**Лабораторная работа №1**

**Знакомство с CMake и Clang**

**Цель работы:** Получение практических навыков в работе с инструментом автоматической сборки проектов CMake и транслятором Clang. А также закрепление теоретических знаний о этапах разработки программ.

**Постановка задачи:**

**Приложение Lab-01C:**

Разработать приложение на языке C содержащее решение для задачи из индивидуального задания\*. При чём код должен быть кроссплатформенным (т.е. компилироваться для Windows и Linux/macOS без внесения дополнительных изменений).

**Вариант 3.** Целой переменной k присвоить значение, равное сумме цифр в записи целого положительного трехзначного числа x.

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  int main() {  int x, k = 0;  printf("Enter a three-digit number: ");  scanf("%d", &x);  if (x < 100 || x > 999) {  printf("Error: a non-three-digit number was entered\n");  }  else {  while (x > 0) {  k += x % 10;  x /= 10;  }  printf("The sum of the digits of a number: %d\n", k);  }  printf("Press Enter to exit...\n");  getchar();  getchar();  return 0;  } |

**Приложение Lab-01CS:**

Разработать приложение аналогичное первому, но на языке C#.

|  |
| --- |
| using System;  class Program  {  static void Main()  {  int x, k = 0;  Console.Write("Enter a three-digit number: ");  if (int.TryParse(Console.ReadLine(), out x))  {  if (x < 100 || x > 999)  {  Console.WriteLine("Error: a non-three-digit number was entered");  }  else  {  while (x > 0)  {  k += x % 10;  x /= 10;  }  Console.WriteLine($"The sum of the digits of a number: {k}");  }  }  else  {  Console.WriteLine("Error: invalid input.");  }  Console.WriteLine("Press Enter to exit...");  Console.ReadLine();  }  } |

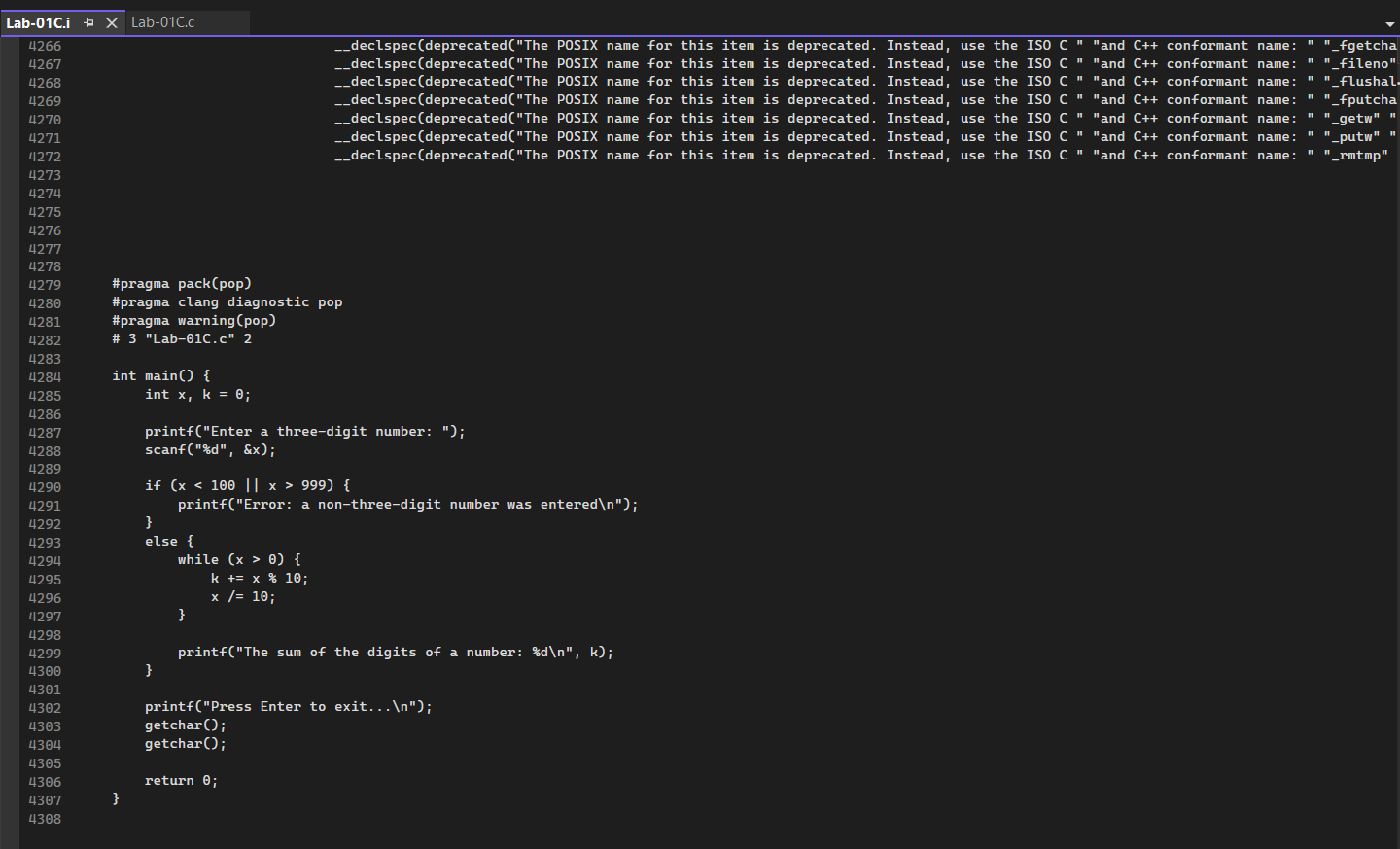
**Подготовить исполняемые файлы, полученные двумя разными способами, а именно:**

* с использованием транслятора Clang и компилятора С# напрямую;
* с использованием CMake.

