БГТУ, ФИТ, ПОИТ, 2 семестр, Языки программирования

Характеристика курса

1. Дисциплина «Языки программирования».
   Лектор: ***Наркевич Аделина Сергеевна***, ст. преподаватель, кафедры программной инженерии (а.408, к.1).
   email: narkevich.adelina@gmail.com

2. Всего 176 часов, из них лекций 88 часов, лабораторных 88 часов:
   1) семестр II: всего 108 часов, из них 54 часа лекций, 54 часа лабораторных работ, ***экзамен***.
   2) семестр III: всего 68 часов, 34 часа лекций, 34 часа лабораторных работ, ***курсовой проект***, ***зачет***.
3. Инструментарий: Visual Studio 2012 и выше; C++; MASM.
4. Курсовой проект: разработка транслятора (спецификация языка, программная реализация транслятора).
   На выполнение курсового проекта отводится 11 недель.
   Защита: первая половина декабря.
   Объем программного кода примерно 2000 - 3000 строк.
5. Контрольные работы.
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

13. Литература:

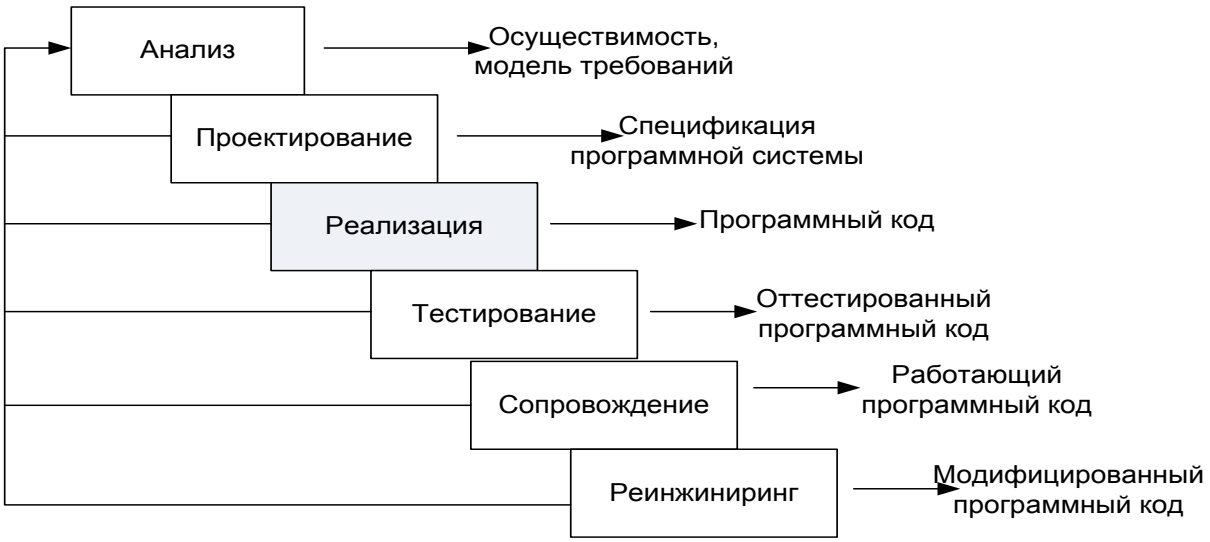
№	Наименование	Кол. экз. в библ.
1	**Ахо А. Компиляторы: принципы, технологии и инструменты / А. Ахо, Р. Сети, Дж. Ульман. – М.: Вильямс, 2003. – 768с.**	1
2	Молчанов А.Ю. Системное программное обеспечение. – СПб.: Питер, 2010. – 400с.	
3	Гагарина Л. Г. Введение в теорию алгоритмических языков и компиляторов / 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.	

Лекция 01
БГТУ, ФИТ, ПОИТ, 2 семестр, Языки программирования

Системы программирования

План лекции:
- современные подходы к разработке языков программирования (скорость разработки, поддержка всего жизненного цикла программы);
- основные определения;
- примеры (1-я и 2-я лабораторные работы).

1. **Классический жизненный цикл** разработки программного обеспечения.



Кроме каскадной модели применяются следующие модели жизненного цикла:

Поэтапная модель с промежуточным контролем

Спиральная модель – эволюционная стратегия.

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

3. **Состав системы программирования**: трансляторы, компоновщики, отладчики, профилировщики, программные библиотеки, редакторы кода, системы поддержки версий и пр.

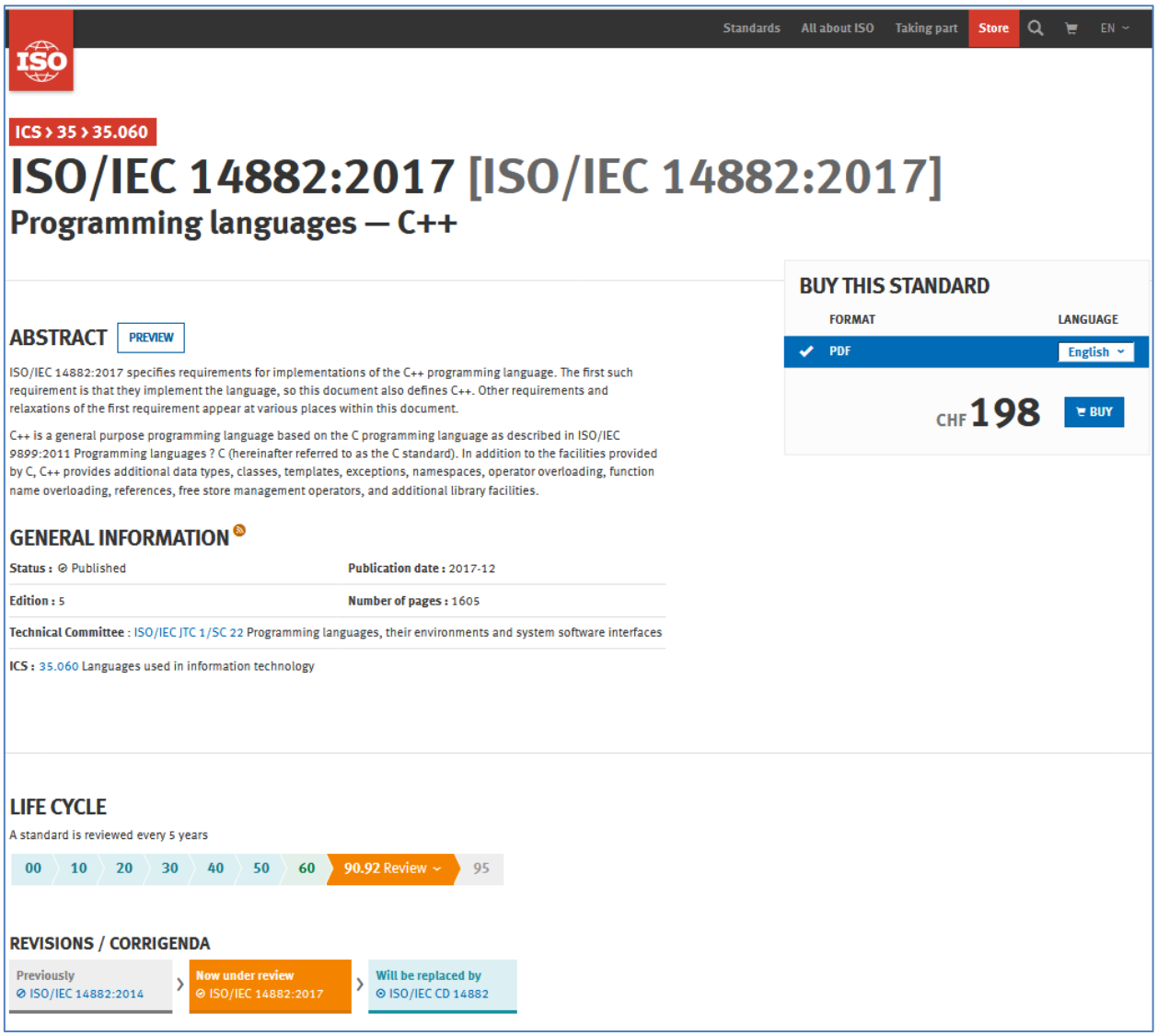
4. **Интегрированная среда разработки (IDE, Integrated development environment)**: редакторы кода, транслятор, компоновщик, отладчик, система поддержки версий. Примеры: Visual Studio, NetBeans, Eclipse, Embarcadero Delphi и пр.

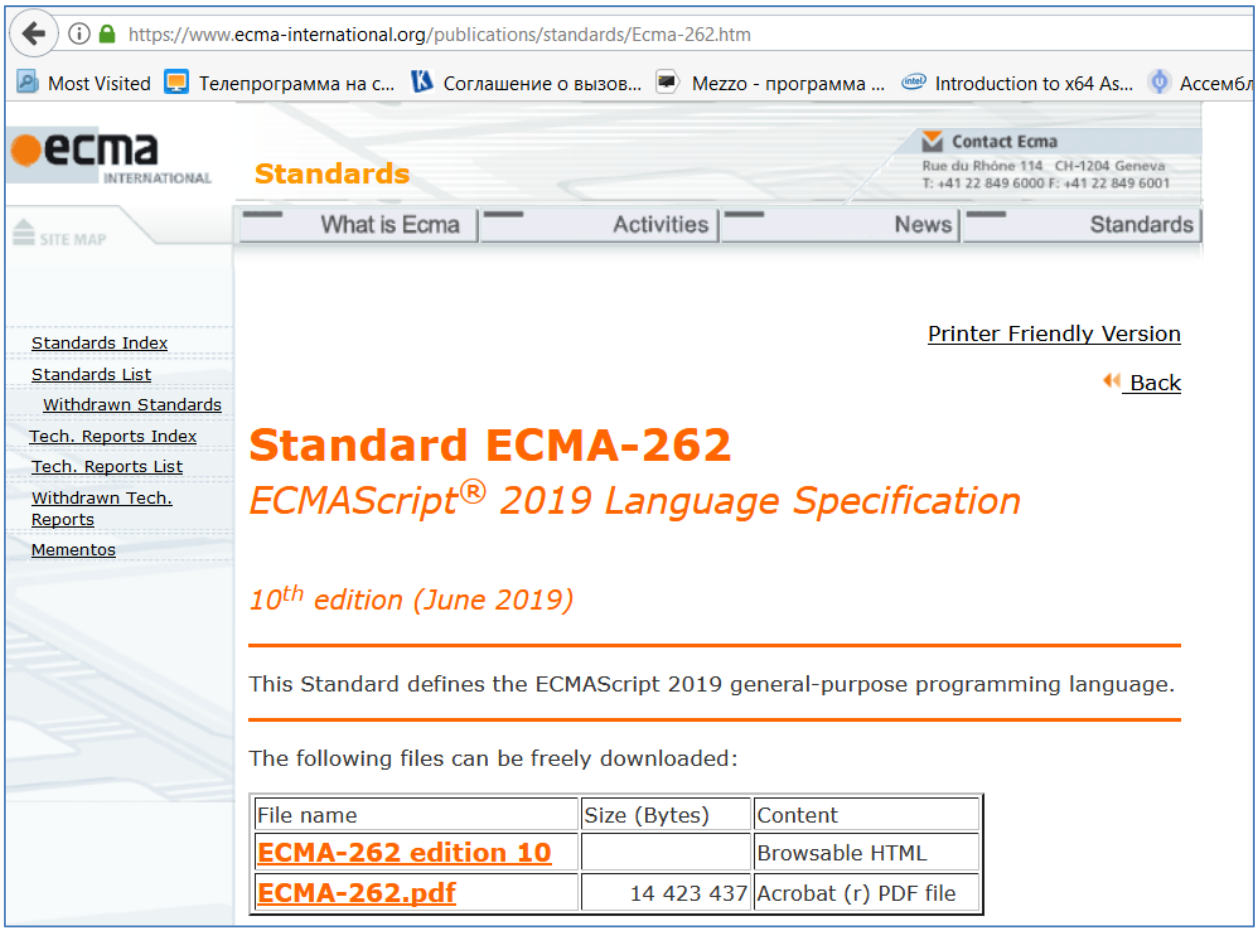
   Интегрированная среда разработки (integrated development environment - IDE) - набор инструментов для разработки и отладки программ, имеющий общую интерактивную графическую оболочку, поддерживающую выполнение всех основных функций жизненного цикла разработки программы.

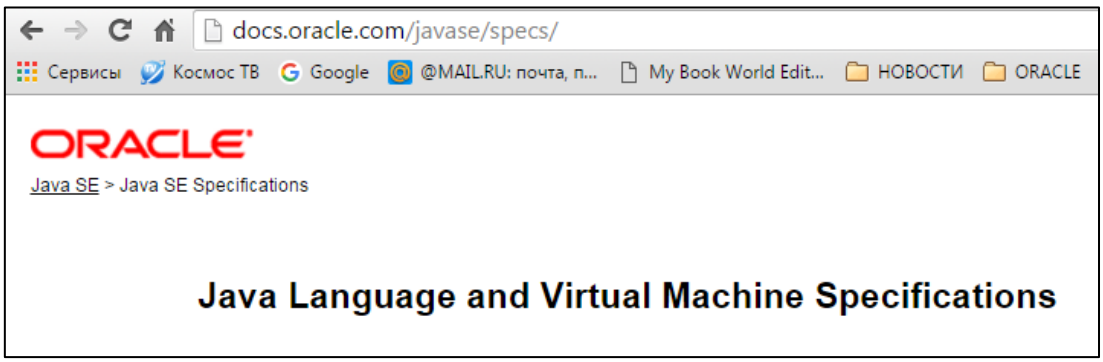
5. **Программный продукт:** программа, работающая без авторского присутствия. Программный продукт исполняется, тестируется, конфигурируется без присутствия автора и сопровождается документацией.

6. **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 программирования: Visual C++ 2017 версия 15.3 – это реализация стандарта C++17 или ISO/IEC 14882:2017.


