenActorName - 1] == " ":

hasError = True

messagebox.showerror('ОШИБКА', 'Вы ввели символы пробела после имени актера')

editActorActorNameEntry.focus()

else:

i = 0

while actorName[i] == ' ':

actorName = actorName[1:]

while actorName[

len(actorName) - 1] == ' ':

actorName = actorName[:-1]

i = 1

while i < len(actorName) - 1:

if actorName[i] == ' ' and actorName[i + 1] == ' ':

actorName = actorName[:i + 1] + actorName[i + 2:]

else:

i += 1

hasActorName = True

editActorActorNameEntry.delete(0, END)

editActorActorNameEntry.insert(0, actorName)

newNoteForActors.append(actorName)

#проверка фамилии актера

actorSurname = editActorActorSurnameEntry.get()

defisCount = 0

spaceCount = 0

if hasError == False:

if len(actorSurname) == 0:

hasError = True

messagebox.showerror("Ошибка", "Вы не ввели фамилию актера! Вы можете использовать русские буквы, пробел и дефисы. Повторите ввод!")

else:

if (actorSurname[0] in ['-', ' '] or actorSurname[-1] in ['-', ':', ' ']):

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера не должна начинаться или оканчиваться на пробел или '-'! Введите имя актера рускими буквами!\nВы можете использовать русские буквы, пробелы и дефисы (не более 1 символа подряд).")

else:

if actorSurname[0] in ruLowerLetters:

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера написана со строчной буквы! Введите имя актера с заглавной буквы!")

else:

for i in actorSurname:

if i not in ruUpperLetters and i not in ruLowerLetters and i not in ["-", " "]:

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера введена неверно!\nВы можете использовать русские буквы, пробелы и дефисы (не более 1 символа подряд).")

break

else:

if defisCount == 0 and i == "-":

defisCount +=1

elif defisCount == 1 and i != "-":

defisCount = 0

elif defisCount == 1 and i == "-":

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера введена неверно! Вы ввели слишком много дефисов подряд! Вы можете использовать русские буквы, пробелы, дефисы (не более 1 символа подряд).")

break

if spaceCount == 0 and i == " ":

spaceCount +=1

elif spaceCount == 1 and i != " ":

spaceCount = 0

elif spaceCount == 1 and i == " ":

hasError = True

messagebox.showerror("Ошибка ввода", "Фамилия актера введена неверно! Вы ввели слишком много пробелов подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать русские буквы, пробелы, дефисы (не более 1 символа подряд).")

break

newNoteForActors.append(actorSurname)

#проверка даты рождения актера

if hasError == False:

if editActorNewActorBirthDayEntry.cget("text") != "Выберите дату рождения":

newNoteForActors.append(actorNewBirthDayDate)

else:

hasError = True

messagebox.showerror("Ошибка", "Дата рождения выбрана неверно! Повторите ввод!")

#проверка пола

actorGender = editActorActorGenderEntry.curselection()

if hasError == False:

if len(actorGender) == 0:

hasError = True

messagebox.showerror("Ошибка", "Вы не указали пол актера! Выберите пол актера в соответствующем поле!")

else:

if actorGender[0] == 0:

newNoteForActors.append("м")

else:

newNoteForActors.append("ж")

addActorInDBActorRolesDataList = []

#проверка полей с ролями актера в фильмах

if hasError == False:

forbiddenCharacters = []

for i in range(len(selectedNewFilmsIndexesList)):

actorCharacter = actorRolesEntryList[selectedFilmsRowIndexesList[i]].get()

if len(actorCharacter) != 0:

for s in actorCharacter:

if (s not in ruUpperLetters) and (s not in ruLowerLetters) and (s not in enUpperLetters) and (s not in enLowerLetters) and (s not in ['(', ')', ':', '-', '.', ',', '\n', ' ']) and (s.isdigit() == False):

hasError = True

if s not in forbiddenCharacters:

forbiddenCharacters.append(s)

print("Актер с индексом: ", addActorInDBActorIndex, "фильм с индексом: ", selectedNewFilmsIndexesList[i], "роль: ", actorCharacter, "строка номер: ", selectedFilmsRowIndexesList[i])

addActorInDBActorRolesDataList.append(tuple([int(addActorInDBActorIndex), int(selectedNewFilmsIndexesList[i]), str(actorCharacter)]))

if hasError == True:

messagebox.showerror("Ошибка ввода", "Вы ввели эти недопустимые символы при описании ролей актеров в фильме: {}! Вы можете указать роли актеров в фильме, используя английские и русские буквы, цифры, символы: пробел, '(', ')', ':', '-', '.', ',', либо оставить эти поля пустыми".format(forbiddenCharacters))

#добавление в БД

print("В таблицу 'Actors' БД будет добавлена запись: \n", newNoteForActors)

print("В таблицу 'Roles' БД будет добавлена запись: \n", addActorInDBActorRolesDataList)

#hasError = True

if hasError == False:

#обновление записи в таблице актеров

cursorAddActorInDB = conn.cursor()

cursorAddActorInDB.execute("{CALL updateActorInfo(?, ?, ?, ?, ?)}", newNoteForActors[1], newNoteForActors[2], newNoteForActors[3], newNoteForActors[4], newNoteForActors[0])

#"INSERT INTO Actors VALUES (?, ?, ?, ?, ?)")#updateActorInfo

conn.commit()

getAllRolesDataFromDB = """\

SELECT \* FROM Role

"""

queryGetAllRolesDataFromDB = conn.cursor()

queryGetAllRolesDataFromDB.execute(getAllRolesDataFromDB)

#allRolesInDBList = []

#oldRolesForNowEditedActor = []

#удаление всех ролей для актера, которого мы сейчас редактируем

cursorDeleteAllActorsRoleFromDB = conn.cursor()

cursorDeleteAllActorsRoleFromDB.execute("{CALL deleteRoleOnActorID(?)}", newNoteForActors[0])

conn.commit()

# for note in queryGetAllRolesDataFromDB:

# allRolesInDBList.append(note)

# if note[0] == newNoteForActors[0]:#deleteRoleOnActorID

# rolesCount = len(allRolesInDBList)

# print("all roles: ", allRolesInDBList)

#занесение записей о ролях в БД

for i in addActorInDBActorRolesDataList:

cursorAddActorRoleInDB = conn.cursor()

cursorAddActorRoleInDB.execute("INSERT INTO Role VALUES (?, ?, ?)", i)

conn.commit()

for i in actorRolesEntryList:

i.delete(0, END)

i["state"] = DISABLED

editActorActorNameEntry.delete(0, END)

editActorActorSurnameEntry.delete(0, END)

messagebox.showinfo("Успешно отредактированы данные актера!", "Успешно отредактированы данные актера %s %s, этот актер участвует в %d фильмах."%(newNoteForActors[1], newNoteForActors[2], len(addActorInDBActorRolesDataList)))

showEditActorsMenu()

'''функция addFilmToDataBase выполняет переход в меню добавления фильма в БД

'''

def addFilmToDataBase():

global checkBoxIntVarListForAddFilmInDB, selectedActorsIndexesList, selectedActorsRowIndexesList, allActorsList, startRowForAllActorsRows

deleteAllFromThisWindow()

selectedActorsIndexesList = []

selectedActorsRowIndexesList = []

filmAddNoteHeader.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmAddNoteBody.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

filmAddNoteFooter.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

#header

addFilmDescriptionLabel.grid(row = 0, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

#body

addFilmFilmNameLabel.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmYearLabel.grid(row = 1, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmLengthLabel.grid(row = 2, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmStyleLabel.grid(row = 3, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmAgeLimitLabel.grid(row = 4, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmCostLabel.grid(row = 5, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmUSAProfitLabel.grid(row = 6, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmWorldProfitLabel.grid(row = 7, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmAwardsLabel.grid(row = 8, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoDataFieldColumnspan = 4

addFilmFilmNameDataField.grid(row = 0, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmYearDataField.grid(row = 1, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmLengthDataField.grid(row = 2, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmStyleDataField.grid(row = 3, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmAgeLimitDataField.grid(row = 4, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmCostDataField.grid(row = 5, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmUSAProfitDataField.grid(row = 6, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmWorldProfitDataField.grid(row = 7, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmFilmAwardsDataField.grid(row = 8, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmChooseRolesDescription.grid(row = 9, column = 0, columnspan = filmInfoDataFieldColumnspan + 2, sticky=N+S+W+E, padx = 5, pady = 5)

#получение информации о всех имеющихся актерах

getAllActorsFromDataBase = """\

SELECT \* FROM getAllActors

"""

queryGetAllActorsIDFromDataBase = conn.cursor()

queryGetAllActorsIDFromDataBase.execute(getAllActorsFromDataBase)

allActorsList = []

for note in queryGetAllActorsIDFromDataBase:

allActorsList.append(note)

actorsCount = len(allActorsList)

#print(allActorsList, actorsCount)

addFilmChooseRoleCheckboxLabel.grid(row = 10, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmChooseRoleFilmNameLabel.grid(row = 10, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmChooseRoleFilmYearLabel.grid(row = 10, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmChooseRoleFilmStyleLabel.grid(row = 10, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmChooseRoleFilmAgeLimitLabel.grid(row = 10, column = 4, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmChooseRoleActorRoleLabel.grid(row = 10, column = 5, sticky=N+S+W+E, padx = 5, pady = 5)

startRowForAllActorsRows = 11

if len(addFilmActorRolesEntryList) != 0:

for i in addFilmActorRolesEntryList:

i.destroy()

addFilmActorRolesEntryList.clear()

for i in range(actorsCount):

labelsListWithFilmsForAddActorInDB.append([])

for j in range(len(allActorsList[i]) + 1):

if j == 0:

checkBoxIntVarListForAddFilmInDB.append(IntVar())

labelsListWithFilmsForAddActorInDB[i].append(Checkbutton(scrollableBodyChooseFilmsForfilm, text='', variable = checkBoxIntVarListForAddFilmInDB[i], onvalue=1, offvalue=0, bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].bind('<Button-1>', chooseActorForFilm)#lambda event, param = event.widget.grid\_info()["row"]:

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i + startRowForAllActorsRows, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

elif j == 3:

labelsListWithFilmsForAddActorInDB[i].append(Label(scrollableBodyChooseFilmsForfilm, text = "%s"%(allActorsList[i][j].strftime("%d.%m.%Y")), wraplength = 300, font = ("Consolas", 20), bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i + startRowForAllActorsRows, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

elif j == 5:

addFilmActorRolesEntryList.append(Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 14), state = DISABLED, width = 45))

addFilmActorRolesEntryList[i].grid(row = i + startRowForAllActorsRows, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

else:

labelsListWithFilmsForAddActorInDB[i].append(Label(scrollableBodyChooseFilmsForfilm, text = "%s"%(allActorsList[i][j]), wraplength = 300, font = ("Consolas", 20), bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i + startRowForAllActorsRows, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

#активация бесконечной прокрутки в области полей данных о фильме

scrollableBodyChooseFilmsForfilm.update()

#Footer

editFilmFinishDescription.grid(row = 0, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

returnBackTofilmsMenu.grid(row = 1, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

addFilmNoteInDataBase.grid(row = 1, column = 2, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

'''Функция chooseActorForFilm вызывается при нажатии чекбокса в окне добавления фильма

при выборе актера для этого фильма, сохраняет в список данные о выбранном актере

'''

def chooseActorForFilm(event):

global allActorsList, selectedActorsIndexesList, selectedActorsRowIndexesList, addFilmActorRolesEntryList, startRowForAllActorsRows

print("chooseActorForFilm")

print(event.widget.grid\_info())

rowIndex = event.widget.grid\_info()["row"] - startRowForAllActorsRows

if checkBoxIntVarListForAddFilmInDB[rowIndex].get() == 0:

addFilmActorRolesEntryList[rowIndex]["state"] = NORMAL

selectedActorsIndexesList.append(allActorsList[rowIndex][0])

selectedActorsRowIndexesList.append(rowIndex)

print("выбран индекс: %d"%rowIndex)

print("индекс выбранных актеров:", selectedActorsIndexesList)

print("индекс выбранных строк:", selectedActorsRowIndexesList)

else:

selectedActorsIndexesList.remove(allActorsList[rowIndex][0])

selectedActorsRowIndexesList.remove(rowIndex)

addFilmActorRolesEntryList[rowIndex]["state"] = DISABLED

print("индекс выбранных актеров:", selectedActorsIndexesList)

print("индекс выбранных строк:", selectedActorsRowIndexesList)

'''Функция checkNewFilmDataNoteAndAddItInDB проверяет введенные пользователем данные о фильме

и, в случае их корректности, записывает данные в базу данных, иначе возвращает ошибку

'''

def checkNewFilmDataNoteAndAddItInDB():

global selectedActorsIndexesList, selectedActorsRowIndexesList, addFilmActorRolesEntryList, addFilmAgeLimitComboboxValue

hasError = False

newFilmData = []

newRoleData = []

#получение индекса для добавляемого фильма

getAllFilmsFromDataBase = """\

SELECT \* FROM Films

"""

queryGetAllFilmsFromDataBase = conn.cursor()

queryGetAllFilmsFromDataBase.execute(getAllFilmsFromDataBase)

allFilmsList = []

for note in queryGetAllFilmsFromDataBase:

allFilmsList.append(note)

filmsCount = len(allFilmsList)

#print(allFilmsList)

maxFilmIndex = -5

if filmsCount != 0:

for i in range(filmsCount):

if allFilmsList[i][0] > maxFilmIndex:

maxFilmIndex = allFilmsList[i][0]

else:

maxFilmIndex = -1

addActorInDBFilmIndex = maxFilmIndex + 1

newFilmData.append(addActorInDBFilmIndex)

print("Фильму будет присвоен индекс: %d"%maxFilmIndex)

ruUpperLetters = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

ruLowerLetters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

enUpperLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

enLowerLetters = "abcdefghijklmnopqrstuvwxyz"

specSymbolsForName = "-: "

#проверка названия фильма

filmName = addFilmFilmNameDataField.get()

defisCount = 0

spaceCount = 0

columnCount = 0

if len(filmName) == 0:

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма не указано! Введите русское название фильма!\nВы можете использовать русские буквы, пробелы, дефисы и двоеточия.")

else:

if (filmName[0] in ['-', ':', ' '] or filmName[-1] in ['-', ':', ' ']):

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма не должно начинаться или оканчиваться на пробел или символы '-', ':'! Введите русское название фильма!\nВы можете использовать русские буквы, пробелы, дефисы и двоеточия.")

else:

for i in filmName:

if i not in ruUpperLetters and i not in ruLowerLetters and i not in specSymbolsForName:

hasError = True

#addFilmFilmNameDataField.delete(0, END)

messagebox.showerror("Ошибка ввода", "Название фильма введено неверно, введите русское название фильма!\nВы можете использовать русские буквы, пробелы, дефисы и двоеточия (не более 1 символа подряд).")

break

else:

if defisCount == 0 and i == "-":

defisCount +=1

elif defisCount == 1 and i != "-":

defisCount = 0

elif defisCount == 1 and i == "-":

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма введено неверно! Вы ввели слишком много дефисов подряд! Вы можете использовать русские буквы, пробелы, дефисы и двоеточия (не более 1 символа подряд).")

break

if spaceCount == 0 and i == " ":

spaceCount +=1

elif spaceCount == 1 and i != " ":

spaceCount = 0

elif spaceCount == 1 and i == " ":

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма введено неверно! Вы ввели слишком много пробелов подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать русские буквы, пробелы, дефисы и двоеточия (не более 1 символа подряд).")

break

if columnCount == 0 and i == ":":

columnCount+=1

elif columnCount == 1 and i != ":":

columnCount = 0

elif columnCount == 1 and i == ":":

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма введено неверно! Вы ввели слишком много символов ':' подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать русские буквы, пробелы, дефисы и двоеточия (не более 1 символа подряд).")

break

newFilmData.append(filmName)

for i in range(filmsCount):

if allFilmsList[i][1] == filmName:

hasError = True

messagebox.showerror("Фильм с таким названием уже есть в ПС", "Фильм с таким названием уже есть в базе данных! Занесите в базу данных информацию о другом фильме!")

break

#проверка года выхода фильма

filmYear = addFilmFilmYearDataField.get()

if hasError == False:

if len(filmYear) != 0:

try:

filmYear = int(filmYear)

except ValueError:

hasError = True

#addFilmFilmYearDataField.delete(0, END)

messagebox.showerror("Ошибка ввода", "Год выпуска фильма введен неверно! Укажите год выхода фильма числом. Вы можете указать год выхода с 1880 года по текущий в момент работы программы.")

return

today = datetime.date.today()

minYear = 1880

maxYear = today.strftime("%Y")

if not (filmYear >= minYear and filmYear <= int(maxYear)):

hasError = True

#addFilmFilmYearDataField.delete(0, END)

messagebox.showerror("Ошибка ввода", "Год выхода фильма указан неверно, вы можете указать год выхода с 1880 по текущий в момент работы программы.")

else:

hasError = True

messagebox.showerror("Ошибка ввода", "Вы не указали год выхода фильма! Вы можете указать год выхода с 1880 года по текущий в момент работы программы.")

newFilmData.append(filmYear)

#проверка продолжительности фильма

filmLength = addFilmFilmLengthDataField.get()

if hasError == False:

if len(filmLength) != 0:

try:

filmLength = int(filmLength)

except ValueError:

hasError = True

#filmLength.delete(0, END)

messagebox.showerror("Ошибка ввода", "Длительность фильма указана неверно! Вы должны указать длительность фильма в минутах от 1 до 100000.")

return

if not (filmLength >= 1 and filmLength <= 100000):

hasError = True

#addFilmFilmYearDataField.delete(0, END)

messagebox.showerror("Ошибка ввода", "Вы неправильно указали длительность фильма! Вы должны указать длительность фильма в минутах от 1 до 100000.")

else:

hasError = True

messagebox.showerror("Ошибка ввода", "Вы не указали длительность фильма в минутах! Вы должны указать длительность фильма в минутах от 1 до 100000.")

newFilmData.append(filmLength)

#проверка жанра фильма

filmStyle = addFilmFilmStyleDataField.get()

if hasError == False:

if len(filmStyle) == 0:

hasError = True

messagebox.showerror("Ошибка ввода", "Жанр фильма не указан! Введите жанр фильма строчными русскими буквами, вы можете использовать дефис и пробел.")

else:

defisCount = 0

spaceCount = 0

for i in filmStyle:

if (i not in ruLowerLetters) and i != "-" and i != " ":

hasError = True

messagebox.showerror("Ошибка ввода", "Жанр фильма указан неверно! Введите жанр фильма строчными русскими буквами, вы можете использовать дефис и пробел.")

break

else:

if defisCount == 0 and i == "-":

defisCount +=1

elif defisCount == 1 and i != "-":

defisCount = 0

elif defisCount == 1 and i == "-":

hasError = True

messagebox.showerror("Ошибка ввода", "Жанр фильма указан неверно! Вы ввели слишком много дефисов подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать дефис и пробел.")

break

if spaceCount == 0 and i == " ":

spaceCount +=1

elif spaceCount == 1 and i != " ":

spaceCount = 0

elif spaceCount == 1 and i == " ":

hasError = True

messagebox.showerror("Ошибка ввода", "Жанр фильма указан неверно! Вы ввели слишком много пробелов подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать дефис и пробел.")

break

newFilmData.append(str(filmStyle))

#проверка возрастного ограничения фильма

#filmAge = addFilmFilmAgeLimitDataField.get()

filmAge = addFilmAgeLimitComboboxValue.get()

if hasError == False:

if filmAge not in [u"0", u"3", u"6", u"12", u"14", u"16", u"18"]:

hasError = True

messagebox.showerror("Ошибка ввода", "Возрастное ограничение фильма указано неверно! Выберите возрастное ограничение строго из списка значений.")

newFilmData.append(int(filmAge))

#проверка бюджета фильма

filmCost = addFilmFilmCostDataField.get()

if hasError == False:

if len(filmCost) != 0:

try:

filmCost = int(filmCost)

except ValueError:

hasError = True

messagebox.showerror("Ошибка ввода", "Бюджет фильма указан неверно! Вы должны указать бюджет фильма в долларах от 1 доллара до 10 миллиардов долларов.")

return

if not (filmCost >= 1 and filmCost <= 10000000000):

hasError = True

messagebox.showerror("Ошибка ввода", "Вы неправильно указали бюджет фильма! Вы должны указать бюджет фильма в долларах от 1 доллара до 10 миллиардов долларов.")

else:

hasError = True

messagebox.showerror("Ошибка ввода", "Вы не указали бюджет фильма в долларах! Вы должны указать бюджет фильма в долларах от 1 доллара до 10 миллиардов долларов.")

newFilmData.append(filmCost)

#проверка дохода в США

filmUSAProfit = addFilmFilmUSAProfitDataField.get()

if hasError == False:

if len(filmUSAProfit) != 0:

try:

filmUSAProfit = int(filmUSAProfit)

except ValueError:

hasError = True

messagebox.showerror("Ошибка ввода", "Сборы фильма в США указаны неверно! Вы должны указать сборы фильма в США в долларах от 0 долларов до 10 миллиардов долларов.")

return

if not (filmUSAProfit >= 0 and filmUSAProfit <= 10000000000):

hasError = True

messagebox.showerror("Ошибка ввода", "Вы неправильно указали сборы фильма в США! Вы должны указать сборы фильма в США в долларах от 0 долларов до 10 миллиардов долларов.")

else:

hasError = True

messagebox.showerror("Ошибка ввода", "Вы не указали сборы фильма в США в долларах! Вы должны указать сборы фильма в США в долларах от 0 долларов до 10 миллиардов долларов.")

newFilmData.append(filmUSAProfit)

#проверка дохода в мире

filmWorldProfit = addFilmFilmWorldProfitDataField.get()

if hasError == False:

if len(filmWorldProfit) != 0:

try:

filmWorldProfit = int(filmWorldProfit)

except ValueError:

hasError = True

messagebox.showerror("Ошибка ввода", "Сборы фильма в мире указаны неверно! Вы должны указать сборы фильма в мире в долларах от 0 долларов до 10 миллиардов долларов, либо оставить поле пустым")

return

if not (filmWorldProfit >= 0 and filmWorldProfit <= 10000000000):

hasError = True

messagebox.showerror("Ошибка ввода", "Вы неправильно указали сборы фильма в мире! Вы должны указать сборы фильма в мире в долларах от 0 долларов до 10 миллиардов долларов, либо оставить поле пустым")

else:

filmWorldProfit = 0

newFilmData.append(filmWorldProfit)

#проверка наград

filmAwards = addFilmFilmAwardsDataField.get(1.0, END)

print("filmAwards = ", filmAwards)

forbiddenCharacters = []

if hasError == False:

if len(filmAwards) != 0:

for i in filmAwards:

if (i not in ruUpperLetters) and (i not in ruLowerLetters) and (i not in enUpperLetters) and (i not in enLowerLetters) and (i not in ['(', ')', ':', '-', '.', ',', '\n', ' ']) and (i.isdigit() == False):

hasError = True

if i not in forbiddenCharacters:

forbiddenCharacters.append(i)

if hasError == True:

messagebox.showerror("Ошибка ввода", "Вы ввели эти недопустимые символы при описании наград фильма: {}! Вы должны указать награды фильма, используя английские и русские буквы, цифры, символы: пробел, '(', ')', ':', '-', '.', ',', либо оставить поле пустым".format(forbiddenCharacters))

if filmAwards == "\n":

filmAwards = ""

newFilmData.append(str(filmAwards))

#формирование данных для таблицы ролей

if hasError == False:

forbiddenCharacters = []

for i in range(len(selectedActorsIndexesList)):

actorCharacter = addFilmActorRolesEntryList[selectedActorsRowIndexesList[i]].get()

if len(actorCharacter) != 0:

for s in actorCharacter:

if (s not in ruUpperLetters) and (s not in ruLowerLetters) and (s not in enUpperLetters) and (s not in enLowerLetters) and (s not in ['(', ')', ':', '-', '.', ',', '\n', ' ']) and (s.isdigit() == False):

hasError = True

if s not in forbiddenCharacters:

forbiddenCharacters.append(s)

print("Актер с индексом: ", selectedActorsIndexesList[i], "фильм с индексом: ", addActorInDBFilmIndex, "роль: ", actorCharacter, "строка номер: ", selectedActorsRowIndexesList[i])

newRoleData.append(tuple([int(selectedActorsIndexesList[i]), int(addActorInDBFilmIndex), str(actorCharacter)]))

if hasError == True:

messagebox.showerror("Ошибка ввода", "Вы ввели эти недопустимые символы при описании ролей актеров в фильме: {}! Вы можете указать роли актеров в фильме, используя английские и русские буквы, цифры, символы: пробел, '(', ')', ':', '-', '.', ',', либо оставить эти поля пустыми".format(forbiddenCharacters))

