10. Автоматизация сборки проекта, утилита make. Сценарий сборки проекта. Простойс
ценарий сборки.
Использование переменных и комментариев. Сборка программы с разными параметрами к
омпиляции.
План ответа:
	1 автоматизация сборки проекта: основные задачи, «исходные» данные;
	2 разновидности утилиты make;
	3 сценарий сборки проекта: название файла, его структура;
	4 простой сценарий сборки;
	5 использование переменных;
	6 условные конструкции в сценарии сборки.

1 автоматизация сборки проекта: основные задачи, «исходные» данные;
	Позволяет распределить работу над проектом между несколькими программиста
ми.
	Код программы более удобочитаем.
	Сокращает время повторной компиляции.
	1.1. Задача автоматизации сборки проекта – избавить программиста от необх
одимости каждый раз печатать объёмные вызовы компилятору и компоновщику в весьма
больших проектах.
	1.2. Данными для автоматической сборки являются файлы заголовков, реализа
ции и библиотеки (вход), исполняемые файлы и библиотеки (выход).
	1.3. Для автоматической сборки проекта применяют несколько способов:
		BAT-файлы
		Специализированные программные средства сборки (т.н. make)

2 разновидности утилиты make;
	2.1. make – утилита, автоматизирующая процесс преобразования файлов из од
ной формы в другую.
	2.2. Известны следующие разновидности средств автоматической сборки проек
та (т.н. make):
	GNU Make (мы им пользовались)
	BSD Make
	Microsoft Make (nmake)

3 сценарий сборки проекта: название файла, его структура;
	Необходимо создать так называемый сценарий сборки проекта (make-файл).
	Этот файл описывает отношения между файлами программы; содержит
команды для обновления каждого файла.
	Утилита make использует информацию из make-файла и время последнего измен
ения каждого файла для того, чтобы решить, какие файлы нужно обновить.
	Утилита make предполагает, что по умолчанию сценарий сборки называется ma
kefile или Makefile.
	цель: зависимость_1 ... зависимость_n
		команда_1
		команда_2

4 простой сценарий сборки;
	Очищаем каталог от старых исполняемых и объектных файлов
	Собираем объектные файлы для модулей
	Собираем объектные файлы для точек входа
	Собираем исполняемые файлы, по одной точке входа на каждый, из объектных
модулей.

```
	greeting.exe : hello.o main.o
		gcc -o greeting.exe hello.o main.o
	hello.o : hello.c hello.h
		gcc -std=c99 -Wall -Werror -pedantic -c hello.c
	main.o : main.c hello.h buy.h
		gcc -std=c99 -Wall -Werror -pedantic -c main.c
	clean :
		rm *.o *.exe
```


5 использование переменных;
	Строки, которые начинаются с символа #, являются комментариями.

Определить переменную в make-файле можно следующим образом: VAR_NAME := value
Чтобы получить значение переменной, необходимо ее имя заключить в круглые скобки и перед ними поставить символ '$'.
$(VAR_NAME)

6 условные конструкции в сценарии сборки.

```
# Опции компиляции
CFLAGS := -std=c99 -Wall -Werror -pedantic

ifeq ($(mode), debug)
    # Отладочная сборка: добавим генерацию отладочной информации
    CFLAGS += -g3
endif

ifeq ($(mode), release)
    # Финальная сборка: исключим отладочную информацию и утверждения (asserts)
    CFLAGS += -DNDEBUG -g0
endif
```