

# Сборка и подключение библиотек

Михаил Смирнов  
Разработчик C++



# Проверка связи



Поставьте “+”, если меня видно и слышно



## Если у вас нет звука:

- убедитесь, что на вашем устройстве и на колонках включен звук
- обновите страницу вебинара (или закройте страницу и заново присоединитесь к вебинару)
- откройте вебинар в другом браузере
- перезагрузите компьютер (ноутбук) и заново попытайтесь зайти

# Михаил Смирнов

О спикере:

- В C++ разработке с 2010 года
- С 2002 года работаю в Муромском Институте Владимирского Государственного Университета
- Цифровая обработка сигналов в радиолокации и гидролокации
- Траекторная обработка для радиолокаторов ближней зоны
- Создание автоматизированного рабочего места для управления гидролокатором



# Вспоминаем прошрое занятие

**Вопрос:** что такое CMake?



# Вспоминаем прошрое занятие

**Вопрос:** что такое CMake?

**Ответ:** это инструмент для абстрагирования  
от конкретных инструментов сборки



# Вспоминаем прошрое занятие

**Вопрос:** как называется специальный файл, в котором описывается проект CMake?



# Вспоминаем прошрое занятие

**Вопрос:** как называется специальный файл, в котором описывается проект CMake?

**Ответ:** CMakeLists.txt



# Вспоминаем прошрое занятие

**Вопрос:** приведите пример команд для определения минимальной версии и названия проекта





# Вспоминаем прошрое занятие

**Вопрос:** приведите пример команд для определения минимальной версии и названия проекта

**Ответ:** `cmake_minimum_required(VERSION 3.22.0)`  
`project(MyProject)`



# Вспоминаем прошрое занятие

**Вопрос:** как называется команда для  
создания исполняемой программы?



# Вспоминаем прошрое занятие

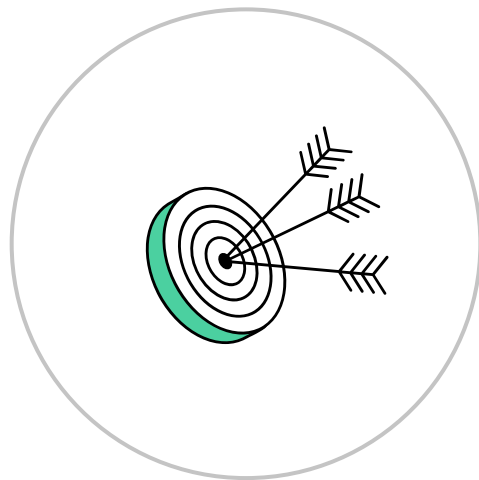
**Вопрос:** как называется команда для  
создания исполняемой программы?

**Ответ:** `add_executable`



# Цели занятия

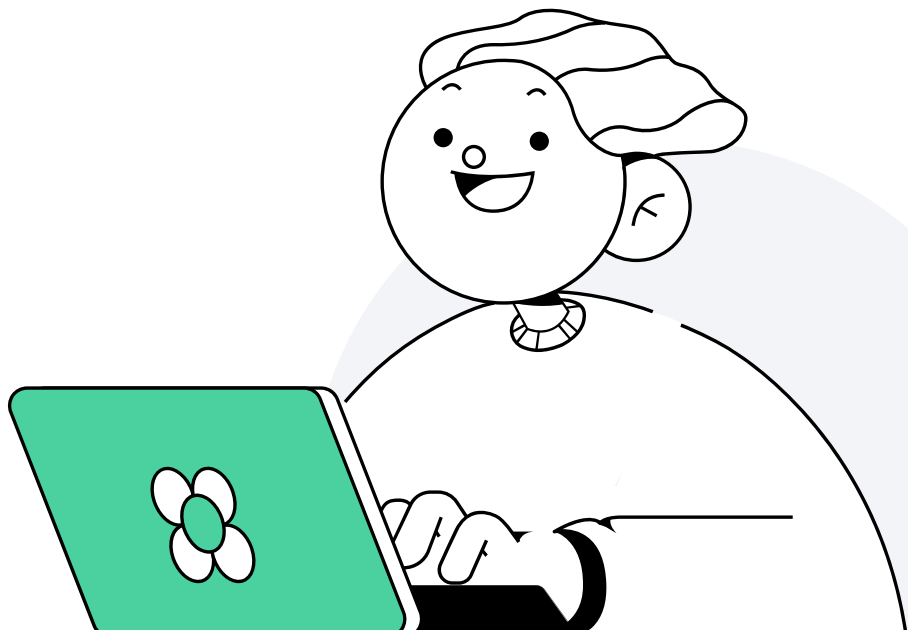
- Разберёмся, что такое библиотеки и зачем они нужны
- Познакомимся с разными видами библиотек
- Узнаем, как работать со статическими библиотеками
- Выясним, как работать с динамическими библиотеками



# План занятия

- 1 Библиотеки
- 2 Статические библиотеки
- 3 Динамические библиотеки
- 4 Итоги
- 5 Домашнее задание

\*Нажми на нужный раздел для перехода



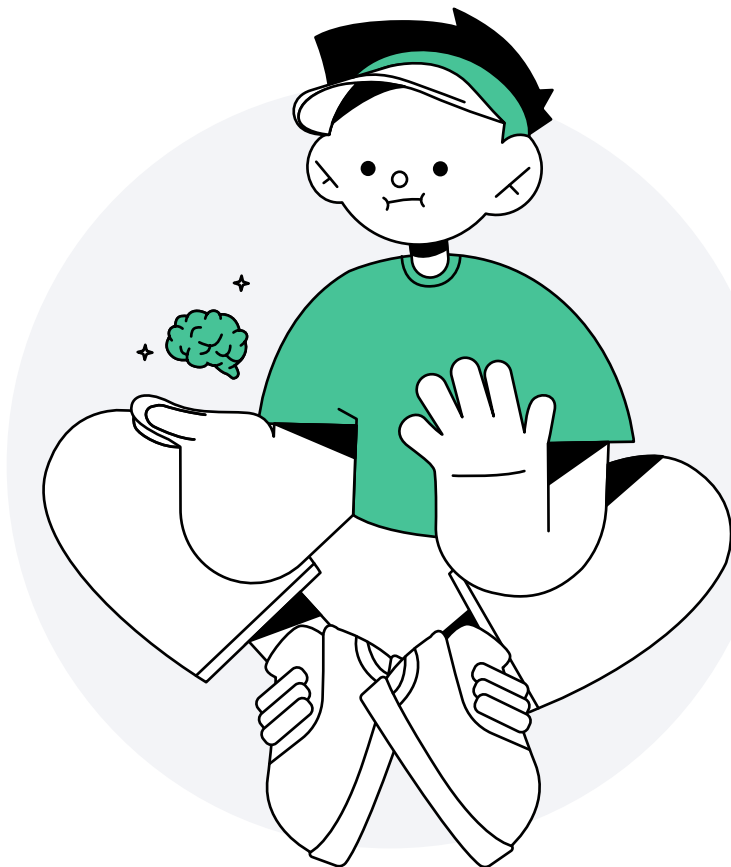
# Библиотеки



1

# Библиотеки

Мы уже несколько раз сталкивались с таким понятием, как библиотеки. Что же это такое?





**Библиотека – это отдельный модуль, который содержит в себе полезную функциональность. Она может существовать в виде классов, функций, переменных**



Как вы думаете,  
зачем нужны библиотеки?

Напишите в чат

# Зачем нужны библиотеки

Библиотеки нужны для того, чтобы не писать один и тот же код два раза

Представьте, что вы создали набор полезных классов, решающих какую-то задачу - например, классы для решения разных уравнений

Вы создали их для того, чтобы использовать в консольном приложении. Но затем вам понадобилось создать ещё и графическое приложение, в котором тоже нужно решать уравнения

# Зачем нужны библиотеки

Если не использовать библиотеки, то варианты такие:

- написать тот же код заново
- скопировать файлы из предыдущего решения

# Зачем нужны библиотеки

Писать тот же код снова - идея заведомо плохая

Скопировать файлы кажется простым и быстрым решением. Но если в процессе работы вы поймёте, что вам нужно что-то в ваших классах доделать, обнаружите ошибку - исправлять её придётся несколько раз, потому что вы скопировали файлы



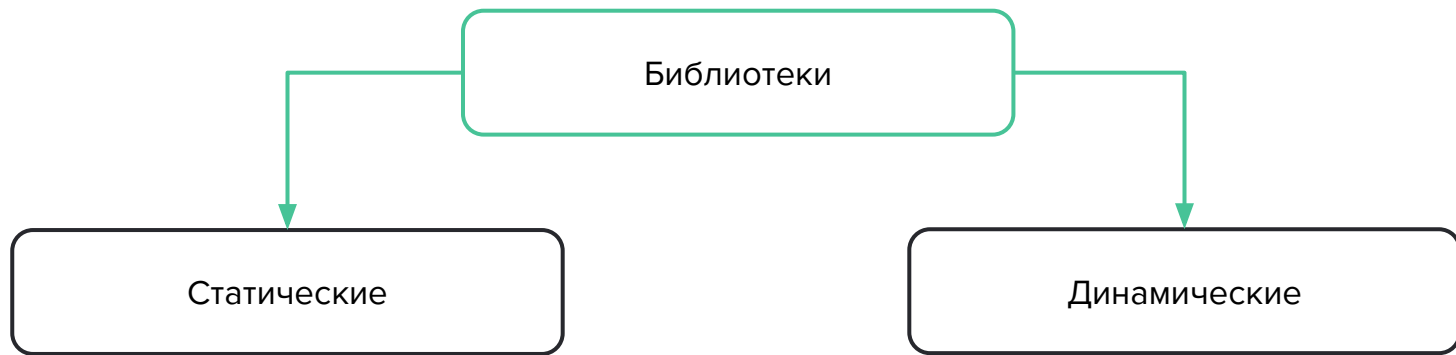
# Зачем нужны библиотеки

В случае с библиотекой вы компилируете ваши классы в отдельный **модуль**, который затем можно **подключать** к разным проектам - и функциональность будет доступна

