**ЛР 1. Создание и использование статической библиотеки C++**

## Цель работы

Целью работы является приобретение студентами навыков выполнения в   
MS Visual Studio процессов создания и сопровождения статических библиотек и их использования в программах на C++.

Задачи, решаемые при выполнении лабораторной работы:

* Создание и использование демонстрационной статической библиотеки в соответствии с пошаговым руководством (проекты статической библиотеки и приложения находятся в одном решении).
* Использование демонстрационной библиотеки в отдельном приложении.
* Внесение изменений в демонстрационную статическую библиотеку.
* Использование изменённой библиотеки в отдельном приложении.
* Создание таблиц БД «Library», которые будут использоваться в качестве тестовых примеров в последующих ЛР.
* Создание решения DBStatLib\_v1 с приложением testDB, использующим для работы с таблицами БД базовую версию библиотеки классов для макета СУБД (библиотечные файлы dbmsStatLib\_v1.lib и dbmsStatLib\_v1.h предоставляются преподавателем).
* Демонстрация работы приложения testDB с таблицами БД (чтение, запись, печать, добавление записей, внесение изменений в поля таблиц).

Планируемое время выполнения работы- 6 часов занятий в компъютерном зале (4 часа на работу с демонстрационной программой + 2 часа на создание и отладку приложения testDB) + 3 часа самостоятельной работы студента (СРС).

## 2. Порядок выполнения работы

## 2.1 Создание и использование демонстрационной статической библиотеки в соответствии с пошаговым руководством (проекты статической библиотеки и приложения находятся в одном решении).

## Созданное решение содержит исходные коды включаемых в библиотеку классов и в дальнейшем может использоваться как «мастерская» для расширения библиотеки и внесения в неё изменений. Решение включает в себя два проекта: первый будем использовать для создания статической библиотеки (тип проекта – консольное приложение, статическая библиотека), а второй будем использовать для тестирования созданной библиотеки (тип проекта – консольное приложение, пустой проект).

***2.1.1 Пошаговое руководство по созданию в MS Visual Studio 2012 статической библиотеки (файл .lib)***

В этом пошаговом руководстве описывается создание в MS Visual Studio 2012

статической библиотеки (файл .lib) для использования с приложениями C++.

 Статические библиотеки являются хорошим способом повторного использования кода. Вместо повторного включения одних и тех же процедур в каждое приложение, требуемые функции объединяются в статическую библиотеку (файл .lib) и затем выполняется ссылка на него из приложений. Код, связанный со статической библиотекой, становится частью вашего приложения и вам не нужно создавать другой файл для использования кода функций.

В пошаговом руководстве, приведенном в [msdn.microsoft.com/ru](https://msdn.microsoft.com/ru), рассматриваются следующие задачи.

* [Создание проекта статической библиотеки](https://msdn.microsoft.com/ru-ru/library/ms235627(v=vs.110).aspx#createlibproject)
* [Добавление класса в статическую библиотеку](https://msdn.microsoft.com/ru-ru/library/ms235627(v=vs.110).aspx#addclasstolib)
* [Создание приложения консоли C++, ссылающегося на статическую библиотеку](https://msdn.microsoft.com/ru-ru/library/ms235627(v=vs.110).aspx#createapptorefthelib)
* [Использование функциональных возможностей статической библиотеки в приложении](https://msdn.microsoft.com/ru-ru/library/ms235627(v=vs.110).aspx#uselibinapp)
* [Запуск приложения](https://msdn.microsoft.com/ru-ru/library/ms235627(v=vs.110).aspx#runapp)