Standards
All about ISO
Taking part
Store
EN
ICS › 35 › 35.060
ISO/IEC 14882:2017 [ISO/IEC 14882:2017]
Programming languages — C++
BUY THIS STANDARD
FORMAT
LANGUAGE
PDF
English
CHF 198
BUY
ABSTRACT
PREVIEW
ISO/IEC 14882:2017 specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, so this document also defines C++. Other requirements and relaxations of the first requirement appear at various places within this document.
C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:2011 Programming languages ? C (hereinafter referred to as the C standard). In addition to the facilities provided by C, C++ provides additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities.
GENERAL INFORMATION
Status : Published
Publication date : 2017-12
Edition : 5
Number of pages : 1605
Technical Committee : ISO/IEC JTC 1/SC 22 Programming languages, their environments and system software interfaces
ICS : 35.060 Languages used in information technology
LIFE CYCLE
A standard is reviewed every 5 years
00
10
20
30
40
50
60
90.92 Review
95
REVISIONS / CORRIGENDA
Previously
ISO/IEC 14882:2014
Now under review
ISO/IEC 14882:2017
Will be replaced by
ISO/IEC CD 14882


https://www.ecma-international.org/publications/standards/Ecma-262.htm
Most Visited
Телепрограмма на с...
Соглашение о вызов...
Mezzo - программа ...
Introduction to x64 As...
ecma INTERNATIONAL
Standards
Contact Ecma
Rue du Rhône 114 CH-1204 Geneva
T: +41 22 849 6000 F: +41 22 849 6001
SITE MAP
What is Ecma
Activities
News
Standards
Standards Index
Standards List
Withdrawn Standards
Tech. Reports Index
Tech. Reports List
Withdrawn Tech. Reports
Mementos
Printer Friendly Version
Back
Standard ECMA-262
ECMAScript® 2019 Language Specification
10th edition (June 2019)
This Standard defines the ECMAScript 2019 general-purpose programming language.
The following files can be freely downloaded:
File name
Size (Bytes)
Content
ECMA-262 edition 10
Browsable HTML
ECMA-262.pdf
14 423 437
Acrobat (r) PDF file


docs.oracle.com/javase/specs/
Сервисы
Космос ТВ
Google
@MAIL.RU: почта, п...
My Book World Edit...
НОВОСТИ
ORACLE
ORACLE
Java SE > Java SE Specifications
Java Language and Virtual Machine Specifications

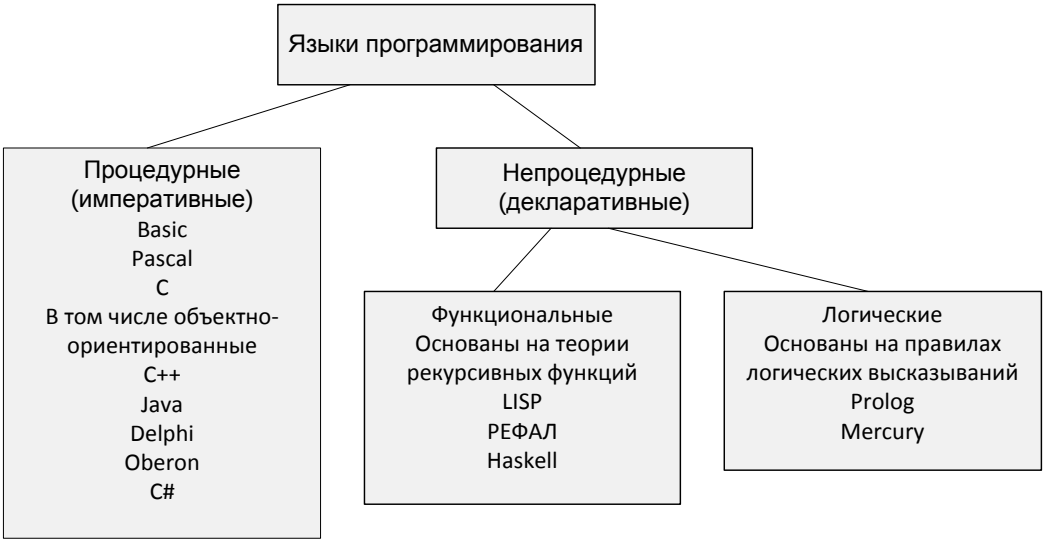
7. **Языки программирования**: процедурно-ориентированные, объектно-ориентированные, декларативные, операторные, функциональные, скриптовые, проблемно-ориентированные, машинно-зависимые (ассемблеры) и т.п.

Основные языки программирования.

1957-1959	**Fortran, LISP, COBOL**	**Авторы**	**Назначение**	**Использовался**
	Старейшие языки программирования. Высокоуровневые, созданы для научных, математических и бизнес вычислений.	Джон Бекус Джон Маккарти Грейс Хоппер («бабушка Кобола»)	Разработка ПО для научных и инженерных вычислений, для обработки списков, бизнеса.	В NASA, кредитных картах и банкоматах
1970	**Pascal**	**Автор**	**Назначение**	**Использовался**
	Высокоуровневый для обучения структурному программированию и структурированию данных.	Никлаус Вирт	Обучение программированию	Skype
1972	**C**	**Автор**	**Назначение**	**Использовался**
	Основан на языке «B». Низкоуровневый, общего назначения. Его синтаксис стал основой для C#, Java, Perl, PHP, Python	Деннис Ритчи	Кроссплатформенное программирование, системное программирование, программирование для UNIX, разработка игр	UNIX (первые веб-серверы и веб-клиенты)
1983	**C++**	**Автор**	**Назначение**	**Использовался**
	Первоначальное название «Си с классами» («++» – оператор инкремента в C). Среднеуровневый, объектно-ориентированный.	Бьёрн Страуструп	Коммерческая разработка приложений, встроенного ПО, клиент-серверных приложений, видеоигр	Adobe, Google, Chrome, Mozilla, Firefox, IE
1983	**Objective-C**	**Автор**	**Назначение**	**Использовался**
	Объектно-ориентированное расширение C. Высокоуровневый, общего назначения	Брэд Кокс и Том Лав	Программирование в Apple	Apple OS X и iOS
1987	**Perl**	**Автор**	**Назначение**	**Использовался**
	Высокоуровневый, общего назначения. Для обработки отчетов в UNIX	Ларри Уолл	Разработка общих шлюзовых интерфейсов, приложений для баз данных, систем администрирования, интернет-программирования, визуальное программирования.	IMDb, Amazon, Priceline, Ticketmaster

1991	Python	Автор	Назначение	Использовался
	Высокоуровневый, общего назначения. Создан для поддержки различных стилей программирования.	Гвидо ван Россум.	Веб-приложения, разработка ПО, защита информации.	Google, Yahoo, Spotify
1993	**Ruby**	**Автор**	**Назначение**	**Использовался**
	Высокоуровневый, общего назначения. Язык с простым синтаксисом, влияние на который оказали Perl, LISP,Smalltolk.	Юкихиро Мацумото	Разработка веб-приложений	Twitter, Hulu, Groupon
1995	**Java**	**Автор**	**Назначение**	**Использовался**
	Высокоуровневый, общего назначения. Был создан для интерактивного ТВ-проекта. Кроссплатформенный.	Джеймс Гослиг	Веб-программирование, разработка веб-приложений и ПО, графического интерфейса пользователя. Разработка нативных и встраеваемых приложений	В системе и приложениях Angroid
1995	**PHP**	**Автор**	**Назначение**	**Использовался**
	Общего назначения, с открытым исходным кодом. Для создания динамических веб-страниц.	Расмус Лердорф	Создание и поддержка динамических веб-страниц, разработка на стороне веб-сервера.	Facebook, Вконтакте, Википедии, Digg, WorldPress, Joomla
1995	**JavaScript**	**Автор**	**Назначение**	**Использовался**
	Высокоуровневый язык. Создан для расширения функциональности веб-страниц.	Брендан Эйх	Динамическая веб-разработка, использование в браузерах, PDF-документах,	Gmail, Adobe, Photoshop, Mozilla, Firefox

Стили программирования



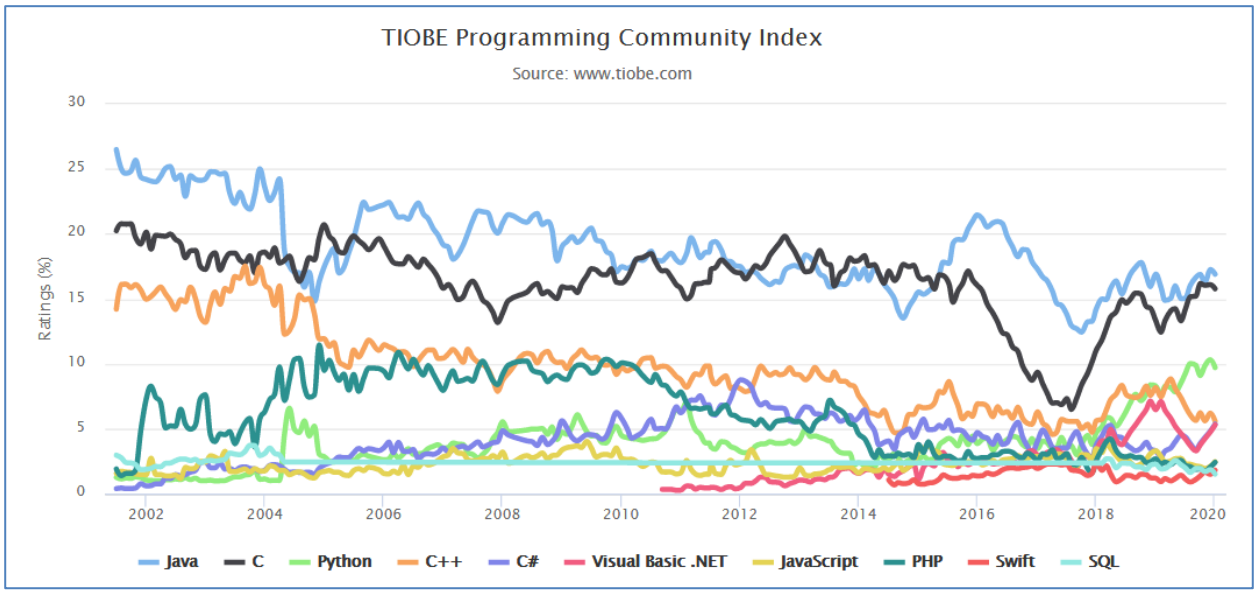


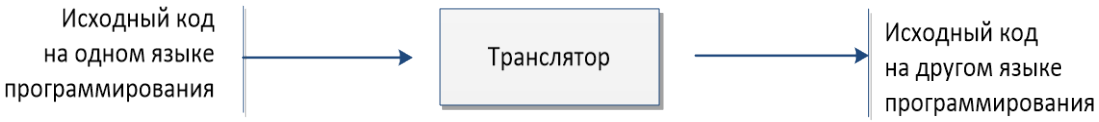
Иллюстрация: TIOBE, январь 2020 года.

8. **Исходный код (исходная программа)**: текст программы, написанный на языке программирования.

   Программа на исходном языке (исходный код) готовится с помощью текстовых редакторов и в виде текстового файла или раздела библиотеки поступает на вход транслятора.

9. **Текстовый редактор:** компонента системы программирования (или IDE) - программа, позволяющая подготовить исходный код программы.

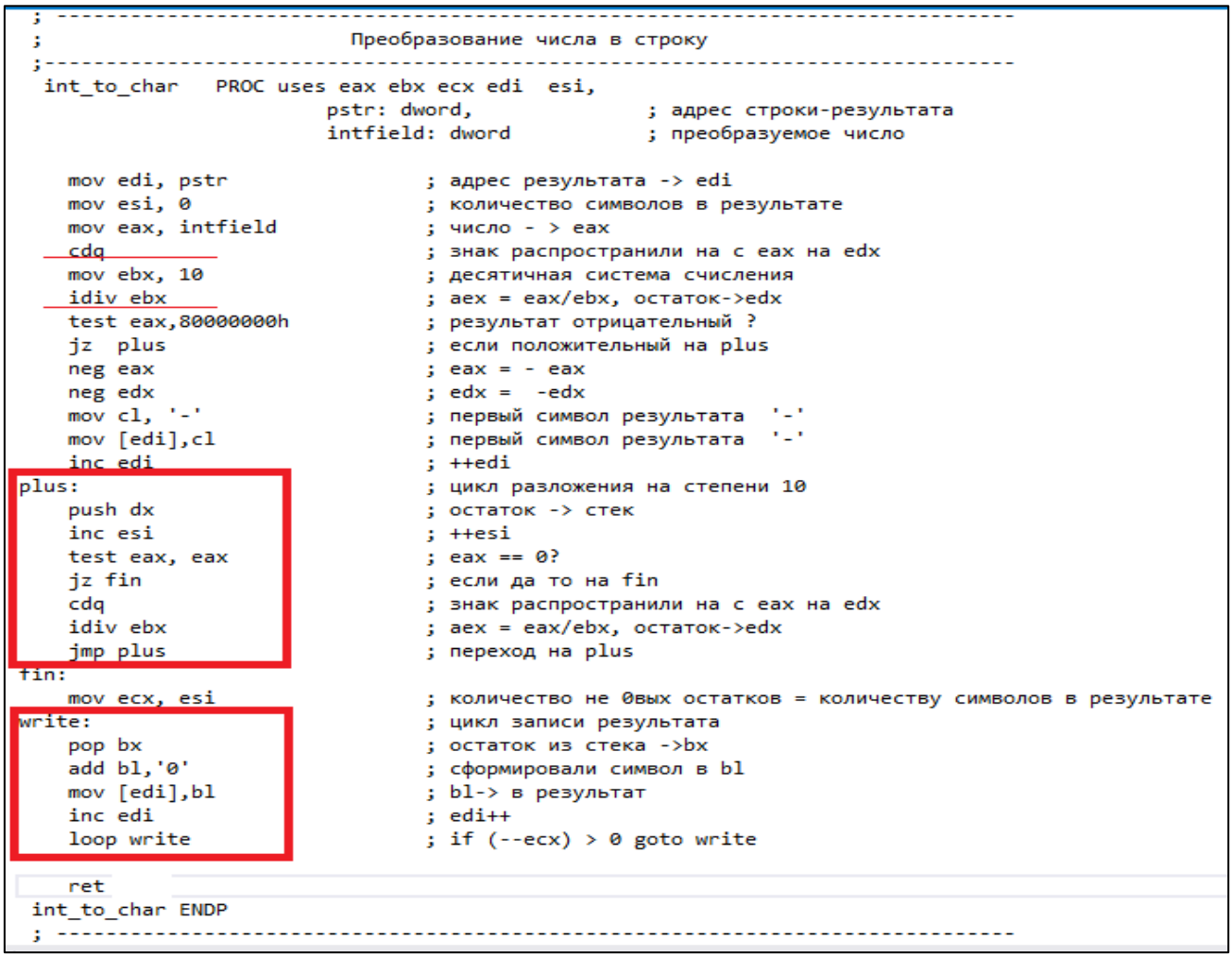
10. **Транслятор:** программа, преобразующая исходный код на одном языке программирования в исходный код на другом языке.