В случае, если понадобится внести изменения - вы вносите их один раз, компилируете новую версию библиотеки и обновляете её в используемых проектах

# Какие бывают библиотеки

В C++ библиотеки делятся на два вида: **статические** и **динамические**



# Статические библиотеки

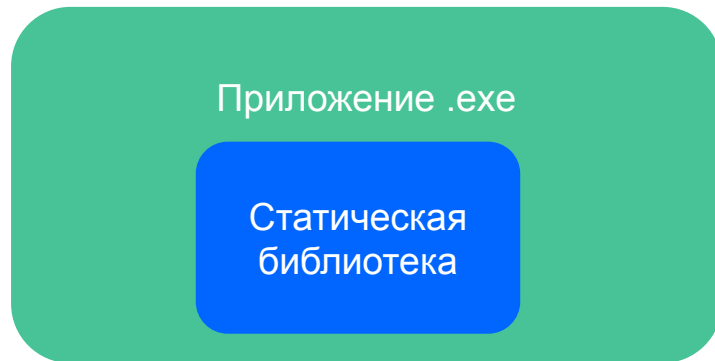


2

# Статические библиотеки

Статические библиотеки компилируются в файлы с расширением **.lib** на ОС Windows

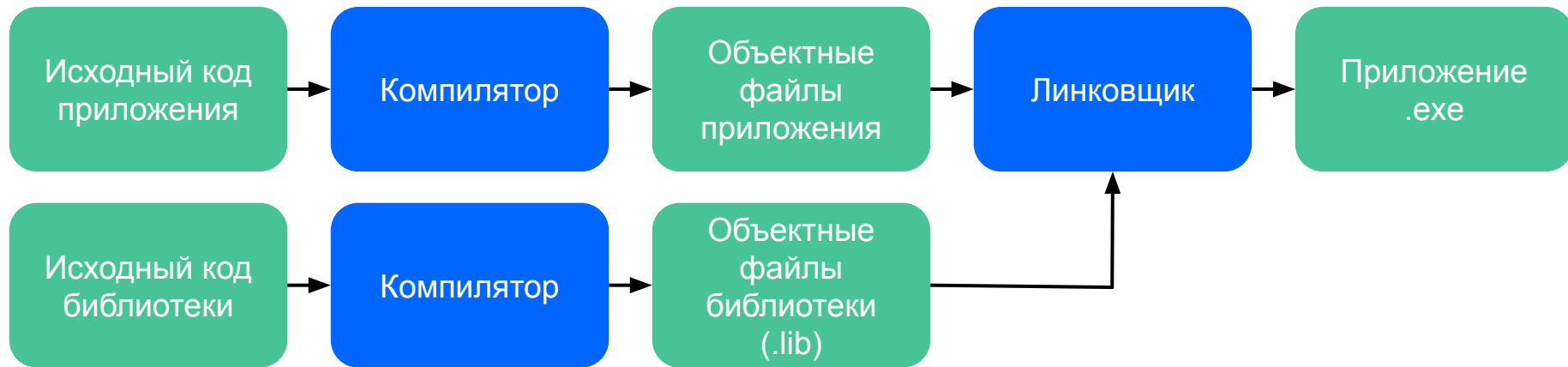
Особенность статических библиотек заключается в том, что они становятся **частью** приложения, которое её подключает - то есть она содержится **внутри** файла с расширением .exe. Файл .lib должен быть скомпилирован перед тем, как будет происходить компиляция исполняемого файла .exe





# Процесс сборки программы

Сборка программы со статической библиотекой осуществляется по следующей схеме:



**Как вы думаете,**  
какие преимущества и недостатки  
у статических библиотек?

**Напишите в чат**

# Недостатки и преимущества

Из-за того, что статическая библиотека встраивается в приложение – становится легче его распространять, ведь не нужно тащить за собой кучу файлов

Однако, если ваша библиотека большая и тяжёлая, то и размер вашего конечного исполняемого файла тоже вырастет

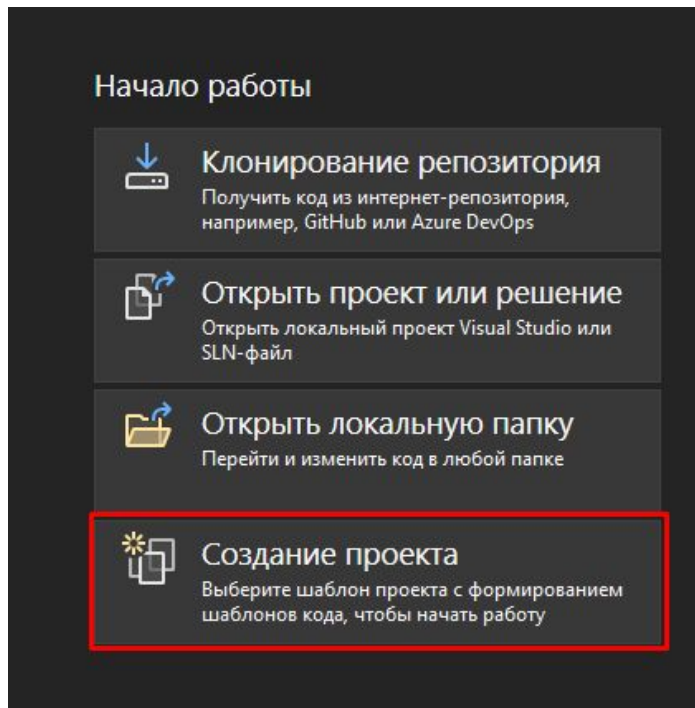
# Недостатки и преимущества

Если вам понадобится обновить библиотеку - например, вы нашли ошибку, хотите её исправить и заменить библиотеку у клиента, для которого вы разработали приложение – вам нужно будет доставить ему новый исполняемый файл для всего приложения

В случае, если вашей библиотекой пользуются сразу несколько модулей в вашем решении (а такое бывает часто), то библиотека будет дублироваться в каждом модуле

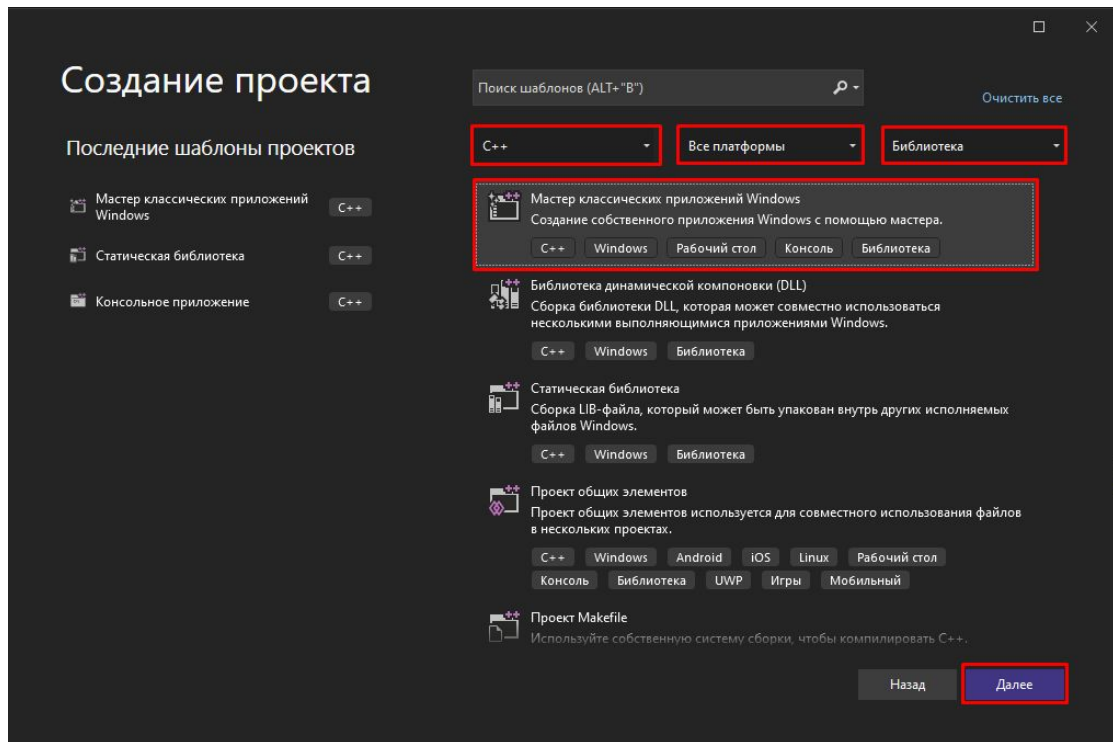
# Статическая библиотека в Visual Studio

Давайте создадим статическую библиотеку в Visual Studio. Для этого откройте начальное окно и выберите пункт **“Создание проекта”**



# Статическая библиотека в Visual Studio

В открывшемся окне “Создание проекта” установите следующие параметры: C++, Все платформы, Библиотека. В отфильтрованном списке шаблонов выберите **“Мастер классических приложений Windows”** и нажмите **“Далее”**



# Статическая библиотека в Visual Studio

В появившемся окне “Настроить новый проект” дайте имя своему проекту - **“MathPowerLibraryStatic”**. Укажите расположение проекта и нажмите кнопку **“Создать”**

Настроить новый проект

Мастер классических приложений Windows C++ Windows Рабочий стол Консоль Библиотека

