МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 7**

по дисциплине:” Системное программирование”

на тему: ”*Автоматическая сборка многофайловых проектов в Linux****”***

Выполнил**:** студент группы 10702221

Баяманова Айжамал

Принял**:** Давыденко Н.В.

Минск, 2023

**Цель работы :**

Изучить встроенный инструментарий для разработки приложений под семейство ОС Linux и фундаментальные основы системного программирования с использованием компиляторов gcc/g++, отладчика gdb и других для проектирования, компиляции, отладки и запуска приложений на языке программирования С/C++

**Требования :**

1) Разработать модульное консольное приложение на С/С++ согласно варианту задания. Для компиляции, компоновки и выполнения программы использовать стандартный компилятор Linux gcc/g++, а для отладки - gdb.

2) Размерность массива задаётся пользователем на стадии выполнения программы. Для повышения производительности программы необходимо использовать динамическое выделение памяти.

3) Во время работы программы должны выводить на экран исходные и конечные данные.

4) При выполнении задания запрещается использовать интегрированные средства разработки (Integrated Development Environment, IDE). Рекомендуется задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++.

5) При разработке программ необходимо придерживаться соглашения по написанию кода на C/C++ (Code-Convention).

**Упражнения:**

**Упражнение 1.**

**Изучите и выполните требования, представленные в лабораторной работе №2 «Компиляция и отладка простейшего приложения в Linux». Разработайте исходные коды приложений в соответствии с заданиями. Компиляцию приложений проведите с помощью консольных команд**

***Задание лабораторной работы №2****. Компиляция и отладка простейшего приложения в Linux.*

Ввести массив вещественных чисел размером N. Найти его наибольший и наименьший элементы и поменять их местами. Найти сумму и произведение всех элементов массива: в одномерном массиве, состоящем из вещественных элементов, вычислить: количество элементов массива, больших заданного числа и произведение элементов массива, расположенных после максимального по модулю элемент.

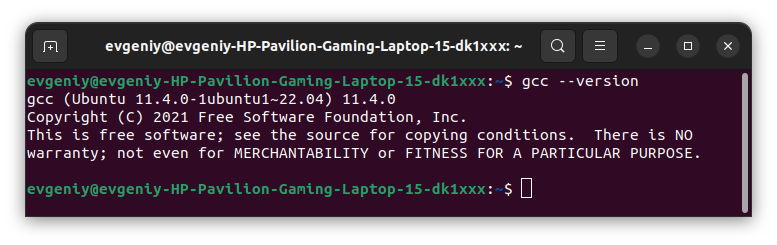
**Упражнение 2.**

**Изучите и выполните требования, представленные в лабораторной работе №3 «Многофайловые проекты и шаблон MVC». Разработайте исходные коды приложений в соответствии с заданиями. Компиляцию приложений проведите с помощью консольных команд**

***Задание лабораторной работы №3****. Многофайловые проекты и шаблон MVC.* Разработать интерактивную программу «TrytoGuesstheNumber» («Попробуй угадать число»), которая эмулирует классическую игру на отгадывание числа. Суть программы (игры) сводиться к следующему: компьютер генерирует случайное число из диапазона, к примеру, от 1 до 100, а пользователь (далее игрок) пытается отгадать число за минимальное количество попыток. При каждой очередной попытке компьютер «подсказывает» игроку, как соизмеряется предложенный вариант игрока с действительным загаданным компьютером числом: загаданное число больше или меньше указанного (higher/lower). Как только игрок отгадывает число, компьютер должен «поздравить» его с выводом на экран угаданного числа и количество затраченных игроком попыток. Далее компьютер может «предложить» повторно сыграть игру или выйти из программы. Для универсальности предложенной программы можно добавить возможность выбора диапазона генерирования компьютером случайных чисел, а также задания ограничения на количество попыток. В случае, если игрок не укладывается в заданное количество попыток (т.е. проигрывает), программа должна выводить суровую надпись «GameOver».