#добавление фильма в БД

print("В таблицу 'Films' БД будет добавлена запись: \n", newFilmData)

print("В таблицу 'Role' БД будет добавлено: \n", newRoleData)

#hasError = True

if hasError == False:

cursorAddActorInDB = conn.cursor()

cursorAddActorInDB.execute("INSERT INTO Films VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", tuple(newFilmData))

conn.commit()

for i in newRoleData:

cursorAddActorRoleInDB = conn.cursor()

cursorAddActorRoleInDB.execute("INSERT INTO Role VALUES (?, ?, ?)", i)

conn.commit()

for i in addFilmActorRolesEntryList:

i.delete(0, END)

i["state"] = DISABLED

#addActorActorNameEntry.delete(0, END)

#addActorActorSurnameEntry.delete(0, END)

messagebox.showinfo("Успешно добавлен фильм!", "Успешно добавлен фильм %s! Количество актеров в этом фильме: %d!"%(newFilmData[1], len(newRoleData)))

'''функция showAllFilmsForEdit отображает на экране все фильмы для их редактирования или удаления

'''

def showAllFilmsForEdit():

global filmInfoDataList, actorsInFilmDataFramesList, actorDataLabelsList, dataList, fromFilm, fromAllFilms, editFilmAllFilmsWigetsList, allFilmsDataList

deleteAllFromThisWindow()

filmsMenuDescription.grid\_forget()

showFilmsButton.grid\_forget()

addFilmButton.grid\_forget()

editFilmButton.grid\_forget()

backToMainMenu.grid\_forget()

#если ранее был какой-то фильм

filmInformationDescription.grid\_forget()

filmInfoName.grid\_forget()

filmInfoYear.grid\_forget()

filmInfoDuration.grid\_forget()

filmInfoStyle.grid\_forget()

filmInfoAge.grid\_forget()

filmInfoProdactionCost.grid\_forget()

filmInfoUSAProfit.grid\_forget()

filmInfoWorldProfit.grid\_forget()

filmInfoAwards.grid\_forget()

backToFilmsFromFilm.grid\_forget()

oneFilmheader.grid\_forget()

oneFilmbody.grid\_forget()

oneFilmfooter.grid\_forget()

if len(filmInfoDataList) != 0:

for i in filmInfoDataList:

i.destroy()

for i in actorsInFilmDataFramesList:

i.grid\_forget()

actorsInFilmDataFramesList.clear()

for i in actorDataLabelsList:

i.grid\_forget()

actorDataLabelsList.clear()

actorsDataList.clear()

# fromFilm = False

# fromAllFilms = True

queryGetAllFilmsFromDB = conn.cursor()

queryGetAllFilmsFromDB.execute('select \* from Films')

allFilmsDataList = []

for note in queryGetAllFilmsFromDB:

allFilmsDataList.append(note)

print("Список всех фильмов: ", allFilmsDataList)

editFilmsHeader.grid(row = 0, sticky=N+S+W+E)

editFilmsBody.grid(row = 1, sticky=N+S+W+E)

editFilmsFooter.grid(row = 2, sticky=N+S+W+E)

# allFilmsHeader.grid(row = 0, sticky=N+S+W+E)

# allFilmsBody.grid(row = 1, sticky=N+S+W+E)

# allFilmsFooter.grid(row = 2, sticky=N+S+W+E)

editFilmsInformationLabel.grid(row = 0, column = 0, columnspan = 5, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmOptionsLabel.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmNameLabel.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmYearLabel.grid(row = 1, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmLengthLabel.grid(row = 1, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmStyleLabel.grid(row = 1, column = 4, sticky=N+S+W+E, padx = 5, pady = 5)

#allFilmsHeader

# allFilmsInformationLabel.grid(row = 0, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

# filmNameLabel.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

# filmYearLabel.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

# filmLengthLabel.grid(row = 1, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

# filmStyleLabel.grid(row = 1, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmAllFilmsWigetsList = []

#scrollableBodyEditFilms 🖉 ⌦

for i in range(len(allFilmsDataList)):

editFilmAllFilmsWigetsList.append([])

editFilmAllFilmsWigetsList[i].append(Button(scrollableBodyEditFilms, text="🖉", bg="#FFE4C4", font = ("Consolas", 18), width = 3))

editFilmAllFilmsWigetsList[i][0].grid(row = i, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmAllFilmsWigetsList[i][0].bind('<Button-1>', editFilmData)

editFilmAllFilmsWigetsList[i].append(Button(scrollableBodyEditFilms, text="⌦", bg="#FFE4C4", font = ("Consolas", 18), width = 3))

editFilmAllFilmsWigetsList[i][1].grid(row = i, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmAllFilmsWigetsList[i][1].bind('<Button-1>', deleteFilmInfoFromDB)

editFilmAllFilmsWigetsList[i].append(Label(scrollableBodyEditFilms, text="%s"%(allFilmsDataList[i][1]), bg="#FFE4C4", font = ("Consolas", 18), wraplength = 300, width = 25))

editFilmAllFilmsWigetsList[i][2].grid(row = i, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmAllFilmsWigetsList[i].append(Label(scrollableBodyEditFilms, text="%s"%(allFilmsDataList[i][2]), bg="#FFE4C4", font = ("Consolas", 18), width = 13))

editFilmAllFilmsWigetsList[i][3].grid(row = i, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmAllFilmsWigetsList[i].append(Label(scrollableBodyEditFilms, text="%s"%(allFilmsDataList[i][3]), bg="#FFE4C4", font = ("Consolas", 18), width = 13))

editFilmAllFilmsWigetsList[i][4].grid(row = i, column = 4, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmAllFilmsWigetsList[i].append(Label(scrollableBodyEditFilms, text="%s"%(allFilmsDataList[i][4]), bg="#FFE4C4", font = ("Consolas", 18), width = 22))

editFilmAllFilmsWigetsList[i][5].grid(row = i, column = 5, sticky=N+S+W+E, padx = 5, pady = 5)

#allFilmsFooter

footerEditFilms.grid(row = 0, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenuFromEditFilmsMenu.grid(row = 1, sticky=N+S+W+E, padx = 5, pady = 5)

#footerAllInformationMenu.grid(row = 0, sticky=N+S+W+E, padx = 5, pady = 5)

#backToMainMenuFromFilms.grid(row = 1, sticky=N+S+W+E, padx = 5, pady = 5)

scrollableBodyEditFilms.update()

'''функция deleteActorInfoFromDB удаляет всю информацию об актере из базы данных

'''

def deleteFilmInfoFromDB(event):

global allFilmsDataList

print("нажата кнопка в ряду: ", event.widget.grid\_info())

userChooseThisFilmRow = event.widget.grid\_info()["row"]

userChooseThisFilmID = allFilmsDataList[userChooseThisFilmRow][0]

userChooseThisFilmData = allFilmsDataList[userChooseThisFilmRow]

print("выбран фильма: ", userChooseThisFilmData, "его айди: ", userChooseThisFilmID)

#actorNewBirthDayDate = userChooseThisActorData[3]

userAnswer = messagebox.askyesno("Попытка удаления фильма из базы данных", "Вы выбрали опцию удаления из базы данных для фильма %s выпуска %d года! Вы действительно хотите удалить всю информацию об этом фильме из базы данных?"%(userChooseThisFilmData[1], userChooseThisFilmData[2]))

if userAnswer:

#удаление всех ролей для фильма

cursorDeleteAllActorRoleFromDB = conn.cursor()

cursorDeleteAllActorRoleFromDB.execute("{CALL deleteRoleOnFilmID(?)}", userChooseThisFilmID)

conn.commit()

#удаление данных о фильме

cursorDeleteActorInfoFromDB = conn.cursor()

cursorDeleteActorInfoFromDB.execute("{CALL deleteFilmOnFilmID(?)}", userChooseThisFilmID)

conn.commit()

messagebox.showinfo("Операция выполнена успешно!", "Фильм %s, вышедший в %d году, и все связанные с ним данные были удалены из базы данных!"%(userChooseThisFilmData[1], userChooseThisFilmData[2]))

deleteAllFromThisWindow()

showAllFilmsForEdit()

else:

messagebox.showinfo("Операция отменена!", "Фильм %s %d НЕ был удален из базы данных!"%(userChooseThisFilmData[1], userChooseThisFilmData[2]))

'''функция addFilmToDataBase выполняет переход в меню редактирования информации о фильме

'''

def editFilmData(event):

global checkBoxListEditFilmChooseActor, selectedActorsIndexesList, selectedActorsRowIndexesList, allActorsList, startRowForAllActorsRows

global labelsListWithFilmsForAddActorInDB, editFilmActorsRoleEntryList, allFilmsDataList, userChooseFilmID, userChooseFilmData

#print("нажата кнопка в ряду: ", event.widget.grid\_info())

rowIndex = event.widget.grid\_info()["row"]

print("Выбран фильм в строке: ", rowIndex)

userChooseFilmID = allFilmsDataList[rowIndex][0]

userChooseFilmData = allFilmsDataList[rowIndex]

print("выбран фильм: ", userChooseFilmData, "его айди: ", userChooseFilmID)

# userChooseThisActorRow = event.widget.grid\_info()["row"]

# userChooseThisActorID = editActorsDataAllActorsList[userChooseThisActorRow][0]

# userChooseThisActorData = editActorsDataAllActorsList[userChooseThisActorRow]

deleteAllFromThisWindow()

selectedActorsIndexesList = []

selectedActorsRowIndexesList = []

editFilmNoteHeader.grid(row = 0, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmNoteBody.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmNoteFooter.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

#header

editFilmDescriptionLabel.grid(row = 0, column = 0, columnspan = 4, sticky=N+S+W+E, padx = 5, pady = 5)

#body

editFilmFilmNameLabel.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmYearLabel.grid(row = 1, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmLengthLabel.grid(row = 2, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmStyleLabel.grid(row = 3, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmAgeLimitLabel.grid(row = 4, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmCostLabel.grid(row = 5, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmUSAProfitLabel.grid(row = 6, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmWorldProfitLabel.grid(row = 7, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmAwardsLabel.grid(row = 8, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

filmInfoDataFieldColumnspan = 4

editFilmFilmNameDataField.grid(row = 0, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)#userChooseFilmData

editFilmFilmNameDataField.insert(0, userChooseFilmData[1])

editFilmFilmYearDataField.grid(row = 1, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmYearDataField.insert(0, userChooseFilmData[2])

editFilmFilmLengthDataField.grid(row = 2, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmLengthDataField.insert(0, userChooseFilmData[3])

editFilmFilmStyleDataField.grid(row = 3, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmStyleDataField.insert(0, userChooseFilmData[4])

editFilmFilmAgeLimitDataField.grid(row = 4, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

values = [u"0", u"3", u"6", u"12", u"14", u"16", u"18"]

choosedFilmAgeLimit = userChooseFilmData[5]

for i in range(len(values)):

if choosedFilmAgeLimit == int(values[i]):

editFilmFilmAgeLimitDataField.current(i)

editFilmFilmCostDataField.grid(row = 5, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmCostDataField.insert(0, userChooseFilmData[6])

editFilmFilmUSAProfitDataField.grid(row = 6, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmFilmUSAProfitDataField.insert(0, userChooseFilmData[7])

editFilmFilmWorldProfitDataField.grid(row = 7, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

if userChooseFilmData[8] == '' or userChooseFilmData[8] == None:

editFilmFilmWorldProfitDataField.insert(0, 0)

else:

editFilmFilmWorldProfitDataField.insert(0, userChooseFilmData[8])

#userChooseFilmData[8]

#editFilmFilmWorldProfitDataField.insert(0, userChooseFilmData[8])

editFilmFilmAwardsDataField.grid(row = 8, column = 2, columnspan = filmInfoDataFieldColumnspan, sticky=N+S+W+E, padx = 5, pady = 5)

if userChooseFilmData[8] != '' and userChooseFilmData[9] != None:

editFilmFilmAwardsDataField.insert(1.0, userChooseFilmData[9])

editFilmChooseRolesDescription.grid(row = 9, column = 0, columnspan = filmInfoDataFieldColumnspan + 2, sticky=N+S+W+E, padx = 5, pady = 5)

#получение информации о всех имеющихся актерах

getAllActorsFromDataBase = """\

SELECT \* FROM getAllActors

"""

queryGetAllActorsIDFromDataBase = conn.cursor()

queryGetAllActorsIDFromDataBase.execute(getAllActorsFromDataBase)

allActorsList = []

for note in queryGetAllActorsIDFromDataBase:

allActorsList.append(note)

actorsCount = len(allActorsList)

print("\nСписок всех актеров\n", allActorsList)

editFilmChooseRoleCheckboxLabel.grid(row = 10, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmChooseRoleFilmNameLabel.grid(row = 10, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmChooseRoleFilmYearLabel.grid(row = 10, column = 2, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmChooseRoleFilmStyleLabel.grid(row = 10, column = 3, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmChooseRoleFilmAgeLimitLabel.grid(row = 10, column = 4, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmChooseRoleActorRoleLabel.grid(row = 10, column = 5, sticky=N+S+W+E, padx = 5, pady = 5)

startRowForAllActorsRows = 11

if len(editFilmActorsRoleEntryList) != 0:

for i in editFilmActorsRoleEntryList:

i.destroy()

editFilmActorsRoleEntryList.clear()

#Получение ролей для этого фильма

cursorAddActorInDB = conn.cursor()

cursorAddActorInDB.execute("{CALL getRolesForFilm(?)}", userChooseFilmID)

#"INSERT INTO Actors VALUES (?, ?, ?, ?, ?)")#updateActorInfo

#conn.commit()

choosedFilmRolesList = []

for note in cursorAddActorInDB:

choosedFilmRolesList.append(note)

print("список ролей для выбранного фильма: ", choosedFilmRolesList)

#получение всех ролей из базы данных

# getAllRolesDataFromDB = """\

# SELECT \* FROM Role

# """

# queryGetAllRolesDataFromDB = conn.cursor()

# queryGetAllRolesDataFromDB.execute(getAllRolesDataFromDB)

labelsListWithFilmsForAddActorInDB = []

for i in range(actorsCount):

labelsListWithFilmsForAddActorInDB.append([])

thisActorInThisFilm = False

for s in choosedFilmRolesList:

if allActorsList[i][0] == s[0]:

thisActorInThisFilm = True

thisActorRoleInThisFilm = s

for j in range(len(allActorsList[i]) + 1):

if j == 0:

checkBoxListEditFilmChooseActor.append(IntVar())

labelsListWithFilmsForAddActorInDB[i].append(Checkbutton(scrollableBodyEditChoosedFilm, text='', variable = checkBoxListEditFilmChooseActor[i], onvalue=1, offvalue=0, bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].bind('<Button-1>', chooseNewActorsListForFilm)#lambda event, param = event.widget.grid\_info()["row"]:

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i + startRowForAllActorsRows, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

if thisActorInThisFilm == True:

checkBoxListEditFilmChooseActor[i].set(1)

selectedActorsIndexesList.append(thisActorRoleInThisFilm[0])

selectedActorsRowIndexesList.append(i)

elif j == 3:

labelsListWithFilmsForAddActorInDB[i].append(Label(scrollableBodyEditChoosedFilm, text = "%s"%(allActorsList[i][j].strftime("%d.%m.%Y")), wraplength = 300, font = ("Consolas", 20), bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i + startRowForAllActorsRows, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

elif j == 5:

editFilmActorsRoleEntryList.append(Entry(scrollableBodyEditChoosedFilm, font = ("Consolas", 14), state = DISABLED, width = 45))

editFilmActorsRoleEntryList[i].grid(row = i + startRowForAllActorsRows, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

if thisActorInThisFilm == True:

editFilmActorsRoleEntryList[i]["state"] = NORMAL

editFilmActorsRoleEntryList[i].insert(0, thisActorRoleInThisFilm[2])

else:

labelsListWithFilmsForAddActorInDB[i].append(Label(scrollableBodyEditChoosedFilm, text = "%s"%(allActorsList[i][j]), wraplength = 300, font = ("Consolas", 20), bg="#FFE4C4"))

labelsListWithFilmsForAddActorInDB[i][j].grid(row = i + startRowForAllActorsRows, column = j, sticky=N+S+W+E, padx = 5, pady = 5)

#активация бесконечной прокрутки в области полей данных о фильме

scrollableBodyEditChoosedFilm.update()

#Footer

editFilmFinishDescription.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

returnBackTofilmsMenuFromEditFilm.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

editFilmNoteInDataBase.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

'''Функция chooseNewActorsListForFilm вызывается при нажатии чекбокса в окне добавления фильма

при выборе актера для этого фильма, сохраняет в список данные о выбранном актере

'''

def chooseNewActorsListForFilm(event):

global allActorsList, selectedActorsIndexesList, selectedActorsRowIndexesList, editFilmActorsRoleEntryList, startRowForAllActorsRows

print("chooseNewActorsListForFilm")

print(event.widget.grid\_info())

rowIndex = event.widget.grid\_info()["row"] - startRowForAllActorsRows

if checkBoxListEditFilmChooseActor[rowIndex].get() == 0:

editFilmActorsRoleEntryList[rowIndex]["state"] = NORMAL

selectedActorsIndexesList.append(allActorsList[rowIndex][0])

selectedActorsRowIndexesList.append(rowIndex)

print("выбран индекс: %d"%rowIndex)

print("индекс выбранных актеров:", selectedActorsIndexesList)

print("индекс выбранных строк:", selectedActorsRowIndexesList)

else:

selectedActorsIndexesList.remove(allActorsList[rowIndex][0])

selectedActorsRowIndexesList.remove(rowIndex)

editFilmActorsRoleEntryList[rowIndex].delete(0, END)

editFilmActorsRoleEntryList[rowIndex]["state"] = DISABLED

print("индекс выбранных актеров:", selectedActorsIndexesList)

print("индекс выбранных строк:", selectedActorsRowIndexesList)

'''Функция checkNewFilmDataAndUpdateItInDB проверяет введенные пользователем данные о фильме

и, в случае их корректности, обновляет данные в базе данных, иначе возвращает ошибку

'''

def checkNewFilmDataAndUpdateItInDB():

global selectedActorsIndexesList, selectedActorsRowIndexesList, editFilmActorsRoleEntryList, addFilmAgeLimitComboboxValue

global userChooseFilmID, userChooseFilmData

hasError = False

newFilmData = []

newRoleData = []

#получение индекса для добавляемого фильма

getAllFilmsFromDataBase = """\

SELECT \* FROM Films

"""

queryGetAllFilmsFromDataBase = conn.cursor()

queryGetAllFilmsFromDataBase.execute(getAllFilmsFromDataBase)

allFilmsList = []

for note in queryGetAllFilmsFromDataBase:

allFilmsList.append(note)

filmsCount = len(allFilmsList)

print(allFilmsList)

# maxFilmIndex = -5

# for i in range(filmsCount):

# if allFilmsList[i][0] > maxFilmIndex:

# maxFilmIndex = allFilmsList[i][0]

# addActorInDBFilmIndex = maxFilmIndex + 1

editFilmInDBFilmIndex = userChooseFilmID

newFilmData.append(editFilmInDBFilmIndex)

print("Фильму будет присвоен индекс: %d"%editFilmInDBFilmIndex)

print("данные редактируемого фильма: ", userChooseFilmData)

print("индексы выбранных актеров, которые учавствуют в фильме: ", selectedActorsIndexesList)

print("индексы выбранных строк с актерами", selectedActorsRowIndexesList)

ruUpperLetters = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

ruLowerLetters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

enUpperLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

enLowerLetters = "abcdefghijklmnopqrstuvwxyz"

specSymbolsForName = "-: "

#проверка названия фильма

filmName = editFilmFilmNameDataField.get()

print("новое название фильма: ", editFilmFilmNameDataField.get())

defisCount = 0

spaceCount = 0

columnCount = 0

if len(filmName) == 0:

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма не указано! Введите русское название фильма!\nВы можете использовать русские буквы, пробелы, дефисы и двоеточия.")

else:

if (filmName[0] in ['-', ':', ' '] or filmName[-1] in ['-', ':', ' ']):

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма не должно начинаться или оканчиваться на пробел или символы '-', ':'! Введите русское название фильма!\nВы можете использовать русские буквы, пробелы, дефисы и двоеточия.")

else:

for i in filmName:

if i not in ruUpperLetters and i not in ruLowerLetters and i not in specSymbolsForName:

hasError = True

#addFilmFilmNameDataField.delete(0, END)

messagebox.showerror("Ошибка ввода", "Название фильма введено неверно, введите русское название фильма!\nВы можете использовать русские буквы, пробелы, дефисы и двоеточия (не более 1 символа подряд).")

break

else:

if defisCount == 0 and i == "-":

defisCount +=1

elif defisCount == 1 and i != "-":

defisCount = 0

elif defisCount == 1 and i == "-":

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма введено неверно! Вы ввели слишком много дефисов подряд! Вы можете использовать русские буквы, пробелы, дефисы и двоеточия (не более 1 символа подряд).")

break

if spaceCount == 0 and i == " ":

spaceCount +=1

elif spaceCount == 1 and i != " ":

spaceCount = 0

elif spaceCount == 1 and i == " ":

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма введено неверно! Вы ввели слишком много пробелов подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать русские буквы, пробелы, дефисы и двоеточия (не более 1 символа подряд).")

break

if columnCount == 0 and i == ":":

columnCount+=1

elif columnCount == 1 and i != ":":

columnCount = 0

elif columnCount == 1 and i == ":":

hasError = True

messagebox.showerror("Ошибка ввода", "Название фильма введено неверно! Вы ввели слишком много символов ':' подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать русские буквы, пробелы, дефисы и двоеточия (не более 1 символа подряд).")

break

newFilmData.append(filmName)

# for i in range(filmsCount):

# if allFilmsList[i][1] == filmName:

# hasError = True

# messagebox.showerror("Фильм с таким названием уже есть в ПС", "Фильм с таким названием уже есть в базе данных! Занесите в базу данных информацию о другом фильме!")

