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 Кругликовский М.А.

Принял**:** ст.пр. Станкевич C.П.

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++.

**Задание**

**Задание 1**

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 1**

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

**Задание 2**

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 2**

Выведите номера столбцов, элементы каждой из которых образуют монотонную последовательность (монотонно убывающую или монотонно возрастающую).

**Задание 3**

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 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?

**Результаты выполнения**

**Задание 1**

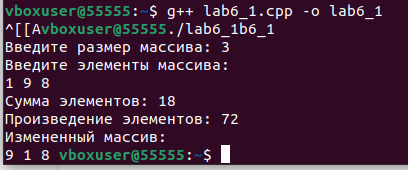
****

Рисунок 1. Результат программы

#include <iostream>

int main() {

std::cout << "Введите размер массива: ";

int N;

std::cin >> N;

double arr[N];

std::cout << "Введите элементы массива:" << std::endl;

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

std::cin >> arr[i];

}

double max = arr[0];

int maxIndex = 0;

double min = arr[0];

int minIndex = 0;

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

if(arr[i] > max) {

max = arr[i];

maxIndex = i;

}

if(arr[i] < min) {

min = arr[i];

minIndex = i;

}

}

std::swap(arr[maxIndex], arr[minIndex]);

double sum = 0;

double product = 1;

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

sum += arr[i];

product \*= arr[i];

}

std::cout << "Сумма элементов: " << sum << std::endl;

std::cout << "Произведение элементов: " << product << std::endl;

std::cout << "Измененный массив:" << std::endl;

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

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

}

return 0;

}

**Индивидуальное задание 1**

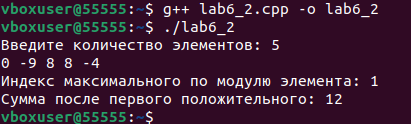


Рисунок 2. Результат программы

#include <iostream>

int main() {

int n;

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

std::cin >> n;

double arr[n];

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

std::cin >> arr[i];

}

int maxIndex = 0;

double max = fabs(arr[0]);

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

if(fabs(arr[i]) > max) {

max = fabs(arr[i]);

maxIndex = i;

}

}

std::cout << "Индекс максимального по модулю элемента: " << maxIndex << std::endl;

bool posFound = false;

double sum = 0;

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

if(arr[i] > 0) {

posFound = true;

}

if(posFound) {

sum += arr[i];

}

}

std::cout << "Сумма после первого положительного: " << sum << std::endl;

return 0;

}

**Задание 2**

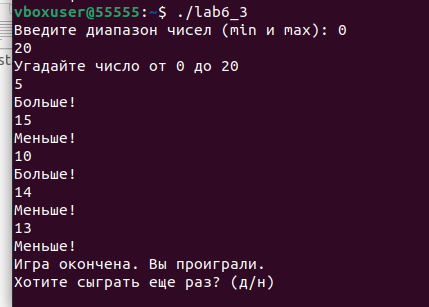


Рисунок 3. Результат программы

#include <iostream>

#include <cstdlib>

#include <ctime>

int main() {

srand(time(0));

int minNum, maxNum;

std::cout << "Введите диапазон чисел (min и max): ";

std::cin >> minNum >> maxNum;

int secretNum = minNum + rand() % (maxNum - minNum + 1);

int tries = 0;

const int maxTries = 5;

int guess;

std::cout << "Угадайте число от " << minNum << " до " << maxNum << std::endl;

do {

std::cin >> guess;

tries++;

if(guess > secretNum) {

std::cout << "Меньше!" << std::endl;

} else if (guess < secretNum) {

std::cout << "Больше!" << std::endl;

} else {

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

}

} while(guess != secretNum && tries < maxTries);

if(guess != secretNum) {

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

}

char playAgain;

std::cout << "Хотите сыграть еще раз? (д/н)" << std::endl;

std::cin >> playAgain;

if(playAgain == 'д') {

main();

}

return 0;

}

**Индивидуальное задание 2**

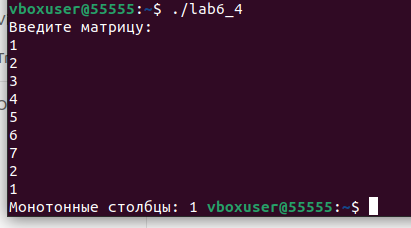


Рисунок 4. Результат программы

#include <iostream>

int main() {

const int ROWS = 3;

const int COLS = 4;

int matrix[ROWS][COLS];

// Ввод матрицы

std::cout << "Введите матрицу: " << std::endl;

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

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

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

}

}

// Поиск монотонных столбцов

std::cout << "Монотонные столбцы: ";

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

bool isMonotonic = true;

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

if(matrix[i][j] < matrix[i-1][j]) {

isMonotonic = false;

break;

}

}

if(isMonotonic) {

std::cout << j + 1 << " ";

}

}

return 0;

