# Разработка макета СУБД

## Цели и содержание практики

Целями учебной практики «Лабораторно-вычислительный практикум» являются:

- приобретение практических навыков объектно-ориентированного программирования с использованием методов групповой разработки программ,

- приобретение практических навыков создания программных продуктов (представление программы в виде библиотеки классов и разработка необходимой для ее использования технической документации),

- приобретение практических навыков использования разработанной библиотеки при выполнении индивидуальных заданий,

- закрепление и углубление теоретической подготовки студентов покурсам «Информатика», «Основы программирования» и «Программирование на основе классов и шаблонов».

При разработке библиотеки классов помните,что одна из главных целей ООП заключается в разработке удобного (понятного) интерфейса и скрытии подробностей реализации, чтобы пользователи вашей библиотеки могли программировать на более высоком уровне, пользуясь интерфейсом классов.

## Содержание

Содержание практики.

1.1.1 Разработать систему классов для указанной в задании предметной области (общей для членов одной бригады), которая обеспечивала бы выполнение индивидуальных заданий каждого члена бригады.

1.1.2. Представить UML- диаграмму разработанной системы классов с указанием связей между классами, интерфейсов и полей классов.

1.1.3. Разработка в соответствии с индивидуальным заданием демонстрационной программы.

1.1.4. Разработать следующую техническую документацию.

- «Техническое задание» (документ должен быть согласован с преподавателем до начала программирования).

- «Программа и методика испытаний».

- «Руководство пользователя».

- Листинги исходных кодов (файлы **.cpp** и **.h** с комментариями).

1.1.5. Этапы практики и текущий контроль.

Практика проводится на 2-м семестре в виде нормируемой СРС, выполняемой под руководством преподавателя. Студенты работают в интегрированной среде программирования MS Visual Studio 2012 C++.

Форма текущего контроля – зачет с оценкой.   
 Общая трудоемкость учебной практики составляет 74 часа, из них  
 32 часа – аудиторные занятия в компьютерном классе, 40 часов – СРС,   
 2 часа – контроль выполнения.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№**  **п/п** | **Разделы (этапы) практики** | **Виды учебной работы, на практике включая самостоятельную работу студентов, и трудоемкость**  **(в часах)** | | | | **Формы текущего контроля** |
| 1 | Подготовительный этап  Разделение студентов на группы по 3-4 человека и назначение бригадиров в группах.  Выдача и согласование с преподавателем индивидуальных заданий. | СРС  1-я нед.  2 час. | Аудиторные занятия в компьютерном классе  2 час. |  |  | Утвержденные списки подгрупп и варианты индивиду-  альных  заданий |
| 2 | Разработка и согласование ТЗ | 5-я нед. 8 час. | 8 час. |  |  | Подписан-ное ТЗ   25 баллов |
| 3 | Создание файлов с таблицами, содержащими тестовые данные, разработка функций для работы с этими файлами и разработка ПМИ. | 8-я нед. 10 час. | 10 час. |  |  | Демонстра-ция резуль-татов пре-подавателю  25 баллов |
| 4 | Создание библиотеки,  разработка в соответствии с индивидуальным заданием демонстрационной программы, выполнение тестирования и разработка документа ОП. | 14-я нед. 20 час. | 12 час. |  |  | Демонстра-ция резуль-татов пре-подавателю  50 баллов |
|  | Контроль выполнения |  | 2 час. |  |  | Зачет с оценкой |

## Общие указания

Для выполнения практики студенты разделяются на подгруппы по 3-4 человека. Один из них выбирается бригадиром. Каждая подгруппа выполняет свое задание, используя технологию групповой разработки программ.

Перед началом программирования разработать тестовый пример.

### Пример разработки таблиц БД *Library* и *Company*.

В целях упрощения разработки и использования системы классов для работы с БД введем ***соглашения*** по именованию и размещению файлов с таблицами БД на диске и по формату таблиц и файлов.

Каждая БД хранится на диске в отдельном каталоге, который содержит текстовые файлы с таблицами. Каждая таблица хранится в отдельном файле.

#### <имя файла> = <имя таблицы>.txt. Данные таблиц хранятся в текстовых файлах формате CSV с разделителем ‘|’.

Таблица состоит из «шапки» (список заголовков столбцов) и строк данных.