**Данный пункт касается только приложения Lab-01C**!!! При использовании Clang напрямую должны быть получены промежуточные файлы сборки, т.е. файлы, которые получаются на выходе каждого этапа сборки (после препроцессинга, после компиляции, после компоновки).

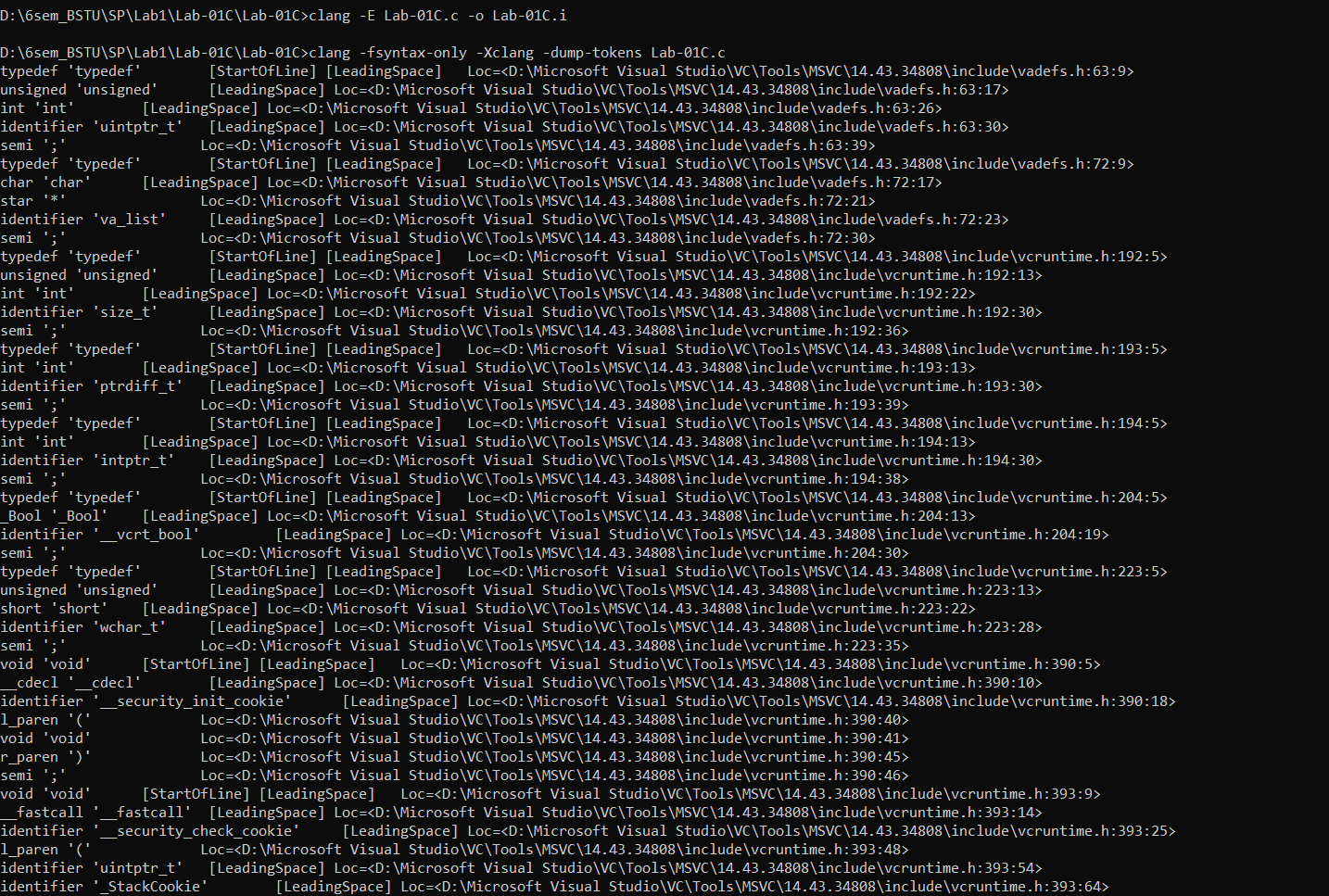
Выполяем только **препроцессинг**:

|  |
| --- |
| clang -E Lab-01C.c -o Lab-01C.i |



Получаем токены, на которые разбит код.

