МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Інтелектуальних комп’ютерних систем»

КУРСОВА РОБОТА

Тема: «Створення бази відеофільмів»

Керівник роботи

ас. каф. ІКС

Кочуєва З.А

Виконавець

студентка групи ІФ-87

Леонт’єва А.В

Харків – 2009р.

ЗАВДАННЯ

на науково-дослідну курсову роботу

Тема: «Створення бази відеофільмів»

Короткий зміст роботи:

а) реферативна частина

Огляд літературних джерел з існуючих методів та алгоритмів рішення задачі створення бази відеофільмів

б) теоретична частина

Опис основних принципів об'єктно-орієнтованого програмування, основ алгоритмізації, типових середовищ програмування та особливостей мови програмування С++, розгляд основних методів рішення даної задачі та їх алгоритмів, опис обраного методу розв’язання задачі та його алгоритму експериментальна частина

Розробка програмного забеспечення рішення задачі створення бази відеофільмів

Дата видачі завдання: 5.10.08 Термін захисту: 30.12.08

Керівник курсової роботи: ас. каф. ІКС Кочуєва З. А.

РЕФЕРАТ

КР: 53 с., 10 рис., 7 источников, 2 приложения

# Данная курсовая работа посвящена реализации создания базы видеофильмов. Программная реализация осуществлена на языке программирования C++ в среде C++Builder 6.

# Курсовая работа состоит из введения, трех частей, заключения, списка литературы, содержащего 7 источников. Содержит 53 страницы, 10 рисунков, 2 приложения, введение, заключение. Во введении описывается современное состояние развития объектно-ориентированного программирования и предпосылки создания баз данных. В первой части описаны основные принципы объектно-ориентированного программирования, типичные среды программирования и особенности языка С++, основы алгоритмизации. Во второй части рассмотрен основной метод решения данной задачи. В третьей части приводится описание программы и результаты, получаемые пользователем при использовании данной программы. В заключении подводится итог проделанной работы и дается критический анализ полученных результатов. Приложение содержит текст программы на языке C++, который реализует создание базы видеофильмов.

Ключевые слова и выражения:

АЛГОРИТМ, C++ BUILDER6, БАЗА ДАННЫХ, СТРУКТУРЫ ДАННЫХ, ТИП, ФУНКЦИЯ, СТРУКТУРА.

РЕФЕРАТ

КР: 53 с., 10 мал., 7 джерел, 2 додатки

# Дана курсова робота присвячена реалізації створенння бази відеофільмів. Програмна реалізація здійснена на мові програмування C++ в середовищі C++Builder 6.

# Курсова робота складається з введення, трьох частин, висновку, списку літератури, що містить 7 джерел. Містить 53 сторінки, 10 малюнків, 2 додатки, введення, висновок. У введенні описується сучасний стан розвитку об’єктно-орієнтованого програмування та передумови створення баз даних. У першій частині описані основні принципи об'єктно-орієнтованого програмування, типові середовища програмування і особливості мови С++, основи алгоритмізації. У другій частині наведе основний метод рішення даної задачі. У третій частині приводиться опис програми і результати, одержані користувачем при використанні даної програми. У висновку підводиться підсумок виконаної роботи і дається критичний аналіз одержаних результатів. Додаток містить текст програми на мові C++, який реалізує створення бази відеофільмів.

Ключові слова і вирази

АЛГОРИТМ, C++ BUILDER6, БАЗА ДАНИХ, СТРУКТУРА ДАНИХ, МЕТОД, ТИП, ФУНКЦІЯ, СТРУКТУРА.

ABSTRACT

TP: 53 р., 10 pic., 7 sources, 2 appendixes

# This term paper is devoted to realization of making the base of movies. Programmatic realization is carried out in programming C++ language in the environment C++Builder 6.

# The term paper consists of introduction, three parts, conclusion, list of literature, containing 7 sources. Contains 53 pages, 10 pictures, 2 appendixes, introduction, conclusion. Modern development status of the object-oriented programming and pre-condition of creation of databases is described in introduction. Basic principles of the object-oriented programming, typical programming environments and features of the language С++, basis of algorithmisation, are described in the first part. The basic method of decision of this task is considered in the second part. In the third part is brought description over of the program and results got an user at the use of this program. In conclusion is worked out the total work do and the critical analysis of the got results is given. The appendix contsins a program text in language C++, which will realize making the base of movies.

Keywords and expressions:

ALGORITHM, C++ BUILDER6, DATABASE, DATA STRUCTURES, TYPE, FUNCTION, STRUCTURE.

ЗМІСТ

Вступ

1 Використання сучасних систем і мов програмування

1.1 Основні принципи об’єктно-орієнтованого програмування

1.2 Типові середовища програмування та особливості мови С++

1.2.1 Borland C++ Builder

1.2.2 Microsoft Visual C++

1.2.3 Мова програмування С++

1.3 Основи алгоритмізації

2 Розробка рішення задачі створення бази відеофілмів

2.1 Бази даних

2.1.1 Основні положення та моделі БД

2.1.2 Типи БД

2.1.3 Етапи проектування БД

2.1.4 Використання БД

2.2 Структури даних

2.3 Функції

3 Розробка програмного забезпечення для реалізації створення бази відеофільмів

3.1 Опис програмної реалізації бази відеофільмів

3.2 Результати, отримані користувачем при використанні програмного забезпечення

Висновок

Список літератури

Додаток А

Додаток Б

ВСТУП

Якісним кроком в розвитку методів структурного програмування став винахід об'єктно-орієнтованого програмування (мов SmallTalk, C++, Turbo Pascal і ін.). Програми стали будуватися з порівняно простих об'єктів, що складаються з даних і підпрограм їх обробки. Гнучкість об'єктів дозволила дуже просто пристосовувати їх для власних цілей, докладаючи для цього мінімум зусиль. Програмісти обзавелися готовими бібліотеками об'єктів, але, як і раніше, створення призначеного для користувача інтерфейсу вимагало силу-силенна часу і сил, особливо коли програма повинна була працювати під управлінням популярної операційної системи Windows і мати графічний призначений для користувача інтерфейс.

Знання про світ розвиваються з неймовірною швидкістю. Причиною тому є науково-технічний прогрес, заснований на полегшенні роботи та життєдіяльності людини та пошуку методів для розв’язання різноманітних задач. Прогрес базується на дослідженні, а будь-яке дослідження, в свою чергу, базується на використанні доступних даних. І у кожного фахівця існує свій, персональний "архів" довідкової інформації, який після машинної обробки та систематизації має вигляд бази даних.

Створення бази даних передбачає рішення цілого комплексу проблем, в тому числі розробку програмних, мовних, організаційних і технічних засобів, призначених для централізованого зберігання і колективного використання інформації.

1 ВИКОРИСТАННЯ СУЧАСНИХ СИСТЕМ І МОВ ПРОГРАМУВАННЯ

1.1 Основні принципи об’єктно-орієнтованого програмування

Якісним кроком в розвитку методів структурного програмування став винахід об'єктно-орієнтованого програмування. Об'єктне і об'єктно-орієнтоване програмування (ООП) виникло внаслідок розвитку ідеології процедурного програмування, де дані і підпрограми (процедури, функції) їх обробки формально не зв'язані. Крім того, в сучасному об'єктно-орієнтованому програмуванні часто велике значення мають поняття події (так зване подієво-орієнтоване програмування) і компоненту (компонентне програмування).