# break

#проверка года выхода фильма

filmYear = editFilmFilmYearDataField.get()

if hasError == False:

if len(filmYear) != 0:

try:

filmYear = int(filmYear)

except ValueError:

hasError = True

#addFilmFilmYearDataField.delete(0, END)

messagebox.showerror("Ошибка ввода", "Год выпуска фильма введен неверно! Укажите год выхода фильма числом. Вы можете указать год выхода с 1880 года по текущий в момент работы программы.")

return

today = datetime.date.today()

minYear = 1880

maxYear = today.strftime("%Y")

if not (filmYear >= minYear and filmYear <= int(maxYear)):

hasError = True

#addFilmFilmYearDataField.delete(0, END)

messagebox.showerror("Ошибка ввода", "Год выхода фильма указан неверно, вы можете указать год выхода с 1880 по текущий в момент работы программы.")

else:

hasError = True

messagebox.showerror("Ошибка ввода", "Вы не указали год выхода фильма! Вы можете указать год выхода с 1880 года по текущий в момент работы программы.")

newFilmData.append(filmYear)

#проверка продолжительности фильма

filmLength = editFilmFilmLengthDataField.get()

if hasError == False:

if len(filmLength) != 0:

try:

filmLength = int(filmLength)

except ValueError:

hasError = True

#filmLength.delete(0, END)

messagebox.showerror("Ошибка ввода", "Длительность фильма указана неверно! Вы должны указать длительность фильма в минутах от 1 до 100000.")

return

if not (filmLength >= 1 and filmLength <= 100000):

hasError = True

#addFilmFilmYearDataField.delete(0, END)

messagebox.showerror("Ошибка ввода", "Вы неправильно указали длительность фильма! Вы должны указать длительность фильма в минутах от 1 до 100000.")

else:

hasError = True

messagebox.showerror("Ошибка ввода", "Вы не указали длительность фильма в минутах! Вы должны указать длительность фильма в минутах от 1 до 100000.")

newFilmData.append(filmLength)

#проверка жанра фильма

filmStyle = editFilmFilmStyleDataField.get()

if hasError == False:

if len(filmStyle) == 0:

hasError = True

messagebox.showerror("Ошибка ввода", "Жанр фильма не указан! Введите жанр фильма строчными русскими буквами, вы можете использовать дефис и пробел.")

else:

defisCount = 0

spaceCount = 0

for i in filmStyle:

if (i not in ruLowerLetters) and i != "-" and i != " ":

hasError = True

messagebox.showerror("Ошибка ввода", "Жанр фильма указан неверно! Введите жанр фильма строчными русскими буквами, вы можете использовать дефис и пробел.")

break

else:

if defisCount == 0 and i == "-":

defisCount +=1

elif defisCount == 1 and i != "-":

defisCount = 0

elif defisCount == 1 and i == "-":

hasError = True

messagebox.showerror("Ошибка ввода", "Жанр фильма указан неверно! Вы ввели слишком много дефисов подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать дефис и пробел.")

break

if spaceCount == 0 and i == " ":

spaceCount +=1

elif spaceCount == 1 and i != " ":

spaceCount = 0

elif spaceCount == 1 and i == " ":

hasError = True

messagebox.showerror("Ошибка ввода", "Жанр фильма указан неверно! Вы ввели слишком много пробелов подряд! Введите жанр фильма строчными русскими буквами, вы можете использовать дефис и пробел.")

break

newFilmData.append(str(filmStyle))

#проверка возрастного ограничения фильма

#filmAge = addFilmFilmAgeLimitDataField.get()

filmAge = editFilmFilmAgeLimitDataField.get()

if hasError == False:

if filmAge not in [u"0", u"3", u"6", u"12", u"14", u"16", u"18"]:

hasError = True

messagebox.showerror("Ошибка ввода", "Возрастное ограничение фильма указано неверно! Выберите возрастное ограничение строго из списка значений.")

newFilmData.append(int(filmAge))

#проверка бюджета фильма

filmCost = editFilmFilmCostDataField.get()

if hasError == False:

if len(filmCost) != 0:

try:

filmCost = int(filmCost)

except ValueError:

hasError = True

messagebox.showerror("Ошибка ввода", "Бюджет фильма указан неверно! Вы должны указать бюджет фильма в долларах от 1 доллара до 10 миллиардов долларов.")

return

if not (filmCost >= 1 and filmCost <= 10000000000):

hasError = True

messagebox.showerror("Ошибка ввода", "Вы неправильно указали бюджет фильма! Вы должны указать бюджет фильма в долларах от 1 доллара до 10 миллиардов долларов.")

else:

hasError = True

messagebox.showerror("Ошибка ввода", "Вы не указали бюджет фильма в долларах! Вы должны указать бюджет фильма в долларах от 1 доллара до 10 миллиардов долларов.")

newFilmData.append(filmCost)

#проверка дохода в США

filmUSAProfit = editFilmFilmUSAProfitDataField.get()

if hasError == False:

if len(filmUSAProfit) != 0:

try:

filmUSAProfit = int(filmUSAProfit)

except ValueError:

hasError = True

messagebox.showerror("Ошибка ввода", "Сборы фильма в США указаны неверно! Вы должны указать сборы фильма в США в долларах от 0 долларов до 10 миллиардов долларов.")

return

if not (filmUSAProfit >= 0 and filmUSAProfit <= 10000000000):

hasError = True

messagebox.showerror("Ошибка ввода", "Вы неправильно указали сборы фильма в США! Вы должны указать сборы фильма в США в долларах от 0 долларов до 10 миллиардов долларов.")

else:

hasError = True

messagebox.showerror("Ошибка ввода", "Вы не указали сборы фильма в США в долларах! Вы должны указать сборы фильма в США в долларах от 0 долларов до 10 миллиардов долларов.")

newFilmData.append(filmUSAProfit)

#проверка дохода в мире

filmWorldProfit = editFilmFilmWorldProfitDataField.get()

if hasError == False:

if len(filmWorldProfit) != 0:

try:

filmWorldProfit = int(filmWorldProfit)

except ValueError:

hasError = True

messagebox.showerror("Ошибка ввода", "Сборы фильма в мире указаны неверно! Вы должны указать сборы фильма в мире в долларах от 0 долларов до 10 миллиардов долларов, либо оставить поле пустым")

return

if not (filmWorldProfit >= 0 and filmWorldProfit <= 10000000000):

hasError = True

messagebox.showerror("Ошибка ввода", "Вы неправильно указали сборы фильма в мире! Вы должны указать сборы фильма в мире в долларах от 0 долларов до 10 миллиардов долларов, либо оставить поле пустым")

else:

filmWorldProfit = 0

newFilmData.append(filmWorldProfit)

#проверка наград

filmAwards = editFilmFilmAwardsDataField.get(1.0, END)

print("filmAwards = ", filmAwards)

forbiddenCharacters = []

if hasError == False:

if len(filmAwards) != 0:

for i in filmAwards:

if (i not in ruUpperLetters) and (i not in ruLowerLetters) and (i not in enUpperLetters) and (i not in enLowerLetters) and (i not in ['(', ')', ':', '-', '.', ',', '\n', ' ']) and (i.isdigit() == False):

hasError = True

if i not in forbiddenCharacters:

forbiddenCharacters.append(i)

if hasError == True:

messagebox.showerror("Ошибка ввода", "Вы ввели эти недопустимые символы при описании наград фильма: {}! Вы должны указать награды фильма, используя английские и русские буквы, цифры, символы: пробел, '(', ')', ':', '-', '.', ',', либо оставить поле пустым".format(forbiddenCharacters))

if filmAwards == "\n":

filmAwards = ""

newFilmData.append(str(filmAwards))

# print("Фильму будет присвоен индекс: %d"%editFilmInDBFilmIndex)

# print("данные редактируемого фильма: ", userChooseFilmData)

# print("индексы выбранных актеров, которые участвуют в фильме: ", selectedActorsIndexesList)

# print("индексы выбранных строк с актерами", selectedActorsRowIndexesList)

newRoleData = []

#формирование данных для таблицы ролей

if hasError == False:

forbiddenCharacters = []

for i in range(len(selectedActorsIndexesList)):

actorCharacter = editFilmActorsRoleEntryList[selectedActorsRowIndexesList[i]].get()

if len(actorCharacter) != 0:

for s in actorCharacter:

if (s not in ruUpperLetters) and (s not in ruLowerLetters) and (s not in enUpperLetters) and (s not in enLowerLetters) and (s not in ['(', ')', ':', '-', '.', ',', '\n', ' ', '"']) and (s.isdigit() == False):

hasError = True

if s not in forbiddenCharacters:

forbiddenCharacters.append(s)

print("Актер с индексом: ", selectedActorsIndexesList[i], "фильм с индексом: ", editFilmInDBFilmIndex, "роль: ", actorCharacter, "строка номер: ", selectedActorsRowIndexesList[i])

newRoleData.append(tuple([int(selectedActorsIndexesList[i]), int(editFilmInDBFilmIndex), str(actorCharacter)]))

if hasError == True:

messagebox.showerror("Ошибка ввода", "Вы ввели эти недопустимые символы при описании ролей актеров в фильме: {}! Вы можете указать роли актеров в фильме, используя английские и русские буквы, цифры, символы: пробел, '(', ')', ':', '-', '.', ',', либо оставить эти поля пустыми".format(forbiddenCharacters))

#редактирование фильма в БД

print("В таблице 'Films' БД будет обновлена запись: \n", newFilmData)

print("В таблице 'Role' БД текущему фильмы будут привязаны роли: \n", newRoleData)

#hasError = True

if hasError == False:

#обновление записи в таблице фильмов

cursorEditFilmInDB = conn.cursor()

cursorEditFilmInDB.execute("{CALL updateFilmInfo(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}", newFilmData[1], newFilmData[2], newFilmData[3], newFilmData[4], newFilmData[5],newFilmData[6],newFilmData[7],newFilmData[8],newFilmData[9],newFilmData[0])

#"INSERT INTO Actors VALUES (?, ?, ?, ?, ?)")#updateActorInfo

conn.commit()

# getAllRolesDataFromDB = """\

# SELECT \* FROM Role

# """

# queryGetAllRolesDataFromDB = conn.cursor()

# queryGetAllRolesDataFromDB.execute(getAllRolesDataFromDB)

#allRolesInDBList = []

#oldRolesForNowEditedActor = []

#удаление всех ролей для фильма, который сейчас редактируется

cursorDeleteAllActorsRoleFromDB = conn.cursor()

cursorDeleteAllActorsRoleFromDB.execute("{CALL deleteRoleOnFilmID(?)}", newFilmData[0])

conn.commit()

# for note in queryGetAllRolesDataFromDB:

# allRolesInDBList.append(note)

# if note[0] == newNoteForActors[0]:#deleteRoleOnActorID

# rolesCount = len(allRolesInDBList)

# print("all roles: ", allRolesInDBList)

#занесение записей о ролях в БД

# for i in newRoleData:

# cursorAddActorRoleInDB = conn.cursor()

# cursorAddActorRoleInDB.execute("INSERT INTO Role VALUES (?, ?, ?)", i)

# conn.commit()

# for i in editFilmActorsRoleEntryList:

# i.delete(0, END)

# i["state"] = DISABLED

# # editActorActorNameEntry.delete(0, END)

# # editActorActorSurnameEntry.delete(0, END)

# messagebox.showinfo("Успешно отредактированы данные актера!", "Успешно отредактированы данные актера %s %s, этот актер участвует в %d фильмах."%(newNoteForActors[1], newNoteForActors[2], len(addActorInDBActorRolesDataList)))

# cursorAddActorInDB = conn.cursor()

# cursorAddActorInDB.execute("INSERT INTO Films VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", tuple(newFilmData))

# conn.commit()

for i in newRoleData:

cursorAddActorRoleInDB = conn.cursor()

cursorAddActorRoleInDB.execute("INSERT INTO Role VALUES (?, ?, ?)", i)

conn.commit()

for i in editFilmActorsRoleEntryList:

i.delete(0, END)

i["state"] = DISABLED

#addActorActorNameEntry.delete(0, END)

#addActorActorSurnameEntry.delete(0, END)

editFilmFilmNameDataField.delete(0, END)

editFilmFilmYearDataField.delete(0, END)

editFilmFilmLengthDataField.delete(0, END)

editFilmFilmStyleDataField.delete(0, END)

editFilmFilmCostDataField.delete(0, END)

editFilmFilmUSAProfitDataField.delete(0, END)

editFilmFilmWorldProfitDataField.delete(0, END)

editFilmFilmAwardsDataField.delete(1.0, END)

messagebox.showinfo("Успешно обновлены данные фильма!", "Успешно обновлены данные фильма %s! Количество актеров в этом фильме: %d!"%(newFilmData[1], len(newRoleData)))

showAllFilmsForEdit()

#запросы

def findActor():

print("Вы выбрали поиск актера")

deleteAllFromThisWindow()

findActorDescriptionLabel.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

findActorNameLabel.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

findActorSurnameLabel.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

findActorNameEntry.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

findActorSurnameEntry.grid(row = 2, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

startFindActorButton.grid(row = 3, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

def findAndShowActor():

global iFindActor, iFoundThisActorData

actorName = findActorNameEntry.get()

actorSurname = findActorSurnameEntry.get()

iFindActor = True

getActorsOnNS = """\

{CALL findActorOnNS(?, ?)}

"""

queryGetActors = conn.cursor()

queryGetActors.execute(getActorsOnNS, actorName, actorSurname)

iFoundThisActorData = []

for note in queryGetActors:

iFoundThisActorData.append(note)

print("По вашему запросу найдено: ", iFoundThisActorData)

if len(iFoundThisActorData) > 0:

messagebox.showinfo("Успешно", "По запросу %s %s успешно найдена информация!"%(actorName, actorSurname))

findActorNameEntry.delete(0, END)

findActorSurnameEntry.delete(0, END)

showAllActors()

else:

iFindActor = False

messagebox.showerror("Неудача", "По вашему запросу %s %s не найдено актеров!"%(actorName, actorSurname))

#findActorOnNS

#Поиск фильма по названию

def findFilmOnName():

deleteAllFromThisWindow()

findFilmDescriptionLabel.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmNameLabel.grid(row = 1, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmNameEntry.grid(row = 2, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

startFindFilmNameButton.grid(row = 3, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

def startFindFilmOnName():

global iFindFilmOnName, iFoundThisFilmData

findFilmName = findFilmNameEntry.get()

iFindFilmOnName = True

getActorsOnNS = """\

{CALL findFilmOnName(?)}

"""

queryGetActors = conn.cursor()

queryGetActors.execute(getActorsOnNS, findFilmName)

iFoundThisFilmData = []

for note in queryGetActors:

iFoundThisFilmData.append(note)

print("По вашему запросу найдено: ", iFoundThisFilmData)

if len(iFoundThisFilmData) > 0:

messagebox.showinfo("Успешно", "По запросу %s успешно найдена информация!"%(findFilmName))

findFilmNameEntry.delete(0, END)

#findActorSurnameEntry.delete(0, END)

showAllFilms()

else:

iFindFilmOnName = False

messagebox.showerror("Неудача", "По вашему запросу %s не найдено фильмов!"%(findFilmName))

#Поиск фильма по жанру

def findFilmOnStyle():

deleteAllFromThisWindow()

findFilmStyleDescriptionLabel.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmStyleLabel.grid(row = 1, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmStyleEntry.grid(row = 2, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

startFindFilmStyleButton.grid(row = 3, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

def startFindFilmOnStyle():

global iFindFilmOnStyle, iFoundThisFilmDataStyle

findFilmStyle = findFilmStyleEntry.get()

iFindFilmOnStyle = True

getActorsOnNS = """\

{CALL findFilmOnStyle(?)}

"""

queryGetActors = conn.cursor()

queryGetActors.execute(getActorsOnNS, findFilmStyle)

iFoundThisFilmDataStyle = []

for note in queryGetActors:

iFoundThisFilmDataStyle.append(note)

print("По вашему запросу найдено: ", iFoundThisFilmDataStyle)

if len(iFoundThisFilmDataStyle) > 0:

messagebox.showinfo("Успешно", "По запросу %s успешно найдена информация!"%(findFilmStyle))

findFilmStyleEntry.delete(0, END)

#findActorSurnameEntry.delete(0, END)

showAllFilms()

else:

iFindFilmOnStyle = False

messagebox.showerror("Неудача", "По вашему запросу %s не найдено фильмов!"%(findFilmStyle))

#Поиск фильма по году

def findFilmOnYear():

deleteAllFromThisWindow()

findFilmYearDescriptionLabel.grid(row = 0, column = 0, columnspan = 2, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmYearLabel.grid(row = 1, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmYearEntry.grid(row = 2, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmYearRadioLabel.grid(row = 1, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

findFilmYearCheckbuCheckbutton.grid(row = 2, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

backToMainMenu.grid(row = 3, column = 0, sticky=N+S+W+E, padx = 5, pady = 5)

startFindFilmYearButton.grid(row = 3, column = 1, sticky=N+S+W+E, padx = 5, pady = 5)

def startFindFilmOnYear():

global iFindFilmOnYearBefore, iFindFilmOnYearAfter, iFoundThisFilmDataYear, filmFindYearCheckbuCheckbuttonVar

findFilmYear = findFilmYearEntry.get()

itHasError = False

try:

findFilmYear = int(findFilmYear)

except ValueError:

messagebox.showerror("Неудача", "Вы ввели не число! Повторите ввод!")

itHasError = True

return

if itHasError == False:

if filmFindYearCheckbuCheckbuttonVar.get() == 0:

iFindFilmOnYearBefore = True

iFindFilmOnYearAfter = False

else:

iFindFilmOnYearBefore = False

iFindFilmOnYearAfter = True

if iFindFilmOnYearBefore == True:

getActorsOnNS = """\

{CALL findFilmOnYearBefore(?)}

"""

queryGetActors = conn.cursor()

queryGetActors.execute(getActorsOnNS, findFilmYear)

iFoundThisFilmDataYear = []

for note in queryGetActors:

iFoundThisFilmDataYear.append(note)

else:

getActorsOnNS = """\

{CALL findFilmOnYearAfter(?)}

"""

queryGetActors = conn.cursor()

queryGetActors.execute(getActorsOnNS, findFilmYear)

iFoundThisFilmDataYear = []

for note in queryGetActors:

iFoundThisFilmDataYear.append(note)

print("По вашему запросу найдено: ", iFoundThisFilmDataYear)

if len(iFoundThisFilmDataYear) > 0:

messagebox.showinfo("Успешно", "По запросу %s год успешно найдена информация!"%(findFilmYear))

findFilmYearEntry.delete(0, END)

filmFindYearCheckbuCheckbuttonVar.set(0)

#findActorSurnameEntry.delete(0, END)

showAllFilms()

else:

iFindFilmOnYearBefore = False

messagebox.showerror("Неудача", "По вашему запросу %s год не найдено фильмов!"%(findFilmYear))

'''Функция deleteAllFromThisWindow предназначена для очистки окна программы при переходе из одного меню в другое

'''

def deleteAllFromThisWindow():

global labelsListWithFilmsForAddActorInDB, checkBoxIntVarListForAddActorInDB, checkBoxListEditFilmChooseActor, addFilmActorRolesEntryList, allActorsEditActorDataWidgetsList

global editFilmAllFilmsWigetsList, checkBoxIntVarListForAddFilmInDB, editFilmActorsRoleEntryList, iFindActor, iFindFilmOnName, iFindFilmOnStyle

global iFindFilmOnYearBefore, iFindFilmOnYearAfter, iFoundThisFilmDataYear

#убираем MainMenu

programmDescription.grid\_forget()

creatorInfoButton.grid\_forget()

filmInformationButton.grid\_forget()

actorInformationButton.grid\_forget()

administratorLoginButton.grid\_forget()

exitButton.grid\_forget()

#скрыть главное меню

footerAllInformationMenu.grid\_forget()

editActorButton.grid\_forget()

findActorButton.grid\_forget()

showActorsButton.grid\_forget()

addActorButton.grid\_forget()

backToMainMenu.grid\_forget()

returnBackToActorsMenu.grid\_forget()

#меню разработчика программы

creatorInfoLabel.grid\_forget()

backToMainMenu.grid\_forget()

#меню фильмов

filmsMenuDescription.grid\_forget()

addFilmButton.grid\_forget()

editFilmButton.grid\_forget()

showFilmsButton.grid\_forget()

findFilmsNameButton.grid\_forget()

findFilmsStyleButton.grid\_forget()

findFilmsYearButton.grid\_forget()

#поиск фильма по названию

findFilmDescriptionLabel.grid\_forget()

findFilmNameLabel.grid\_forget()

findFilmNameEntry.grid\_forget()

startFindFilmNameButton.grid\_forget()

iFindFilmOnName = False

findFilmNameEntry.delete(0, END)

#по жанру

findFilmStyleDescriptionLabel.grid\_forget()

findFilmStyleLabel.grid\_forget()

findFilmStyleEntry.grid\_forget()

startFindFilmStyleButton.grid\_forget()

iFindFilmOnStyle = False

findFilmStyleEntry.delete(0, END)

#по году

iFindFilmOnYearBefore = False

iFindFilmOnYearAfter = False

findFilmYearDescriptionLabel.grid\_forget()

findFilmYearLabel.grid\_forget()

findFilmYearEntry.grid\_forget()

findFilmYearRadioLabel.grid\_forget()

findFilmYearCheckbuCheckbutton.grid\_forget()

startFindFilmYearButton.grid\_forget()

#меню актеров

iFindActor = False

actorsMenuDescription.grid\_forget()

addActorButton.grid\_forget()

editActorButton.grid\_forget()

showActorsButton.grid\_forget()

backToMainMenu.grid\_forget()

findActorDescriptionLabel.grid\_forget()

findActorNameLabel.grid\_forget()

findActorSurnameLabel.grid\_forget()

findActorNameEntry.grid\_forget()

findActorSurnameEntry.grid\_forget()

startFindActorButton.grid\_forget()

#меню авторизации

loginDescriptionLabel.grid\_forget()

passwordEntry.grid\_forget()

checkPasswordButton.grid\_forget()

backToMainMenu.grid\_forget()

loginDescriptionLabel.grid\_forget()

editPasswordLabel.grid\_forget()

editPasswordEntry.grid\_forget()

editPasswordButton.grid\_forget()

youAreAdminLabel.grid\_forget()

toBeUser.grid\_forget()

#меню с бесконечной прокруткой

allFilmsHeader.grid\_forget()

allFilmsBody.grid\_forget()

allFilmsFooter.grid\_forget()

backToMainMenuFromFilms.grid\_forget()

#окно с добавлением информации об актере

actorAddNoteHeader.grid\_forget()

actorAddNoteBody.grid\_forget()

actorAddNoteFooter.grid\_forget()

addActorDescriptionLabel.grid\_forget()

returnBackToActorsMenu.grid\_forget()

addActorNoteInDataBase.grid\_forget()

addActorChooseRoleCheckboxLabel.grid\_forget()

addActorChooseRoleFilmNameLabel.grid\_forget()

addActorChooseRoleFilmYearLabel.grid\_forget()

addActorChooseRoleFilmStyleLabel.grid\_forget()

addActorChooseRoleFilmAgeLimitLabel.grid\_forget()

addActorChooseRoleActorRoleLabel.grid\_forget()

for i in range(len(labelsListWithFilmsForAddActorInDB)):

for j in range(len(labelsListWithFilmsForAddActorInDB[i])):

labelsListWithFilmsForAddActorInDB[i][j].destroy()

labelsListWithFilmsForAddActorInDB.clear()

checkBoxIntVarListForAddActorInDB.clear()

checkBoxListEditFilmChooseActor.clear()

checkBoxIntVarListForAddFilmInDB.clear()

addActorActorNameLabel.grid\_forget()

addActorActorSurnameLabel.grid\_forget()

addActorActorBirthDayLabel.grid\_forget()

addActorActorGenderLabel.grid\_forget()

addActorChooseRolesDescription.grid\_forget()

addActorFinishDescription.grid\_forget()

#убираем меню со всеми фильмами

allActorsheader.grid\_forget()

allActorsbody.grid\_forget()

allActorsfooter.grid\_forget()

#окно с добавлением информации о фильме

addFilmFilmNameLabel.grid\_forget()

addFilmFilmYearLabel.grid\_forget()

addFilmFilmLengthLabel.grid\_forget()

addFilmFilmStyleLabel.grid\_forget()

addFilmFilmAgeLimitLabel.grid\_forget()

addFilmFilmCostLabel.grid\_forget()

addFilmFilmUSAProfitLabel.grid\_forget()

addFilmFilmWorldProfitLabel.grid\_forget()

addFilmFilmAwardsLabel.grid\_forget()

filmAddNoteHeader.grid\_forget()

filmAddNoteBody.grid\_forget()

filmAddNoteFooter.grid\_forget()

addFilmChooseRolesDescription.grid\_forget()

addFilmFinishDescription.grid\_forget()

addFilmNoteInDataBase.grid\_forget()

returnBackTofilmsMenu.grid\_forget()

addFilmFilmNameDataField.grid\_forget()

addFilmFilmYearDataField.grid\_forget()

addFilmFilmLengthDataField.grid\_forget()

addFilmFilmStyleDataField.grid\_forget()

addFilmFilmAgeLimitDataField.grid\_forget()

addFilmFilmCostDataField.grid\_forget()

addFilmFilmUSAProfitDataField.grid\_forget()

addFilmFilmWorldProfitDataField.grid\_forget()

addFilmFilmAwardsDataField.grid\_forget()

addFilmChooseRoleCheckboxLabel.grid\_forget()

addFilmChooseRoleFilmNameLabel.grid\_forget()

addFilmChooseRoleFilmYearLabel.grid\_forget()

addFilmChooseRoleFilmStyleLabel.grid\_forget()

addFilmChooseRoleFilmAgeLimitLabel.grid\_forget()

addFilmChooseRoleActorRoleLabel.grid\_forget()

#если было открыто меню с выбором актера для редактирования

editActorsheader.grid\_forget()

editActorsbody.grid\_forget()

editActorsfooter.grid\_forget()

if len(allActorsEditActorDataWidgetsList) != 0:

for wigetRow in allActorsEditActorDataWidgetsList:

for wiget in wigetRow:

wiget.destroy()

allActorsEditActorDataWidgetsList.clear()

editActorNewActorBirthDayEntry.grid\_forget()

#если редактировался актер

editActorActorNameEntry.delete(0, END)

editActorActorSurnameEntry.delete(0, END)

editorAddNoteHeader.grid\_forget()

editorAddNoteBody.grid\_forget()

editorAddNoteFooter.grid\_forget()

# editActorDescriptionLabel

# editActorActorNameLabel

# editActorActorSurnameLabel

# editActorActorBirthDayLabel

# editActorActorGenderLabel

# editActorActorNameEntry

# editActorActorSurnameEntry

# editActorNewActorBirthDayEntry

# editActorActorGenderEntry

# editActorNewActorBirthDayEntry

# editActorChooseRolesDescription

# editActorChooseRoleCheckboxLabel

# editActorChooseRoleFilmNameLabel

# editActorChooseRoleFilmYearLabel

# editActorChooseRoleFilmStyleLabel

# editActorChooseRoleFilmAgeLimitLabel

# editActorChooseRoleActorRoleLabel

#список всех фильмов для редактирования

if len(editFilmAllFilmsWigetsList) != 0:

for wigetRow in editFilmAllFilmsWigetsList:

for wiget in wigetRow:

wiget.destroy()

editFilmAllFilmsWigetsList.clear()

editActorNewActorBirthDayEntry.grid\_forget()

editFilmsHeader.grid\_forget()

editFilmsBody.grid\_forget()

editFilmsFooter.grid\_forget()

