

**БГТУ, ФИТ, ПОИТ, 2 семестр,
Конструирование программного обеспечения**

Характеристика курса

1. Дисциплина «Конструирование программного обеспечения».
Лектор: **Наркевич Аделина Сергеевна**, старший преподаватель, кафедры программной инженерии (а.408, к.1).
email: narkevich.adelina@gmail.com
2. Всего 324 часа – 154 аудиторных часа, из них лекций 68 часов, лабораторных 86 часов:
 - 1) семестр II: 64 аудиторных часа, из них 32 часа лекций, 32 часа лабораторных работ, **зачет**.
 - 2) семестр III: 36 часов лекций, 54 часа лабораторных работ, **курсовой проект, экзамен**.
3. **Важно!** Инструментарий: Visual Studio 2013 и выше; C+ (**без классов!**); MASM.
4. Курсовой проект: разработка транслятора (спецификация языка, программная реализация транслятора).
На выполнение курсового проекта отводится 11 недель.
Объем программного кода примерно 2000 - 3000 строк.
5. Контрольные работы, тестирование, промежуточные аттестации.
6. Лекции и задания для лабораторных работ доступны в электронном виде:

<https://diskstation.belstu.by:5001/>

login: student

pass: fitfit

Папка: Для_студентов_ФИТ_БГТУ -> Преподаватели -> Наркевич

7. Литература:

№	Наименование	Кол. экз. в библ.
1	Ахо А. Компиляторы: принципы, технологии и инструменты / А. Ахо, Р. Сети, Дж. Ульман. – М.: Вильямс, 2003. – 768с.	1
2	Молчанов А.Ю. Системное программное обеспечение. – СПб.: Питер, 2010. – 400с.	
3	Гагарина Л. Г. Введение в теорию алгоритмических языков и компиляторов / Л. Г. Гагарина, Е. В. Кокорева. – М.: ФОРУМ, 2014. – 178с	
4	Волкова И. А. Системы программирования / И. А. Волкова, И. Г. Головин, Л. Е. Карпов. . – М.: ВКМ МГУ, 2009. – 129с.	
5	Карпов Ю. Г. Теория и технология программирования. Основы построения трансляторов / Ю. Г. Карпов. – СПб.: БХВ-Петербург, 2005. – 272с.	
6	Вирт Н. Построение компиляторов / Н. Вирт. – М.: ДМК Прес, 2010. – 192с.	
7	Ирвин К. Р. Язык ассемблера для процессоров Intel / К. Р. Ирвин. – М.: Вильямс, 2005. – 912с.	2
8	Калашников О. А. Ассемблер – это просто. Учимся программировать/ О. А. Калашников – СПб.: БХВ-Петербург, 2011. – 336с.	

БГТУ, ФИТ, ПОИТ, 2 семестр, Конструирование программного обеспечения

Введение

План лекции:

- современные подходы к разработке программного обеспечения (скорость разработки, поддержка всего жизненного цикла ПО);
- основные определения;
- примеры.

1. Понятие конструирования программного обеспечения



Конструирование – единственный процесс, который выполняется всегда, – это процесс создания какого-нибудь объекта, может включать в себя некоторые аспекты планирования, проектирования и тестирования.

Компоненты разработки ПО:

- определение проблемы (анализ);
- выработка требований;
- создание плана конструирования;
- разработка архитектуры ПО, или высокоуровневое проектирование;
- детальное проектирование;
- кодирование и отладка;
- блочное тестирование;
- интеграционное тестирование;
- интеграция;
- тестирование системы;
- корректирующее сопровождение.



Конструирование – часть процесса разработки ПО. В зависимости от размера проекта на конструирование обычно уходит 30–80% общего времени работы.

Конструирование занимает центральное место в процессе разработки ПО. Требования к приложению и его архитектура разрабатываются до этапа конструирования, что гарантирует его эффективность. Тестирование системы выполняется после конструирования и служит для проверки его правильности.

Результат конструирования – исходный код (часто является единственным верным описанием программы).

Конструирование – единственный процесс, который выполняется во всех случаях. После конструирования должно быть выполнено исчерпывающее, статистически контролируемое тестирование системы (может отсутствовать).

- **конструирование** – главный этап разработки ПО, без которого не обходится ни один проект;
- **основные этапы конструирования: *детальное проектирование, кодирование, отладка, интеграция и тестирование приложения разработчиками*** (блочное тестирование и интеграционное тестирование);
- **конструирование** часто называют «кодированием» и «программированием»;
- от качества конструирования во многом зависит качество ПО;
- компетентность в конструировании ПО определяет то, насколько хорошим программистом вы являетесь.

Основные решения, которые принимаются при конструировании:

- выбор языка программирования;
- конвенции программирования;
- выбор технологий;
- выбор основных методик конструирования.

2. Основные определения

Система программирования – комплекс программных средств, предназначенных для автоматизации процесса разработки, отладки программного обеспечения и подготовки программного кода к выполнению.

Интегрированная среда разработки – набор инструментов для разработки и отладки программ, имеющий общую интерактивную графическую оболочку, поддерживающую выполнение всех основных функций жизненного цикла разработки программы.

Язык программирования – формальная знаковая система, предназначенная для записи компьютерных программ. Знаковая система определяет набор лексических, синтаксических и семантических правил написания программы (программного кода). Язык программирования представляется в виде набора спецификаций, определяющих его синтаксис и семантику.

Стандарт языка – набор спецификаций, определяющих его синтаксис, семантику, может исторически развиваться.

Парадигмы программирования – совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию).

Исходный код (исходная программа) – программа, написанная на языке программирования, в текстовом формате. Программа на исходном языке (исходный код) готовится с помощью текстовых редакторов и в виде текстового файла или раздела библиотеки поступает на вход транслятора.

Язык ассемблера – машинно-ориентированный язык программирования (для конкретной архитектуры компьютера, команды которого соответствуют машинным командам).

Текстовый редактор – компонента системы программирования (или IDE) – программа, позволяющая подготовить исходный код программы.

Способы реализации языков программирования: компилируемые, интерпретируемые.

Компилятор (транслятор) – программа, преобразующая исходный код на одном языке программирования в исходный код на другом языке.

Интерпретатор – разновидность транслятора. Переводит и выполняет программу с языка высокого уровня в машинный код строка за строкой.

Объектный код – результат работы транслятора. Один файл объектного кода – объектный модуль, файл с расширением `.obj`.

Компоновщик (linker, редактор связей): программа, принимающая один или несколько объектных модулей и формирующая на их основе загрузочный модуль. Если программа собирается из нескольких объектных файлов, компоновщик может собирать эти файлы в единый исполнимый модуль, вычисляя и подставляя адреса вместо символов, в течение времени компоновки (статическая компоновка) или во время исполнения (динамическая компоновка).

Загрузочный код: результат работы компоновщика. Один файл загрузочного кода – загрузочный модуль.

Загрузчик (loader): программа, обычно входящая в состав операционной системы, предназначенная для запуска процесса операционной системы на основе загрузочного модуля.

Препроцессор – программа для обработки текста. Может существовать как отдельная программа, так и быть интегрированной в компилятор. Входные и выходные данные для препроцессора имеют текстовый формат. Препроцессор преобразует текст в соответствии с директивами препроцессора.

Отладчик (debugger) – компонента системы программирования (или IDE) – программа, позволяющая контролировать ход выполнения программы

(приостанавливать, выполнять пошагово), просматривать и изменять области памяти.

Программа – завершённый продукт, пригодный для запуска своим автором на системе, на которой она была разработана.

Программный продукт – программа, работающая без авторского присутствия. Программный продукт выполняется, тестируется, конфигурируется без присутствия автора и сопровождается документацией.

Программное обеспечение (ПО) – совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ (ГОСТ 19781-90).

Жизненный цикл ПО – непрерывный процесс с момента принятия решения о создании ПО до снятия его с эксплуатации.

Объявление – *описание некоторой сущности* (определение типа, сигнатура функции, описание внешней переменной, шаблон и т.п.). Объявление уведомляет компилятор о существовании сущности и её свойствах.

Определение – *реализация некоторой сущности* (переменная, функция, метод класса и т.п.). При обработке определения компилятор генерирует информацию для объектного модуля – исполняемый код, резервирование памяти под переменную и т.д.