Першою мовою програмування, в якій були запропоновані принципи об'єктної орієнтованості, була Симула. У момент своєї появи (у 1967 році), ця мова програмування запропонувала справді революційні ідеї: об'єкти, класи, віртуальні методи і ін., проте це все не було сприйнято сучасниками як щось грандіозне. Однак, більшість концепцій була розвинена Аланом Кейем і Деном Інгаллсом в мові Smalltalk. Саме вона стала першою широко поширеною об'єктно-орієнтованою мовою програмування.

С++ додає до С об'єктно-орієнтовані можливості. Він запроваджує класи, які забезпечують три найважливіші властивості ООП:

1. Інкапсуляція;

2. Наслідування;

3. Поліморфізм.

Додаткові властивості корисні, та не обов'язкові.

Розглянемо сі ці властивості детальніше.

Інкапсуляція

Основним способом організації інформації в С++ є класи. На відміну від типу структура (struct) мови С, яка може складатися лише з полів і вкладених типів, клас (class) С++ може складатися з полів, вкладених типів і функцій-членів (member functions). Члени класу бувають публічними (відкритими, public), захищеними (protected) і власними (закритими, приватними, private). В С++ тип структура аналогічний типу клас, відмінність в тому, що за умовчанням члени і базові класи в структурі публічні, а в класі - власні.

З відкритими (публічними) членами класу можна робити зовні класу все, що завгодно. До закритих (приватних) членів не можна звертатися ззовні класу, щоб не порушити цілісність даних класу. Спроба такого звертання спричине помилку компіляції. До таких членів можуть звертвтися тільки функції-члени класу. Крім відкритих і закритих членів класу, можуть бути ще і захищені - це члени, доступні класу, що містить їх, його друзям, а також похідним від нього класам. Такий захист членів називається інкапсуляцією.

Використовуючи інкапсуляцію, автор класу може захистити свої дані від некоректного використання. Крім того, вона замислювалася для полегшення спільної розробки класів. Малося на увазі, що при зміні способу зберігання даних, якщо вони оголошені як захищені або власні, не вимагається відповідних змін в класах, які використовують змінений клас. Навіть оператор доступу operator [] може бути визначений як така стандартна функція.

Наслідування

Для створення класів з доданою функціональністю вводять наслвдування. Клас-наслідувач має поля і функції-члени базового класу, але не має права звертатися до власних (private) полів і функцій базового класу. У цьому і полягає різниця між власними і захищеними членами.

Клас-наслідувач може додавати свої поля і функції або перевизначати функції базового класу.

За умовчанням, конструктор спадкоємця без параметрів викликає конструктор базового класу, а потім конструктори нестатичних членів-даних, що є екземплярами класів. Деструкція працює в зворотном порядку. Інші конструктори доводиться визначати кожного раз наново. На щастя, це можна зробити викликом конструктора базового класу.

Наслідувач – це більш ніж базовий клас, тому, якщо спадкоємств відкрите, то він може використовуватися скрізь, де використовується базовий клас, але не навпаки.

Наслідування буває публічним, захищеним і власним. При публічному спадкоємстві, публічні й захищені члени базового класу зберігають свій статус, а до власних не можуть звертатися навіть функції-члени спадкоємця. Захищене наслідування відрізняється тим, що при ньому публічні члени базового класу є захищеними членами спадкоємця. При власному спадкоємстві всі члени базового класу стають власними членами класу-спадкоємця. Таким чином, користувач похідного класу не може звертатися до членів базового класу, навіть якщо вони оголошені я публічні. Клас-наслідувач робить їх власними за допомогою власного спадкоємства. Як правило, публічне наслідування зустрічається значно частіше за інших.

Клас може бути спадкоємцем декількох класів. Це називається множинним наслідуванням. Такий клас володіє полями і функціями-членами всіх його предків.

Поліморфізм

Поліморфізмом в програмуванні називається перевизначення снаслідувачем функцій-членів базового класу.

У C++ є як статичний, так і динамічний поліморфізм, коли функція, що викликається, визначається під час виконання. Для цього функції-члени повинні бути віртуальними.

Чисто віртуальною функцією називається функція-член, яка оголошена зі специфікатором = 0:

Чисто віртуальна функція може бути залишена без визначення, крім випадку, коли вимагається зробити її виклик. Абстрактним класом називається такій, у якого є хоча б одна чисто віртуальна функція-член. Об'єкти таких класів створювати заборонено. Абстрактні класи часто використовуються як інтерфейси.

1.2 Типові середовища програмування та особливості мови С++

1.2.1 Borland C++ Buіlder

Система програмування Borland C++ Buіlder об'єднала в собі ідеї інтегрованого середовища розробки, реалізовані компанією в системах програмування Turbo Pascal й Borland Delphі з можливостями мови програмування C++. Історія цієї системи програмування починається з інтегрованого середовища розробки Borland Turbo.

Середовище Turbo являла собою реалізацію ідей, закладених компанією-розроблювачем у системі програмування Turbo Pascal для мови програмування C. Компанія Borland прагнула перенести вдалу реалізацію ідей інтегрованого середовища розробки на нову основу. Компілятор Turbo не був однопрохідним, і тому час компіляції вихідної програми перевищувало час компіляції аналогічної програми в Turbo Pascal. Крім того, у системі програмування використався стандартний компоновщик файлів, що виконують, MSDOS.

Сучасна реалізація Borland C++ Buіlder орієнтована на розробку результуючих програм, що виконуються під керуванням ОС Mіcrosoft Wіndows всіх типів. Сама система програмування Borland C++ Buіlder, як й Borland Delphі, також функціонує під керуванням ОС типу Mіcrosoft Wіndows. Вона повністю підтримує стандарт мови C, що уможливлює створення за допомогою даної системи програмування модулів і бібліотек, використовуваних в інших засобах розробки (чого дуже складно досягти за допомогою Borland Delphі).

За можливостями, зовнішнім виглядом й технологіями система програмування Borland C++ Buіlder схожа з системою програмування Borland Delphі. Вее основу покладені ті ж основні ідеї й технології. Структура класів мови C++ у системі програмування Borland C++ Buіlder побудована в тій же бібліотеці VCL (vіsual control lіbrary), у якій будується структура класів Object Pascal у системі програмування Borland Delphі. Правда, розроблювач, що створює програми на C++, може не користуватися класами VCL і взяти за основу будь-яку іншу бібліотеку, чого не можна сказати про розроблювача, що використає Object Pascal, набір доступних бібліотек для останньої мови сильно обмежений.

Успішне поширення систем програмування Turbo Pascal й Borland Delphі сприяло й впровадженню на ринок системи програмування Borland C++ Buіlder від тієї ж компанії-розроблювача. Ця система програмування займає міцну позицію на ринку засобів розробки для мови C++, де існує досить тверда конкуренція.

1.2.2 Mіcrosoft Vіsual C++

Система програмування Mіcrosoft Vіsual C++ являє собою реалізацію середовища розробки для розповсюдженої мови системного програмування C++, виконану компанією Mіcrosoft. Ця система програмування в цей час побудована у вигляді інтегрованого середовища розробки, що включає в себе всі необхідні засоби для розробки результуючих програм, орієнтованих на виконання під керуванням ОС типу Mіcrosoft Wіndows різних версій.

