Министерство образования и науки Челябинской области

государственное бюджетное профессиональное

образовательное учреждение

«Златоустовский индустриальный колледж им. П.П. Аносова»

**ЗАЩИТА**

Руководитель УП ПМ.03

Преподаватель ГБОУ ЗлатИК

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ю.В.Майер

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

По учебной практике

Специальность: 09.02.07 «Информационные системы и программирование»

ПМ.03 «Ревьюирирование программных продуктов»

Выполнил:

Студент группы

Кочев.А.С.

\_\_\_\_\_\_\_\_\_\_\_дата

2020-2021 уч.г.

План прохождения учебной практики

по ПМ.03 «Ревьюирирование программных продуктов»

Студент Кочев А.С.

Группа ИС-32

Специальность 09.02.07 «Информационные системы и программирование»

Количество часов \_\_\_\_\_36\_\_\_\_\_\_\_\_\_\_

Руководитель практики \_\_\_Ю.В.Майер\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сроки проведения практики\_\_\_\_29.11.-04.12.2021\_\_\_\_\_\_\_

Таблицу 1 – План прохождения практики

|  |  |  |
| --- | --- | --- |
| Этапы практики | Дата | Освоен  (да/нет) |
| Измерение производительности приложения посредством анализа использования ЦП | 29.11 |  |
| Установка ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования | 30.11 |  |
| Обратное проектирование | 01.12 |  |
| Дизассемблирование | 02.12 |  |
| Разработать сопроводительную документацию | 03.12 |  |
| Защита практики | 04.12 |  |

**Отзыв руководителя**

Студент Кочев А.С

Группа ИС-32

Специальность 09.02.07 «Информационные системы и программирование»

Количество часов \_\_\_\_\_36\_\_\_\_\_\_\_\_\_\_

Руководитель практики \_\_\_Ю.В.Майер\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сроки проведения практики\_\_\_\_29.11– 04.11 2021г.\_\_\_\_\_\_\_

Место работы ГБПОУ «Златоустовский индустриальный колледж им. П.П. Аносова»

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Показатели и критерии оценивания** | **Баллы** | **Фактический балл** |
| **1** | **Структура отчета по практике** | | |
|  | Структура полностью соответствует заданию | 3 |  |
| Структура имеет несущественное несоответствие заданию | 2 |  |
| Структура существенно не соответствует заданию | 1 |  |
| **2** | **Соответствие содержания рабочей теме,цели,задачам** |  |  |
|  | Полное соответствие | 3 |  |
| Частичное соответствие | 2 |  |
| Низкая степень соответствия | 1 |  |
| **3** | **Полнота раскрытия темы** | | |
|  | Тема раскрыта полностью, приведены необходимые расчеты, пояснения, аргументы, сделаны выводы | 3 |  |
| Тема раскрыта полностью, однако приведены не все необходимые расчеты, пояснения и аргументы | 2 |  |
| Тема раскрыта частично, нет необходимых расчетов, пояснений, аргументов, не сделаны выводы | 1 |  |
| **4** | **Логика изложения материала** | | |
|  | Все структурные элементы организованы в систему, прослеживается логика в раскрытии темы | 3 |  |
| Все структурные элементы организованы в систему, логика в раскрытии темы частично нарушена | 2 |  |
| Все структурные элементы организованы в систему, но нет логики в раскрытии темы | 1 |  |
| **5** | **Соблюдение требований ГОСТ к оформлению ПЗ** | | |
|  | Требования ГОСТ соблюдены полностью | 3 |  |
| Имеются незначительные отклонения от ГОСТ | 2 |  |
| Есть существенные нарушения требований ГОСТ | 1 |  |
| **6** | **Практическая часть** | | |
|  | Выполнена в соответствии с требованиями, без отклонений от нормативов | 3 |  |
| Имеется несущественное отклонение от нормативов | 2 |  |
| Имеется существенное отклонение от нормативов | 1 |  |
| ИТОГО | | |  |

Критерии оценивания:

18-15 б. – «отлично»;

11-14 б. – «хорошо»;

8-10 б. – «удовлетворительно»;

Меньше 8 б. работа не оценивается.