Имя проекта

MathPowerLibraryStatic

Расположение

D:\Projects\Education

Решение

Создать новое решение

Имя решения ⓘ

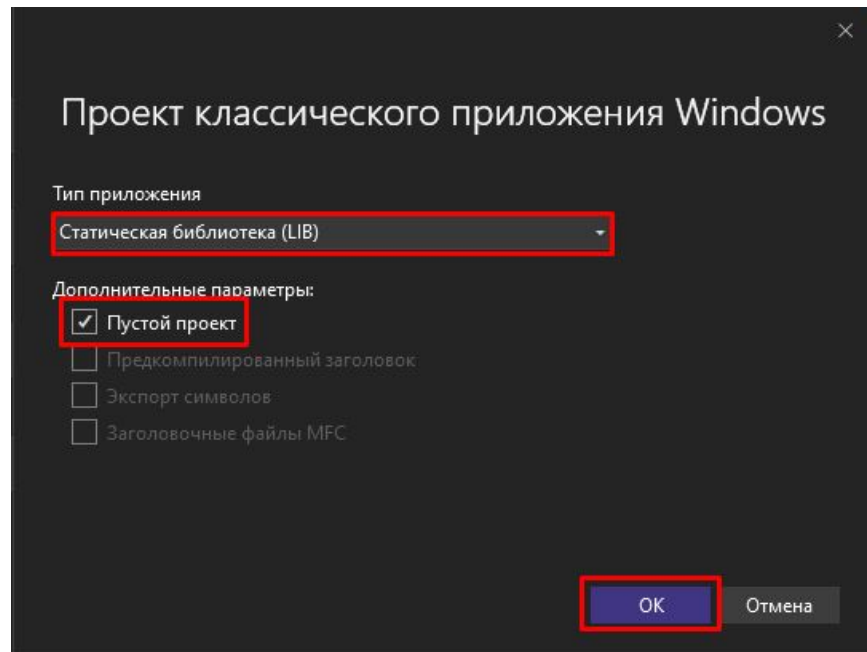
MathPowerLibraryStatic

☐ Поместить решение и проект в одном каталоге

Назад Создать

# Статическая библиотека в Visual Studio

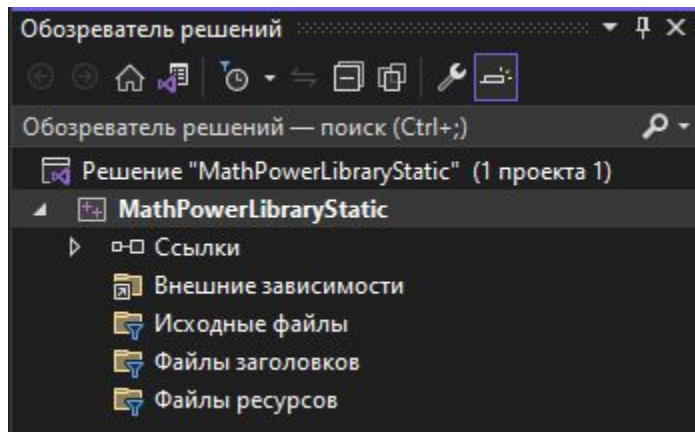
В появившемся окне “Проект классического приложения Windows” выберите тип приложения **“Статическая библиотека (LIB)”**, отметьте галочку **“Пустой проект”** и нажмите **“Ок”**





# Статическая библиотека в Visual Studio

После этого в Обозревателе решений у вас появится структура:



# Статическая библиотека в Visual Studio

Создадим в проекте MathPowerLibraryStatic заголовочный файл **MathPower.h**, в котором разместим объявление класса **Calculator** в пространстве имён **MathPower** с методами суммы и разности

```
// файл MathPower.h
#pragma once
namespace MathPower
{
    class Calculator
    {
    public:
        int Sum(int a, int b);
        int Sub(int a, int b);
    };
}
```

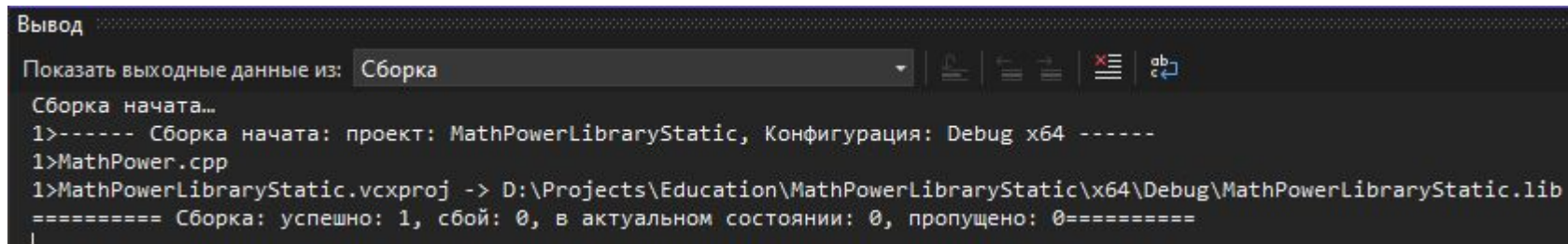
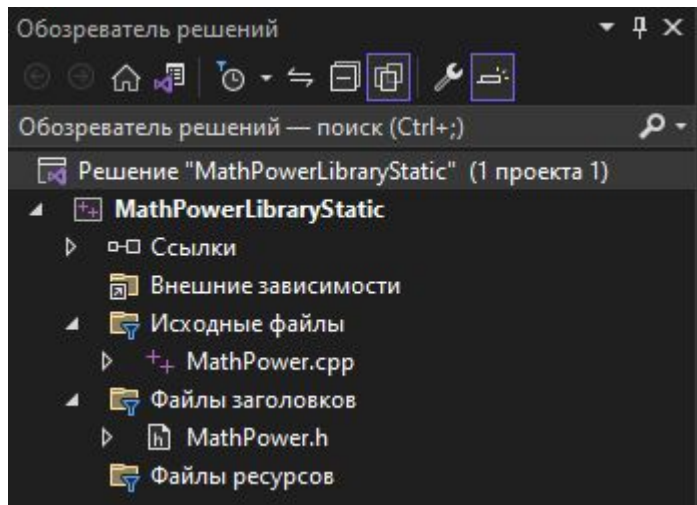
# Статическая библиотека в Visual Studio

Создадим в проекте MathPowerLibraryStatic файл исходного кода **MathPower.cpp**, в котором предоставим реализацию для методов суммы и разности

```
// файл MathPower.cpp
#include "MathPower.h"
namespace MathPower
{
    int Calculator::Sum(int a, int b)
    {
        return a + b;
    }
    int Calculator::Sub(int a, int b)
    {
        return a - b;
    }
}
```

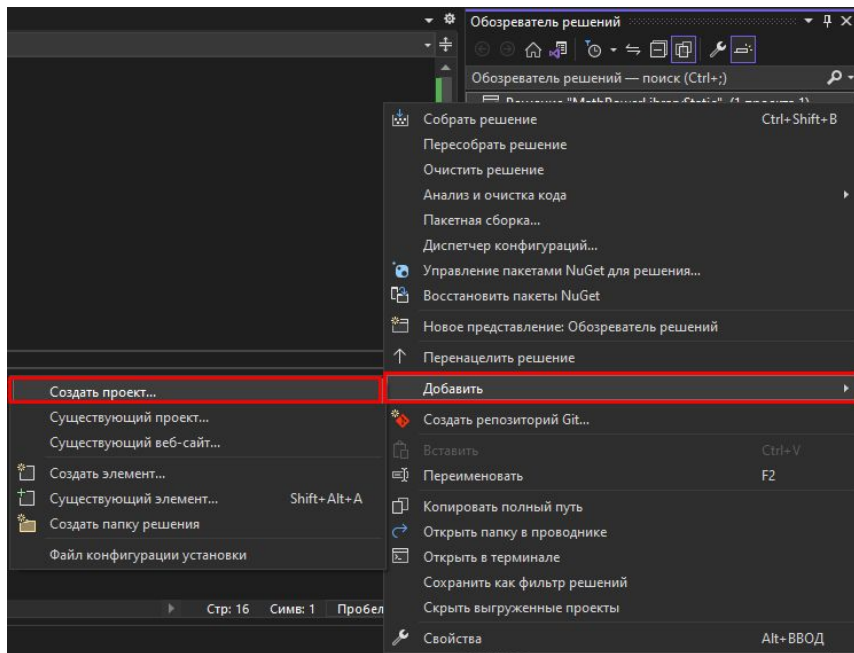
# Статическая библиотека в Visual Studio

В итоге у нас должна получиться вот такая структура. Соберите решение, убедитесь, что сборка прошла успешно



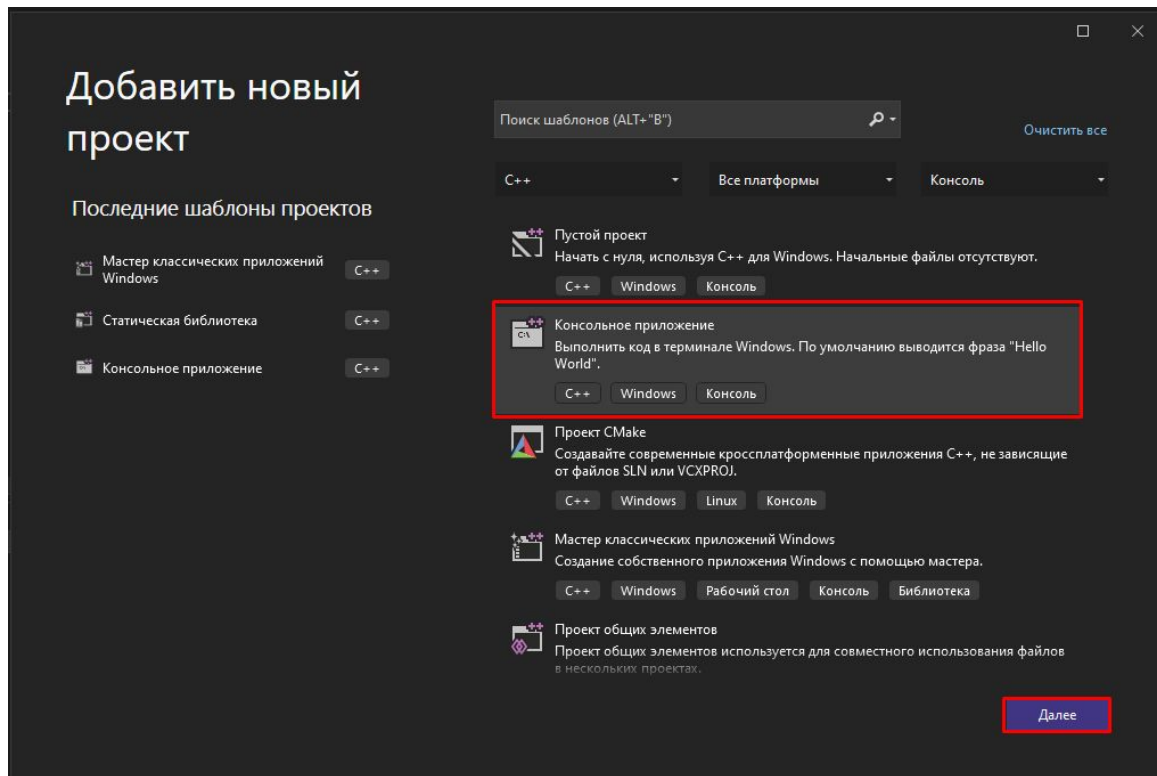
# Статическая библиотека в Visual Studio