3. Примеры

Программа «HelloWorld» выводит сообщение, используя стандартную библиотеку, заголовок которой подключается директивой препроцессора `#include <iostream>`. Программа завершает выполнение с кодом возврата 0.

```
1      #include <iostream>
2
3      int main()
4      {
5          std::cout << "Hello, world!!" << std::endl;
6          system("pause");
7          return 0;
8      }
```

Текущая конфигурация и целевая платформа (DEBUG, X86), на которой разрабатывается проект, отображается в верхней части окна MSVS. Изменить настройки конфигурации можно в **Диспетчере конфигураций**.

Иерархическая структура компонентов в Visual C++.

Глобальный контейнером (компонент, включающий в себя другие компоненты) является **Решение**. Решение может содержать один или несколько проектов.

Проекты являются независимыми компонентами. Они имеют собственную структуру, состоящую из четырех основных каталогов:

Внешние зависимости – содержит ссылки на все модули, которые использует программа.

Файлы заголовков – содержит файлы кода C++ с расширением `h`.

Исходные файлы – содержит файлы кода C++ с расширением `cpp`.

Файлы ресурсов – содержит файлы, непосредственно не относящиеся к языку C++, но необходимые для работы приложения. Например, мультимедийные файлы.

Программный код проекта может иметь сложную структуру и состоять из нескольких файлов исходного кода, конфигурационных файлов и т.п.

Страницы свойств проекта. Разделы свойств. Параметры.

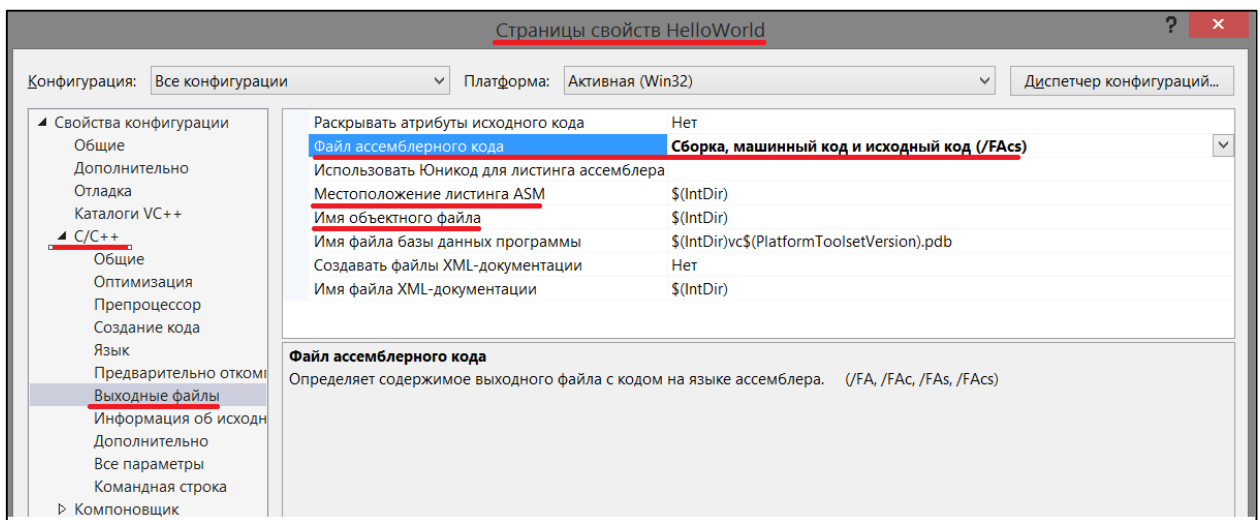
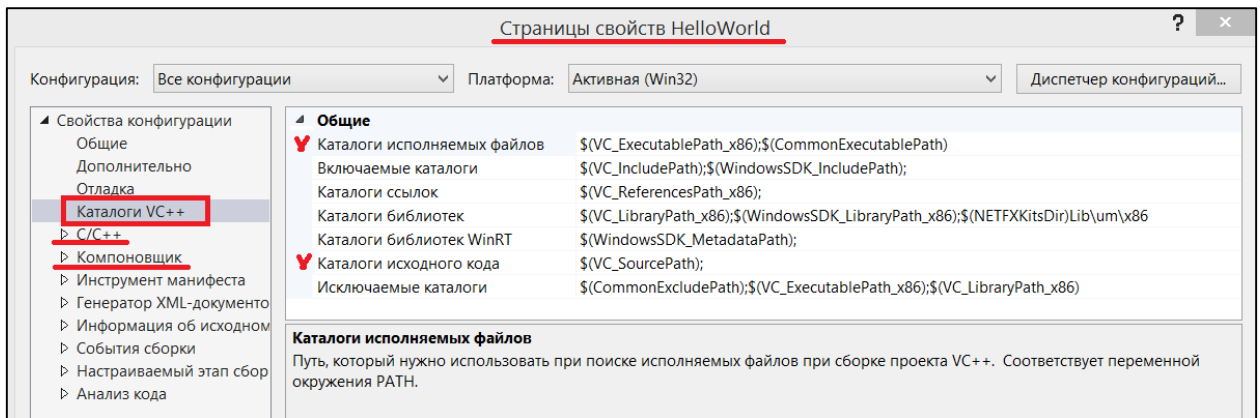
Общие – ключ **Уровень предупреждений** позволяет отключить все предупреждения (`/W0`), либо ужесточить уровень проверок и считать все предупреждения ошибками (`/Wall`).

Предварительно откомпилированные заголовки позволяют их включить/отключить, определить **имя** создаваемого предварительно откомпилированного заголовочного файла и местоположение для полученного выходного файла (с расширением `pch`).

Можно настроить имена и папки для размещения **Выходных файлов**.

Командная строка компилятора C++ отображает, с какими параметрами (ключами) выполняется текущая компиляция.

Раздел **Компоновщик** отображает и позволяет изменить текущие настройки и ключи компоновки.



Папка Debug проекта:

Примеры к лабораторным работам ▸ HelloWorld ▸ HelloWorld ▸ Debug				Поиск: Debug
Имя	Дата изменения	Тип	Размер	
HelloWorld.tlog	12.02.2022 11:01	Папка с файлами		
Hello.cod	12.02.2022 11:01	C/C++ Code Listing	75 КБ	
Hello.obj	12.02.2022 11:01	Object File	58 КБ	
HelloWorld.Build.CppClean.log	12.02.2022 11:01	Текстовый докум...	2 КБ	
HelloWorld.log	12.02.2022 11:01	Текстовый докум...	3 КБ	
HelloWorld.vcxproj.FileListAbsolute.txt	12.02.2022 11:01	Текстовый докум...	0 КБ	
vc142.idb	12.02.2022 11:01	VC++ Minimum R...	147 КБ	
vc142.pdb	12.02.2022 11:01	Program Debug D...	412 КБ	

Папка Debug решения:

Имя	Дата изменения	Тип	Размер
HelloWorld.exe	12.02.2022 11:01	Приложение	49 КБ
HelloWorld.ilink	12.02.2022 11:01	Incremental Linker...	384 КБ
HelloWorld.pdb	12.02.2022 11:01	Program Debug D...	452 КБ

ASM-листинг кода (откомпилирован с ключём /FACS)

```
00025 e8 00 00 00 00 call    @__checkForDebuggerJustMyCode@4
; 5      :      std::cout << "Hello, world!!" << std::endl;

00028 68 00 00 00 00 push    OFFSET ??_C@_0P@EIJKCAOK@Hello?0?5world?$CB
0002d a1 00 00 00 00 mov     eax, DWORD PTR __imp_?cout@std@@3V?$basic_o
00032 50          push    eax
00033 e8 00 00 00 00 call    ???$?6U?$char_traits@D@std@@@std@@YAAAV?$bas
00038 83 c4 08      add     esp, 8
0003b 89 85 3c ff ff
          ff      mov     DWORD PTR tv71[ebp], eax
00041 8b f4          mov     esi, esp
00043 68 00 00 00 00 push    OFFSET ??$endl@DU?$char_traits@D@std@@@std@@
00048 8b 8d 3c ff ff
          ff      mov     ecx, DWORD PTR tv71[ebp]
0004e ff 15 00 00 00
          00      call    DWORD PTR __imp_??6?$basic_ostream@DU?$char_traits@D@st
00054 3b f4          cmp     esi, esp
00056 e8 00 00 00 00 call    __RTC_CheckEsp

; 6      :      system("pause");

0005b 8b f4          mov     esi, esp
0005d 68 00 00 00 00 push    OFFSET ??_C@_05PDJBECF@pause@
```

