# Программирование на C++

Минцифры России

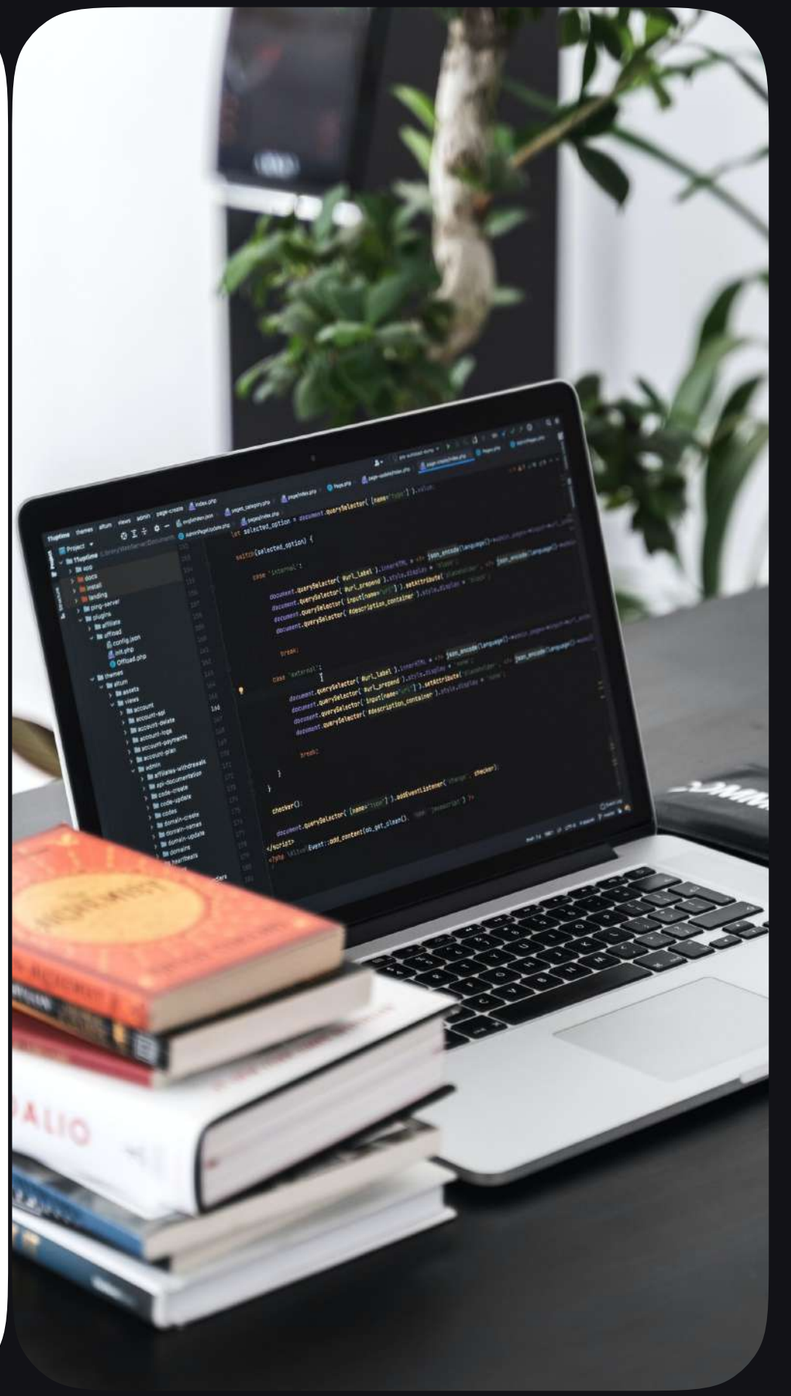
UCHi DOMA

20.35 УНИВЕРСИТЕТ

Урок 1 Модуль 4

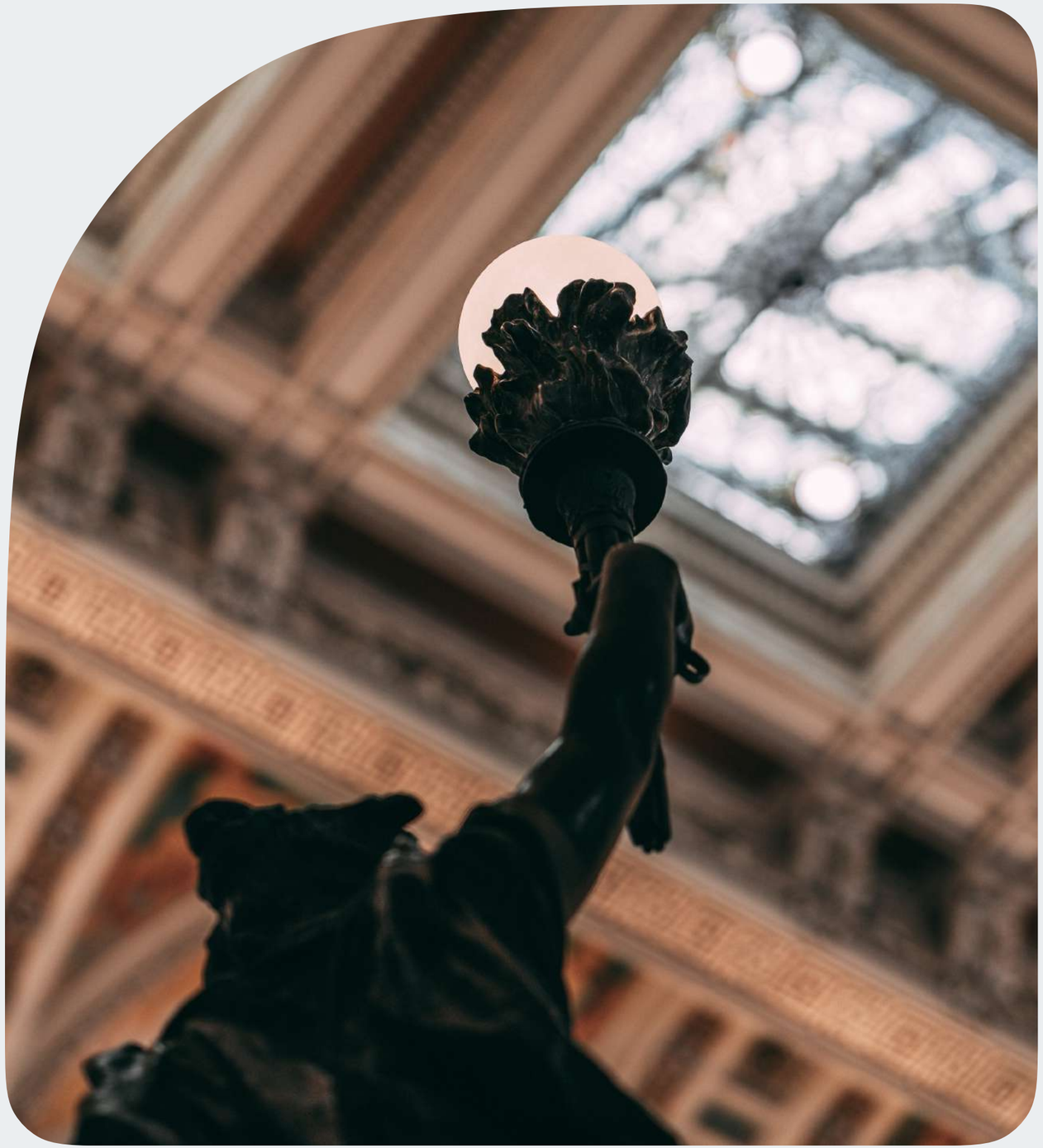
# Библиотека. Статическая библиотека

Полезные материалы



# Цели урока

- изучить, как устроены статические библиотеки
- отработать на практике написание алгоритмов с использованием статических библиотек на C++



# Библиотека

**Библиотека** — это пакет кода, который предназначен для повторного использования многими программами.

Обычно библиотека C++ состоит из двух частей:

1. Заголовочный файл, который определяет функциональность, которую библиотека предоставляет (предлагает) программам, использующим ее.
2. Предварительно скомпилированный двоичный файл, который содержит реализацию этой функциональности, предварительно скомпилированную в машинный код.