Основу системи програмування Mіcrosoft Vіsual C++ становить бібліотека класів MFC (Mіcrosoft foundatіon classes). Вэтой бібліотеці реалізовані у вигляді класів C++ всі основні органи керування й інтерфейсу ОС. Також у її склад входять класи, що забезпечують розробку додатків для архітектури клієнт-сервер і трехуровневой архітектури (у сучасних версіях бібліотеки). Система програмування Mіcrosoft Vіsual C++ дозволяє розробляти будь-які додатки, що виконуються в середовищі ОС типу Mіcrosoft Wіndows, у тому числі серверні або клієнтські результуючі програми, що здійснюють взаємодію між собою по одній із зазначених вище архитектур.

Класи бібліотеки MFC орієнтовані на використання технологій COM/DCOM, а також побудованої на їхній основі технології Actіve для організації взаємодії між клієнтською й серверною частиною розроблювальних додатків. На основі класів бібліотеки користувач може створювати свої власні класи в мові C++, організовувати свої структури даних.

Навідміну від систем програмування компанії Borland, система програмування Mіcrosoft Vіsual C++ орієнтована на використання стандартних засобів зберігання й обробки ресурсів інтерфейсу користувача в ОС Wіndows. Це не дивно, оскільки всі версії ОС типу Wіndows розробляються самою компанією Mіcrosoft. Mіcrosoft Vіsual C++ забезпечує всі необхідні засоби для створення професійних Wіndows-приложений. Від версії до версії продукт стає простіше у використанні, розширюються можливості застосування, підвищується продуктивність.

Система програмування Mіcrosoft Vіsual C++ витримала кілька реалізацій. Впроцессе виходу нових версій системи програмування було випущено й кілька версій бібліотеки MFC, на якій заснована дана система.

Сама по собі бібліотека MFC є, на думку автора, досить удалою реалізацією широкого набору класів мови C++, орієнтованого на розробку результуючих програм, що виконуються під керуванням ОС типу Mіcrosoft Wіndows. Це багато в чому обумовлене тим, що творець бібліотеки компанія Mіcrosoft одночасно є й творцем ОС типу Mіcrosoft Wіndows, на які орієнтований об'єктний код бібліотеки. Бібліотека може бути підключена до результуючої програми за допомогою звичайного компоновщика, або використатися як динамічна бібліотека, що підключає до програми під час її виконання. Бібліотека MFC досить широко поширена. Її можливо використати не тільки в складі систем програмування виробництва компанії Mіcrosoft, але й у системах програмування інших виробників.

1.2.3 Мова програмування С++

Мова С++ виникла на початку 1980-х років, коли співробітник фірми «Bell Laboratories» Бьерн Страуструп вигадав ряд удосконалень до мови С. До початку офіційної стандартизації мова розвивалася в основному силами Страуструпа. У 1998 році був ратифікований міжнародний стандарт мови С++: «Standard for the C++ Programming Language».

Страуструп почав працювати над «Сі з класами» в 1979 році. Він вирішив доповнити мову С (наступник BCPL) можливостями, що є в мові Симула (Simula). Мова С, будучи базовою мовою системи UNIX, на якій працювали комп'ютери «Bell», є швидкою, багатофункціональною і переносною. Страуструпдодав до неї моєливість роботи з класами і об'єктами. В результаті, практичні задачі моделювання виявилися доступними для вирішення як з погляду часу розробки (завдяки використанню Симула - подібних класів) так і з погляду часу оючислень (завдяки швидкодії С). На початку в С були додані класи (з інкапсуляцією), похідні класи, сувора перевірка типів, inline-функції і аргументи за умовчанням.

У 1983г. відбулося перейменування мови з С з класами в С++ .

Мова С++ є сумісним знизу-вгору розширенням безпосередньо мови С. Принципова зміна полягає в появі нового реального примітивного типу даних - класу. Ця мова не містить типів даних високого рівня і примітивів, а, подібно мові С, забезпечує розширення за допомогою бібліотек. Таким чином, нові типи задаються безпосередньо усередині самої мови.

У мові С++ особлива увага надається можливості підтримки значної частини існуючого коду, написаного на мові С. Кроме того стають можливим порушення захисту даних і правил типізації, внаслідок чого доступ до даних може виявитися неправильним. Мова С++ є компромісним рішенням між об'єктно-орієнтованим ідеалом і прагматичним підходом.

Мова С++ підтримує абстракцію, спадкоємство, рекурсію і динамічне скріплення. Підтримується також і статичний, і динамічний контроль типів. Система типів є колекцією відносно невеликих дерев. Тому для С++ характерною є тенденція підтримування широких та пласких структур класів, яка буде сприяти створенню вузьких і глибоких структур класів.

Одна оригінальна властивість мови С++ пов'язана з можливістю представлення деякій функції (або класу) привілейованого доступу до закритих частин декількох класів, членом яких ця функція (або клас) не є. З цією метою така функція оголошується «другом» даних класів. Ця властивість може бути використане для спрощення опису операції над двома типами. Ця властивість вимагає украй обережного ставлення внаслідок можливого порушення інкапсуляції.

В мові С++ не передбачена збірка сміття. Подібна властивість повинна реалізуватися програмістом або надаватися середовищем у вигляді відповідних бібліотек коду. Останніми в мову С++ були додані шаблони. За їх допомогою підтримується необмежена універсальність; проте шаблони були піддані критиці з огляду на те, що вони приводять до роздування коду і уповільнення компіляції. Існує стандартна бібліотека шаблонів (STL), яка може використовуватися будь-яким компілятором.

Мову С++ можна використовувати як мову об'єктно-орієнтованого програмування, проте всі (або деякі) її об'єктно-орієнтовані властивості можна ігнорувати або неправильно використовувати. Нині використовується декілька різних версій мови С++, деякі з них є відкритими і загальнодоступними на базі найрізноманітніших машин.

Незважаючи на всі ці зауваження мова С++ є однією з най прийнятніших об'єктно-орієнтованих мов програмування для комерційни додатків, призначених для робочих станцій і персональних комп'ютерів.

Переваги мови С++ такі:

* Можливе виконання будь-яких дій на будь-якому рівні операційної системи;
* Мова С++ є найшвидшою з існуючих нині об'єктно-орієнтованих мов;
* Ринок програмістів, які пишуть на мові С++, достатньо великий.
* Можливість роботи на низькому рівні з пам'яттю, адресами, портами.
* Можливість створення узагальнених алгоритмів для різних типів даних, їх спеціалізація, і обчислення на етапі компіляції, використовуючи шаблони.

Недоліки С++ полягають в наступному:

* Відсутність автоматичного управління пам'яттю (складки сміття) і широко поширеневикористовування покажчиків призводять до того, що гарантії безпеки на стадії виконання досягаються насилу;
* Погана підтримка модульної. Підключення інтерфейсу зовнішнього модуля через препроцесорну вставку заголовного файлу (#include) серйозно уповільнює компіляцію, при підключенні великої кількості модулів.
* Нестача інформації про типи даних під час компіляції (CTTI).
* Мова С++ є складною для вивчення і для компіляції.
* Деякі перетворення типів неінтуїтивні. Зокрема, операція над беззнаковим і знаковим числами видає беззнаковий результат.
* Препроцесор С++ (успадковуваний від С) дуже примітивний. Це призводить з одного боку до того, що з його допомогою не можна (або важко) здійснювати деякі задачі метапрограмування, а з іншого, внаслідок своєї примітивності, він часто приводить до помилок і вимагає багато дій по обходу потенційних проблем.
* З кінця XX століття в співтоваристві С++ набуло поширення так зване метапрограмування на базі шаблонів. По суті, воно використовує особливості шаблонів C++ в цілях реалізації на їх базі інтерпретатора примітивної функціональної мови програмування, що виконується під час компіляції. Сама по собі дана можливість дуже приваблива, проте, внаслідок вищесказаного, такий код вельми важко сприймати і відладжувати.

