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: ”Компиляция и отладка простейшего приложения в Linux ***”***

Выполнил**:** студент группы 10701321 Нестерков Д.А.

Принял**:** пр.Давыденко Н.

Минск 2023

# **Лабораторная работа № 7**

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++.

**Задания**

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

**Индивидуальное задание**

В одномерном массиве, состоящем из к целых элементов, вычислить: количество положительных элементов массива и сумму элементов массива, расположенных после последнего элемента, равного нулю.

**Выполнение лабораторной работы**

**Код программы**

#include <iostream>

#include <cmath>

int task1(){

// Vvod razmera massiva

int n;

std::cout << "Vvedite razmer massiva: ";

std::cin >> n;

// Dinamicheskoe vydelenie pamyati

double\* arr = new double[n];

// Vvod elementov massiva

std::cout << "Vvedite elementy massiva:\n";

for(int i = 0; i < n; ++i) {

std::cin >> arr[i];

}

// Poisk min i max

double min = arr[0];

double max = arr[0];

int min\_i = 0;

int max\_i = 0;

for(int i = 1; i < n; ++i) {

if(arr[i] < min) {

min = arr[i];

min\_i = i;

}

if(arr[i] > max) {

max = arr[i];

max\_i = i;

}

}

// Obmen min i max

double temp = arr[min\_i];

arr[min\_i] = arr[max\_i];

arr[max\_i] = temp;

// Vyvod rezul'tata

std::cout << "\nMassiv s zamenennymi min i max:\n";

for(int i = 0; i < n; ++i) {

std::cout << arr[i] << " ";

}

// Vychislenie summy i proizvedeniya

double sum = 0;

double mult = 1;

for(int i = 0; i < n; ++i) {

sum += arr[i];

mult \*= arr[i];

}

std::cout << "\nSumma elementov: " << sum;

std::cout << "\nProizvedenie elementov: " << mult;

// Osvobozhdenie pamyati

delete[] arr;

return 0;

}

int task2(){

int k;

std::cout << "Enter number of elements: ";

std::cin >> k;

int arr[k];

int positive\_count = 0;

int sum\_after\_zero = 0;

bool zero\_found = false;

for(int i = 0; i < k; ++i) {

std::cin >> arr[i];

if(arr[i] > 0) {

positive\_count++;

}

if(arr[i] == 0) {

zero\_found = true;

}

if(zero\_found && arr[i] != 0) {

sum\_after\_zero += arr[i];

}

}

std::cout << "Positive elements count: " << positive\_count << "\n";

std::cout << "Sum after last zero: " << sum\_after\_zero << "\n";

return 0;

