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 № 2\_1**

по дисциплине: “Системное программирование”

на тему: ***“Компиляция и отладка простейшего приложения в Linux”***

Вариант 1

Выполнил**:** студент группы 10702121 Меркулова М. С.

Принял**:** Давыденко Н.В.

Минск 2023

# Лабораторная работа №2\_1. Компиляция и отладка простейшего приложения в Linux

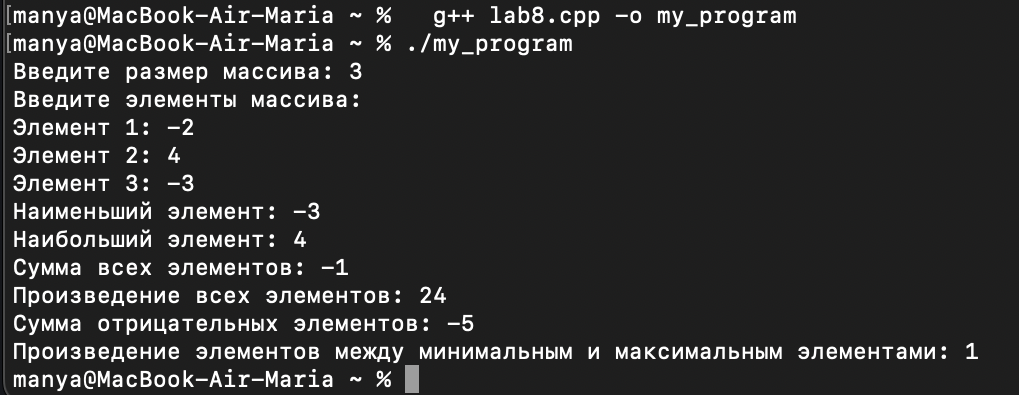
Цель работы: изучить встроенный инструментарий для разработки приложений под семейство ОС Linux и фундаментальные основы системного программирования с использованием компиляторов gcc/g++, отладчика gdb и других для проектирования, компиляции, отладки и запуска приложений на языке программирования С/C++.

**Основное задание**

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**

Водномерноммассиве,состоящемизnвещественныхэлементов,вычислить: сумму отрицательных элементов массива и произведение элементов массива, расположенных между максимальным и минимальным элементами.



**Код программы**

#include <iostream>

#include <vector>

int main() {

int n;

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

std::cin >> n;

if (n <= 0) {

std::cout << "Размер массива должен быть положительным числом." << std::endl;

return 1;

}

std::vector<double> arr(n);

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

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

std::cout << "Элемент " << (i + 1) << ": ";

std::cin >> arr[i];

}

// Находим наибольший и наименьший элементы и меняем их местами

double min\_element = arr[0];

double max\_element = arr[0];

int min\_index = 0;

int max\_index = 0;

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

if (arr[i] < min\_element) {

min\_element = arr[i];

min\_index = i;

}

if (arr[i] > max\_element) {

max\_element = arr[i];

max\_index = i;

}

}

std::swap(arr[min\_index], arr[max\_index]);

// Находим сумму и произведение всех элементов

double sum = 0.0;

double product = 1.0;

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

sum += arr[i];

product \*= arr[i];

}

std::cout << "Наименьший элемент: " << min\_element << std::endl;

std::cout << "Наибольший элемент: " << max\_element << std::endl;

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

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

// Вычисляем сумму отрицательных элементов и произведение элементов между min и max

double sum\_of\_negatives = 0.0;

double product\_between\_min\_max = 1.0;

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

if (arr[i] < 0) {

sum\_of\_negatives += arr[i];

}

}

if (min\_index < max\_index) {

for (int i = min\_index + 1; i < max\_index; i++) {

product\_between\_min\_max \*= arr[i];

}

} else {

for (int i = max\_index + 1; i < min\_index; i++) {

product\_between\_min\_max \*= arr[i];

}

}

std::cout << "Сумма отрицательных элементов: " << sum\_of\_negatives << std::endl;

std::cout << "Произведение элементов между минимальным и максимальным элементами: " << product\_between\_min\_max << std::endl;

return 0;

}

**Контрольный вопрос**

**Опишите базовый инструментарий системного программиста, проектирую- щий и разрабатывающий свои приложения на языке С/C++.**

Системный программист, проектирующий и разрабатывающий приложения на языке C/C++, должен обладать широким набором инструментов и знаний, чтобы успешно выполнять свои задачи. Вот базовый инструментарий и знания, которые понадобятся системному программисту:

Интегрированная среда разработки (IDE):

• Программист может выбрать подходящую IDE для разработки на C/C++. Некоторые популярные варианты включают в себя Visual Studio, CLion, Xcode (для macOS), Code::Blocks и другие.

Компиляторы:

• Для разработки на C/C++ необходим компилятор. GCC (GNU Compiler Collection) является одним из самых популярных, но существуют и другие компиляторы в зависимости от платформы (например, Clang для macOS).

Отладчики:

• Инструменты отладки, такие как GDB, LLDB (для macOS), и встроенные отладчики в IDE, позволяют обнаруживать и исправлять ошибки в коде.

Системы контроля версий:

• Системы контроля версий, такие как Git, SVN и Mercurial, необходимы для управления версиями кода и совместной работы над проектами.

Библиотеки и фреймворки:

• Системные программисты часто используют библиотеки и фреймворки для упрощения разработки. Например, для сетевого программирования можно использовать библиотеку Boost, а для разработки веб-приложений - фреймворк Qt.

Знание языка C/C++:

• Это, конечно, самое важное. Программист должен обладать глубоким пониманием C/C++ и их особенностей.

Знание операционных систем:

• Знание особенностей операционных систем (Linux, Windows, macOS) и их API позволяет создавать приложения, которые взаимодействуют с операционной системой и аппаратными ресурсами.

Понимание многопоточности и параллелизма:

• Понимание работы многопоточности и возможности разработки параллельных приложений является важным, особенно при работе с многозадачными системами.

Знание архитектуры компьютера:

• Знание архитектуры процессора и памяти позволяет оптимизировать код для максимальной производительности.

Умение анализировать и оптимизировать код:

• Умение выявлять и устранять узкие места в производительности, а также оптимизировать код для эффективной работы.

Понимание безопасности:

• Понимание принципов безопасности и умение писать безопасный код, чтобы предотвращать уязвимости.

Умение работать с сетями и базами данных:

• Если приложение взаимодействует с сетями или базами данных, системный программист должен знать соответствующие технологии и протоколы.

Умение работать с потоками данных и файлами:

• Обработка потоков данных и взаимодействие с файловой системой часто встречаются в системном программировании.

Тестирование и юнит-тесты:

• Знание методов тестирования и создание юнит-тестов для обеспечения качества кода.

Документирование и отладка:

• Умение создавать документацию и эффективно отлаживать код.

Опыт работы с командой:

• Если проект требует совместной работы, системный программист должен уметь взаимодействовать с другими членами команды.

Этот инструментарий и знания позволят системному программисту разрабатывать высококачественное программное обеспечение, которое взаимодействует с операционной системой и аппаратным обеспечением компьютера.