# Лабораторная работа ЛВП2018\_1

## Использование контейнеров STL *map* и *vector*

## Цель работы

Целью работы является приобретение студентами навыков использования шаблонов контейнеров *map* и *vector* из Стандартной Библиотеки С++.

Задачи, решаемые при выполнении лабораторной работы:

* Создание приложения myDbms для работы с таблицами БД. Приложение использует статическую библиотеку классов для макета СУБД (библиотечные файлы dbmsStatLib\_v1.lib и dbmsStatLib\_v1.h предоставляются преподавателем).
* Подключение к проекту тестовой БД «Библиотека».
* Внесение изменений в данные таблиц БД с помощью методов класса DBTableTxt из статической библиотеки классов макета СУБД. Для хранения данных в DBTableTxt используется контейнер vector<Row>.
* Разработка собственной функции ReadDBTable1() для чтения таблиц БД из файла в объект типа DBTableTxt.

Планируемое время выполнения работы- 6 часов занятий в компъютерном зале (2 часа на создание проекта + 4 часа на разработку функции ReadDBTable1) + 4 часа самостоятельной работы студента (СРС).

## 2. Порядок выполнения работы

2.1. Описание выполнения действий во всех лабораторных работах ЛВП приводится применительно к MS Visual Studio 2012.

2.2. Создайте консольное приложение myDbms и подключите к нему библиотеку dbmsStatLib\_v1 (Как это делать, смотри ПКШ2018\_ЛР1). Для выполнения операций с БД разработайте функцию *int Menu()*.

2.3. Подключите к проекту тестовые БД. Для обращения к таблицам БД разработайте ***функцию***, которая возвращает путь к файлам с таблицами относительно текущей папки проекта, используя в качестве параметров имя БД и имя таблицы. Функция использует соглашения по размещению БД, приведенные в ПКШ2018\_ЛР1.   
*!!! Обязательно сохраните копии тестовых БД.*

2.4. Проверьте работоспособность приложения myDbms, используя интерфейс класса *DBTableTxt*.  
Интерфейс описан в заголовочном файлеdbmsLib\_v1\_1.h.  
Для проверки используйте методы *ReadDBTable*, *PrintTable* и *WriteTable*.   
Приложение считается работоспособным, если оно правильно читает, распечатывает и записывает таблицы БД.  
Вся последующая *разработка библиотеки классов макета СУБД будет выполняться в приложении myDbms*.   
Чтобы исключить потери времени из-за ошибок в подключении и в содержании таблиц БД, *каждую ЛР начинайте с проверки работоспособности приложения myDbms*.

2.5. В целях освоения интерфейса класса *DBTableTxt*, выполните следующие действия:   
- измените число экземпляров книги;   
- переведите студента в другую группу;   
- добавьте новую книгу в библиотеку.

- проверьте выполненные действия с помощью функций *ReadDBTable*, *PrintTable* и *WriteTable*.   
Для выполнения различных экспериментов в процессе программирования удобно включить в меню пункт «Тестирование», при выборе которого вызывалась бы функция Testing (). Чтобы не захламлять или случайно не испортить код разрабатываемой программы, все перечисленные в п. 2.5 действия выполняйте внутри этой функции. После проверки или демонстрации результатов преподавателю, фрагменты кода в функции Testing() можно удалить или закомментировать.

2.6. Разработайте собственную функцию ReadDBTable1() для чтения таблиц БД из файла в объект типа DBTableTxt. Разработку выполняйте путем перегрузки дружественной функции  
 friend void ReadDBTable1(DBTableTxt& tab, string tabName).

Дружественные функции имеют доступ к закрытым членам классов.