Некоторые библиотеки могут быть разделены на несколько файлов и/или иметь несколько файлов заголовков.

# Библиотеки предварительно скомпилированы по причинам:

1. Поскольку библиотеки меняются редко, их не нужно часто перекомпилировать.
2. Поскольку предварительно скомпилированные объекты представлены машинным кодом, люди не могут получить доступ к исходному коду или изменить его, что важно для предприятий или людей, которые не хотят делать свой исходный код доступным из соображений интеллектуальной собственности.

# Виды библиотек

Существует два типа библиотек:

- статические библиотеки
- динамические библиотеки

# Статическая библиотека

**Статическая библиотека** (иногда называемая **archive**, «архив») состоит из подпрограмм, которые скомпилированы и линкуются непосредственно с программой.

Когда компилируется программа, использующая статическую библиотеку, все функции статической библиотеки, которые использует программа, становятся частью исполняемого файла.

В Windows статические библиотеки обычно имеют расширение **.lib**, а в Linux — расширение **.a** (archive, архив).

# Статическая библиотека

## Плюсы +

Преимуществом статических библиотек является то, что необходимо распространять только исполняемый файл, чтобы пользователи могли запускать программу.

Поскольку библиотека становится частью программы, это гарантирует, что с программой всегда будет использоваться правильная версия библиотеки. Кроме того, поскольку статические библиотеки становятся частью программы, можно использовать их так же, как функции, которые вы написали для своей программы.

# Статическая библиотека

## Минусы –

Поскольку копия библиотеки становится частью каждого исполняемого файла, который ее использует, это может привести к потере большого количества места.

Статические библиотеки также не могут быть легко обновлены — для обновления библиотеки необходимо заменить весь исполняемый файл.

# Использование библиотек

Включите с помощью **#include** заголовочный файл(ы) библиотеки в вашу программу. Это сообщит компилятору обо всех функциях, предлагаемых библиотекой, чтобы ваша программа могла правильно компилироваться.

# Используемые расширения

В C++ используются следующие расширения:

- **.cpp, .cxx, .cc, .c** — для файлов с исходным кодом
- **.hpp, .hxx, .hh, .h** — для заголовочных файлов

Файлы стандартной библиотеки C++ не используют расширение, например:

- **iostream**
- **algorithm**
- **string**

# Практика

Создадим статическую библиотеку (LIB-файл) для использования с приложениями C++.

Статические библиотеки являются хорошим способом повторного использования кода. Вместо того чтобы каждый раз реализовывать одни и те же подпрограммы для обеспечения той или иной функциональности в каждом создаваемом приложении, их можно создать единожды и затем вызывать из приложений.

Код, подключенный из статической библиотеки, становится частью вашего приложения — для использования кода не нужно устанавливать еще какой-либо файл.

# Практика

- Создание проекта статической библиотеки
- Добавление класса в статическую библиотеку
- Создание консольного приложения C++, ссылающегося на статическую библиотеку
- Использование функциональности из статической библиотеки в приложении
- Запуск приложения

# Создание проекта статической библиотеки

Для создания библиотеки будем использовать **MS Visual Studio**

# Создание проекта статической библиотеки в Visual Studio

1. В строке меню выберите Файл>Новый>проект , чтобы открыть диалоговое окно Создание проекта .
2. В верхней части диалогового окна задайте для параметра Язык значение C++, для параметра Платформа — значение Windows, а для параметра Тип проекта — значение Библиотека.
3. В отфильтрованном списке типов проектов выберите пункт Мастер классических приложений Windows, а затем нажмите кнопку Далее.
4. На странице Настроить новый проект введите MathLibrary в поле Имя проекта. В поле Имя решения введите StaticMath. Нажмите кнопку Создать, чтобы открыть диалоговое окно Проект классического приложения Windows.
5. В диалоговом окне Проект классического приложения Windows в разделе Тип приложения выберите Статическая библиотека (.lib) .
6. В разделе Дополнительные параметры снимите флажок Предварительно откомпилированный заголовок, если он установлен. Установите флажок Пустой проект.
7. Нажмите кнопку ОК, чтобы создать проект.