Подпись руководителя:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ю.В.Майер

\_\_\_\_\_\_\_\_\_\_\_\_\_дата

**Содержание**

[Введение 5](#_Toc88822817)

[1. Список терминов и определений 7](#_Toc88822818)

[2. Функциональные требования 8](#_Toc88822819)

[3. Измерение производительности приложения посредством анализа использования ЦП 9](#_Toc88822820)

[4. Установка ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования 14](#_Toc88822821)

[5. Дизассемблирование 21](#_Toc88822824)

[6. Обратное проектирование 23](#_Toc88822825)

[ЗАКЛЮЧЕНИЕ 24](#_Toc88822826)

ЛИТЕРАТУРА………………………………………………………...................25

**Введение**

Процессы разработки, приобретения и внедрения сложных систем, к которым относятся в частности программные комплексы, должны находится под жестким управленческим контролем. В настоящее время практически во всех организациях обеспечивается контроль важнейших характеристик, связанных с производством и использованием программных продуктов, таких как время, финансовые средства, ресурсы и т.п. Однако в большинстве случаев вне пределов сферы контроля оказывается наиболее важная характеристика программных продуктов, ради которой, собственно и осуществляются затраты времени, финансовых средств и ресурсов – это качество продукта, поскольку «невозможно контролировать то, что нельзя измерить» (“You cannot control what you cannot measure”).

Дизассемблирование **–** преобразованиепрограммы на машинном языке к ее ассемблерному представлению. Декомпиляция – получение кода языка высокого уровня из программы на машинном языке или ассемблере.

Под *анализом потоков данных* понимают совокупность задач, нацеленных на выяснение некоторых глобальных свойств программы, то есть извлечение информации о поведении тех или иных конструкций в некотором контексте.

Основным результатом деятельности группы разработчиков являются не диаграммы, а программное обеспечение, поэтому модели и основанные на них реализации должны соответствовать друг другу с минимальными затратами по поддержанию синхронизации между ними. Чаще всего разработанные модели преобразуются в программный код. Хотя UML не определяет конкретного способа отображения на какой-либо объектно-ориентированный язык, он проектировался с учетом этого требования. В наибольшей степени это относится к диаграммам классов, содержание которых без труда отображается на такие известные объектно-ориентированные языки программирования, как Java, C++, ObjectPascal, Visual Basic и др.

*Прямым проектированием*(Forward engineering) называется процесс преобразования модели в код путем отображения на некоторый язык реализации.

*Обратным проектированием*(Reverse engineering) называется процесс преобразования в модель кода, записанного на каком-либо языке программирования.

Цель учебной практики: Научиться выполнять прямое и обратное проектирование ПО, овладеть навыками приостановки и возобновления работы с рабочими задачами иинспекцией кода в Visual Studio, совместно работать над проектом в системе контроля версий GIT.

В ходе учебной практики для достижения цели, ставятся задачи:

* измерить производительность приложения посредством анализа использования Центральный Процессор,
* установить ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования,
* выполнить дизассемблирование,
* выполнить обратное проектирование.

# **1. Список терминов и определений**

***Ubuntu*** – это разрабатываемая сообществом, основанная на ядре Linux операционная система, которая идеально подходит для использования на персональных компьютерах, ноутбуках и серверах.

***Терминал*** – это графическая программа эмулирующая консоль.

**Виртуальная машина** - это виртуальный компьютер, который использует выделенные ресурсы реального компьютера.

**Декомпиляция** – получение кода языка высокого уровня из программы на машинном языке или ассемблере.

**Дизассемблирование –** преобразованиепрограммы на машинном языке к ее ассемблерному представлению.

**Прямое проектирование** - процесс преобразования модели в код путем отображения на некоторый язык реализации.

**Обратное проектирование** - процесс преобразования в модель кода, записанного на каком-либо языке программирования.

**Точка останова** – это преднамеренное прерывание выполнения программы, при котором выполняется вызов отладчика.

Центральный Процессор – центральный процессор компьютера.

## **2. Функциональные требования**

Для корректного выполнения задания по учебной практике нам нужны:

Во-первых, среда разработки для написания, отладки и тестирования кода Visual Studio 2019. Во-вторых, текстовый редактор для работы с отчётом Word 2019. ПП виртуализации для операционных систем Microsoft, Linux Oracle VirtualBox.