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

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

12. **Ассемблер:** транслятор с исходного кода на языке ассемблера в программу на машинном языке (язык, который может интерпретироваться процессором).



```
; ---------------------------------------------------------------------------
;                          Преобразование числа в строку
;---------------------------------------------------------------------------
 int_to_char   PROC uses eax ebx ecx edi  esi,
                        pstr: dword,              ; адрес строки-результата
                        intfield: dword           ; преобразуемое число

    mov edi, pstr               ; адрес результата -> edi
    mov esi, 0                  ; количество символов в результате
    mov eax, intfield           ; число - > eax
    cdq                         ; знак распространили на с eax на edx
    mov ebx, 10                 ; десятичная система счисления
    idiv ebx                    ; aex = eax/ebx, остаток->edx
    test eax,80000000h          ; результат отрицательный ?
    jz  plus                    ; если положительный на plus
    neg eax                     ; eax = - eax
    neg edx                     ; edx =  -edx
    mov cl, '-'                 ; первый символ результата  '-'
    mov [edi],cl                ; первый символ результата  '-'
    inc edi                     ; ++edi
plus:                           ; цикл разложения на степени 10
    push dx                     ; остаток -> стек
    inc esi                     ; ++esi
    test eax, eax               ; eax == 0?
    jz fin                      ; если да то на fin
    cdq                         ; знак распространили на с eax на edx
    idiv ebx                    ; aex = eax/ebx, остаток->edx
    jmp plus                    ; переход на plus
fin:
    mov ecx, esi                ; количество не 0вых остатков = количеству символов в результате
write:                          ; цикл записи результата
    pop bx                      ; остаток из стека ->bx
    add bl,'0'                  ; сформировали символ в bl
    mov [edi],bl                ; bl-> в результат
    inc edi                     ; edi++
    loop write                  ; if (--ecx) > 0 goto write

    ret
 int_to_char ENDP
 ; ---------------------------------------------------------------------------
```

13. **Объектный код:** результат работы транслятора.
    Один файл объектного кода – **объектный модуль.**

14. **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).

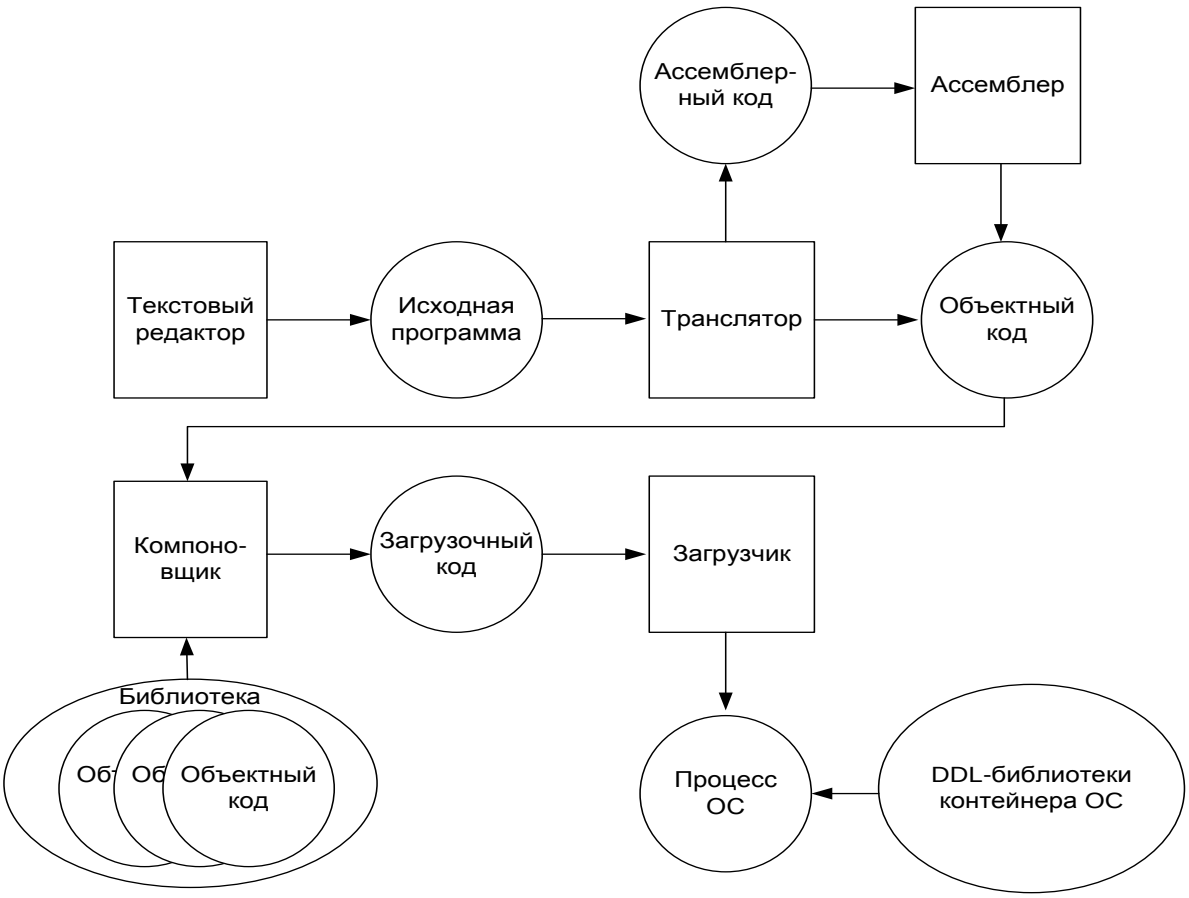
15. **Загрузочный код:** результат работы компоновщика. Один файл загрузочного кода – **загрузочный модуль.**

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

17. **Отладчик (debugger):** компонента системы программирования (или IDE) - программа, позволяющая контролировать ход выполнения программы (приостанавливать, выполнять пошагово), просматривать и изменять области памяти.

18. **Профилировщик:** компонента системы программирования (или IDE) - программа, позволяющая оптимизировать код программы (устранять утечки памяти, оптимизировать циклы, анализировать производительность).
    Профилировщики, определяют, какой процент времени выполняется та или иная часть программы. Это позволяет выявить наиболее интенсивно используемые фрагменты программы и оптимизировать их или на исходном языке, или, например, переписав эти фрагменты на Ассемблер.

19. **Структура классической системы программирования**

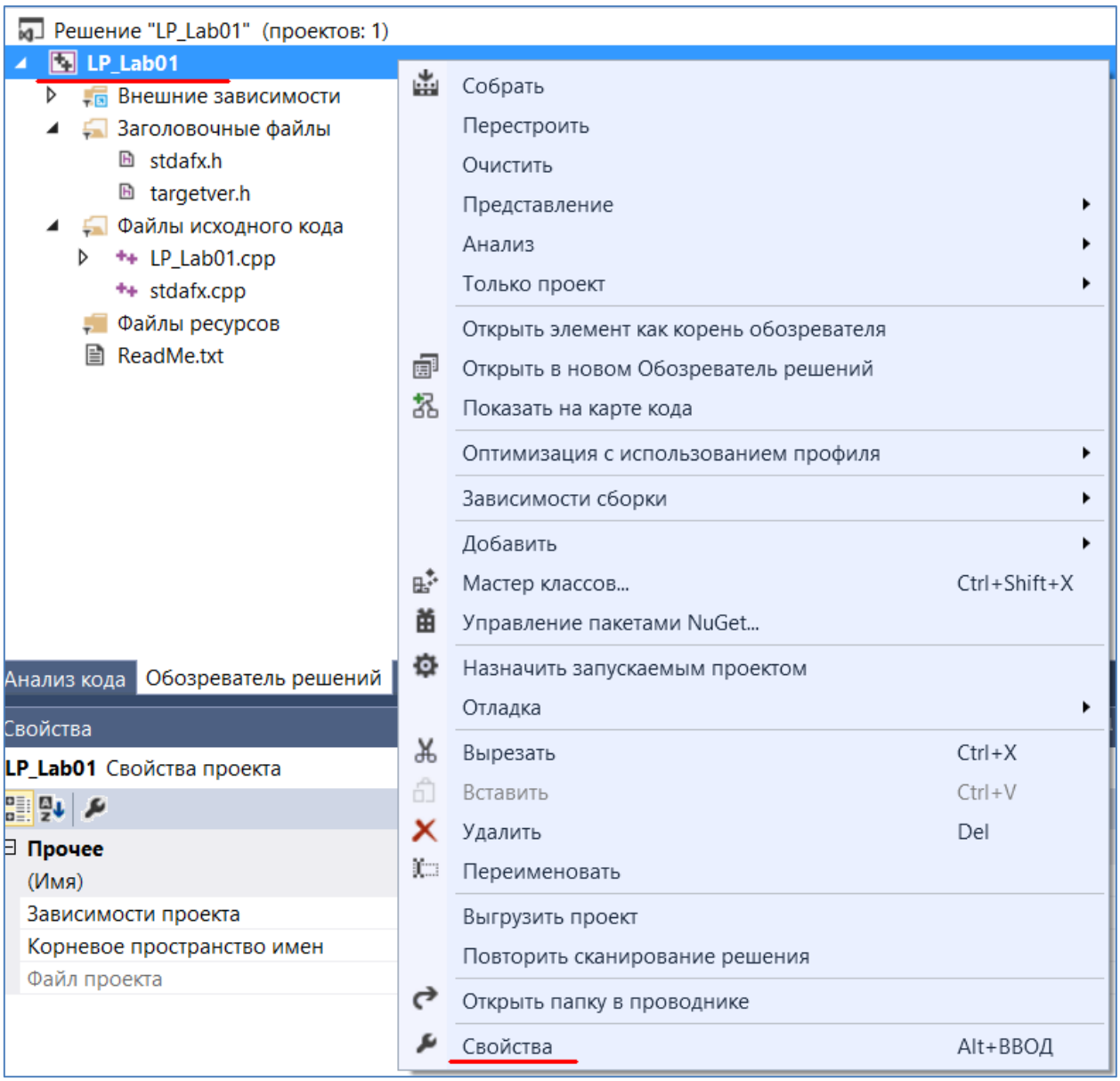


20. **Пример** (материал для лабораторных работ 1 и 2)

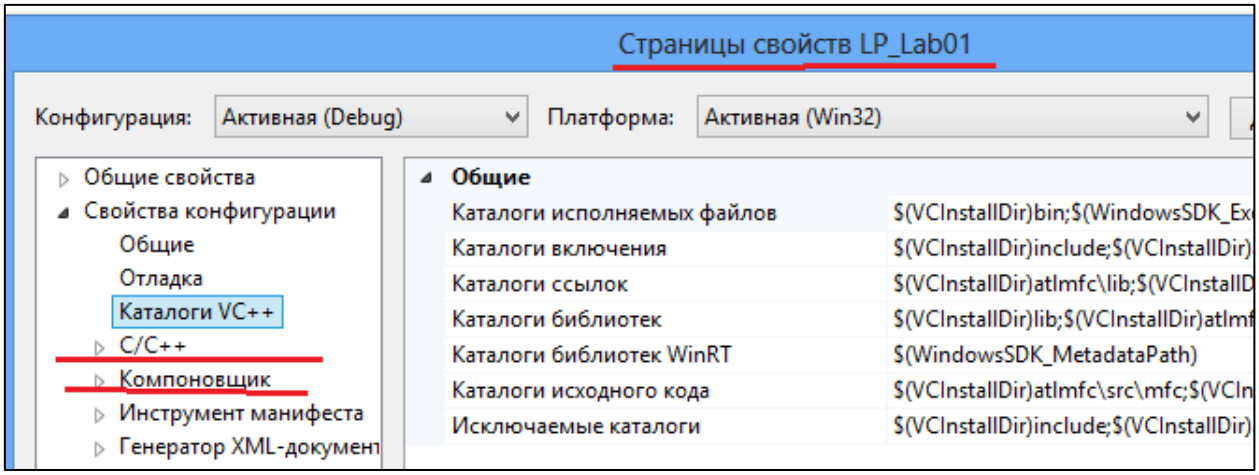
Пример программы «HelloWorld», которая выводит сообщение, используя стандартную библиотеку, заголовок этой библиотеки подключается директивой препроцессора #include <iostream>. Программа завершается с кодом возврата 0.

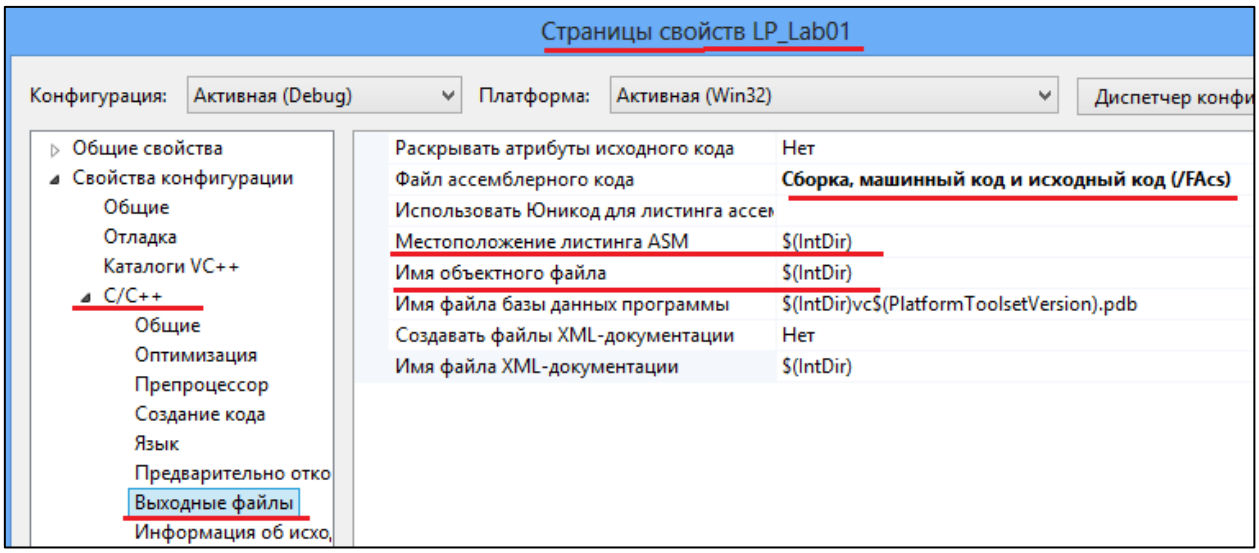
```
#include "stdafx.h"
#include <stdlib.h>
#include <iostream>


int _tmain(int argc, _TCHAR* argv[])
{
    std::cout << "Hello World!!!\n";
    system("pause");
    return 0;
}
```