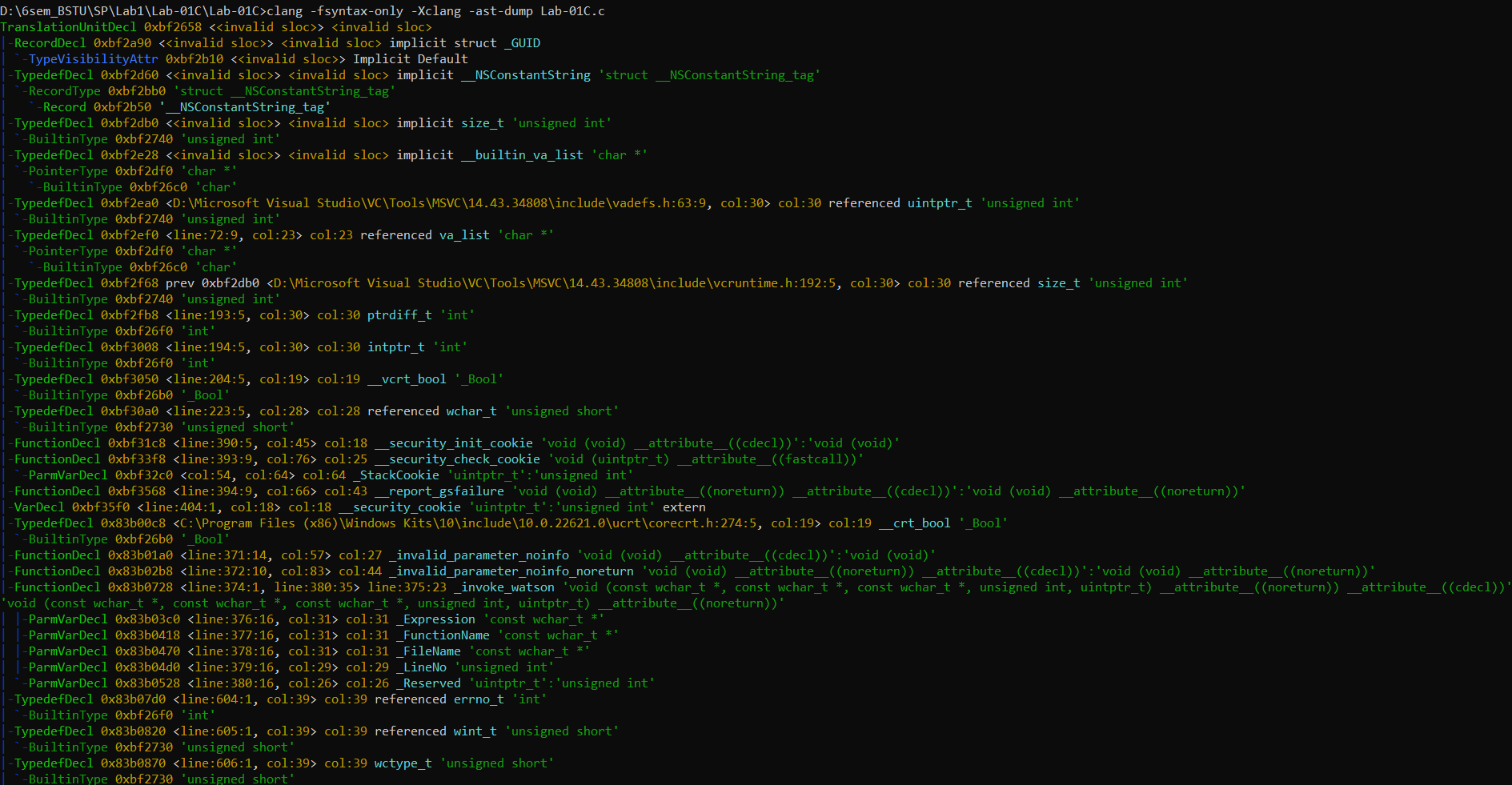
|  |
| --- |
| clang -fsyntax-only -Xclang -dump-tokens Lab-01C.c |