Настройки VirtualBox:

Оперативная память: 1815 МБ

Процессоры: 2

Порядок загрузки: Гибкий диск, Оптический диск, Жёсткий диск

Ускорение: VT-x/AMD-V, Nested Paging, PAE/NX, Паравиртуализация KVM

Видеопамять: 16 МБ

Графический контроллер: VMSVGA

Сервер удалённого дисплея: выключен

Запись: выключена

Контроллер SATA

SATA порт 0: (обычный, 15,94 ГБ)

## **3. Измерение производительности приложения посредством анализа Центральный Процессор**

Запускаем полнофункциональную среду разработки Visual Studio 2019, открываем проект с написанным кодом, производительность которого нам необходимо измерить и ставим первую точку остановы на участке исследования.



Рис. 1

Таким же образом располагаем вторую точку, так мы отделяем нужный участок кода, чтобы более точечно провести измерения.



Рис. 2

Нажатием клавиши F5 или нажатием кнопки запуска программы в панели разработчика запускаем программу в режиме отладки.

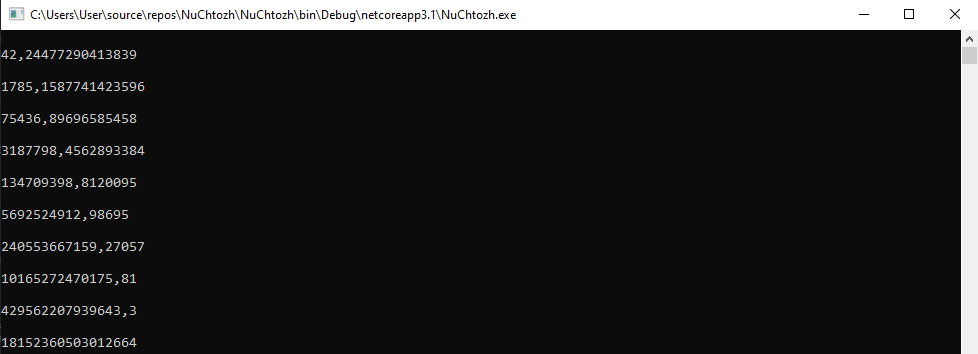


Рис. 3

После запуска программы, в правой части отладчика у нас появляется окно средств диагностики, в котором мы можем увидеть данные о событиях. Начинается потребление мощности Центральный Процессор.

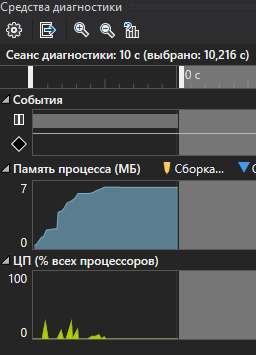


Рис. 4

Для получения информации о загрузке Центральный Процессор перейдём во вкладку Использование Центральный Процессор

Повторно нажимаем клавишу F5 (или кнопку запуска) и входим в первую точку остановы.

Мы видим цветовую диаграмму, описывающую процессы, которые задействуют Центральный Процессор, список функций, кнопки для сортировки.

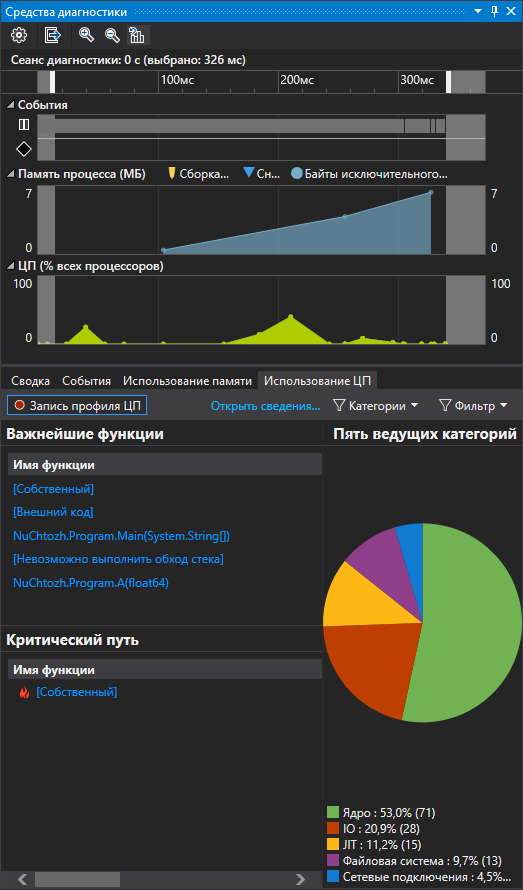


Рис. 5

Щёлкнув дважды одну из функций нашего приложения из списка, откроется представление Вызывающий\вызываемый. В этом представлении выбранная функции отображается в заголовке и в поле Текущая функция. Функция, вызывавшая текущую функцию, отображается в левой части окна в разделе Вызывающие функции, а все функции, вызываемые текущей функцией, отображаются в поле Вызываемые функции справа. (Можно выбрать любое поле, чтобы изменить текущую функцию).