Структура файлов:

- первая строка файла содержит список заголовков всех столбцов таблицы, а остальные строки – записи данных таблицы,

- заголовок каждого столбца таблицы состоит из двух слов:

<имя столбца> и <имя типа данных в столбце>.

- строки данных содержат значения полей таблицы.

В макете СУБД достаточно ограничиться 4-мя типами данных:

string, int, double и DBDate.

DBDate – тип для представления времени в БД. Он должен быть создан пользователем для выполнения операций со временем при работе с БД.

***База данных библиотеки*** ***Library*** содержит служебный файл DBTables с именами всех таблиц БД и 4 таблицы с данными:

Abonements, Students, Books и Subjects.

Поля таблиц для БД Library:

Таблица Students: StudentID, Name, Group.

Таблица Subjects: SubjectID, Subject, где:

Subject – тематика книги ( физика, математика, информатика)

Таблица Books: BookID, Title, SubjectID, Quantity, Number, где:

Quantity – количество поступивших в библиотеку экземпляров данной книги.

Number – количество выданных экземпляров данной книги.

Таблица Abonements: StudentID, BookID, OutDate, InDate, где:

DateOut, DateIn – дата выдачи и дата возврата книги.

Заполнить файлы тестовыми данными. Ввести не менее трех книг по каждому из трех предметов, не менее 5 записей в таблицу Students и не менее 10 записей в таблицу Abonements.

При заполнении таблицы данными проверять уникальность значений идентификаторов и ссылочную целостность данных, то есть наличие в основных (родительских) таблицах Students, Books и Subjects значений идентификаторов StudentID, BookID и SubjectID, использованных в дочерних таблицах Abonements и Books.

***База данных Company*** компании по оптовой торговле содержит служебную таблицу DBTables с именами всех таблиц БД и 4 таблицы с данными:

Orders, OrderDetails, Customers, Products.

Поля таблиц для БД ***Company***:

Customers: CustomerID, CompanyName, ContactName

OrderDetails: OrderID, ProductID, UnitPriceOut, Quantity

Orders: OrderID, CustomerID,|OrderDate

Products: ProductID, UnitPriceIn, UnitInStock

Товары отпускаются в упаковке. Покупатель – магазин. Quantity – число упаковок.

Типы данных в столбцах таблиц:

- CustomerID, OrderID, Quantity, UnitInStock – int (имя типа Int32);

- CompanyName, ContactName, CustomerID – string (имя типа String);

- UnitPriceIn, UnitPriceOut (закупочная и отпускная цена) – double (имя типа Double);

- OrderDate (дата исполнения заказа - DBDate (имя типа DBDate);

## Варианты заданий

***Вариант 1***

Оформление заказа на отпуск товаров со склада (добавить запись и сделать необходимые изменения в таблицах). Перед оформлением заказа сделать проверку наличия товара на складе.

**Вариант 2**

Выдача книги в библиотеке. На одних руках не должно быть двух книг по одному предмету. Перед выдачей сделать проверку наличия экземпляра книги в книгохранилище.

**Вариант 3**

Для книг, которых нет на полках, определить студента, который дольше всех держит на руках эту книгу.

**Вариант 4**

Подсчитать доход компании от продажи товаров за указанный период времени.

Доход получается за счет разницы оптовой (указана в таблице Products) и отпускной (указана в таблице OrderDetails) цен товара.

**Вариант 5**

Подсчитать расходы указанного покупателя за указанный период времени.

**Вариант 6**

Вывести список книг по физике, находящихся на руках у заданного абонента.

**Вариант 7**

Определить покупателя, который потратил больше всех денег за указанный период времени.

**Вариант 8**

Записать студента в библиотеку (обеспечить уникальность ID) и

выдать ему книгу. У студента не должно быть на руках больше одной книги по каждому из предметов.

**Вариант 9**

Определить студента, который дольше всех держит на руках экземпляр указанной книги.

**Вариант 10**

Определить предмет, по которому осталось меньше всего экземпляров книг в хранилище.

**Вариант 11**

Определить покупателя, который потратил больше всех денег на покупки товаров в предыдущем месяце.

**Вариант 12**

Определить товар, продажа которого в прошедшем году принесла магазину наибольшую прибыль.