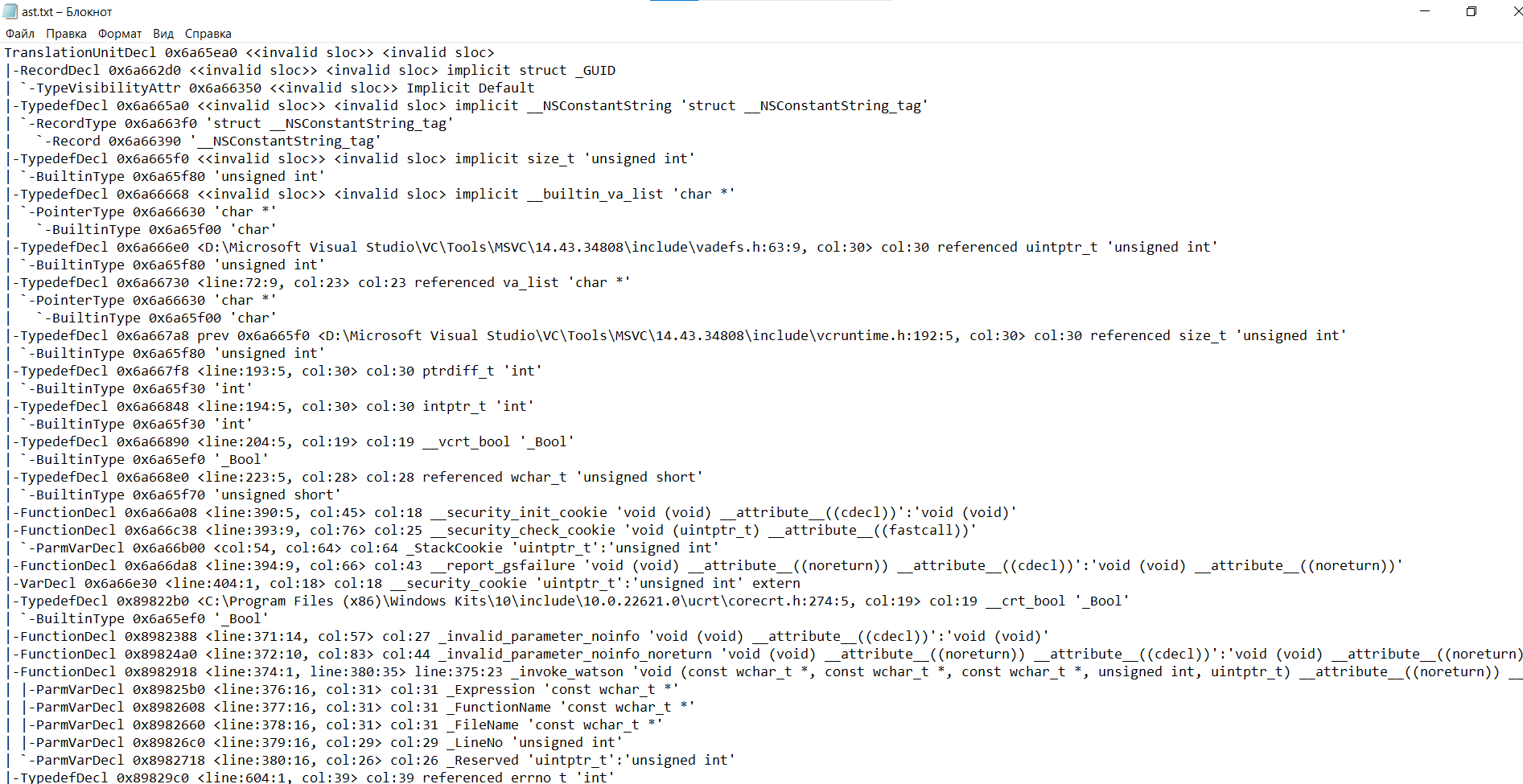


Выводим **абстрактное синтаксическое дерево (AST)**. AST показывает, как компилятор структурирует программу, включая функции, операторы, переменные и их типы.

Абстрактное синтаксическое дерево — это структура данных, которая представляет синтаксис исходного кода в виде дерева, где каждый узел соответствует конструкции языка (например, выражению, оператору или типу).