1.3 Основи алгоритмізації

Одним з основоположних понять інформатики є алгоритм.

Алгоритм - система правив, що чітко описує послідовність дій, які необхідно виконати для вирішення задачі.

Алгоритм

Вихідні дані (результат)

Початкові

(вхідні)

дані

Властивості алгоритмів:

1. Дискретність;

2. Однозначність або детермінована. Кожна дія алгоритму повинна бути чіткою і однозначною. Для однакових початкових даних результат повинен бути одним і тим же.

3. Кінцівка, результативність. Алгоритм повинен призводити до рішення задачі за кінцеве число кроків.

4. Масовість або спільність. Алгоритм повинен призначатися не для однієї задачі, а цілого класу задач.

Для вирішення однієї і тієї ж задачі можуть застосовуватися декілька алгоритмів. Показником якості алгоритму є його ефективність. Ефективність алгоритму визначається часом рішення задачі і обсягом, необхідним для цього вирішення.

Алгоритм має ієрархічну структуру. Для деяких кроків алгоритму можна складати детальніші алгоритми, що містять опис даного кроку.

Ступінь деталізації залежить від мови програмування, яка вибрана для вирішення даної задачі. Мови високого рівня вимагають високий ступінь деталізації, ніж низького.

Існують різні способи запису алгоритмів:

1. Словесний опис кожного кроку на словесній мові.

2. Графічне зображення алгоритму у вигляді блок-схеми (flowchart).

У структурному програмуванні застосовуються декілька основних прийомів розробки алгоритмів.

1. Метод покрокової деталізації.

Спочатку створюється загальна структура алгоритму, а потім приробляються окрем частини.

2. Метод «зверху - вниз».

Спочатку розробляється задача на родовому рівні, всі підзадачі залишаючи у вигляді чорних ящиків, в яких чітко прописані вхідні-вихідні дані.

3. Метод «знизу - вгору».

Велика увага надається розробці даних, задачі верхнього (родового) рівня використовують вже налагоджені і протестовані підзадачі. Але в цьому випадку, якщо виникає необхідність зміни родового рівня, часто доводиться змінювати й підзадачі.

4. Метод «від центру по краях».

Виділяється найскладніша частина задачі, йде пошук ї рішення, а потім проводиться вся робота, що залишилася.

5. Модульний метод.

Має на увазі, що програма складається з окремих модулів. Під «модулем» розуміється функціонально і логічно замкнута частина програми, реалізовуюча деякі алгоритми. Модуль може бути функцією з чітко певними вхідними і виходять даними.

Основна мета використовування модуля – можливість його повторного використання для вирішення різних задач.

2 РОЗРОБКА РІШЕННЯ ЗАДАЧІ СТВОРЕННЯ БАЗИ ВІДЕОФІЛЬМІВ

2.1 Бази даних

2.1.1 Основні положення та моделі БД

Взагалі технологія баз даних як самостійна гілка розвитку інформатики з’явилася порівняно недавно: початок досліджень в цій області пов'язують з епохою великих ЕВМ – шістдесятими роками XX в. Тоді ж сформувався теоретичний апарат технології, почалася розробка програмного забезпечення. Термін «база даних» вперше з’явився в 1962г., відтоді дещо змінилося його змістовне наповнення (у зв'язку з розвитком області) і в найзагальнішому вигляді може бути сформульований так: база даних - це сукупність взаємозв'язаних даних при такій мінімальній надмірності, яка допускає їх використання оптимальним чином для одного або декількох додатків в певній предметній області людської діяльності.

Насправді сьогодні існує безліч різних визначень терміну БД, що обумовлене наявністю «можливості перетворення інформаційних масивів інших форматів в бази даних в строгому значенні цього поняття (і навпаки)". Практично будь-яку колекцію даних, які може зчитувати машина, можна при незначній доробці перевести в базу даних, що дозволяє максимально ефективно проводити інформаційний пошук за допомогою відповідної системи управління (СУБД).

Отже, база даних - структурований організований набір даних, що описують характеристики яких-небудь фізичних або віртуальних систем.

«Базою даних» часто спрощено або помилково називають Системи Управління Базами даних (СУБД). Потрібно розрізняти набір даних (власне БД) і програмне забезпечення, призначене для організації і ведення бази даних (СУБД).

Організація структури БД формується виходячи з таких міркувань:

1. Адекватність описуваному об'єкту/системі - на рівні концептуальної і логічної моделі.

2. Зручність використання для ведення обліку і аналізу даних - на рівні так званої фізичної моделі.

Види концептуальних (інфологічних) моделей БД: «сутність - зв'язок», семантичні, графські.

Види логічних (даталогічних) моделей БД:

1. Документальні (архіви) - орієнтовані на формат документа, дескрипторні, тезаурусні.

2. Фактографічні (картотеки):

- теоретико-графські: ієрархічна модель, мережева модель.

- теоретико-множинні: реляційна модель (ER-модель), багатомірна модель.

- об'єктно-орієнтовані: об'єктна модель.

- засновані на інвертованих файлах.

На рівні фізичної моделі електронна БД є файлом або їх набором у форматі TXT, CSV, Excel, DBF, XML або в спеціалізованому форматі конкретної СУБД. Також в СУБД в поняття фізичної моделі включають спеціалізовані віртуальні поняття, існуючи в її рамках - таблиця, табличний простір, сегмент, куб, кластер і т.д.

Нині найбільше поширення набули реляційні бази даних. Картотеками користувалися до появи електронних баз даних. Мережеві та ієрархічні бази даних вважаються застарілими, об'єктно-орієнтовані поки ніяк не стандартизовані і не набули широкого поширення. Деяке відродження одержали ієрархічні бази даних у зв'язку з появою і поширенням XML.

2.1.2 Типи БД

У класичній теорії виділяють три основні типи баз даних: ієрархічні, мережеві та реляційні. Останніми роками, з широким розповсюдженням об'єктно-орієнтованих мов програмування, стали активно розвиватися об'єктні бази даних.

Першими з’явилися ієрархічні бази даних. Інформація в ієрархічній базі організована за принципом деревовидної структури, у вигляді відносин предок/нащадок. Кожен запис може мати не більш одного батьківського запису і декілька підлеглих. Зв'язки записів реалізуються у вигляді фізичних покажчиків з одного запису на інший. Основний недолік ієрархічної структури бази даних - неможливість реалізувати відносини "багато до багатьох", а також ситуації, коли запис має декілька предків.

Мережева структура баз даних з’явилася як розвиток ієрархічної. Треба підкреслити, що термін «мережева» уживається тут в зовсім незвичному для епохи интернета значенні. Це слово підкреслює модель зв'язків даних в базі, коли кожен запис може знаходитися у відносинах «багато до багатьох» з іншими записами, що робить графічну модель бази схожою на рибальську мережу. Розробляти серйозні додатки в рамках мережевої моделі бази даних досить важко, причому складність розробки при ускладненні задач зростає в геометричній прогресії.

Справжній прорив в розвитку баз даних відбувся тоді, коли збільшена потужність комп'ютерів дозволила повною мірою реалізувати реляційну модель даних. Теорія реляційних баз даних була розроблена доктором Коддом на початку 70-х років 20 століття. У реляційних базах дані зберігаються у вигляді таблиць, що складаються з рядків і стовпців. Стовпці таблиць реляційної бази можуть містити скалярні дані фіксованого типу - числа, рядка, дати... Таблиці в реляційній базі даних можуть бути зв'язані відносинами "один до одного" або "один до багатьох".