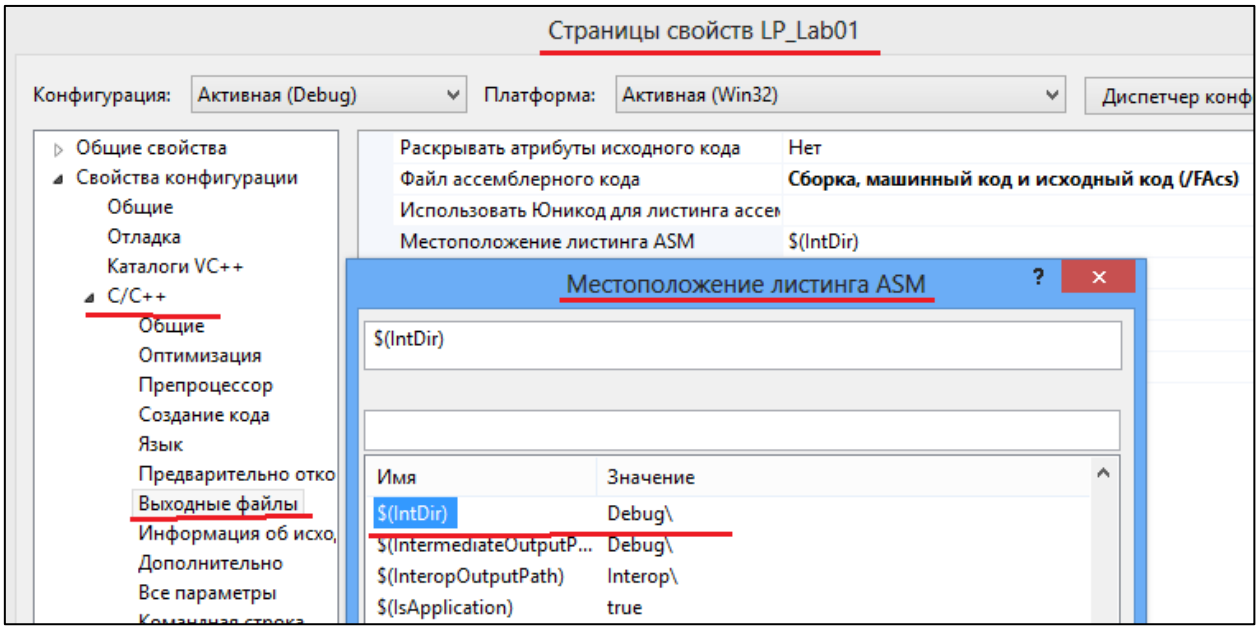
Обозреватель решений. Глобальный контейнер (верхний уровень)

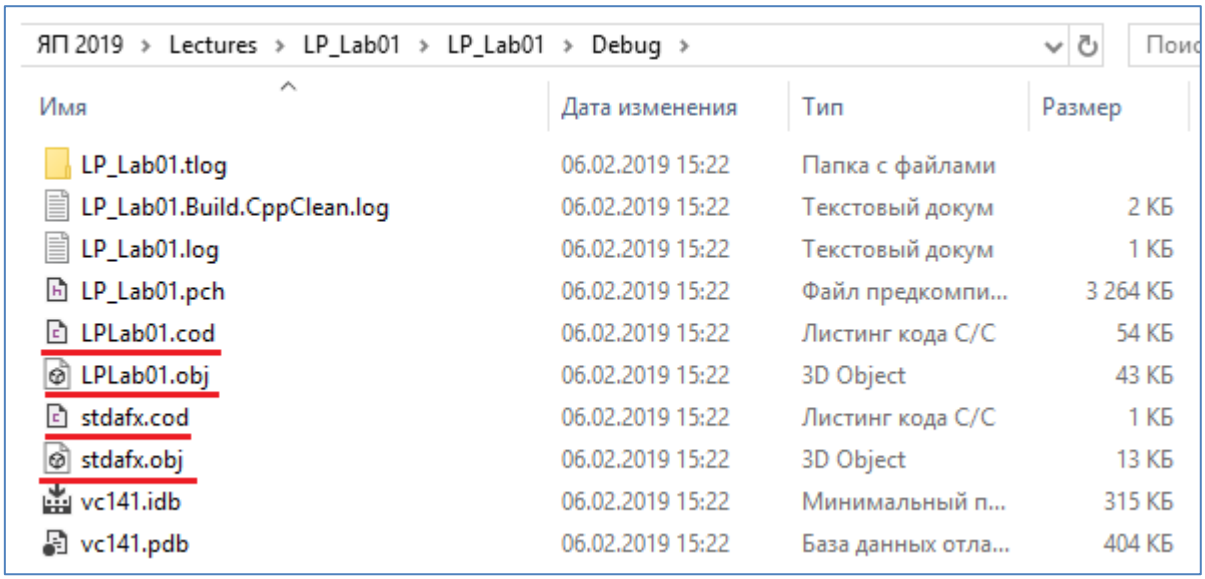


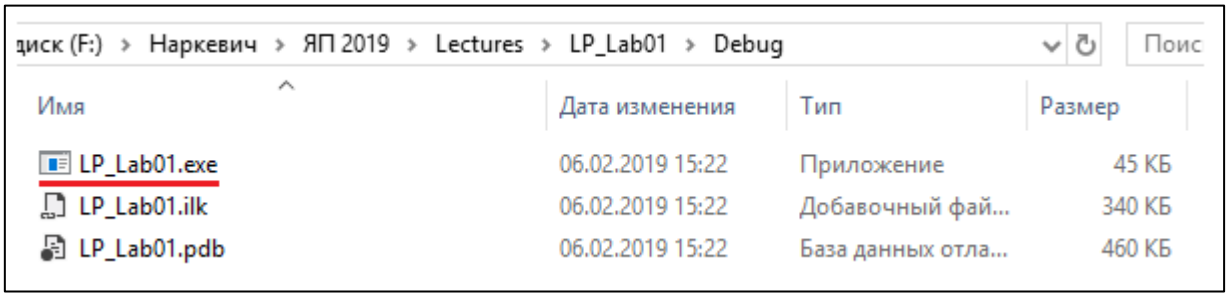
Страницы свойств проекта. Разделы свойств.

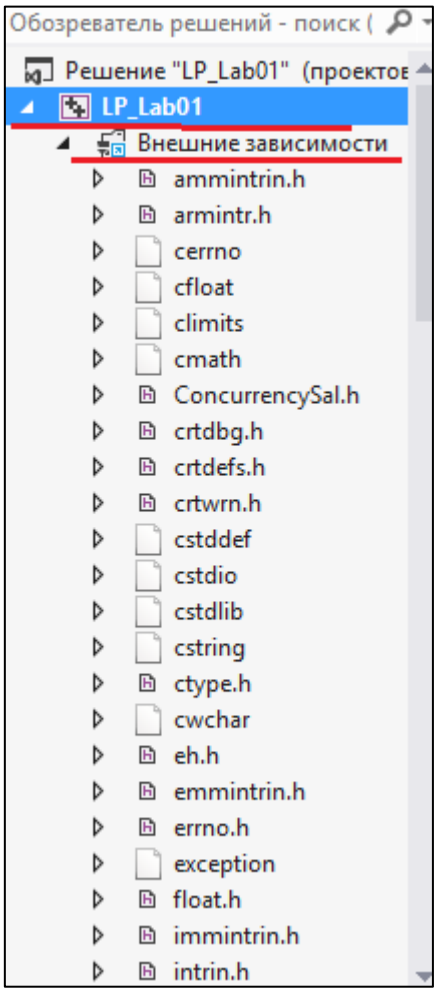


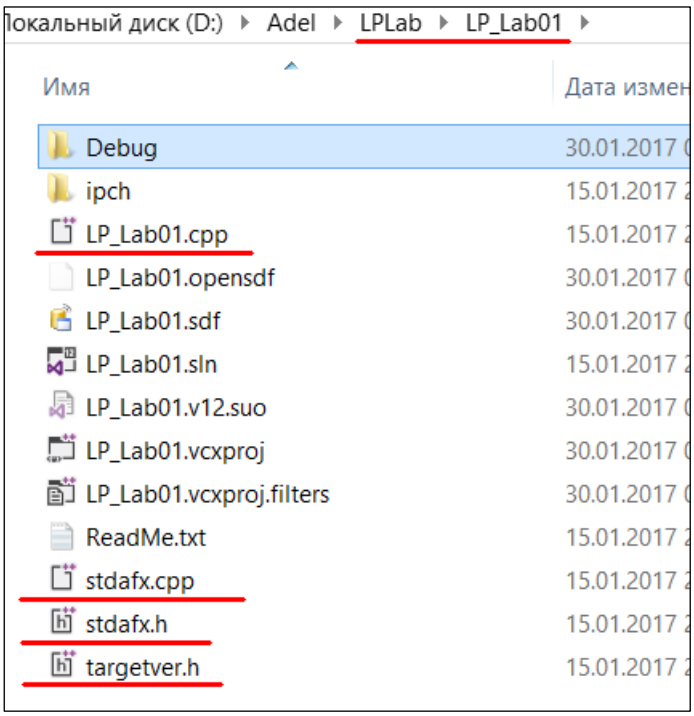





ЯП 2019 › Lectures › LP_Lab01 › LP_Lab01 › Debug ›
Имя
Дата изменения
Тип
Размер
LP_Lab01.tlog 06.02.2019 15:22 Папка с файлами
LP_Lab01.Build.CppClean.log 06.02.2019 15:22 Текстовый докум 2 КБ
LP_Lab01.log 06.02.2019 15:22 Текстовый докум 1 КБ
LP_Lab01.pch 06.02.2019 15:22 Файл предкомпи... 3 264 КБ
LPLab01.cod 06.02.2019 15:22 Листинг кода C/C 54 КБ
LPLab01.obj 06.02.2019 15:22 3D Object 43 КБ
stdafx.cod 06.02.2019 15:22 Листинг кода C/C 1 КБ
stdafx.obj 06.02.2019 15:22 3D Object 13 КБ
vc141.idb 06.02.2019 15:22 Минимальный п... 315 КБ
vc141.pdb 06.02.2019 15:22 База данных отла... 404 КБ


диск (F:) › Наркевич › ЯП 2019 › Lectures › LP_Lab01 › Debug
Имя
Дата изменения
Тип
Размер
LP_Lab01.exe 06.02.2019 15:22 Приложение 45 КБ
LP_Lab01.ilk 06.02.2019 15:22 Добавочный фай... 340 КБ
LP_Lab01.pdb 06.02.2019 15:22 База данных отла... 460 КБ

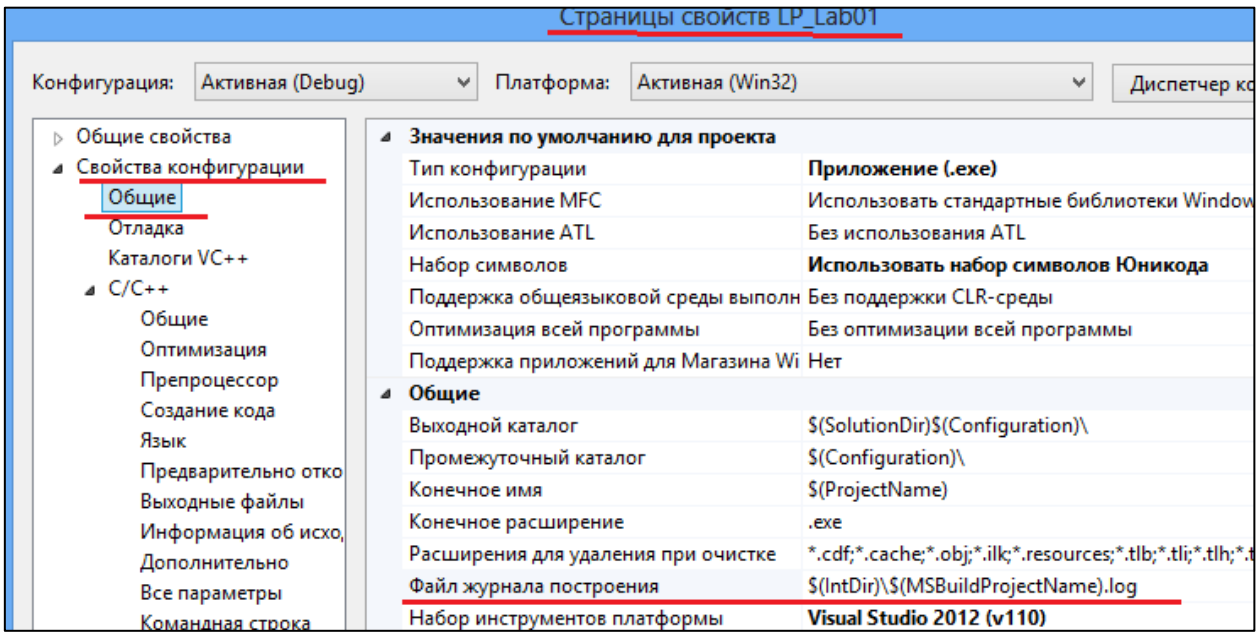

Обозреватель решений - поиск (
Решение "LP_Lab01" (проектов
LP_Lab01
Внешние зависимости
ammintrin.h
armintr.h
cerrno
cfloat
climits
cmath
ConcurrencySal.h
crtdbg.h
crtdefs.h
crtwrn.h
cstddef
cstdio
cstdlib
cstring
ctype.h
cwchar
eh.h
emmintrin.h
errno.h
exception
float.h
immintrin.h
intrin.h

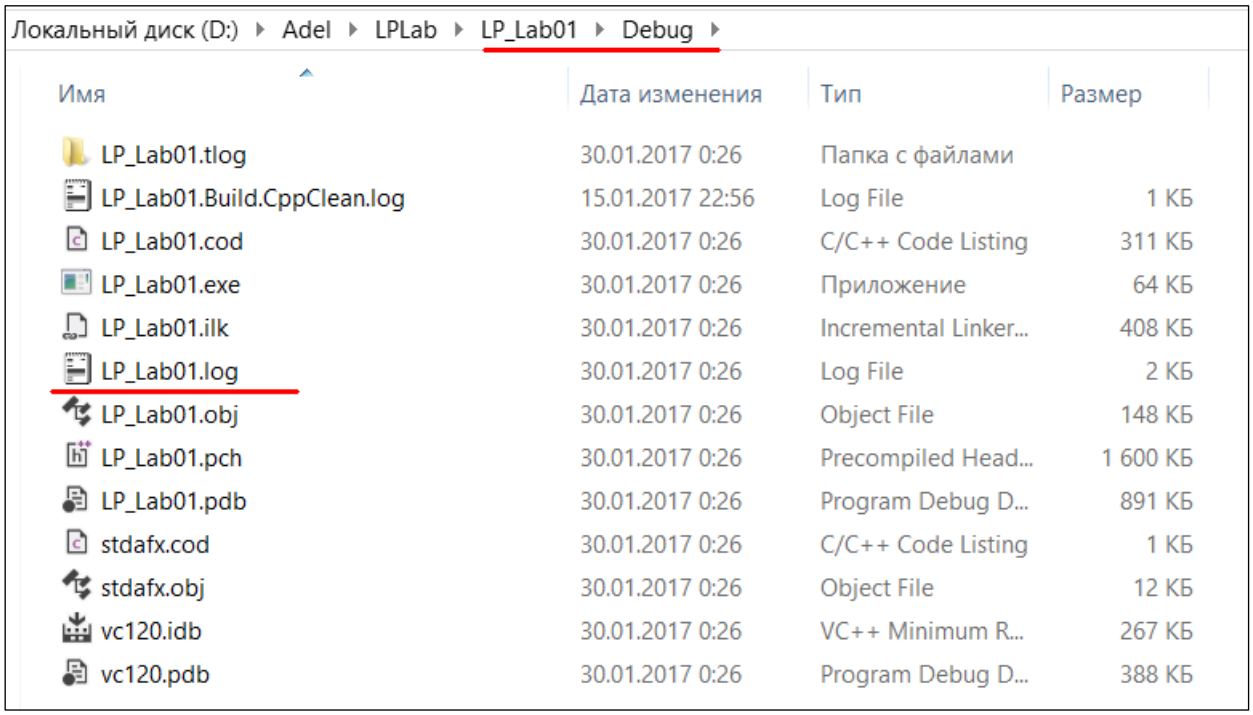

Локальный диск (D:) ▸ Adel ▸ LPLab ▸ LP_Lab01 ▸
Имя
Дата измен
Debug 30.01.2017
ipch 15.01.2017
LP_Lab01.cpp 15.01.2017
LP_Lab01.opensdf 30.01.2017
LP_Lab01.sdf 30.01.2017
LP_Lab01.sln 15.01.2017
LP_Lab01.v12.suo 30.01.2017
LP_Lab01.vcxproj 30.01.2017
LP_Lab01.vcxproj.filters 30.01.2017
ReadMe.txt 15.01.2017
stdafx.cpp 15.01.2017
stdafx.h 15.01.2017
targetver.h 15.01.2017