|  |
| --- |
| clang -fsyntax-only -Xclang -ast-dump Lab-01C.c > ast.txt |





Генерируем **LLVM IR (промежуточное представление):**

|  |
| --- |
| clang -fsyntax-only -Xclang -emit-llvm Lab-01C.c |



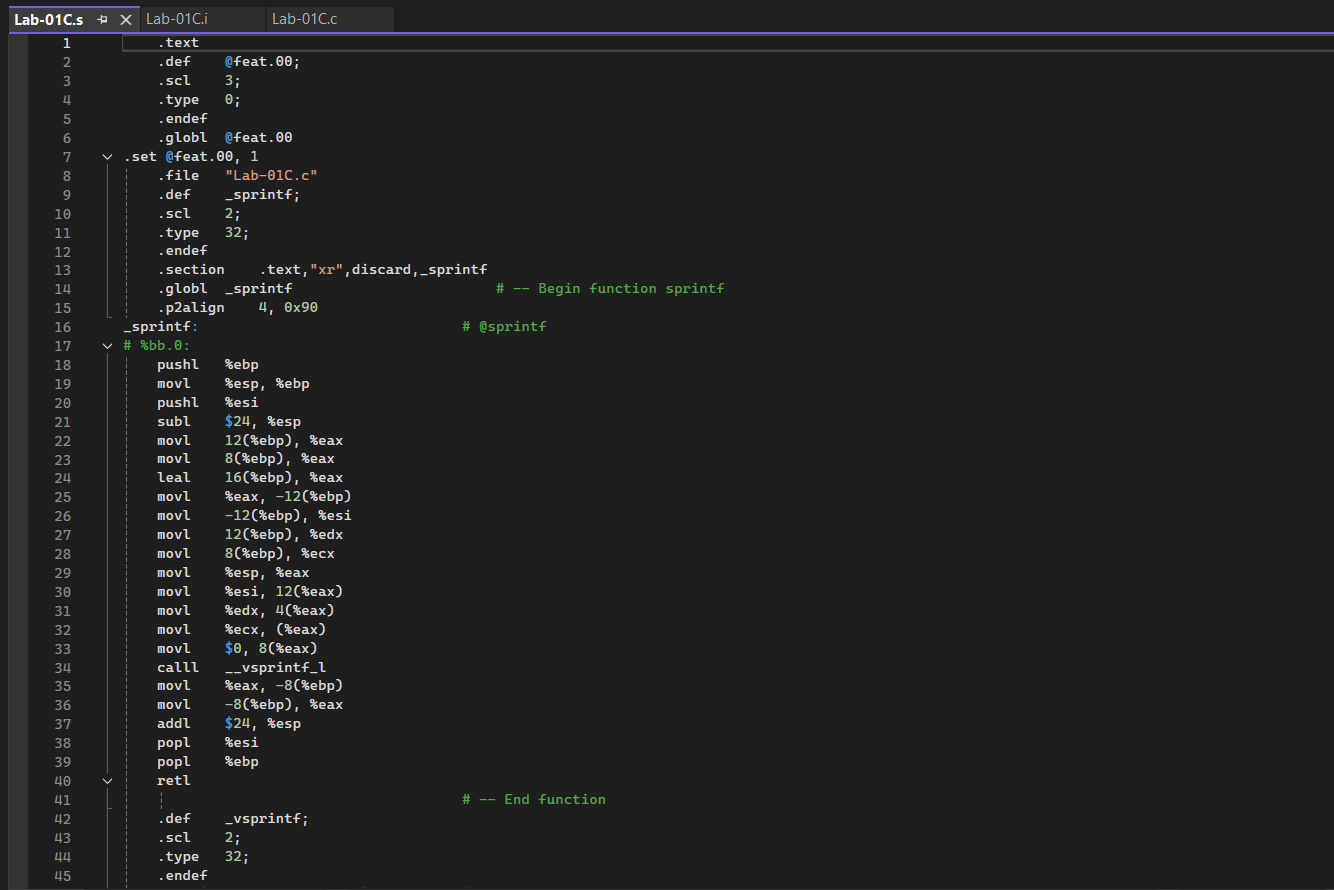
Оптимизируем код:

|  |
| --- |
| clang -fsyntax-only -O3 -Xclang -emit-llvm Lab-01C.c |



Генерируем ассемблерный код:

|  |
| --- |
| clang -S Lab-01C.c -o Lab-01C.s |

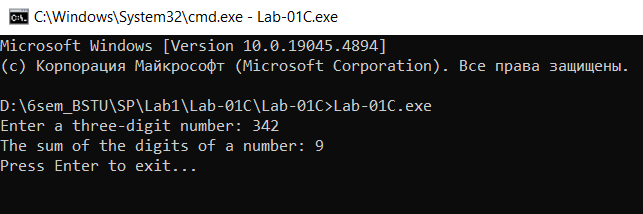


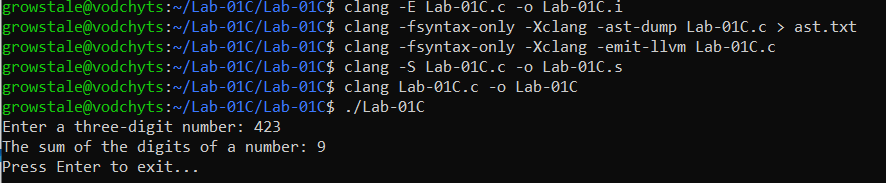
Компилируем исходный файл в объектный:

|  |
| --- |
| clang -c Lab-01C.c -o Lab-01C.o |

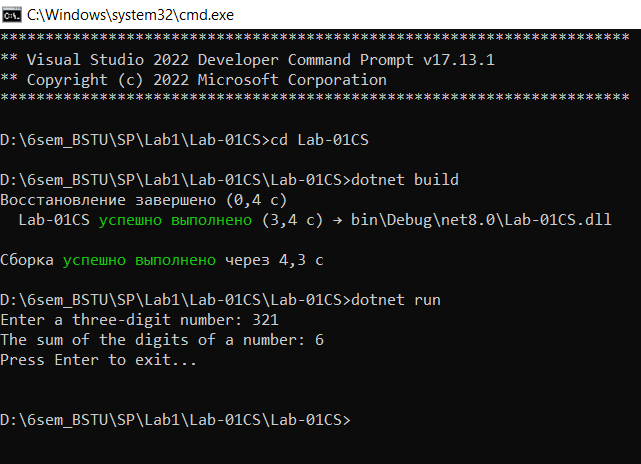
В результате выполнения следующей команды будет создан исполнимый файл:

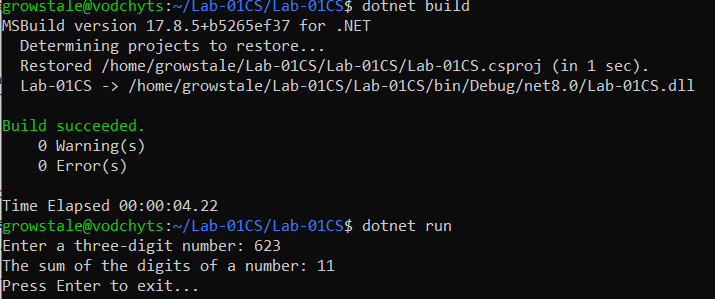
|  |
| --- |
| clang Lab-01C.c -o Lab-01C.exe |