**Вариант 13**

Снизить на 10% цену товара, который дольше всех хранится на складе в магазине.

**Вариант 14**

Определить по результатам сессии в 1-ом семестре группу кафедры ИУ5, которая лучше всех сдала экзамен по дисциплине «Основы программирования».

**Вариант 15**

Определить по результатам сессии в 1-ом семестре 10 лучших студентов кафедры ИУ5.

**Вариант 16**

Определить по результатам сессии в 1-ом семестре лучшую группу кафедры ИУ5 (оценку производить по среднему баллу по всем дисциплинам).

**Вариант 17**

Определить по результатам сессии в 1-ом семестре дисциплину, по которой студенты вашей группы имеют наименьший средний балл.

**Вариант 18**

Определить по результатам сессии в 1-ом семестре дисциплину, по которой студенты вашей группы имеют наибольший средний балл.

**Вариант 19**

Создать программу, автоматизирующую процесс разработки содержания технического документа (план документа). Под содержанием документа понимается создание 3-х уровневого оглавления документа (часть, глава, параграф) и краткого описания того, что предполагается поместить в каждый раздел, включая его размер. Вывести на печать оглавление с указанием номера раздела, его названия и номеров страниц.  
 Обеспечить автоматическое изменение нумерации разделов и страниц при перестановке разделов в оглавлении, а также возможность просмотра и изменения наименований и краткого содержания разделов.   
*Указание.*   
Имена разделов и их краткое содержание хранить в текстовом файле. Для работы с файлами использовать текстовый редактор «Блокнот», который вызывается из программы оператором *system(“notepad”),* а имя файла хранить в таблице.

**Вариант 20**

СМУ. (Стройка, строительные бригады, работники. Предусмотреть объединение и разделение бригад, добавление и удаление работников

**Вариант 21**

Отдел кадров. (отделы, штатное расписание, сотрудники). Предусмотреть операции приема на работу, увольнения, изменение окладов и должностей.

**Вариант 22**

Моделирование выполнения программ в ЭВМ. Класс ОП (список блоков занятой и свободной оперативной памяти) Предусмотреть сборку мусора (программа, которая объединяет соседние блоки памяти, которые помечены как свободные после завершения задачи).   
Класс очередей задач к ресурсам ОП и класс задач. Задача рассматривается как элемент с различными атрибутами (требуемая ОП и время счета). Момент завершения программы задается вручную, с помощью меню, и модельное время изменяется на время счета выполненной программы. Определить зависимость времени выполнения пакета программ от размера ОП, запуска программы сбора мусора и создания двух очередей: одной для программ с малым объёмом требуемой ОП, а другой для программ с большим объёмом ОП. Очереди организуются в режиме FIFO.

**Вариант 23**

Используя класс связей между таблицами, с помощью  
методов ParentRow() и ChildRows(), создать и распечатать таблицу, содержащую список книг, находящихся на руках у студента, с указанием имени и группы студента, наименования и автора книги и даты её выдачи.

## 3. Рекомендации по выполнению работы

### 3.1. Шаги разработки программы

3.1.1. Разработать и записать в текстовые файлы таблицы с данными тестовых примеров для баз данных «Библиотека» и «Компания по оптовой торговле». Для первоначальной записи данных в файл использовать текстовый редактор MS Visual C++.

3.1.2. Разработать функцию main() для тестирования функций. Для вызова тестируемых функций из main() использовать меню. Использование меню обеспечит удобство тестирования и сократит время разработки программы.

3.1.3. Разработать класс DBTable, способный хранить данные произвольных таблиц.

3.1.4. Разработать функцию readTable() для чтения данных из файла в объект класса DBTable (доработать функцию readTable() из ЛР1 так, чтобы она считывала в DBTable не только заголовки, но и строки таблицы).

Сделать вызов этой функции из меню.

3.1.5. Разработать функцию printTable() для печати данных из структуры DBTable. Сделать вызов этой функции из меню.

3.1.6. Разработать функцию writeTable() для записи данных из структуры DBTable в файл в формате CSV. Сделать вызов этой функции из меню.

3.1.7. Создать файл DBLibrary.cpp, включающий структуру Table и функции readTable(), writeTable(), printTable(), и подключить его к проекту.