**Выполнение упражнений:**

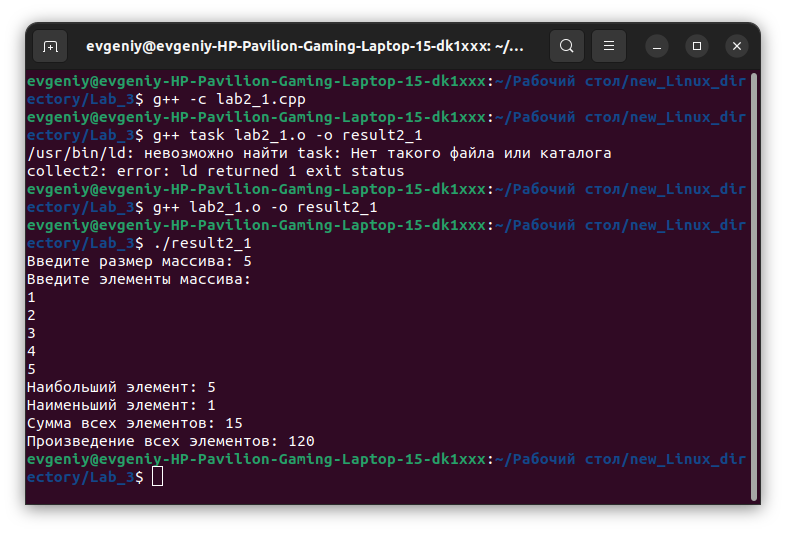
**Версия gcc**



**Задание лабораторной №2.**

**Основное**

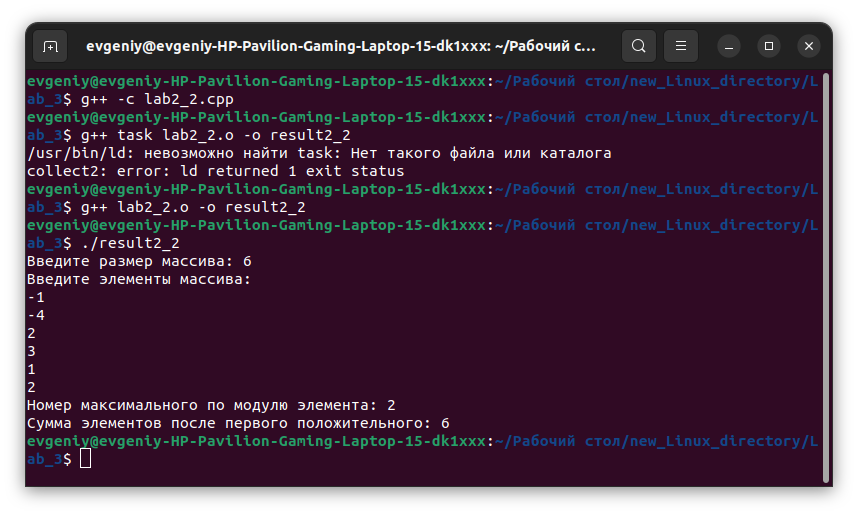
Ввести массив вещественных чисел размером N. Найти его наибольший и наименьший элементы и поменять их местами. Найти сумму и произведение всех элементов массива.



**Скрипт к заданию:**



**6 вариант:** В одномерном массиве, состоящем из n вещественных элементов, вычислить: номер максимального по модулю элемента массива и сумму элементов массива, расположенных после первого положительного элемента.