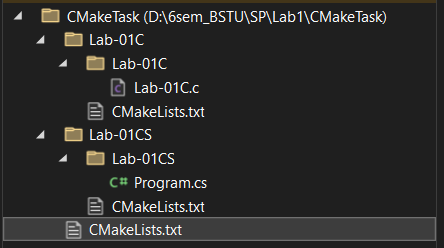


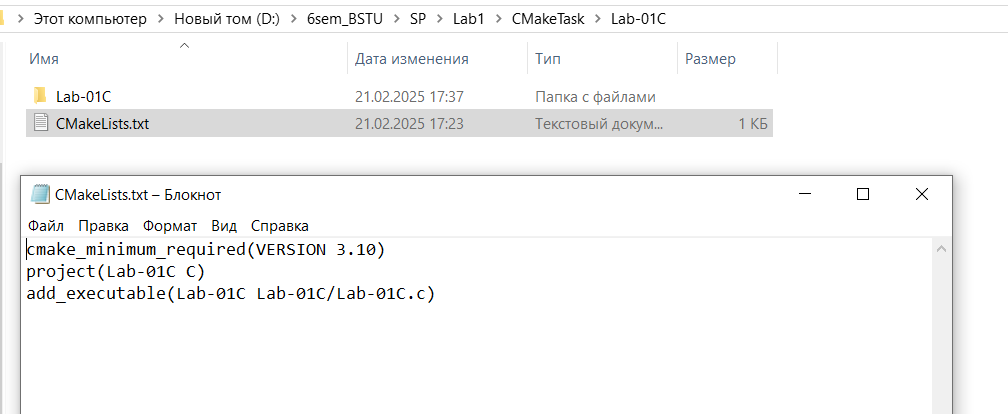


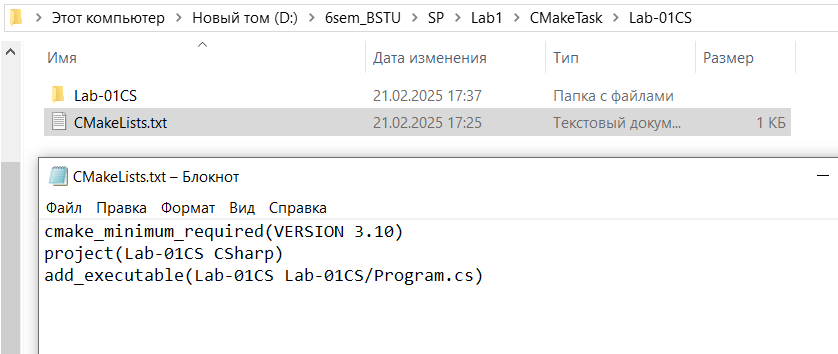


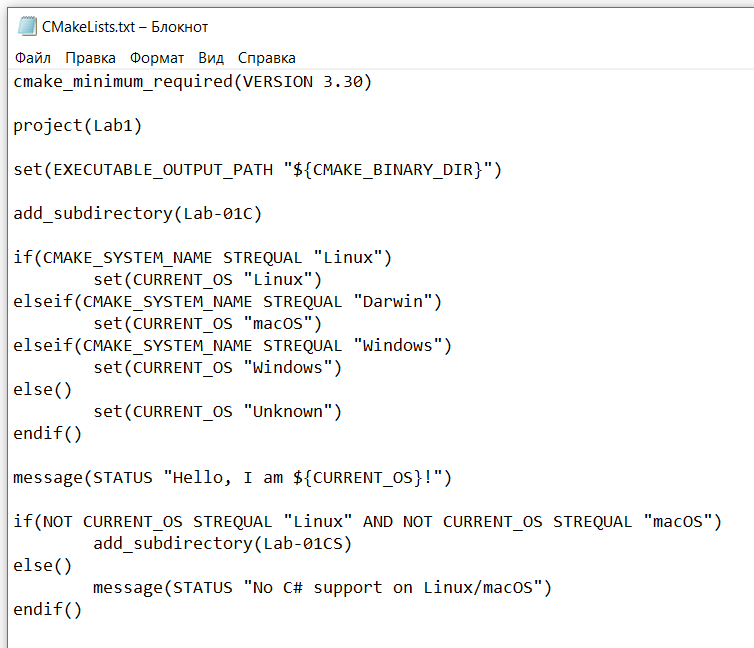
При использовании CMake должно быть выполнено следующее:

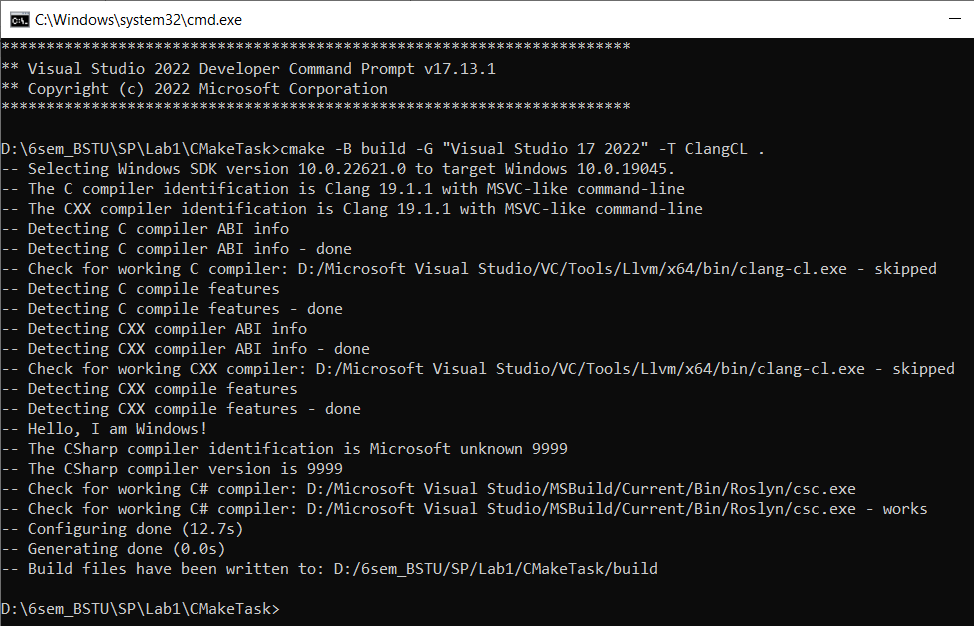
* Проект для сборки должен включать в себя два приложения на C и C#;
* На ОС семейства Linux/macOS проект С# должен игнорироваться при сборке (должно быть прописано в виде условия);
* Сборка должна быть «out-of-source»;
* Для Windows использовать генератор «Visual Studio» (для тех, у кого нет VS проще использовать генератор «[Ninja](https://ninja-build.org/)»), для Linux – «Unix Makefiles», для macOS – «XCode»;
* В конфигурации CMake (CMakeLists.txt) должны быть использованы переменные для определения текущей ОС, а также вывод сообщения в логи CMake сообщения вида «Hello, I am [название ОС]!».