Реляційні бази даних займають зараз домінуюче становище. Не буде великим перебільшенням сказати, що ієрархічна і мережева структури баз даних пішли в минуле, поступившися своїм місцем реляційним базам. MS SQL Server і MS Access, InterBase і FoxPro, PostgreSQL і Paradox... Всі вони побудовані на реляційній моделі даних. Реляційною базою в чистому вигляді є і MySQL.

Об'єктно-орієнтовані бази даних з’явилися зовсім нещодавно як природний розвиток об'єктно-орієнтованих мов програмування. На сьогоднішній день ООБД поки не мають широкого розповсюдження, але, поза сумнівом, вони найближчим часом бурхливо розвиватимуться. Це підтверджує і той факт, що розробники багатьох реляційних БД включають в свої бази засоби роботи з об'єктними типами даних. Такі бази даних одержали назву об'єктно-реляційних. Таким же чином, зокрема, розвивається і Oracle. Що була раніше чисто реляційною базою, Oracle починаючи з 8 версії підтримує можливість зберігання і обробки об'єктів і без жодної натяжки може бути віднесена до об'єктно-реляційного класу баз даних.

## 2.1.3 Етапи проектування БД

I. Концептуальне проектування - збирання, аналіз і редагування вимог до даних. Для цього здійснюються слідуючі заходи:

1. обстеження предметної області, вивчення її інформаційної структури.
2. виявлення всіх фрагментів, кожний з яких характеризується призначеним для користувача представленням, інформаційними об'єктами і зв'язками між ними, процесами над інформаційними об'єктами.
3. моделювання і інтеграція всіх представлень.

Після закінчення даного етапу одержуємо концептуальну модель, інваріантну до структури бази даних. Часто вона представляється у вигляді моделі «сутність-зв'язок».

II. Логічне проектування - перетворення вимог до даних в структури даних. На виході одержуємо СУБД-орієнтовану структуру бази даних і специфікації прикладних програм. На цьому етапі часто моделюють бази даних стосовно різних СУБД і проводять порівняльний аналіз моделей.

III. Фізичне проектування - визначення особливостей зберігання даних, методів доступу і т.д.

Відмінність рівнів представлення даних на кожному етапі проектування реляційної бази даних:

КОНЦЕПТУАЛЬНИЙ РІВЕНЬ - Представлення аналітика (використовується інфологічна модель «сутність-зв'язок»)

* сутність
* атрибути
* зв'язки

ЛОГІЧНИЙ РІВЕНЬ - Представлення програміста

* записи
* елементи даних
* зв'язки між записами

ФІЗИЧНИЙ РІВЕНЬ - Представлення адміністратора

* групування даних
* індекси
* методи доступу

2.1.4 Використання БД

Одним з найбільш розроблених напрямів в інформатиці сьогодні можна назвати створення баз даних (БД). Дійсно, будь-яке дослідження базується на використанні інформації і у кожного фахівця існує свій, персональний архів довідкової інформації, який зберігається звичайно у вигляді картотеки з різних вирізок, заміток або публікацій. Природно, що першою перевагою при знайомстві з комп'ютером, предстає можливість систематизації своїх матеріалів в базу даних, електронний варіант якої буде набагато зручнішим. Крім того, тут є і інша перевага – можливість використати чужу БД як вторинне джерело і заощадити таким чином масу часу і зусиль, які були б витрачені на дублювання, по суті, однієї і тієї ж роботи. Систематизований і проаналізований розробником БД матеріал, система посилань і пошуку інформації дозволяють, минувши трудомісткий етап чисто механічної роботи, зосередитися безпосередньо на рішенні поставлених задач.

Створення бази даних передбачає рішення цілого комплексу проблем, в т.ч. розробку програмних, мовних, організаційних і технічних засобів, призначених для централізованого зберігання і колективного використання інформації. Проте заздалегідь слід здійснити відбір цієї самої інформації (яка і складе фонд БД), структурувати її. Тут і починаються основні складнощі. Фактично БД повинна бути інформаційною моделлю проблеми, яка розв’язується за допомогою БД. Визначення об'єкту бази даних принципово важливе, оскільки впливає як на відбір джерел, так і на способи зберігання.

З одного боку, більшість БД створюється в ході рішення конкретних задач. Це проблемно-орієнтовані БД. Але є і такі бази, метою створення яких є збереження інформації, що міститься в джерелі, і велика доступність її для дослідника. Ці напрями розвиваються в руслі відповідно методико- і источнико-орієнтованих підходів.

Ідеальною була б БД, що суміщає обидва принципи. Дослідник у такому разі дістає можливість не просто доступу до великого масиву структурованих даних у вигляді, доступному машині для зчитування, але і може створити в цій структурі свої власні моделі, орієнтовані на рішення конкретних історичних задач.

На даний момент можна констатувати значну роль БД в дослідженні будь-якого фахівця, що використовує в своїй роботі комп'ютер. Важливим досягненням представляється все більший перехід останніми роками з рішення чисто практичних задач до осмислення теоретичних і методологічних проблем.

2.2 Структури даних

В обчислювальній техніці структура даних - це програмна одиниця, що дозволяє зберігати і обробляти безліч однотипних і/або логічно зв'язаних даних. Для додававння, пошуку, зміни і видалення даних структура даних надає деякий набір функцій, що становлять інтерфейс структури даних. Структура даних часто є реалізацією якого-небудь абстрактного типу даних .

При розробці програмного забезпечення велику роль виконує проектування сховища даних, і представлення всіх даних у вигляді безлічі зв'язаних структур даних. Добре спроектоване сховище даних оптимізує використання ресурсів (таких як час виконання операцій, використовуваний обсяг оперативної пам'яті, число звернень до дискових накопичувачів), що вимагаються для виконання найкритичніших операцій.

Структури даних формуються за допомогою типів даних посилань і операцій над ними у вибраній мові програмування .

Різні види структур даних підходять для різних додатків; деякі з них мають вузьку спеціалізацію для певних задач. Наприклад, Б-деревья звичайно підходять для створення баз даних, тоді як хэш-таблиці використовуються повсюдно для створення різного роду словників, наприклад, для відображення доменних імен в інтернет адреси комп'ютерів.

При розробці програмного забезпечення складність реалізації і якість роботи програм істотно залежить від правильного вибору структур даних. Це розуміння дало початок формальним методам розробки і мовам програмування, в яких саме структури даних, а не алгоритми, ставляться в розділ архітектури програмного засобу. Велика частина таких мов володіє певним типом модульної, дозволяючим структурам даних безпечно перевикористовуватися в різних додатках. Об'єктно-орієнтовані мови, такі як Java, C# і C++, є прикладами такого підходу.

Багато класичних структур даних представлені в стандартних бібліотеках мов програмування або безпосередньо вбудоване в мови програмування. Наприклад, структура даних хэш-таблиця вбудована в мови програмування Lua, Perl, Python, Ruby, Tcl і ін. Широко використовується стандартна бібліотека шаблонів STL мови C++. Фундаментальними будівельними блоками для більшої частини структур даних є масиви, записи (конструкція struct в мові С і конструкція record в мові Паскаль ), розмічені об'єднання (конструкція union в мові С) і посилання.