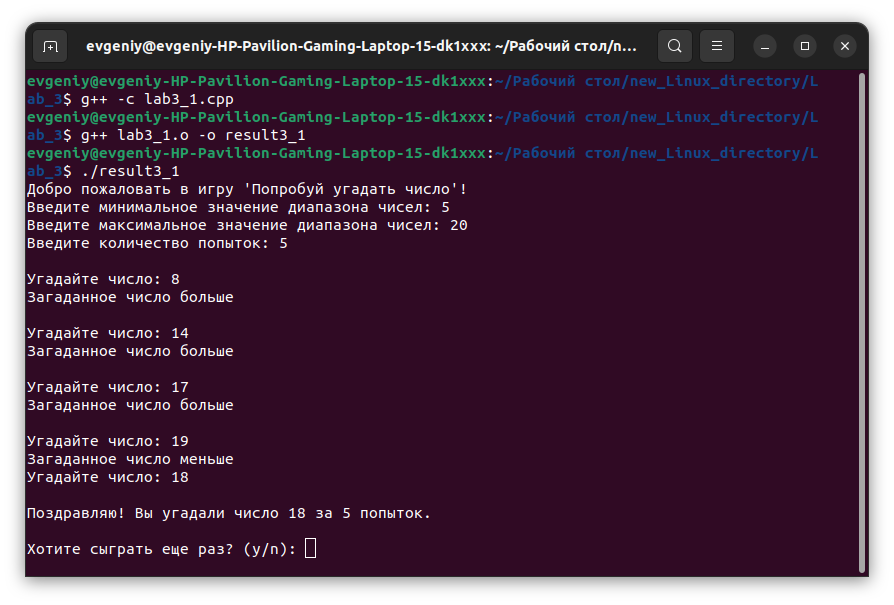


**Скрипт к заданию:**

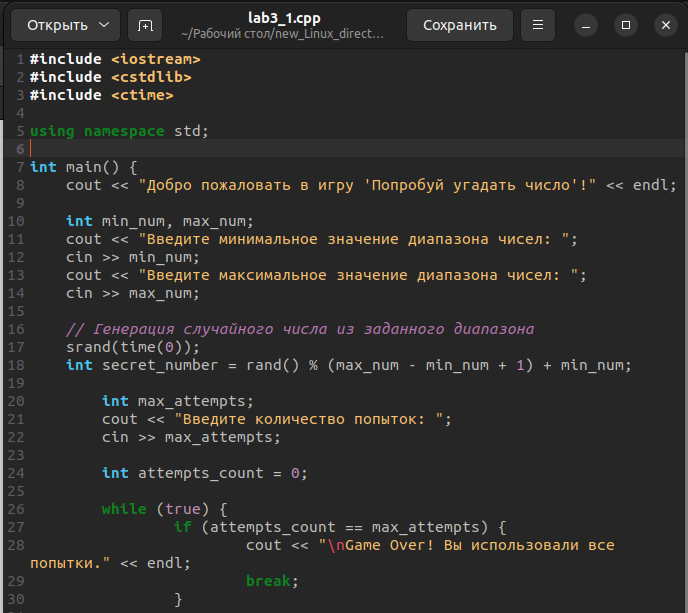


**Задание лабораторной №3.**

Игра «Угадай число»:



**Скрипт к заданию:**

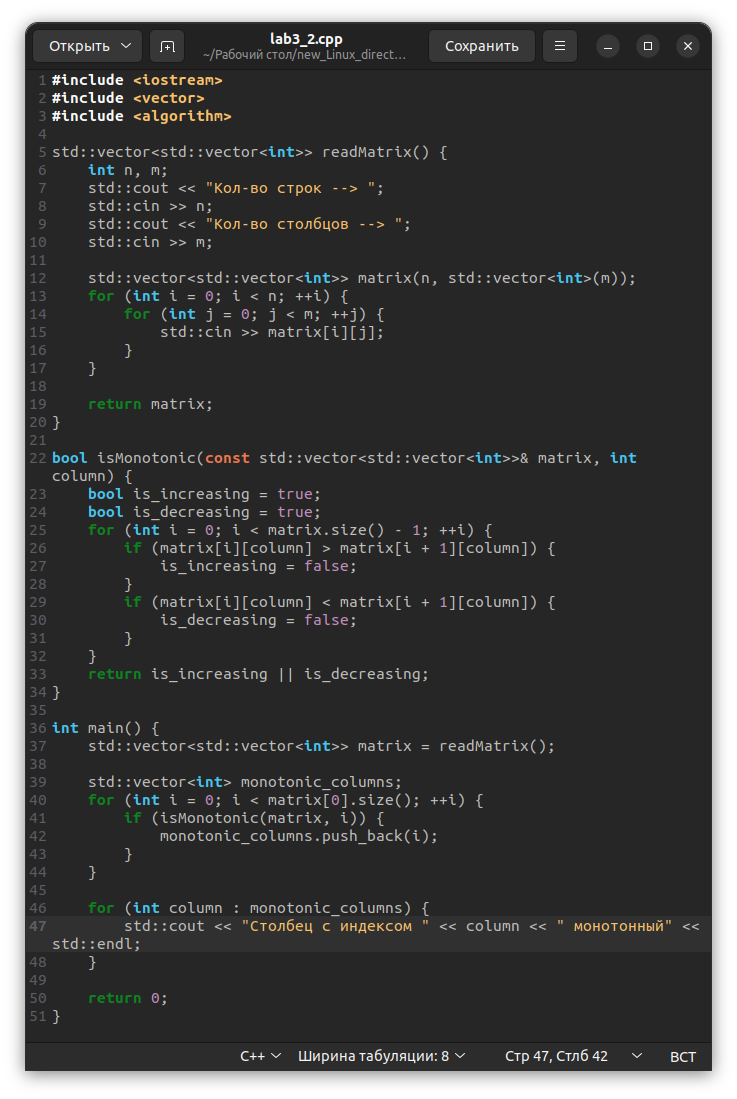


****

**6 вариант:** Выведите номера столбцов, элементы каждой из которых образуют монотонную последовательность (монотонно убывающую или монотонно возрастающую). элементов массива, расположенных после первого положительного элемента.