#редактирование информации о фильме

editFilmFilmNameDataField.delete(0, END)

editFilmFilmYearDataField.delete(0, END)

editFilmFilmLengthDataField.delete(0, END)

editFilmFilmStyleDataField.delete(0, END)

editFilmFilmCostDataField.delete(0, END)

editFilmFilmUSAProfitDataField.delete(0, END)

editFilmFilmWorldProfitDataField.delete(0, END)

editFilmFilmAwardsDataField.delete(1.0, END)

editFilmNoteHeader.grid\_forget()

editFilmNoteBody.grid\_forget()

editFilmNoteFooter.grid\_forget()

#Подключаем БД к python

dataFile = "CourseWorkDataBase.accdb"

databaseFilePath = os.getcwd() + "\\" + dataFile

connectionString = "Driver={Microsoft Access Driver (\*.mdb, \*.accdb)};Dbq=%s;PWD=123456" % databaseFilePath

conn = pyodbc.connect(connectionString)

#row = cursor.fetchone()

#rows = cursor.fetchall()

# for row in rows:

# print(row[0])

dataList = []

root=Tk()

root.geometry('+50+100')

root.resizable(width=False, height=False)

root.title('Курсовая работа ПС "Киностудия"')

#главное меню

programmDescription = Label(root, bg ="lightgreen", text='Добро пожаловать в Программную систему «Киностудия»!\nВ этой информационной системе вы можете ознакомиться\nс фильмами студии Warner Bros.\nВыберите в меню интересующее вас действие', font = ("Consolas", 20))

creatorInfoButton = Button(root, text='Информация о разработчике ПС "Киностудия"', font = ("Consolas", 20), bg="#FFE4C4", command = showCreatorInfo)

filmInformationButton= Button(root, text='Информация о фильмах', font = ("Consolas", 20), bg="#FFE4C4", command = showFilmsMenu)

actorInformationButton = Button(root, text='Информация об актерах', font = ("Consolas", 20), bg="#FFE4C4", command = showActorsMenu)

administratorLoginButton= Button(root, text='Вход в режим администратора', font = ("Consolas", 20), bg="#FFE4C4", command = showLoginMenu)

exitButton = Button(root, text='Выход', font = ("Consolas", 20), bg="#FFE4C4", command = closeProgram)

#меню Информация о разработчике

creatorInfoLabel = Label(root, text='Программную систему «Киностудия»!\nразработал студент 3-го курса\nгруппы 18-ВТ\nКалистый Никита', bg ="lightgreen", font = ("Consolas", 20))

backToMainMenu = Button(root, text='Вернуться в главное меню', font = ("Consolas", 20), bg="#FFE4C4", command = showMainMenu)

#меню Информация о фильмах

filmsMenuDescription = Label(root, text='Вы перешли в раздел просмотра информации о фильмах\nВыберите в меню интересующее вас действие', bg ="lightgreen", font = ("Consolas", 20))

addFilmButton = Button(root, text='Добавить фильм в ПС', font = ("Consolas", 20), bg="#FFE4C4", command = addFilmToDataBase)

editFilmButton = Button(root, text='Редактирование информации о фильмах', font = ("Consolas", 20), bg="#FFE4C4", command = showAllFilmsForEdit)

showFilmsButton = Button(root, text='Просмотр информации о фильмах', font = ("Consolas", 20), bg="#FFE4C4", command = showAllFilms)

findFilmsNameButton = Button(root, text='Поиск фильма по названию', font = ("Consolas", 20), bg="#FFE4C4", command = findFilmOnName)

findFilmsStyleButton = Button(root, text='Поиск фильма по жанру', font = ("Consolas", 20), bg="#FFE4C4", command = findFilmOnStyle)

findFilmsYearButton = Button(root, text='Поиск фильма до/после года', font = ("Consolas", 20), bg="#FFE4C4", command = findFilmOnYear)

#поиск по названию

findFilmDescriptionLabel = Label(root, text='Вы перешли в раздел поиска информации о фильме по названию\nВведите название фильма и нажмите\nкнопку "Найти фильм" для поиска фильма', bg ="lightgreen", font = ("Consolas", 20))

findFilmNameLabel = Label(root, text='Введите название фильма', bg ="lightgreen", width = 40, font = ("Consolas", 20))

#findFilmSurnameLabel = Label(root, text='Фамилия', bg ="lightgreen", width = 40, font = ("Consolas", 20))

findFilmNameEntry = Entry(root, font = ("Consolas", 20))

#findFilmSurnameEntry = Entry(root, font = ("Consolas", 20))

startFindFilmNameButton = Button(root, text='Найти фильм по названию', font = ("Consolas", 20), bg="#FFE4C4", command = startFindFilmOnName)

#поиск по жанру

findFilmStyleDescriptionLabel = Label(root, text='Вы перешли в раздел поиска информации о фильме по жанру\nВведите жанр фильма и нажмите\nкнопку "Найти фильм" для поиска фильма', bg ="lightgreen", font = ("Consolas", 20))

findFilmStyleLabel = Label(root, text='Введите жанр фильма', bg ="lightgreen", width = 40, font = ("Consolas", 20))

findFilmStyleEntry = Entry(root, font = ("Consolas", 20))

startFindFilmStyleButton = Button(root, text='Найти фильм по жанру', font = ("Consolas", 20), bg="#FFE4C4", command = startFindFilmOnStyle)

#поиск по году

findFilmYearDescriptionLabel = Label(root, text='Вы перешли в раздел поиска информации о фильме по году\nВведите год выхода фильма, выберите поиск до/после введенного года,\nзатем нажмите кнопку "Найти фильм" для поиска фильма', bg ="lightgreen", font = ("Consolas", 20))

findFilmYearLabel = Label(root, text='Введите год выхода фильма', bg ="lightgreen", width = 40, font = ("Consolas", 20))

findFilmYearEntry = Entry(root, font = ("Consolas", 20))

findFilmYearRadioLabel = Label(root, text='Выберите поиск до/после введенного года', bg ="lightgreen", width = 40, font = ("Consolas", 20))

filmFindYearCheckbuCheckbuttonVar = IntVar()

filmFindYearCheckbuCheckbuttonVar.set(0)

findFilmYearCheckbuCheckbutton = Checkbutton(text="Включите, чтобы проводить поиск\nпосле введенного года",font = ("Consolas", 20), onvalue=1, offvalue = 0, variable=filmFindYearCheckbuCheckbuttonVar, padx=15, pady=10)

startFindFilmYearButton = Button(root, text='Найти фильм по году', font = ("Consolas", 20), bg="#FFE4C4", command = startFindFilmOnYear)

#меню Информация об актерах

actorsMenuDescription = Label(root, text='Вы перешли в раздел просмотра информации об актерах\nВыберите в меню интересующее вас действие', bg ="lightgreen", font = ("Consolas", 20))

showActorsButton = Button(root, text='Просмотр информации об актерах', font = ("Consolas", 20), bg="#FFE4C4", command = showAllActors)

addActorButton = Button(root, text='Добавить информацию об актере в ПС', font = ("Consolas", 20), bg="#FFE4C4", command = addActorInDataBase)

editActorButton = Button(root, text='Редактирование информации об актере', font = ("Consolas", 20), bg="#FFE4C4", command = showEditActorsMenu)

findActorButton = Button(root, text='Поиск актера по фамилии и имени', font = ("Consolas", 20), bg="#FFE4C4", command = findActor)

findActorDescriptionLabel = Label(root, text='Вы перешли в раздел поиска информации об актере\nВведите имя и фамилию актера и нажмите\nкнопку "Найти актера" для поиска актера', bg ="lightgreen", font = ("Consolas", 20))

findActorNameLabel = Label(root, text='Имя', bg ="lightgreen", width = 40, font = ("Consolas", 20))

findActorSurnameLabel = Label(root, text='Фамилия', bg ="lightgreen", width = 40, font = ("Consolas", 20))

findActorNameEntry = Entry(root, font = ("Consolas", 20))

findActorSurnameEntry = Entry(root, font = ("Consolas", 20))

startFindActorButton = Button(root, text='Найти актера', font = ("Consolas", 20), bg="#FFE4C4", command = findAndShowActor)

#меню ввода пароля администратора

loginDescriptionLabel = Label(root, text='Вы перешли в меню авторизации\nВведите пароль администратора для того, чтобы вам\nстали доступны все функции ПС "Киностудия"', bg ="lightgreen", font = ("Consolas", 20))

passwordEntry = Entry(font = ("Consolas", 20), width = 40)

checkPasswordButton = Button(root, text='ВХОД', font = ("Consolas", 20), bg="#FFE4C4", command = checkPassword)

editPasswordLabel = Label(root, text='Для установки нового пароля администратора,\nвведите его в поле ниже и нажмите кнопку\n"Установить введенный пароль\nкак пароль администратора"', bg ="lightgreen", font = ("Consolas", 20))

editPasswordEntry = Entry(root, font = ("Consolas", 20))

editPasswordButton = Button(root, text='Установить введенный пароль\nкак пароль администратора', font = ("Consolas", 20), bg="#FFE4C4", command = checkNewPassword)

youAreAdminLabel = Label(root, text='Вы авторизованы как администратор!', bg ="lightgreen", font = ("Consolas", 20))

toBeUser = Button(root, text='Выйти из режима администатора', font = ("Consolas", 20), bg="#FFE4C4", command = returnToUserAccess)

#добавление информации об актере

labelsListWithFilmsForAddActorInDB = []

checkBoxIntVarListForAddActorInDB = []

actorRolesEntryList = []

actorAddNoteHeader = Frame(root)

actorAddNoteBody = Frame(root)

actorAddNoteFooter = Frame(root)

addActorDescriptionLabel = Label(actorAddNoteHeader, width = 90, text='Вы находитесь в разделе добавления информации об актере\nзаполните все поля и нажмите кнопку "Добавить информацию в базу данных"', bg ="lightgreen", font = ("Consolas", 20))

addActorNoteInDataBase = Button(actorAddNoteFooter, text = "Добавить информацию в базу данных", font = ("Consolas", 20), width = 45, bg="#FFE4C4", command = addActorToDB)

scrollableBodyChooseFilmsForActor = Scrollable(actorAddNoteBody, width=30, windowHeight = 350)

returnBackToActorsMenu = Button(actorAddNoteFooter, text='Вернуться обратно в меню действий с актерами', width = 45, font = ("Consolas", 20), bg="#FFE4C4", command = showActorsMenu)

footerAllInformationMenu = Label(root, width = 70, text='Если вы закончили работу, вы можете вернуться в главное меню', bg ="lightgreen", font = ("Consolas", 20))

addActorActorNameLabel = Label(actorAddNoteHeader, font = ("Consolas", 20), bg="#FFE4C4", text="Имя актера\n(русские буквы,\nпробел, дефис,\nпервая буква\nзаглавная).", wraplength = 250)

addActorActorSurnameLabel = Label(actorAddNoteHeader, font = ("Consolas", 20), bg="#FFE4C4", text="Фамилия актера\n(русские буквы,\nпробел, дефис,\nпервая буква\nзаглавная)", wraplength = 250)

addActorActorBirthDayLabel = Label(actorAddNoteHeader, font = ("Consolas", 20), bg="#FFE4C4", text="День рождения\nактера(Выберите дату)", wraplength = 250)

addActorActorGenderLabel = Label(actorAddNoteHeader, font = ("Consolas", 20), bg="#FFE4C4", text="Пол актера\n(выберите из списка)", wraplength = 250)

addActorChooseRolesDescription = Label(actorAddNoteHeader, width = 80, text='Выберите с помощью флажка фильмы, в которых снимался актер', bg ="lightgreen", font = ("Consolas", 20))

addActorChooseRoleCheckboxLabel = Label(scrollableBodyChooseFilmsForActor, font = ("Consolas", 20), bg="#FFE4C4", text="✔", wraplength = 250)

addActorChooseRoleFilmNameLabel = Label(scrollableBodyChooseFilmsForActor, font = ("Consolas", 20), bg="#FFE4C4", text="Название", wraplength = 250)

addActorChooseRoleFilmYearLabel = Label(scrollableBodyChooseFilmsForActor, font = ("Consolas", 20), bg="#FFE4C4", text="Год", wraplength = 250)

addActorChooseRoleFilmStyleLabel = Label(scrollableBodyChooseFilmsForActor, font = ("Consolas", 20), bg="#FFE4C4", text="Жанр", wraplength = 250)

addActorChooseRoleFilmAgeLimitLabel = Label(scrollableBodyChooseFilmsForActor, font = ("Consolas", 20), bg="#FFE4C4", text="Возрастной\nрейтинг", wraplength = 250)

addActorChooseRoleActorRoleLabel = Label(scrollableBodyChooseFilmsForActor, font = ("Consolas", 20), bg="#FFE4C4", text="Роль актера в фильме", wraplength = 250)

addActorActorNameEntry = Entry(actorAddNoteHeader, font = ("Consolas", 20))

addActorActorSurnameEntry = Entry(actorAddNoteHeader, font = ("Consolas", 20))

addActorActorBirthDayEntry = Button(actorAddNoteHeader, text = "Выберите дату рождения", background='#FFE4C4', font = ("Consolas", 20), command = chooseDate)

addActorActorGenderEntry = Listbox(actorAddNoteHeader, font = ("Consolas", 16), height = 2, width = 50)

addActorActorGenderEntry.insert(END, "м")

addActorActorGenderEntry.insert(END, "ж")

addActorFinishDescription = Label(actorAddNoteFooter, width = 90, text='Если вы закончили работу, вы можете добавить запись в базу\nданных или вернуться обратно в меню актеров', bg ="lightgreen", font = ("Consolas", 20))

#добавление информации о фильме

labelsListWithActorsForAddFilmInDB = []

checkBoxIntVarListForAddFilmInDB = []

allActorsFromDBDataList = []

addFilmActorRolesEntryList = []

filmAddNoteHeader = Frame(root)

filmAddNoteBody = Frame(root)

filmAddNoteFooter = Frame(root)

addFilmDescriptionLabel = Label(filmAddNoteHeader, width = 98, text='Вы находитесь в разделе добавления информации о фильме\nзаполните поля информации о фильме и выберите актеров, которые в нем снимались,\nдля каждого актера вы можете указать персонажа, которого он играет.\nПосле того, как все поля будут заполнены, нажмите кнопку "Добавить информацию в базу данных"', bg ="lightgreen", font = ("Consolas", 20))

scrollableBodyChooseFilmsForfilm = Scrollable(filmAddNoteBody, width=30, windowHeight = 560)

addFilmFilmNameLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Название фильма\n(можно использовать:\nрусские буквы, пробел и дефис\n(не более 1 символа подряд))")

addFilmFilmYearLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Год выхода фильма\n(запишите год\nот 1880 до текущего)")

addFilmFilmLengthLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Продолжительность\n(количество минут)")

addFilmFilmStyleLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Жанр фильма\n(можно использовать:\nрусские буквы, пробел и дефис\n(не более 1 символа подряд))")

addFilmFilmAgeLimitLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Возрастной рейтинг\n(Выберите из списка)")

addFilmFilmCostLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Бюджет\n(введите количество $\nот 1 $ до 10 млрд $)")

addFilmFilmUSAProfitLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Сборы в США\n(введите количество $\nот 1 $ до 10 млрд $)")

addFilmFilmWorldProfitLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Сборы в мире\n(введите количество $\nот 1 $ до 10 млрд $,\nесли у вас нет данных,\nоставьте поле пустым)")

addFilmFilmAwardsLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 18), bg="#FFE4C4", text="Награды\n(перечислите награды\nфильма, используя:\nанглийские и русские буквы,\nцифры, символы:\nпробел, '(', ')', ':', '-',\nлибо оставьте поле пустым,\nмаксимум 3000 символов)")

addFilmChooseRolesDescription = Label(scrollableBodyChooseFilmsForfilm, text='Выберите актеров, которые снимались в этом фильме:', bg ="lightgreen", font = ("Consolas", 20))

#addFilmDataFieldEntryWidth = 60

addFilmFilmNameDataField = Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

addFilmFilmYearDataField = Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

addFilmFilmLengthDataField = Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

addFilmFilmStyleDataField = Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

addFilmAgeLimitComboboxValue = StringVar()

addFilmFilmAgeLimitDataField = ttk.Combobox(scrollableBodyChooseFilmsForfilm, values = [u"0", u"3", u"6", u"12", u"14", u"16", u"18"], textvariable= addFilmAgeLimitComboboxValue, font = ("Consolas", 16), height=7)#Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

addFilmFilmAgeLimitDataField.current(0)#.set(u"18")

#addFilmFilmAgeLimitDataField.bind("<<ComboboxSelected>>", combobxSelected)

addFilmFilmCostDataField = Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

addFilmFilmUSAProfitDataField = Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

addFilmFilmWorldProfitDataField = Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

addFilmFilmAwardsDataField = Text(scrollableBodyChooseFilmsForfilm, height = 2, wrap = WORD, font = ("Consolas", 16))

# addFilmFilmAwardsDataFieldScrollbar = Scrollbar(command=addFilmFilmAwardsDataField.yview)

# #addFilmFilmAwardsDataFieldScrollbar.pack(side=LEFT, fill=Y)

# addFilmFilmAwardsDataField.config(yscrollcommand=addFilmFilmAwardsDataFieldScrollbar.set)

addFilmChooseRoleCheckboxLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 20), bg="#FFE4C4", text="✔", wraplength = 250)

addFilmChooseRoleFilmNameLabel = Label(scrollableBodyChooseFilmsForfilm, width = 20, font = ("Consolas", 20), bg="#FFE4C4", text="Имя", wraplength = 250)

addFilmChooseRoleFilmYearLabel = Label(scrollableBodyChooseFilmsForfilm, width = 20, font = ("Consolas", 20), bg="#FFE4C4", text="Фамилия", wraplength = 250)

addFilmChooseRoleFilmStyleLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 20), bg="#FFE4C4", text="Дата рождения", wraplength = 250)

addFilmChooseRoleFilmAgeLimitLabel = Label(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 20), bg="#FFE4C4", text="Пол", wraplength = 250)

addFilmChooseRoleActorRoleLabel = Label(scrollableBodyChooseFilmsForfilm, width = 30, font = ("Consolas", 20), bg="#FFE4C4", text="Роль актера в фильме", wraplength = 320)

addFilmFinishDescription = Label(filmAddNoteFooter, width = 98, text='Если вы закончили работу, вы можете добавить запись в базу\nданных или вернуться обратно в меню фильмов', bg ="lightgreen", font = ("Consolas", 20))

returnBackTofilmsMenu = Button(filmAddNoteFooter, text='Вернуться обратно в меню действий с фильмами', font = ("Consolas", 20), bg="#FFE4C4", command = showFilmsMenu)

addFilmNoteInDataBase = Button(filmAddNoteFooter, text = "Добавить фильм в базу данных", font = ("Consolas", 20), bg="#FFE4C4", command = checkNewFilmDataNoteAndAddItInDB)

#footerAllInformationMenu = Label(root, width = 70, text='Если вы закончили работу, вы можете вернуться в главное меню', bg ="lightgreen", font = ("Consolas", 20))

# addActorActorNameEntry = Entry(actorAddNoteHeader, font = ("Consolas", 20))

# addActorActorSurnameEntry = Entry(actorAddNoteHeader, font = ("Consolas", 20))

# addActorActorBirthDayEntry = Entry(actorAddNoteHeader, font = ("Consolas", 20))

# addActorActorGenderEntry = Listbox(actorAddNoteHeader, font = ("Consolas", 16), height = 2, width = 50)

# addActorActorGenderEntry.insert(END, "м")

# addActorActorGenderEntry.insert(END, "ж")

#Entry(actorAddNoteHeader, font = ("Consolas", 20), wraplength = 250)

#вывод информации обо всех фильмах

fromAllFilms = False

allFilmsHeader = Frame(root)

allFilmsBody = Frame(root)

allFilmsFooter = Frame(root)

scrollableBodyAllFilms = Scrollable(allFilmsBody, width=30)

footerAllInformationMenu = Label(allFilmsFooter, width = 70, text='Если вы закончили работу, вы можете вернуться в главное меню', bg ="lightgreen", font = ("Consolas", 20))

backToMainMenuFromFilms = Button(allFilmsFooter, font = ("Consolas", 20), bg="#FFE4C4", text="Вернуться в главное меню", command = showMainMenu)

allFilmsInformationLabel = Label(allFilmsHeader, font = ("Consolas", 20), width = 70, text="Вы находитесь в меню просмотра всех фильмов\nНажмите на интересующий вас фильм,\nчтобы получить больше информации о нем", bg ="lightgreen")

filmNameLabel = Label(allFilmsHeader, font = ("Consolas", 20), bg="#FFE4C4", text="Название\nфильма", width = 12)

filmYearLabel = Label(allFilmsHeader, font = ("Consolas", 20), bg="#FFE4C4", text="Год\nвыхода", width = 1)

filmLengthLabel = Label(allFilmsHeader, font = ("Consolas", 20), bg="#FFE4C4", text="длительность\n(минут)", width = 2)

filmStyleLabel = Label(allFilmsHeader, font = ("Consolas", 20), bg="#FFE4C4", text="жанр", width = 12)

#вывод информации о конкретном фильме

filmInfoDataList = []

actorsInFilmDataFramesList = []

filmData = []

fromFilm = False

filmID = -5

oneFilmheader = Frame(root)

oneFilmbody = Frame(root)

oneFilmfooter = Frame(root)

scrollableBodyOneFilm = Scrollable(oneFilmbody, width=30, windowHeight= 700)

filmInformationLabelWidth = 25

filmInformationDescription = Label(oneFilmheader, width = 80, text='Вы открыли окно информации о фильме', bg ="lightgreen", font = ("Consolas", 20))

filmInfoName = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Название фильма", width = filmInformationLabelWidth)

filmInfoYear = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Год выпуска", width = filmInformationLabelWidth)

filmInfoDuration = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Длительность в минутах", width = filmInformationLabelWidth)

filmInfoStyle = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="жанр", width = filmInformationLabelWidth)

filmInfoAge = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Возрастной рейтинг", width = filmInformationLabelWidth)

filmInfoProdactionCost = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Бюджет", width = filmInformationLabelWidth)

filmInfoUSAProfit = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Сборы в США", width = filmInformationLabelWidth)

filmInfoWorldProfit = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Сборы в мире", width = filmInformationLabelWidth)

filmInfoAwards = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Награды", width = filmInformationLabelWidth)

backToFilmsFromFilm = Button(oneFilmfooter, font = ("Consolas", 20), bg="#FFE4C4", text="Вернуться в меню фильмов", command = showAllFilms)

actorsInFilmDescriptionLabel = Label(scrollableBodyOneFilm, font = ("Consolas", 20), text="Актеры, снявшиеся в фильме", bg ="lightgreen", width = filmInformationLabelWidth)

#вывод информации о конкретном актере

actorsInOneRowCount = 3

actorInformationLabelWidth = 38

actorRolesLabelsList = []

transitionBackToFilmFromActor = False

fromActor = False

actorsDataList = []

actorRolesDataList = []

actorDataLabelsList = []

fromThisFilmIDToActor = -5

oneActorheader = Frame(root)

oneActorbody = Frame(root)

oneActorfooter = Frame(root)

scrollableBodyOneActor = Scrollable(oneActorbody, width=30)

actorInformationDescription = Label(oneActorheader, text='Вы открыли окно информации об актере', bg ="lightgreen", font = ("Consolas", 20), width = 84)

actorInfoName = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Имя актера",bg ="#FFE4C4", width = 20)

actorInfoSurname = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Фамилия актера",bg ="#FFE4C4", width = 20)

actorBirthDay = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Дата рождения",bg ="#FFE4C4", width = 20)

actorRolesDescriptionLabel = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Актер принимал участие в фильмах:", bg ="lightgreen")

actorRolesFilmName = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Название\nфильма",bg ="#FFE4C4", width = 20)

actorRolesFilmYear = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Год\nвыхода",bg ="#FFE4C4", width = 15)

actorRolesStyle = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Жанр",bg ="#FFE4C4", width = 15)

actorRolesCharacter = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Персонаж",bg ="#FFE4C4", width = 15)

actorRolesAge = Label(scrollableBodyOneActor, font = ("Consolas", 20), text="Возрастной\nрейтинг",bg ="#FFE4C4", width = 10)

backToFilmFromActor = Button(oneActorfooter, font = ("Consolas", 20), bg="#FFE4C4", text="Вернуться обратно к фильму")

backToFilmFromActor.bind('<Button-1>', lambda event: showFilmData(event, "обратно к фильму от актера"))

backToActorsFromActor = Button(oneActorfooter, font = ("Consolas", 20), text = "Вернуться обратно к списку актеров", bg ="#FFE4C4", command = showAllActors)

