

Структура языка программирования. Статическая библиотека

План лекции:

- назначение и классификация библиотек;
- преимущества и недостатки статических библиотек;
- создание статических библиотек;
- использование статических библиотек;
- пример.

1. Назначение библиотек:

предоставить стандартный простой и надёжный *механизм повторного использования кода*.

2. Использование:

- для использования функций из библиотеки в разных программах;
- при разработке большого проекта отлаженные функции помещают в библиотеку (время трансляции уменьшается).

3. Классификация библиотек:

- *библиотеки на языках программирования* (библиотеки классов, шаблонов, функций и т. п.). Компилируются вместе с исходными файлами проекта;
- *библиотеки объектных модулей* (статические библиотеки). Компилируются вместе с объектными файлами проекта;
- *библиотеки исполняемых модулей* (динамические библиотеки). Загружаются в память в момент запуска программы или во время ее исполнения, по мере надобности.

4. Статические библиотеки. Библиотеки для компилируемых языков

Статическая библиотека – файл с объектными модулями, предназначенный для подключения к программе на этапе компоновки.

Библиотеки, распространяемые в виде исходного кода, преобразуются компилятором в объектные файлы. Затем компоновщик соединяет объектные файлы библиотек и объектные файлы программы в один исполняемый файл.

Например, в исходных текстах распространяются:

- библиотеки для языка Fortran;
- библиотека Boost для языка C++.

Справка.

Библиотека BOOST – это набор C++ библиотек, созданных независимыми разработчиками, которые используются для создания проектов, предназначенных для реализации на различных платформах.

В стандарт C++ включена библиотека файловой системы, основанная на boost::filesystem. Спецификации для стандарта C++17 были опубликованы в декабре 2017 года.

Расширения объектных файлов модулей статических библиотек:

для ОС Microsoft Windows	.lib
для ОС UNIX	.a

5. Библиотеки для интерпретируемых языков

Библиотека – файл, содержащий либо код на интерпретируемом языке, либо **байт-код** для виртуальной машины.

Например, библиотеки для языка Python могут распространяться либо в виде файлов с исходным кодом (расширение «py»), либо в виде файлов с байт-кодом (расширение «pyc», (пайк) «py» + буква «с» от англ. *compiled*).

6. Статическая библиотека:

файл (обычно с расширением **lib**), содержащий объектные модули;
входной файл для компоновщика (**linker**).

Достоинства:

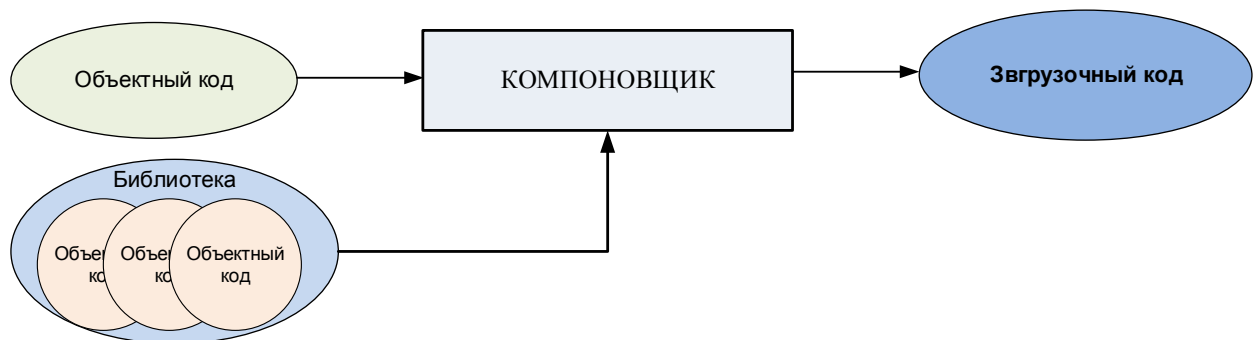
- просто использовать;
- не требуется наличие самой библиотеки;
- исполняемый файл один (расширение .exe).

Недостатки:

- платформенно зависима;
- загружается в память с каждым экземпляром запущенного приложения;
- при изменении кода библиотеки необходима компоновка всех приложений, которые используют библиотеку.

Статическая библиотека – файл с объектными модулями, который подключается к программе на этапе компоновки.

Компоновщик (linker, редактор связей) – программа, принимающая один или несколько объектных модулей и формирующая на их основе загрузочный модуль:



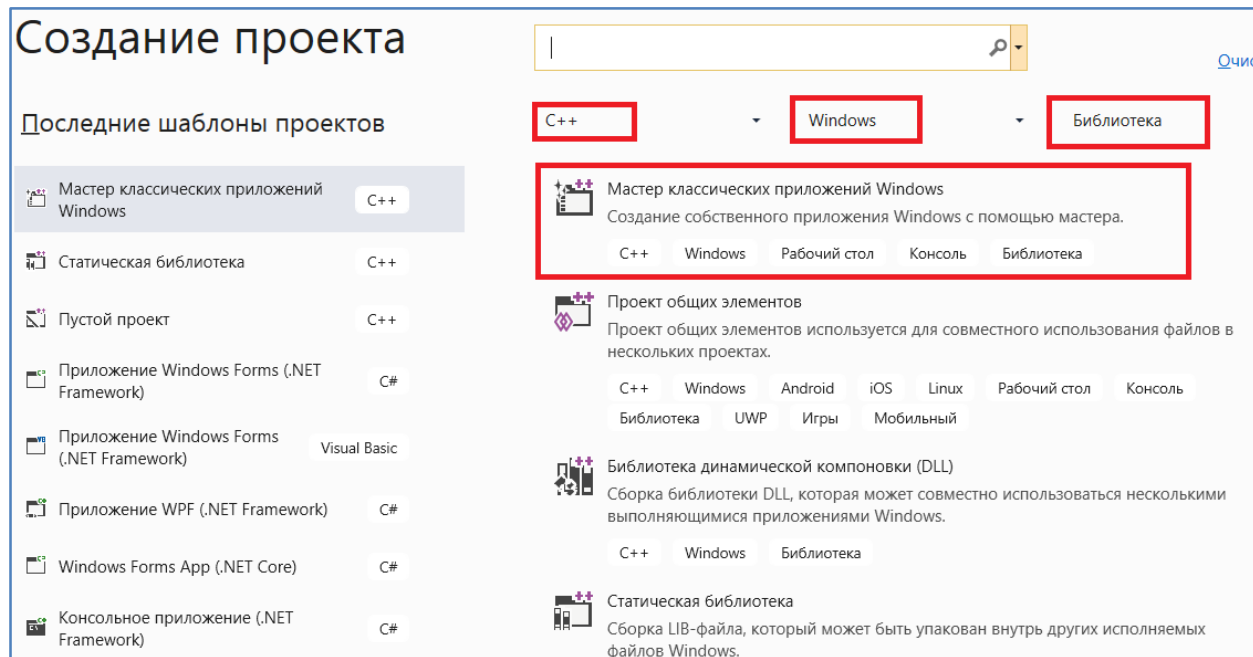
Если программа состоит из нескольких объектных файлов, компоновщик собирает эти файлы в единый исполнимый модуль, вычисляя и подставляя адреса вместо неопределенных внешних имен, в течение **времени компоновки** (статическая компоновка) или во **время исполнения** (динамическая компоновка).

7. Статическая библиотека Microsoft:

файл с расширением **lib**.

Для работы с библиотекой предназначена утилита **LIB**.

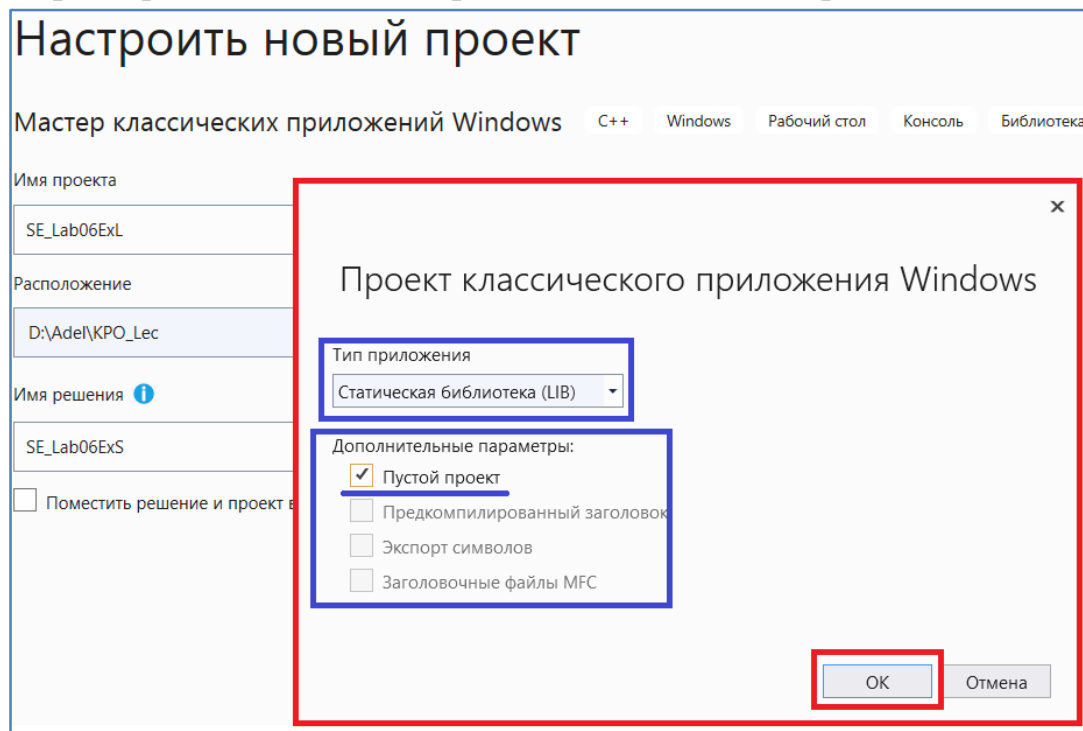
7.1. Создание статической библиотеки с помощью «Мастера классических приложений Windows». Шаг 1:



Определяем имя решения, проекта и выбираем место размещения на диске.

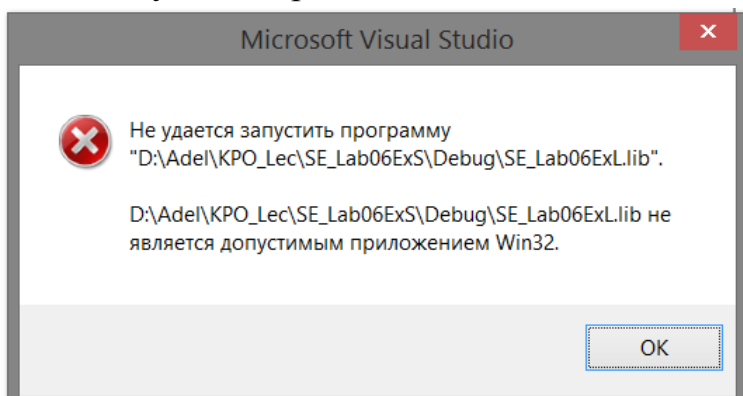
7.2. Создание статической библиотеки с помощью «Мастера классических приложений Windows». Шаг 2.