***Создание проекта статической библиотеки***

1. В строке меню выберите **Файл**, **Создать**, **Проект**.
2. В левой области диалогового окна **Создать проект** разверните   
   **Установленные**, **Шаблоны**,**Visual C++** и затем выберите Win32.
3. В центральной области, выберите **Консольное приложение Win32**.
4. Укажите имя для проекта, например, MathFuncsLib- в поле **Имя**.   
   Укажите имя для решения, например, StaticLibrary- в поле **Имя решения**.  
   В поле **Расположение** установите имя каталога для решения  
   Установите флажок **Создать каталог для решения**.   
   Нажмите кнопку **ОК**.
5. На странице **Обзор** диалогового окна **Мастер приложений Win32**, нажмите кнопку **Далее**.
6. На странице **Параметры приложения** в поле **Тип приложения**, выберите **Статическая библиотека.**
7. На странице **Параметры приложения** в поле **Дополнительные параметры** установите флажок **Пустой проект** и снимите флажок **Предварительно скомпилированный заголовок**.
8. Нажмите кнопку **Готово**, чтобы создать проект.

***Добавление класса в статическую библиотеку***

1. Для создания файла заголовка для добавляемых в библиотеку классов откройте в **Обозревателе решений** контекстное меню для проекта MathFuncsLib, а затем в поле **Добавить** выберите **Создать элемент**. В диалоговом окне **Добавление нового элемента** в левой панели выберите **Visual C++**, **Код**.   
   В центральной области выберите **Заголовочный файл (.h)**.  
   Укажите имя для заголовочного файла, например, MathFuncsLib.h, а затем нажмите кнопку **Добавить**. В окне проекта будет показан пустой заголовочный файл.
2. Добавьте в файл MathFuncsLib.h класс с именем MyMathFuncs, включающий общедоступные статические функции Add и Subtract. Использование спецификатора static позволит вызывать из приложений функции, не создавая объект класса MyMathFuncs, а используя только его имя для указания пространства имен.  
   Код должен выглядеть следующим образом:

// MathFuncsLib.h

namespace MathFuncs

{

class MyMathFuncs

{

public:

// Returns a + b

static double Add(double a, double b);

// Returns a - b

static double Subtract(double a, double b);

};

}

1. Чтобы создать файл источника (.cpp) для нового класса откройте в **Обозревателе решений** контекстное меню для проекта MathFuncsLib, а затем в поле **Добавить** выберите **Создать элемент**. В диалоговом окне **Добавление нового элемента** в левой панели выберите **Visual C++**, **Код**. В центральной области, выберите   
   **Файл C++ (.cpp)**. Укажите имя для источника, например, MathFuncsLib.cpp, а затем нажмите кнопку **Добавить**. В окне проекта будет показан пустой файл источника. Для реализации функциональности для MyMathFuncs код должен выглядеть следующим образом:

// MathFuncsLib.cpp

#include "MathFuncsLib.h"

using namespace std;

namespace MathFuncs

{

double MyMathFuncs::Add(double a, double b)

{

return a + b;

}

double MyMathFuncs::Subtract(double a, double b)

{

return a - b;

}

}

1. Компилировать статическую библиотеку, выбрав в строке меню **Построение**, **Построить решение**. В результате будет создана статическая библиотека, которая может использоваться другими программами

**Примечание**

***Создание приложения консоли C++, ссылающегося на статическую библиотеку***

1. В строке меню выберите **Файл**, **Создать**, **Проект**.
2. В левой панели выберите **Visual C++**, выберите **Win32**.
3. В центральной области, выберите **Консольное приложение Win32**.
4. Укажите в поле **Имя** имя проекта, например, MyExecRefsLib.   
   В раскрывающемся списке **Решение**, выберите **Добавить в решение**.   
   Нажмите кнопку **ОК**.  
   После этого новый проект будет добавлен в решение, содержащее статическую библиотеку.
5. На странице **Обзор** диалогового окна **Мастер приложений Win32**, нажмите кнопку **Далее**.
6. На странице **Параметры приложения** в поле **Тип приложения**, выберите **Консольное приложение**.
7. На странице **Параметры приложения** в поле **Дополнительные параметры**, снимите флажок **предкомпилированный заголовок** и установите флажок **Пустой проект**.
8. Нажмите кнопку **Готово**, чтобы создать проект.

