

**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 | Молchanov A.YU. 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 – завершенный продукт, пригодный для запуска своим автором на системе, на которой она была разработана.

Программный продукт – программа, работающая без авторского присутствия. Программный продукт исполняется, 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.

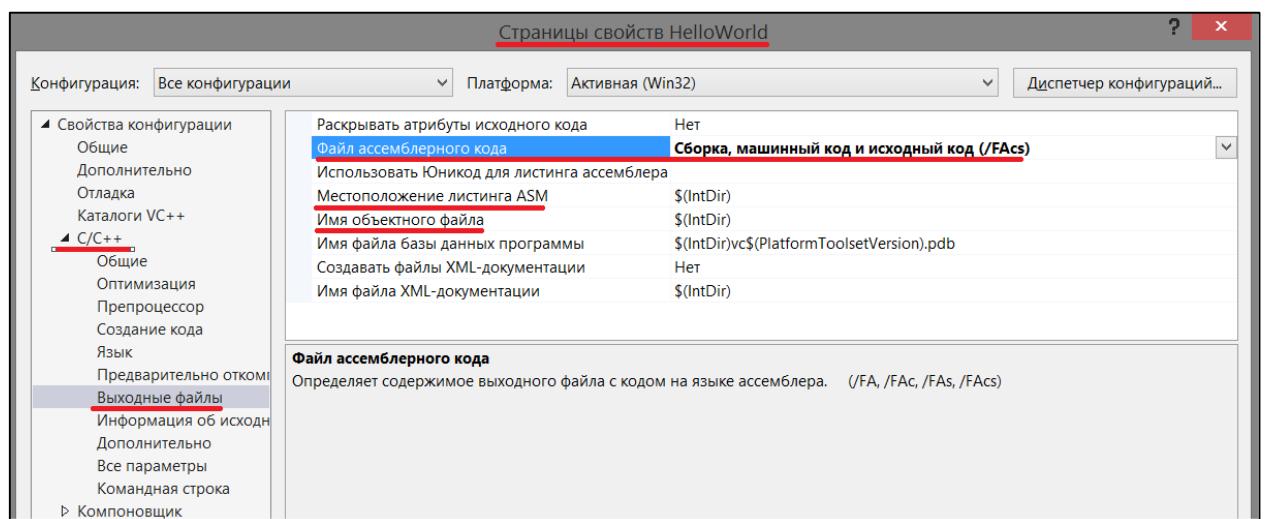
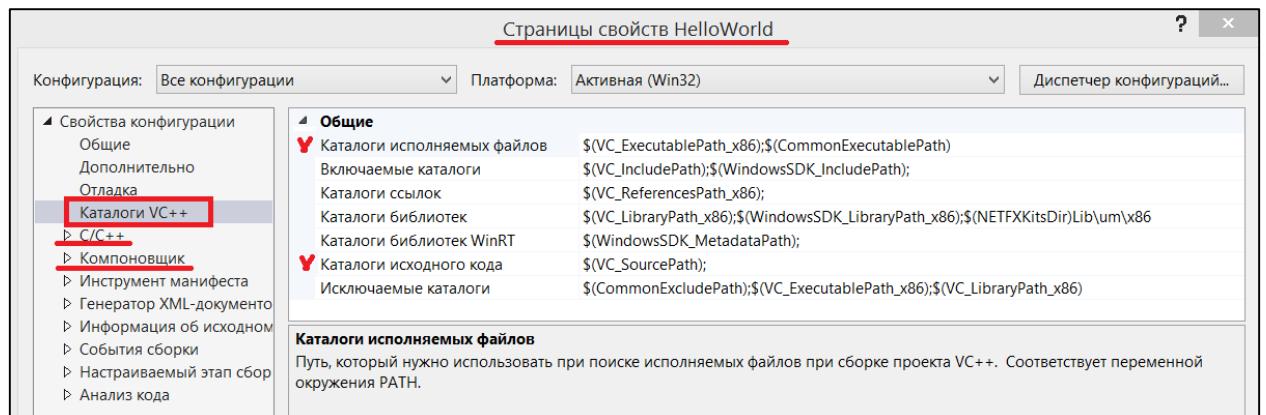
Общие – ключ **Уровень предупреждений** позволяет отключить все предупреждения (/W0), либо ужесточить уровень проверок и считать все предупреждения ошибками (/Wall).

