Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 4**

Дисциплина: Низкоуровневое программирование.

Тема: раздельная компиляция.

Выполнил студент гр. 3530901/10003 \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Л. М. Жилкина

(подпись)

Принял старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. А. Коренев

(подпись)

“ ” 2022 г.

Санкт-Петербург

2022

Оглавление.

ТЗ……………………………………………………………………………...3 стр.

Метод решения………………………………………………………………3 стр.

Листинг программ………………………………………………………...…3 стр.

Препроцессирование………………………………………………………...4 стр.

Компиляция………………………………………………………....………..6 стр.

Ассемблирование……………………………………………………………8 стр.

Компоновка…..…………………………………………………………..…13 стр.

Создание статической библиотеки и make-файлов…………..……..……16 стр.

Вывод……………………………………………………………………..…18 стр.

**Формулировка задачи.**

1) На языке C разработать функцию, реализующую определенную вариантом задания функциональность. Поместить определение функции в отдельный исходный файл, оформить заголовочный файл. Разработать тестовую программу на языке C.

2) Собрать программу «по шагам». Проанализировать выход препроцессора и компилятора. Проанализировать состав и содержимое секций, таблицы символов, таблицы перемещений и отладочную информацию, содержащуюся в объектных файлах исполняемом файле.

3) Выделить разработанную функцию в статическую библиотеку. Разработать make-файлы для сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

Вариант задания:найти максимальный элемент массива чисел (вариант 12).

**Метод решения**.

Будем итерироваться по массиву и сравнивать очередной элемент с максимальным найденным, изначальное значение которого – нулевой элемент массива. Если очередной элемент больше максимального найденного, будем его сохранять.