3.1.9. Разработать класс DBTableSet, хранящий все таблицы БД.  
3.1.10. Разработать новый (пользовательский) тип данных DBDate для представления времени в таблицах БД (ЛР5).

3.1.11. Добавить в классы методы, которые позволят всем членам группы выполнить свои варианты заданий.

3.1.12. Создать статическую библиотеку классов DBMSLib.lib.

3.1.13. Разработать функции для выполнения индивидуальных заданий, использующие разработанный интерфейс классов.

### 3.2. Групповая разработка программ

       Групповая разработка учебного проекта позволяет ощутить реальную обстановку проектирования систем программного обеспечения. При обучении программированию слишком долго господствовал миф о том, будто программирование является родом индивидуальной деятельности. Даже самый беглый анализ показывает, что большинство проектов есть продукт коллективного труда.   
       Каждая группа должна выполнить весь проект от начала до конца.

Если программист знает, что кто-то другой будет читать его программу, то отношение к кодированию резко изменяется. Программу снабжают комментариями, делят на параграфы, она становится ясной и точной.   
       В некоторых разработках использовался метод, гарантирующий простоту программирования, — так называемая *контролируемая система программирования*.   
       В этой системе над составлением каждой программы работают по крайней мере два программиста. Первый пишет программу, второй должен просмотреть и понять ее. Эта система имеет два преимущества. Во-первых, она гарантирует получение простых и понятных программ. Во-вторых, если первый программист по какой-либо причине не может довести программу до конца, другой в состоянии заменить его.   
       Чтобы повысить эффективность разработки программ, облегчить отладку и тестирование, а следовательно, и сократить работу по созданию программ, используйте библиотеки программ. Это правильный подход к программированию, плагиат в программировании не преследуется.

       Уже на стадии проектирования должны быть установлены определенные цели. Вот некоторые из них:  
 1) высокий уровень надежности;   
       2) выполнение некоторого объема работы к определенной дате;   
       3) минимальное время разработки или минимальная стоимость;   
       4) удобство и простота эксплуатации;   
       5) эффективность (объем памяти или быстродействие);   
       6) возможность введения модификаций;   
       7) универсальность.   
       При разработке каждого проекта по программированию обычно ставят несколько целей, однако часто их не фиксируют. Иногда программисты не вполне осознают поставленную цель, а бывает и так, что два программиста, работая над одним и тем же проектом, имеют в виду различные цели. Один пытается минимизировать любой ценой объем памяти, другой — расходы на разработку.

*Устанавливайте цели проекта заблаговременно и точно.*

       Предположим, что некоторая программа должна быть закончена к январю, использована один раз, а если не будет сделана в срок, то не будет использована совсем. В данном случае ни эффективность, ни простота модификации программы не являются целями. Рассмотрим другой случай. Необходимо разработать еженедельно обновляемую программу, которая будет использоваться в течение длительного времени. Здесь существенны эффективность и простота внесения исправлений и, если программа будет сделана неделей позже, это не имеет решающего значения. Установление и описание целей гарантирует то, что все будут работать, руководствуясь едиными принципами. Однажды установленные цели должны быть упорядочены по степени важности и присвоенным весовым коэффициентам в случае, если необходимо какой-либо из них отдать предпочтение.   
       Каждому известны два наиболее важных параметра программ: память и время. Однако существует еще один параметр, который необходимо учитывать, — сложность. Чем сложнее программа, тем труднее ее контролировать, тестировать, отлаживать и эксплуатировать. Ограничив параметр сложности, мы можем получить намного легче поддающийся управлению результат. В настоящее время существуют методы ограничения сложности программ.

       Бригадный подход дает много преимуществ. Во-первых, менее квалифицированные программисты получают возможность участвовать в больших проектах и наблюдать разработку всего проекта. Сотрудничество с квалифицированными специалистами — тремя основными членами бригады — позволяет молодым сотрудникам овладеть профессиональными и техническими навыками. Все выполненные работы проверяются опытными программистами. Кроме того, так как молодые члены бригады будут читать главную программу, она будет тщательно проверена. Таким образом, программирование переходит от индивидуального умения к коллективному сотрудничеству, когда каждый анализирует успехи и ошибки как свои собственные, так и других. Среда, в которую попадают молодые сотрудники, способствует овладению профессиональным мастерством и дальнейшему продвижению. 