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: «Автосборка, сборка приложений »

Выполнила: ст. гр. 10702221

Багапов Д.М.

Матвейчук К.А.

Яцук З.Р.

Проверил: Давыденко Н.В.

Минск 2023

**ЛАБОРАТОРНАЯ РАБОТА #7**

**Компиляция и отладка простейшего приложения в Linux**

**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++.

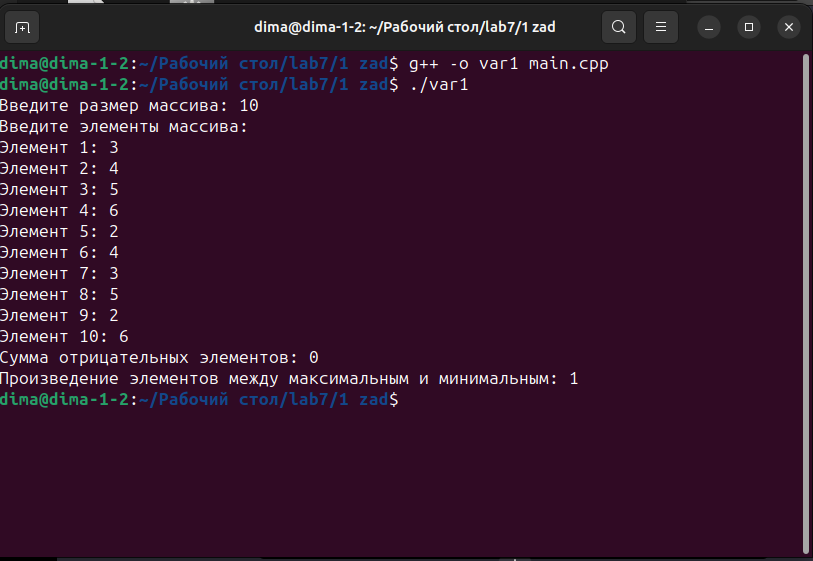
**Индивидуальные задания первой лабораторной:**

1) В одномерном массиве, состоящем из n вещественных элементов, вычислить: сумму отрицательных элементов массива и произведение элементов массива, расположенных между максимальным и минимальным элементами.

2) В одномерном массиве, состоящем из n целых элементов, вычислить: произведение элементов массива с четными номерами и сумму элементов массива, расположенных между первым и последним нулевыми элементами.

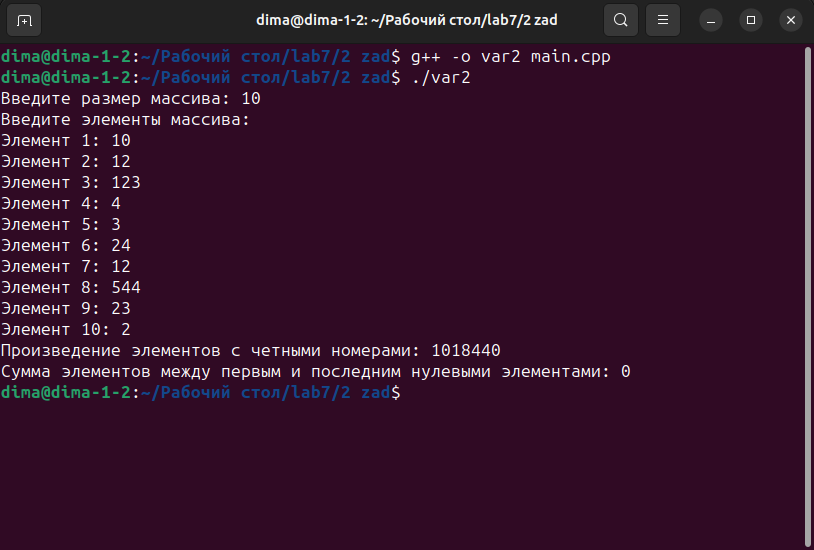
3) В одномерном массиве, состоящем из n вещественных элементов, вычислить: максимальный элемент массива и сумму элементов массива, расположенных до последнего положительного элемента.