В этом представлении показано общее время (мс) и доля общего времени выполнения приложения, затраченного на выполнение функции. В поле Тело функции также показан общий объем времени (и доля времени), затраченного в теле функции за исключением времени, затраченного в вызываемых и вызывающих функциях.

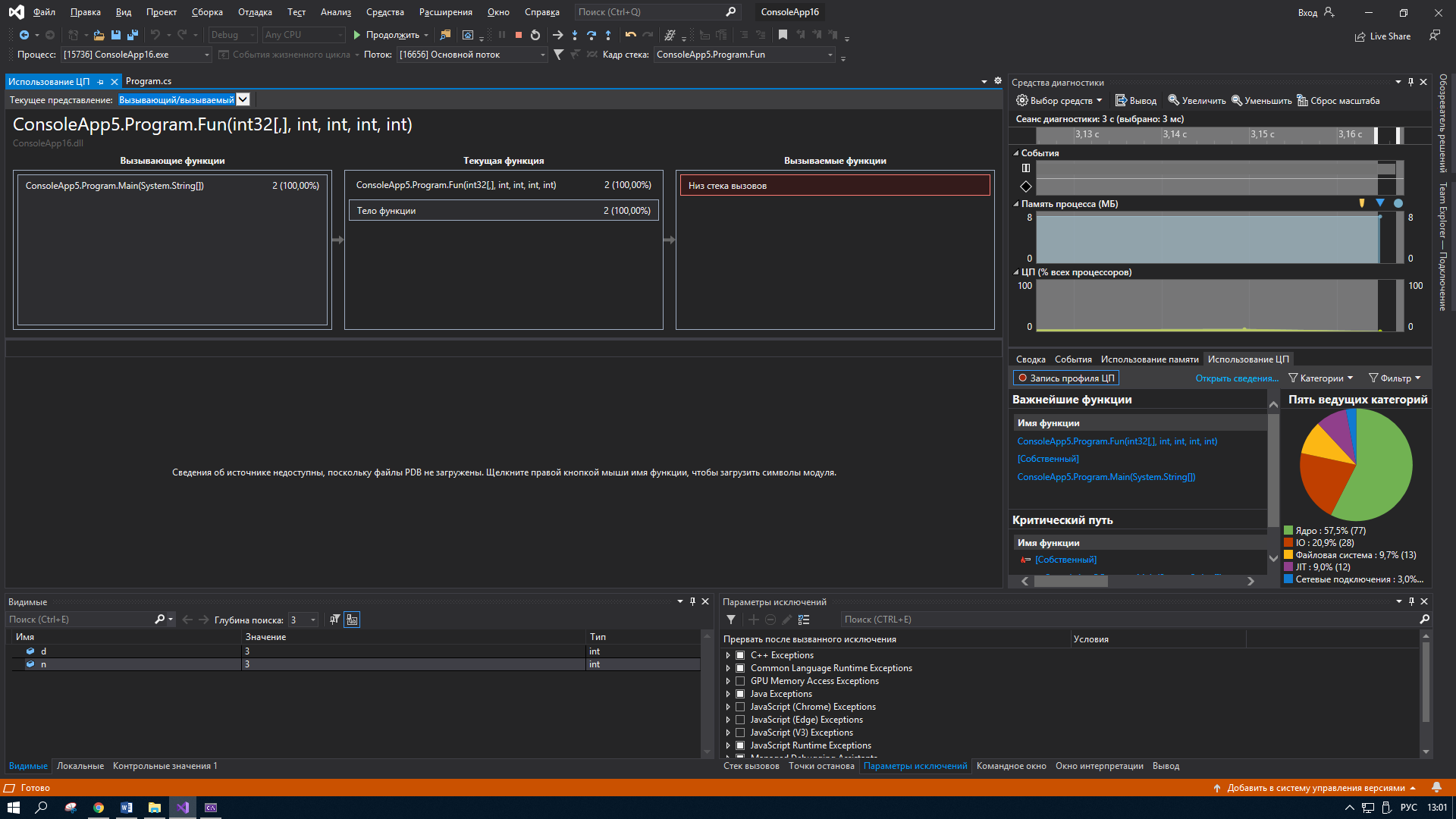


Рис. 7

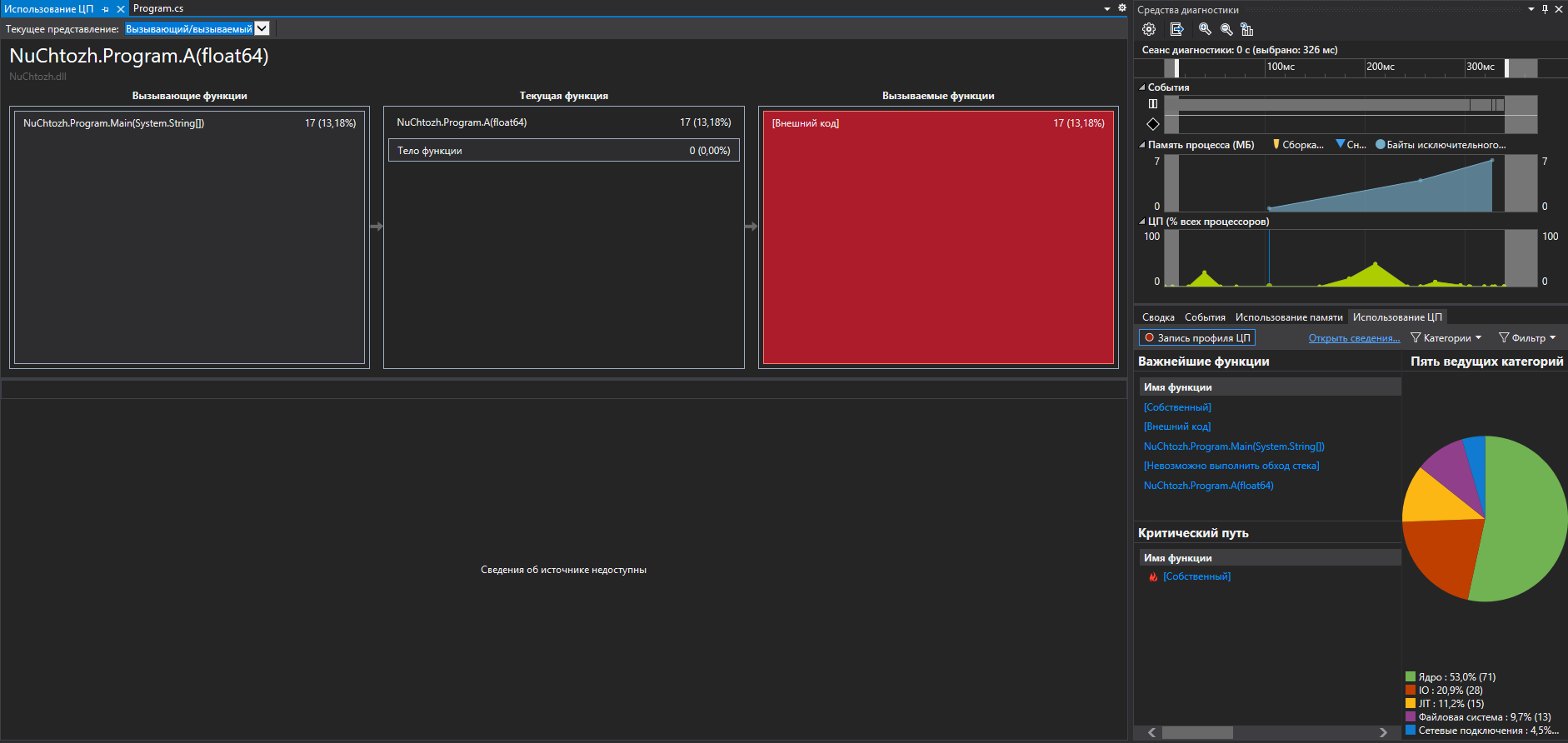


Рис. 8

В выпадающем списке на панели сверху меняем Текущее представление на Дерево вызовов. Каждая нумерованная область на рисунке соответствует определенному шагу в процедуре.