Теперь создадим приложение, которое будет использовать нашу библиотеку. В Обозревателе решений щёлкните правой кнопкой мыши по решению (**Решение “MathPowerLibraryStatic”**), выберите пункт **“Добавить” -> “Создать проект”**



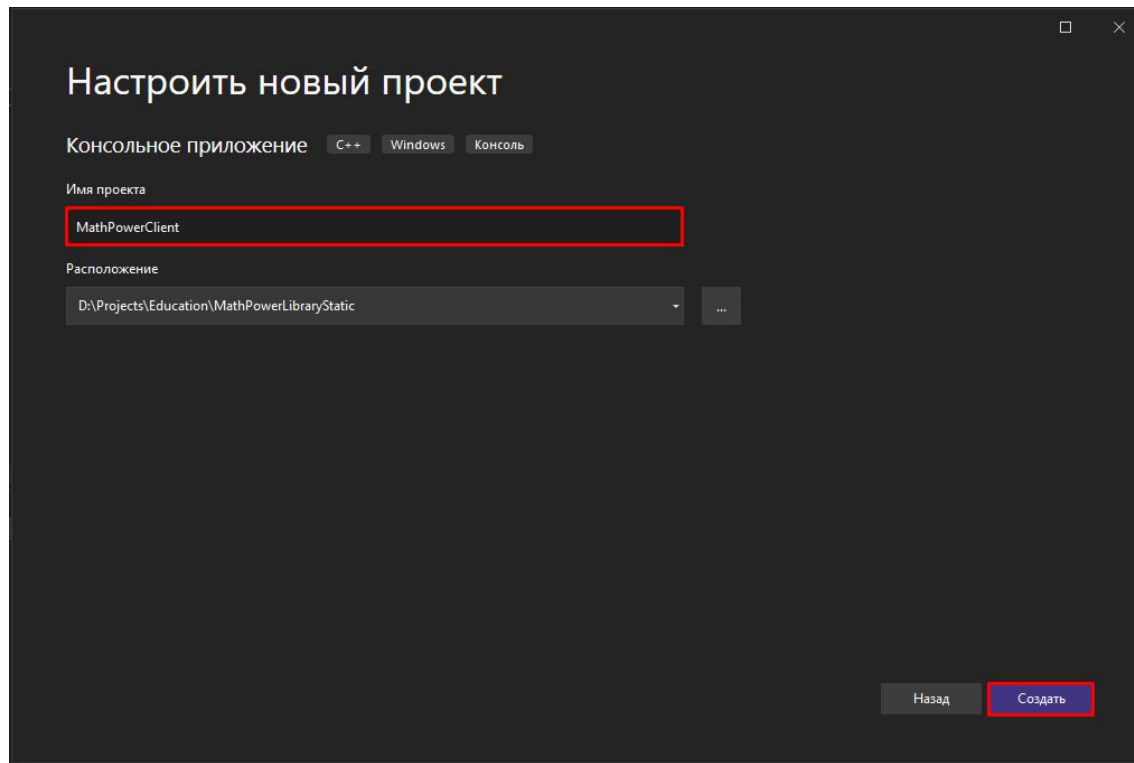
# Статическая библиотека в Visual Studio

В появившемся окне выберите настройки C++, Все платформы и Консоль. Выберите шаблон **“Консольное приложение”** и нажмите кнопку **“Далее”**



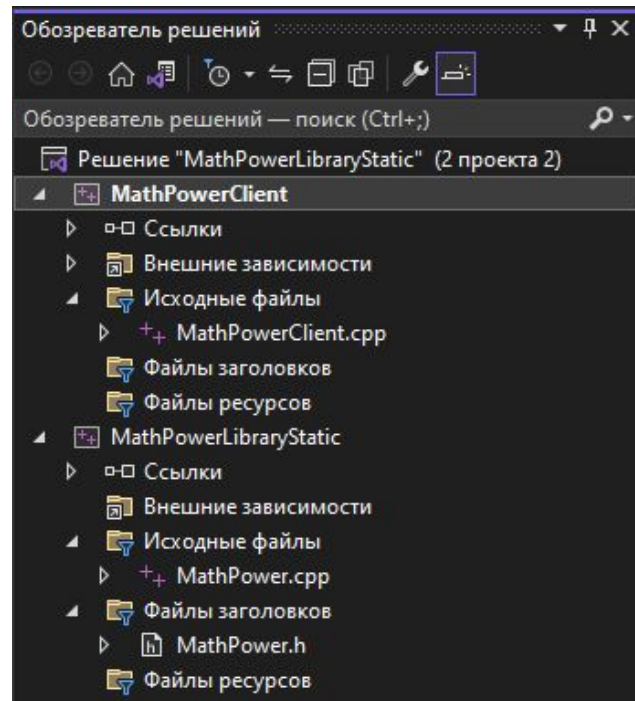
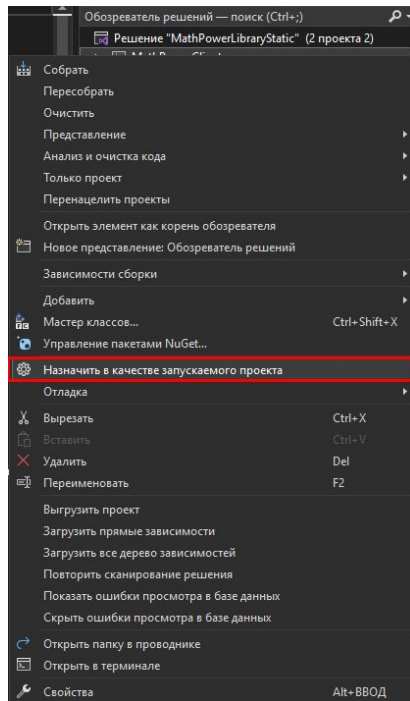
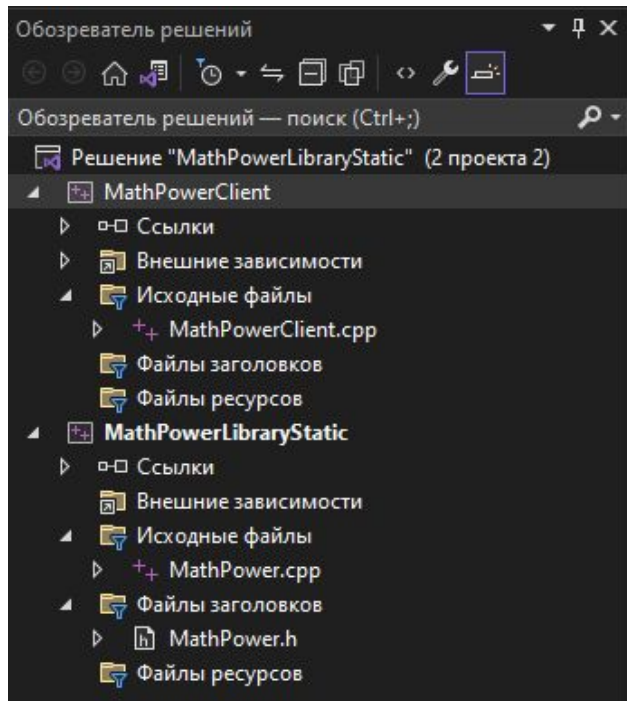
# Статическая библиотека в Visual Studio

Назовите новый проект “**MathPowerClient**” и нажмите кнопку “**Создать**”



