# **ЛР 4.** Работа с полиморфной версией библиотеки dbmsLib\_v2\_1

1. Создайте новый проект и подключите к проекту полиморфную версию библиотеки (файлы dbmsLib\_v2\_1.h и dbmsStatLib\_v2\_1.lib находятся в папке ПКШ2018\_Лаб4).  
Новая версия содержит полиморфный класс DBTableSet, который использует интерфейс абстрактного базового класса DBTable для работы с реляционными БД, таблицы которых могут размещаться как в текстовых файлах формата CSV так и в бинарных файлах с прямым доступом.

2. Используя тестовые версии БД LibraryTxt и CompanyTxt создайте с помощью методов DBTableTxt:: ReadDBtableBin() и DBTableTxt:: WriteDBtableBin бинарные версии таблиц БД и поместите текстовые и бинарные версии в папки LibraryTxt, LibraryBin, CompanyTxt и CompanyBin соответственно.

*Справка.*  
Файл – это *структурированная* последовательность байтов, то есть последовательность, организованная по определённым правилам. Знание правил организации данных необходимо для их правильного считывания.   
В методах чтения файлов, открытых в текстовом режиме, байты интерпретируются как символы и некоторые из них используются для выделения структур данных, например, символы-разделители в текстовых файлах в формате CSV, символ конца строки или символ конца файла.  
 Достоинство текстовых файлов – удобство создания, чтения и внесения изменений. Для этого имеются специальные программы - текстовые редакторы. Кроме того, использование контейнеров типа *map* (ассоциативных массивов) позволяет обращаться к данным полей таблицы по имени столбца, что значительно облегчает чтение программ.  
Недостаток текстовых таблиц БД – необходимость переформатирования данных при чтении из файла и при записи в файл.

В методах чтения файлов, открытых в бинарном режиме, таких как read() и write(), байты интерпретируются как копия байтов ОП, а для выделения структур данных используется размер структуры. Если нам нужно читать бинарные файлы с различной структурой, то размер структуры в байтах записывается в файл перед её телом. При чтении сначала в переменную *int size* считывается размер структуры (целое число – 4 байта), а затем считываются *size* байт данных структуры. Для просмотра бинарных файлов используются специальные программы - редакторы файлов.   
Достоинство бинарных таблиц БД – отсутствие необходимости переформатирования данных при чтении из файла и при записи в файл. Это достоинство будет проявляться с наибольшей эффективностью, если в ОП данные структур хранятся в последовательных ячейках памяти, друг за другом. Для реализации этого требования структуры данных следует хранить в массивах.  
Недостаток использования массивов – необходимость использования целочисленного индекса (порядкового номера структуры, а не её имени) для обращения к конкретной структуре. Это сильно ухудшает читабельность программы. Однако, этот существенный недостаток можно обойти, если для хранения данных использовать *бинарные файлы прямого доступа*. В этом случае для конкретной таблицы размер полей будет одинаковым, что позволяет достаточно легко реализовать интерфейс ассоциативного массива не на базе бинарного дерева (память динамически выделяется для каждой структуры) а на базе массива. В этом случае память выделяется сразу на всю строку таблицы, а затем из этой памяти, зная размер данных в столбцах таблицы, выбираются байты отдельных структур.  
При чтении таблицы необходимо сначала прочитать заголовок, так как содержащиеся в нём данные используются при чтении строк таблицы.

3. Добавьте в каждую БД текстовый файл с фиксированным именем DBTables.txt, содержащий перечень имен файлов с таблицами БД, и файл с именем TableRelations и расширением .txt для текстовых БД и .bin для бинарных БД, который содержит описания связей между таблицами реляционной БД.

Примеры файлов DBTables.txt и TableRelations.txt для БД LibraryTxt.

Файл DBTables.txt:   
Students.txt|Books.txt|Abonements.txt|TableRelations.txt

Файл TableRelations.txt:   
TableRelations|RelationName

ChildSecondaryKey|String|12|ChildTable|String|12|ParentPrimaryKey|String|12|ParentTable|String|12|RelationName|String|20

StudentID| Abonements| StudentID| Students| StudentsAbonements

BookID| Abonements| BookID| Books| BooksAbonements

Пример файла DBTables.txt для БД LibraryBin: Students.bin|Abonements.bin|TableRelations.bin|Books.bin

Файл TableRelations.bin в Word выглядит нечитабельным и поэтому здесь не приводится, но его можно создать из файла TableRelations.txt с помощью функции   
DBTableTxt::WriteDBtableBin().

4. Создайте тестовую программу, использующую для выбора действий функцию   
int Menu(string date, string dbName), где date и dbName – текущая дата и имя БД, с которой работает в данный момент программа. Если эти данные печатать при каждом вызове меню в строке над перечнем возможных действий, то это облегчит работу с БД.  
Организуйте так работу тестирующей программы, чтобы имена БД , текущая дата и имена таблиц (если они используются в меню) вводились бы по запросу, а путь к таблицам БД вычислялся в программе по имени БД и имени таблицы на основании соглашений по именованию и расположению папок и файлов.

5. Проверьте правильность созданных БД, вызывая из меню функции  
DBTableSet::WriteDB();  
DBTableSet::PrintDB();  
DBTableSet::ReadDB();