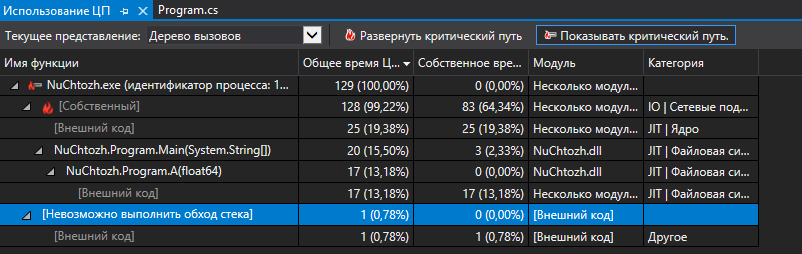


Рис. 9

Ставим галочку «Показать внешний код».

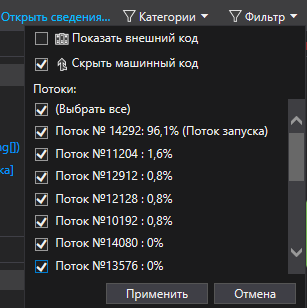


Рис. 10

«Дерево вызовов» с внешними кодами.

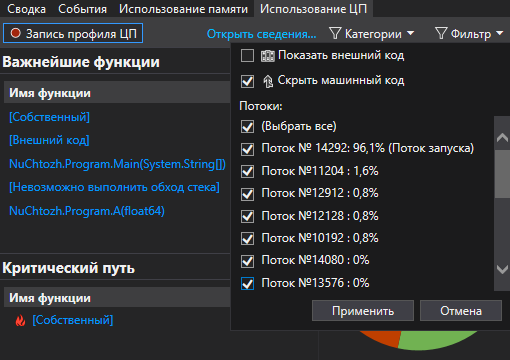


Рис. 11

В итоге проделанной работы я ознакомился с измерением производительности путём анализа Центральный Процессор и понял, что это можно использовать для оптимизации разрабатываемых приложений . В моей работе после вхождения в границу точек остановы потребление Центральный Процессор увеличилось.

**4. Установка ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования**

### 

### Установка Ubuntu

На первом шаге работы создаём новую виртуальную машину для работы с операционной среде Linux, с помощью программы VirtualBox на оболочке Ubuntu.

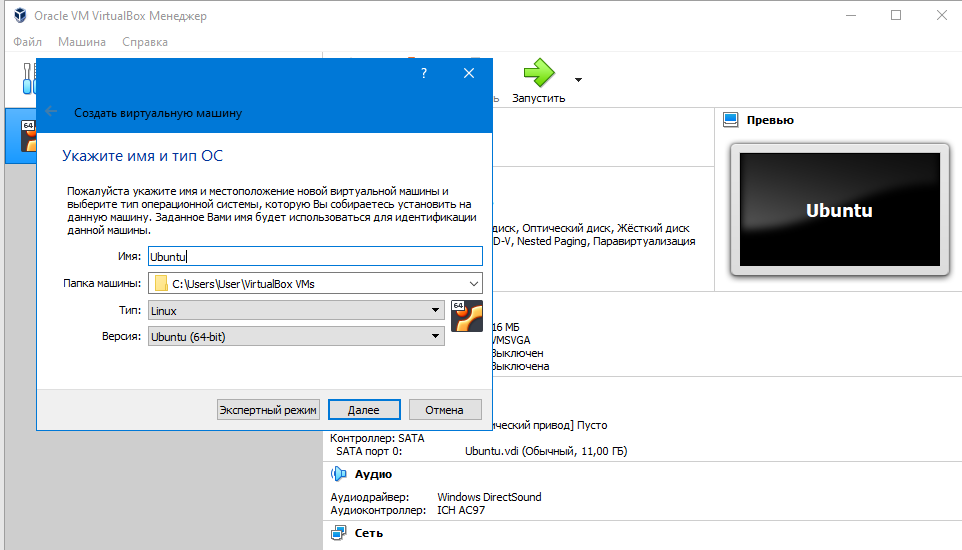


Рис. 12

Указываем объём оперативной памяти, выделяемый для работы виртуальной машины. Примерно 2гб (главное не выходить за зелёный на шкале).