# Статическая библиотека в Visual Studio

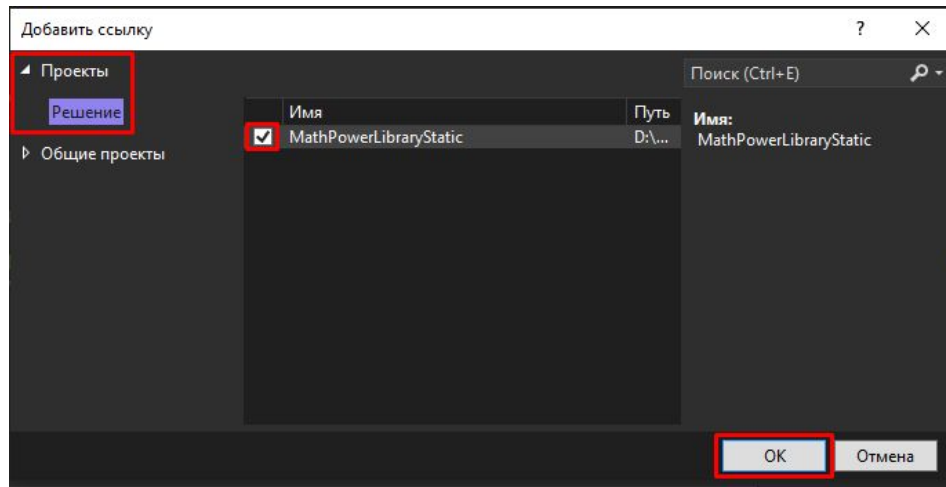
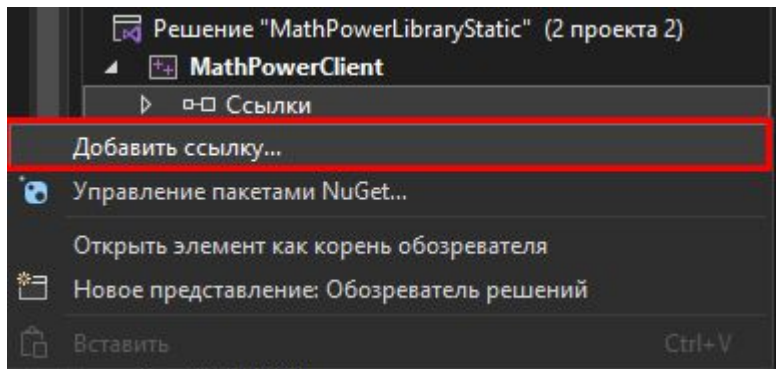
После проделанных операций структура вашего проекта будет выглядеть следующим образом. Щёлкните правой кнопкой мыши по проекту **MathPowerClient** и выберите “Назначить в качестве запускаемого проекта”





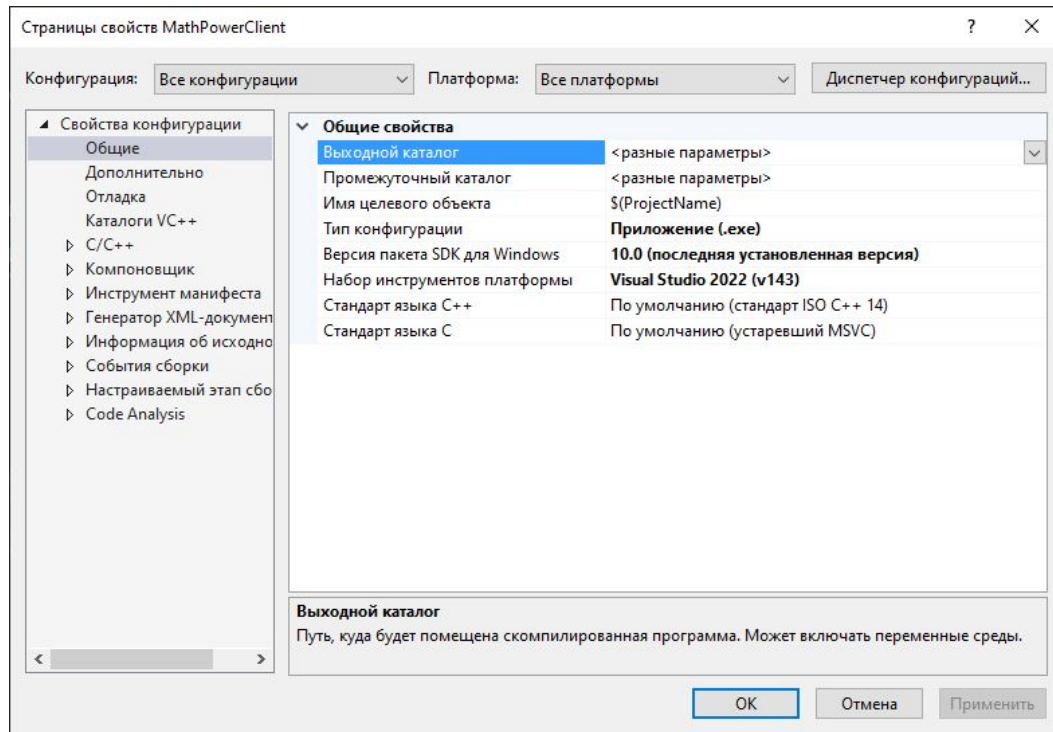
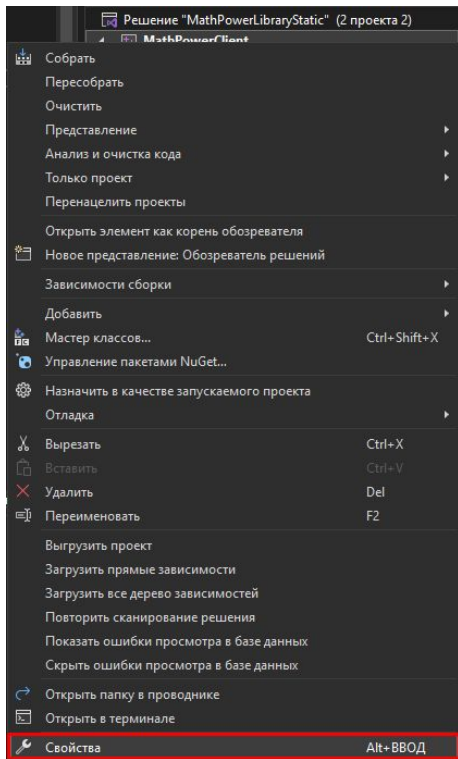
# Статическая библиотека в Visual Studio

Для того, чтобы можно было воспользоваться библиотекой MathPowerLibraryStatic в проекте MathPowerClient, нам нужно указать, что MathPowerClient **зависит** от MathPowerLibraryStatic. Для этого щёлкните правой кнопкой мыши по пункту **“Ссылки”** в проекте MathPowerClient, выберите пункт **“Добавить ссылку”**. В появившемся окне в пункте **Проекты -> Решение** поставьте **галочку** рядом с проектом **MathPowerLibraryStatic** и нажмите **“OK”**



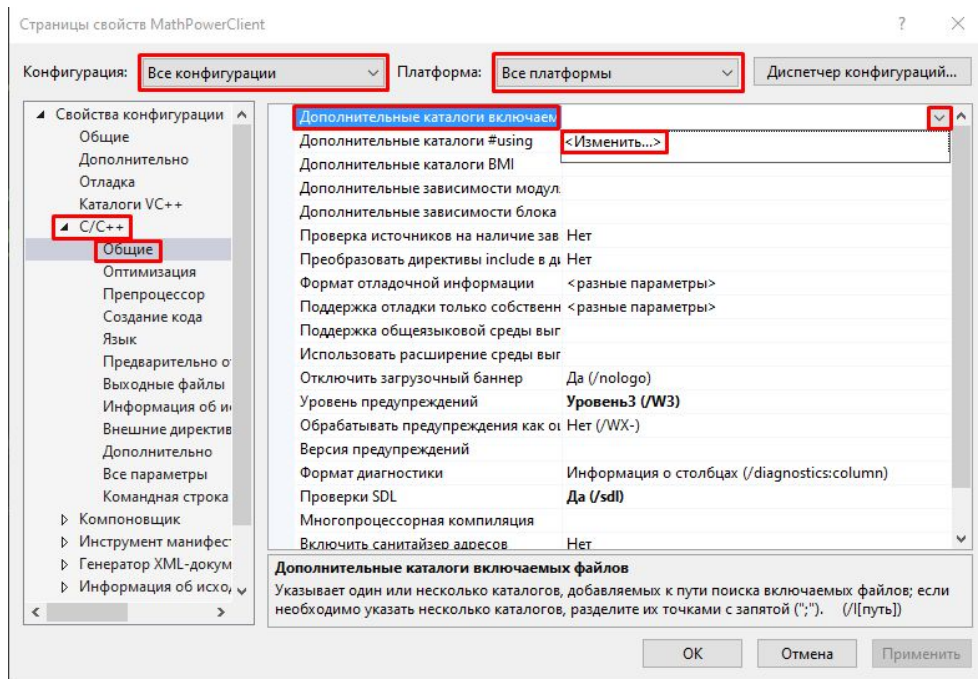
# Статическая библиотека в Visual Studio

Помимо добавления ссылки нужно также сообщить проекту MathPowerClient, **где** он может найти **заголовочные файлы** проекта **MathPowerLibraryStatic** для включения. Для этого щёлкните правой кнопкой по проекту **MathPowerClient** и выберите пункт **“Свойства”**. Откроется такое окно