Использование функциональных возможностей статической библиотеки в приложении

1. Откройте в **Обозревателе решений** контекстное меню для проекта MyExecRefsLib и добавьте в него пустое приложения консоли с именем MyExecRefsLib.cpp.
2. Для использования в консольном приложении MyExecRefsLib.cpp математических функций из статической библиотеки, необходимо связать это приложение с библиотекой MathFuncsLib, добавив в него ссылки на файлы .lib и .h библиотеки.
3. Для ссылки на файл MathFuncsLib.lib откройте в **Обозревателе решений** контекстное меню для проекта MyExecRefsLib, а затем пункт **Ссылки**. В диалоговом окне **Страницы свойств** MyExecRefsLib разверните узел **Общие свойства**, выберите **.NET Framework и ссылки** и затем нажмите кнопку **Добавить новую ссылку**.   
    В диалоговом окне **Добавить ссылку** перечислены библиотеки, на которые можно ссылаться. На вкладке **Проекты** перечисляются проекты текущего решения и все библиотеки, что они содержат. На вкладке **Проекты** установите флажок MathFuncsLib, а затем нажмите кнопку **ОК**.
4. Для ссылки на файл заголовка MathFuncsLib.h необходимо изменить путь к включаемым файлам на путь к каталогу с файлом MathFuncsLib.h.  Для этого в диалоговом окне **Страницы свойств** для MyExecRefsLib разверните узел **Свойства конфигурации**, разверните узел **C/C++**, а затем выберите **Общие**. Рядом с окном **Дополнительные каталоги включаемых файлов** откройте раскрывающийся список значений свойств, а затем выберите **Изменить**. В появившемся диалоговом окне **Дополнительные каталоги включаемых файлов** в пустом текстовом поле выполните двойной щелчок и затем нажмите кнопку с многоточием (**…**) в конце незаполненной полосы. В диалоговом окне **Выбор каталога** выберите каталог MyExecRefsLib, в котором находится файл MathFuncsLib.h, а затем нажмите кнопку **Выбор папки** для сохранения выбора и закрытия диалогового окна. В диалоговом окне **Дополнительные каталоги включаемых файлов**, нажмите кнопку **ОК.**
5. Для сохранения в проекте изменений, связанных с созданием ссылок на статическую библиотеку, в диалоговом окне **Страницы свойств**, нажмите кнопку **ОК.**
6. Теперь можно использовать функции, определенные в классе MyMathFuncs библиотеки MathFuncsLib, в созданном данного приложения. Для этого замените содержимое файла MyExecRefsLib.cpp следующим кодом:

// MyExecRefsLib.cpp

#include <iostream>

#include "MathFuncsLib.h"

using namespace std;

int main()

{

double a = 7.4;

int b = 99;

cout << "a + b = " <<

MathFuncs::MyMathFuncs::Add(a, b) << endl;

cout << "a - b = " <<

MathFuncs::MyMathFuncs::Subtract(a, b) << endl;

system(“pause”);

return 0;

}

1. Создайте исполняемый файл, выбрав **Построение**, **Построить решение** в строке меню.

Запуск приложения

1. Чтобы запустить проект MyExecRefsLib, его нужно назначить проектом по умолчанию.  
   Для этого в **Обозревателе решений** откройте контекстное меню для проекта MyExecRefsLib, а затем выберите **Назначить запускаемым проектом**.
2. Результат выполнения должен выглядеть так:

a + b = 106,4   
 a - b = -91,6

2.2 Использование демонстрационной библиотеки в отдельном проекте.

### 2.2.1 Создание проекта MyExecRefsLib\_1 Создайте проект MyExecRefsLib\_1 и добавьте в него файл MyExecRefsLib.cpp из проекта MyExecRefsLib.

2.2.2 Подключение к проекту **MyExecRefsLib\_1** демонстрационной библиотеки   
Для подключения к проекту библиотеки MathFuncsLib выполните следующие действия:  
 - добавьте к заголовочным файлам проекта MyExecRefsLib\_1 заголовочный файл MathFuncsLib.h из проекта MathFuncsLib, предварительно скопировав его в соответствующую папку проекта;

- добавьте к файлам исходного кода проекта MyExecRefsLib\_1 файл MathFuncsLib.lib из проекта статической библиотеки MathFuncsLib, предварительно скопировав его в соответствующую папку проекта;

2.2.2 Выполнение программы  
- выполните программу MyExecRefsLib.cpp в созданном проекте.