Файл журнала построения

```
Файл  Правка  Формат  Вид  Справка
Сборка начата 12.02.2022 11:01:15.
1>Проект "D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld\HelloWorld\HelloWorld.vcxproj" в узле 2 (целевые объекты Rebu
1>_PrepareForClean:
  Файл "Debug\HelloWorld.tlog\HelloWorld.lastbuildstate" удаляется.
InitializeBuildStatus:
  Создание "Debug\HelloWorld.tlog\unsuccessfulbuild", так как было задано "AlwaysCreate".
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 /W
_MBCS /Gm- /EHsc /RTC1 /MDd /GS /fp:precise /permissive- /Zc:wchar_t /Zc:forScope /Zc:inline /std:c++17 /FACS /Fa"Debug\\" /Fo"Debug\\" /Fd"Deb
/errorReport:prompt Hello.cpp
Link:
  C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Tools\MSVC\14.23.28105\bin\HostX86\x86\link.exe /ERRORREPORT:PROMPT
лабораторным работам\HelloWorld\Debug\HelloWorld.exe" /INCREMENTAL /NOLOGO kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32
uid.lib odbccp32.lib odbccp32.lib /MANIFEST /MANIFESTUAC:"level='asInvoker' uiAccess='false'" /manifest:embed /DEBUG:FASTLINK /PDB:"D:\Adel\Кафе
\HelloWorld\Debug\HelloWorld.pdb" /SUBSYSTEM:CONSOLE /TLBID:1 /DYNAMICBASE /NXCOMPAT /IMPLIB:"D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным р
/MACHINE:X86 Debug\Hello.obj
HelloWorld.vcxproj -> D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld\Debug\HelloWorld.exe
FinalizeBuildStatus:
  Файл "Debug\HelloWorld.tlog\unsuccessfulbuild" удаляется.
  Обращение к "Debug\HelloWorld.tlog\HelloWorld.lastbuildstate".
1>Сборка проекта "D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld\HelloWorld\HelloWorld.vcxproj" завершена (целевые объ
Сборка успешно завершена.
Предупреждений: 0
Ошибок: 0
Прошло времени 00:00:01.45
```

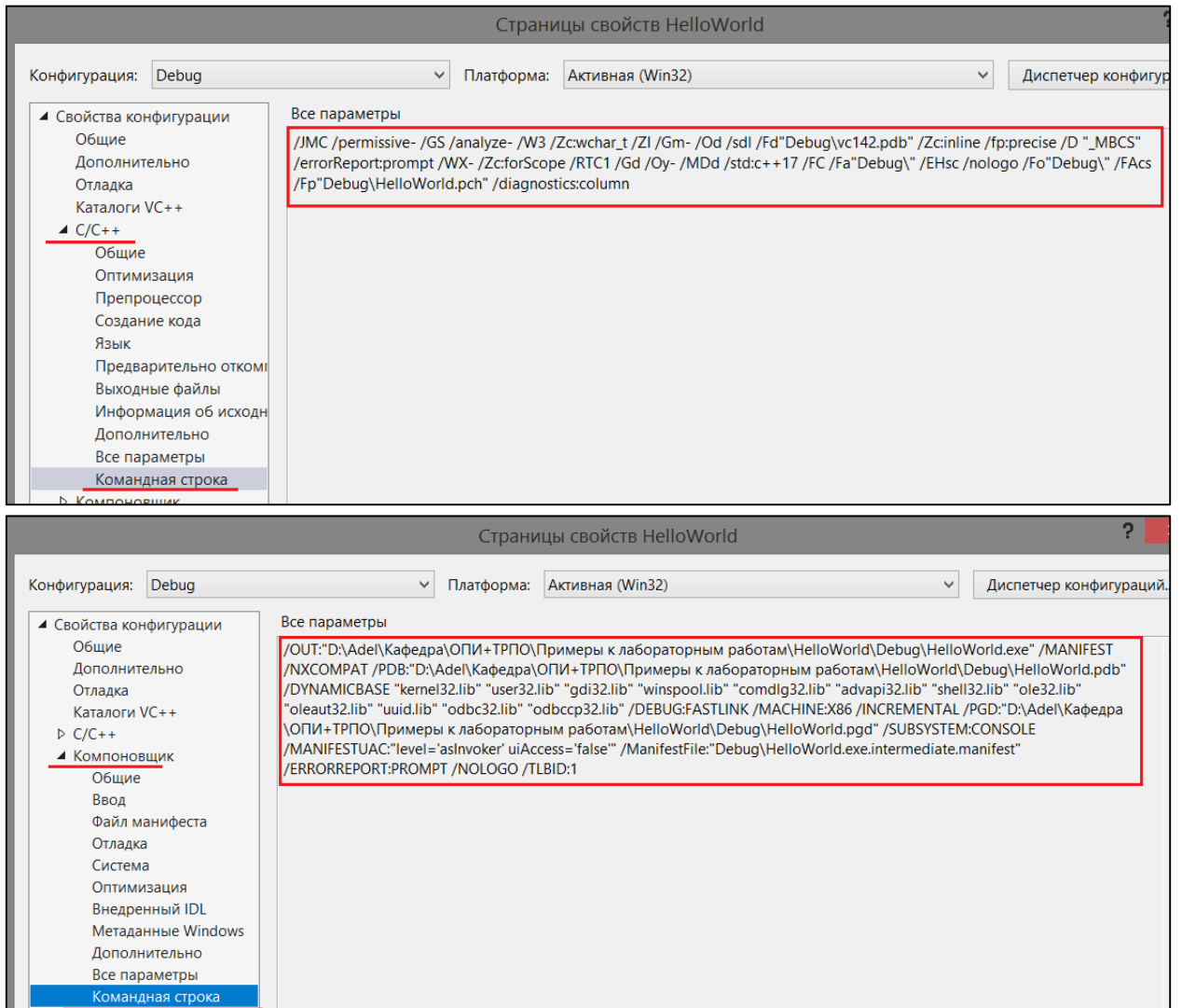
В VS2017 и выше:

для изменения объема сведений, включаемых в журнал сборки необходимо:

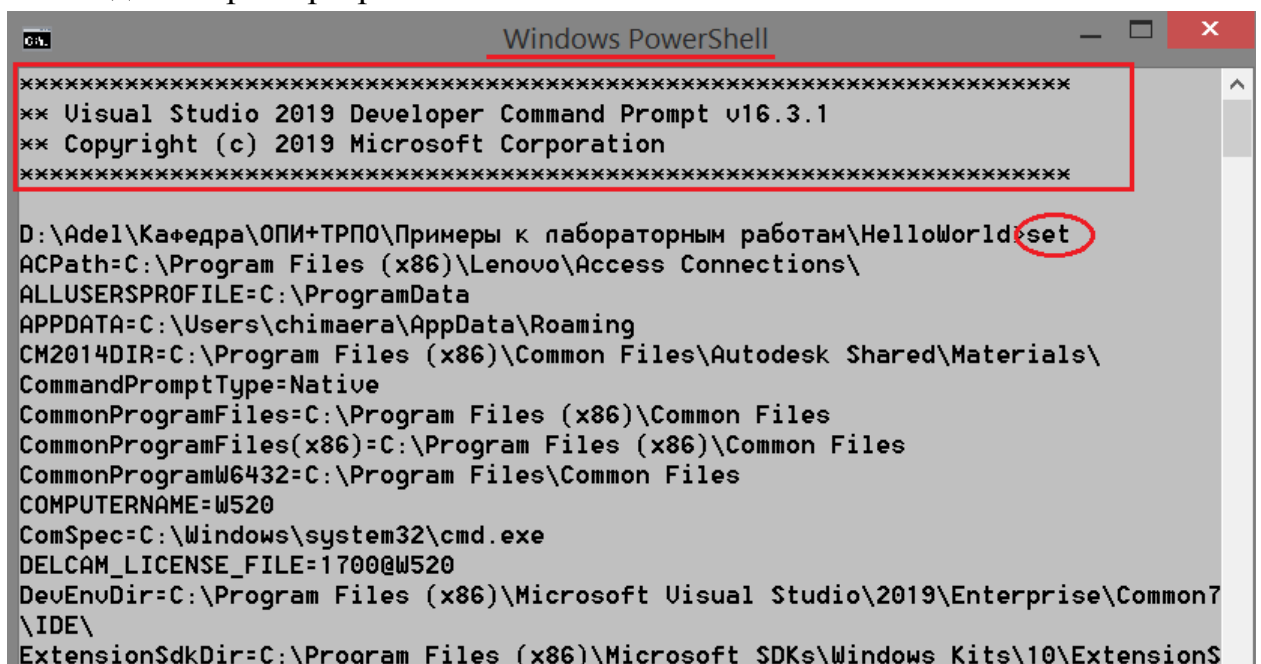
- меню **Сервис (Средства)** -> **Параметры**
- на странице **Проекты** и решения выбрать **Сборка и запуск**
- в списке Степень подробности сообщений при построении проекта MSBuild выбрать **Обычный** и нажать ОК