1.



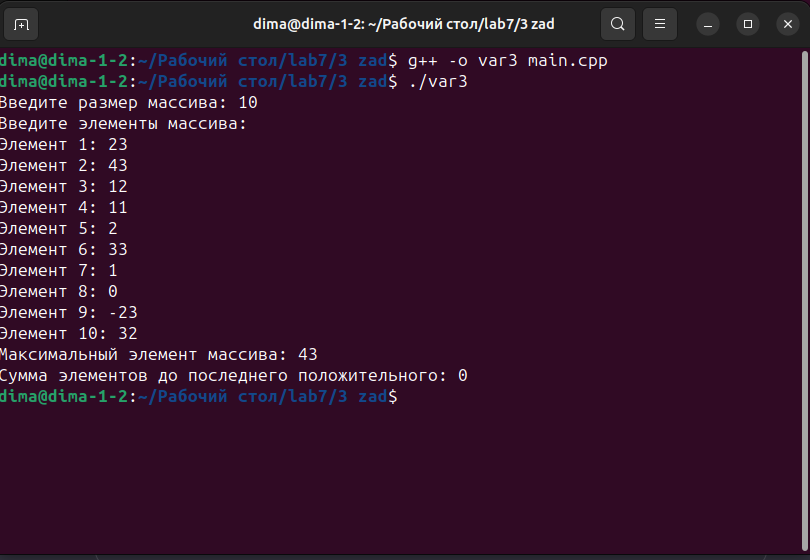
Успешное выполнение

2.



Успешное выполнение

3.



Успешное выполнение

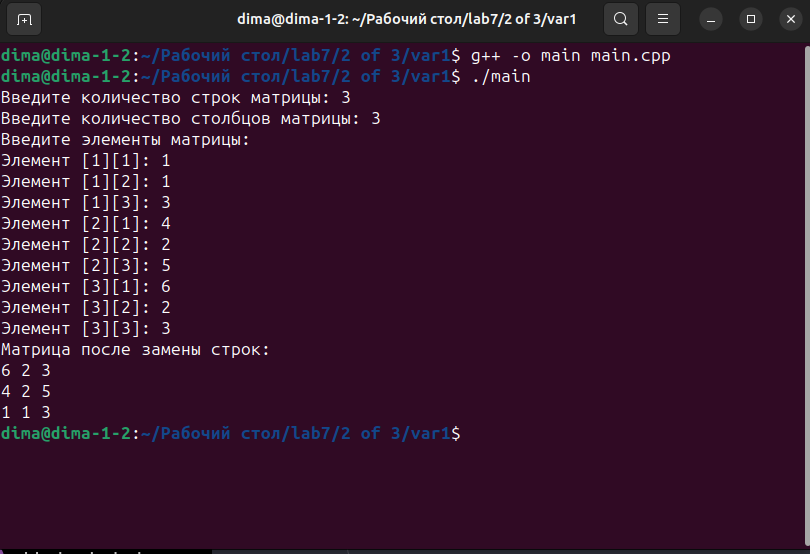
**Индивидуальные задания второй лабораторной:**

1) Поменять местами строку, содержащую элемент с наибольшим значением в матрице со строкой, содержащей элемент с наименьшим значением. Вывести на экран полученную матрицу. Для каждой строки с нулевым элементом на главной диагонали вывести ее номер и значение наибольшего из элементов этой строки.