****

**Скрипт к заданию:**

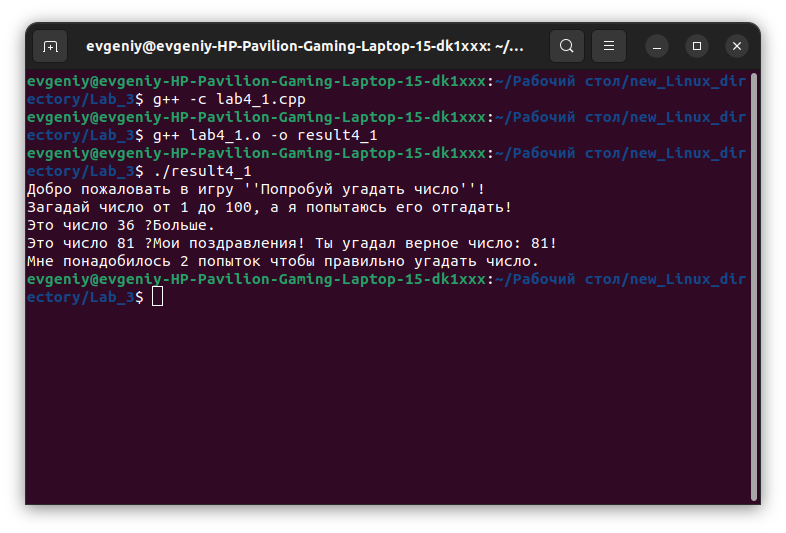
****

**Задание лабораторной №4.**

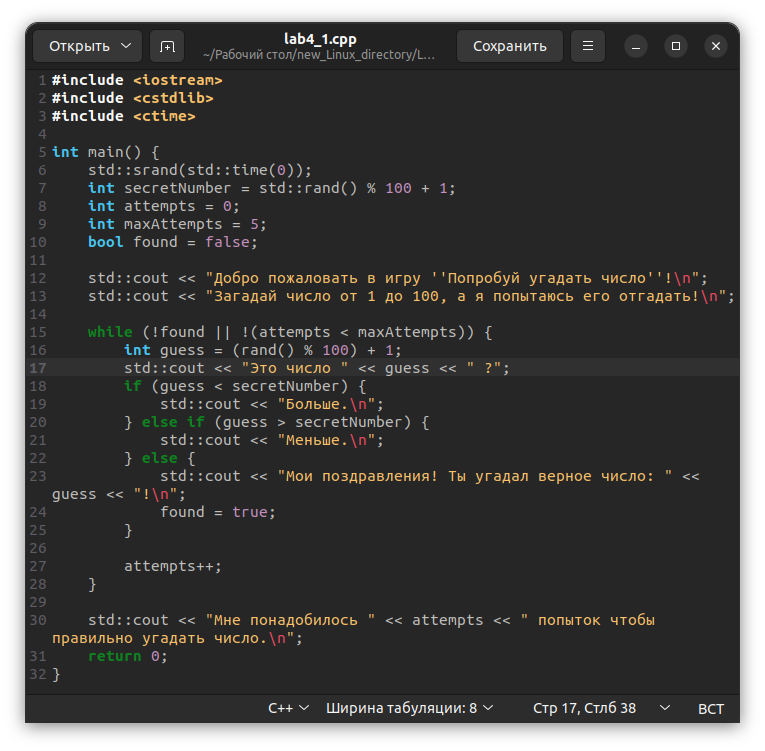
Необходимо переработать основное задание из предыдущей лабораторной

работы таким образом, чтобы пользователь (игрок) загадывал число, а компь-

ютер, используя оптимальный и эффективный алгоритм, его отгадывал.



**Скрипт к заданию:**

****

**Задания:**

**Задание 1**

Используя материалы (исходные файлы) упражнения №1 проведите компиляцию однофайлового проекта с прохождением всех стадий комппиляции.

Для ускорения работы примените скрипты bash.

Исходные файлы программ обязательно должны содержать коментарии.

pdfСборка проекта должна содержать файлы с результатами препроцессинга. Исследуйте фалы препроцессора, найдите в них код своей программы.

Определите размеры исходных, препроцессорных, ассемблерных, объектных

и исполняемых файлов. С помощью соответствующий консольных команд

определите форматы этих файлов. Результаты подтвердите скриншотами.

**Задание 2**

Используя материалы (исходные файлы) упражнения №1 и №2 создайте один

многофайловый проект, руководствуясь принципом Single Responsibility

Principle. Каждое задание должно быть представлено отдельным исходным

файлом. Для связывания файлов проекта обязательно используйте заголовочные файлы. Все файлы могут располагаться в одной директории.

Все задания должны вызываться консольным меню. Для выхода из приложения предусмотрите отдельный параметр.

Проведите автосборку проект с прохождением всех стадий компиляции с использованием утилиты make.

**Задание 3**

Используя материалы (исходные файлы) задания №2 создайте один многофайловый проект, руководствуясь принципом Single Responsibility Principle.

Каждое задание должно быть представлено отдельным исходным файлом. Используйте архитектурный шаблон проектирования MVC. Здесь под шаблоном

MVC понимается файлы или группа файлов в трех отдельных директориях

(папках), а не просто три отдельных файла.

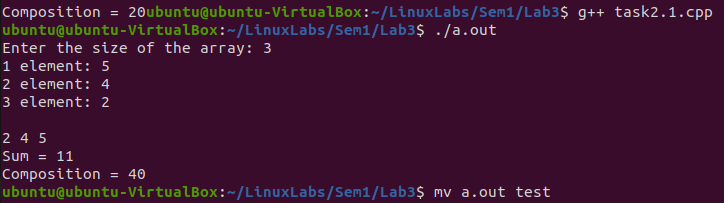
Все задания должны вызываться консольным меню. Для выхода из приложения предусмотрите отдельный параметр. Для связывания файлов проекта обязательно используйте заголовочные файлы.