**Листинг программ.**

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Файл main.c

Изображение выглядит как текст, снимок экрана, монитор, электроника

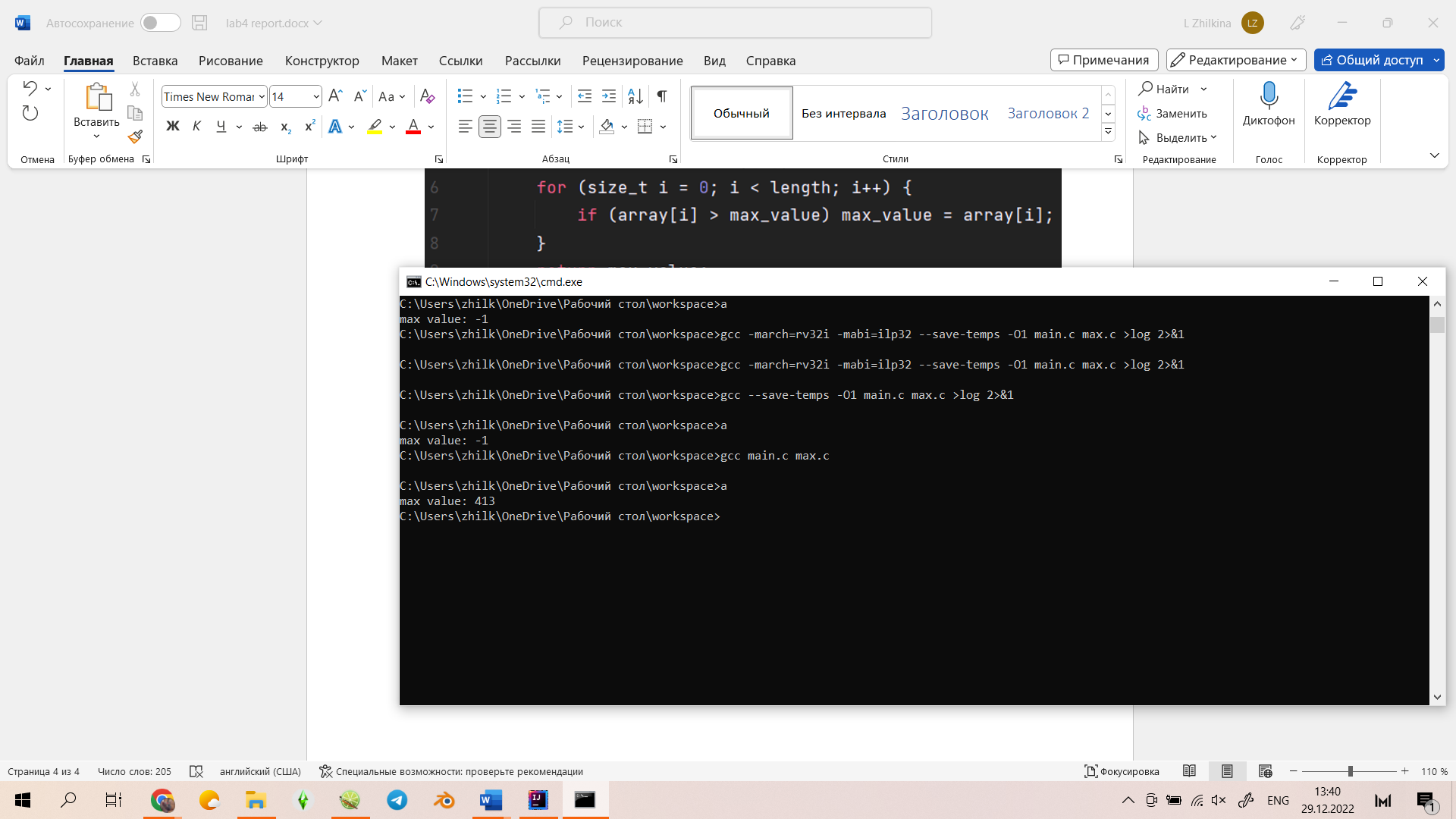
Автоматически созданное описание

Файл max.c

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Файл max.h



Результат работы программы

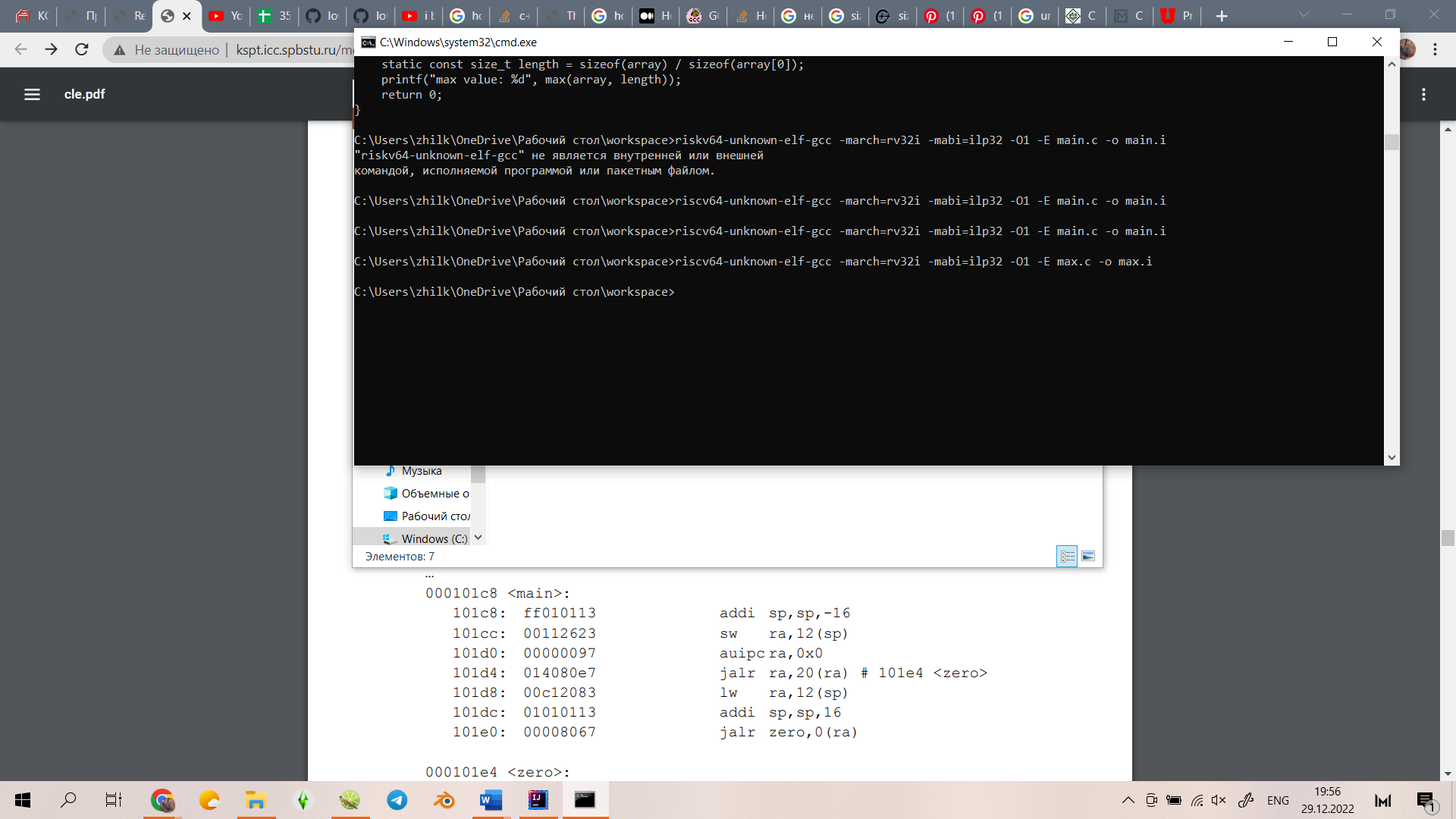
В файле “max.c” реализована функция max(), в которую передаётся массив чисел array[] и размер этого массива length.