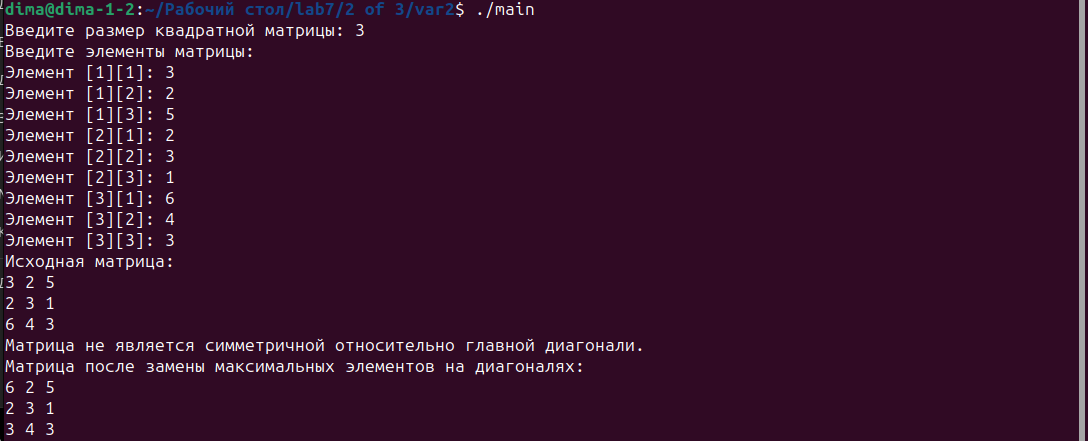
2) Определить, является ли матрица симметричной (относительно главной диагонали). Вывести на экран соответствующее сообщение. Найти максимальный элемент среди стоящих на главной и побочной диагонали и поменять местами с элементом, стоящим на пересечении этих диагоналей.

3) Выведите номера столбцов, все элементы, которых четны. Для каждого столбца с отрицательным элементом на главной диагонали вывести его номер и сумму всех элементов этого столбца.

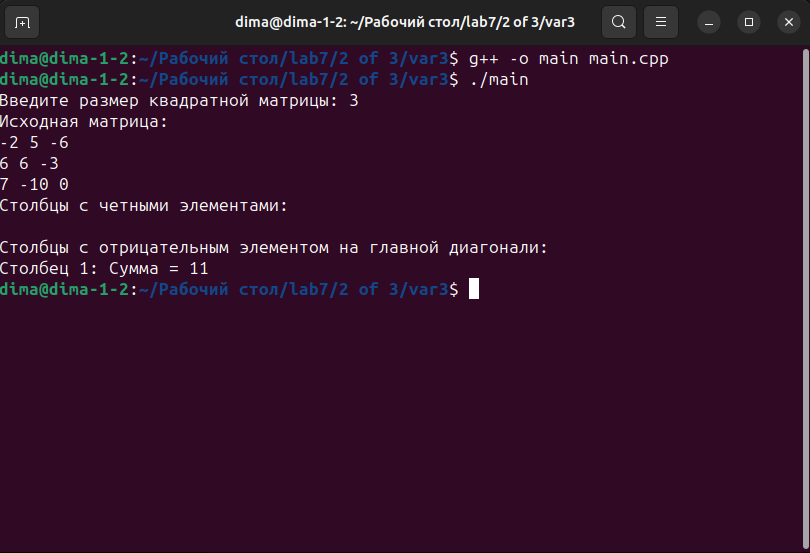
1.



**2.**

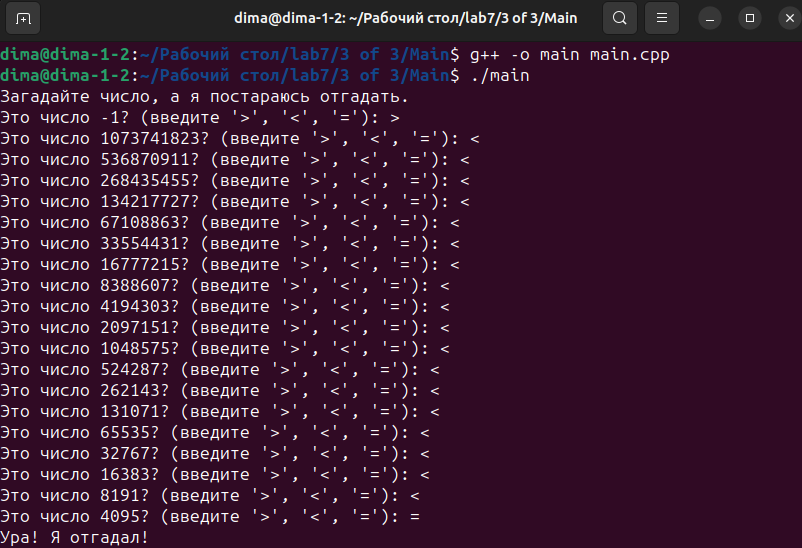


3.