Выбираем тип приложения «Статическая библиотека» (снимаем флажок «Предварительно скомпилированные заголовки» при необходимости):

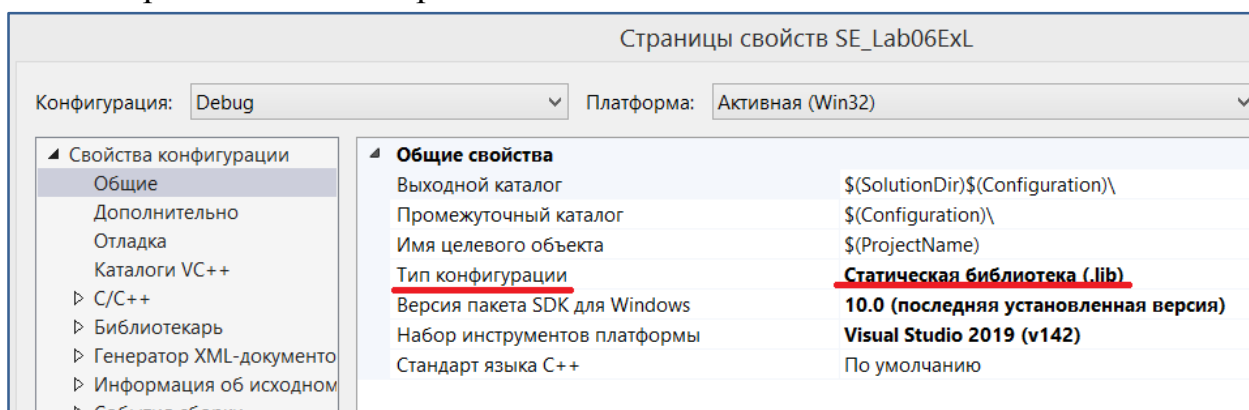


В проект добавляем один или несколько файлов, содержащих реализации функций библиотеки.

7.3. Запускаем проект на выполнение:



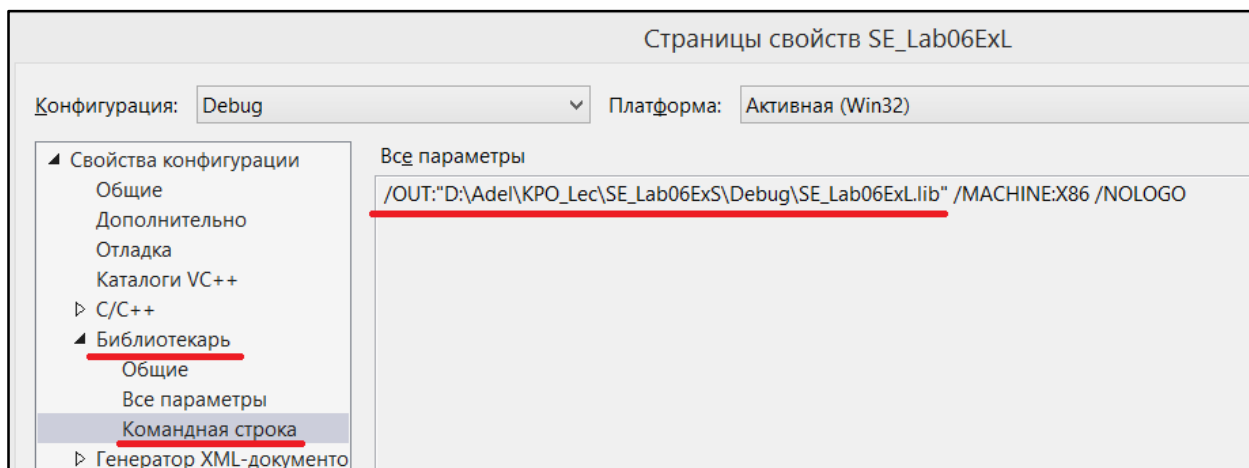
7.4. Страница свойств проекта:





В пункте раздела *«Библиотекарь»* → *«Командная строка»* отображается текущее значение параметра */OUT*.

Расширение выходных файлов определено как *.lib*.






Директорий – папка проекта *Debug*.



После построения проекта в папке решения Debug размещен файл статической библиотеки (.lib).

KPO_Lec ▸ SE_Lab06ExS ▸ Debug			
Поиск: Debug			
Имя	Дата изменения	Тип	Размер
 <u>SE_Lab06ExL.lib</u>	24.03.2022 12:50	Object File Library	4 КБ
 SE_Lab06ExL.pdb	24.03.2022 12:50	Program Debug D...	68 КБ

В папке проекта Debug размещен файл журнала построения:

SE_Lab06ExS ▸ SE_Lab06ExL ▸ Debug			
Поиск: Debug			
Имя	Дата изменения	Тип	Размер
 SE_Lab06ExL.tlog	24.03.2022 12:50	Папка с файлами	
 Lec_Ex.obj	24.03.2022 12:50	Object File	3 КБ
 SE_Lab06ExL.idb	24.03.2022 12:50	VC++ Minimum R...	19 КБ
 <u>SE_Lab06ExL.log</u>	24.03.2022 12:50	Текстовый докум...	2 КБ
 SE_Lab06ExL.pdb	24.03.2022 12:50	Program Debug D...	68 КБ

В журнале проекта зафиксировано выполнение сборки проекта.