Заголовочный файл “max.h” содержит в себе определение функции max().

В файле “main.c” мы подключаем заголовочный файл “max.h” и используем реализацию функции из “max.c”.

**Препроцессирование.**

Начнем сборку программы по шагам. Произведём препроцессирование файлов «main.c» и «max.c» в файлы “main.i” и “max.i” с помощью следующих команд:



Поскольку main.c содержит заголовочный файл стандартной библиотеки языка С stdio.h, результат препроцессирования этого файла имеет много добавочных строк.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Фрагменты файла main.i

Изображение выглядит как текст

Автоматически созданное описание

Файл max.i

**Компиляция.**

Скомпилируем файлы main.i и max.i в файлы main.s и max.s. Выполним команды:

Изображение выглядит как текст, оранжевый, снимок экрана, набор

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Файл main.s

Изображение выглядит как текст

Автоматически созданное описание

Файл max.s

Результатом работы компилятора стал код ассемблера RISC-V для соответствующих программ.

**Ассемблирование.**

Выполним ассемблирование файлов main.s и max.s в файлы main.o и max.o. Выполним команды:

Изображение выглядит как текст, оранжевый, снимок экрана

Автоматически созданное описание

Результат работы ассемблера – бинарные файлы, просто так их содержимое не прочесть. Воспользуемся утилитой objdump:

Изображение выглядит как текст

Автоматически созданное описание

Вывод информации о файлах main.o и max.o

Оба файла формата ELF, содержат символы (флаг HAS\_SYMS) и таблицу перемещений (флаг HAS\_RELOC). Объектные файлы не содержат адресов точек входа, однако поскольку соответствующее поле присутствует в заголовке файла формата ELF, оно заполняется 0.

Теперь посмотрим на секции заголовков и таблицы символов файлов.

Изображение выглядит как текст

Автоматически созданное описание

Секция заголовков файла main.o

Изображение выглядит как текст

Автоматически созданное описание

Секция заголовков файла max.o

Изображение выглядит как текст

Автоматически созданное описание

Таблица символов файла main.o

Изображение выглядит как текст

Автоматически созданное описание

Таблица символов файла max.o

Обе таблицы содержат 1 глобальный символ (флаг g) функции (флаг F) – main и max. Таблица для “main.o” содержит также 2 неопределенных символа (UND): max и printf. Эти символы использовались в ассемблерном коде, из которого был получен данный объектный фал, но не были определены. Значит, символы должны быть определены где-то еще, что и отобразилось в таблице.