2.3 Функції

Функція - в програмуванні - один з видів підпрограми . Особливість, що відрізняє її від іншого виду підпрограм - процедури, полягає в тому, що функція повертає значення, а її виклик може використовуватися в програмі як вираження.

З погляду теорії систем, функція в програмуванні - окрема система (підсистема, підпрограма), на вхід якої поступають управляючі дії у вигляді значень аргументів . На виході системи одержуємо результат виконання програми. По ходу виконання функції можуть виконуватися також деякі зміни в керованій системі, причому як зворотні, так і незворотні.

У деяких мовах програмування (наприклад, в Паскалі) функції існують разом з процедурами (підпрограмами, що не повертають значення), в інших, наприклад, в С, є єдиним реалізованим видом підпрограми (тобто всі підпрограми є функціями і можуть повертати значення).

Побічним ефектом функції називається будь-яка зміна функцією стану програмного середовища, окрім повернення результату (зміна значень глобальних змінних, виділення і звільнення пам'яті, введення-виведення і так далі). Теоретично найправильнішимє використання функцій, що не мають побічного ефекту (тобто таких, в результаті виклику яких повертається обчислене значення, і лише), хоча на практиці доводиться використовувати функції з побічним ефектом, хоча б для забезпечення введення-виведення і відображення результатів роботи програми. Існує специфічна парадигма програмування - функціональне програмування, в якій будь-яка програма є набором вкладених викликів функцій, що не викликають побічних ефектів. Найвідоміша мова програмування, реалізовуюча цю парадигму - Лісп. У ньому будь-яка операція, будь-яка конструкція мови, будь-яке вираження, окрім константи, є викликами функцій.

програмування база відеофільм

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ БАЗИ ВІДЕОФІЛЬМІВ

3.1 Опис програмної реалізації бази відеофільмів

На початку роботи було підключено стандартні бібліотеки мови С <vcl.h>, <iostream.h> (для роботи з файлами), <conio.h> (для затримання екрана за допомогою функції getch()), <string.h> (для роботи зі строками), <stdio.h> (введення-виведення даних), <stdlib.h> (для використання функцій псевдовипадкових чисел).

Структура даних videoteka складається з даних символьного типу (char), які ініціалізують назву фільму (name), студію запису фільму (studio), жанр фільму (genre), рік випуску фільму (year), режисера фільму (producer), акторів, що грають у фільмі (actors), коротку характеристики фільму (about), суб'єктивну оцінку фільму (bal) і прізвище користувача відеотеки, охочого одержати або повернути касету (FIO).

В моїй програмі використовуються різноманітні функції, які відповідають за:

1. Функція showlist() записує дані про фільм і зчитує їх при необхідності.
2. Функція recive() дозволяє користувачу повернути касету. Функція також має перевірку на наявність помилок. Якщо касета знаходиться в відеотеці, про що свідчить надпис «No» в полі «Taken», то програма видасть повідомлення про неможливість сдачі касети в відеотеку: «You can not give current movie back, because it is not given out».
3. Функція delivery() містить інформацію про видачу касети. Якщо касета знаходиться на руках, то функція видає повідомлення про неможливість видачі касети «You can not receive current movie, because it is in».
4. Функція showfull() показує повну інформацію про фільм.
5. Функція list() організовує меню:

«1-View full information about movie

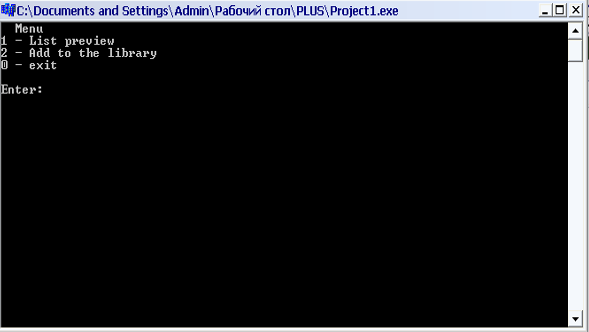
2-Distribute movie

3-Return movie», за допомогою якого стає можливим проглянути всю інформацію про фільм, одержати фільм і повернути його у відеотеку відповідно, що значно спрощує роботу з програмою.

1. Функція library() містить всі фільми, які присутні в відеотеці, а також всю інформацію про ці фільми. За допомогою цієї функції також можна додавати фільми до складу відеотеки.
2. Функція menu() дозволяє переглянути інформіцію про всі фільми відеотеки з подальшою можливістю замовити або повернути будь-яку касету з наявних, додати фільм до відеотеки та вийти з меню за допомогою трьох цифр 1, 2, 0 відповідно: «1 - List preview 2 - Add to the library 0 - exit».

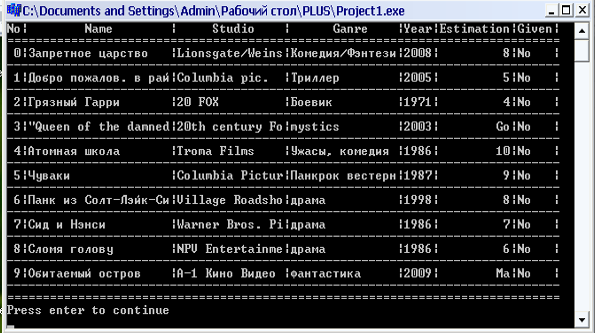
3.2 Результати, отримані користувачем при використанні отриманого програмного забезпечення

На малюнку 3.1 зображено початковий стан поля відеотеки, тобто головне меню, за допомогою якого можна рухатися далі, поступово виконуючи потрібні нам операції.



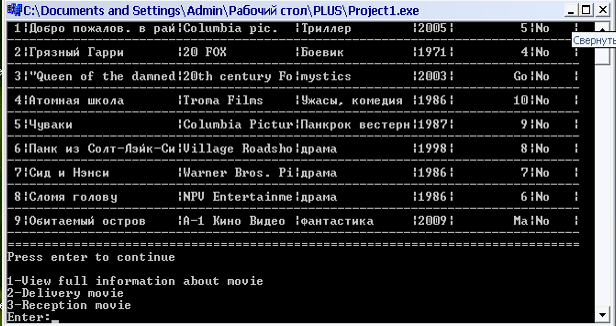
Малюнок 3.1 Головне меню

Натискаємо 1 для перегляду відеотеки.



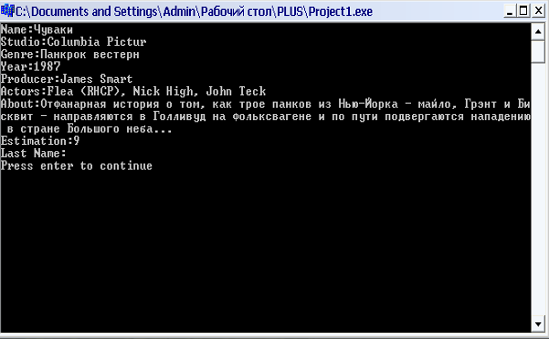
Малюнок 3.2 Зміст відеотеки

Натискаємо клавішу Ентер і перед нами з’являється меню, розташоване під переліком фільмів відеотеки.