Обычный – отображает сводку о сборке, ошибки, предупреждения и сообщения с высокой степенью важности, а также основные шаги сборки.

4. Компиляция и компоновка в командной строке



Командная строка разработчика:



```
D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld>c1
Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.23.28105.4 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

использование: c1 [ параметр... ] имя_файла... [ /link параметр_компоновки... ]
```

```
D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld>link
Microsoft (R) Incremental Linker Version 14.23.28105.4
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
использование: LINK [параметры] [файлы] [@командный_файл]
```

```
параметры:
```

```
/ALIGN:#
/ALLOWBIND[:NO]
/ALLOWISOLATION[:NO]
/APPCONTAINER[:NO]
/ASSEMBLYDEBUG[:DISABLE]
/ASSEMBLYLINKRESOURCE:имя_файла
/ASSEMBLYMODULE:имя_файла
/ASSEMBLYRESOURCE:имя_файла[, [имя][,PRIVATE]]
/BASE:{адрес[,размер]|@имя_файла,ключ}
/CLRIMAGETYPE:{IJW|PURE|SAFE|SAFE32BITPREFERRED}
/CLRLOADEROPTIMIZATION:{MD|MDH|NONE|SD}
/CLRSUPPORTLASTERROR[:{NO|SYSTEMDLL}]
/CLRTHREADATTRIBUTE:{MTA|NONE|STA}
/CLRUNMANAGEDCODECHECK[:NO]
/DEBUG[:{FASTLINK|FULL|NONE}]
/DEF:имя_файла
/DEFAULTLIB:библиотека
/DELAY:{NOBIND|UNLOAD}
/DELAYLOAD:dll
/DELAYSIGN[:NO]
```

```
(для продолжения нажмите клавишу ВВОД)
```

```
D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld>cd D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello
```

```
D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello>dir
```

```
Том в устройстве D не имеет метки.
```

```
Серийный номер тома: 34DB-9113
```

```
Содержимое папки D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello
```

```
12.02.2022  11:47    <DIR>          .
12.02.2022  11:47    <DIR>          ..
12.02.2022  10:59                120 Hello.cpp
                1 файлов                120 байт
                2 папок    2а676а199а424 байт свободно
```

```
D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello>
```

5. Простой (сокращенный) вариант

```
D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello>cl Hello.cpp /c /EHsc
Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.23.28105.4 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Hello.cpp

D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello>dir
Том в устройстве D не имеет метки.
Серийный номер тома: 34DB-9113

Содержимое папки D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello

12.02.2022  11:54    <DIR>          .
12.02.2022  11:54    <DIR>          ..
12.02.2022  10:59                120 Hello.cpp
12.02.2022  11:54           94a943 Hello.obj
                2 файлов             95a063 байт
                2 папок       2a676a101a120 байт свободно
```

Ключи компилятора:

/c – компиляция без компоновки;

/EHsc – модель обработки исключений (перехватываются исключения C++).

```
D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello>link Hello.obj /out:Hello.exe
Microsoft (R) Incremental Linker Version 14.23.28105.4
Copyright (C) Microsoft Corporation. All rights reserved.

D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello>dir
Том в устройстве D не имеет метки.
Серийный номер тома: 34DB-9113

Содержимое папки D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello

12.02.2022  11:55    <DIR>          .
12.02.2022  11:55    <DIR>          ..
12.02.2022  10:59                120 Hello.cpp
12.02.2022  11:55           193a536 Hello.exe
12.02.2022  11:54           94a943 Hello.obj
                3 файлов             200a533 байт
                2 папок       2a675a904a512 байт свободно
```

Ключи компоновщика:

/out – указывает имя загрузочного файла.

Выполнение приложения из командной строки разработчика:

```
D:\Adel\Кафедра\КПО\Примеры к лабораторным работам\Hello>Hello.exe
Hello, world!!
Для продолжения нажмите любую клавишу . . .
```

6. Загрузчик

```
#include "stdafx.h"
#include <Windows.h>

int _tmain(int argc, _TCHAR* argv[])
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    ZeroMemory(&si, sizeof(STARTUPINFO));
    si.cb=sizeof(STARTUPINFO);
    BOOL rc = CreateProcess(L"d:\\PLab01\\LP_Lab01.exe",
                           NULL, NULL, NULL, FALSE, CREATE_NEW_CONSOLE, NULL, NULL, &si, &pi);

    return 0;
}
```

API-функция `CreateProcessA` создает новый процесс и его основной поток.

Синтаксис:

```
BOOL CreateProcessA(
    [in, optional] LPCSTR          lpApplicationName,
    [in, out, optional] LPSTR       lpCommandLine,
    [in, optional] LPSECURITY_ATTRIBUTES lpProcessAttributes,
    [in, optional] LPSECURITY_ATTRIBUTES lpThreadAttributes,
    [in] BOOL bInheritHandles,
    [in] DWORD dwCreationFlags,
    [in, optional] LPVOID lpEnvironment,
    [in, optional] LPCSTR lpCurrentDirectory,
    [in] LPSTARTUPINFOA lpStartupInfo,
    [out] LPPROCESS_INFORMATION lpProcessInformation
);
```

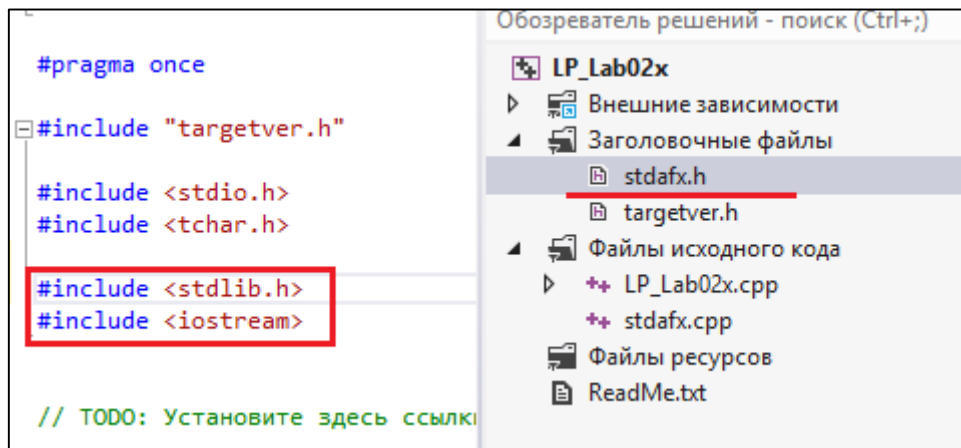
<code>LPCTSTR lpApplicationName</code>	// имя исполняемого модуля
<code>LPTSTR lpCommandLine</code>	// командная строка
<code>LPSECURITY_ATTRIBUTES lpProcessAttributes</code>	// SD (дескриптор безопасности)
<code>LPSECURITY_ATTRIBUTES lpThreadAttributes</code>	// SD
<code>BOOL bInheritHandles</code>	// дескриптор параметра наследования
<code>DWORD dwCreationFlags</code>	// флажки создания
<code>LPVOID lpEnvironment</code>	// новый блок конфигурации
<code>LPCTSTR lpCurrentDirectory</code>	// имя текущего каталога
<code>LPSTARTUPINFO lpStartupInfo</code>	// информация предустановки
<code>LPPROCESS_INFORMATION lpProcessInformation</code>	// информация о процессе