Копирование библиотечных файлов в папку проекта (а не использование ссылок на папки библиотеки) сделает программу перемещаемой и поможет избежать ошибок, связанных с поиском файлов, при компиляции программы.

## 2.3 Внесение изменений в демонстрационную статическую библиотеку.

Добавьте в библиотеку функции Multiply и Divide для выполнения операций умножения и деления чисел типа double. Для сопровождения библиотеки (внесения изменений и тестирования) используйте решение StaticLibrary.

2.4 Использование изменённой библиотеки в отдельном проекте.

Выполните в проекте MyExecRefsLib\_1 с измененной библиотекой приложение MyExecRefsLib1.cpp, в котором используются функции для выполнения сложения, вычитания, умножения и деления.

## 2.5 Создание таблиц БД «Library

Таблицы БД «Library» будут использоваться в качестве тестовых примеров в последующих ЛР.

В целях упрощения разработки и использования библиотеки 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 отдельной папке, которая содержит текстовые файлы с таблицами. Имя папки для БД «Library» - LibraryTxt.   
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 имя таблицы и имя первичного ключа (или NoPrimaryKey, если первичный ключ в таблице отсутствует);

- вторая строка файла содержит заголовки всех столбцов таблицы

- остальные строки содержат записи данных таблицы.

Заголовок каждого столбца таблицы состоит из трех слов:

<имя столбца>, <имя типа данных в столбце> и <максимальная длина 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 даты в БД. Он должен быть создан пользователем для выполнения операций с датами при работе с БД.

***База данных библиотеки*** ***Library*** содержит 3 таблицы с данными:

Abonements, Students и Books.

Имена столбцов и типы данных в столбцах таблиц БД Library:

*Таблица Students.* Имена столбцов: StudentID, Name, Group.

Типы данных в столбцах таблицы:

StudentID - int (имя типа Int32, длина 8);

Name – string (имя типа String, длина 16);

Group– string (имя типа String, длина 8);

*Таблица Books*. Имена столбцов: Author, BookID, Title, Quantity, Number, где:

Quantity – количество поступивших в библиотеку экземпляров данной книги.

Number – количество выданных экземпляров данной книги.

Типы данных в столбцах таблицы:

Author - string (имя типа String, длина 12);

BookID - int (имя типа Int32, длина 8);

Title – string (имя типа String, длина 16);

Quantity - int (имя типа Int32, длина 8);

Number - int (имя типа Int32, длина 8);

*Таблица Abonements*. Имена столбцов: StudentID, BookID, OutDate, InDate, где:

DateOut, DateIn – дата выдачи и дата возврата книги.

Типы данных в столбцах таблицы:

StudentID - int (имя типа Int32, длина 8);

BookID - int (имя типа Int32, длина 8);

OutDate - DBDate (имя типа DBDate, длина 12);

InDate - DBDate (имя типа DBDate, длина 12);

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 таблиц данными в текстовом редакторе проверяйте уникальность значений первичных ключей и ссылочную целостность данных, то есть наличие в основных (родительских) таблицах Students и Books значений идентификаторов StudentID и BookID, использованных в производной (дочерней) таблице Abonements.

Пример файла Students.txt:

Students|StudentID

Group|String|8|Name|String|16|StudentID|Int32|8

IU5-21|Ivanov|1  
IU5-21|Petrov|2

Пример файла Abonements.txt:

Abonements|NoPrimaryKey

BookID|Int32|8|DateIn|DBDate|12|DateOut|DBDate|12|StudentID|Int32|8

1|5.11.2016|6.4.2016|1

1|6.2.2016|1.1.1|2

*Примечание: При заполнении таблиц в текстовом редакторе следите за тем, чтобы в конце таблицы не было пустой строки. Она не видна, но ее наличие приведет к ошибке при чтении таблицы.*

Сделайте копии таблиц БД тестового примера, чтобы их можно было бы восстанавливать при возможной потере данных в процессе отладки.