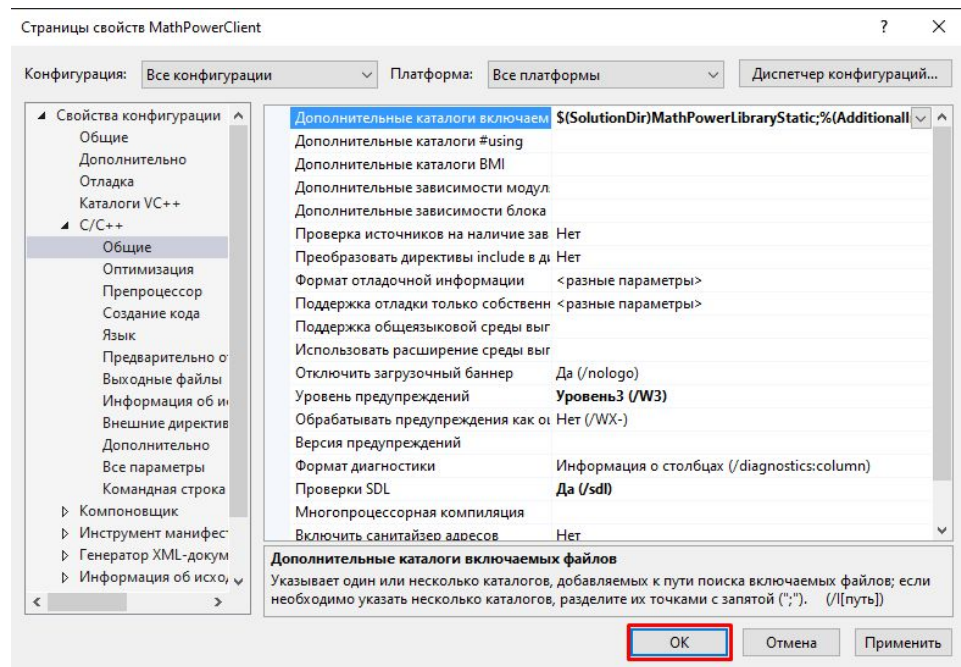
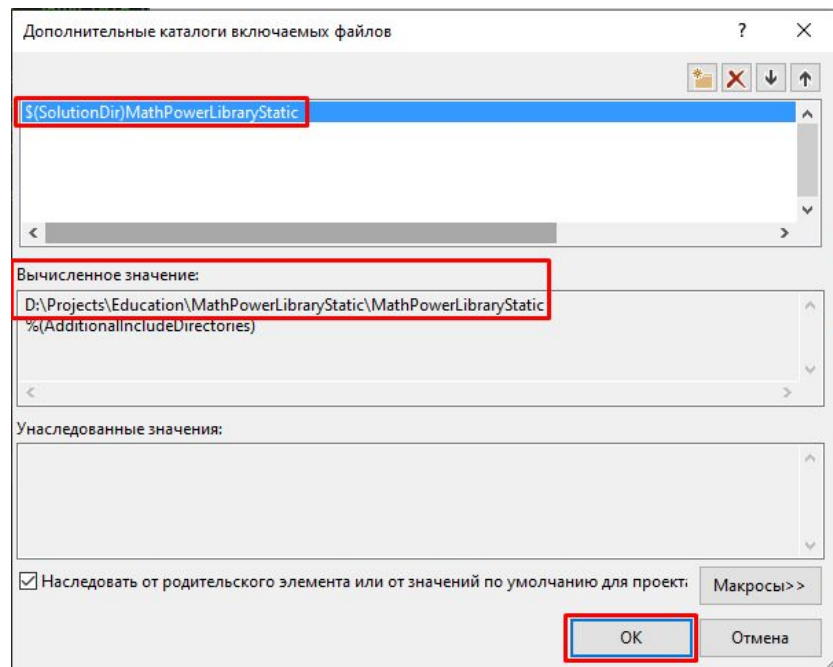
# Статическая библиотека в Visual Studio

В появившемся окне убедитесь, что в поле **“Конфигурация”** у вас выбрано значение **“Все конфигурации”**, а в поле **“Платформа”** выбрано **“Все платформы”**. Перейдите в пункт меню **“С/С++” -> “Общие”**. Нажмите левой кнопкой мыши по пустому полю в первом пункте (**“Дополнительные каталоги включаемых файлов”**), нажмите на **галочку сбоку** и щёлкните по пункту **“<Изменить...>”**



# Статическая библиотека в Visual Studio

Появится ещё одно окно. В нём щёлкните левой кнопкой мыши по пустому текстовому полю и введите строку **\$(SolutionDir)MathPowerLibraryStatic**. Проверьте, что в директории, которая отобразилась в поле **“Вычисленное значение”**, действительно находится заголовочный файл **MathPower.h**. Нажмите **“OK”**, в предыдущем окне тоже нажмите **“OK”**




# Статическая библиотека в Visual Studio

Откройте файл **MathPowerClient.cpp** и подключите заголовочный файл **MathPower.h**. Используйте класс **Calculator** из нашей библиотеки и убедитесь, что всё работает. Готово

```
#include <iostream>
#include "MathPower.h"

int main()
{
    MathPower::Calculator calculator;
    int a = 8, b = 5;
    std::cout << a << " + " << b << " = " << calculator.Sum(a, b) << std::endl;
    std::cout << a << " - " << b << " = " << calculator.Sub(a, b) << std::endl;
}
```

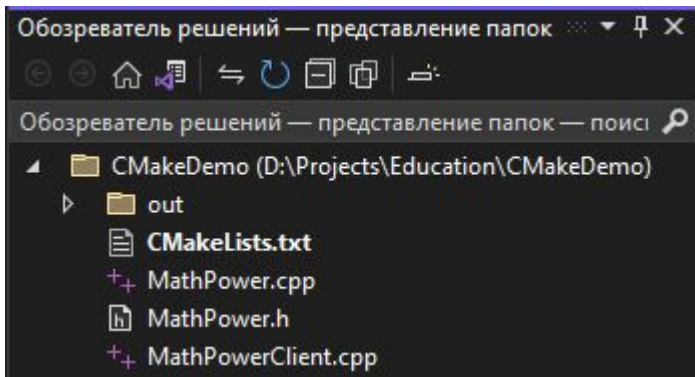
 Консоль отладки Microsoft Visual Studio

8 + 5 = 13

8 - 5 = 3

# Статическая библиотека в CMake

С CMake всё немного проще, так как он сам генерирует нужные проекты - вам нужно только сказать, что вы от него хотите. Для исходных файлов, расположенных в одной директории (см. скриншот), в файл CMakeLists.txt нужно будет добавить две новые команды - **add\_library** и **target\_link\_libraries**. **add\_library** подобна **add\_executable** за исключением указания типа библиотеки, а **target\_link\_libraries** связывает библиотеку и исполняемое приложение



```
nt.cpp  MathPower.cpp  MathPower.h  CMakeLists.txt  Страница 1 из 1

cmake_minimum_required(VERSION 3.22.0)
project(MathPower)
add_library(MathPowerStaticLib STATIC MathPower.h MathPower.cpp)
add_executable(MathPowerExe MathPowerClient.cpp)
target_link_libraries(MathPowerExe MathPowerStaticLib)
```

# Динамические библиотеки



3

# Динамические библиотеки

Динамические библиотеки компилируются в файлы с расширением **.dll** на ОС Windows. На этой ОС к ним также компилируется маленькая статическая библиотека **.lib**

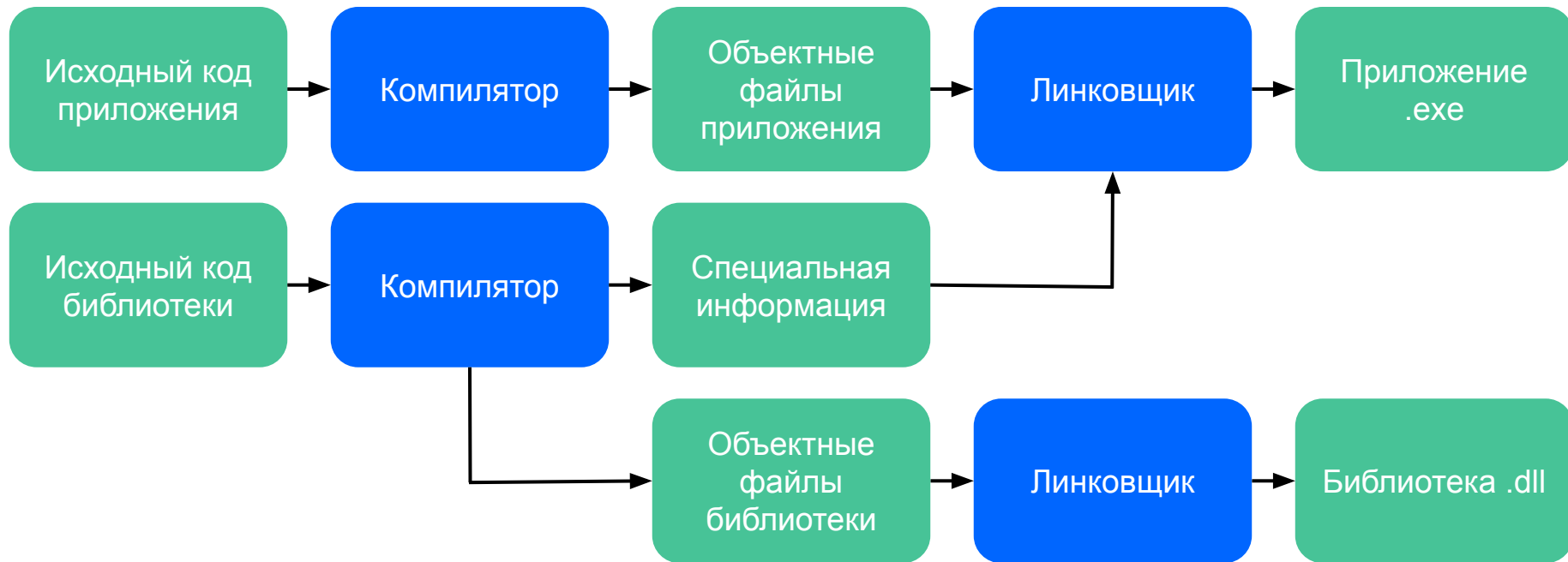
Особенность динамических библиотек заключается в том, что они находятся в **своём собственном файле** - то есть она содержится **снаружи** файла с расширением **.exe**. Файл **.dll** не связан жёстко с приложением **.exe** - он загружается в процессе работ приложения





# Процесс сборки программы

Сборка программы со статической библиотекой осуществляется по следующей схеме:



**Как вы думаете,**  
какие преимущества и недостатки  
у динамических библиотек?

**Напишите в чат**

# Недостатки и преимущества

Из-за того, что динамическая библиотека находится отдельно от приложения – становится сложнее его распространять, так как потеря или неправильная версия файла библиотеки приведёт к неправильной работе программы

Однако, если ваша библиотека большая и тяжёлая, размер вашего конечного исполняемого файла останется прежним