Видим, что файл статической библиотеки создан утилитой LIB.

```

SE_Lab06ExL.log — Блокнот
Файл  Правка  Формат  Вид  Справка
Сборка начата 24.03.2022 12:50:27.
1>Проект "D:\Adel\KPO_Lec\SE_Lab06ExS\SE_Lab06ExL\SE_Lab06ExL.vcxproj" в узле 2 (целевые объекты Build).
1>PrepareForBuild:
  Создание каталога "D:\Adel\KPO_Lec\SE_Lab06ExS\Debug\".
  Создание каталога "Debug\SE_Lab06ExL.tlog\".
  InitializeBuildStatus:
  Создание "Debug\SE_Lab06ExL.tlog\unsuccessfulbuild", так как было задано "AlwaysCreate".
1) clcompile:
  C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Tools\MSVC\14.23.28105\bin\HostX86\x86\CL.exe /c /ZI /JMC /nologo /W3 /WX- /diagnostics
/sdl /Od /Oy- /D WIN32 /D _DEBUG /D _LIB /D _UNICODE /D UNICODE /Gm- /EHsc /RTC1 /MDd /GS /fp:precise /permissive- /Zc:wchar_t /Zc:forScope /Zc:inline /Fo"Debug\
\Fd"Debug\SE_Lab06ExL.pdb" /Gd /TP /analyze- /FC /errorReport:prompt Lec_Ex.cpp
  Lec_Ex.cpp
2) lib:
  C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Tools\MSVC\14.23.28105\bin\HostX86\x86\LIB.exe /OUT:"D:\Adel\KPO_Lec\SE_Lab06ExS\Debug
\SE_Lab06ExL.lib" /NOLOGO /MACHINE:X86 Debug\Lec_Ex.obj
  SE_Lab06ExL.vcxproj -> D:\Adel\KPO_Lec\SE_Lab06ExS\Debug\SE_Lab06ExL.lib
  CopyFilesToOutputDirectory:
  Копирование файла из "D:\Adel\KPO_Lec\SE_Lab06ExS\SE_Lab06ExL\Debug\SE_Lab06ExL.pdb" в "D:\Adel\KPO_Lec\SE_Lab06ExS\Debug\SE_Lab06ExL.pdb".
  FinalizeBuildStatus:
  Файл "Debug\SE_Lab06ExL.tlog\unsuccessfulbuild" удалится.
  Обращение к "Debug\SE_Lab06ExL.tlog\SE_Lab06ExL.lastbuildstate".
1>Сборка проекта "D:\Adel\KPO_Lec\SE_Lab06ExS\SE_Lab06ExL\SE_Lab06ExL.vcxproj" завершена (целевые объекты Build).

Сборка успешно завершена.
Предупреждений: 0
Ошибок: 0
Прошло времени: 00:00:00.51

```

7.5. Режимы использования утилиты LIB:

- построение или изменение библиотеки;
- извлечение элемента-объекта библиотеки в файл;
- создание файла экспорта и библиотеки импорта.

Эти режимы *взаимоисключающие*, LIB можно использовать только в одном режиме.

LIB принимает те или иные входные файлы в зависимости от режима использования.

8. Статическая библиотека. Параметры утилиты LIB.

Параметры LIB:

<https://docs.microsoft.com/ru-ru/cpp/build/reference/overview-of-lib?view=vs-2019>

/DEF

Создание библиотеки импорта и файла экспорта

Дополнительно в [Построение библиотеки импорта и файла экспорта](#).

/ERRORREPORT

Передача Майкрософт сведений о внутренних ошибках.

Дополнительно в [Запуск программы LIB](#).

/EXPORT

Экспорт функции из программы.

Смори в разделе [Построение библиотеки импорта и файла экспорта](#).

/EXTRACT

Создание объектного файла (OBJ-файла), содержащего копию элемента существующей библиотеки.

Дополнительно в разделе [Извлечение члена библиотеки](#).

/INCLUDE

Добавление символа в таблицу символов.

Дополнительные сведения см. в разделе [Построение библиотеки импорта и файла экспорта](#).

/LIBPATH

Переопределяет путь к библиотеке среды.

Дополнительные сведения см. в разделе [Управление библиотекой](#).

/LIST

Отображает информацию о выходной библиотеке в стандартном виде.

Дополнительные сведения см. в разделе [Управление библиотекой](#).

/LTCG

Иницирует построение библиотеки с помощью создания кода времени компоновки.

Дополнительные сведения см. в разделе [Запуск программы LIB](#).

/MACHINE

Задание целевой платформы для программы.

Дополнительные сведения см. в разделе [Запуск программы LIB](#).

/NAME

При построении библиотеки импорта указывает имя библиотеки DLL, для которой была создана библиотека импорта.

Дополнительные сведения см. в разделе [Управление библиотекой](#).

/NODEFAULTLIB

Удаляет одну или несколько библиотек по умолчанию из списка искомых библиотек при разрешении внешних ссылок.

Дополнительные сведения см. в разделе [Управление библиотекой](#).

/NOLOGO

Отключает вывод программой LIB уведомления об авторских правах и номере версии, а также отображение команд командного файла.