Листинг ASM кода (/FAcs)

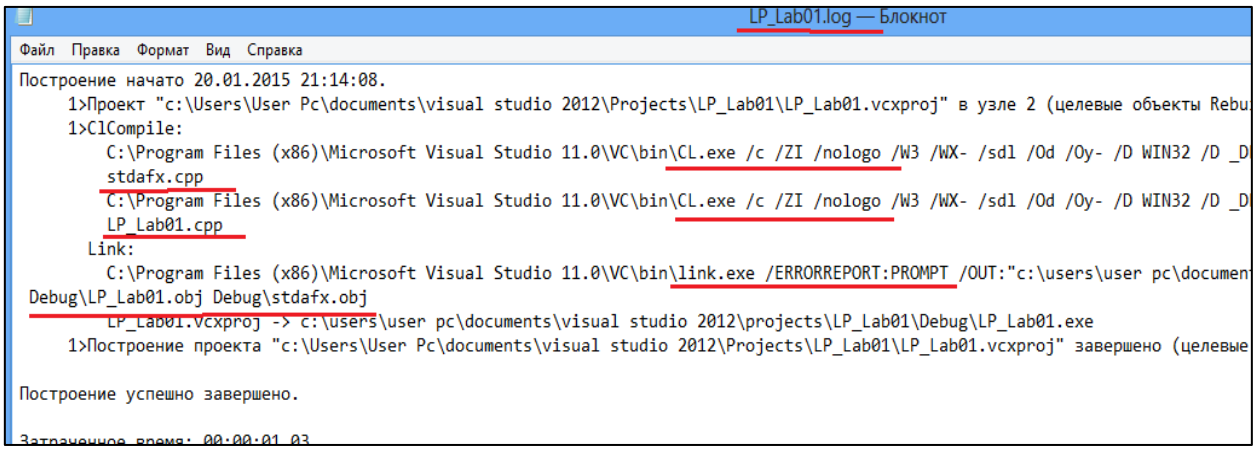
LP_Lab01.cod | stdafx.cod | LP_Lab01.cpp

```
_argc$ = 8                      ; size = 4
_argv$ = 12                     ; size = 4
_wmain  PROC                        ; COMDAT

; 9    : {

  00000 55       push    ebp
  00001 8b ec        mov     ebp, esp
  00003 81 ec c0 00 00
    00       sub     esp, 192       ; 000000c0H
  00009 53       push    ebx
  0000a 56       push    esi
  0000b 57       push    edi
  0000c 8d bd 40 ff ff
    ff       lea     edi, DWORD PTR [ebp-192]
  00012 b9 30 00 00 00   mov     ecx, 48            ; 0000003
  00017 b8 cc cc cc cc   mov     eax, -858993460        ; ccc
  0001c f3 ab        rep stosd

; 10   :    std::cout<<"Hello World!!!";

  0001e 68 00 00 00 00   push    OFFSET ??_C@_0P@MKFFDJMN@Hel
  00023 a1 00 00 00 00   mov     eax, DWORD PTR __imp_?cout@s
  00028 50       push    eax
  00029 e8 00 00 00 00   call    ??$?6U?$char_traits@D@std@@@
  0002e 83 c4 08     add     esp, 8

; 11   :    system("pause");

  00031 8b f4        mov     esi, esp
  00033 68 00 00 00 00   push    OFFSET ??_C@_05PDJBBECF@paus
  00038 ff 15 00 00 00
    00       call    DWORD PTR __imp__system
  0003e 83 c4 04     add     esp, 4
  00041 3b f4        cmp     esi, esp
```





Файл журнала построения



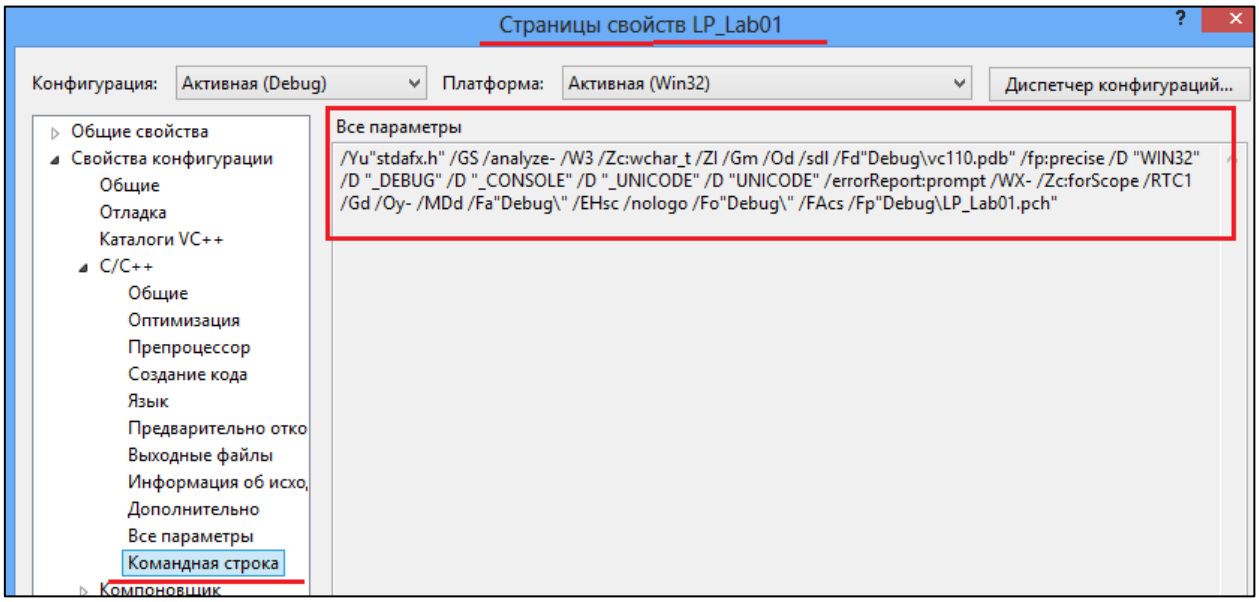
В VS2017 и выше.

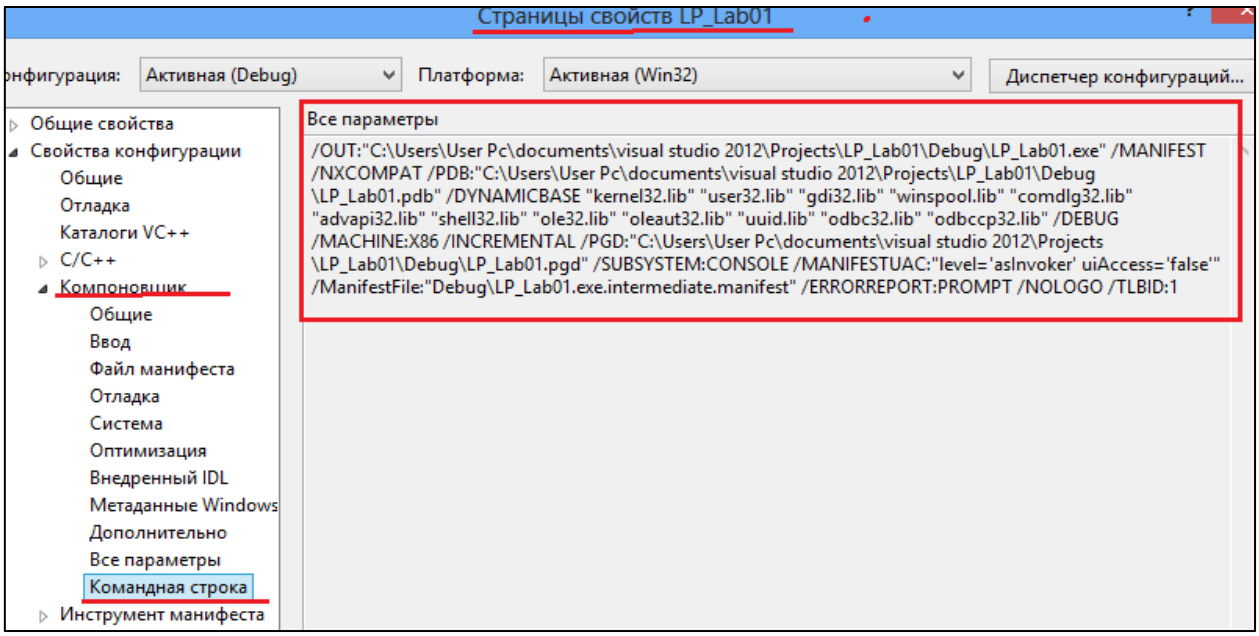
для изменения объема сведений, включаемых в журнал сборки необходимо:

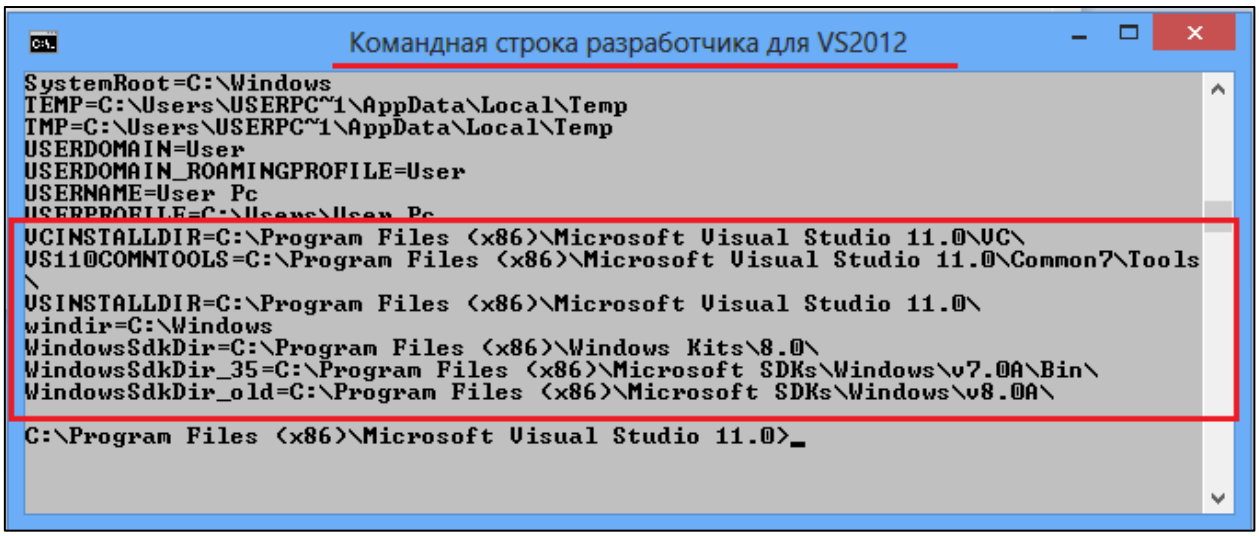
- меню ***Сервис*** (***Средства***) -> ***Параметры***
- на странице ***Проекты*** и решения выбрать ***Сборка и запуск***
- в списке Степень подробности сообщений при построении проекта MSBuild выбрать ***Обычный*** и нажать ОК

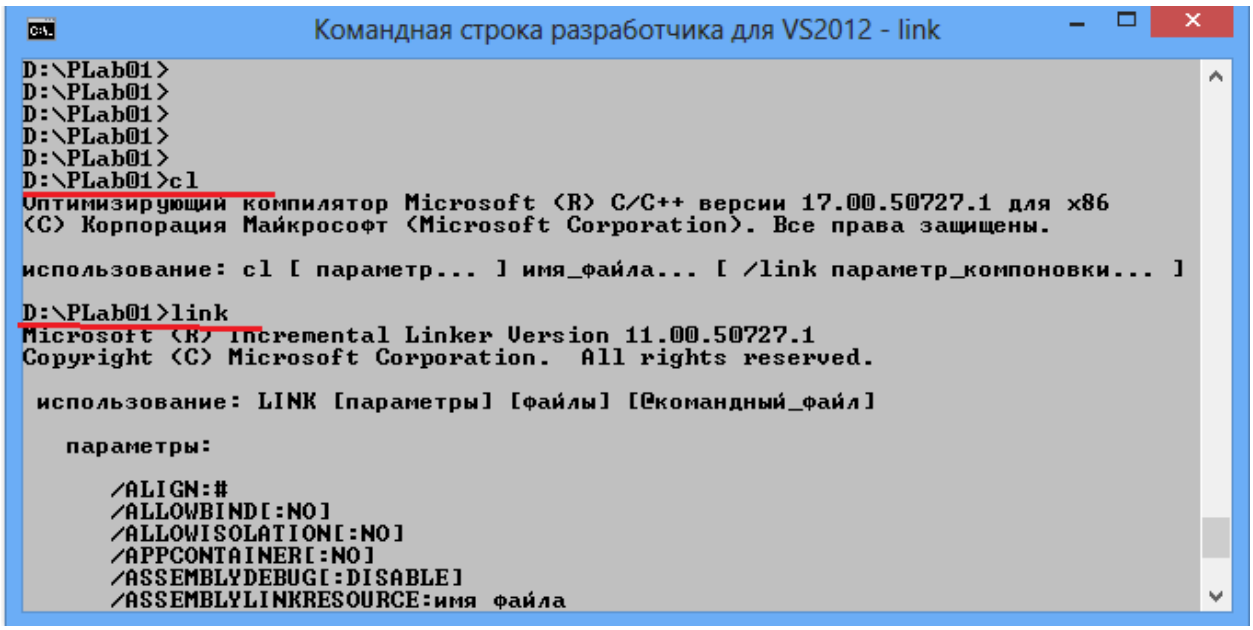
Обычный – отображает сводку о сборке, ошибки, предупреждения и сообщения с высокой степенью важности, а также основные шаги сборки.