}

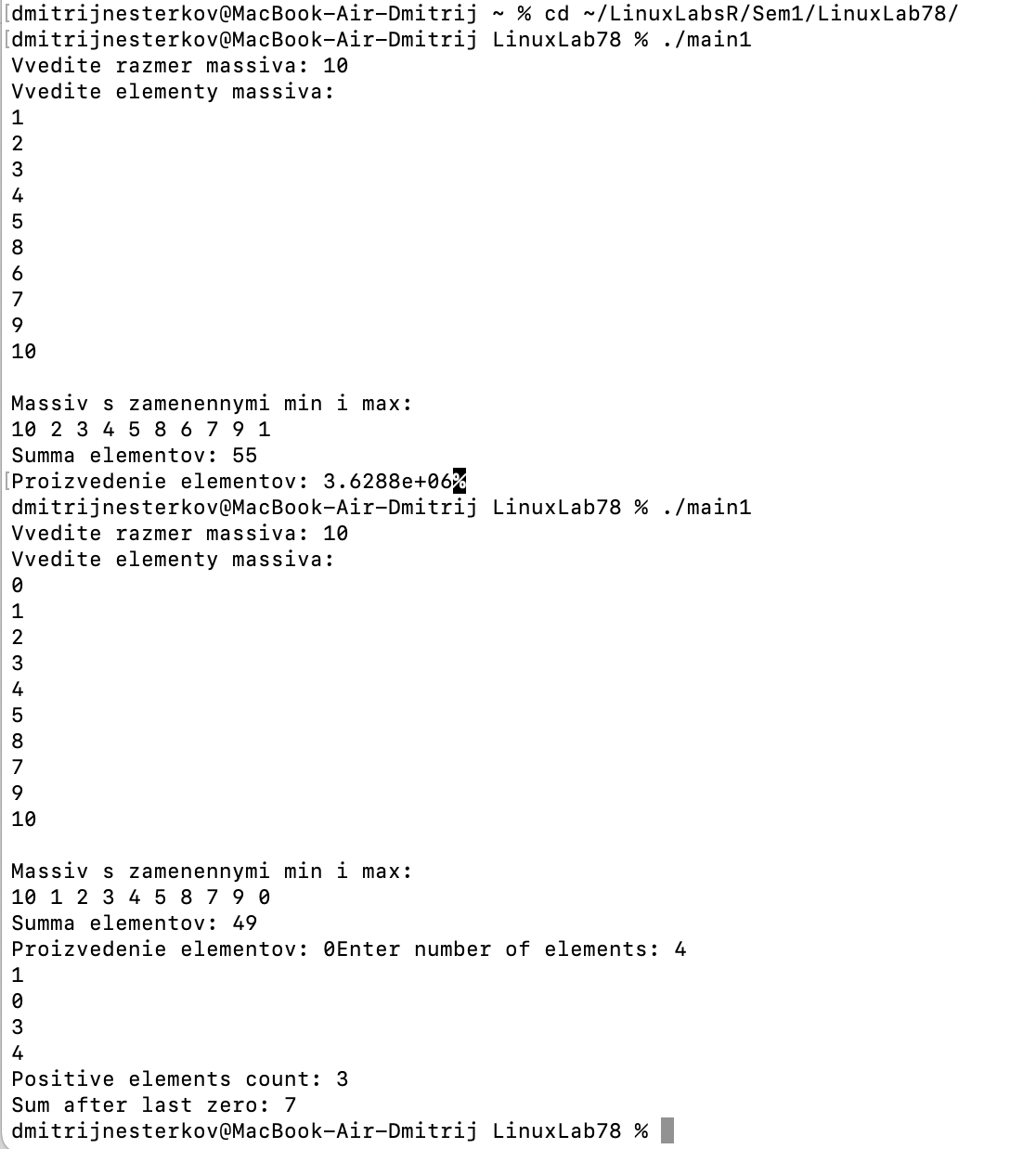
int main() {

task1();

task2();

return 0;

}

****

**Рисунок 1 результаты выполнения**

**Контрольные вопросы**

Что такое GCC? Опишите наиболее востребования опции компилятора gcc/g++. Чем отличается gcc и g++?

**GCC (GNU Compiler Collection) - это набор компиляторов, включающий в себя:**

**- gcc - компилятор языка Си**

**- g++ - компилятор языка C++**

**- и другие компиляторы для Fortran, Java, Ada и др. языков**

**Отличия gcc и g++:**

**- gcc используется для компиляции кода на Си, g++ - для C++**

**- g++ подключает стандартную библиотеку C++ и добавляет дополнительные возможности, необходимые для C++**

**- Команды компиляции и линковки немного различаются для C и C++ кода**

**В целом, для компиляции C++ программ используется g++, для Си - gcc.**

**Задание**

Разработать интерактивную программу «Try to Guess the Number» («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 «Game Over».

**Индивидуальное задание**

Найти номер строки заданной матрицы, в которой находится самая длинная серия (последовательность одинаковых элементов).

Код программы

#include <iostream>

#include <cstdlib>

#include <ctime>

void task2(){

int rows, cols;

std::cout << "Введите количество строк: ";

std::cin >> rows;

std::cout << "Введите количество столбцов: ";

std::cin >> cols;

int mat[rows][cols];

// Ввод матрицы

for(int i = 0; i < rows; i++) {

for(int j = 0; j < cols; j++) {

std::cin >> mat[i][j];

}

}

// Поиск строки с максимальной серией

int max\_row = 0;

int max\_streak = 0;

for(int i = 0; i < rows; i++) {

int curr\_streak = 1;

for(int j = 1; j < cols; j++) {

if(mat[i][j] == mat[i][j-1]) {

curr\_streak++;

} else {

curr\_streak = 1;

}

if(curr\_streak > max\_streak) {

max\_streak = curr\_streak;

max\_row = i;

}

}

}

std::cout << "Номер строки с максимальной серией: " << max\_row << "\n";

}

int main() {

srand(time(0)); // инициализация генератора случайных чисел

int lower\_bound = 1;

int upper\_bound = 100;

std::cout << "Введите нижнюю границу диапазона: ";

std::cin >> lower\_bound;

std::cout << "Введите верхнюю границу диапазона: ";

std::cin >> upper\_bound;

int max\_attempts = 10;

std::cout << "Введите максимальное кол-во попыток: ";

std::cin >> max\_attempts;

int number = rand() % (upper\_bound - lower\_bound + 1) + lower\_bound;

int attempts = 0;

while(attempts < max\_attempts) {

int guess;

std::cout << "Ваша догадка: ";

std::cin >> guess;

attempts++;

if(guess == number) {

std::cout << "Поздравляю, вы угадали число за " << attempts << " попыток!";

break;

}

else if(guess < number) {

std::cout << "Загаданное число больше!\n";

}

else {

std::cout << "Загаданное число меньше!\n";