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:

| Папка Debug проекта: | | | |
|---|------------------|--------------------|--------|
| 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.ilk | 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 /FAs)

```
00025 E8 00 00 00 00    call    w__SpecктроДебаггер_JustMyCode@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@_05PDJBBECF@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 /FAs /Fa"Debug\" /Fo"Debug\" /Fd"Deb
/errorReport:prompt Hello.cpp
  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
uuid.lib odbc32.lib odbcpp32.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 обь
сборка успешно завершена.
Предупреждений: 0
Ошибка: 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

Страницы свойств HelloWorld

Конфигурация: Debug Платформа: Активная (Win32) Диспетчер конфигур

Свойства конфигурации

- ▲ Общие
- Дополнительно
- Отладка
- Каталоги VC++
- △ C/C++**
- Общие
- Оптимизация
- Препроцессор
- Создание кода
- Язык
- Предварительно откомпилированные файлы
- Выходные файлы
- Информация об исходном коде
- Дополнительно
- Все параметры
- Командная строка**

Все параметры

```
/JMC /permissive- /GS /analyze- /W3 /Zc:wchar_t /ZI /Gm- /Od /sdl /Fd"Debug\vc142.pdb" /Zc:inline /fp:precise /D "_MBCS" /errorReport:prompt /WX- /Zc:forScope /RTC1 /Gd /Oy- /MDd /std:c++17 /FC /Fa"Debug\" /EHsc /nologo /Fo"Debug\" /FAs /Fp"Debug\HelloWorld.pch" /diagnostics:column
```


Страницы свойств HelloWorld

Конфигурация: Debug Платформа: Активная (Win32) Диспетчер конфигур.

Свойства конфигурации

- ▲ Общие
- Дополнительно
- Отладка
- Каталоги VC++
- △ C/C++**
- △ Компоновщик**
- Общие
- Ввод
- Файл манифеста
- Отладка
- Система
- Оптимизация
- Внедренный IDL
- Метаданные Windows
- Дополнительно
- Все параметры
- Командная строка**

Все параметры

```
/OUT:"D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld\Debug\HelloWorld.exe" /MANIFEST /NXCOMPAT /PDB:"D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld\Debug\HelloWorld.pdb" /DYNAMICBASE "kernel32.lib" "user32.lib" "gdi32.lib" "winspool.lib" "comdlg32.lib" "advapi32.lib" "shell32.lib" "ole32.lib" "oleaut32.lib" "uuid.lib" "odbc32.lib" "odbccp32.lib" /DEBUG:FASTLINK /MACHINE:X86 /INCREMENTAL /PGD:"D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld\Debug\HelloWorld.pgd" /SUBSYSTEM:CONSOLE /MANIFESTUAC:"level='asInvoker' uiAccess='false'" /ManifestFile:"Debug\HelloWorld.exe.intermediate.manifest" /ERRORREPORT:PROMPT /NOLOGO /TLBID:1
```

Командная строка разработчика:

Windows PowerShell

```
=====
** Visual Studio 2019 Developer Command Prompt v16.3.1
** Copyright (c) 2019 Microsoft Corporation
=====
```

D:\Adel\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld> set

ACPath=C:\Program Files (x86)\Lenovo\Access Connections\
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\chimera\AppData\Roaming
CM2014DIR=C:\Program Files (x86)\Common Files\Autodesk Shared\Materials\
CommandPromptType=Native
CommonProgramFiles=C:\Program Files (x86)\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=W520
ComSpec=C:\Windows\system32\cmd.exe
DELCAM_LICENSE_FILE=1700@W520
DevEnvDir=C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE\
ExtensionSdkDir=C:\Program Files (x86)\Microsoft SDKs\Windows Kits\10\Extensions

```
D:\Ade1\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld>cl  
Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.23.28105.4 для x86  
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
```

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

```
D:\Ade1\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:\Ade1\Кафедра\ОПИ+ТРПО\Примеры к лабораторным работам\HelloWorld>cd D:\Ade1\Кафедра\КПО\Примеры к лабораторным работам\Hello
```

```
D:\Ade1\Кафедра\КПО\Примеры к лабораторным работам\Hello>dir
```

Том в устройстве D не имеет метки.
Серийный номер тома: 34DB-9113

Содержимое папки D:\Ade1\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 | 2a676a199a424 байт свободно |

```
D:\Ade1\Кафедра\КПО\Примеры к лабораторным работам\Hello>
```

5. Простой (сокращенный) вариант

```
D:\Ade1\Кафедра\КП0\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:\Ade1\Кафедра\КП0\Примеры к лабораторным работам\Hello>dir
Том в устройстве D не имеет метки.
Серийный номер тома: 34DB-9113

Содержимое папки D:\Ade1\Кафедра\КП0\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 файла       95a063 байт
              2 папок      2a676a101a120 байт свободно
```

Ключи компилятора:

/c – компиляция без компоновки;
/EHsc – модель обработки исключений (перехватываются исключения C++).