Дополнительные сведения см. в разделе [Запуск программы LIB](#).

/OUT

Переопределяет имя выходного файла, используемое по умолчанию.

Дополнительные сведения см. в разделе [Управление библиотекой](#).

/REMOVE

Пропуск объекта из выходной библиотеки.

Дополнительные сведения см. в разделе [Управление библиотекой](#).

/SUBSYSTEM

Сообщает операционной системе способ запуска программы, созданной путем привязки к выходной библиотеке.

Дополнительные сведения см. в разделе [Управление библиотекой](#).

/VERBOSE

Отображает подробные сведения о ходе сеанса, включая имена добавляемых OBJ-файлов.

Дополнительные сведения см. в разделе [Запуск программы LIB](#).

/WX

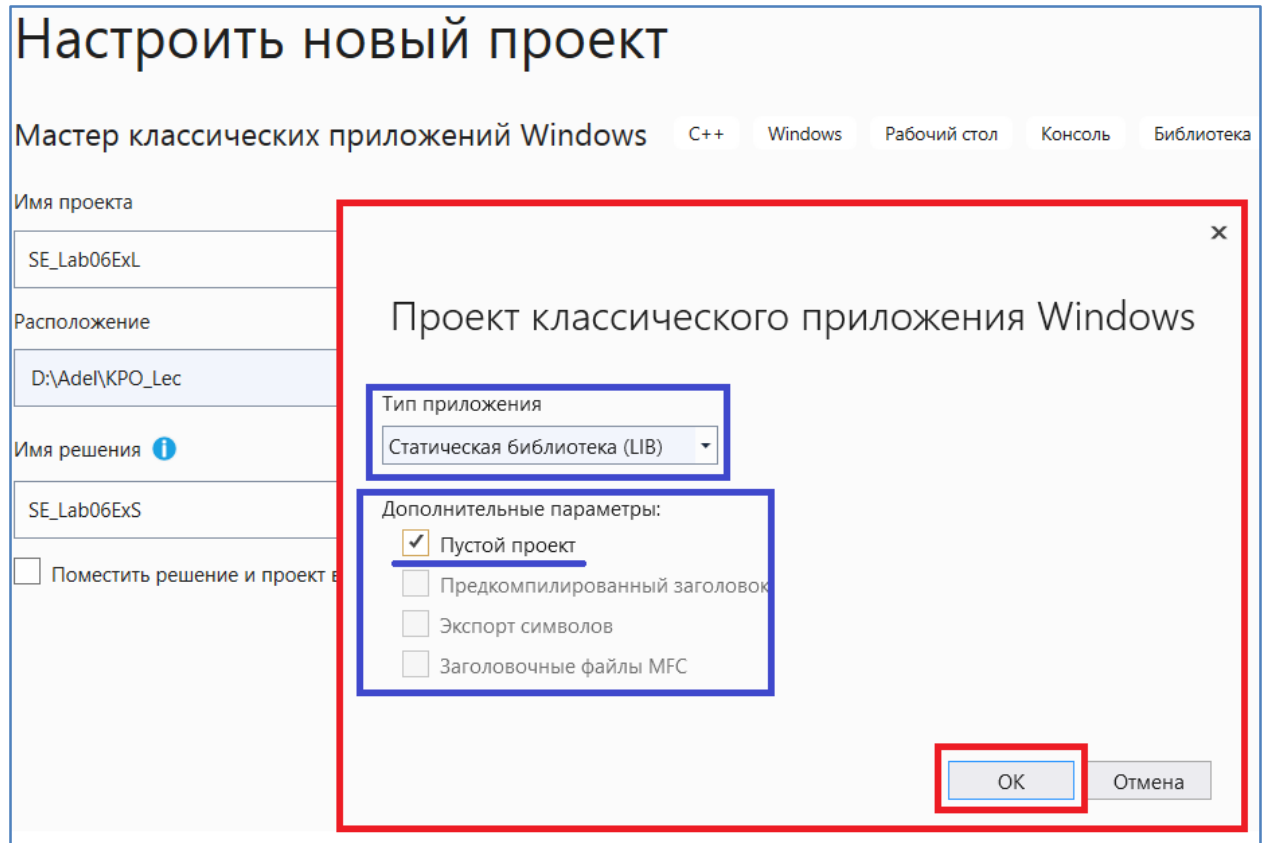
Обработка предупреждений, как ошибок.

Дополнительные сведения см. в разделе [Запуск программы LIB](#).

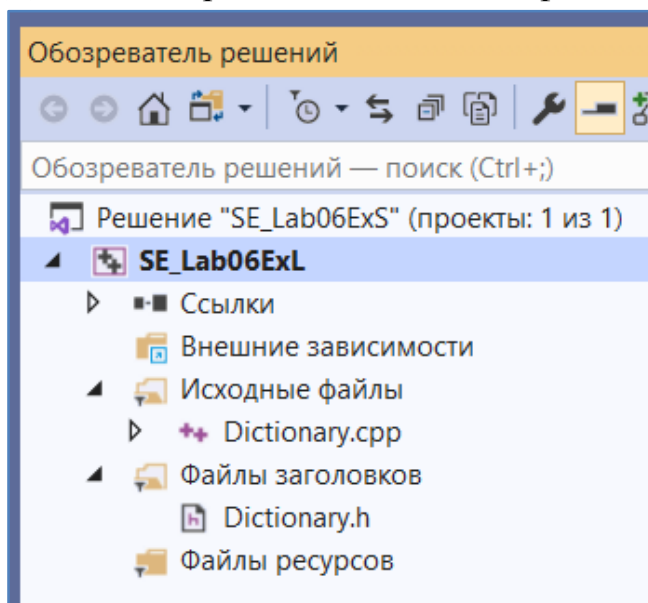
9. Статическая библиотека. Пример применения в C++