# Добавление класса в статическую библиотеку

1. Чтобы создать файл заголовка для нового класса откройте контекстное меню проекта MathLibrary в обозревателе решений, а затем выберите Добавить>Новый элемент.

2. В диалоговом окне Добавление нового элемента выберите пункт Visual C++>Код. В центральной области выберите Заголовочный файл (.h) . Укажите имя для файла заголовка, например **MathLibrary.h**, и нажмите кнопку Добавить. Отобразится почти пустой файл заголовка.

3. Добавьте объявление класса с именем **Arithmetic** для выполнения обычных арифметических операций, таких как сложение, вычитание, умножение и деление.

# Добавление класса в статическую библиотеку

```
// MathLibrary.h
#pragma once

namespace MathLibrary
{
    class Arithmetic
    {
    public:
        // Returns a + b
        static double Add(double a, double b);

        // Returns a - b
        static double Subtract(double a, double b);

        // Returns a * b
        static double Multiply(double a, double b);

        // Returns a / b
        static double Divide(double a, double b);
    };
}
```

# Добавление класса в статическую библиотеку

4. Чтобы создать исходный файл для нового класса, откройте контекстное меню проекта MathLibrary в обозревателе решений, а затем выберите Добавить>Новый элемент.

5. В диалоговом окне Добавление нового элемента в центральной области выберите Файл C++ (.cpp) . Укажите имя исходного файла, например MathLibrary.cpp, и нажмите кнопку Добавить. Отобразится пустой исходный файл.

6. Используйте этот исходный файл для реализации функций класса **Arithmetic**.

# Добавление класса в статическую библиотеку

```cpp
// MathLibrary.cpp

#include "MathLibrary.h"

namespace MathLibrary
{
    double Arithmetic::Add(double a, double b)
    {
        return a + b;
    }

    double Arithmetic::Subtract(double a, double b)
    {
        return a - b;
    }

    double Arithmetic::Multiply(double a, double b)
    {
        return a * b;
    }

    double Arithmetic::Divide(double a, double b)
    {
        return a / b;
    }
}
```

# Добавление класса в статическую библиотеку

7 Чтобы выполнить сборку статической библиотеки, выберите в строке меню команду Сборка>Собрать решение.
В результате будет создана статическая библиотека MathLibrary.lib, которая может использоваться другими программами.

**Примечание**

При выполнении сборки из командной строки Visual Studio программа собирается в два этапа. Сначала запустите cl /c /EHsc MathLibrary.cpp, чтобы скомпилировать код и создать объектный файл с именем MathLibrary.obj. (Команда cl вызывает компилятор Cl.exe, а параметр /c дает указание компилировать без компоновки. Дополнительные сведения см. в разделе /c (компиляция без компоновки).) Во-вторых, выполните команду lib MathLibrary.obj , чтобы связать код и создать статическую библиотеку MathLibrary.lib. (Команда lib вызывает диспетчер библиотек, Lib.exe.)

# Создание консольного приложения C++, ссылающегося на статическую библиотеку, в Visual Studio

1. В обозревателе решений щелкните правой кнопкой мыши узел верхнего уровня Решение StaticMath, чтобы открыть контекстное меню. Выберите пункты Добавить>Новый проект, чтобы открыть диалоговое окно Добавить новый проект.
2. В верхней части диалогового окна задайте для фильтра Тип проекта значение Консоль.
3. В отфильтрованном списке типов проектов щелкните Консольное приложение, а затем нажмите кнопку Далее. На следующей странице в поле Имя введите имя проекта MathClient.
4. Нажмите кнопку Создать, чтобы создать клиентский проект.
5. После создания консольного приложения будет создана пустая программа. Имя исходного файла будет совпадать с ранее выбранным именем. В этом примере он имеет имя MathClient.cpp.