Проведите рекурсивную автосборку проект с прохождением всех стадий компиляции с использованием утилиты make

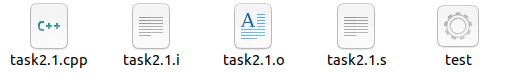
**Выполнение заданий:**

**Задание 1.**

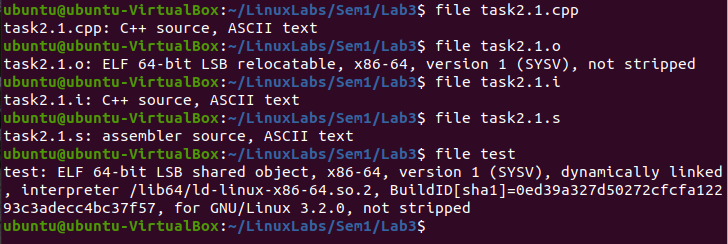
Компиляция однофайлового проекта



Демонстрация того, что был автоматически создан файл a.out и потом переименован в test:

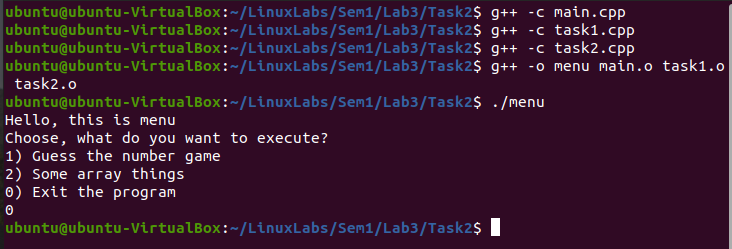


Формат файлов и размеры:

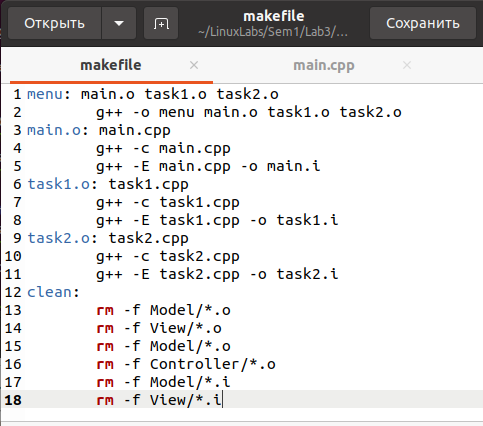


**Задание 2.**

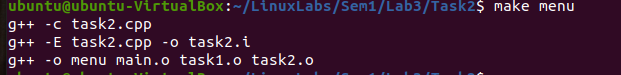
Создание многофайлового проекта.



Содержание Makefile’а:

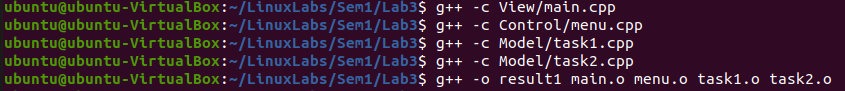


Результат использование утилиты make:

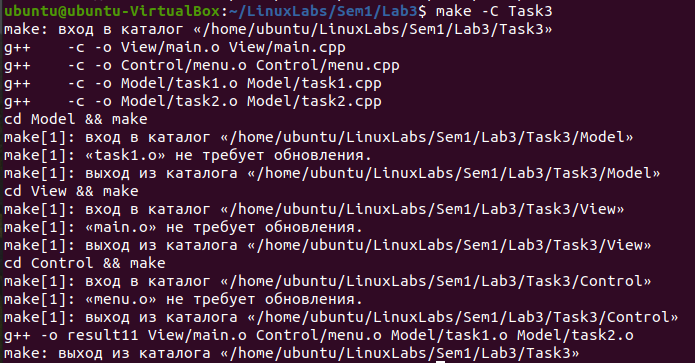


**Задание 3.**

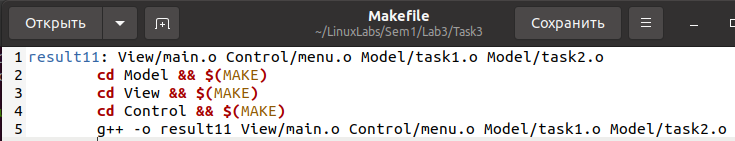
Создание многофайлового проекта с использованием архитектурного шаблона проектирования MVC



