

Lab1. Настройка окружения. Некоторые методы повышения производительности приложения. Эффективность работы с подсистемой памяти. Векторизация.

Подготовка к выполнению лабораторных работ

1. Необходимо установить инструменты из пакета **Intel oneAPI**:

- compiler
- advisor
- vtune
- inspector

Первые три доступны в рамках **Intel oneAPI Base Toolkit**. Последний в **Intel oneAPI HPC Toolkit**.

2. Настройка окружения

Прежде всего нужно выполнить скрипт **setvars.bat**, который располагается в корневой папке с установленными инструментами из пакета **oneAPI**. Например, путь до него может быть следующим:

```
"C:\Program Files (x86)\Intel\oneAPI\setvars.bat"
```

Для запуска скрипта нужно открыть командную строку: **win + s** -> ищем **cmd** (Командная строка) и запускаем это приложение. Далее в открывшейся командной строке вводим полный путь до скрипта, описанного выше, после чего в консоли появится следующее:

```
Microsoft Windows [Version 10.0.19044.2728]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\k.sandalov>"C:\Program Files (x86)\Intel\oneAPI\setvars.bat"
:: initializing oneAPI environment...
Initializing Visual Studio command-line environment...
Visual Studio version 17.6.1 environment configured.
"C:\Program Files\Microsoft Visual Studio\2022\Community\"
Visual Studio command-line environment initialized for: 'x64'
: advisor -- latest
: compiler -- latest
: debugger -- latest
: dev-utilities -- latest
: dpl -- latest
: tbb -- latest
: vtune -- latest
:: oneAPI environment initialized ::
```

```
C:\Users\k.sandalov>
```

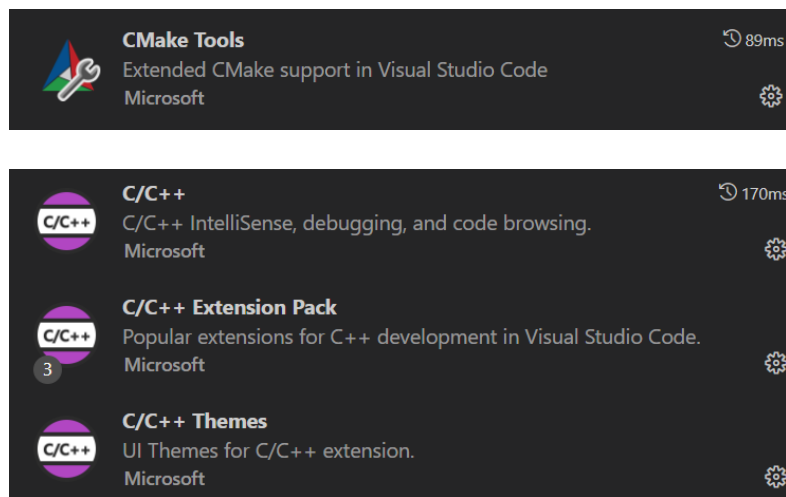
Затем открываем IDE, используя эту же командную строку. И в целом, все инструменты разработки и анализа можно запускать в этой среде. В качестве возможных средств разработки предлагается рассмотреть VS Code или же просто Visual Studio (версия от 2016 года или новее).

- VS Code

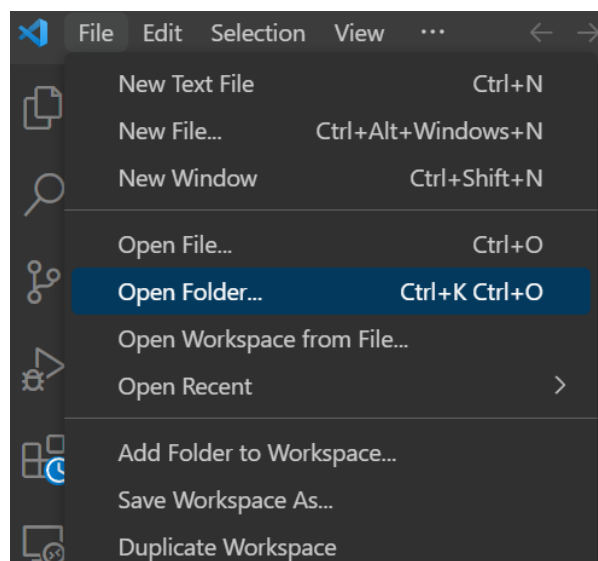
Для запуска VS Code можно ввести команду в терминале:

```
C:\Users\k.sandalov>code
```