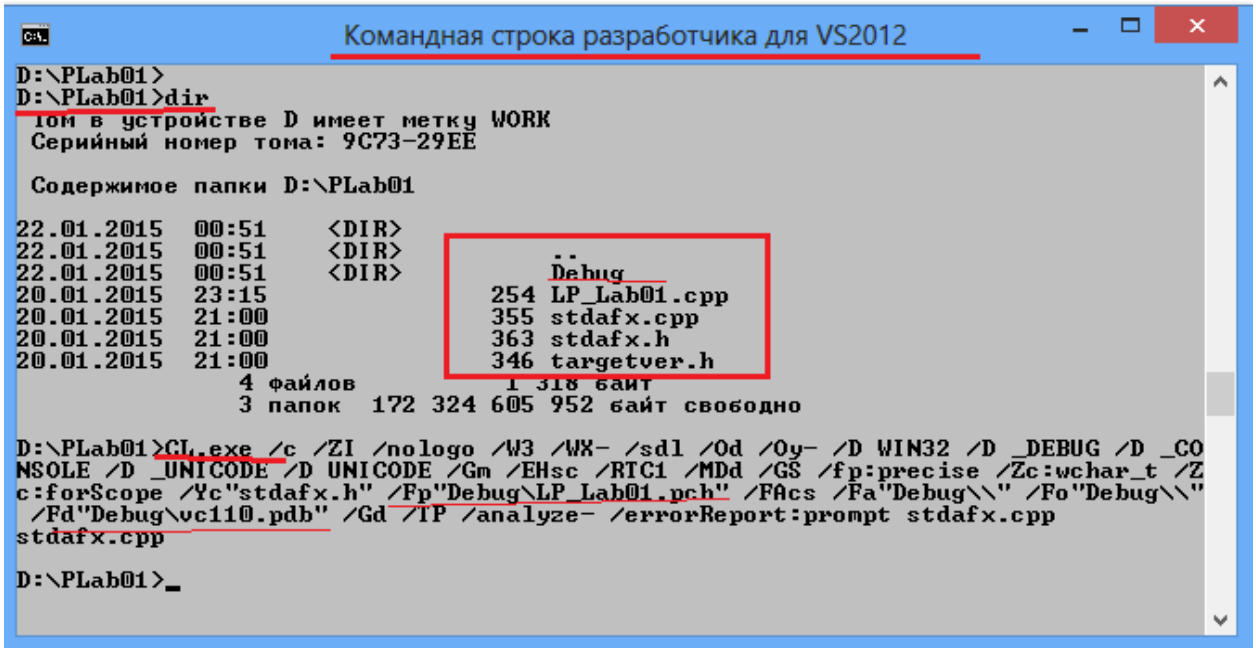
21. Компиляция в командной строке

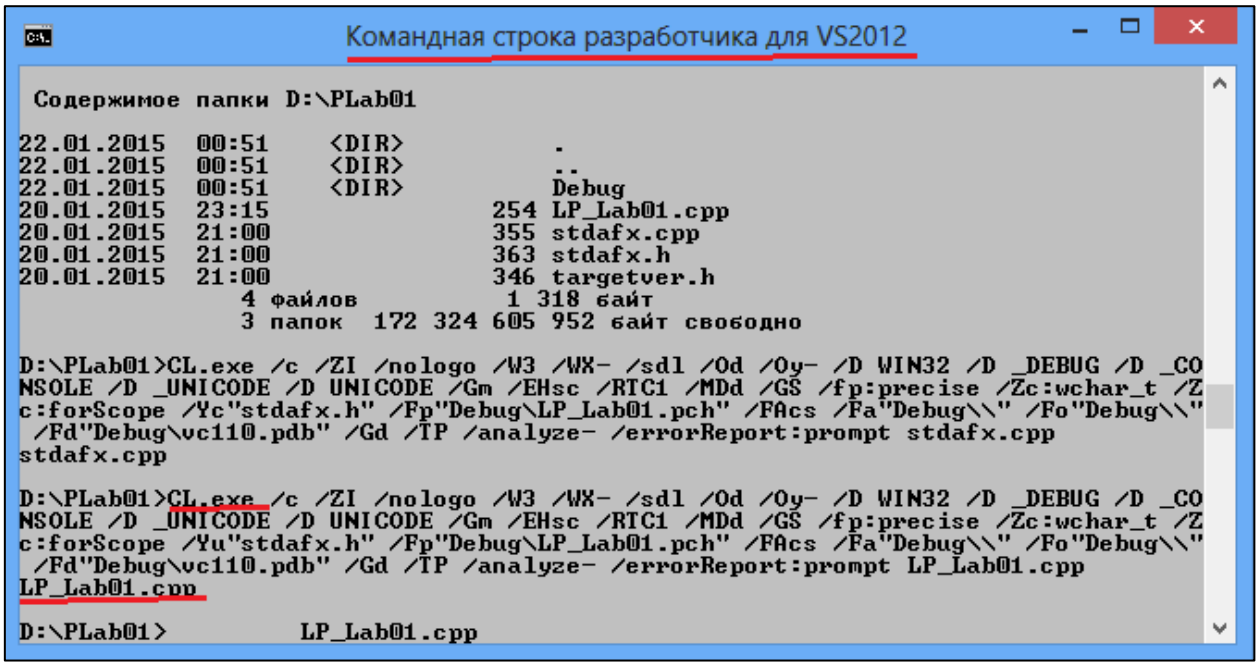


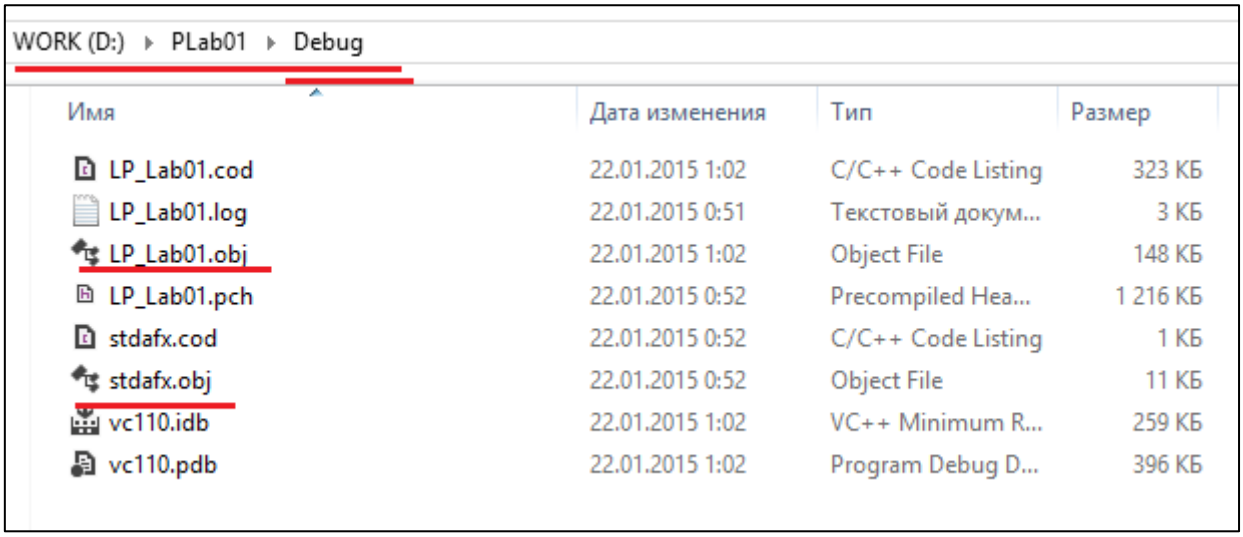


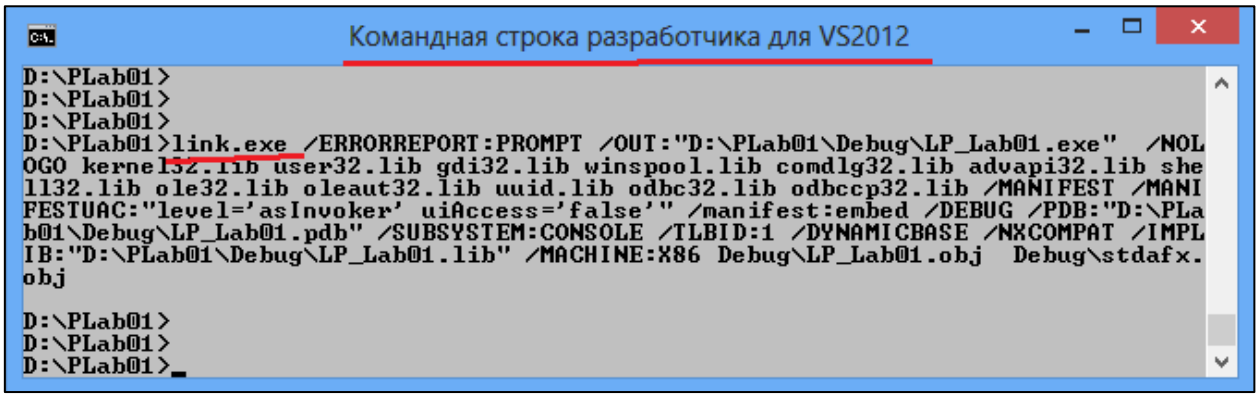


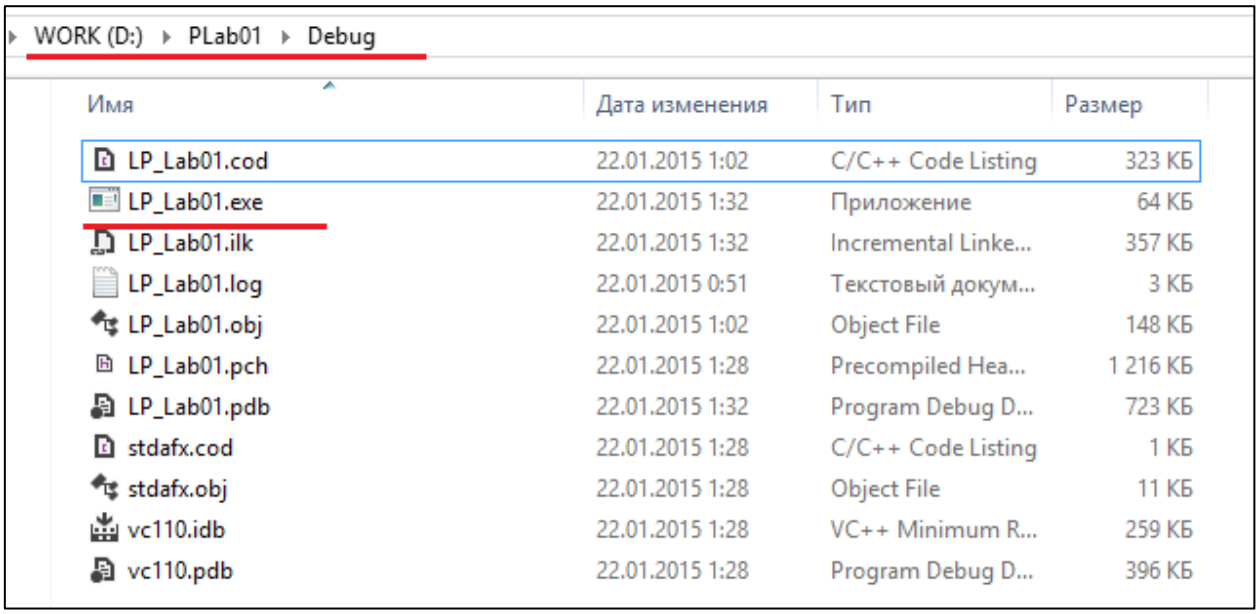

Командная строка разработчика для VS2012 - link
D:\PLab01>
D:\PLab01>
D:\PLab01>
D:\PLab01>
D:\PLab01>
D:\PLab01>cl
Оптимизирующий компилятор Microsoft (R) C/C++ версии 17.00.50727.1 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
использование: cl [параметр...] имя_файла... [/link параметр_компоновки...]
D:\PLab01>link
Microsoft (R) Incremental Linker Version 11.00.50727.1
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


Командная строка разработчика для VS2012
D:\PLab01>
D:\PLab01>dir
Том в устройстве D имеет метку WORK
Серийный номер тома: 9C73-29EE
Содержимое папки D:\PLab01
22.01.2015 00:51 <DIR> ..
22.01.2015 00:51 <DIR> Debug
22.01.2015 00:51 <DIR>
20.01.2015 23:15 254 LP_Lab01.cpp
20.01.2015 21:00 355 stdafx.cpp
20.01.2015 21:00 363 stdafx.h
20.01.2015 21:00 346 targetver.h
4 файлов 1 318 байт
3 папок 172 324 605 952 байт свободно
D:\PLab01>CL.exe /c /ZI /nologo /W3 /WX- /sdl /Od /Oy- /D WIN32 /D _DEBUG /D _CONSOLE /D _UNICODE /D UNICODE /Gm /EHsc /RTC1 /MDd /GS /fp:precise /Zc:wchar_t /Zc:forScope /Yc"stdafx.h" /Fp"Debug\LP_Lab01.pch" /FAcs /Fa"Debug\\" /Fo"Debug\\" /Fd"Debug\vc110.pdb" /Gd /TP /analyze- /errorReport:prompt stdafx.cpp
stdafx.cpp
D:\PLab01>_