`ZeroMemory(&si, sizeof(si));`

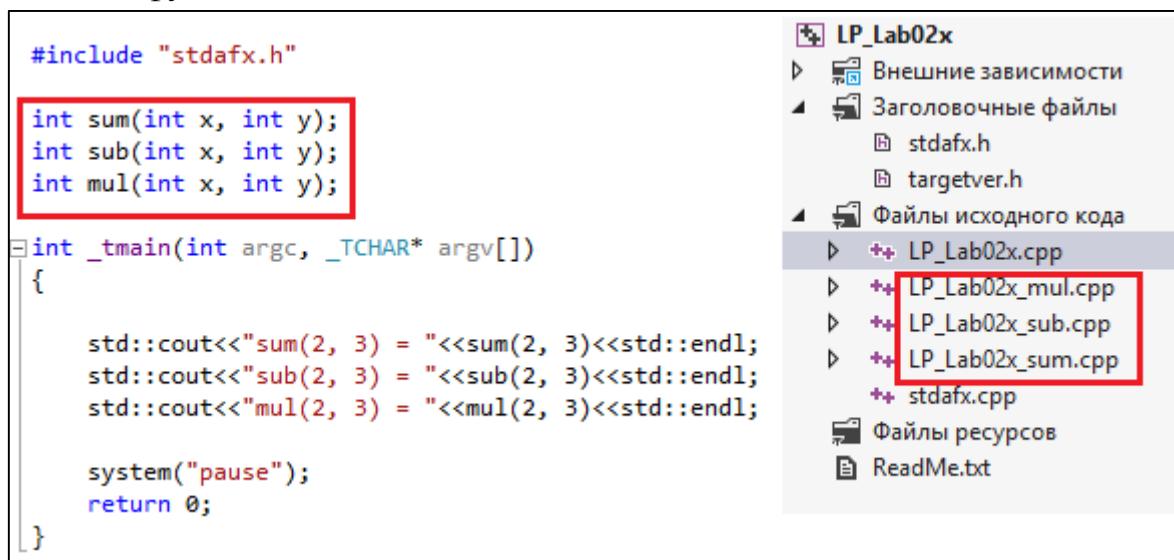
гарантирует, что все поля структуры `si` инициализированы значением 0 или `NULL`, предотвращая ошибки из-за мусора в памяти.

7. Компиляция и компоновка в командной строке многофайлового проекта

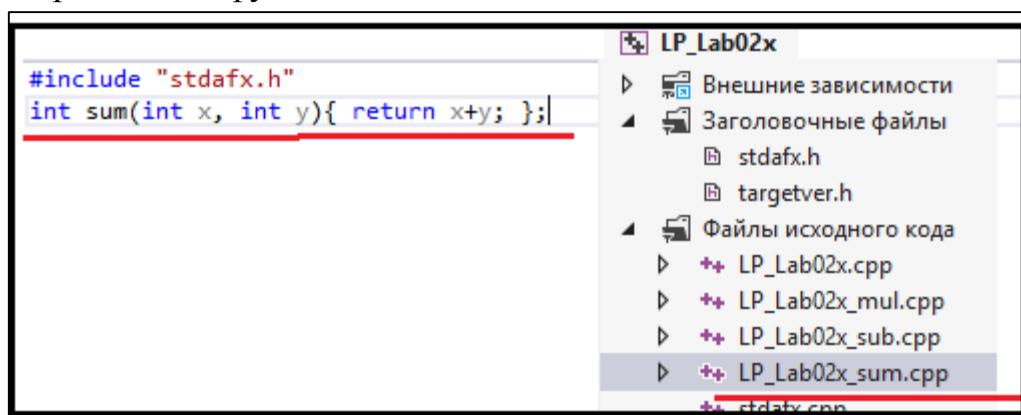
Заголовочный файл:

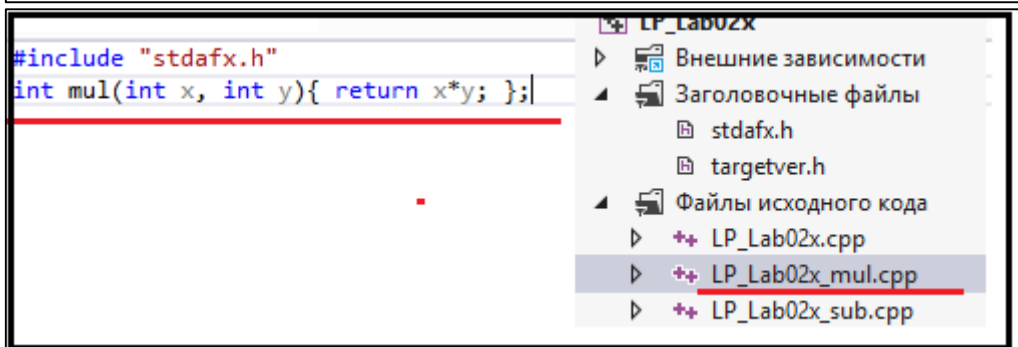
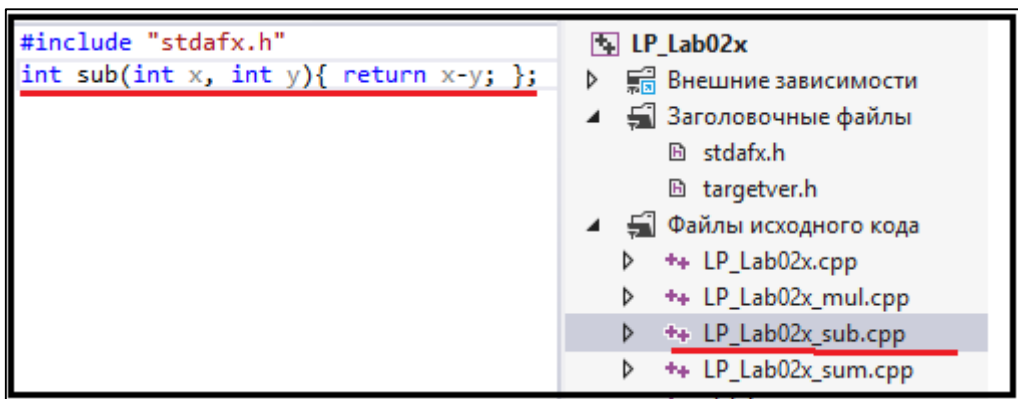


Главная функция:



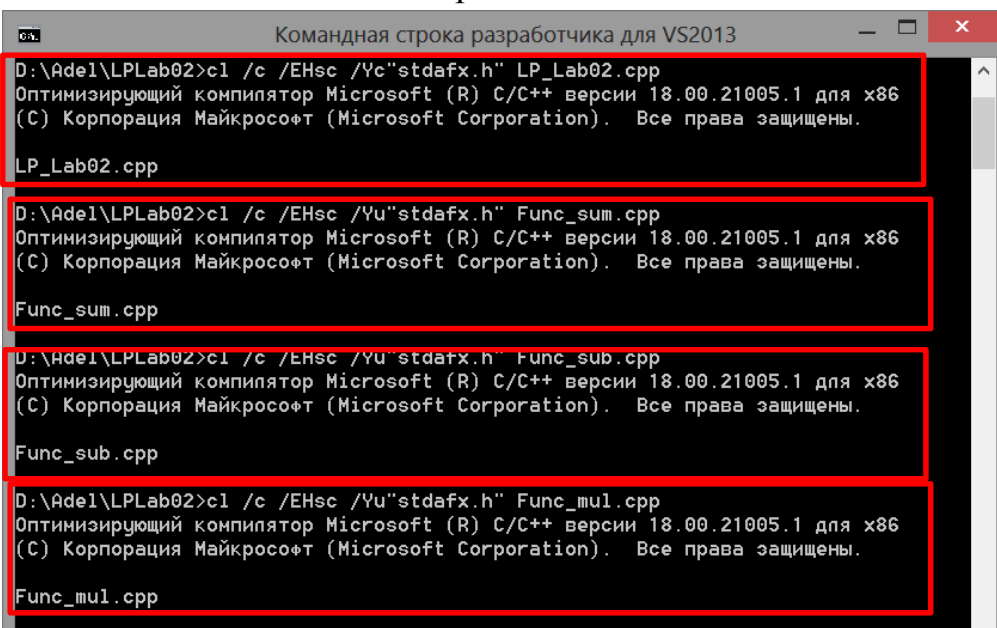
Определения функций:





WORK (D:) ▸ PLab02		Поиск: PLab02
Имя	Дата изменения	
++ LP_Lab02x.cpp	26.01.2015 0:40	
++ LP_Lab02x_mul.cpp	26.01.2015 0:40	
++ LP_Lab02x_sub.cpp	26.01.2015 0:40	
++ LP_Lab02x_sum.cpp	26.01.2015 0:40	
++ stdafx.cpp	26.01.2015 0:11	
stdafx.h	26.01.2015 0:40	

Компиляция в командной строке:



кальный диск (D:) ▸ Adel ▸ LPLab02

Имя	Дата изменения	Тип
Func_mul.cpp	17.01.2017 17:05	C++ Source
<u>Func_mul.obj</u>	17.01.2017 19:08	Object File
Func_sub.cpp	17.01.2017 17:04	C++ Source
<u>Func_sub.obj</u>	17.01.2017 19:08	Object File
Func_sum.cpp	17.01.2017 17:03	C++ Source
<u>Func_sum.obj</u>	17.01.2017 19:08	Object File
LP_Lab02.cpp	17.01.2017 16:02	C++ Source
<u>LP_Lab02.obj</u>	17.01.2017 19:07	Object File
stdafx.cpp	17.01.2017 15:14	C++ Source
stdafx.h	17.01.2017 15:24	C/C++ Header
stdafx.pch	17.01.2017 19:07	Precompiled Head...
targetver.h	17.01.2017 15:14	C/C++ Header

Компоновка в командной строке:

```
D:\Adel\LPLab02>link /out:LP_Lab02.exe LP_Lab02.obj Func_sum.obj Func_sub.obj Func_mul.obj
Microsoft (R) Incremental Linker Version 12.00.21005.1
Copyright (C) Microsoft Corporation. All rights reserved.
```

кальный диск (D:) ▸ Adel ▸ LPLab02

Имя	Дата изменения	Тип
Func_mul.cpp	17.01.2017 17:05	C++ Source
Func_mul.obj	17.01.2017 19:08	Object File
Func_sub.cpp	17.01.2017 17:04	C++ Source
Func_sub.obj	17.01.2017 19:08	Object File
Func_sum.cpp	17.01.2017 17:03	C++ Source
Func_sum.obj	17.01.2017 19:08	Object File
LP_Lab02.cpp	17.01.2017 16:02	C++ Source
<u>LP_Lab02.exe</u>	17.01.2017 19:29	Приложение
LP_Lab02.obj	17.01.2017 19:07	Object File
stdafx.cpp	17.01.2017 15:14	C++ Source
stdafx.h	17.01.2017 15:24	C/C++ Header
stdafx.pch	17.01.2017 19:07	Precompiled Head...
targetver.h	17.01.2017 15:14	C/C++ Header

8. Приложение А.

Параметры компилятора C++

<https://docs.microsoft.com/kk-kz/cpp/build/reference/compiler-options-listed-alphabetically?view=vs-2017>

Параметр	Цель
@	Указывает файл ответа.
/?	Отображает список параметров компилятора.
/AI	Указывает каталог поиска для разрешения ссылок на файлы, указанные в директиве #using .
/analyze	Включение анализа кода.
/arch	Задаёт архитектуру для создания кода.
/await	Включите расширения сопрограммы (возобновляемые функции).
/bigobj	Увеличивает число адресуемых секций в OBJ-файле.
/C	Сохраняет комментарии на этапе предварительной обработки.
/c	Задаёт компиляцию без компоновки.
/cgthreads	Задаёт число потоков cl.exe, используемых для оптимизации и создания кода.
/clr	Создаёт выходной файл, предназначенный для выполнения в среде CLR.
/constexpr	Управлять вычислением constexpr во время компиляции.
/D	Определяет константы и макросы.
/Diagnostics	Определяет формат диагностических сообщений.
/doc	Сведение документирующих комментариев в XML-файл.
/E	Копирует выходные данные препроцессора в стандартный вывод.
/EH	Задаёт модель обработки исключений.
/EP	Копирует выходные данные препроцессора в стандартный вывод.
/errorReport	Разрешает передавать данные о внутренних ошибках компилятора (ICE) непосредственно в группу Visual C++.
/Execution-CharSet	Задание набора символов исполнения.
/F	Задаёт размер стека.
/favor	Создаёт код, который оптимизирован для конкретных x64 архитектуры или для специфики микроархитектур в AMD64 и расширенной памяти 64 архитектурах технологии (EM64T).
/FA	Создаёт файл листинга.
/Fa	Задаёт имя файла листинга.
/FC	Вывод полного пути файлов исходного кода, переданных программе cl.exe, в диагностическом тексте.
/Fd	Переименовывает файл базы данных программы.
/Fe	Переименовывает исполняемый файл.

Параметр	Цель
/FI	Выполняет предварительную обработку указанного включаемого файла.
/Fi	Задаёт предобработанное имя выходного файла.
/Fm	Создаёт файл сопоставления.
/Fo	Создаёт объектный файл.
/fp	Задаёт поведение чисел с плавающей запятой.
/Fp	Задаёт имя файла предкомпилированного заголовка.
/FR /Fr	Создаёт файлы браузера. /Fr не рекомендуется к использованию.
/FS	Обеспечивает принудительную сериализацию записей в файл базы данных программы (PDB) с помощью MSPDBSRV.EXE.
/FU	Принудительное использование имени файла, как если бы оно было указано в директиве #using .
/Fx	Включает введенный код в исходный файл.
/GA	Выполняет оптимизацию кода для приложений Windows.
/Gd	Использует соглашение о вызовах <code>__cdecl</code> (только архитектура x86).
/Ge	Не рекомендуется. Включает стековые зонды.
/GF	Включает объединение строк.
/GH	Вызывает функцию-обработчик <code>_pexit</code> .
/Gh	Вызывает функцию-обработчик <code>_penter</code> .
/GL	Включает оптимизацию всей программы.
/Gm	Включает минимальное перепостроение.
/GR	Включает информацию о типах во время выполнения (RTTI).
/Gr	Использует соглашение о вызовах <code>__fastcall</code> (только архитектура x86).
/GS	Буферизует проверку безопасности.
/Gs	Управляет стековыми зондами.
/GT	Поддерживает безопасность относительно волокон для данных, размещаемых с помощью статической локальной памяти потока.
/guard:cf	Добавление проверок безопасности для защиты потока управления.
/Gv	Использует соглашение о вызовах <code>__vectorcall</code> . (только x86 и x64)
/Gw	Включает глобальную оптимизацию данных всей программы.
/GX	Не рекомендуется. Включает синхронную обработку исключений. Используйте вместо этого параметр /EH .
/Gy	Включает компоновку на уровне функций.
/GZ	Не рекомендуется. Аналогично /RTC1 .