#просмотр информации обо всех актерах

fromAllActors = False

allActorsheader = Frame(root)

allActorsbody = Frame(root)

allActorsfooter = Frame(root)

scrollableBodyallActors = Scrollable(allActorsbody, width=30)

allActorsInformationLabelWidth = 25

allActorsInformationDescription = Label(allActorsheader, text='Вы открыли окно просмотра информации об актерах\nВыберите актера, информацию о котором хотите просмотреть', bg ="lightgreen", font = ("Consolas", 20), width = 70)

allActorsInfoName = Label(allActorsheader, font = ("Consolas", 20), text="Имя\nактера", width = 25)

allActorsInfoSurname = Label(allActorsheader, font = ("Consolas", 20), text="Фамилия\nактера", width = 16)

allActorsBirthDay = Label(allActorsheader, font = ("Consolas", 20), text="Дата рождения\nдд.мм.гггг", width = 17)

allActorsGender = Label(allActorsheader, font = ("Consolas", 20), text="Пол\nактера", width = 12)

backToActorsMenuFromActors = Button(allActorsfooter, font = ("Consolas", 20), text = "Вернуться обратно в меню актеров", bg ="#FFE4C4", command = showActorsMenu)

#меню со списоком всех актеров для редактирования

fromAllActors = False

editActorsheader = Frame(root)

editActorsbody = Frame(root)

editActorsfooter = Frame(root)

scrollableBodyEditAllActors = Scrollable(editActorsbody, width=30)

editActorsInformationLabelWidth = 25

editActorsInformationDescription = Label(editActorsheader, text='Вы открыли окно редактирования информации об актерах\nНайдите актера, информацию о котором вы хотите отредактировать.\nНажмите на кнопку "🖉" для того, чтобы редактировать информацию об актере.\nНажмите кнопку "⌦", чтобы удалить информацию об актере из ПС.', bg ="lightgreen", font = ("Consolas", 20), width = 79)

editActorsInfoOptions = Label(editActorsheader, font = ("Consolas", 20), text="Опции", width = 6)

editActorsInfoName = Label(editActorsheader, font = ("Consolas", 20), text="Имя\nактера", width = 18)

editActorsInfoSurname = Label(editActorsheader, font = ("Consolas", 20), text="Фамилия\nактера", width = 20)

editActorsBirthDay = Label(editActorsheader, font = ("Consolas", 20), text="Дата рождения\nдд.мм.гггг", width = 14)

editActorsGender = Label(editActorsheader, font = ("Consolas", 20), text="Пол\nактера", width = 8)

backToActorsMenuFromEditActors = Button(editActorsfooter, font = ("Consolas", 20), text = "Вернуться обратно в меню актеров", bg ="#FFE4C4", command = showActorsMenu)

#редактирование информации о выбранном актере

allActorsEditActorDataWidgetsList = []

editActorsDataAllActorsList = []

editorAddNoteHeader = Frame(root)

editorAddNoteBody = Frame(root)

editorAddNoteFooter = Frame(root)

scrollableBodyEditActor = Scrollable(editorAddNoteBody, width=30, windowHeight = 350)

editActorDescriptionLabel = Label(editorAddNoteHeader, width = 90, text='Вы находитесь в разделе добавления информации об актере\nзаполните все поля и нажмите кнопку "Добавить информацию в базу данных"', bg ="lightgreen", font = ("Consolas", 20))

editActorActorNameLabel = Label(editorAddNoteHeader, font = ("Consolas", 20), bg="#FFE4C4", text="Имя актера\n(русские буквы,\nдефис, пробел,\nпервая буква\nзаглавная)", wraplength = 250)

editActorActorSurnameLabel = Label(editorAddNoteHeader, font = ("Consolas", 20), bg="#FFE4C4", text="Фамилия актера\n(русские буквы,\nдефис, пробел,\nпервая буква\nзаглавная)", wraplength = 250)

editActorActorBirthDayLabel = Label(editorAddNoteHeader, font = ("Consolas", 20), bg="#FFE4C4", text="День рождения\nактера\n(выберите из списка)", wraplength = 250)

editActorActorGenderLabel = Label(editorAddNoteHeader, font = ("Consolas", 20), bg="#FFE4C4", text="Пол актера\n(выберите из списка)", wraplength = 250)

editActorChooseRolesDescription = Label(editorAddNoteHeader, width = 80, text='Выберите фильмы, в которых снимался актер', bg ="lightgreen", font = ("Consolas", 20))

editActorChooseRoleCheckboxLabel = Label(scrollableBodyEditActor, font = ("Consolas", 20), bg="#FFE4C4", text="✔", wraplength = 250)

editActorChooseRoleFilmNameLabel = Label(scrollableBodyEditActor, font = ("Consolas", 20), bg="#FFE4C4", text="Название", wraplength = 250)

editActorChooseRoleFilmYearLabel = Label(scrollableBodyEditActor, font = ("Consolas", 20), bg="#FFE4C4", text="Год", wraplength = 250)

editActorChooseRoleFilmStyleLabel = Label(scrollableBodyEditActor, font = ("Consolas", 20), bg="#FFE4C4", text="Жанр", wraplength = 250)

editActorChooseRoleFilmAgeLimitLabel = Label(scrollableBodyEditActor, font = ("Consolas", 20), bg="#FFE4C4", text="Возрастной\nрейтинг", wraplength = 250)

editActorChooseRoleActorRoleLabel = Label(scrollableBodyEditActor, font = ("Consolas", 20), bg="#FFE4C4", text="Роль актера в фильме", wraplength = 250)

editActorActorNameEntry = Entry(editorAddNoteHeader, font = ("Consolas", 20))

editActorActorSurnameEntry = Entry(editorAddNoteHeader, font = ("Consolas", 20))

editActorNewActorBirthDayEntry = Button(editorAddNoteHeader, text = "Выберите дату рождения", background='#FFE4C4', font = ("Consolas", 20), command = chooseNewDateForActor)

editActorActorGenderEntry = Listbox(editorAddNoteHeader, font = ("Consolas", 16), height = 2, width = 50)

editActorActorGenderEntry.insert(END, "м")

editActorActorGenderEntry.insert(END, "ж")

editActorFinishDescription = Label(editorAddNoteFooter, width = 90, text='Если вы закончили работу, вы можете добавить запись в базу\nданных или вернуться обратно в меню актеров', bg ="lightgreen", font = ("Consolas", 20))

editActorNoteInDataBase = Button(editorAddNoteFooter, text = "Обновить информацию в базе данных", font = ("Consolas", 20), width = 45, bg="#FFE4C4", command = checkNewActorDataAndAddItInDB)

returnBackToActorMenuFromActorEdit = Button(editorAddNoteFooter, text='Вернуться обратно в меню действий с актерами', width = 45, font = ("Consolas", 20), bg="#FFE4C4", command = showActorsMenu)

#меню со списоком фильмов для редактирования

editFilmAllFilmsWigetsList = []

editFilmsHeader = Frame(root)

editFilmsBody = Frame(root)

editFilmsFooter = Frame(root)

scrollableBodyEditFilms = Scrollable(editFilmsBody, width=30, windowHeight = 520)

editFilmsInformationLabel = Label(editFilmsHeader, font = ("Consolas", 20), width = 77,

text="Вы находитесь в меню редактирования информации о фильмах\nВыберите фильм для редактирования, после чего\n\

нажмите на кнопку '🖉' рядом с ним для того, чтобы ввести новые данные.\n\

Нажмите кнопку '⌦', чтобы удалить информацию о фильме из ПС.", bg ="lightgreen")

editFilmOptionsLabel = Label(editFilmsHeader, font = ("Consolas", 20), text="Опции", width = 6)

editFilmNameLabel = Label(editFilmsHeader, font = ("Consolas", 20), text="Название\nфильма", width = 20)

editFilmYearLabel = Label(editFilmsHeader, font = ("Consolas", 20), text="Год\nвыхода", width = 10)

editFilmLengthLabel = Label(editFilmsHeader, font = ("Consolas", 20), text="длительность\n(минут)", width = 12)

editFilmStyleLabel = Label(editFilmsHeader, font = ("Consolas", 20), text="жанр", width = 20)

footerEditFilms = Label(editFilmsFooter, width = 77, text='Если вы закончили работу, вы можете вернуться в главное меню', bg ="lightgreen", font = ("Consolas", 20))

backToMainMenuFromEditFilmsMenu = Button(editFilmsFooter, font = ("Consolas", 20), bg="#FFE4C4", text="Вернуться в главное меню", command = showMainMenu)

#редактирование конкретного фильма

editFilmNoteHeader = Frame(root)

editFilmNoteBody = Frame(root)

editFilmNoteFooter = Frame(root)

scrollableBodyEditChoosedFilm = Scrollable(editFilmNoteBody, width=30, windowHeight = 520)

checkBoxListEditFilmChooseActor = []

editFilmActorsRoleEntryList = []

#добавление информации о фильме

labelsListWithActorsForEditFilmInDB = []

checkBoxIntVarListForEditFilmInDB = []

# allActorsFromDBDataList = []

# addFilmActorRolesEntryList = []

# filmAddNoteHeader = Frame(root)

# filmAddNoteBody = Frame(root)

# filmAddNoeditFilmNoteFooterteFooter = Frame(root)

editFilmDescriptionLabel = Label(editFilmNoteHeader, width = 98, text='Вы находитесь в разделе редактирования информации о фильме\nзаполните поля информации о фильме и выберите актеров, которые в нем снимались,\nдля каждого актера вы можете указать персонажа, которого он играет.\nПосле того, как все поля будут заполнены, нажмите кнопку "Добавить информацию в базу данных"', bg ="lightgreen", font = ("Consolas", 20))

# scrollableBodyChooseFilmsForfilm = Scrollable(filmAddNoteBody, width=30, windowHeight = 560)

editFilmFilmNameLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Название фильма\n(можно использовать:\nрусские буквы, пробел и дефис\n(не более 1 символа подряд))")

editFilmFilmYearLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Год выхода фильма\n(запишите год\nот 1880 до текущего)")

editFilmFilmLengthLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Продолжительность\n(количество минут)")

editFilmFilmStyleLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Жанр фильма\n(можно использовать:\nрусские буквы, пробел и дефис\n(не более 1 символа подряд))")

editFilmFilmAgeLimitLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Возрастной рейтинг\n(Выберите из списка)")

editFilmFilmCostLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Бюджет\n(введите количество $\nот 1 $ до 10 млрд $)")

editFilmFilmUSAProfitLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Сборы в США\n(введите количество $\nот 1 $ до 10 млрд $)")

editFilmFilmWorldProfitLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Сборы в мире\n(введите количество $\nот 1 $ до 10 млрд $,\nесли у вас нет данных,\nоставьте поле пустым)")

editFilmFilmAwardsLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 18), bg="#FFE4C4", text="Награды\n(перечислите награды\nфильма, используя:\nанглийские и русские буквы,\nцифры, символы:\nпробел, '(', ')', ':', '-',\nлибо оставьте поле пустым,\nмаксимум 3000 символов)")

editFilmChooseRolesDescription = Label(scrollableBodyEditChoosedFilm, text='Выберите актеров, которые снимались в этом фильме:', bg ="lightgreen", font = ("Consolas", 20))

#addFilmDataFieldEntryWidth = 60

editFilmFilmNameDataField = Entry(scrollableBodyEditChoosedFilm, font = ("Consolas", 16))

editFilmFilmYearDataField = Entry(scrollableBodyEditChoosedFilm, font = ("Consolas", 16))

editFilmFilmLengthDataField = Entry(scrollableBodyEditChoosedFilm, font = ("Consolas", 16))

editFilmFilmStyleDataField = Entry(scrollableBodyEditChoosedFilm, font = ("Consolas", 16))

editFilmAgeLimitComboboxValue = StringVar()

editFilmFilmAgeLimitDataField = ttk.Combobox(scrollableBodyEditChoosedFilm, values = [u"0", u"3", u"6", u"12", u"14", u"16", u"18"], textvariable= addFilmAgeLimitComboboxValue, font = ("Consolas", 16), height=7)#Entry(scrollableBodyChooseFilmsForfilm, font = ("Consolas", 16))

editFilmFilmAgeLimitDataField.current(0)#.set(u"18")

#addFilmFilmAgeLimitDataField.bind("<<ComboboxSelected>>", combobxSelected)

editFilmFilmCostDataField = Entry(scrollableBodyEditChoosedFilm, font = ("Consolas", 16))

editFilmFilmUSAProfitDataField = Entry(scrollableBodyEditChoosedFilm, font = ("Consolas", 16))

editFilmFilmWorldProfitDataField = Entry(scrollableBodyEditChoosedFilm, font = ("Consolas", 16))

editFilmFilmAwardsDataField = Text(scrollableBodyEditChoosedFilm, height = 2, wrap = WORD, font = ("Consolas", 16))

# addFilmFilmAwardsDataFieldScrollbar = Scrollbar(command=addFilmFilmAwardsDataField.yview)

# #addFilmFilmAwardsDataFieldScrollbar.pack(side=LEFT, fill=Y)

# addFilmFilmAwardsDataField.config(yscrollcommand=addFilmFilmAwardsDataFieldScrollbar.set)

editFilmChooseRoleCheckboxLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 20), bg="#FFE4C4", text="✔", wraplength = 250)

editFilmChooseRoleFilmNameLabel = Label(scrollableBodyEditChoosedFilm, width = 20, font = ("Consolas", 20), bg="#FFE4C4", text="Имя", wraplength = 250)

editFilmChooseRoleFilmYearLabel = Label(scrollableBodyEditChoosedFilm, width = 20, font = ("Consolas", 20), bg="#FFE4C4", text="Фамилия", wraplength = 250)

editFilmChooseRoleFilmStyleLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 20), bg="#FFE4C4", text="Дата рождения", wraplength = 250)

editFilmChooseRoleFilmAgeLimitLabel = Label(scrollableBodyEditChoosedFilm, font = ("Consolas", 20), bg="#FFE4C4", text="Пол", wraplength = 250)

editFilmChooseRoleActorRoleLabel = Label(scrollableBodyEditChoosedFilm, width = 30, font = ("Consolas", 20), bg="#FFE4C4", text="Роль актера в фильме", wraplength = 320)

editFilmFinishDescription = Label(editFilmNoteFooter, width = 98, text='Если вы закончили редактирование, нажмите кнопку "Редактировать запись в базе данных"', bg ="lightgreen", font = ("Consolas", 20))

returnBackTofilmsMenuFromEditFilm = Button(editFilmNoteFooter, text='Вернуться обратно в меню действий с фильмами', font = ("Consolas", 20), bg="#FFE4C4", command = showFilmsMenu)

editFilmNoteInDataBase = Button(editFilmNoteFooter, text = "Редактировать запись в базе данных", font = ("Consolas", 20), bg="#FFE4C4", command = checkNewFilmDataAndUpdateItInDB)

accessLevel = "User"

showMainMenu()

root.mainloop()

# Руководство системного программиста

## 11.1. Требования к информационной и программной совместимости

Программа должна быть написан на языке программирования Python версии 3.8 для операционной системы Windows различных версий.

## 11.2. Требования к составу и параметрам технических средств

Программа должна работать на персональных компьютерах, которые имеют следующие минимальные характеристики:

* ОС на ПК
* Оперативная память - 512 МВ:
* Тактовая частота процессора – 1,3 Ггц;
* Размер жесткого диска не менее 40 Гб.

## 11.3. Установка компонентов для корректной работы программы

Для установки Python необходимо найти на официальной [странице скачивания](https://www.python.org/downloads/) официального сайта python.org скачать необходимый выпуск ЯП Python 3-ей версии, после чего запустить скачанный файл. По умолчанию в комплекте с Python устанавливается IDLE (среда разработки Python-программ), документация, менеджер пакетов pip (для скачивания и установки других пакетов на Python), стандартный набор тестов и Python Launcher для запуска программ двойным щелчком мыши по файлу .py в Проводнике Windows.

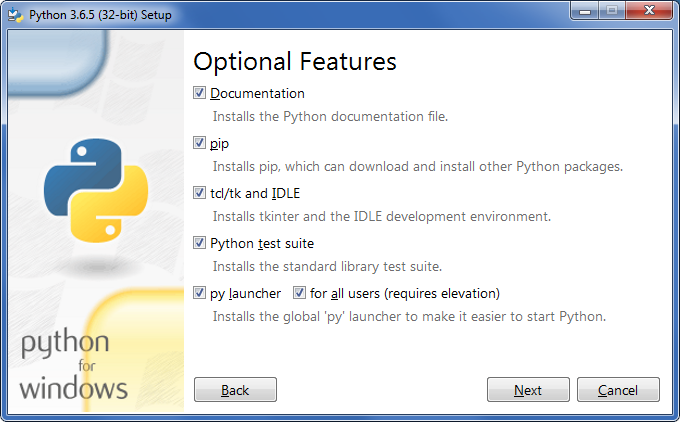


Рис. 35. Установщик Python

На следующем этапе можно выбрать путь установки и желательно указать «Установку для всех пользователей» (Install for all users) (Рис.9.)*.*

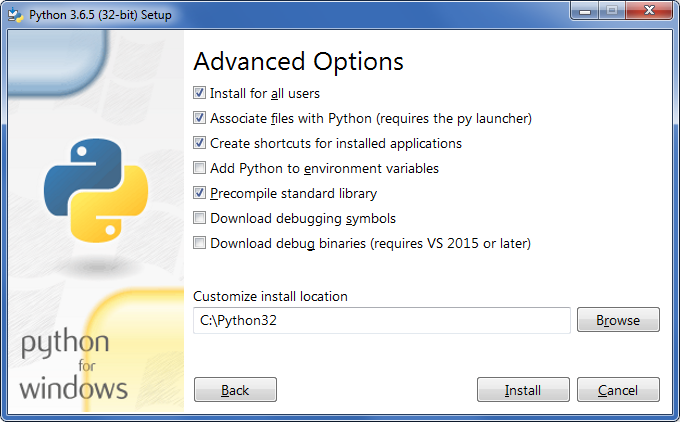


Рис. 36. Продвинутые настройки при установке Python

Для открытия программы необходимо нажать на программу правой кнопкой мыши, выбрать из выпадающего окна пункт “Edit with IDLE” и нажать на неё.

Для запуска программы нудно нажать в выпадающем меню “Run” –> “Run Modul” или нажать “F5”.

Для установки библиотек, необходимых для работы ПС «Киностудия», необходимо вызвать строку «Выполнить» комбинацией клавиш Win+R, прописать в строке «Выполнить» команду «CMD», нажать клавишу Enter, после чего будет запущена командная строка Windows.

В командной строке Windows небходимо выполнить установку необходимых библиотек командой: pip install [имя\_библиотеки], где имя\_библиотеки – название библиотеки.

## 11.4. Безопасность ПС «Киностудия»

На момент выпуска, ПС киностудия имеет парольную защиту, отраженную в таблице 4.

Табл. 4. – Парольная защита ПС «Киностудия»

|  |  |
| --- | --- |
| Элемент | Пароль |
| ПС «Киностудия» | abc123 |
| Файл базы данных \*.accdb | 123456 |

# Руководство пользователя

ПС Киностудия направленена на создание надежного хранилища данных о киностудии, обеспечение удобного графического интерфейса для работы с базой данных.

Основные пользователи ПС – обычные пользователи, однако в программе предусмотрен доступ к режиму администратора, тогда у вас также появляется возможность добавлять, редактировать и удалять информацию и фильмах, и актерах.

Сразу при запуске программы на рис. 37 вы можете увидеть главное меню программы, которое служит «отправной точкой» при работе с ПС.

Самый «верхний» пункт меню «Информация о разработчике ПС «Киностудия»» вызывает показ информации о разработчике. При выборе этого пункта меню окно преобразуется в представленное на рис. 38.

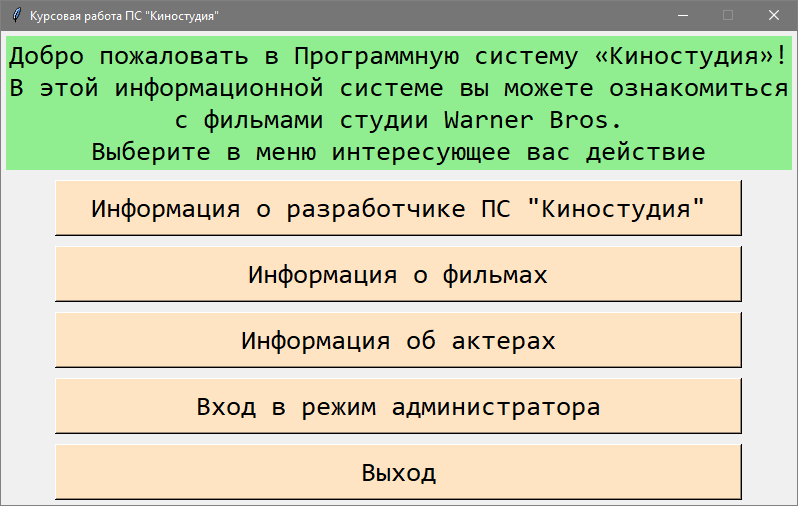


Рис. 37. Вид главного меню после запуска программы

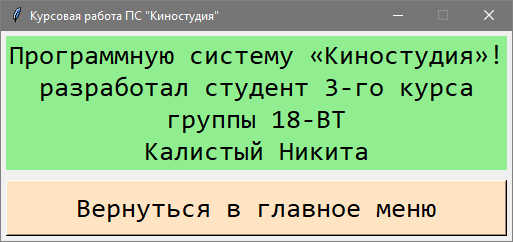


Рис. 38. Окно с информацией о разработчике ПС Киностудия

В окне информации о разработчике можно только посмотреть информацию о разработчике, поэтому вернемся обратно в главное меню (рис. 37), после чего перейдем в подменю «Информация о фильмах» рис. 39. В этом окне выполняется просмотр информации о фильмах, запрос на поиск фильма по названию, поиск фильма по жанру, по году. Также, здесь, при наличии прав администратора будут доступны пункты добавления и редактирования фильмов в ПС.

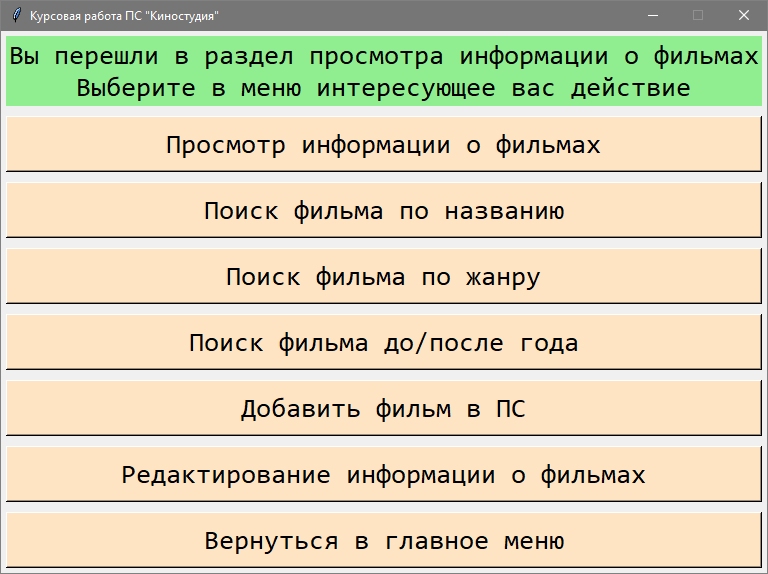


Рис. 39. Окно «просмотра информации о фильмах

Начнем просмотр информации о фильмах. После выбора этого пункта окно преобразуется в представленное на рис. 40. В окне «просмотра всех фильмов» можно выбрать фильм, информацию о котором необходимо просмотреть, и получить более детальные сведения, а также узнать, какие актеры в нем снимались, это можно увидеть на рис. 41.