```
D:\Ade1\Кафедра\КП0\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:\Ade1\Кафедра\КП0\Примеры к лабораторным работам\Hello>dir
Том в устройстве D не имеет метки.
Серийный номер тома: 34DB-9113

Содержимое папки D:\Ade1\Кафедра\КП0\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 файла       200a530 байт
              2 папок      2a675a904a512 байт свободно
```

Ключи компоновщика:

/out – указывает имя загрузочного файла.

Выполнение приложения из командной строки разработчика:

```
D:\Ade1\Кафедра\КП0\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:\\P\\Lab01\\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]               LPPPROCESS_INFORMATION lpProcessInformation
);

LPCTSTR lpApplicationName,           // имя исполняемого модуля
LPTSTR lpCommandLine,              // командная строка
LPSECURITY_ATTRIBUTES lpProcessAttributes, // SD (дескриптор безопасности)
LPSECURITY_ATTRIBUTES lpThreadAttributes, // SD
BOOL bInheritHandles,             // дескриптор параметра
                                 // наследования
DWORD dwCreationFlags,            // флаги создания
LPVOID lpEnvironment,             // новый блок конфигурации
LPCTSTR lpCurrentDirectory,       // имя текущего каталога
LPSTARTUPINFO lpStartupInfo,      // информация предустановки
LPPPROCESS_INFORMATION lpProcessInformation // 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:

```
#pragma once

#include "targetver.h"

#include <stdio.h>
#include <tchar.h>

#include <stdlib.h>
#include <iostream>

// TODO: Установите здесь ссылки
```

Главная функция:

```
#include "stdafx.h"

int sum(int x, int y);
int sub(int x, int y);
int mul(int x, int y);

int _tmain(int argc, _TCHAR* argv[])
{
    std::cout<<"sum(2, 3) = "<<sum(2, 3)<<std::endl;
    std::cout<<"sub(2, 3) = "<<sub(2, 3)<<std::endl;
    std::cout<<"mul(2, 3) = "<<mul(2, 3)<<std::endl;

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

Определения функций:

```
#include "stdafx.h"
int sum(int x, int y){ return x+y; }
```

The screenshot shows two separate code editor windows. Both windows have a title bar with the project name 'LP_Lab02x' and a file list on the right.

Top Window:

```
#include "stdafx.h"
int sub(int x, int y){ return x-y; }
```

- File list: LP_Lab02x (selected), Внешние зависимости, Заголовочные файлы (stdafx.h, targetver.h), Файлы исходного кода (LP_Lab02x.cpp, LP_Lab02x_mul.cpp, LP_Lab02x_sub.cpp, LP_Lab02x_sum.cpp).

Bottom Window:

```
#include "stdafx.h"
int mul(int x, int y){ return x*y; }
```

- File list: LP_Lab02x (selected), Внешние зависимости, Заголовочные файлы (stdafx.h, targetver.h), Файлы исходного кода (LP_Lab02x.cpp, LP_Lab02x_mul.cpp, LP_Lab02x_sub.cpp).

| 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:

The terminal window shows four separate compilation commands for different source files. Each command is highlighted with a red box.

- D:\Adel\LPLab02>c1 /c /EHsc /Yu"stdafx.h" LP_Lab02.cpp
Optимизирующий компилятор Microsoft (R) C/C++ версии 18.00.21005.1 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
LP_Lab02.cpp
- D:\Adel\LPLab02>c1 /c /EHsc /Yu"stdafx.h" Func_sum.cpp
Optимизирующий компилятор Microsoft (R) C/C++ версии 18.00.21005.1 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Func_sum.cpp
- D:\Adel\LPLab02>c1 /c /EHsc /Yu"stdafx.h" Func_sub.cpp
Optимизирующий компилятор Microsoft (R) C/C++ версии 18.00.21005.1 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Func_sub.cpp
- D:\Adel\LPLab02>c1 /c /EHsc /Yu"stdafx.h" Func_mul.cpp
Optимизирующий компилятор Microsoft (R) C/C++ версии 18.00.21005.1 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Func_mul.cpp

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 |
| 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 |

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 |
| LP_Lab02.exe | 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 <code>#using</code> . |
| /analyze | Включение анализа кода. |
| /arch | Задает 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 | Задает компиляцию без компоновки. |
| /cgthreads | Задает число потоков cl.exe, используемых для оптимизации и создания кода. |
| /clr | Создает 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 | Задает 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 | Задает размер стека. |
| /favor | Создает код, которая оптимизирована для конкретных x64 архитектуры или для специфики микроархитектур в AMD64 и расширенной памяти 64 архитектурах технологии (EM64T). |
| /FA | Создает файл листинга. |
| /Fa | Задает 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 | Задает предобработанное имя выходного файла. |
| /Fm | Создает файл сопоставления. |
| /Fo | Создает объектный файл. |
| /fp | Задает поведение чисел с плавающей запятой. |
| /Fp | Задает имя файла предкомпилированного заголовка. |
| /FR | |
| /Fr | Создает 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 __cdecl (только архитектура x86). |
| /Ge | Не рекомендуется. Включает стековые зонды. |
| /GF | Включает объединение строк. |
| /GH | Вызывает функцию-обработчик _rexit. |
| /Gh | Вызывает функцию-обработчик _penter. |
| /GL | Включает оптимизацию всей программы. |
| /Gm | Включает минимальное перепостроение. |
| /GR | Включает информацию о типах во время выполнения (RTTI). |
| /Gr | Использует соглашение о вызовах __fastcall (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 __vectorcall . (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 входении 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. |
| /Ld | Создает отладочную библиотеку динамической компоновки. |
| /link | Передает 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 pragma-директиву <code>#pragma omp</code> 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>/QIfist</u> | Не рекомендуется. Подавляет использование функции _ftol при необходимости преобразования из типа с плавающей запятой в целочисленный тип (только архитектура x86). |
| <u>/Qimprecise_fwaits</u> | Удаляет команды fwait внутри блоков try . |
| <u>/Qpar (автоматический параллелизатор)</u> | Включает автоматическую параллелизацию циклов, которые помечены с помощью директивы <u>#pragma loop()</u> . |
| <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 С. |
| <u>/TC</u> | Указывает, что все исходные файлы, С. |
| <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 vtordisp-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 volatile. |

| Параметр | Цель |
|---|--|
| /w | Отключает все предупреждения. |
| /W0, /W1, /W2, /W3, /W4 | Задает уровень предупреждения для вывода. |
| /w1, /w2, /w3, /w4 | Задает 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 односторочные 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 | Создает 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. |
| /Z7 | Приводит к возникновению ошибки совместимости с С 7.0 отладочную информацию. |
| /Za | Отключает расширения языка. |
| /Zc | Задает 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 | Создает 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 | Задает выравнивание каждой секции. |
| /ALLOWBIND | Указывает на то, что библиотека DLL не может быть привязана. |
| /ALLOWISOLATION | Задает поведение нахождения файлов манифеста. |
| /APPCONTAINER | Определяет, должно ли приложение выполняться в среде процесса контейнера приложений. |
| /ASSEMBLYDEBUG | Добавляет атрибут DebuggableAttribute в управляемый образ. |
| /ASSEMBLYLINKRESOURCE | Создает ссылку на управляемый ресурс. |
| /ASSEMBLYMODULE | Указывает на то, что в сборку должен быть импортирован модуль MSIL. |
| /ASSEMBLYRESOURCE | Внедряет файл управляемых ресурсов в сборку. |
| /BASE | Задает базовый адрес для программы. |
| /CGTHREADS | Задает число потоков cl.exe, используемых для оптимизации и создания кода, если задано создание кода во время компоновки. |
| /CLRIMAGETYPE | Задает 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> | Задает 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> | Задает начальный адрес. |
| <u>/errorReport</u> | Передает 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> | Создает 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>/FUNCTIONADMIN</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). <u>/GENPROFILE</u> и <u>/FASTGENPROFILE</u> используют разные параметры по умолчанию. |
| <u>/GUARD</u> | Включает защиту потока управления. |
| <u>/HEAP</u> | Задает 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> | Задает контейнер ключей для подписи сборки. |
| <u>/KEYFILE</u> | Задает 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 среди библиотеки. |
| <u>/LTCG</u> | Задает 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> | Задает 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> | Задает 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>Задает 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> | Задает 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> | Задает 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> | Задает 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> | Задает имя файла для выходного файла метаданных среды выполнения Windows (winmd), создаваемого параметром компоновщика <u>/WINMD</u> . |
| <u>/WINMDKEYFILE</u> | Задает 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. |