# Использование функциональности из статической библиотеки в приложении

1 Для использования математических процедур из статической библиотеки необходимо сослаться на эту библиотеку. В обозревателе решений откройте контекстное меню проекта MathClient, а затем выберите команду «Добавить>Ссылка».

2 В диалоговом окне Добавление ссылки перечислены библиотеки, на которые можно создать ссылку. На вкладке «Проекты» перечислены проекты текущего решения и все библиотеки, на которые они ссылаются. На вкладке Проекты установите флажок MathLibrary, а затем нажмите кнопку «ОК».

3 Для создания ссылки на файл заголовка **MathLibrary.h** необходимо изменить путь к каталогам включаемых файлов.
В обозревателе решений щелкните правой кнопкой мыши проект MathClient, чтобы открыть контекстное меню.
Выберите пункт «Свойства», чтобы открыть диалоговое окно «Страницы свойств MathClient».

4 В диалоговом окне Страницы свойств MathClient в раскрывающемся списке Конфигурация выберите пункт Все конфигурации. В раскрывающемся списке Платформа выберите пункт «Все платформы».

# Использование функциональности из статической библиотеки в приложении

**5** Перейдите на страницу свойств «Свойства конфигурации>C/C++>Общие». В свойстве «Дополнительные каталоги включаемых файлов» укажите путь к каталогу MathLibrary или найдите этот каталог. Чтобы найти путь к каталогу, выполните указанные ниже действия.

**a** Откройте раскрывающийся список значений свойства «Дополнительные каталоги включаемых файлов», а затем выберите «Изменить».

**b** В диалоговом окне «Дополнительные каталоги включаемых файлов» дважды щелкните в верхней части текстового поля. Нажмите кнопку с многоточием ( ... ) в конце строки.

**c** В диалоговом окне «Выбор каталога» перейдите на уровень вверх и выберите каталог «MathLibrary». Затем нажмите кнопку «Выбрать папку», чтобы сохранить выбор.

**d** В диалоговом окне Дополнительные каталоги включаемых файлов нажмите кнопку «ОК».

**e** В диалоговом окне Страницы свойств нажмите кнопку «ОК», чтобы сохранить изменения в проекте.

# Использование функциональности из статической библиотеки в приложении

6 Теперь в этом приложении можно использовать класс **Arithmetic**, включив в код заголовок **#include "MathLibrary.h"**. Замените содержимое **MathClient.cpp** на следующий код:

```
// MathClient.cpp
// compile with: cl /EHsc MathClient.cpp /link MathLibrary.lib

#include <iostream>
#include "MathLibrary.h"

int main()
{
    double a = 7.4;
    int b = 99;

    std::cout << "a + b = " <<
        MathLibrary::Arithmetic::Add(a, b) << std::endl;
    std::cout << "a - b = " <<
        MathLibrary::Arithmetic::Subtract(a, b) << std::endl;
    std::cout << "a * b = " <<
        MathLibrary::Arithmetic::Multiply(a, b) << std::endl;
    std::cout << "a / b = " <<
        MathLibrary::Arithmetic::Divide(a, b) << std::endl;

    return 0;
}
```

7 Чтобы выполнить сборку исполняемого файла, выберите в строке меню команду «Сборка>Собрать решение».

# Запуск приложения

1. Убедитесь в том, что проект MathClient выбран в качестве проекта по умолчанию. Чтобы выбрать его, в обозревателе решений откройте контекстное меню проекта MathClient и выберите команду «Назначить запускаемым проектом».

2. Чтобы запустить проект, в строке меню выберите «Отладка>Запуск без отладки». Выходные данные должны выглядеть примерно так:

Output

```
a + b = 106.4
a – b = –91.6
a * b = 732.6
a / b = 0.0747475
```