Малюнок 3.3 Зміст відеотеки з меню

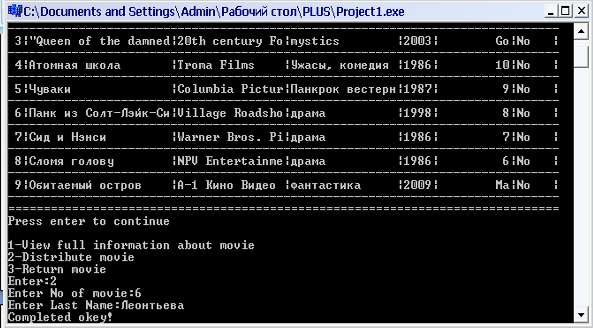
Натискаємо 1 для перегляду повнішої інформації про будь-який фільм. Потімвводтмо номер фільму, що цікавить нас, і читаємо відповідну інформацію. Дляприкладу візьмемо фільм під номером 5.



Малюнок 3.4 Інформація про фільм під номером 5 в відеотеці

Далі ми повертаємося до головного меню, використовуючи клавішу Ентер.

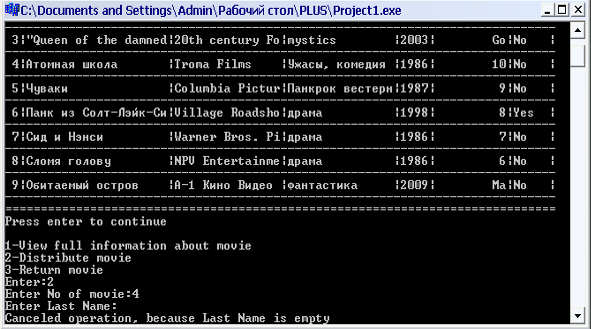
Для того, щоб узяти касету, необхідно в головному меню вибрати пункт «Distribute», ввести номер фільму і своє прізвище для успішного замовлення.



Малюнок 3.5 Успішне замовлення касети

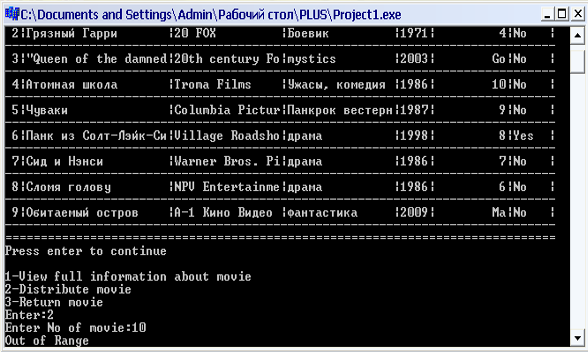
При правильності введення всіх даних в полі «Given» навпроти фільму з’явиться «Yes», що означатиме відсутність касети у відеотеці.

Якщо ви не ввели своє прізвище, то програма видасть помилку «Canceled operation, because Last Name is empty».



Малюнок 3.6 Помилка при замовленні касети

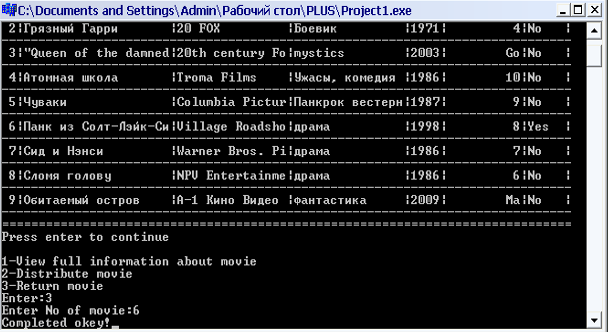
Також буде виведена помилка «Out of Range», якщо ви ввели номер, який не відповідає жодному з фільмів (в даному випадку більше 9).



Малюнок 3.7 Помилка при замовленні касети

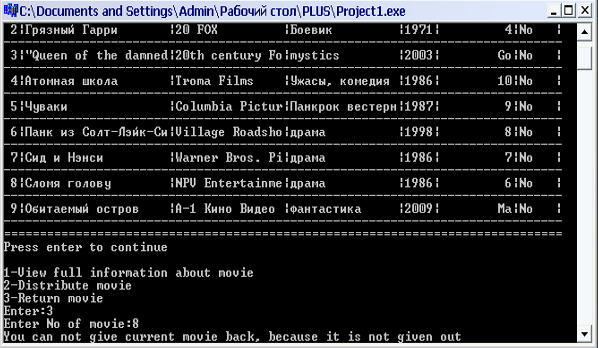
Повернення до головного меню здійснюється за допомогою клавіші Ентер.

Для повернення касети в тому ж меню натискаємо цифру 3 («Return»), потім номер фільму.



Малюнок 3.8 Успішне повернення касети

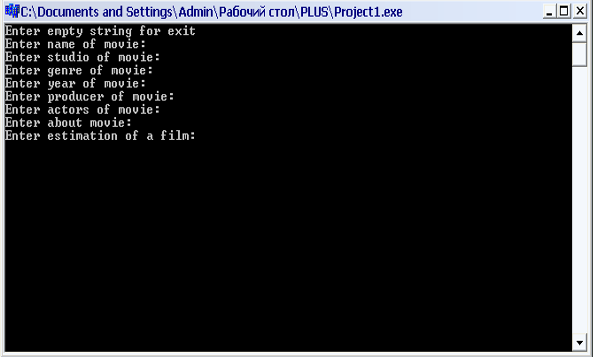
Буде видана помилка, якщо ви хочете повернути касету, яка в даний момент є в наявності, тобто знаходиться не на руках.



Малюнок 3.9 Помилка при поверненні касети

При успішному поверненні касети в полі «Given» навпроти фільму висвітиться «No», що означає наявність касети у відеотеці і можливість її замовлення.

За допомогою натиснення клавіші «2» в головному меню можна додавати фільми у відеотеку, слідуючи простим навігаціям програми.



Малюнок 3.10 Меню додавання фільмів до відеотеки

ВИСНОВОК

У ході виконання даного курсового проекту була розроблена програма мовою високого рівня у середовищі Builder C++, а також вивчені можливості даної мови.

Систематизовано й закріплено практичні навички програмного забезпечення, а також теоретичні знання по основних розділах курсу "Об’єктно-орієнтованого програмування". При виконанні курсового проекту проведено знайомство з інформаційними джерелами по об’єктно- орієнтованому програмуванню з метою аналізу стану розв'язуваного завдання.

Отримано практичні навички роботи в середовищі Builder C++.

В даній курсовій роботі була розроблена система, яка дозволяє зберігати інформацію та використовувати її в праці для рішення конкретних задач. Ця система має назву проблемно-орієнтовані бази даних.

За допомогою розробленої бази відеофільмів можна переглядати інформацію про наявні фільми, а також замовляти або повертати касети, що значно полегшує роботу продавця або виключає його посередність взагалі.

На даний момент можна констатувати значну роль баз даних в дослідженні будь-якого фахівця, що використовує в своїй роботі комп'ютер.

СПИСОК ЛІТЕРАТУРИ

* 1. http:www.intuit.ru
  2. Перколаб Ю.В., Браткевич В.В. Объектно-ориентированное программирование. Конспект лекций. – Харьков: Изд. ХГЭУ, 2003. – 168с.
  3. Гиковатый В.М. Основы алгоритмизации на базе ООП. Учебное пособие. Харьков: Изд. ХНЭУ, 2006. – 92 с.
  4. Грэхели Иан Объектно-ориентированные методы. Принципы и практика. 3-е издание: Пер. с англ. – М.: Изд. дом «Вильямс», 2004. – 880с.
  5. Кузин А.В., Левонисова С.В. Базы данных. Учебное пособие для студентов высших учебных заведений. – СПб.: Academia, 2005. – 320 с.
  6. Грабер М. SQL для простых смертных. – М.: Изд. Лори, 2008. – 375 с.
  7. Альфред Ахо, Джеффри Д.Ульман. Структуры данных и алгоритмы. 2003. - 367с.