Рис. 40. Окно просмотра всех фильмов в ПС

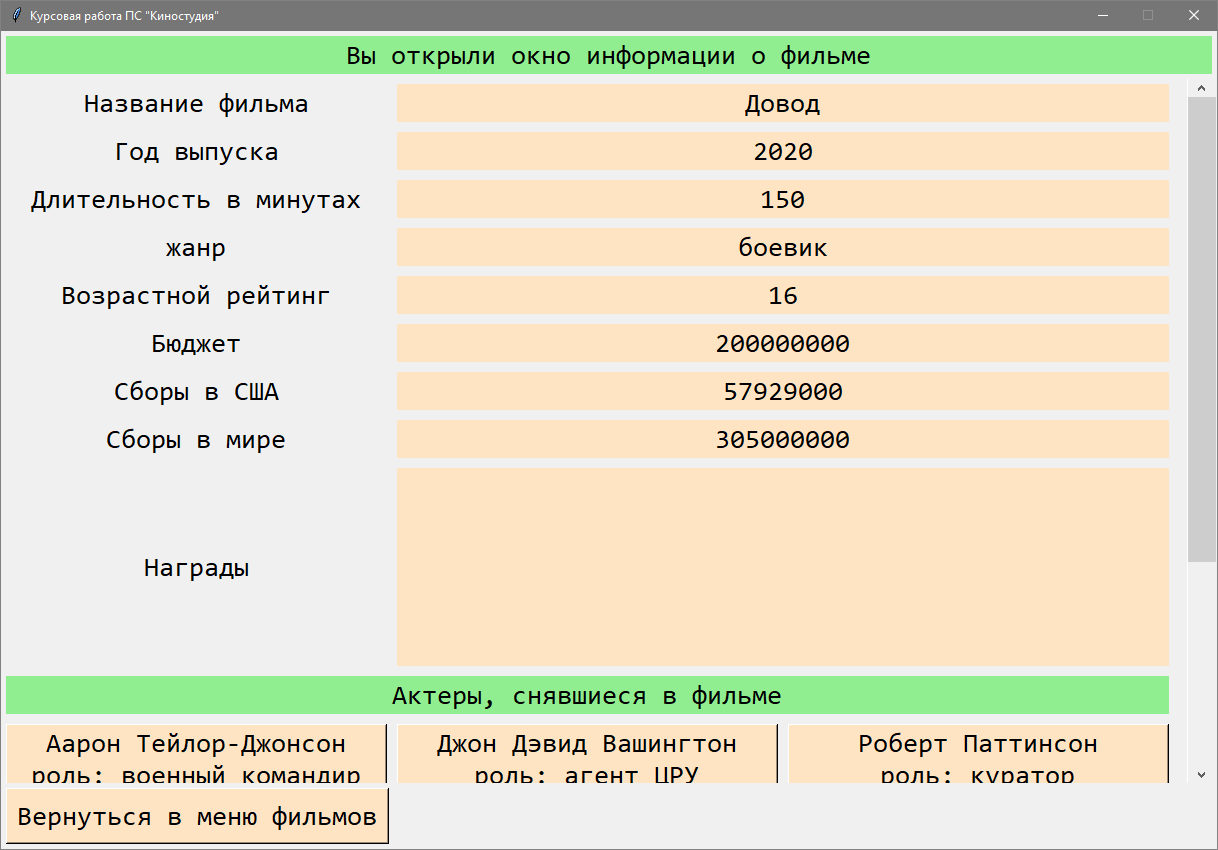


Рис. 41. Окно просмотра информации о выбранном фильме



Рис. 41. Продолжение

Просматривая информацию о фильме, вернемся обратно в меню фильмов (рис. 39) для того, чтобы просмотреть остальные функции, связанные с фильмами. Выберем пункт меню «поиск фильма по названию», после чего окно станет выглядеть, как показано на рис. 42.

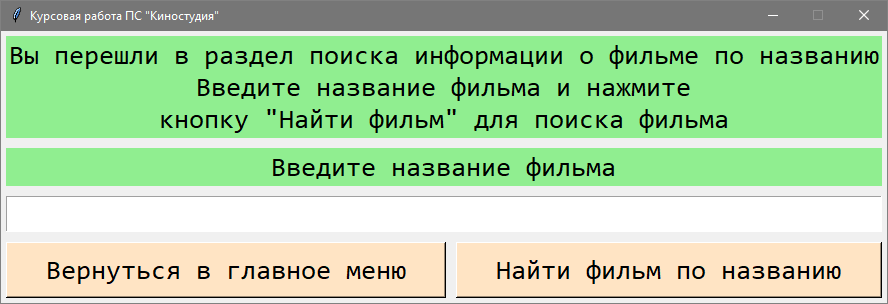


Рис. 42. Поиск фильма по названию

Выполним поиск по названию фильма, для этого впишем в текстовое поле название фильма, например «Довод» (рис. 43).

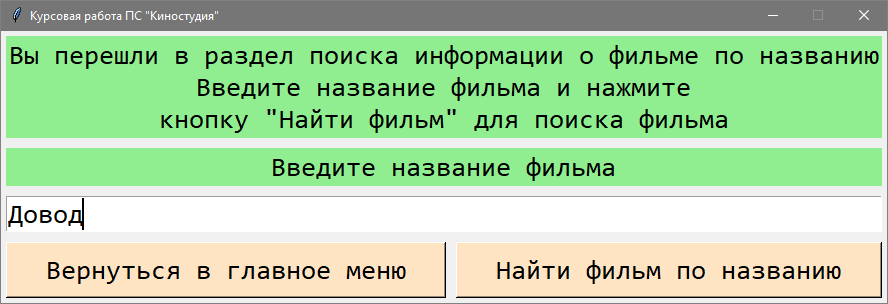


Рис. 43. Поиск фильма «Довод»

После нажатия клавиши «найти фильм по названию» появится окно о том, что операция выполнена успешно, и что по нашему запросу найдена запись в базе данных рис. 44, однако если фильма с таким названием не будет найдено, то пользователь увидит сообщение, представленное на рис. 45.

Если фильм был найден в ПС, то в окне пользователь увидит окно с результатом поиска, как, например на рис. 46.

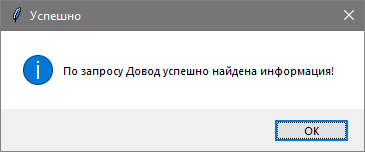


Рис. 44. Поиск фильма «Довод»

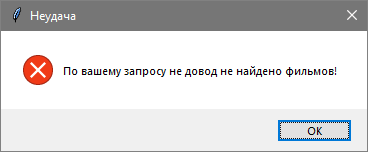


Рис. 45. По запросу фильма по названию не найдено фильма

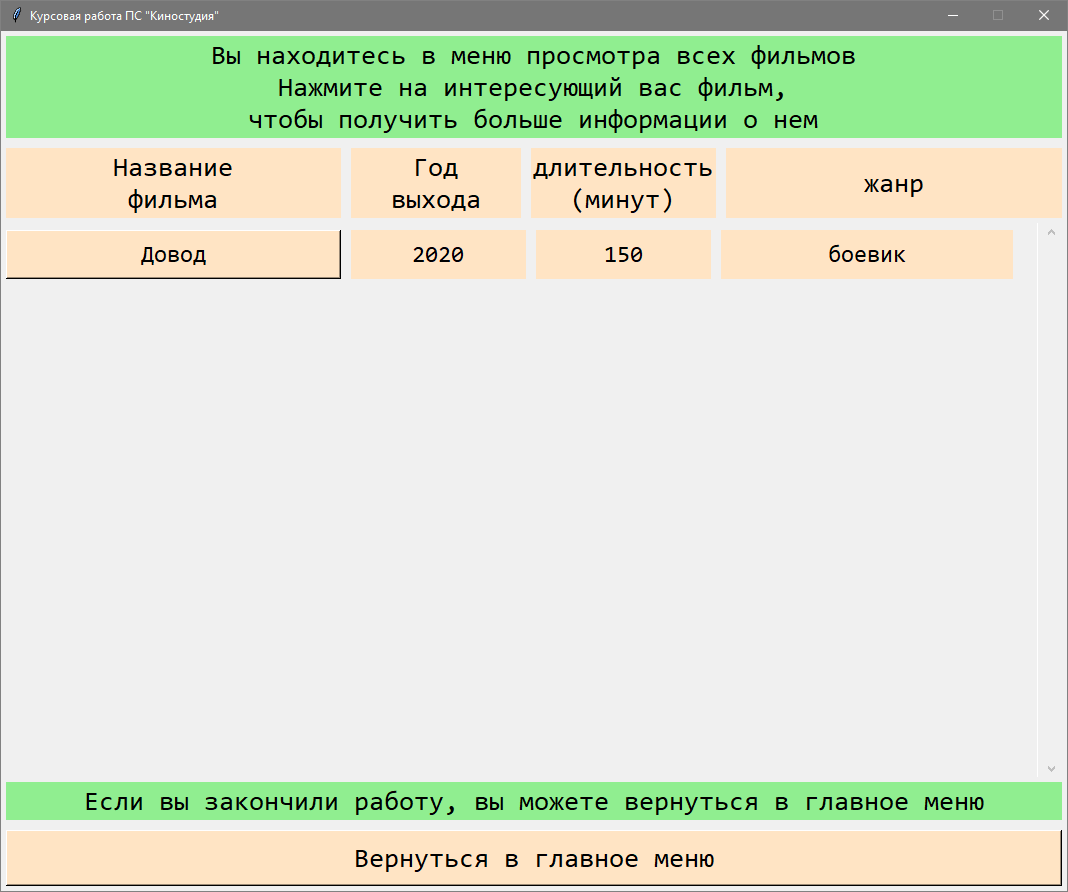


Рис. 46. Результаты поиска по запросу фильма «Довод»

Вернемся обратно в меню фильмов (рис. 39), и найдем фильм по жанру. В окне появится надпись о том, что пользователь находится в меню поиска фильма по жанру и строка для ввода жанра, попробуем найти фильм с жанром «боевик» (рис. 47).

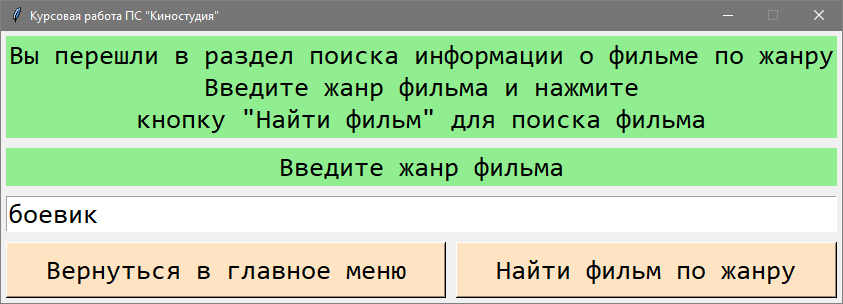


Рис. 47. Поиск фильма по жанру «боевик»

Если фильмы с таким жанром будут найдены, то пользователь увидит сообщение (рис. 44), что по такому запросу были найдены фильмы, иначе получит сообщение (рис. 45) о том, что таких фильмов не нашлось. Если же фильмы нашлись, пользователь в окне, на рис. 48 увидит все фильмы, которые принадлежат к введенному жанру.

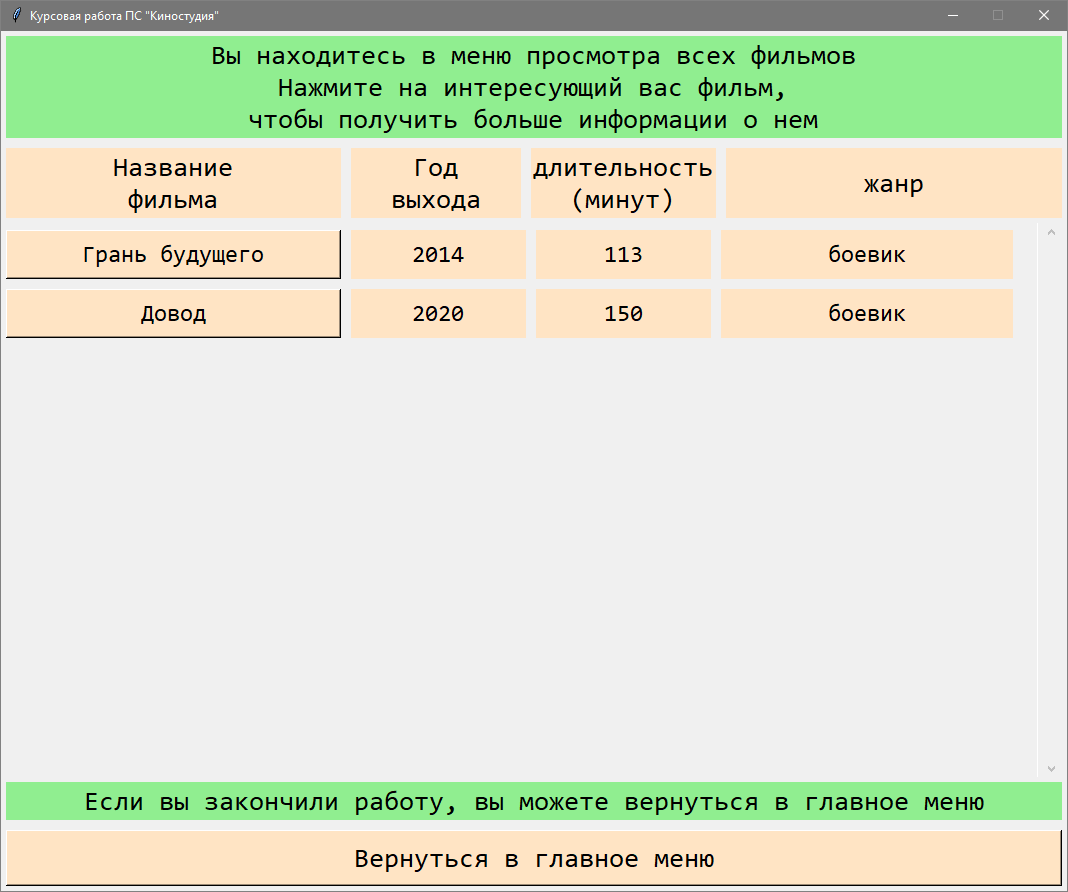


Рис. 48. Фильмы с жанром «боевик»

Вернемся обратно в меню фильмов (рис. 39) и выберем поиск фильма до/после года, окно, после выбора этого пункта представлено на рис. 49.

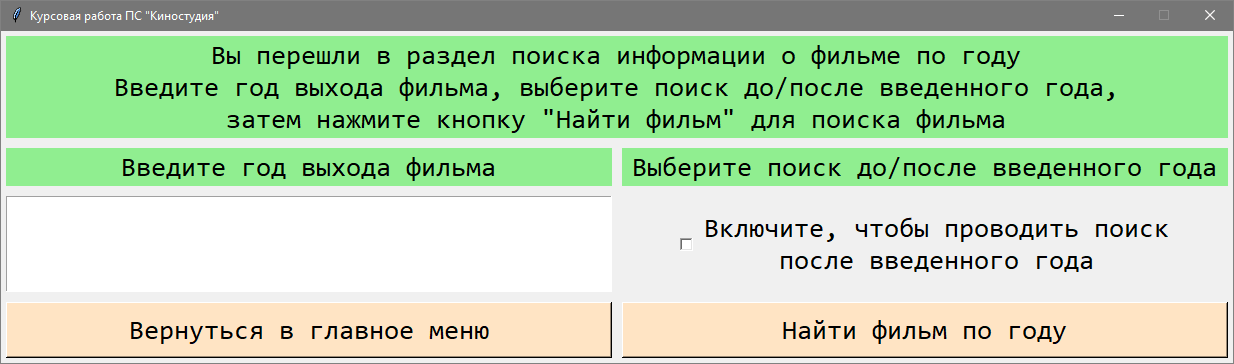


Рис. 49. Поиск фильма по году

В окне, изображенном на рис. 49 присутствует строка для ввода года и переключатель (флажок). В строку ввода нужно ввести года, который вас интересует, а флажок нужно перести во включенное состояние для того, чтобы поиск проводился по фильмам, имеющим год выпуска строго после введенного вами года, если переключатель не активирован, то поиск будет проводиться по фильмам, имеющим год выпуска раньше или равным введенному вами. Пример ввода для поиска по году приведен на рис. 50.

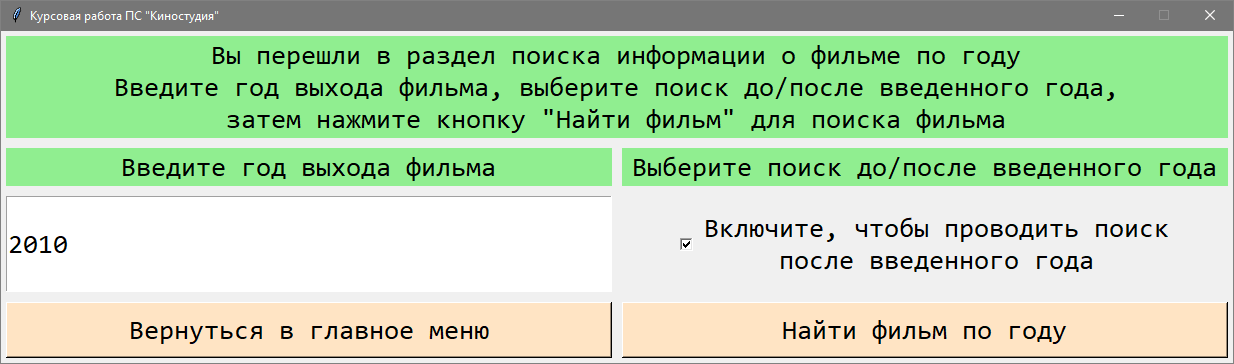


Рис. 50. Пример поиска фильма по году

Если фильмы с таким жанром будут найдены, то пользователь увидит сообщение (рис. 44), что по такому запросу были найдены фильмы, иначе получит сообщение (рис. 45) о том, что таких фильмов не нашлось. Если же фильмы нашлись, пользователь в окне, на рис. 51 увидит список фильмов, которые удовлетворяют его условиям.



Рис. 51. Результат поиска фильмов с годом выпуска после 2010

Вернемся обратно в меню фильмов (рис. 39) и выберем опцию «добавить фильм в ПС». После выбора этой опции, окно станет таким, как показано на рис. 52.

В окне добавления фильма в ПС вам потребуется добавить следующую информацию о фильме (см. рис. 52 – 53): название фильма, год выхода, продолжительность, жанр фильма, возрастной рейтинг, бюджет, сборы в США, сборы в мире (можно оставить путсым), награды, полученные фильмом (можно оставить пустым), а также вы можете указать актеров, которые снимались в этом фильме (см. рис. 53). Чтобы выбрать актеров, которые снимались в добавляемом фильме, необходмо прежде всего активировать на строке с данными актера переключатель (флажок), после чего в строке данных актера станет активной строка ввода, в нее вы можете написать, какого персонажа в добавляемом фильме играет каждый из актеров.

После того, как все данные заполенены, обязательно нажмите кнопку «Добавить фильм в базу данных»!