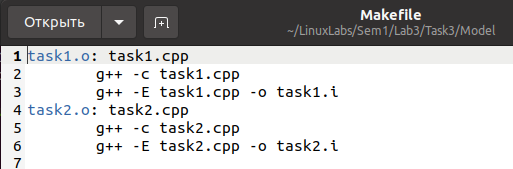
Рекурсивная автосборка проекта с прохождением всех стадий компиляции с использованием утилиты make



Содержание основного Makefile’а:



Makefile для Model (аналогично для Controller & View):



**Контрольные вопросы:**

**1) Что такое компоновка (линковка, сборка)?**

Компоновка (линковка, сборка) - это процесс объединения нескольких объектных файлов (object files) вместе для создания исполняемого файла или библиотеки.

**2) Что такое рефакторинг кода?**

Рефакторинг кода - это процесс изменения структуры и внутреннего устройства программного кода с целью улучшить его качество, читаемость, поддерживаемость и эффективность, не меняя при этом его внешнего поведения.

**3) Опишите базовый синтаксис make-файла и общие правила определения целевых связок в make-файле: целей (targets), зависимостей (dependencies) и инструкций (instructions)? Какая цель будет обрабатываться самой первой при запуске утилиты make? Что такое «основная цель сборки» и что в неё входит?**

**5) Назовите основные стадии компиляции программы?**

Процесс компиляции состоит из следующих этапов:

1. Лексический анализ.

2. Синтаксический анализ.

3. Семантический анализ.

4. Оптимизация.

5. Ассемблирование.

6. Генерация кода.

7. Связывание.

**6) В чем различие компиляции и интерпритации программы?**

Интерпретатор берет одну инструкцию, транслирует и выполняет ее, а затем берет следующую инструкцию. Компилятор же транслирует всю программу сразу, а потом выполняет ее

**7) Приведите известные вам примеры компилируемых и интерпритируемых языков программирования**

Примерами компилируемых языков являются Pascal, C, C++

Примерами интерпретируемых языков являются PHP, Python, JavaScript

**8) Что такое препроцессор и зачем он нужен?**

Препроцессор — это компьютерная программа, принимающая данные на входе и выдающая данные, предназначенные для входа другой программы

**9) В чем разница между понятиями «аргумент функции» и «параметр функции» ?**

Параметром (формальным параметром) функции называется переменная в функции, которая будет содержать передаваемое снаружи входное значение. Аргументом называют фактическую переменную или выражение, значение которого используется как входное значение при вызове функции

**10) В чем разница использования двух форм директивы: #include <filename> или #include “filename”?**

Для #include <filename> препроцессор выполняет поиск в зависимости от реализации. Для #include "filename" препроцессор сначала выполняет поиск в том же каталоге, что и файл, содержащий директиву, а затем следует по пути поиска

**11) Что такое автосборка? Какие приемущества она дает?**

Сборкой называется процесс подготовки программы к непосредственному использованию. Собирать программы вручную неудобно, поэтому программисты, как правило, прибегают к различным приемам, позволяющим автоматизировать этот процесс. Самый простой способ – написать сценарий оболочки (shell-скрипт), который будет автоматически выполнять все то, что вы обычно вводите вручную

**12) Какой командой можно определить определить формат файла?**

**13) Что такое утилита make?**

Утилита make — наиболее популярное и проверенное временем средство автоматической сборки программ в Linux

**14) В чем разница автосборки скриптами bash и утилитой make?**

make — утилита предназначенная для автоматизации преобразования файлов из одной формы в другую. Bash позволяет создавать сценарии для автоматизации различных операций

**Вывод:** в данной лабораторной работе были изучены встроенный инструментарий для разработки приложений под семейство ОС Linux и этапы такие, как проектирование, компиляция, отладка и запуск.