9.1. Создание

Создадим проект с именем SE_Lab06ExL решения SE_Lab06ExS, который является статической библиотекой:



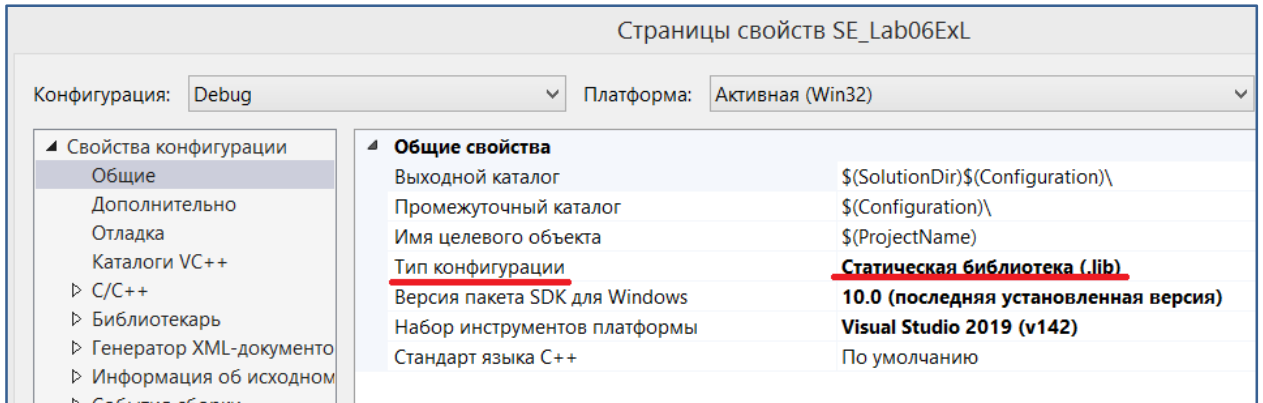
Добавим в проект заголовочный файл и файл исходного кода библиотеки.



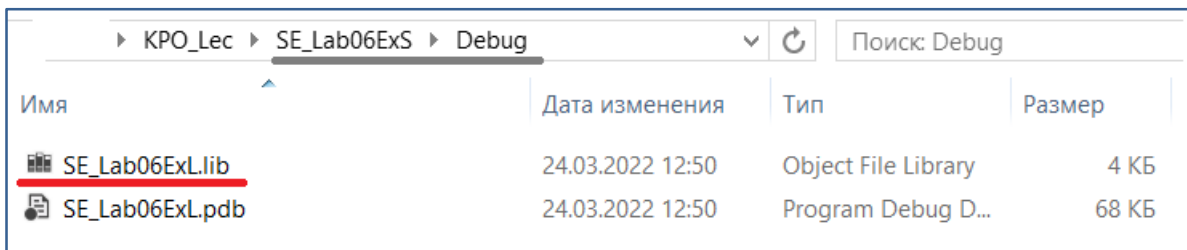
Заголовочные файлы. Правила создания.

В заголовочном файле задают:

- прототипы всех функций, которые входят в библиотеку (*интерфейс библиотеки*), например: `void Print(Instance d);`
- типы пользователя;
- константы, например: `#define DICTNAMEMAXSIZE 20`

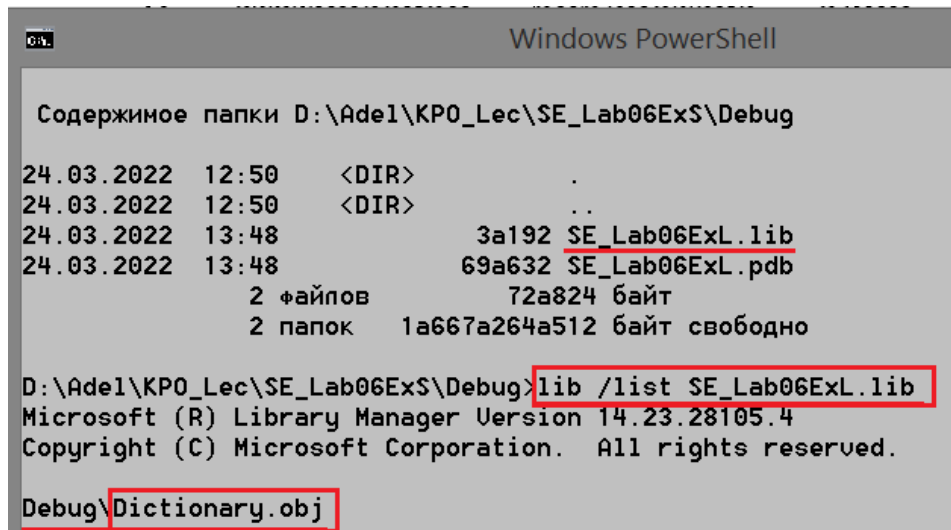


Файл статической библиотеки размещен в папке решения Debug:



В командной строке разработчика с помощью *утилиты LIB* с ключом /LIST можно получить перечень `obj`-модулей, содержащихся в `LIB`-файле.