**Основное задание третьей лабораторной:**

Необходимо переработать основное задание из предыдущей лабораторной работы таким образом, чтобы пользователь (игрок) загадывал число, а компьютер, используя оптимальный и эффективный алгоритм, его отгадывал.



**Индивидуальные задания третьей лабораторной:**Для программ, которые были разработаны в двух предыдущих лабораторных

работах №2 и №3 необходимо добавить автоматизацию сборки многофайлового проекта с использованием сценарных оболочек и автосборщиков.

**СДЕЛАНО**

**Контрольный вопросы:**

**1. Способы и средства для автоматической сборки многофайловых проектов:**

- Make: Универсальный инструмент для сборки проектов. Make использует make-файлы, где описываются правила сборки.

- CMake: Кроссплатформенная утилита для автоматизации сборки проектов. Создает make-файлы или файлы проекта для различных сред разработки (Visual Studio, Xcode).

- Autotools (Autoconf, Automake): Набор инструментов для создания конфигурационных скриптов, make-файлов и т.д.

- SCons: Система сборки на Python, автоматически обнаруживает зависимости.

- Bazel: Инструмент сборки, разработанный Google, подходит для больших проектов.

**2. Сборка и запуск проекта с использованием bash:**

- Для сборки и запуска проекта с использованием bash, вы можете написать скрипт, который вызывает необходимые команды компиляции и выполнения. Например:

```bash

#!/bin/bash

g++ -o my\_program main.cpp

./my\_program

```

**3. Утилита GNU make (автосборщик):**

- `make` - это инструмент автоматической сборки, который использует make-файлы для определения правил сборки.

**4. Make-файл:**

- `Makefile` - это текстовый файл, который содержит правила для сборки проекта. Включает в себя цели (targets), зависимости (dependencies), и инструкции (instructions) для компиляции и линковки программы.

**5. Алгоритм процесса автоматической сборки программы:**

- 1. Определение зависимостей между файлами.

- 2. Проверка временных меток (timestamps) файлов.

- 3. Перекомпиляция только необходимых файлов.

- 4. Линковка объектных файлов в исполняемый файл.

**6. Синтаксис make-файла:**

- Цель (target): `target: dependencies`

- Инструкция (instruction): ` command`

- Пример:

```make

my\_program: main.cpp

g++ -o my\_program main.cpp

```

**7. Имя make-файла и вызов утилиты make:**

- Стандартное имя: `Makefile` или `makefile`.

- Если используется нестандартное имя: `make -f имя\_файла`.

**8. Параметризация процесса сборки и переменные make-файла:**

- Использование переменных и параметров для упрощения и параметризации сценария сборки. Например:

```make

CXX = g++

CXXFLAGS = -std=c++11 -Wall

my\_program: main.cpp

$(CXX) $(CXXFLAGS) -o my\_program main.cpp

```

**9. Рекурсивный вызов make:**

- Рекурсивный вызов make используется, когда нужно вызвать make в поддиректории проекта. Это может быть полезно для сборки подпроектов.

**10. Избежание перекомпиляции:**

- Правильно установленные зависимости в make-файле помогают избежать перекомпиляции. Make проверяет временные метки файлов и перекомпилирует только те, которые изменились.

**11. Генерация случайных значений:**

- Используйте функции `rand()` и `srand()` из стандартной библиотеки `cstdlib`. Пример:

```cpp

#include <cstdlib>

#include <ctime>

#include <iostream>

int main() {

std::srand(std::time(0)); // Инициализация генератора случайных чисел

int randomValue = std::rand() % 100; // Генерация случайного числа от 0 до 99

std::cout << "Случайное число: " << randomValue << std::endl;

return 0;

}

```

С помощью `srand()` можно установить начальное значение генератора случайных чисел, чтобы результаты были разными при каждом запуске программы.