Разработку делайте по шагам, контролируя выполнение каждого шага:   
- Шаг 1. Чтение имени таблицы и имени ключевого столбца;   
- Шаг 2. Чтение заголовка таблицы в поле *Header columnHeaders*, где *Header – имя типа*  
 *map<string, ColumnDesc>*;

При записи данных в *map*, контейнер можно рассматривать как массив, в котором в качестве индекса используется имя ключа (для *map* перегружена операция индексации).   
Определения типов *Header* и *ColumnDesc* приведены в заголовочном файле библиотеки *dbmsStatLib\_v1*.  
- Шаг 3. Чтение данных таблицы в поле *vector<Row> data*, где *Row – имя типа* *map<string, void\*>.*

Для преобразования читаемых из текстового файла данных из типа string   
в тип typeName, имя которого указано в заголовке столбца таблицы в CVS-файле, используйте функцию из библиотеки dbmsStatLib\_v1

void\* GetValue(string value, string columnName, Header hdr);

Для проверки правильности выполненных действий вставьте в меню строку для вызова функции ReadDBTable1() и выполняйте ее для чтения таблицы вместо вызова ReadDBTable().   
- Шаг 4. Разработайте собственную функцию   
 void\* GetValue1(string value, string columnName, Header hdr)   
и вставьте её в ReadDBTable1() вместо функции GetValue().

После выполнения каждого шага выводите таблицу на экран.   
После выполнения 1-го шага у вас должно распечататься имя таблицы и пустая таблица.  
После выполнения 2-го шага в таблице должен появиться заголовок,  
 а после выполнения 3-его шага должна распечататься заполненная таблица.

Перед внесением изменений в разрабатываемый проект «Макет СУБД» обязательно сохраняйте его текущую *работающую* версию. Чтобы не запутаться в версиях, сохраняйте их в папках с указанием даты создания и включайте номер версии в имя файла. Так как макет СУБД будет разрабатываться в течение всего семестра, это поможет вам избежать путаницы в именах файлов и связанных с ней потерь времени.

Ниже приведены фрагменты кода программы *ReadDBTable()*, выполняющей чтение таблиц БД, которые должны помочь вам в выполнении задания.

Чтение заголовка в columnHeaders.  
columnHeaders.clear();  
char \*token, \*next\_token;  
fin.getline(line, 200); //чтение заголовка  
 next\_token=line;  
 //цикл по словам (лексемам) в строке  
 ColumnDesc colHdr;

while( (token = strtok\_s( next\_token, delims, &next\_token)) != NULL )

{

strcpy\_s(colHdr.colName,token);

token = strtok\_s( next\_token, delims, &next\_token);

colHdr.colType=GetType(token);

token = strtok\_s( next\_token, delims, &next\_token);

colHdr.length=atoi(token);

strArray.push\_back(colHdr);

}

Header hdr;

for (unsigned int j = 0; j < strArray.size() ; j ++)

{

hdr[strArray[j].colName]=strArray[j];

}

SetHeader(hdr);

Чтение строк данных.

//читаем строки в line (до EOF) и записываем их в table.data

data.clear();

while (fin.getline(line, 200))

{

Row row=\*(new Row());//буфер для формирования строки таблицы

int j = 0;

token = strtok\_s( line, delims, &next\_token);

//цикл по столбцам (словам) в строке

while(token)

{

string value=token;

//добавление поля в строку с преобразованием типа

//strArray[j] - имя столбца в заголовке таблицы

row[strArray[j].colName]=GetValue(value,strArray[j].colName,columnHeaders); j=j++;//индекс следуещего столбца в векторе strArray

token = strtok\_s( next\_token, delims, &next\_token);

}

data.push\_back(row);//добавить строку данных в таблицу

## Контрольные вопросы и задания

3.1. Что такое контейнер?   
3.2. Для чего и как используются шаблоны?   
3.3. Что такое последовательный контейнер?   
3.4. Что такое ассоциативный контейнер?   
3.5. Что хранится в узлах контейнера *map*?   
3.6. Что такое итератор?   
3.7. Как получить доступ к данным в контейнере типа *vector<Row>*?   
3.8. Как получить доступ к данным в контейнере типа *Header*?

## Рекомендуемые источники информации

1. Р. Лафоре. Объектно-ориентированное программирование в С++. Издательство ПИТЕР, 2004 г. – 532с.
2. Г. Шилдт “Полный справочник по С++”: Пер. с англ.- М., Издательский дом “Вильямс”, 2006 г. – 800с
3. MSDN Library for Visual Studio 2012 (<http://msdn.microsoft.com/ru-ru/library/> , раздел: Справочник по С++.)
4. Эккель Б. Философия С++. Введение в стандартный С++. 2-е изд.- СПб.: Питер, 2004.- 572с.: ил.