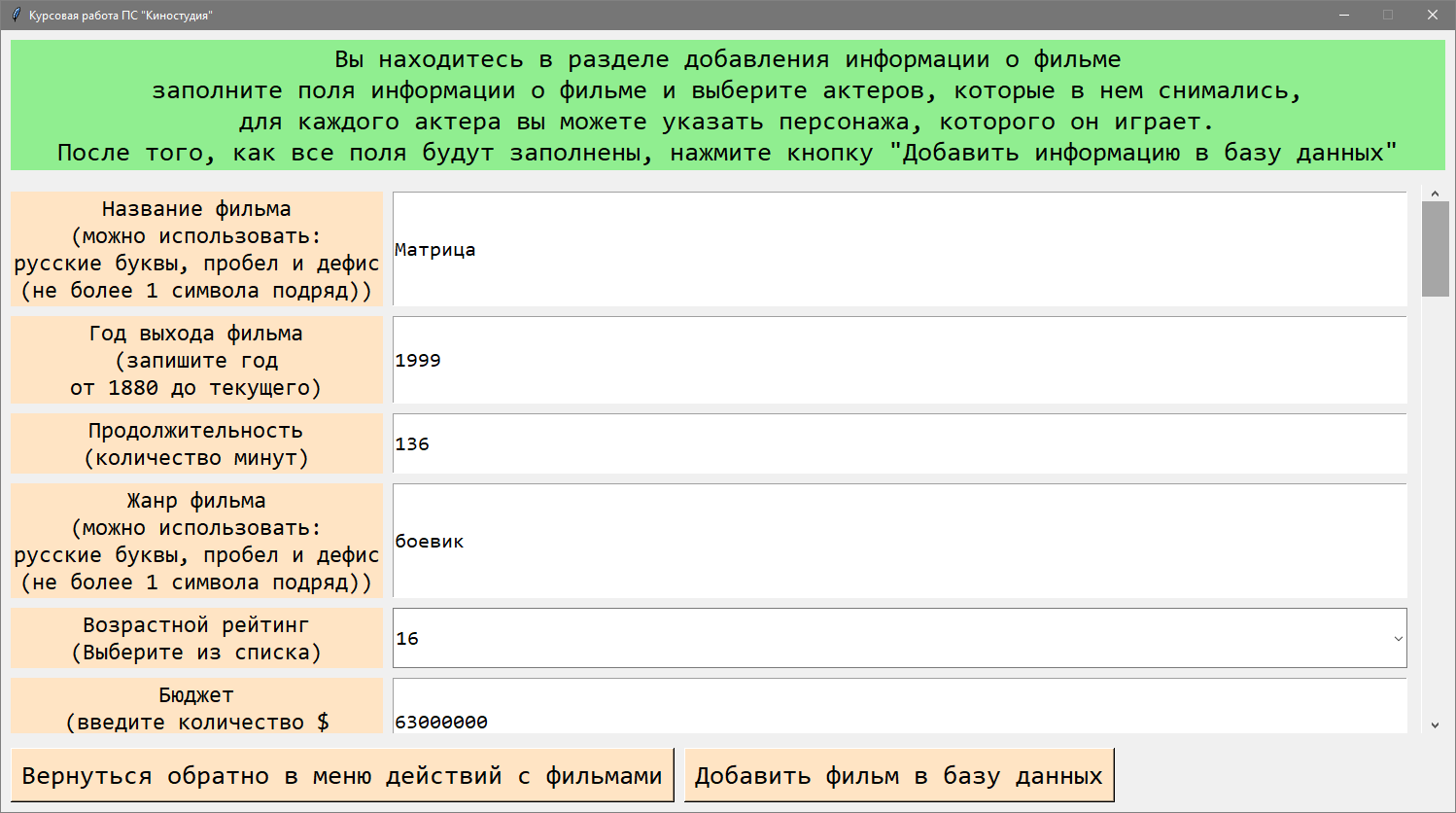


Рис. 52. Добавление фильма в ПС

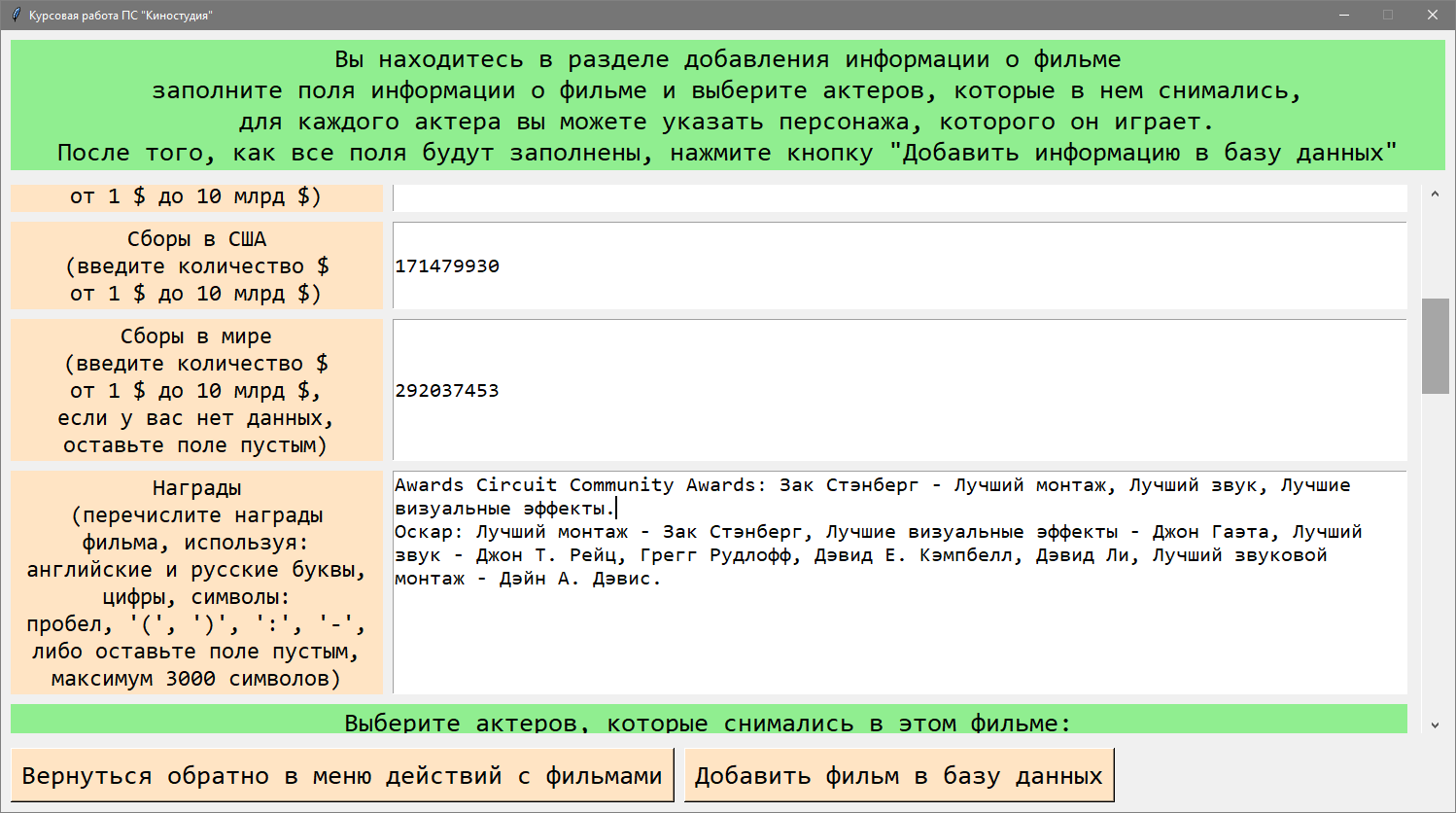


Рис. 53. Добавление фильма в ПС

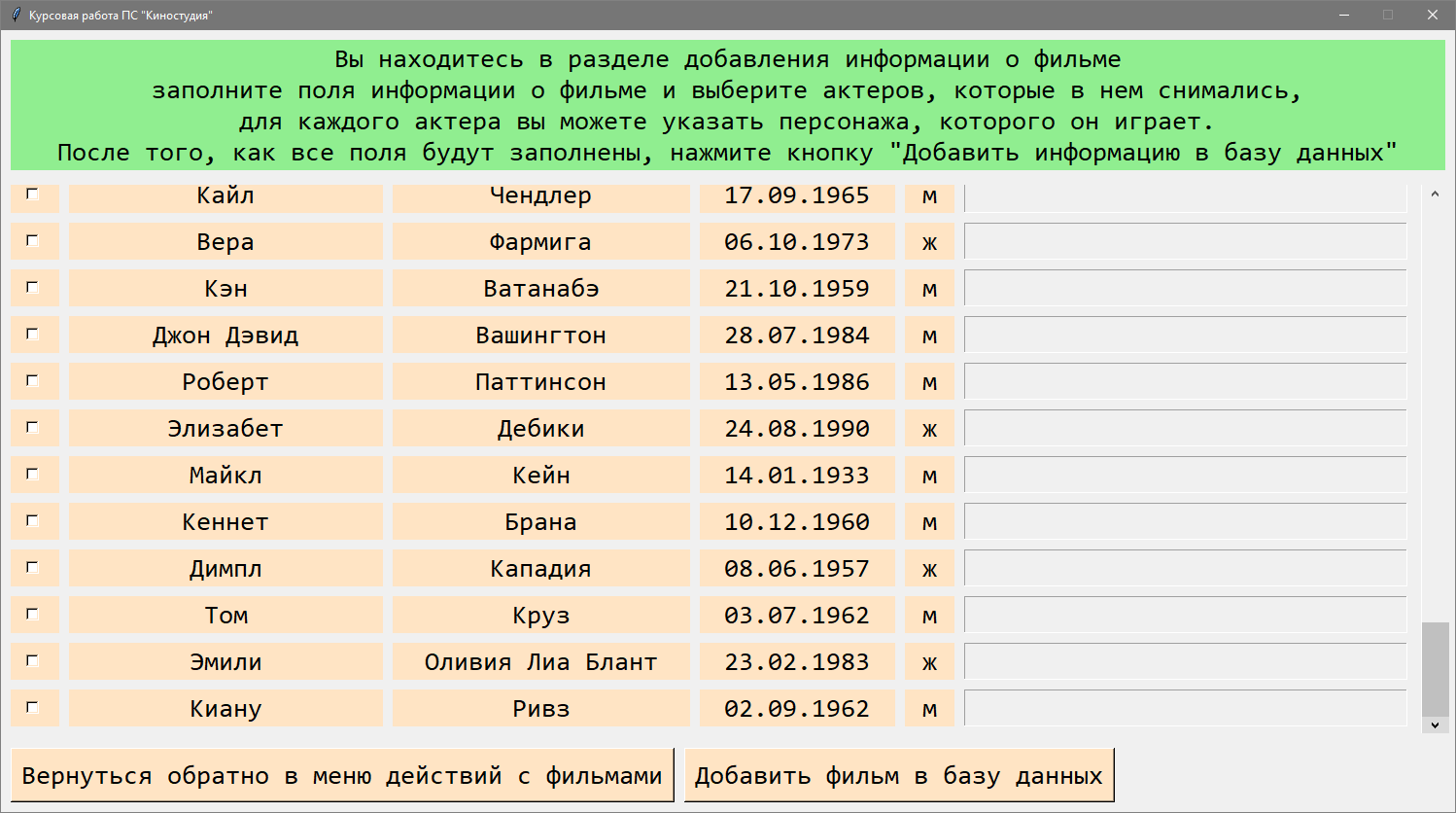
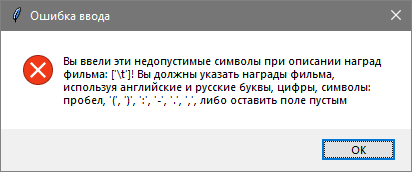


Рис. 53. Добавление фильма в ПС, выбор актеров

Если при вводе какие-то поля были заполнены неправильно, после нажатия кнопки «Добавить фильм в базу данных» вы получите сообщение об ошибке, например, как на рис. 54.



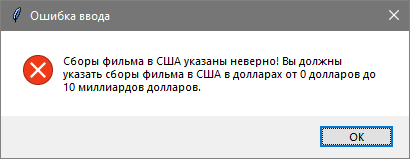


Рис. 54. Введены некорректные данные при добавлении фильма

ПС также проверяет ввод в поля ролей актеров для того, чтобы предатвратить некорректный ввод в них, если в поле роли актера введен некорректный символ, система сообщит об этом в окне, приведенном на рис. 55.

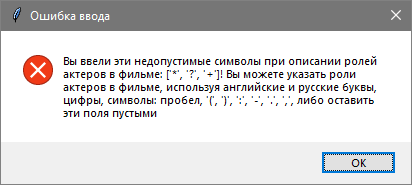


Рис. 55 введены запрещенные символы при описании роли актера

Если же при нажатии кнопки «Добавить фильм в базу данных» все данные введены корректно, то пользователь получит сообщение о том, что фильм успешно добавлен в базу данных ПС (рис. 56), однако, если фильм с таким названием уже существует в БД, будет сгенерировано сообщение о том, что такой фильм уже есть в ПС «Киностудия» (рис. 57).

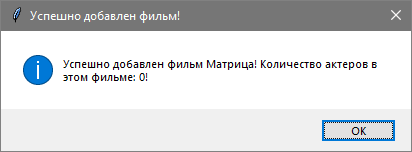


Рис. 56 Фильм успешно добавлен в ПС «Киностудия»

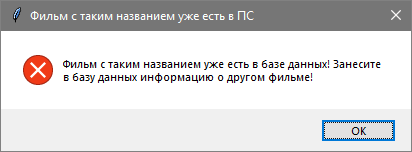


Рис. 57 Фильм с таким названием уже есть в базе данных ПС «Киностудия»

Вернемся обратно в меню действий с фильмами (рис. 39) и выберем опцию «Редактирование информации о фильмах», после этого откроется окно с выбором фильма для редактирования, как на рис. 58.



Рис. 58 окно выбора фильма для редактирования

В окне выбора фильма для редактирования пользователь может с любым фильмом из ПС «Киностудия» выполнить два действия: удалить фильм (нажав на символ удаления рядом с ним) и редактировать данные о фильме (нажав на карандаш).

В случае попытки удаления фильма, всплывет окно о попытке удаления фильма и система потребует подтверждения удаления фильма (см. рис. 59) в форме вопроса «да/нет», при выборе ответа «да» система удалит фильм и записи о ролях актеров в этом фильме из базы данных.

Нажмем кнопку с изображением карандаша рядом с фильмом «Матрица» для того, чтобы редактировать его данные. Внешний вид окна редактирования будет почти полностью совпадать с окном добавления фильма, однако в поля ввода информации уже будут вставлены данные выбранного ранее фильма, как на рис. 60.

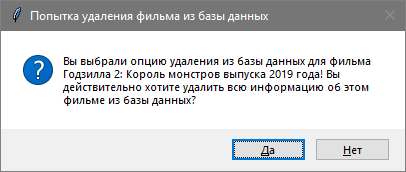


Рис. 59 окно с подтверждением удаления фильма из ПС

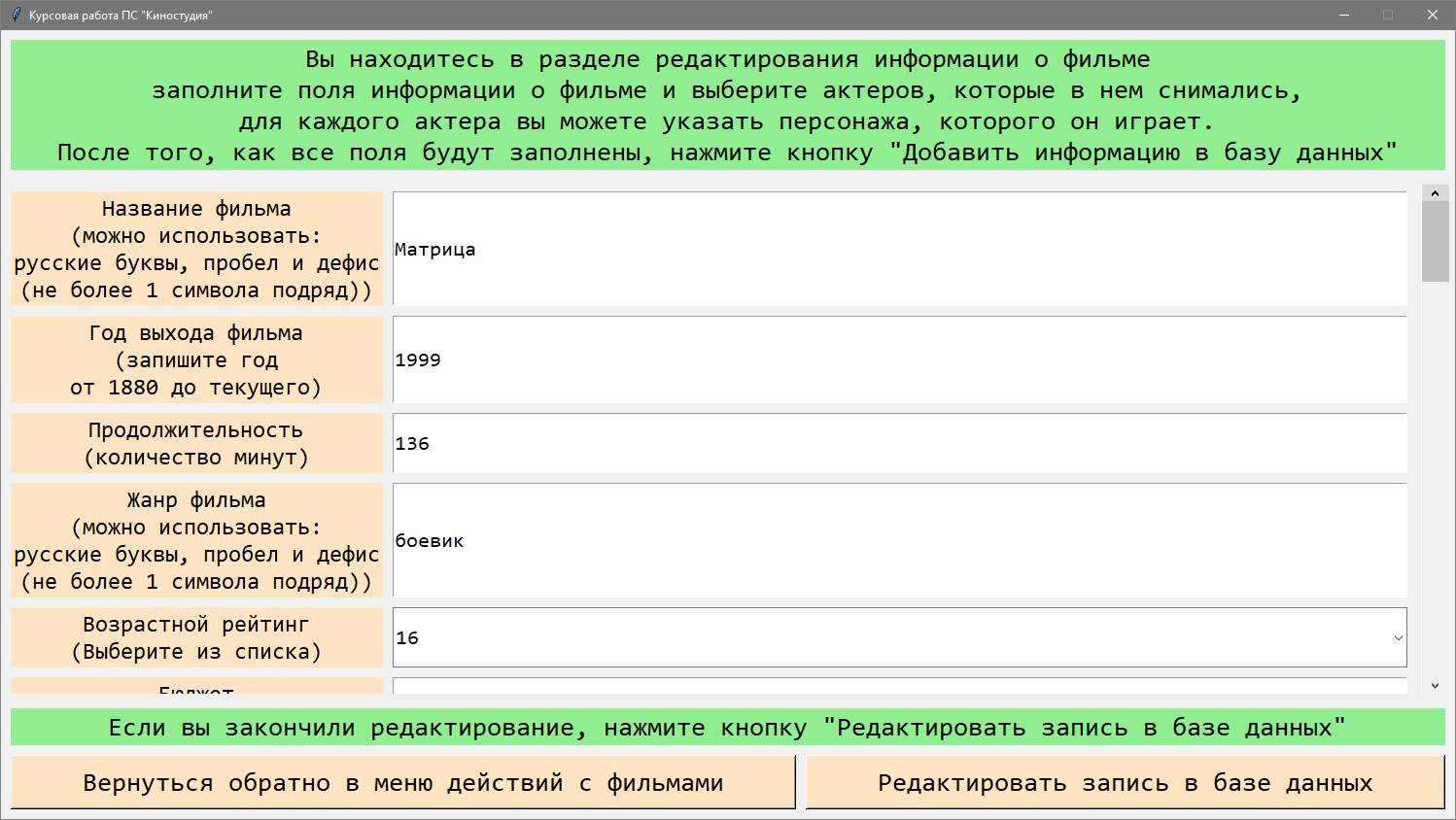


Рис. 60 Окно редактирования фильма

В окне редактирования информации о фильме вы можете изменить любую ранее внесенную информацию о фильме, и, в том числе, изменить роли актеров (рис. 61).



Рис. 61 Редактирование информации о фильме, редактирование ролей актеров

После того, как все необходимые правки будут сделаны, пользователь нажимает кнопку «Редактировать запись в базе данных», после чего ПС проверяет вводимые данные, выдает ошибки (рис 54, 55, 57) в случае некорректного ввода, и, если все проверки выполнены успешно, система выводит уведомление о том, что фильм отредактирован, как на рисунке 62.

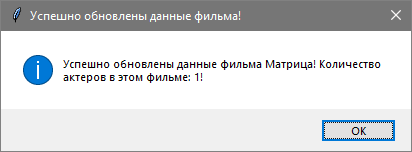


Рис. 62 Сообщение о том, что фильм успешно отредактирован

Итак, на этом меню фильмов полностью рассмотрено, перейдем обратно в главное меню, изображенное на рисунке 37, и выберем пункт «Информация об актерах». После выбора этого пункта будет выполнен переход в меню актеров (рис. 63).

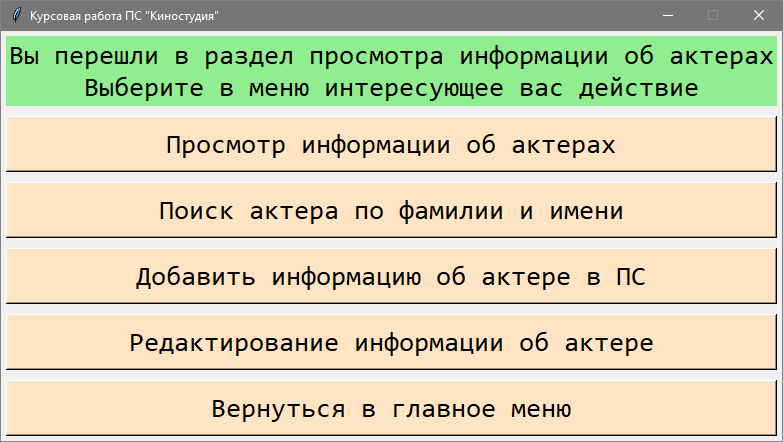


Рис. 63 Меню актеров

В меню актеров пользователь может просмотреть всех актеров, выполнить запрос поиска актера по имени и фамилии, добавить актера в ПС, отредактировать информацию об актере в ПС.

Перейдем в пункт просмотра информации об актерах (рис. 64).



Рис. 64 Окно просмотра всех актеров

В окне просмотра всех актеров пользователю представлен список всех актеров из ПС: их имена, фамилии, даты рождения и пол. В этом окне пользовтаель может открыть каждого актера и просмотреть фильмы, в которых этот актер снимался, как например на рис. 65.

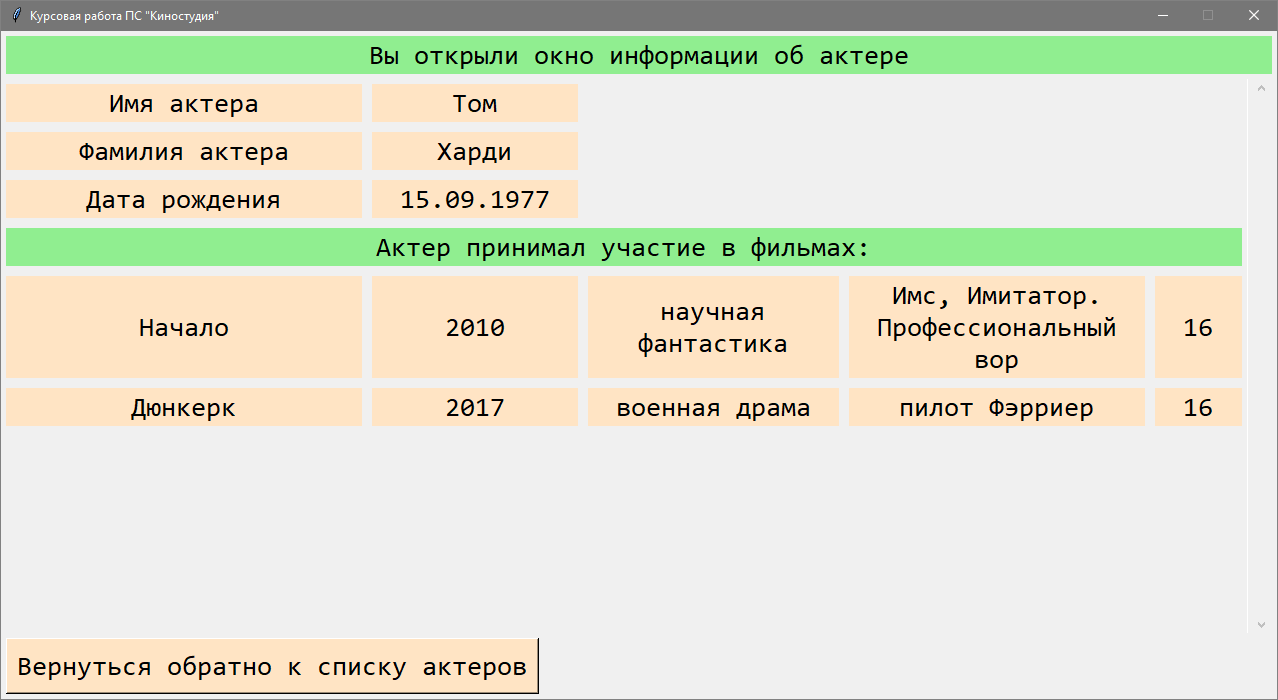


Рис. 65 Просмотр инфомрации об актере «Том Харди»

Вернемся обратно в меню актеров (рис. 63) и выберем пункт меню «поиск актера по фамилии и имени, откроется окно, приведенное на рис. 66.

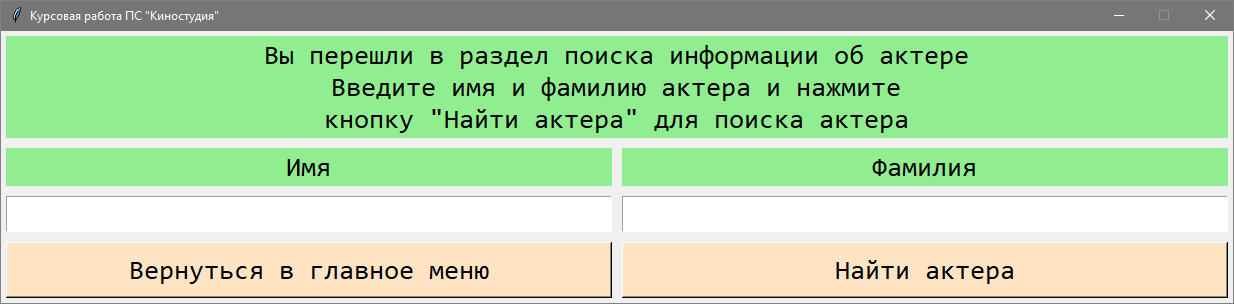


Рис. 66 Окно запроса поиска актера по имени и фамилии

В окне поиска актера пользователь в соответсвующие текстовые поля вписывает фамилию и имя актера, которого хочет найти, после чего нажимает кнопку «найти актера», после чего выдается окно с сообщением о том, что актер был найден (рис. 67), либо, что данных о таком актере нет в ПС (рис. 68). Пример выполнения поиска актера приведен на рис. 69 – 70.

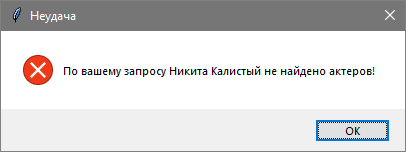


Рис. 67 По запросу не найдено актера в ПС «Киностудия»

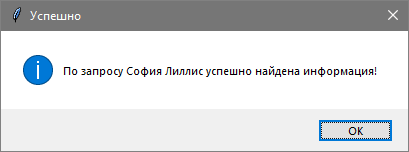


Рис. 68 Успешно найдена информация о запрашиваемом актере

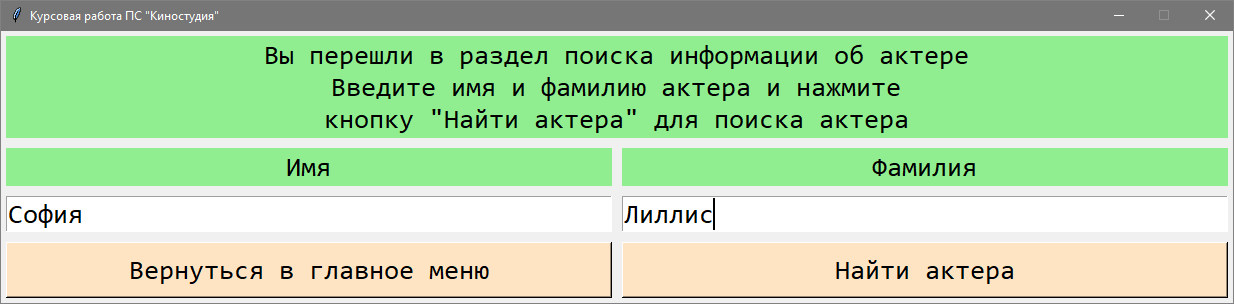


Рис. 69 Пример запроса поиска актера по имени и фамилии

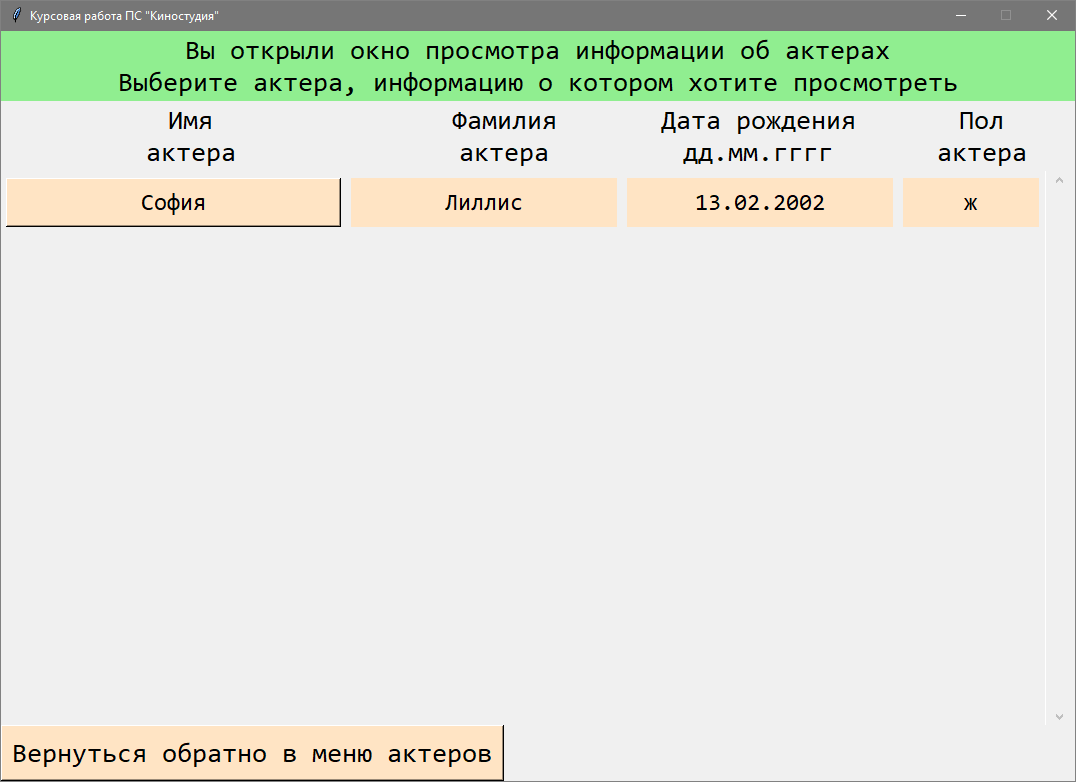


Рис. 70 Вывод актера по запросу по имени и фамилии

Вернемся обратно в меню актеров (рис. 63) и выберем опцию «Добавить информацию об актере в ПС, окно преобразуется в приведенное на рис. 71.