ДОДАТОК А

Текст програмного коду

#include <vcl.h>

#include <iostream.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#pragma hdrstop

//---------------------------------------------------------------------------

#pragma argsused

struct videoteka

{

char name[21];

char studio[16];

char genre[16];

char year[5];

char producer[16];

char actors[101];

char about[201];

char bal[3];

char FIO[51];

};

void showlist(FILE\* f1)

{

clrscr();

fseek(f1,0,SEEK\_SET);

cout<<"No| Name | Studio | Ganre |Year|Estimation|Given|\n";

cout<<"==============================================================================\n";

int i=0;

videoteka ex;

while (fread(&ex, sizeof(ex), 1, f1))

{

printf ("%2i|",i);

printf ("%-20s|",ex.name);

printf ("%-15s|",ex.studio);

printf ("%-15s|",ex.genre);

printf ("%4s|",ex.year);

printf ("%10s|",ex.bal);

if (ex.FIO[0]==0)printf ("No |");

else

printf("Yes |");

cout<<"\n------------------------------------------------------------------------------\n";

i++;

} cout<<"==============================================================================\n";

}

void recive(FILE\* f1,char line[255])

{

videoteka ex;

int count=0;

fseek(f1,0,SEEK\_SET); //првернення до початку потоку

while (count<=StrToInt(line))

{

if(!(fread(&ex, sizeof(ex), 1, f1)))

{

cout<<("Out of Range");

getchar();

return;

}

count++;

}

if (ex.FIO[0]==0)

{

cout<<"You can not give current movie back, because it is not given out";

getchar();

return;

}

for (int i=0;i<51;i++) ex.FIO[i]=0;

fseek(f1,-51,SEEK\_CUR);

fputs(ex.FIO,f1);fputc(0,f1);

cout<<"Completed okey!";

getchar();

}

void delivery(FILE\* f1,char line[255])

{

videoteka ex;

int count=0;

fseek(f1,0,SEEK\_SET);

while (count<=StrToInt(line))

{

if(!(fread(&ex, sizeof(ex), 1, f1)))

{

cout<<("Out of Range");//номер не відповідає жодному з фільмів

getchar();

return;

}

count++;

}

if (ex.FIO[0]!=0)

{

cout<<"You can not receive current movie, because it is in "<<ex.FIO;

getchar();

return;

}

cout<<"Enter Last Name:"; cin.getline(line,255);

if (line[0]==0)

{

cout<<"Canceled operation, because Last Name is empty";//ви не ввели прізвище замовника

getchar();

return;

}

for (int i=0;i<50;i++) ex.FIO[i]=line[i];

ex.FIO[50]=0;

fseek(f1,-51,SEEK\_CUR);

fputs(ex.FIO,f1);fputc(0,f1);

cout<<"Completed okey!";

getchar();

}

void showfull(FILE\* f1, char line[255])

{

clrscr();

fseek(f1,0,SEEK\_SET);

videoteka ex;

int count=0;

while (count<=StrToInt(line))

{

if(!(fread(&ex, sizeof(ex), 1, f1)))

{

cout<<("Out of Range");

getchar();

return;

}

count++;

}

cout<<"Name:"<<ex.name<<"\n";

cout<<"Studio:"<<ex.studio<<"\n";

cout<<"Genre:"<<ex.genre<<"\n";

cout<<"Year:"<<ex.year<<"\n";

cout<<"Producer:"<<ex.producer<<"\n";

cout<<"Actors:"<<ex.actors<<"\n";

cout<<"About:"<<ex.about<<"\n";

cout<<"Estimation:"<<ex.bal<<"\n";

cout<<"Last Name:"<<ex.FIO<<"\n";

cout<<"Press enter to continue\n"; cin.getline(line,255);

}

void list()

{

FILE\* f1;

if (!(f1=fopen("lyb.txt","r+")))//відкриття файлу за допомогою режима «r+», який означає зчитування та запис інформації

{

perror("Error in open file!!!");// помилка у відкритті файла

getchar();

exit(0);

}

showlist(f1);

char line[255];

cout<<"Press enter to continue\n"; cin.getline(line,255);

cout<<"1-View full information about movie\n";

cout<<"2-Distribute movie\n"; //видача касети

cout<<"3-Return movie\n"; //повернення касети

cout<<"Enter:"; cin.getline(line,255);

if (line[0]=='1')

{

cout<<"Enter No of movie:"; cin.getline(line,255);

showfull(f1,line);

line[0]=0;

}

if (line[0]=='2')

{

cout<<"Enter No of movie:"; cin.getline(line,255);

delivery(f1,line);

line[0]=0;

}

if (line[0]=='3')

{

cout<<"Enter No of movie:"; cin.getline(line,255);

recive(f1,line);

line[0]=0;

}

fclose(f1);

}

void library()

{

clrscr();

FILE\* f1;

if (!(f1=fopen("lyb.txt","a+")))//відкриття файлу за домогою режима «а+», що означає зчитування та додавання інформації

{

perror("Error in open file!!!");

getchar();

exit(0);

}

do

{

clrscr();

cout<<"Press Enter for exit\n";

videoteka ex1;

char line[255];

for (int i=0;i<255;i++)line[i]=0;

cout<<"Enter name of movie:"; cin.getline(line,255);

if (line[0]=='\0')

{

fclose(f1);

return;

}

for (int i=0;i<20;i++) ex1.name[i]=line[i];

cout<<"Enter studio of movie:"; cin.getline(line,255);

for (int i=0;i<15;i++) ex1.studio[i]=line[i];

cout<<"Enter genre of movie:"; cin.getline(line,255);

for (int i=0;i<15;i++) ex1.genre[i]=line[i];

cout<<"Enter year of movie:"; cin.getline(line,255);

for (int i=0;i<4;i++) ex1.year[i]=line[i];

cout<<"Enter producer of movie:"; cin.getline(line,255);

for (int i=0;i<15;i++) ex1.producer[i]=line[i];

cout<<"Enter actors of movie:"; cin.getline(line,255);

for (int i=0;i<100;i++) ex1.actors[i]=line[i];

cout<<"Enter about movie:"; cin.getline(line,255);

for (int i=0;i<200;i++) ex1.about[i]=line[i];

cout<<"Enter estimation of a film:"; cin.getline(line,255);

for (int i=0;i<2;i++) ex1.bal[i]=line[i];

ex1.FIO[0]='\0';

ex1.name[20]=0;

ex1.studio[15]=0;

ex1.genre[15]=0;

ex1.year[4]=0;

ex1.producer[15]=0;

ex1.actors[100]=0;

ex1.about[200]=0;

ex1.bal[2]=0;

ex1.FIO[50]=0;

fwrite(&ex1,sizeof(ex1),1,f1);//запис даних про тільки-но введені фільми до файлу

}

while (1==1);

}

void menu()

{

char i[255];

do

{

clrscr();

cout<<" Menu\n";

cout<<"1 - List preview\n";

cout<<"2 - Add to the library\n";

cout<<"0 - exit\n";

cout<<"\nEnter:";

cin.getline(i,255);

if (i[0]=='1') list();

if (i[0]=='2') library();

}

while (i[0]!='0');

}

void main()

{

menu(); }

ДОДАТОК Б

Перелік фільмів в відеотеці