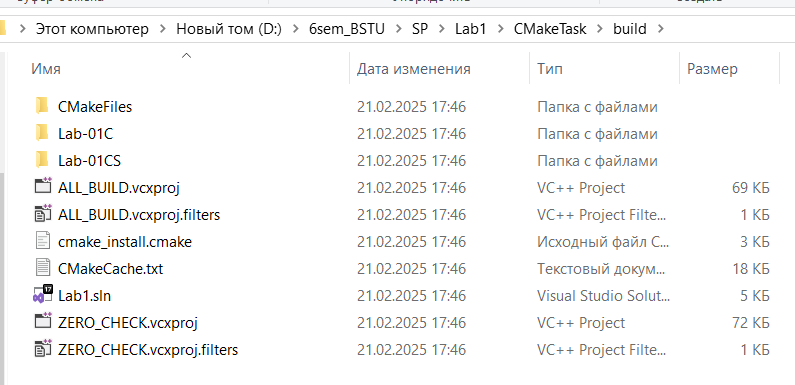


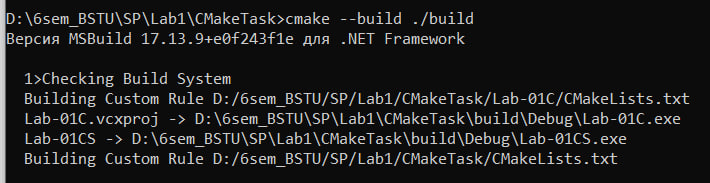


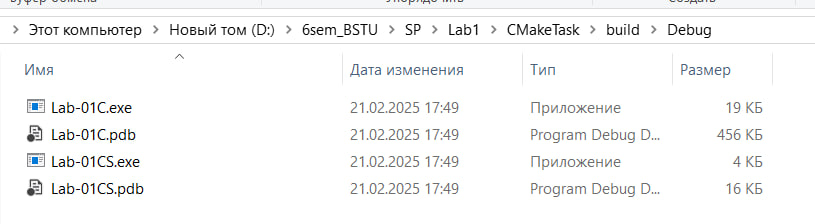


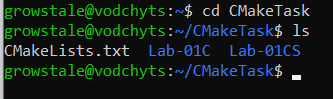


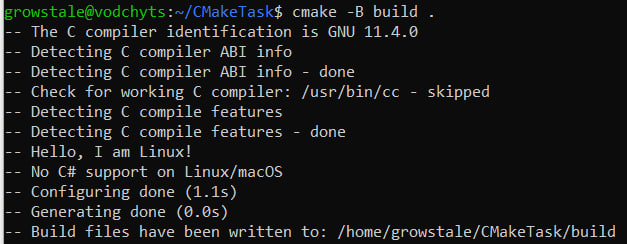


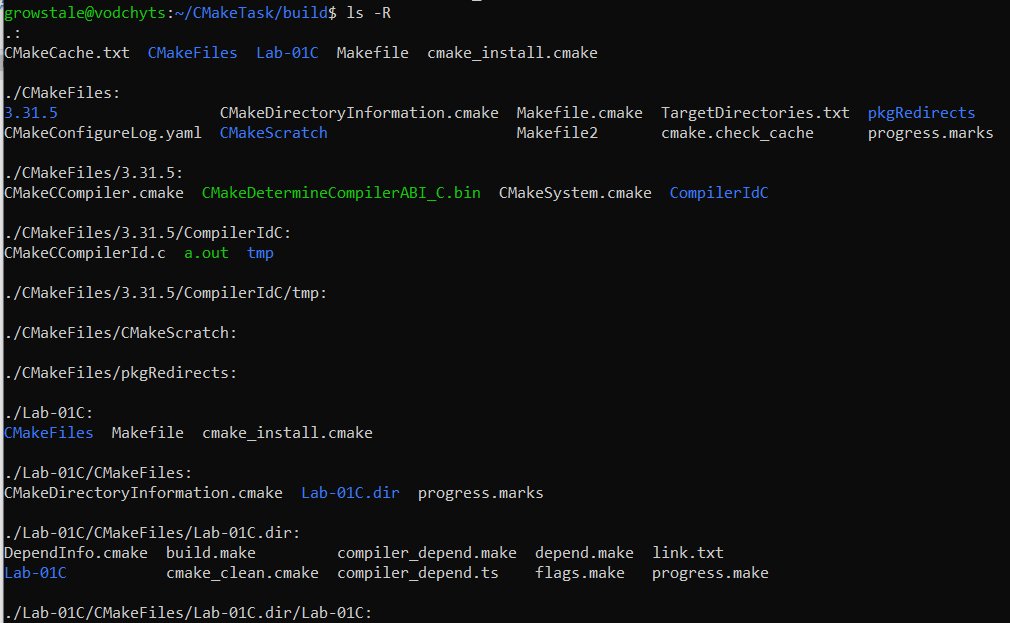


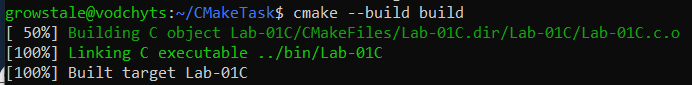


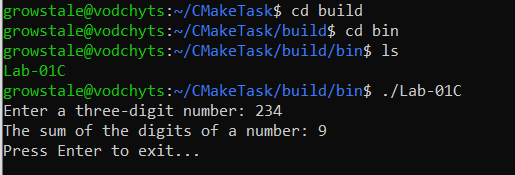








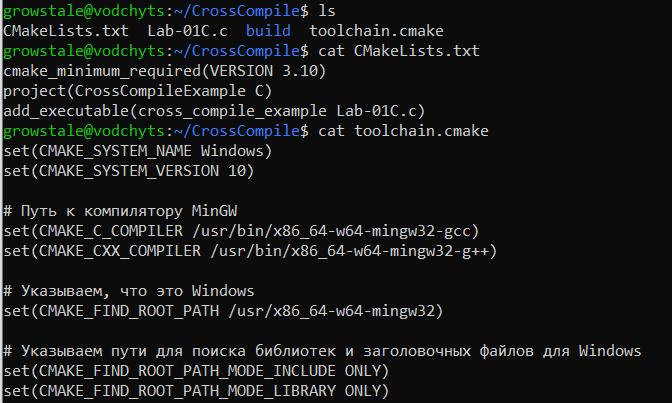


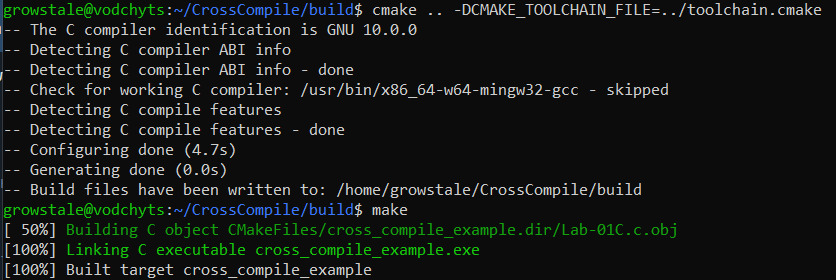


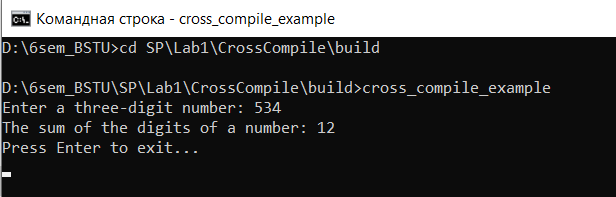
Разработанные приложения должны быть скомпилированы и протестированы как на Windows, так и на ОС из семейства Linux или macOS (для владельцев MacBook’ов).

Дополнительно (совершенно не обязательно, но огромный плюс):

Продемонстрировать кросс-компиляцию проекта (в любом направлении: Windows->Linux/macOS или Linux/macOS->Windows).







Собрать проект на C# через CMake на Linux/macOS.