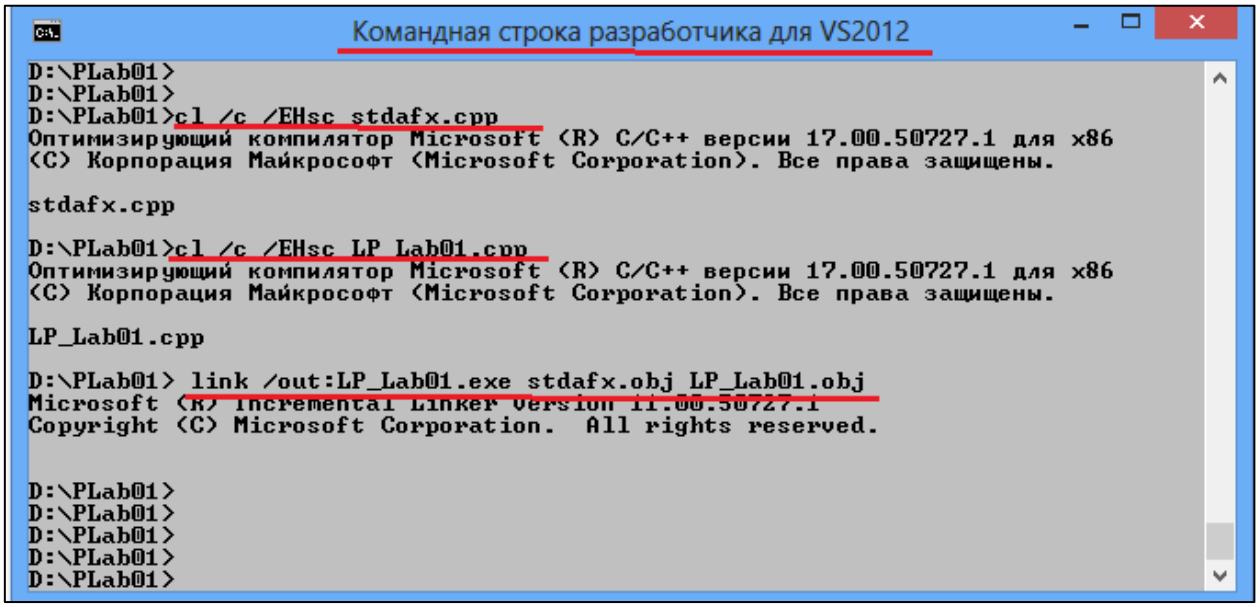

Командная строка разработчика для VS2012
Содержимое папки D:\PLab01
22.01.2015 00:51 <DIR> .
22.01.2015 00:51 <DIR> ..
22.01.2015 00:51 <DIR> Debug
20.01.2015 23:15 254 LP_Lab01.cpp
20.01.2015 21:00 355 stdafx.cpp
20.01.2015 21:00 363 stdafx.h
20.01.2015 21:00 346 targetver.h
4 файлов 1 318 байт
3 папок 172 324 605 952 байт свободно
D:\PLab01>CL.exe /c /ZI /nologo /W3 /WX- /sdl /Od /Oy- /D WIN32 /D _DEBUG /D _CONSOLE /D _UNICODE /D UNICODE /Gm /EHsc /RTC1 /MDd /GS /fp:precise /Zc:wchar_t /Zc:forScope /Yc"stdafx.h" /Fp"Debug\LP_Lab01.pch" /FAcs /Fa"Debug\\" /Fo"Debug\\" /Fd"Debug\vc110.pdb" /Gd /TP /analyze- /errorReport:prompt stdafx.cpp
stdafx.cpp
D:\PLab01>CL.exe /c /ZI /nologo /W3 /WX- /sdl /Od /Oy- /D WIN32 /D _DEBUG /D _CONSOLE /D _UNICODE /D UNICODE /Gm /EHsc /RTC1 /MDd /GS /fp:precise /Zc:wchar_t /Zc:forScope /Yu"stdafx.h" /Fp"Debug\LP_Lab01.pch" /FAcs /Fa"Debug\\" /Fo"Debug\\" /Fd"Debug\vc110.pdb" /Gd /TP /analyze- /errorReport:prompt LP_Lab01.cpp
LP_Lab01.cpp
D:\PLab01> LP_Lab01.cpp

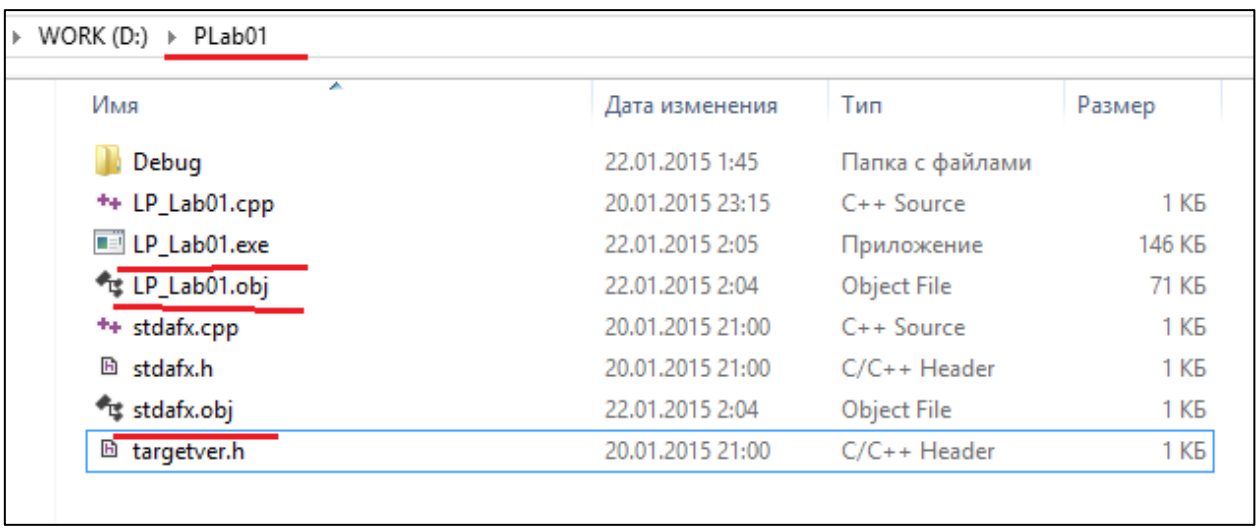

WORK (D:) ▸ PLab01 ▸ Debug
Имя
Дата изменения
Тип
Размер
LP_Lab01.cod 22.01.2015 1:02 C/C++ Code Listing 323 КБ
LP_Lab01.log 22.01.2015 0:51 Текстовый докум... 3 КБ
LP_Lab01.obj 22.01.2015 1:02 Object File 148 КБ
LP_Lab01.pch 22.01.2015 0:52 Precompiled Hea... 1 216 КБ
stdafx.cod 22.01.2015 0:52 C/C++ Code Listing 1 КБ
stdafx.obj 22.01.2015 0:52 Object File 11 КБ
vc110.idb 22.01.2015 1:02 VC++ Minimum R... 259 КБ
vc110.pdb 22.01.2015 1:02 Program Debug D... 396 КБ


Командная строка разработчика для VS2012
D:\PLab01>
D:\PLab01>
D:\PLab01>
D:\PLab01>link.exe /ERRORREPORT:PROMPT /OUT:"D:\PLab01\Debug\LP_Lab01.exe" /NOLOGO kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /MANIFEST /MANIFESTUAC:"level='asInvoker' uiAccess='false'" /manifest:embed /DEBUG /PDB:"D:\PLab01\Debug\LP_Lab01.pdb" /SUBSYSTEM:CONSOLE /TLBID:1 /DYNAMICBASE /NXCOMPAT /IMPLIB:"D:\PLab01\Debug\LP_Lab01.lib" /MACHINE:X86 Debug\LP_Lab01.obj Debug\stdafx.obj
D:\PLab01>
D:\PLab01>
D:\PLab01>



22. Простой (сокращенный) вариант
/EHsc – модель обработки исключений (перехватываются исключения C++)





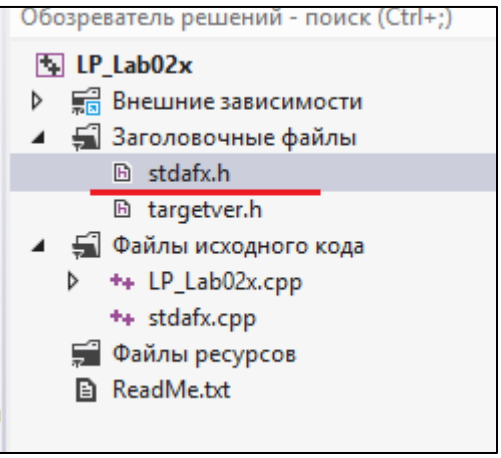
23. Загрузчик

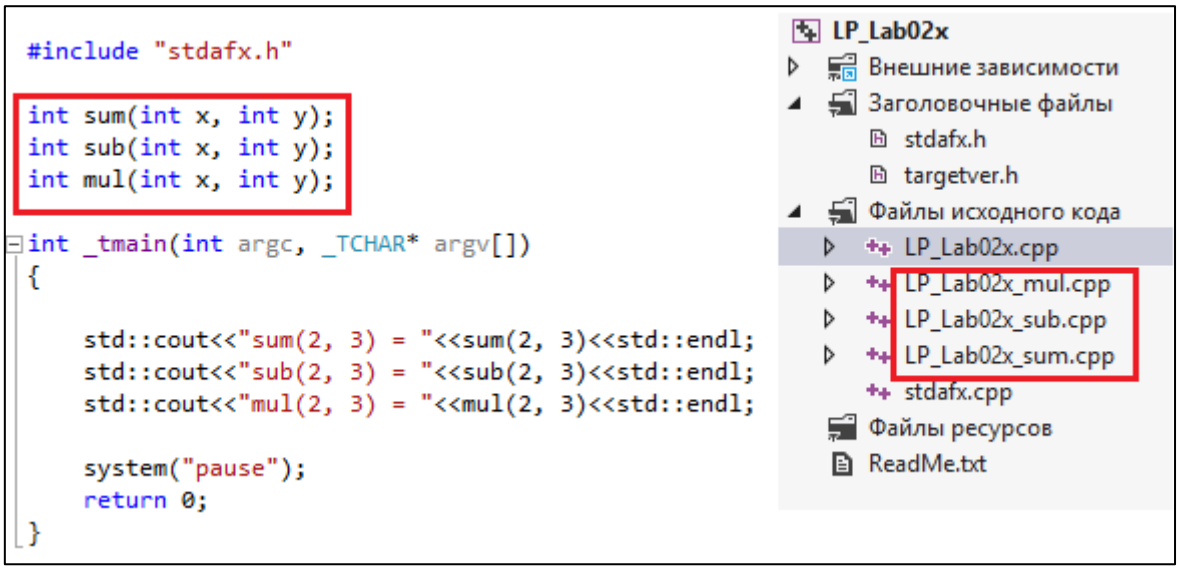
```
#include "stdafx.h"
#include <Windows.h>

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;
}
```

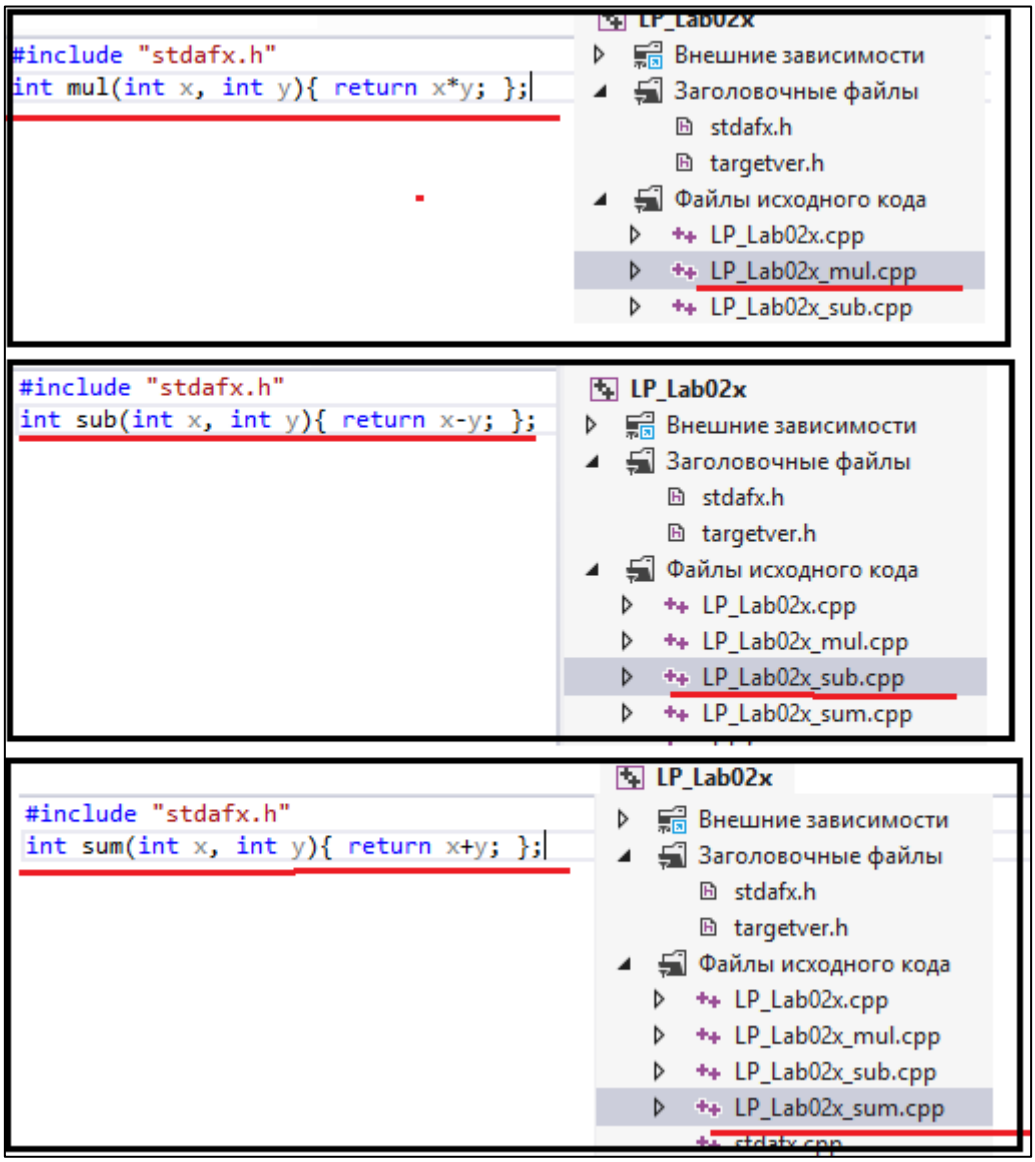
24. Многофайловый проект

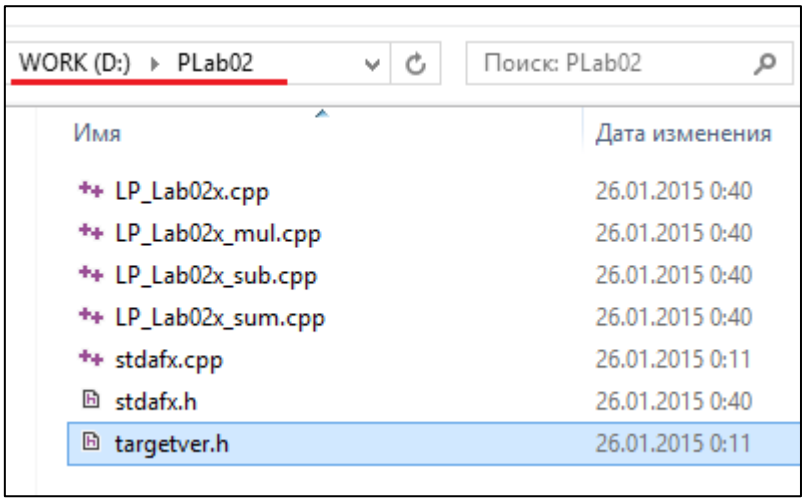
```
#include "stdafx.h"

int sum(int x, int y){ return x+y; };
int sub(int x, int y){ return x-y; };
int mul(int x, int y){ return x*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;
}
```