В статической библиотеке один объектный модуль с именем `Dictionary.obj`:



9.2. Главная программа для использования статической библиотеки.

Добавляем в решение новый пустой проект SE_Lab06Ex – консольное приложение, которое будет использовать созданную нами библиотеку.

Настроить новый проект

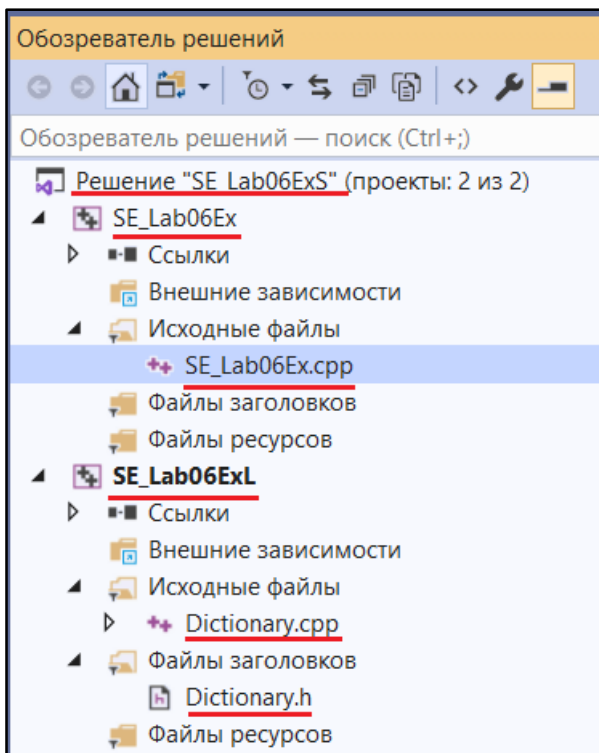
Пустой проект C++ Windows Консоль

Имя проекта

Расположение

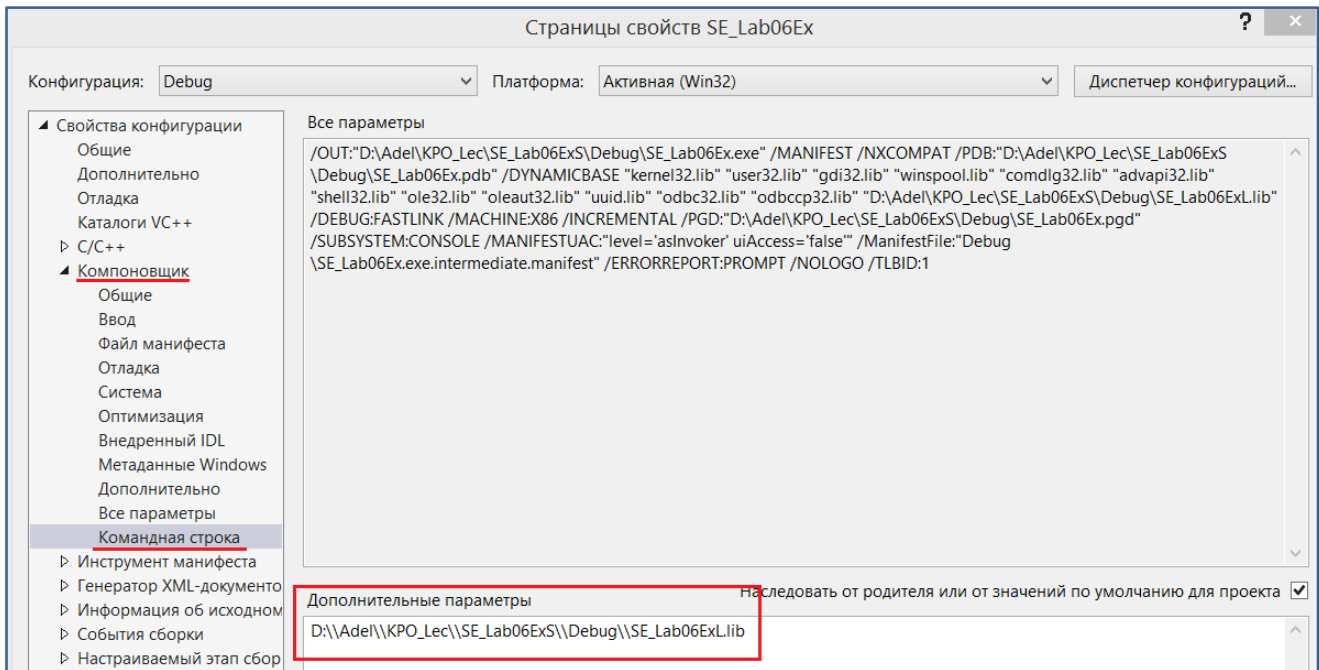
Решение

Имя решения ⓘ



Назначаем проект SE_Lab06Ex запускаемым и выполняем его.

На странице свойств проекта в разделе «Компоновщик» → *Командная строка* в окне «Дополнительные параметры» добавляем параметр, указывающий местоположение и имя статической библиотеки.