Параметр	Цель
/Gz	Использует соглашение о вызовах __stdcall (только архитектура x86).
/H	Не рекомендуется. Ограничивает длину внешних (открытых) имен.
/HELP	Отображает список параметров компилятора.
/homeparams	Принудительная запись параметров, переданных в регистрах, в соответствующие места в стеке при вхождении в функцию. Этот параметр компилятора предназначен только для x64 компиляторы (собственные и кросс-компиляция).
/hotpatch	Создает образ, допускающий горячее обновление.
/I	Осуществляет поиск включаемых файлов в каталоге.
/J	Изменяет тип char по умолчанию.
/JMC	Поддерживает отладку собственного C++ Just My Code.
/kernel	Компилятор и компоновщик создадут двоичный файл для выполнения в ядре Windows.
/LD	Создает библиотеку динамической компоновки.
/LDd	Создает отладочную библиотеку динамической компоновки.
/link	Передаёт указанный параметр в программу LINK.
/LN	Создает модуль MSIL.
/MD	Создает многопоточную библиотеку DLL с помощью библиотеки MSVCRT.lib.
/MDd	Создает отладочную многопоточную библиотеку DLL с помощью библиотеки MSVCRTD.lib.
/MP	Компилирует несколько исходных файлов с помощью нескольких процессов.
/MT	Создает многопоточный исполняемый файл с помощью библиотеки LIBCMT.lib.
/MTd	Создает отладочный многопоточный исполняемый файл с помощью библиотеки LIBCMTD.lib.
/nologo	Подавление отображения приветствия.
/O1	Уменьшает размер кода.
/O2	Создает быстрый код.
/Ob	Управляет подстановкой подставляемых функций.
/Od	Отключает оптимизацию.
/Og	Не рекомендуется. Использует глобальную оптимизацию.
/Oi	Создает встроенные функции.
/openmp	Включает прагма-директиву #pragma omp в исходном коде.
/Os	Отдает приоритет уменьшению размера кода.
/Ot	Отдает приоритет быстрому коду.
/Ox	Использует максимальную оптимизацию (/Ob2gity /Gs).

Параметр	Цель
<u>/Oy</u>	Отказ от использования указателя фрейма (только архитектура x86).
<u>/P</u>	Записывает выходные данные препроцессора в файл.
<u>/permissive-</u>	Режим соответствия standard.
<u>/Qfast_transcendentals</u>	Создает быстрые трансцендентные функции.
<u>/Qlfist</u>	Не рекомендуется. Подавляет использование функции <code>_ftol</code> при необходимости преобразования из типа с плавающей запятой в целочисленный тип (только архитектура x86).
<u>/Qimprecise_fwaits</u>	Удаляет команды <code>fwait</code> внутри блоков <code>try</code> .
<u>/Qpar (автоматический параллелизатор)</u>	Включает автоматическую параллелизацию циклов, которые помечены с помощью директивы <code>#pragma loop()</code> .
<u>/Qsafe_fp_loads</u>	Использует целочисленные инструкции перемещения значений с плавающей запятой и отключает определенные оптимизации загрузки значений с плавающей запятой.
<u>/Qvec/report (уровень отчетности автоматического векторизатора)</u>	Включает уровни отчетов для автоматической векторизации.
<u>/RTC</u>	Включает проверку ошибок во время выполнения.
<u>/sdl</u>	Включает дополнительные функции безопасности и предупреждения.
<u>/showIncludes</u>	Отображает список включаемых файлов во время компиляции.
<u>кодировки/Source</u>	Задание исходной кодировки.
<u>/std</u>	Селектор совместимости стандартной версии C++.
<u>/Tc</u>	Указывает исходный файл на языке C.
<u>/TC</u>	Указывает, что все исходные файлы, C.
<u>/Tp</u>	Указывает исходный файл на языке C++.
<u>/TP</u>	Указывает, что все исходные файлы C++.
<u>/U</u>	Удаляет предварительно определенный макрос.
<u>/u</u>	Удаляет все предварительно определенные макросы.
<u>/utf-8</u>	Набор источника и выполнения кодировки UTF-8.
<u>/V</u>	Не рекомендуется. Задает строку версии OBJ-файла.
<u>/ Validate/CharSet</u>	Проверка файлов UTF-8 только совместимости символов.
<u>/vd</u>	Подавляет или включает скрытые <code>vtordisp</code> -члены класса.
<u>/vmb</u>	Использует оптимальное основание для указателей на члены.
<u>/vmg</u>	Использует полное обобщение для указателей на члены.
<u>/vmm</u>	Объявляет множественное наследование.
<u>/vms</u>	Объявляет одиночное наследование.
<u>/vmv</u>	Объявляет виртуальное наследование.
<u>/volatile</u>	Выбирает способ интерпретации ключевого слова <code>volatile</code> .

Параметр	Цель
/w	Отключает все предупреждения.
/W0, /W1, /W2, /W3, /W4	Задаёт уровень предупреждения для вывода.
/w1, /w2, /w3, /w4	Задаёт уровень для указанного предупреждения.
/Wall	Включает все предупреждения, в том числе предупреждения, отключенные по умолчанию.
/wd	Отключает указанное предупреждение.
/we	Обрабатывает указанное предупреждение как ошибку.
/WL	Включает однострочные диагностические сообщения об ошибках и предупреждения в ходе компиляции исходного кода C++ из командной строки.
/wo	Отображает указанное предупреждение только один раз.
/Wp64	Является устаревшей. Выявляет проблемы 64-битной переносимости.
/Wv	Не отображает предупреждения, появившиеся после указанной версии компилятора.
/WX	Обрабатывает предупреждения как ошибки.
/X	Пропускает стандартный каталог включаемых файлов.
/Y-	Пропускает все прочие параметры компилятора, относящиеся к предварительно скомпилированным заголовкам, в текущем построении.
/Yc	Создаёт файл предкомпилированного заголовка.
/Yd	Не рекомендуется. Размещает полную отладочную информацию во всех объектных файлах. Используйте вместо этого параметр /ZI .
/Yl	Вводит ссылку РСН при создании отладочной библиотеки.
/Yu	Использует файл предкомпилированного заголовка при построении.
/Zl	Приводит к возникновению ошибки совместимости с C 7.0 отладочную информацию.
/Za	Отключает расширения языка.
/Zc	Задаёт стандартное поведение /Ze . / /Za , /Ze (отключить расширения языка)
/Ze	Не рекомендуется. Включает расширения языка.
/Zf	Улучшает время создания в параллельные сборки PDB-файла.
/Zg	Удален в Visual C++ 2015. Создает прототипы функций.
/ZI	Включает отладочную информацию в базу данных программы, совместимую с функцией "Изменить и продолжить".
/Zi	Создаёт полную отладочную информацию.
/Zl	Удаляет имя библиотеки по умолчанию из файла OBJ (только архитектура x86).

Параметр	Цель
/Zm	Указывает предел выделения памяти для предкомпилированного заголовка.
/Zp	Упаковывает члены структур.
/Zs	Проверяет только синтаксис.
/ZW	Создает выходной файл для запуска в среде выполнения Windows.

9. Приложение В

Параметры компоновщика

<https://docs.microsoft.com/kk-kz/cpp/build/reference/linker-options?view=vs-2017>

Параметр	Цель
@	Указывает файл ответа.
/ALIGN	Задаёт выравнивание каждой секции.
/ALLOWBIND	Указывает на то, что библиотека DLL не может быть привязана.
/ALLOWISOLATION	Задаёт поведение нахождения файлов манифеста.
/APPCONTAINER	Определяет, должно ли приложение выполняться в среде процесса контейнера приложений.
/ASSEMBLYDEBUG	Добавляет атрибут DebuggableAttribute в управляемый образ.
/ASSEMBLYLINKRESOURCE	Создаёт ссылку на управляемый ресурс.
/ASSEMBLYMODULE	Указывает на то, что в сборку должен быть импортирован модуль MSIL.
/ASSEMBLYRESOURCE	Внедряет файл управляемых ресурсов в сборку.
/BASE	Задаёт базовый адрес для программы.
/CGTHREADS	Задаёт число потоков cl.exe, используемых для оптимизации и создания кода, если задано создание кода во время компоновки.
/CLRIMAGETYPE	Задаёт тип (IJW, pure или safe) CLR-образа.
/CLRSUPPORTLASTERROR	Сохраняет последний код ошибки функций, вызываемых с помощью механизма P/Invoke.