}

}

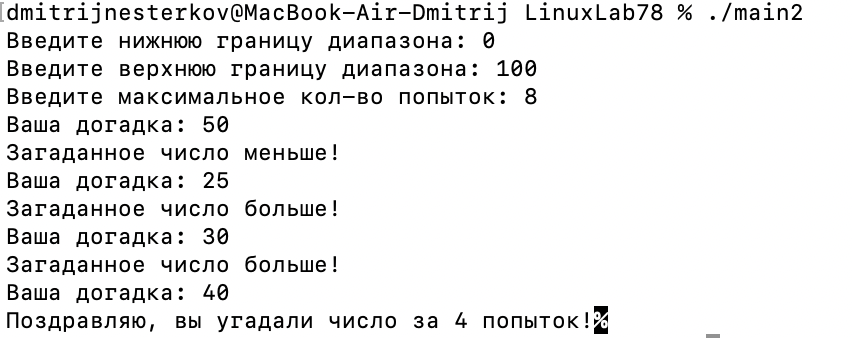
if(attempts == max\_attempts) {

std::cout << "Игра окончена. Вы проиграли.";

}

return 0;

}



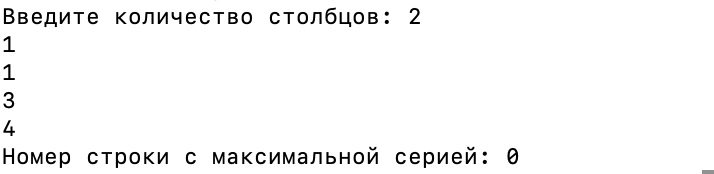


Рисунок 1-2 результаты

**Контрольные вопросы**

Объясните принцип "Единственной ответственности" (Single Responsibility Principle, SRP), который используется бля проектирования и реализации независимого или слабосвязанного кода.

Принцип единственной ответственности (Single Responsibility Principle) - это один из основных принципов проектирования кода, согласно которому каждый модуль, класс или функция должны нести ответственность только за одну вещь и решать только одну задачу.

Это позволяет сделать код:

- Понятным

- Гибким

- Поддерживаемым

**Задание**

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

**Индивидуальное задание**

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.

Код программы:

#include <iostream>

int main() {

int number;

std::cout << "Загадайте число от 1 до 100: ";

std::cin >> number;

int lower\_bound = 1;

int upper\_bound = 100;

while (lower\_bound <= upper\_bound) {

int guess = (lower\_bound + upper\_bound) / 2;

if (guess == number) {

std::cout << "Я угадал ваше число! Это " << guess << std::endl;

break;

}

else if (guess < number) {

std::cout << "(<)Мое число:" << guess << std::endl;

lower\_bound = guess + 1;

}

else {

std::cout << "(>)Мое число: " << guess << std::endl;

upper\_bound = guess - 1;

}

}

return 0;

}

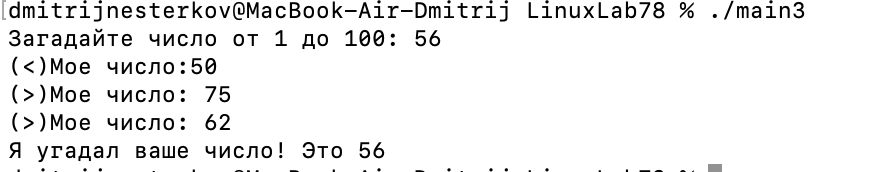


Рисунок 1 результат

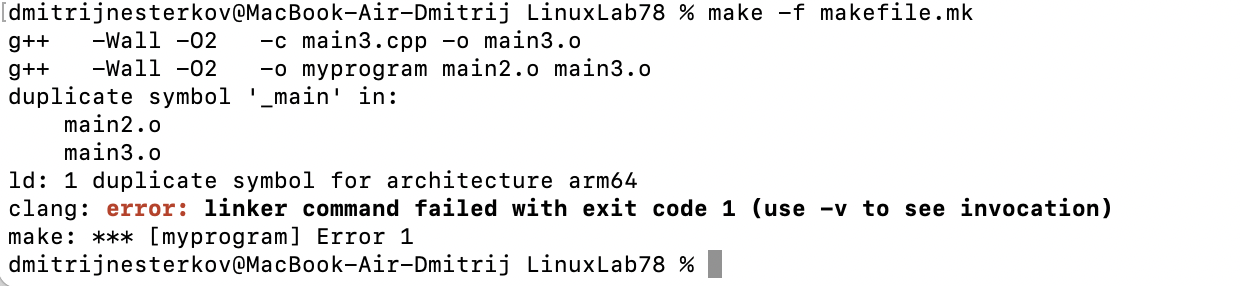


Рисунок 2 попытка объединения программ

Контрольные вопросы

Параметризация процесса сборки и переменные make-файла?

В Make-файлах вы можете определять переменные, которые позволяют параметризовать процесс сборки. Это полезно, например, для задания параметров компиляции, путей к файлам и других параметров.