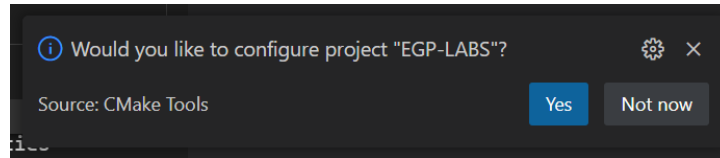
Для работы с проектом в VS Code нужны следующие расширения (Скачать можно на вкладке *Extensions*):



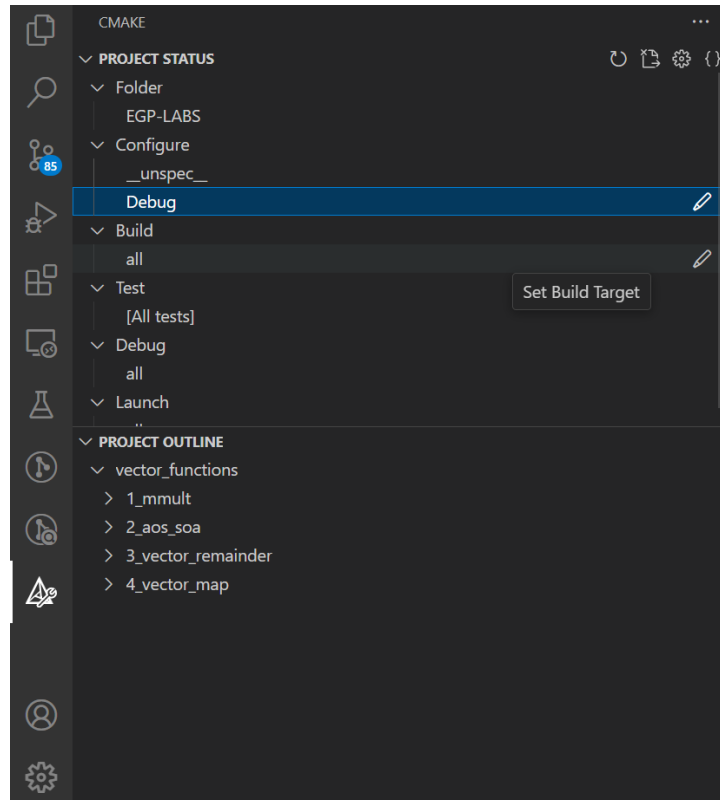
Далее в открывшемся окне переходим в верхнем меню к выбору папки, в которой располагается проект(корневая директория проекта):



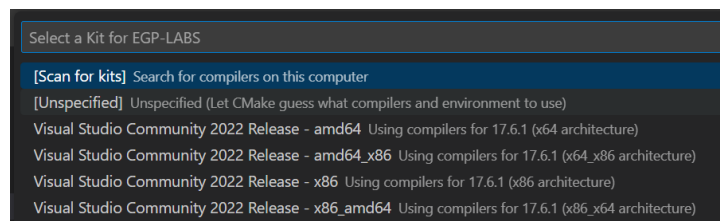
После открытия папки с проектом должно появиться сообщение с предложением сконфигурировать CMake проект, необходимо сделать это:



Для удобной работы с CMake проектом можно воспользоваться меню слева:



Например, в пункте **Configure** первой строчкой можно выбрать используемые средства для сборки проекта (в нашем случае unspecified)




- Visual Studio

Для запуска Visual Studio используем команду:

```
C:\Users\k.sandalov>devenv
```


Далее необходимо так же, просто открыть папку с проектом:

Начало работы




Клонирование репозитория

Получить код из интернет-репозитория, например, GitHub или Azure DevOps




Открыть проект или решение

Открыть локальный проект Visual Studio или SLN-файл



Открыть локальную папку

Перейти и изменить код в любой папке



Создание проекта

Выберите шаблон проекта с формированием шаблонов кода, чтобы начать работу

[Продолжить без кода →](#)

В случае, если была корректно выбрана корневая директория проекта и настроено окружение, Visual Studio успешно сможет настроить CMake проект:

Вывод

Показать выходные данные из: CMake

```
1> Запущено создание CMake для конфигурации по умолчанию: "x64-Debug".
1> Командная строка: "C:\Windows\system32\cmd.exe" /c "XSYSTEMROOT%\System32\chcp.com 65001 && "C:\PROGRAM FILES\MICROSOFT VISUAL ...
1> Рабочий каталог: D:\workfolder\EGP-LABS\out\build\64-Debug
1> [CMake] -- The C compiler identification is IntelLVM 2023.1.0 with MSVC-like command-line
1> [CMake] -- The CXX compiler identification is IntelLVM 2023.1.0 with MSVC-like command-line
1> [CMake] -- Detecting C compiler ABI info
1> [CMake] -- Detecting C compiler ABI info - done
1> [CMake] -- Check for working C compiler: C:/Program Files (x86)/Intel/oneAPI/compiler/latest/windows/bin/icx-cl.exe - skipped
1> [CMake] -- Detecting C compile features
1> [CMake] -- Detecting C compile features - done
1> [CMake] -- Detecting CXX compiler ABI info
1> [CMake] -- Detecting CXX compiler ABI info - done
1> [CMake] -- Check for working CXX compiler: C:/Program Files (x86)/Intel/oneAPI/compiler/latest/windows/bin/icx-cl.exe - skipped
1> [CMake] -- Detecting CXX compile features
1> [CMake] -- Detecting CXX compile features - done
1> [CMake] -- Detecting CXX compile features
1> [CMake] -- Detecting CXX compile features - done
1> [CMake] -- D:/workfolder/EGP-LABS/1_mmult
1> [CMake] -- D:/workfolder/EGP-LABS/2_0os_10a
1> [CMake] -- D:/workfolder/EGP-LABS/3_vector_remainder
1> [CMake] -- D:/workfolder/EGP-LABS/4_vector_map
1> [CMake] -- Configuring done (0.2s)
1> [CMake] -- Generating done (0.0s)
1> [CMake] -- Build files have been written to: D:/workfolder/EGP-LABS/out/build/64-Debug
1> Извлеченные переменные CMake.
1> Извлеченные исходные файлы и заголовки.
1> Извлеченная модель кода.
1> Извлеченные конфигурации цепочек инструментов.
1> Извлеченные данные включают пути.
1> Создание CMake завершено.
```

Список ошибок

Вывод

Далее настройки проекта в первую очередь будут браться из файла **CMakeLists.txt** (смотрите справа в обозревателе решений). Однако так же есть возможность настраивать его через файл **CMakeSettings.json**. Для этого можно использовать переменные и кэш CMake:

Переменные и кэш CMake

Содержит пару имя-значение для переменных CMake. Измененные переменные будут сохранены в файле CMakeSettings.json.