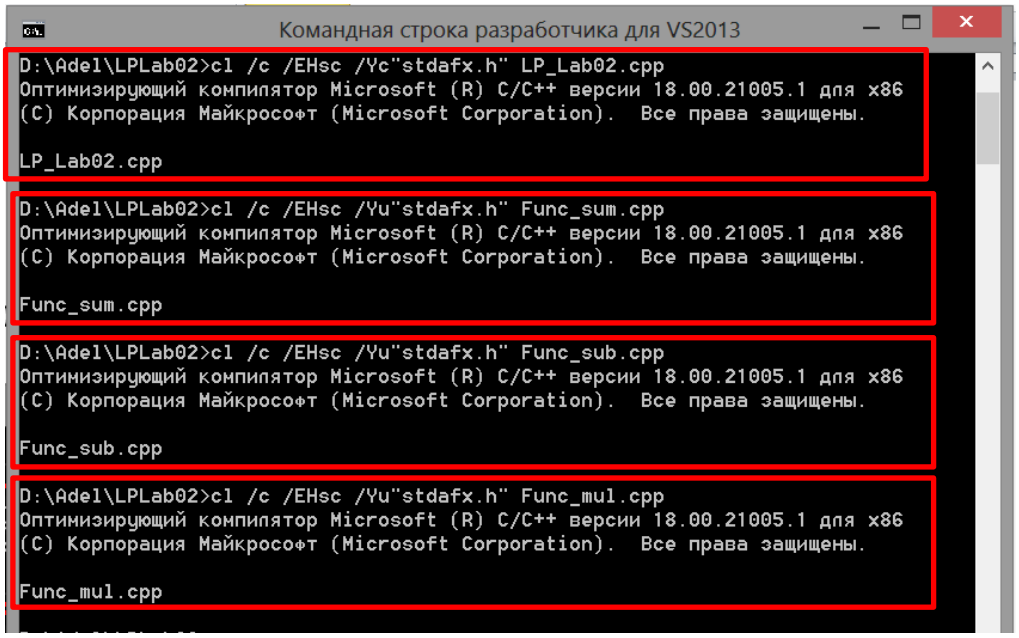
```
#pragma once

#include "targetver.h"

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

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


// TODO: Установите здесь ссылки
```



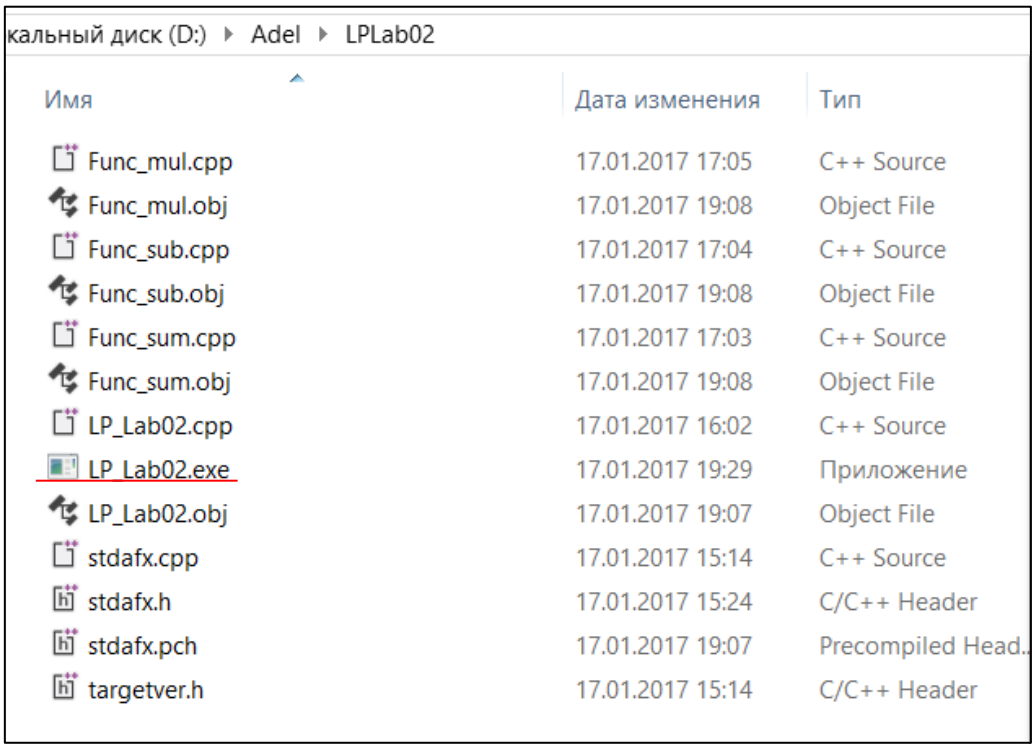

#include "stdafx.h"
int sum(int x, int y);
int sub(int x, int y);
int mul(int x, int y);
int _tmain(int argc, _TCHAR* argv[])
{
std::cout<<"sum(2, 3) = "<<sum(2, 3)<<std::endl;
std::cout<<"sub(2, 3) = "<<sub(2, 3)<<std::endl;
std::cout<<"mul(2, 3) = "<<mul(2, 3)<<std::endl;
system("pause");
return 0;
}
LP_Lab02x
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
stdafx.cpp
Файлы ресурсов
ReadMe.txt


#include "stdafx.h"
int mul(int x, int y){ return x*y; };
#include "stdafx.h"
int sub(int x, int y){ return x-y; };
#include "stdafx.h"
int sum(int x, int y){ return x+y; };
LP_Lab02x
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


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
targetver.h 26.01.2015 0:11


Командная строка разработчика для VS2013
D:\Adel\LPLab02>cl /c /EHsc /Yc"stdafx.h" LP_Lab02.cpp
Оптимизирующий 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>cl /c /EHsc /Yu"stdafx.h" Func_sum.cpp
Оптимизирующий 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>cl /c /EHsc /Yu"stdafx.h" Func_sub.cpp
Оптимизирующий 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>cl /c /EHsc /Yu"stdafx.h" Func_mul.cpp
Оптимизирующий компилятор Microsoft (R) C/C++ версии 18.00.21005.1 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Func_mul.cpp


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



25. **Заключение:**
    - стандарты ISO/IEC, ECMA, спецификации языков программирования;
    - определения: IDE, система программирования, язык программирования, транслятор, компоновщик, ассемблер, исходный, объектный и загрузочный код, профилировщик, отладчик;
    - структура классической системы программирования
    - примеры;
    - материал для 1 и 2 лабораторных работ.

26. 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	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.

Параметр	Цель
/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.
/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 _pexit.
/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,

Параметр	Цель
	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.
/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	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.

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

Параметр	Цель
/vd	Подавляет или включает скрытые vtordisp-члены класса.
/vmb	Использует оптимальное основание для указателей на члены.
/vmg	Использует полное обобщение для указателей на члены.
/vmm	Объявляет множественное наследование.
/vms	Объявляет одиночное наследование.
/vmv	Объявляет виртуальное наследование.
/volatile	Выбирает способ интерпретации ключевого слова volatile.
/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 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	**Создает файл предкомпилированного заголовка.**
/Yd	Не рекомендуется. Размещает полную отладочную информацию во всех объектных файлах. Используйте вместо этого параметр /Zi .
/Yl	Вводит ссылку PCH при создании отладочной библиотеки.
/Yu	**Использует файл предкомпилированного заголовка при построении.**
/Z7	Приводит к возникновению ошибки совместимости с C 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.

Параметр	Цель
/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).
/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.

27. 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.
/CLRTHREADATTRIBUTE	Указывает атрибут потока для применения к точке входа CLR-программы.
/CLRUNMANAGEDCODECHECK	Указывает, должен ли компоновщик применять атрибут SuppressUnmanagedCodeSecurity к создаваемым компоновщиком заглушкам PInvoke, осуществляющим вызовы из управляемого кода в библиотеки DLL неуправляемого кода.
/DEBUG	Создает отладочную информацию.
/DEBUGTYPE	Указывает, какие данные необходимо включить в отладочную информацию.
/DEF	Передает компоновщику файл определения модуля (DEF).
/DEFAULTLIB	Проводит поиск по указанной библиотеке при разрешении внешних ссылок.
/DELAY	Управляет отложенной загрузкой библиотек DLL.
/DELAYLOAD	Включает отложенную загрузку указанной библиотеки DLL.
/DELAYSIGN	Частично подписывает сборку.
/ DEPENDENTLOADFLAG	Задает флаги по умолчанию для зависимой загрузки DLL.
/DLL	Выполняет сборку библиотеки DLL.
/DRIVER	Создает драйвер режима ядра.
/DYNAMICBASE	Указывает, следует ли создавать исполняемый образ, базовый адрес которого может быть случайным образом изменен во время загрузки с помощью технологии ASLR.
/ENTRY	Задает начальный адрес.
/errorReport	Передает сведения о внутренних ошибках компоновщика в Майкрософт.

/EXPORT	Экспортирует функцию.
/ FILEALIGN	Выравнивание разделов в выходном файле на кратные с указанным значением.
/FIXED	Создает программу, которая может загружаться только по предпочтительному базовому адресу.
/FORCE	Принудительное завершение компоновки даже в случае наличия неразрешенных или многократно определенных символов.
/FUNCTIONPADMIN	Создает образ, для которого можно выполнять горячее обновление.
/GENPROFILE, /FASTGENPROFILE	Оба эти параметра задают создание PGD-файла компоновщиком для поддержки профильной оптимизации (PGO). /GENPROFILE и /FASTGENPROFILE используют разные параметры по умолчанию.
/GUARD	Включает защиту потока управления.
/HEAP	Задает размер кучи в байтах.
/HIGHENTROPYVA	Определяет поддержку 64-разрядной функции Address Space Layout Randomization (ASLR) с высоким уровнем энтропии.
/IDLOUT	Указывает имя файла IDL и имена других выходных файлов MIDL.
/IGNORE	Отменяет вывод указанных предупреждений компоновщика.
/IGNOREIDL	Предотвращает преобразование сведений атрибутов в файл IDL.
/IMPLIB	Переопределяет имя библиотеки импорта по умолчанию.
/INCLUDE	Принудительное использование ссылок на символы.
/INCREMENTAL	Управляет инкрементной компоновкой.
/INTEGRITYCHECK	Указывает на то, что модуль требует проверки подписи во время загрузки.
/KEYCONTAINER	Задает контейнер ключей для подписи сборки.
/KEYFILE	Задает ключ или пару ключей для подписи сборки.

/LARGEADDRESSAWARE	Указывает компилятору на то, что приложение поддерживает адреса, превышающие два гигабайта.
/LIBPATH	Указывает путь для поиска перед путем среды библиотеки.
/LTCG	Задает создание кода во время компоновки.
/MACHINE	**Указывает целевую платформу.**
/MANIFEST	Создает параллельный файл манифеста и при необходимости включает его в двоичный файл.
/MANIFESTDEPENDENCY	Указывает <dependentAssembly > раздела в файле манифеста.
/MANIFESTFILE	Изменяет имя файла манифеста по умолчанию.
/MANIFESTINPUT	Задает входной файл манифеста для обработки и внедрения компоновщиком в двоичный файл. Этот параметр можно использовать несколько раз, чтобы указать несколько входных файлов манифеста.
/MANIFESTUAC	Указывает, следует ли внедрять в манифест программы сведения о контроле учетных записей.
/MAP	Создает файл сопоставления.
/MAPINFO	Включает указанные сведения в файл сопоставления.
/MERGE	Объединяет разделы.
/MIDL	Задает параметры командной строки MIDL.
/ NATVIS	Добавляет визуализаторы отладчика из файла Natvis в PDB-ФАЙЛ.
/NOASSEMBLY	Подавляет создание сборки .NET Framework.
/NODEFAULTLIB	Пропускает все (или только указанные) библиотеки по умолчанию при разрешении внешних ссылок.
/NOENTRY	Создает библиотеку DLL, содержащую только ресурсы.
/NOLOGO	Отключает загрузочный баннер.
/NXCOMPAT	Помечает исполняемый файл как файл, проверенный на совместимость с компонентом предотвращения выполнения данных Windows.

/OPT	Управляет оптимизацией LINK.
/ORDER	Помещает секции COMDAT в образ в предопределенном порядке.
/OUT	Задает имя выходного файла.
/PDB	Создает файл базы данных программы (PDB).
/PDBALTPATH	Использует альтернативное местоположение для сохранения файла PDB.
/PDBSTRIPPED	Создает файл базы данных программы (PDB), не содержащий закрытых символов.
/PGD	Задает файл PGD для профильных оптимизаций.
/POGOSAFEMODE	Устаревшие **создает сборку инструментирования профильной Оптимизации поточно ориентированными.**
/PROFILE	Создает выходной файл, который может быть использован для профилировщика производительности инструментов.
/RELEASE	Задает контрольную сумму в заголовке файла EXE.
/SAFESEH	Указывает на то, что образ будет содержать таблицу безопасных обработчиков исключений.
/SECTION	Переопределяет атрибуты секции.
/ SOURCELINK	Указывает файл SourceLink для добавления в PDB.
/STACK	Задает размер стека (в байтах).
/STUB	Присоединяет программу-заглушку MS-DOS к программе Win32.
/SUBSYSTEM	**Указывает операционной системе, как запускать файл EXE.**
/SWAPRUN	Указывает операционной системе на необходимость копирования выходного файла компоновщика в файл подкачки перед его запуском.
/TLBID	Указывает идентификатор ресурса библиотеки типов, создаваемой компоновщиком.
/TLBOUT	Указывает имя файла TLB и имена других выходных

	файлов MIDL.
/TSAWARE	Создает приложение, специально рассчитанное на запуск под управлением сервера терминалов.
/USEPROFILE	Использует профильной оптимизации обучающих данных для создания оптимизированного образа.
/VERBOSE	Печатает сообщения хода выполнения компоновщика.
/VERSION	Присваивает номер версии.
/ WHOLEARCHIVE	Включает в себя каждого файла объект из указанного статических библиотек.
/WINMD	Включает создание файлов метаданных среды выполнения Windows.
/WINMDFILE	Задает имя файла для выходного файла метаданных среды выполнения Windows (winmd), создаваемого параметром компоновщика /WINMD .
/WINMDKEYFILE	Задает ключ или пару ключей для подписи файла метаданных среды выполнения Windows.
/WINMDKEYCONTAINER	Указывает контейнер ключей для подписания файла метаданных Windows.
/WINMDDELAYSIGN	Частично подписывает файл метаданных среды выполнения Windows (.winmd), установив открытый ключ в файле winmd.
/WX	Обрабатывает предупреждения компоновщика как ошибки.