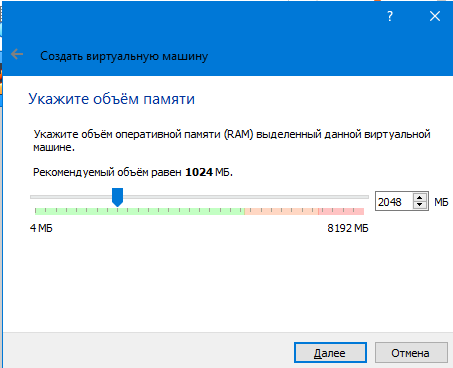


Рис. 13

Создаём новый виртуальный диск.



Рис. 14

Указываем размер диска (10-15 ГБ).

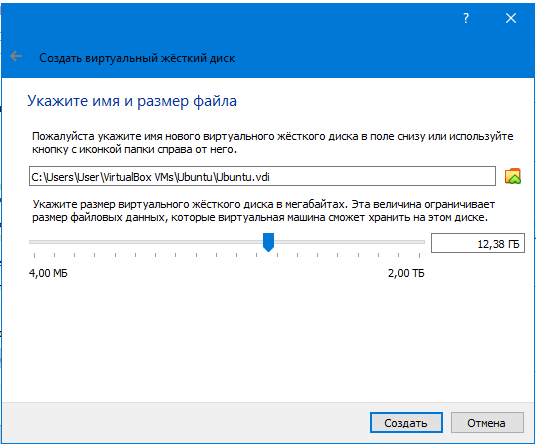


Рис. 15

Виртуальная машина создана. Дальше нужно открыть настройки. Заходим в раздел «Дисплей» и ставим галочку у пункта «Включить 3D-ускорение».

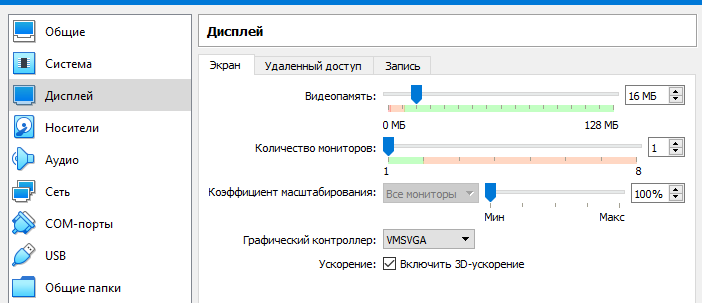


Рис. 16

Дальше переходим в раздел «Система», выбираем категорию «Процессор» и добавляем ещё один процессор.

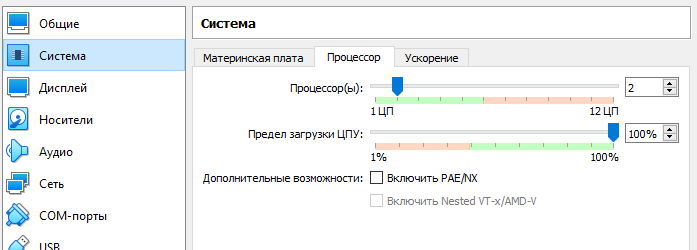


Рис. 17

После чего переходим в раздел «Носители». Нажимаем на пустой контроллер, и подключаем образ Ubuntu.