Посмотрим на секции “.text”:

Изображение выглядит как текст

Автоматически созданное описание

Содержимое секции “.text” файла main.o и результат дизассемблирования

Изображение выглядит как текст

Автоматически созданное описание

Содержимое секции “.text” файла max.o и результат дизассемблирования

Изучим содержимое секций “.data” и “.bss”:

Изображение выглядит как текст

Автоматически созданное описание

Секция “.data” обоих файлов

Изображение выглядит как текст

Автоматически созданное описание

Секция “.bss” обоих файлов

Секция “.data” файла main.o содержит массив чисел, у файла max.o она пуста. Секции “.bss” обоих файлов пусты.

Изучим содержимое секции “.comment”:

Изображение выглядит как текст

Автоматически созданное описание

Секция “.comment” обоих файлов

Секция “.comment” – секция данных о версиях – для обоих файлов содержит одни и те же значения – сведения о GCC от SiFive версии 8.3.0.

Изучим содержимое секции “.riscv.attributes”:

Изображение выглядит как текст

Автоматически созданное описание

Секция “.riscv.attributes ” обоих файлов

Секция “.riscv.attributes” обоих файлов содержит одну и ту же информацию об используемой архитектуре команд RV32I.

Изучим таблицы перемещений:

Изображение выглядит как текст

Автоматически созданное описание

Таблицы перемещений обоих файлов

**Компоновка.**

Произведём компоновку и формирование исполняемых фалов программ. Выполним команду:



Сформированный компоновщиком файл - “a.out”. Посмотрим на его содержимое:

Изображение выглядит как текст

Автоматически созданное описание

Содержимое файла a.out

Флаг EXEC\_P указывает, что файл является исполняемым. После загрузки его выполнение должно начаться с адреса 0x00010090. Изображение выглядит как текст

Автоматически созданное описание

Секция заголовков файла a.out

В исполняемом файле производится слияние содержания секций обоих объектных файлов, а также значительное расширение списка секций новыми блоками.

Изображение выглядит как текст

Автоматически созданное описание

Фрагмент таблицы символов файла a.out

Таблица символов содержит множество дополнительных вхождений, однако определены все нужные секции, метки и адреса. Функции max и printf также помечены флагом F, но в отличие от стадии ассемблирования, являются определенными и содержатся по корректным адресам для успешного вызова этих функций из других участков программ.

Изображение выглядит как текст

Автоматически созданное описание

Таблица перемещений файла a.out

Таблица перемещений оказывается пуста, все необходимое было проведено компоновщиком.

Итог сборки программ на языке C по шагам - исполняемый на процессорах архитектуры RISC-V файл.

**Формирование статической библиотеки, разработка make-файлов для сборки библиотеки.**

Статическая библиотека – набор объектных файлов, среди которых компоновщик выбирает необходимые для данной программы. Функция поиска максимального элемента массива содержится в объектном файле max.o. Выделим этот файл в статическую библиотеку.

Изображение выглядит как текст

Автоматически созданное описание

Сборка библиотеки maxLib.a

Кодом “T” обозначаются символы, определенные в соответствующем объектном файле, кодом “U” - внешние символы. Как видно, в данном случае внешних символов нет. Символ функции max - основный символ, определяемым в этом объектном файле, остальные символы определяют лишь локальные для этого файла метки.

Соберем исполняемый файл, используя получившуюся библиотеку:

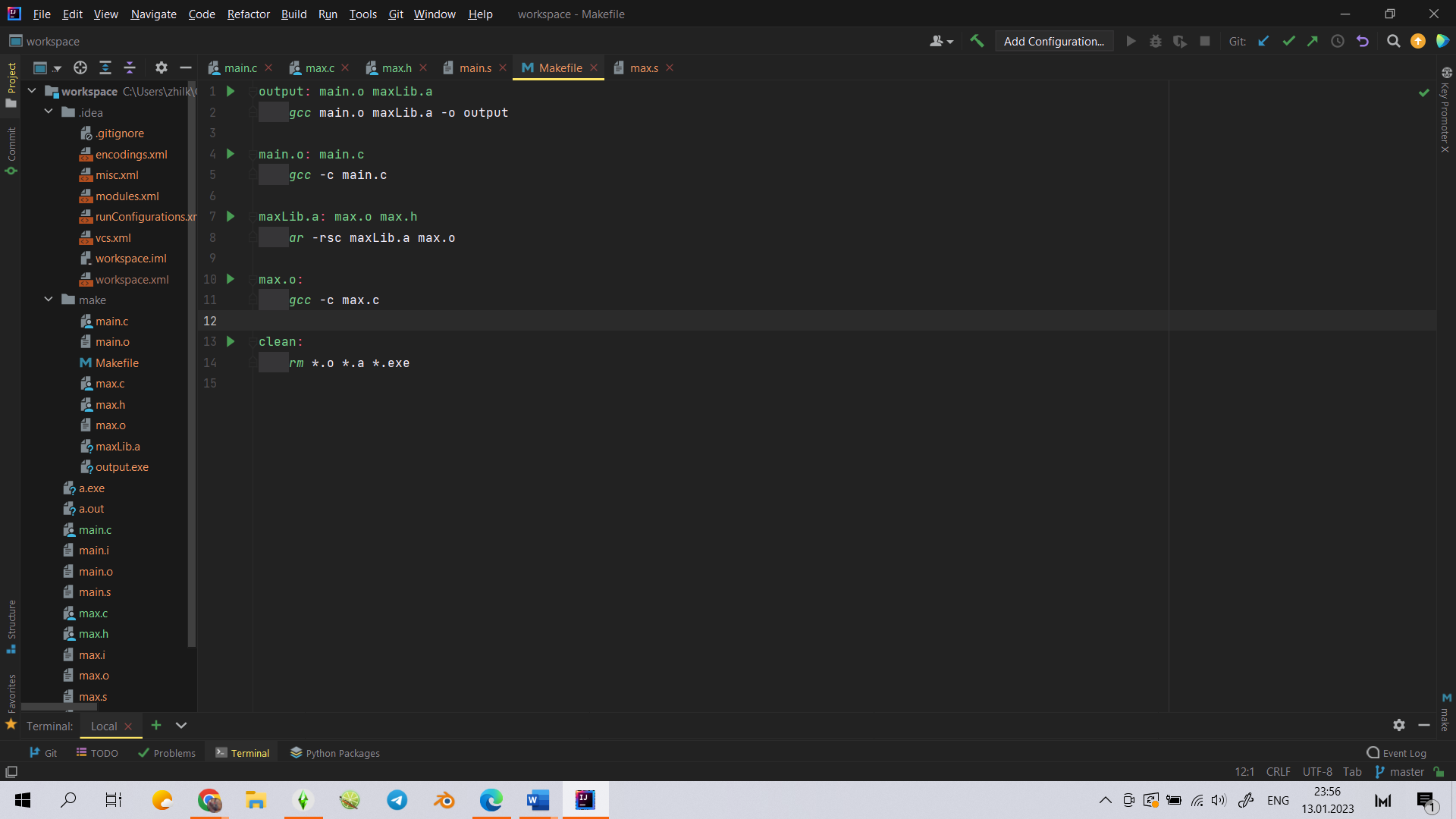


Изображение выглядит как текст

Автоматически созданное описание

Фрагмент таблицы символов файла a.out

Процесс выполнения команд выше можно заменить make-файлами, которые произведут создание библиотеки и сборку программы. Makefile - это набор инструкций для программы make, которая позволяет собирать проекты, состоящие из большого числа “\*.c” и “\*.h” файлов. Обычно эта программа используется в связке с системами сборки, например cmake, позволяя вести проекты модульно (т. е. проект с включенными подпроектами).



Текст файла Makefile

**Вывод.**

В данной лабораторной работе мы познакомились с процессом сборки проекта на языке C.

Он состоит из:

1. **Препроцессирования**: исходного *.c* файл препроцессируем в *.i* файл
2. **Компиляции**: полученный *.i* файл компилируется в файл ассемблера *.s*
3. **Ассемблирования**: файл *.s* ассемблируется в объектный файл *.o*
4. **Компоновки**: объектный файл *.o* компонуется в исполняемый файл

Также мы ознакомились в *makefile’ами,* которые упрощают процесс сборки.