Модуль "Утилиты". Вопросы.

1. **Сборка**

   a. **Раздельная компиляция**
   Что такое файл исходного кода и исполняемый файл? Этап сборки программы: препроцессинг, ассемблирование, компиляция и линковка. Что такое заголовочные файлы (header-файлы)? Что делает директива препроцессора `#include`? Что такое единица трансляции? Компиляция программы с помощью `g++`. Опции компиляции `-E`, `-S` и `-c`. Что такое раздельная компиляция и в чём её преемущества?

   b. **Библиотеки**
   Что такое библиотека? Виды библиотек: header-only библиотеки, open-source библиотеки, статические библиотеки, динамические библиотеки. В чём различия между этими видами библиотек? В чём преимущества и недостатки каждого из видов библиотек? Как подключить библиотеки к своему проекту?

   c. **Статические библиотеки**
   Как создать статическую библиотеку? Как подключить статическую библиотеку? Опции компилятора `-I`, `-L` и `-l`. Характерные расширения файлов статических библиотек на Linux и Windows.

   d. **Динамические библиотеки**
   В чём главная разница между статическими и динамическими библиотеками? Как создать динамическую библиотеку? Как подключить динамическую библиотеку? Характерные расширения файлов динамических библиотек на Linux и Windows.

   e. **Опции компилятора `g++`**
   - Опции для указания стандарта языка, например `-std=c++20`
   - Опции для включения/отключения предупреждений: `-Wall`, `-Wextra`, `-Werror`.
   - Опция для указания директорий заголовочных файлов, необходимых для компиляции `-I`
   - Опция для указания директорий библиотек, необходимых для компиляции `-L`
   - Опция для указания названий библиотек, необходимых для компиляции `-l`
   - Опция для включения возможности проведения дебага: `-g`
   - Опции для включения оптимизаций: `-O0`, `-O1`, `-O2`, `-O3`, `-Os`
   - Опция `-DNDEBUG`
   - Опция `-D` для задания `#define`-макросов. Как ёё использовать? Пример использования данной опции.

2. **CMake как система сборки**

   a. **Основы CMake**
   Что такое Cmake и для чего он нужен? Основы работы с CMake. Структура CMake-проекта. Файл CMakeListis.txt. Как скомпилировать проект с помощью CMake? Что делают следующие команды CMake:
   - `cmake_minimum_required`
   - `project`
   - `add_executable`
   - `message`

   Как собрать проект с использованием CMake? Генерация файлов проекта для данной среды. Выбор генератора. Опции программы `cmake`: `-S`, `-B`, `-G`, `--build`.

   b. **Таргеты**
   Что такое таргет (target)? Что делают следующие команды CMake:
   - `add_executable`
   - `add_library` и её опции `STATIC` и `SHARED`
   - `target_link_libraries` (если аргумент является таргетом)

   c. **Свойства таргетов**
   Что делают следующие команды CMake:
   - `target_include_directories`
   - `target_link_directories`
   - `target_link_libraries` (если аргумент не является таргетом)

- `target_compile_features`
- `target_compile_definitions`
- `target_compile_options`

d. **Типы зависимостей между таргетам**
Типы связей между двумя таргетам `PRIVATE`, `PUBLIC` и `INTERFACE`. Типы связей между таргетом и его свойством `PRIVATE`, `PUBLIC` и `INTERFACE`. В чём отличия между этими типами зависимостей? Зачем нужно указывать тип для каждой связи? Примеры ситуаций когда нужно использовать ту или иную связь.

e. **Простые переменные CMake**
Простые переменные CMake. Какие бывают типы у переменных языка CMake? Как создать простую переменную в CMake? Команда `set`. Как напечатать значение переменной на экран? Основные стандартные переменные:

- `CXX_STANDARD`
- `CMAKE_CXX_COMPILER`
- `CMAKE_SOURCE_DIR`
- `CMAKE_BUILD_DIR`
- `BUILD_SHARED_LIBS`
- `WIN32`, `LINUX`, `APPLE`, `MSVC`, `MINGW`

f. **Поддиректории**
Как добавить новую поддиректорию в CMake проект? Команда `add_subdirectory`. Что происходит при выполнении этой команды? Область видимости переменных. Видны ли переменные, созданные в родительской Cmake-директории, в поддиректории? Видны ли переменные, созданные в поддиректории, в родительской Cmake-директории? Опция `PARENT_SCOPE` команды `set`. Переменные:

- `CMAKE_CURRENT_SOURCE_DIR`
- `CMAKE_CURRENT_BUILD_DIR`

3. **CMake как язык программирования**

a. **Переменные**
Какие бывают типы у переменных языка CMake? Как создать простую переменную в CMake? Команда `set`. Как получить значение переменной по её названию?

b. **Условная команда `if`**
Как пользоваться командой `if` и сопутствующими командами в языке CMake? Какие строки команда `if` воспринимает как истинные, а какие как ложные? Использование переменных как аргуметы команды `if`.

c. **Списки**
Что представляет собой список в языке CMake? Как создать список? Как работать со списком? Передача списка в функцию. Команда `list`. Опции этой команды: `LENGTH`, `GET`, `FIND`, `APPEND`, `SORT`.

d. **Циклы**
Команда `while`. Команда `foreach`. Опции команды `foreach`: `RANGE` и `IN LISTS`. Итерирование по списку с помощью команды `foreach`.

e. **Функции**
Функции в языке CMake. Как создать функцию с помощью команды `function`? Как передавать в функцию? Переменные `ARGC`, `ARGV`, `ARGN`. Как возвращать из функции. Опция `PARENT_SCOPE` команды `set`. Команда `return`. Области видимости функций.

f. **Манипуляции со строками**
Команда `string` и её опции:

- `FIND`
- `REPLACE`
- `APPEND`
- `JOIN`
- `TOLOWER`
- `TOUPPER`
- `LENGTH`
- `SUBSTRING`
- `COMPARE`

g. **Файлы**
Команда `file` и её опции:

- `READ`
- `STRINGS`
- `WRITE`
- `MAKE_DIRECTORY`
- `REMOVE`
- `RENAME`
- `COPY`
- `SIZE`
- `CHMOD`
- `REAL_PATH`
- `DOWNLOAD`
- `GLOB`

Является ли хорошей идеей использование команды `file` с опцией `GLOB`, чтобы найти названия всех файлов исходного кода некоторого таргета?

h. **Модули**
Что представляет собой модуль в языке CMake. Подключение модулей. Команда `include`. В каких папках ищутся модули? Переменная `CMAKE_MODULE_PATH`. Область видимости переменных. Переменная `CMAKE_CURRENT_LIST_FILE`. Чем команда `include` отличается от команды `add_subdirectory`?

4. **CMake - дополнительные возможности**

a. **Кешированные переменные CMake.**
Переменные среды. Кешированные переменные. Чем кешированные переменные отличаются от обычных переменных? Поле `type` при создании кешированной переменной и какие значения оно может принимать. Изменение кешированных переменных. Задание кешированных переменных внутри CMake-скрипта, в командной строке и путём изменения файла CMakeCache.txt.

5. **CMake - подключение сторонних библиотек**
6. **Git - локально**
7. **Git - удалённые репозитории**
8. **Тестирование**