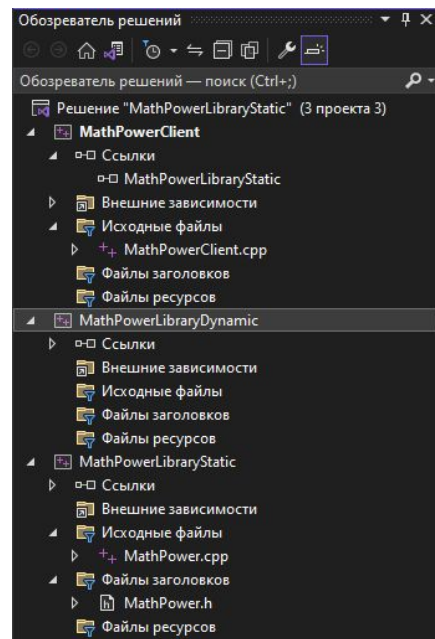
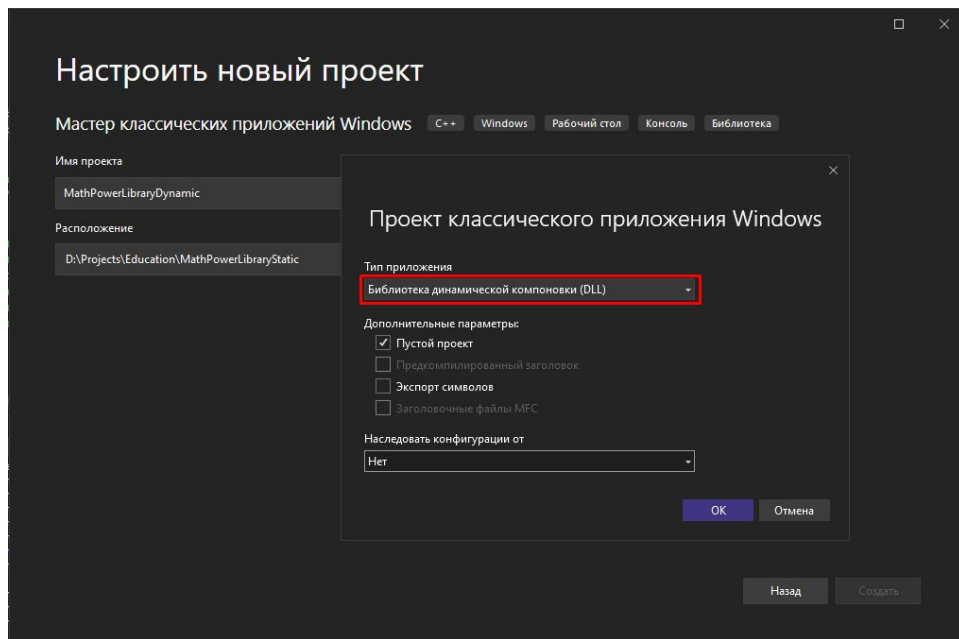
# Недостатки и преимущества

Если вам понадобится обновить библиотеку - например, вы нашли ошибку, хотите её исправить и заменить библиотеку у клиента, для которого вы разработали приложение – вам нужно будет всего лишь предоставить ему новую (обратно совместимую) версию вашей библиотеки в виде файла .dll - остальные файлы менять не нужно

В случае, если вашей библиотекой пользуются сразу несколько модулей в вашем решении (а такое бывает часто), то все модули будут пользоваться одним и тем же файлом библиотеки, что положительно скажется на размере и поддержке

# Динамическая библиотека в Visual Studio

Давайте создадим динамическую библиотеку в Visual Studio. Для этого добавим в наше предыдущее решение ещё один проект – “**MathPowerLibraryDynamic**”. Добавьте в решение пустой проект с помощью знакомого нам шаблона “Мастер классических приложений Windows”, но в типе приложения выберите “**Библиотека динамической компоновки (DLL)**”



# Динамическая библиотека в Visual Studio

Создадим в проекте MathPowerLibraryDynamic заголовочный файл **DynamicMathPower.h**, в котором разместим объявление класса **DynamicCalculator** в пространстве имён **DynamicMathPower** с методом умножения. В заголовочных файлах библиотек DLL используется специальный синтаксис

```
// файл DynamicMathPower.h
#pragma once
#ifdef MATHPOWERLIBRARYDYNAMIC_EXPORTS
#define MATHPOWERLIBRARY_API __declspec(dllexport)
#else
#define MATHPOWERLIBRARY_API __declspec(dllimport)
#endif
namespace DynamicMathPower
{
    class DynamicCalculator
    {
    public:
        MATHPOWERLIBRARY_API int Mult(int a, int b);
    };
}
```

# Динамическая библиотека в Visual Studio

Специальный синтаксис нужен потому, что компилятор Visual C++ по умолчанию не экспортирует ничего из библиотеки – а это сделать нужно, так как библиотекой будут пользоваться внешние по отношению к ней модули

Конструкция из прошлого слайда позволяет разделить два сценария – когда заголовочный файл используется при сборке библиотеки (в проекте библиотеки DLL Visual Studio автоматически определяет макрос <ИМЯПРОЕКТА>\_EXPORTS) и когда он используется при сборке модуля-клиента (тогда макрос не определён и символы не экспортируются, а импортируются)

# Динамическая библиотека в Visual Studio

Общее правило для библиотек DLL – размещать такой макрос (здесь это `MATHPOWERLIBRARY_API`) в заголовочных файлах перед всем, что должно быть доступно внешним модулям (глобальные функции, методы и поля класса – если поставить её перед целым классом, то будет экспортирован весь класс со всеми членами)



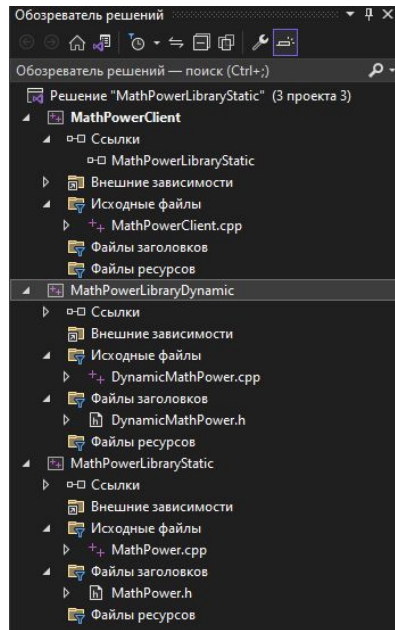
# Динамическая библиотека в Visual Studio

Создадим в проекте MathPowerLibraryDynamic файл исходного кода **DynamicMathPower.cpp**, в котором предоставим реализацию для метода умножения

```
// файл DynamicMathPower.cpp
#include "DynamicMathPower.h"
namespace DynamicMathPower
{
    int DynamicCalculator::Mult(int a, int b)
    {
        return a * b;
    }
}
```

# Динамическая библиотека в Visual Studio

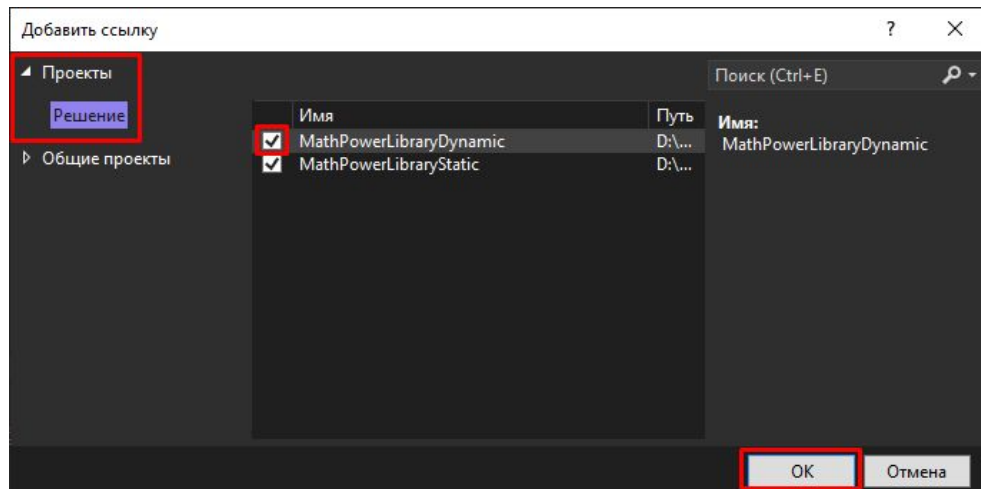
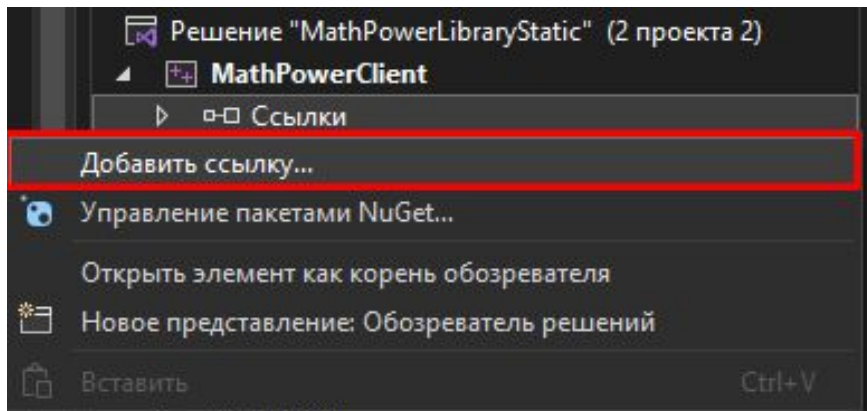
В итоге у нас должна получиться вот такая структура. Соберите решение, убедитесь, что сборка прошла успешно



```
Вывод
Показать выходные данные из: Сборка
Сборка начата...
1>----- Сборка начата: проект: MathPowerLibraryDynamic, Конфигурация: Debug x64 -----
1>DynamicMathPower.cpp
1>MathPowerLibraryDynamic.vcxproj -> D:\Projects\Education\MathPowerLibraryStatic\x64\Debug\MathPowerLibraryDynamic.dll
===== Сборка: успешно: 1, сбой: 0, в актуальном состоянии: 2, пропущено: 0=====
```