<u>/CLRTHREADATTRIBUTE</u>	Указывает атрибут потока для применения к точке входа CLR-программы.
<u>/CLRUNMANAGEDCODECHECK</u>	Указывает, должен ли компоновщик применять атрибут SuppressUnmanagedCodeSecurity к создаваемым компоновщиком заглушкам PInvoke, осуществляющим вызовы из управляемого кода в библиотеки DLL неуправляемого кода.
<u>/DEBUG</u>	Создает отладочную информацию.
<u>/DEBUGTYPE</u>	Указывает, какие данные необходимо включить в отладочную информацию.
<u>/DEF</u>	Передает компоновщику файл определения модуля (DEF).
<u>/DEFAULTLIB</u>	Проводит поиск по указанной библиотеке при разрешении внешних ссылок.
<u>/DELAY</u>	Управляет отложенной загрузкой библиотек DLL.
<u>/DELAYLOAD</u>	Включает отложенную загрузку указанной библиотеки DLL.
<u>/DELAYSIGN</u>	Частично подписывает сборку.
<u>/DEPENDENTLOADFLAG</u>	Задаёт флаги по умолчанию для зависимой загрузки DLL.
<u>/DLL</u>	Выполняет сборку библиотеки DLL.
<u>/DRIVER</u>	Создает драйвер режима ядра.
<u>/DYNAMICBASE</u>	Указывает, следует ли создавать исполняемый образ, базовый адрес которого может быть случайным образом изменен во время загрузки с помощью технологии ASLR.
<u>/ENTRY</u>	Задаёт начальный адрес.
<u>/errorReport</u>	Передаёт сведения о внутренних ошибках компоновщика в Майкрософт.
<u>/EXPORT</u>	Экспортирует функцию.
<u>/FILEALIGN</u>	Выравнивание разделов в выходном файле на кратные с указанным значением.
<u>/FIXED</u>	Создаёт программу, которая может загружаться только по предпочтительному базовому адресу.

<u>/FORCE</u>	Принудительное завершение компоновки даже в случае наличия неразрешенных или многократно определенных символов.
<u>/FUNCTIONPADMIN</u>	Создает образ, для которого можно выполнять горячее обновление.
<u>/GENPROFILE,</u> <u>/FASTGENPROFILE</u>	Оба эти параметра задают создание PGD-файла компоновщиком для поддержки профильной оптимизации (PGO). /GENPROFILE и /FASTGENPROFILE используют разные параметры по умолчанию.
<u>/GUARD</u>	Включает защиту потока управления.
<u>/HEAP</u>	Задаёт размер кучи в байтах.
<u>/HIGHENTROPYVA</u>	Определяет поддержку 64-разрядной функции Address Space Layout Randomization (ASLR) с высоким уровнем энтропии.
<u>/IDLOUT</u>	Указывает имя файла IDL и имена других выходных файлов MIDL.
<u>/IGNORE</u>	Отменяет вывод указанных предупреждений компоновщика.
<u>/IGNOREIDL</u>	Предотвращает преобразование сведений атрибутов в файл IDL.
<u>/IMPLIB</u>	Переопределяет имя библиотеки импорта по умолчанию.
<u>/INCLUDE</u>	Принудительное использование ссылок на символы.
<u>/INCREMENTAL</u>	Управляет инкрементной компоновкой.
<u>/INTEGRITYCHECK</u>	Указывает на то, что модуль требует проверки подписи во время загрузки.
<u>/KEYCONTAINER</u>	Задаёт контейнер ключей для подписи сборки.
<u>/KEYFILE</u>	Задаёт ключ или пару ключей для подписи сборки.
<u>/LARGEADDRESSAWARE</u>	Указывает компилятору на то, что приложение поддерживает адреса, превышающие два гигабайта.
<u>/LIBPATH</u>	Указывает путь для поиска перед путем среды библиотеки.
<u>/LTCG</u>	Задаёт создание кода во время компоновки.

<u>/MACHINE</u>	Указывает целевую платформу.
<u>/MANIFEST</u>	Создает параллельный файл манифеста и при необходимости включает его в двоичный файл.
<u>/MANIFESTDEPENDENCY</u>	Указывает <dependentAssembly> раздела в файле манифеста.
<u>/MANIFESTFILE</u>	Изменяет имя файла манифеста по умолчанию.
<u>/MANIFESTINPUT</u>	Задаёт входной файл манифеста для обработки и внедрения компоновщиком в двоичный файл. Этот параметр можно использовать несколько раз, чтобы указать несколько входных файлов манифеста.
<u>/MANIFESTUAC</u>	Указывает, следует ли внедрять в манифест программы сведения о контроле учетных записей.
<u>/MAP</u>	Создает файл сопоставления.
<u>/MAPINFO</u>	Включает указанные сведения в файл сопоставления.
<u>/MERGE</u>	Объединяет разделы.
<u>/MIDL</u>	Задаёт параметры командной строки MIDL.
<u>/NATVIS</u>	Добавляет визуализаторы отладчика из файла Natvis в PDB-ФАЙЛ.
<u>/NOASSEMBLY</u>	Подавляет создание сборки .NET Framework.
<u>/NODEFAULTLIB</u>	Пропускает все (или только указанные) библиотеки по умолчанию при разрешении внешних ссылок.
<u>/NOENTRY</u>	Создает библиотеку DLL, содержащую только ресурсы.
<u>/NOLOGO</u>	Отключает загрузочный баннер.
<u>/NXCOMPAT</u>	Помечает исполняемый файл как файл, проверенный на совместимость с компонентом предотвращения выполнения данных Windows.
<u>/OPT</u>	Управляет оптимизацией LINK.
<u>/ORDER</u>	Помещает секции COMDAT в образ в предопределенном порядке.
<u>/OUT</u>	<u>Задаёт имя выходного файла.</u>

<u>/PDB</u>	Создает файл базы данных программы (PDB).
<u>/PDBALTPATH</u>	Использует альтернативное местоположение для сохранения файла PDB.
<u>/PDBSTRIPPED</u>	Создает файл базы данных программы (PDB), не содержащий закрытых символов.
<u>/PGD</u>	Задаёт файл PGD для профильных оптимизаций.
<u>/POGOSAFEMODE</u>	Устаревшие создает сборку инструментирования профильной Оптимизации поточно ориентированными.
<u>/PROFILE</u>	Создает выходной файл, который может быть использован для профилировщика производительности инструментов.
<u>/RELEASE</u>	Задаёт контрольную сумму в заголовке файла EXE.
<u>/SAFESEH</u>	Указывает на то, что образ будет содержать таблицу безопасных обработчиков исключений.
<u>/SECTION</u>	Переопределяет атрибуты секции.
<u>/SOURCELINK</u>	Указывает файл SourceLink для добавления в PDB.
<u>/STACK</u>	Задаёт размер стека (в байтах).
<u>/STUB</u>	Присоединяет программу-заглушку MS-DOS к программе Win32.
<u>/SUBSYSTEM</u>	Указывает операционной системе, как запускать файл EXE.
<u>/SWAPRUN</u>	Указывает операционной системе на необходимость копирования выходного файла компоновщика в файл подкачки перед его запуском.
<u>/TLBID</u>	Указывает идентификатор ресурса библиотеки типов, создаваемой компоновщиком.
<u>/TLBOUT</u>	Указывает имя файла TLB и имена других выходных файлов MIDL.
<u>/TSAWARE</u>	Создает приложение, специально рассчитанное на запуск под управлением сервера терминалов.
<u>/USEPROFILE</u>	Использует профильной оптимизации обучающих данных для создания оптимизированного образа.

<u>/VERBOSE</u>	Печатает сообщения хода выполнения компоновщика.
<u>/VERSION</u>	Присваивает номер версии.
<u>/WHOLEARCHIVE</u>	Включает в себя каждого файла объект из указанного статических библиотек.
<u>/WINMD</u>	Включает создание файлов метаданных среды выполнения Windows.
<u>/WINMDFILE</u>	Задаёт имя файла для выходного файла метаданных среды выполнения Windows (winmd), создаваемого параметром компоновщика <u>/WINMD</u> .
<u>/WINMDKEYFILE</u>	Задаёт ключ или пару ключей для подписи файла метаданных среды выполнения Windows.
<u>/WINMDKEYCONTAINER</u>	Указывает контейнер ключей для подписания файла метаданных Windows.
<u>/WINMDDELAYSIGN</u>	Частично подписывает файл метаданных среды выполнения Windows (.winmd), установив открытый ключ в файле winmd.
<u>/WX</u>	Обрабатывает предупреждения компоновщика как ошибки.