2.6 Создание решения DBStatLib\_v1 для работы с таблицами БД

Создайте в решении DBStatLib\_v1 проект testDB и подключите к нему базовую версию статической библиотеки классов для макета СУБД (библиотечные файлы dbmsStatLib\_v1.lib и dbmsLib\_v1.h предоставляются преподавателем).   
Для подключения библиотеки скопируйте файлы dbmsStatLib\_v1.lib и dbmsLib\_v1.h в текущую папку проекта (папка, в которой находится файл testDB.cpp), а затем добавьте их в проект, как существующие.   
Поместите папку LibraryTxt с таблицами БД в ту же папку, где расположен файл запуска решения DBStatLib\_v1.sln.  
Для демонстрация работы приложения testDB с таблицами БД (чтение, запись, печать, добавление записи, внесение изменений в поля таблиц) используйте меню.

***2.7 Демонстрация работы приложения testDB***

Демонстрация работы приложения testDB с таблицами БД (чтение, запись, печать, внесение изменений в поля таблиц).

## Контрольные вопросы и задания

3.1. Как создать решение для создания и сопровождения статической библиотеки C++?   
3.2. Как подключить статическую библиотеку к консольному приложению?   
3.3. Как вызвать для выполнения метод класса?   
3.3. Используя описание класса, приведенное ниже, и приложение testDB, добавьте новую книгу в таблицу Books.   
3.4. Используя приложение testDB, измените число экземпляров добавленной в п.3.3 книги.

Приложение1

//============ Класс "Таблица БД"==============

//--------DBType-перечисление типов полей таблиц БД------

enum DBType {

NoType,

Int32,

Double,

String,

Date

};

const int LENGTH = 24;//длина имени таблицы и имени столбца.

//ColumnDesc - описание данных в столбце таблицы

struct ColumnDesc {

char colName[LENGTH];//имя столбца

DBType colType;//тип данных в столбце таблицы

int length; //максимальное число символов, допустимое

//для представления данных в столбце

};

Объявления *typedef* используются для создания более коротких и значимых имен для типов, определенных в языке, или для типов, которые объявили вы.   
Кроме того, имена *typedef* позволяют инкапсулировать детали реализации типа в случае их изменения.

В отличие от объявлений **class**, **struct**, **union** и **enum**, объявления **typedef** не вводят новый тип — они вводят новые имена для уже существующих типов.

typedef map<string, void\*> Row;

typedef map<string, ColumnDesc> Header;

//------------------- класс DBTableTxt ----------------------

class DBTableTxt{

Header columnHeaders;

string tableName;

string primaryKey;

vector<Row> data;

string fileName;

public:

DBTableTxt(){}

DBTableTxt(string tabName);

DBTableTxt(string tabName,Header hdr,string primKey);

const char\* TypeName(DBType type);

~DBTableTxt(){}

vector<int> IndexOfRecord(void\* keyValue,string keyColumnName);

string valueToString(Row& row,string columnName);

void ReadDBTable(string fileName);//fileName=path+tableName

void PrintTable(int screenWidth);

void WriteDBTable(string fileName);//fileName=path+tableName

int GetSize();

DBType GetType(char\* columnName);

Row operator[](int ind);// перегрузка оператора индексации

void SetFileName(string path);

void SetTableName(string tName);

void SetPrimaryKey(string key);

string GetFileName();

string GetTableName();

string GetPrimaryKey(){return primaryKey;}

Header GetHeader();

void SetHeader(Header& hdr);

Row CreateRow();//создать строку таблицы

Row GetRow(int index);

void AddRow(Row row,int index);//добавить строку в таблицу

friend void ReadDBTable1(DBTableTxt& tab, string fileName); //fileName=path+tableName

};

## 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.