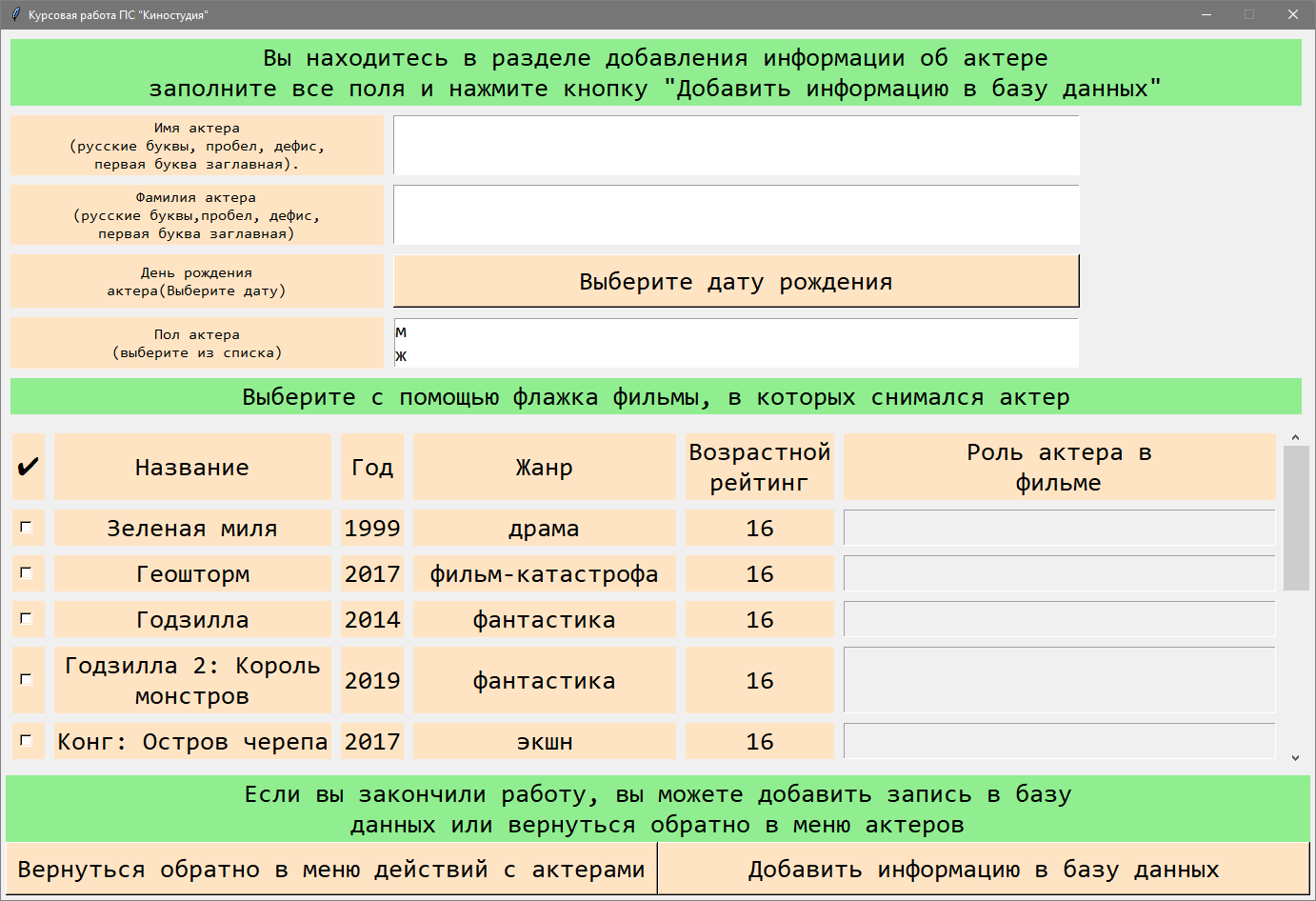


Рис. 71 Окно добавления актера в ПС Киностудия

В окне добавления информации об актере, пользователь должен ввести имя, фамилию, день рождения актера (нажав на кнопку «выберите дату рождения, после чего откроется окно с рисунка 72, в котором нужно будет выбрать дату рождения) и пол актера. Также здесь можно выбрать фильмы и написать роли, которые играл актер в этих фильмах.

Для того, чтобы выбрать фильм для актера, нужно нажать на переключатель (флажок) рядом с фильмом, а затем вписать роль, которую играет добавляемый актер в этом фильме.

Пример добавления актера приведен на рис. 73-74.

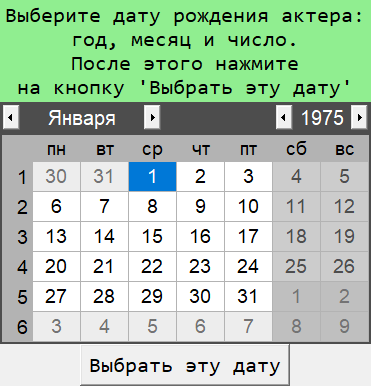


Рис. 72 окно выбора даты рождения актера

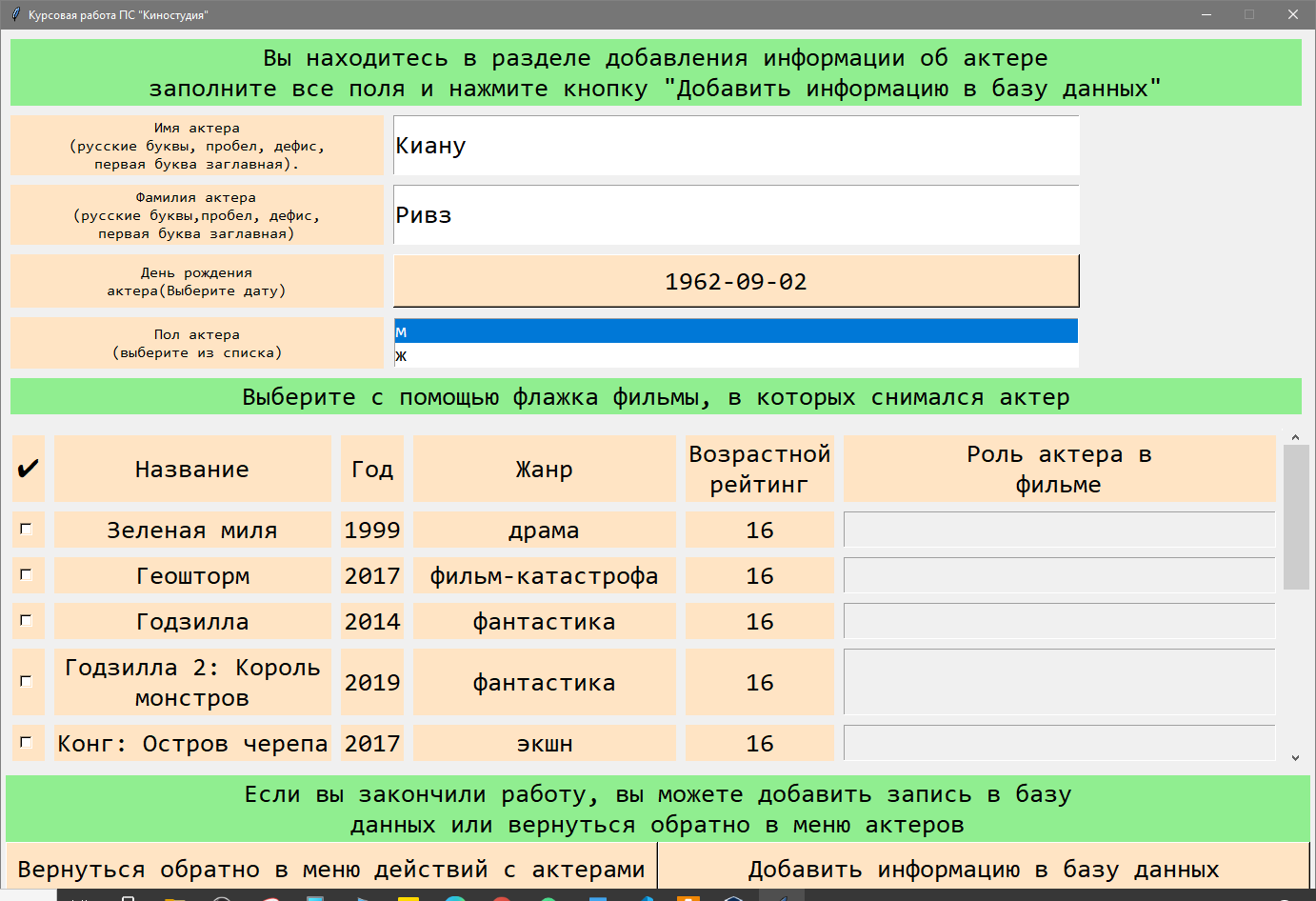


Рис. 73 Пример добавления актера, ввод основных данных

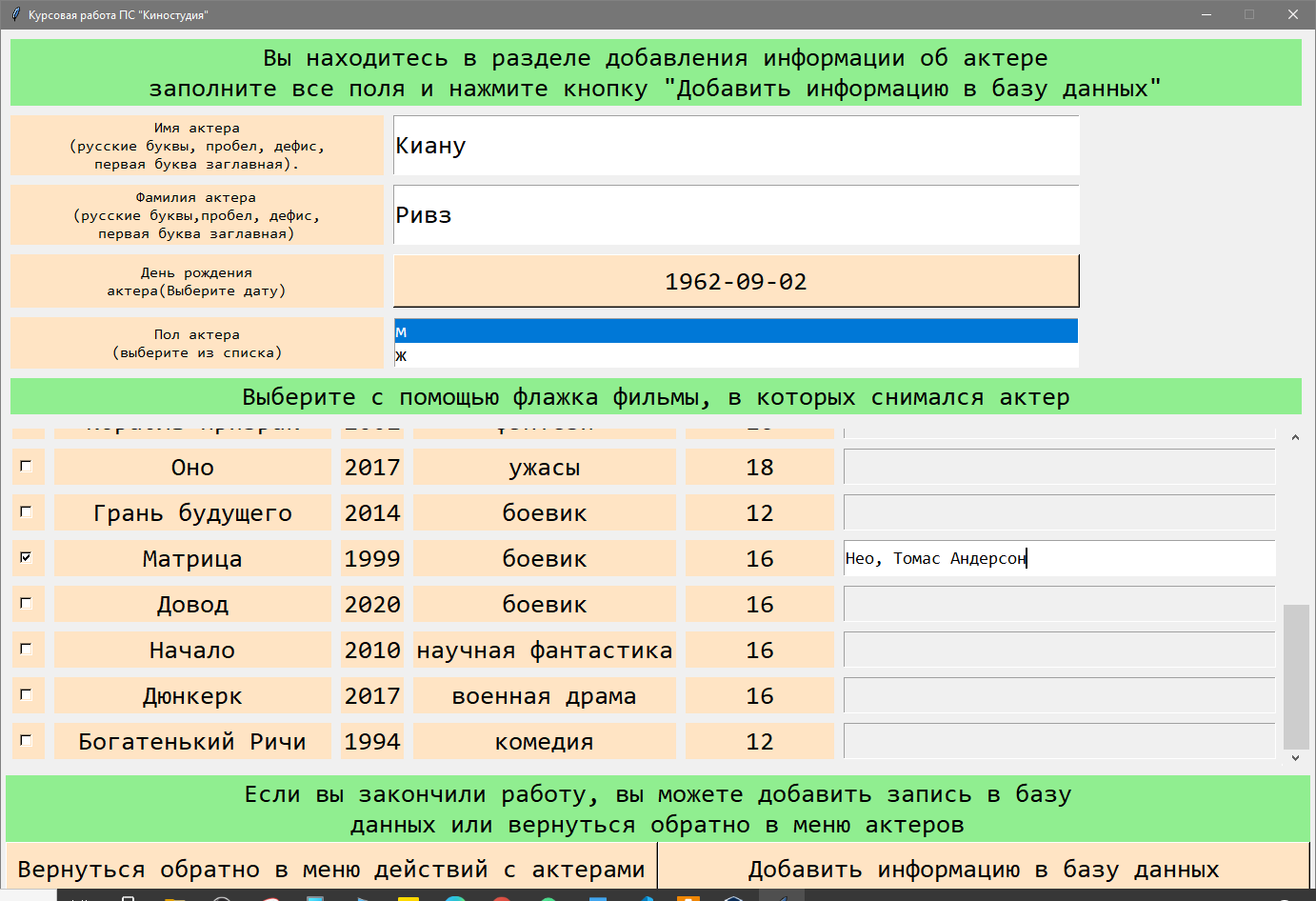


Рис. 74 Пример добавления актера, выбор фильмов и ролей для актера

ПС «Киностудия» проверяет все вводимые данные при добавлении актера, поэтому в случае записи некорректных данных, после нажатия кнопки «добавить информацию в базу данных» пользователь увидит окно с сообщением об ошибке (рис. 75), или, если в описании ролей будут содержаться некорректные символы (рис. 76), если же все данные корректны, то после нажатия кнопки добавления, пользователь увидит сообщение о добавлении актера в ПС (рис. 77).

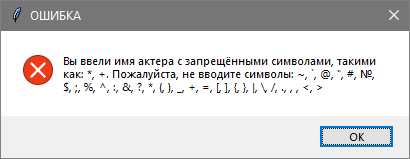


Рис. 75 Введены некорректные данные при вводе данных актера

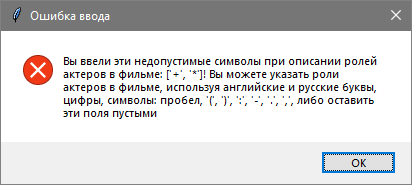


Рис. 76 Введены некорректные данные при описании ролей актера

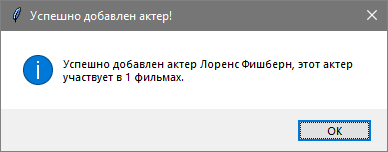


Рис. 77 Актер успешно добавлен в ПС «Киностудия»

После добавления актера в ПС «Киностудия», вернемся обратно в меню актеров (рис 63) и выберем пункт меню «Редактирование информации об актере», после чего на экран будет выведен список всех актерев и рядом с каждым из них появится по две кнопки, кнопка с карандашом нужно для редактирования данных актера, кнопка с символом удаления удаляет актера из базы данных.

При нажатии на символ удаления, ПС запросит подтверждение удаления актера из фильма (рис. 78), и в случае ответа «Да», все данные актера и записи о его ролях в фильмах будут удалены.

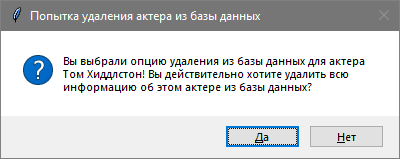


Рис. 78 Окно подтверждения удаления актера из ПС «Киностудия»

При нажатии на кнопку с символом карандаша, будет выполнен переход в окно редактирования актера, оно будет похоже на окно добавления актера, но в поля данных уже будут вставлены имеющиеся данные об актере. Пример окна редактирования приведен на рис. 79.

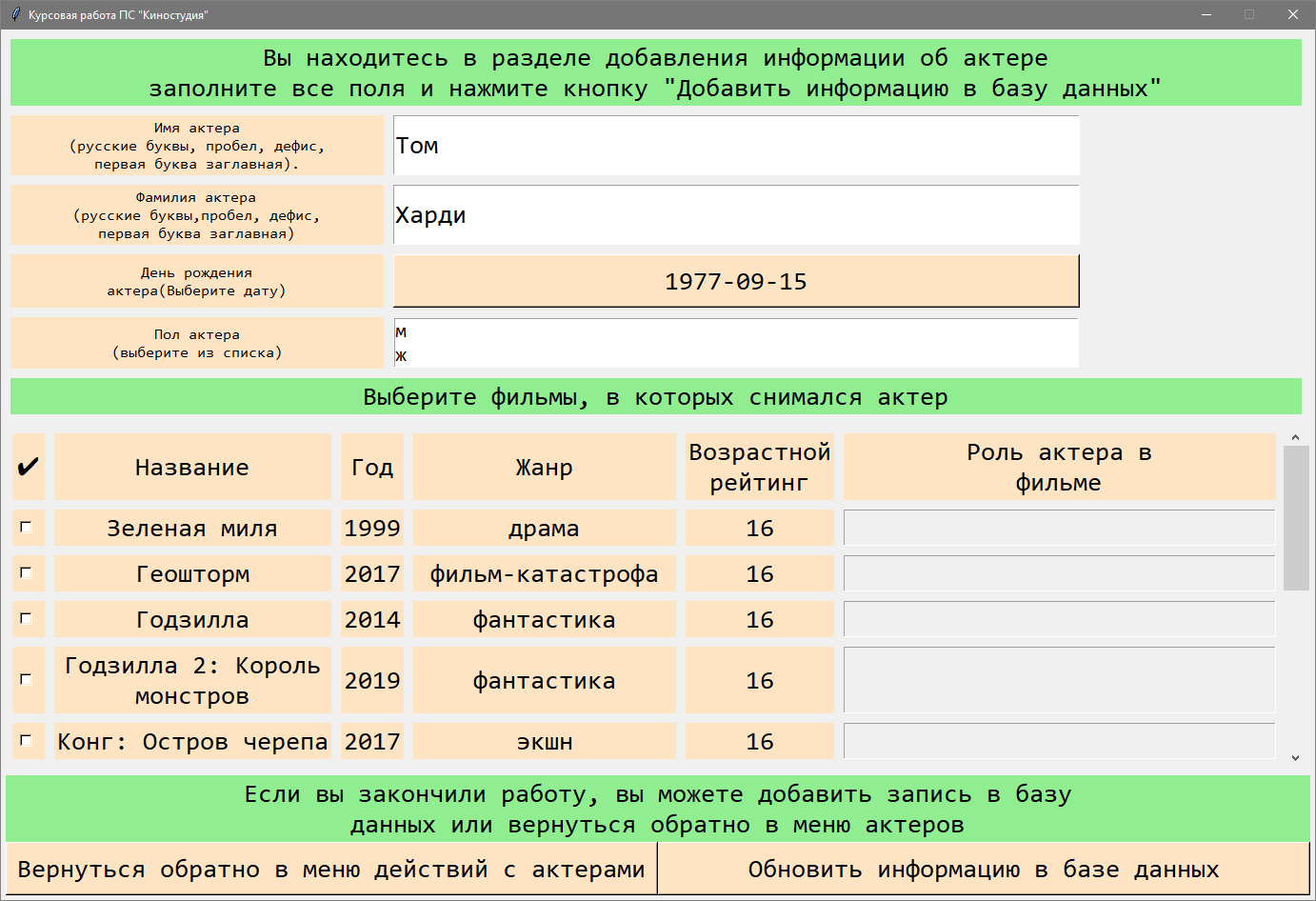


Рис. 79 окно редактирования информации об актере «Том Харди»

В окне редактирования информации об актере можно изменить все данные об актере, а также изменить выбор фильмов и роли, которые играл актер в этих фильмах (рис. 80).

В окне редактирования все вводимые данные также проверяются ПС «Киностудия», в случае ввода некорректных данных, пользователь будет получать сообщения об ошибках (рис. 75, 76).

После того, как редактирование информации завершено, пользователь должен нажать кнопку «обновить инормацию в базе данных, и выплывет окно, которое сообщит о том, что редактирование информации выполнено успешно, пример на рис. 81.



Рис. 80 Редактирование ролей актера «Том Харди»

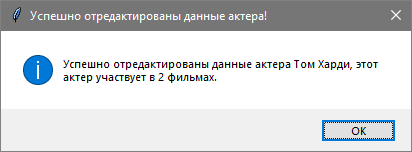


Рис. 81 Окно с сообщением о том, что данные актера успешно отредактированы

На редактировании информации об актере рассмотрение меню актеров завершено, и в главном меню (рис. 37) осталась нерассмотренной только опция «Вход в режим администратора».

После выбора этой опции, будет осуществлен переход в меню авторизации. Если пользователь еще не авторизован как администратор, окно авторизации будет выглядеть, как представлено на рис. 82, если уже администратор, то как на рис. 83.

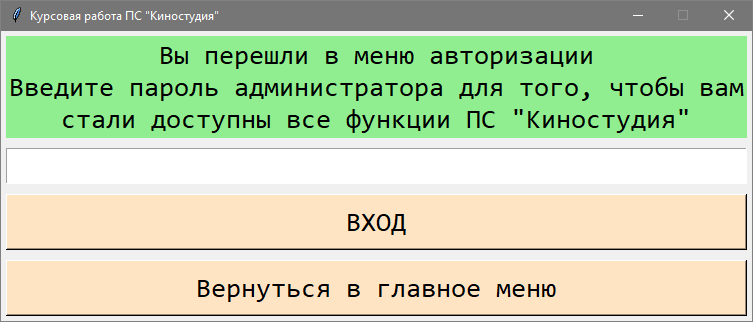


Рис. 82 Меню авторизации неавторизованного как администратор пользователя

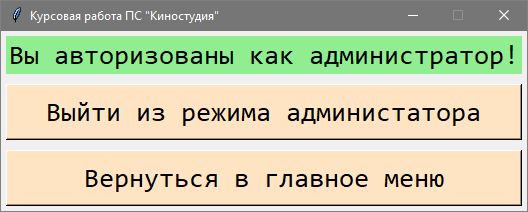


Рис. 83 Меню авторизации авторизаванного как администратор пользователя

Если пароль администратора устарел, т.е. прошло 10 дней со дня его установки, то система выдаст предупреждение о том, что для продолжения работы его нужно сменить (рис. 84). Также окно авторизации изменится, и в нем появится поле для ввода нового пароля (рис. 86).

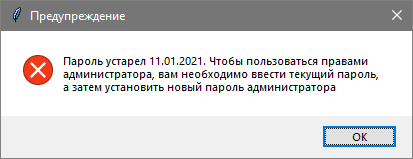


Рис. 84 сообщение об устаревшем пароле

Если сообщение об устаревшем пароле было выведено на экран, пользователь все равно должен ввести пароль администратора и нажать кнопку «Вход», после чего он получит повторное сообщение о том, что пароль устарел (рис. 85). Внимание! В этот момент пользователь еще не имеет прав администратора! Пользователь должен установить новый пароль на ПС «Киностудия».

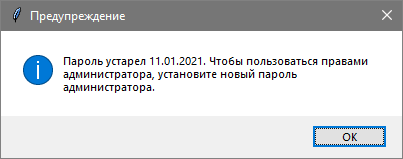


Рис. 85 Повторное сообщение о том, что пароль устарел

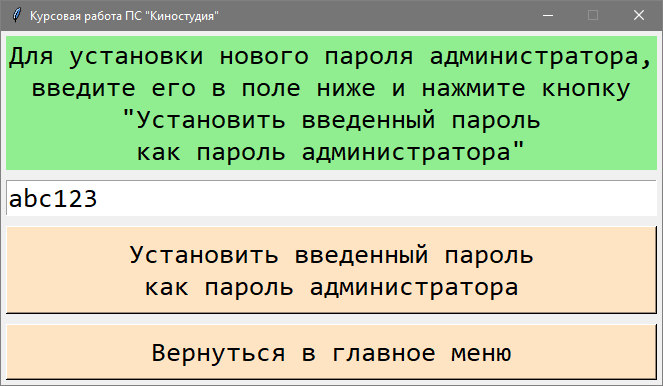


Рис. 86 Окно авторизации в режиме смены пароля

После успешного ввода нового пароля, он будет сохранен в ПС «Киностудия» и активен 10 дней, после чего его снова потребуется сменить. После этого вы будете возвращены в главное меню и будете обладать правами администратора.

На этом рассмотрение возможностей ПС «Киностудия» завершено.

## Вывод

В ходе выполнения курсовой работы было сформулировано и утверждено техническое задание на разработку ПС «Киностудия». За время разработки был разработан алгоритм решения задачи, было найдено решение множества различных задач, составлена и отлажена программа, была сформирована, налажена и заполнена исходными данными база данных ПС, оформлена документация на программу. Проведенные испытания ПС показали, что программа все требования технического задания выполняет в полном объеме. Результаты испытаний свидетельствуют о правильности работы программы. Ошибочные действия пользователя не приведут к нарушению работоспособности программы, утрате целостности базы данных. За время работы над ПС был получен большой опыт взаимодействия с базой данных, создания запросов.

# Список литературы

1. «ВикипедиЯ свободная энциклопедия» [электронный ресурс]. Режим доступа – свободный. URL: <https://ru.wikipedia.org/wiki/Python> (Дата обращения 25.12.2020)
2. Описание библиотеки «tkcalendar» [электронный ресурс]. Режим доступа – свободный. URL: <https://pypi.org/project/tkcalendar/>
3. «Работаем с SQL Server с помощью Python» [электронный ресурс]. Режим доступа – свободный. URL:  [https://proglib.io/p/rabotaem-s-sql-server-s-pomoshchyu-python-2020-04-18](https://pythonru.com/osnovy/sqlite-v-python%20%20)  (Дата обращения 20.12.2020)
4. Базы данных в Python [электронный ресурс]. Режим доступа – свободный. URL: <https://python-scripts.com/database>