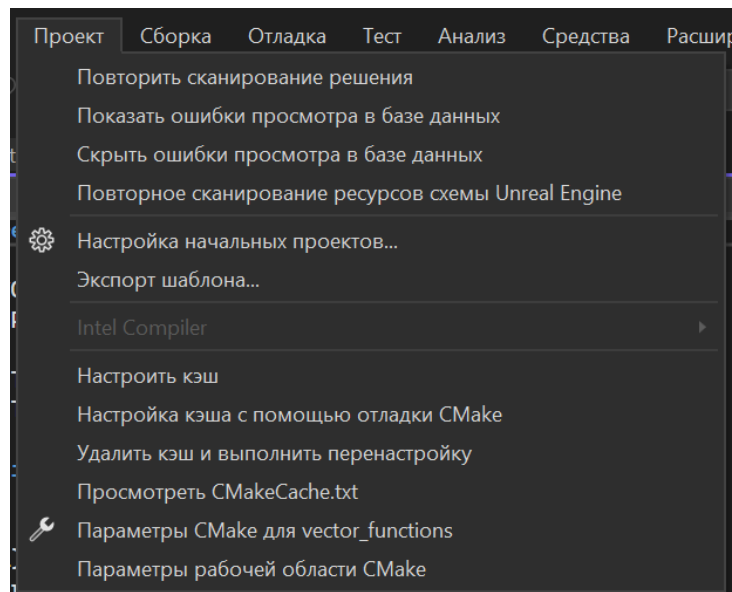
Сохранить и создать кэш CMake для загрузки переменных

☒ Показать дополнительные переменные

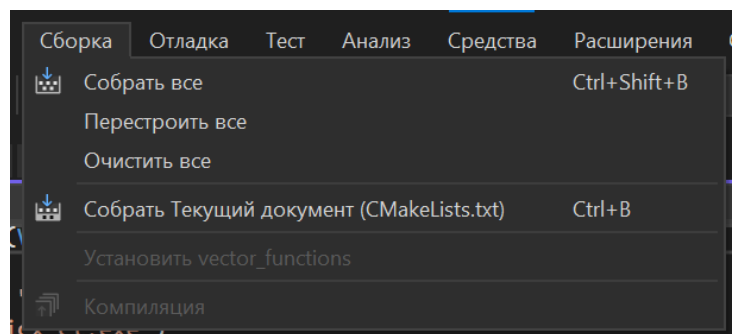
Имя	Значение	С...
CMAKE_CXX_COMPILER	C:/Program Files (x86)/Intel/oneAPI/compile	<input type="checkbox"/>
CMAKE_CXX_COMPILER_AR	C:/Program Files (x86)/Intel/oneAP	<input type="checkbox"/>
CMAKE_CXX_COMPILER_RANLIB	CMAKE_CXX_COMPILER_RANLIB-N	<input type="checkbox"/>
CMAKE_CXX_FLAGS	/DWIN32 /D_WINDOWS /W3 /GR /EHsc	<input type="checkbox"/>
CMAKE_CXX_FLAGS_DEBUG	/MDd /Zi /Ob0 /Od /RTC1	<input type="checkbox"/>
CMAKE_CXX_FLAGS_MINSIZEREL	/MD /O1 /Ob1 /DNDEBUG	<input type="checkbox"/>
CMAKE_CXX_FLAGS_RELEASE	/MD /O2 /Ob2 /DNDEBUG	<input type="checkbox"/>
CMAKE_CXX_FLAGS_RELWITHDEBINFO	/MD /O3 /Zi /Ob1 /DNDEBUG	<input type="checkbox"/>

4 / 5

А для того, чтобы заново переконфигурировать проект можно заново создать кэш CMake, если до этого обновили значения каких-то переменных, чтобы использовались актуальные значения (Удалить кэш и выполнить перенастройку):



Для сборки используем пункт меню сборки -> перестроить все:



Методы повышения эффективности работы с подсистемой памяти.
Векторизация.

1. **MMULT**
TBD
2. **SOA_TO_AOS**
TBD
3. **VECTORIZATION_PEEL_BODY_REMAINDER**
TBD
4. **VECTORIZATION_MAP**
TBD
5. **VECTORIZED_FUNCTIONS**
TBD