Подключаем в main заголовочный файл библиотеки:






```
#include <iostream>
#include <locale>
#include "D:\Adel\KPO_Lec\SE_Lab06ExS\SE_Lab06ExL\Dictionary.h"

int main()
{
    setlocale(LC_ALL, "rus");

    try
    {
        Dictionary::Instance d1 = Dictionary::Create("xxx", 10);
        Dictionary::Entry e1 = { 1, "abcd" }, e2 = { 2, "efg" };
        Dictionary::AddEntry(d1, e1);
        Dictionary::AddEntry(d1, e2);
    }
    catch (char* e) // обработка исключений
    {
        std::cout << "Ошибка: " << e << std::endl;
    };

    system("pause");
    return 0;
}
```

Получаем исполнимый файл SE_Lab06Ex:

KPO_Lec ▸ SE_Lab06ExS ▸ Debug				Поиск: Debug
Имя	Дата изменения	Тип	Размер	
 SE_Lab06Ex.exe	25.03.2022 0:38	Приложение	59 КБ	
 SE_Lab06Ex.ilnk	25.03.2022 0:38	Incremental Linker...	474 КБ	
 SE_Lab06Ex.pdb	25.03.2022 0:38	Program Debug D...	548 КБ	
 SE_Lab06ExL.lib	25.03.2022 0:38	Object File Library	70 КБ	
 SE_Lab06ExL.pdb	25.03.2022 0:38	Program Debug D...	396 КБ	

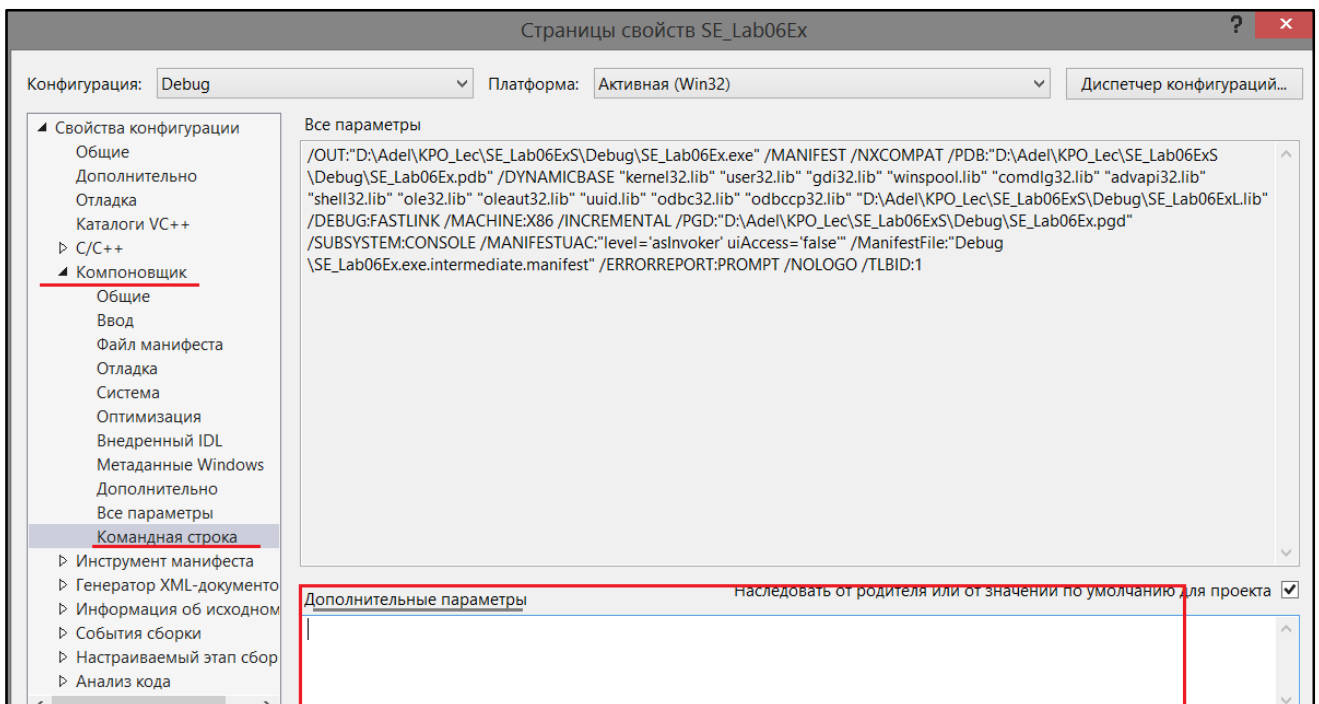
10. Статическая библиотека: директива `#pragma comment`

Автоматическое связывание (от англ. *Auto-linking*) – механизм автоматического определения необходимых библиотек при сборке программ на языках C/C++.

Активируется при помощи директивы препроцессора:

`#pragma comment(lib, <название>)`

В этом случае для подключения статической библиотеки дополнительные параметры командной строки компоновщика не используются:



Статическая библиотека подключается при помощи строки кода

```
#pragma comment(lib, <путь\\имя_библиотеки.lib>)
```

Директива размещается в главной функции:

```
#include <iostream>
#include <locale>
#include "D:\Ade1\KPO_Lec\SE_Lab06ExS\SE_Lab06ExL\Dictionary.h"
#pragma comment(lib, "D:\\Ade1\\KPO_Lec\\SE_Lab06ExS\\Debug\\SE_Lab06ExL.lib")

int main()
{
    setlocale(LC_ALL, "rus");
    try
    {
        Dictionary::Inctance d1 = Dictionary::Create("xxx", 10);
        Dictionary::Entry e1 = { 1, "abcd" }, e2 = { 2, "efg" };
        Dictionary::AddEntry(d1, e1);
        Dictionary::AddEntry(d1, e2);
    }
    catch (char* e)        // обработка исключений
    {
        std::cout << "Ошибка: " << e << std::endl;
    };

    system("pause");
    return 0;
}
```