}

**Задание 3**

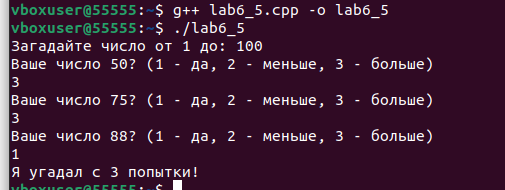


Рисунок 5. Результат программы

#include <iostream>

int main() {

int minNum, maxNum;

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

std::cin >> maxNum;

minNum = 1;

int tries = 0;

const int maxTries = ceil(log2(maxNum));

int left = minNum;

int right = maxNum;

int guess;

while (left <= right) {

guess = left + (right - left) / 2;

tries++;

std::cout << "Ваше число " << guess << "? (1 - да, 2 - меньше, 3 - больше)" << std::endl;

int ans;

std::cin >> ans;

if (ans == 1) {

std::cout << "Я угадал с " << tries << " попытки!" << std::endl;

break;

}

else if (ans == 2) {

right = guess - 1;

}

else {

left = guess + 1;

}

}

return 0;

}

**Индивидуальное задание 3**

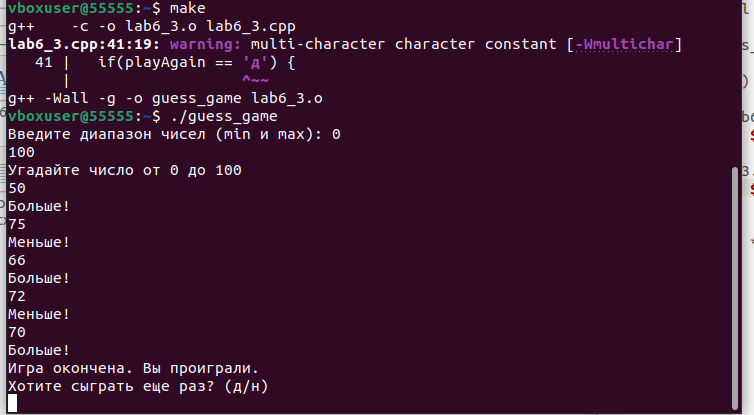
****

Рисунок 6. Результат программы

CC = g++

CFLAGS = -Wall -g

TARGET = guess\_game

all: $(TARGET)

$(TARGET): lab6\_3.o

$(CC) $(CFLAGS) -o $(TARGET) lab6\_3.o

main.o: lab6\_3.cpp

$(CC) $(CFLAGS) -c lab6\_3.cpp

clean:

rm -f \*.o $(TARGET)

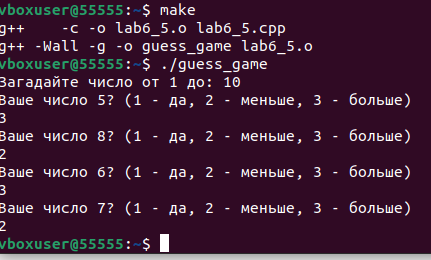


Рисунок 7. Результат программы

CC = g++

CFLAGS = -Wall -g

TARGET = guess\_game

all: $(TARGET)

$(TARGET): lab6\_5.o

$(CC) $(CFLAGS) -o $(TARGET) lab6\_5.o

main.o: lab6\_5.cpp

$(CC) $(CFLAGS) -c lab6\_5.cpp

clean:

rm -f \*.o $(TARGET)

**Контрольные вопросы**

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

Компоновка (линковка) в программировании - это процесс объединения всех отдельных объектных файлов вместе, чтобы создать исполняемый файл или библиотеку.

**Что такое рефакторинг кода?**

Рефакторинг кода - это процесс изменения внутренней структуры программного кода с целью улучшения его читаемости, понимаемости, поддерживаемости и расширяемости, при этом без изменения внешнего поведения программы.

**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?**

Makefile представляет собой текстовый файл, который содержит набор целей (targets), зависимостей (dependencies) и инструкций (instructions) для управления процессом сборки программного обеспечения. Он используется утилитой make для автоматизации процесса компиляции программ.

Вот базовый синтаксис make-файла:

target: dependencies

instructions

target представляет собой объект, который требуется создать. Это может быть исполняемый файл, объектный файл, или какой-либо другой файл, необходимый для проекта.

dependencies - файлы или цели, от которых зависит целевой файл. Если одна или несколько зависимостей изменяются, целевой файл будет пересобран.

instructions представляют собой команды, необходимые для создания целевого файла из его зависимостей.

Первая цель в make-файле обрабатывается при запуске утилиты make, если не указана цель явным образом. Эта цель называется «основной целью сборки». Обычно она связана с компиляцией основного исполняемого файла или библиотеки.

Основная цель сборки обычно включает в себя компиляцию основных исходных файлов проекта, линковку объектных файлов и создание окончательного исполняемого файла или библиотеки. Это может также включать создание дополнительных ресурсов, таких как документация или тесты, в зависимости от специфики проекта.