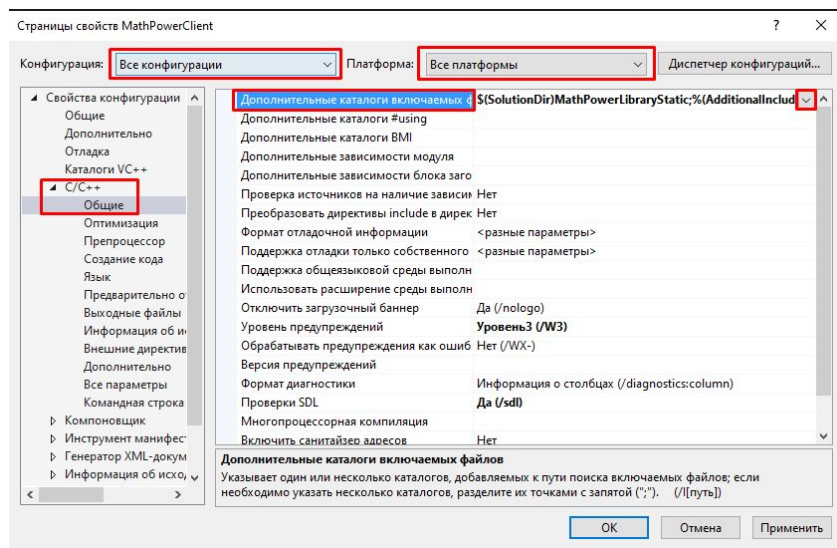
# Динамическая библиотека в Visual Studio

Для того, чтобы можно было воспользоваться библиотекой MathPowerLibraryDynamic в проекте MathPowerClient, нам нужно указать, что MathPowerClient **зависит** от MathPowerLibraryDynamic. Для этого щёлкните правой кнопкой мыши по пункту **“Ссылки”** в проекте MathPowerClient, выберите пункт **“Добавить ссылку...”**. В появившемся окне в пункте **Проекты -> Решение** поставьте **галочку** рядом с проектом **MathPowerLibraryDynamic** и нажмите **“OK”**



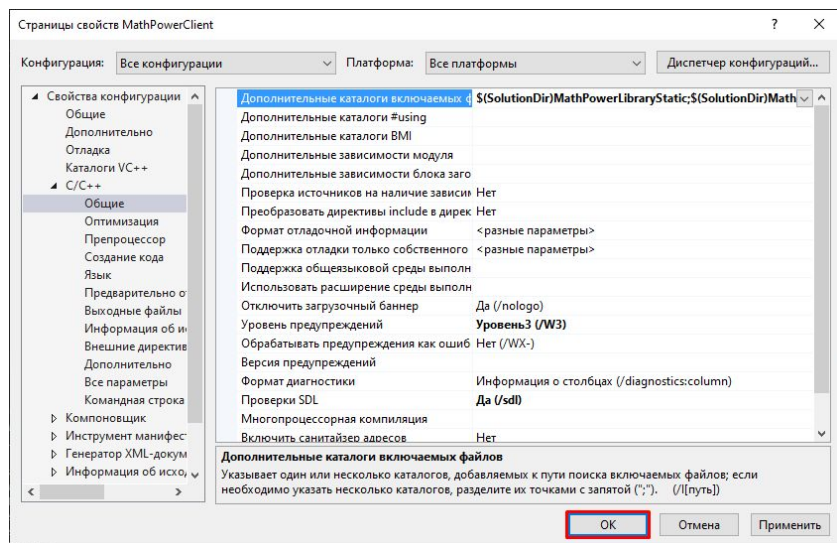
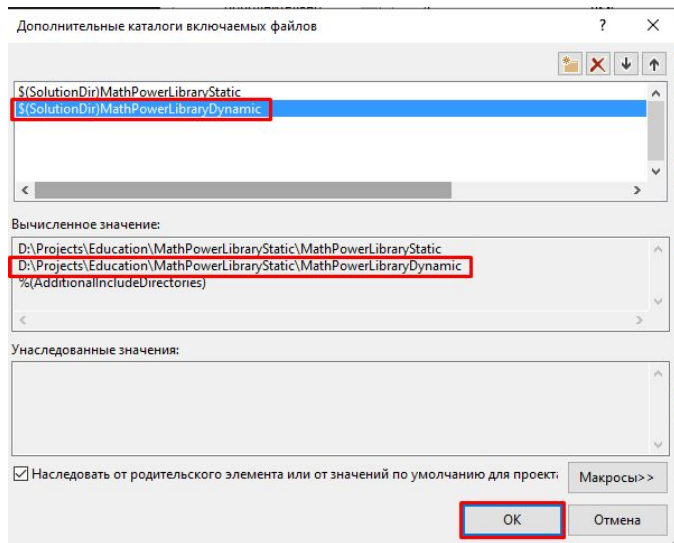
# Динамическая библиотека в Visual Studio

Помимо добавления ссылки нужно также сообщить проекту MathPowerClient, где он может найти **заголовочные файлы** проекта **MathPowerLibraryDynamic** для включения. Откройте **“Свойства”** проекта **MathPowerClient**, убедитесь, что выбраны **“Все конфигурации”** и **“Все платформы”**. Перейдите в пункт меню **“C/C++”** -> **“Общие”**. Откройте окно редактирования пункта **“Дополнительные каталоги включаемых файлов”**



# Динамическая библиотека в Visual Studio

Появится ещё одно окно. В нём добавьте строку **\$(SolutionDir)MathPowerLibraryDynamic**. Проверьте, что в директории, которая отобразилась в поле “**Вычисленное значение**”, действительно находится заголовочный файл **DynamicMathPower.h**. Нажмите “**OK**”, в предыдущем окне тоже нажмите “**OK**”




# Динамическая библиотека в Visual Studio

Откройте файл **MathPowerClient.cpp** и подключите заголовочный файл **DynamicMathPower.h**. Используйте класс **DynamicCalculator** из нашей библиотеки и убедитесь, что всё работает. Готово!

```
#include <iostream>
#include "MathPower.h"
#include "DynamicMathPower.h"

int main()
{
    MathPower::Calculator calculator;
    DynamicMathPower::DynamicCalculator dynamic_calculator;
    int a = 8, b = 5;
    std::cout << a << " + " << b << " = " << calculator.Sum(a, b) << std::endl;
    std::cout << a << " - " << b << " = " << calculator.Sub(a, b) << std::endl;
    std::cout << a << " * " << b << " = " << dynamic_calculator.Mult(a, b) << std::endl;
}
```

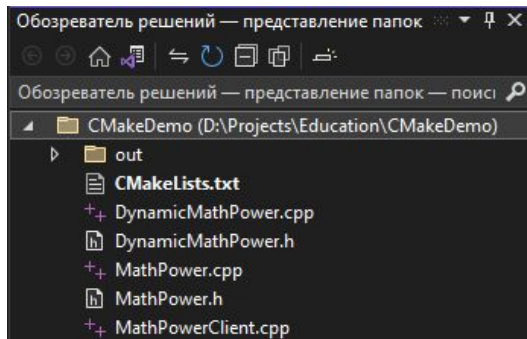
 Консоль отладки Microsoft Visual Studio

```
8 + 5 = 13
8 - 5 = 3
8 * 5 = 40
```

# Динамическая библиотека в CMake

Для исходных файлов, расположенных в одной директории (см. скриншот), в файл CMakeLists.txt нужно будет добавить ещё одну команду `add_library` – уже для динамической библиотеки (определяется с помощью ключевого слова `SHARED`). В команде `target_link_libraries` появится идентификатор вашей новой библиотеки – остальной код не меняется

```
cmake_minimum_required(VERSION 3.22.0)
project(MathPower)
add_library(MathPowerStaticLib STATIC MathPower.h MathPower.cpp)
add_library(MathPowerDynamicLib SHARED DynamicMathPower.h DynamicMathPower.cpp)
add_executable(MathPowerExe MathPowerClient.cpp)
target_link_libraries(MathPowerExe MathPowerStaticLib MathPowerDynamicLib)
```





# Динамическая библиотека в CMake

Есть вариант сборки DLL с помощью CMake без специального синтаксиса (который `__declspec(dllexport)` и `__declspec(dllimport)`). Для этого перед командой `add_library` нужно добавить команду `set(CMAKE_WINDOWS_EXPORT_ALL_SYMBOLS ON)` – она устанавливает значение переменной для CMake, благодаря которой CMake понимает, что нужно по умолчанию экспортировать из библиотеки всё

```
cmake_minimum_required(VERSION 3.22.0)
project(MathPower)
add_library(MathPowerStaticLib STATIC MathPower.h MathPower.cpp)
set(CMAKE_WINDOWS_EXPORT_ALL_SYMBOLS ON)
add_library(MathPowerDynamicLib SHARED DynamicMathPower.h DynamicMathPower.cpp)
add_executable(MathPowerExe MathPowerClient.cpp)
target_link_libraries(MathPowerExe MathPowerStaticLib MathPowerDynamicLib)
```



# Итоги



# Итоги занятия

Сегодня мы

- 1 Разобрались, что такое библиотеки и зачем они нужны
- 2 Познакомились с разными видами библиотек
- 3 Узнали, как работать со статическими библиотеками
- 4 Выяснили, как работать с динамическими библиотеками



# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- 1 Вопросы по домашней работе задавайте в чате группы
- 2 Задачи можно сдавать по частям
- 3 Зачёт по домашней работе ставят после того, как приняты все задачи



# Дополнительные материалы

- [Статические библиотеки](#)
- [Динамические библиотеки](#)



# Задавайте вопросы и пишите отзыв о лекции

Михаил Смирнов  
Разработчик C++