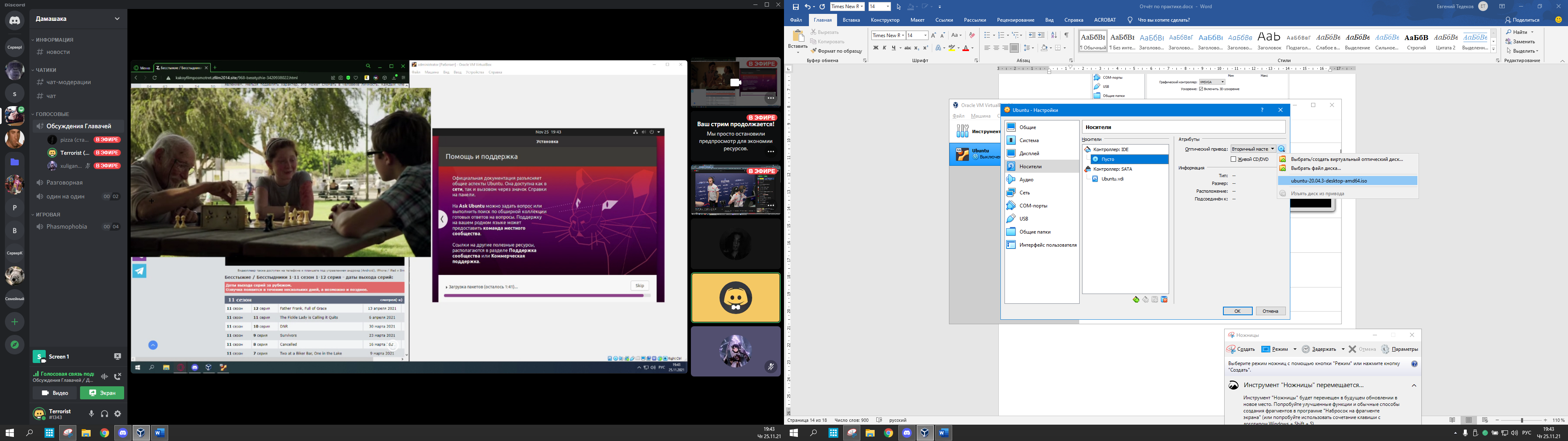


Рис.18

### Настройка прокси на компьютерах колледжа

Создав виртуальный образ, мы заходим в него и открываем терминал. С помощью команды nano открываем панель nano.



Рис. 19

Вписываем туда данные, необходимые для работы с прокси.

Рис. 20

Устанавливаем компилятор, программу, которая будет транслировать нам программу, составленную на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком к машинному коду. GCC (набор компиляторов для различных языков) и все необходимые для него компоненты.

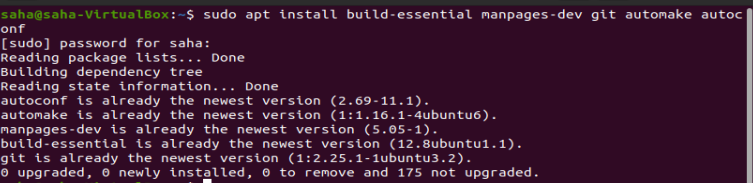


Рис. 21

Создаём на рабочем столе файл формата C.



Рис. 22

Открываем его и вписываем туда программный код.

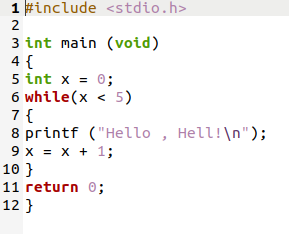


Рис. 23

Компилируем данный файл (преобразуем в язык программирования более низкого уровня), указав ему новое имя.





Рис. 24

В терминале запускаем скомпилированный файл и проверяем правильность выполнения.

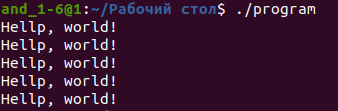


Рис. 25

Дизассемблирование

Устанавливаем необходимые пакеты

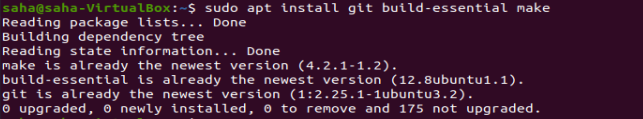


Рис. 26

Прописываем

****

Рис. 27

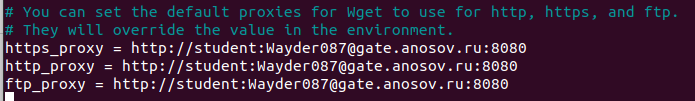


Рис. 28

Рис. 29

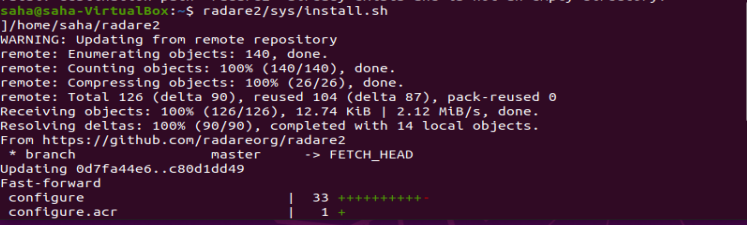


Рис. 30

Устанавливаем необходимые компоненты для графической оболочки «Iaito»

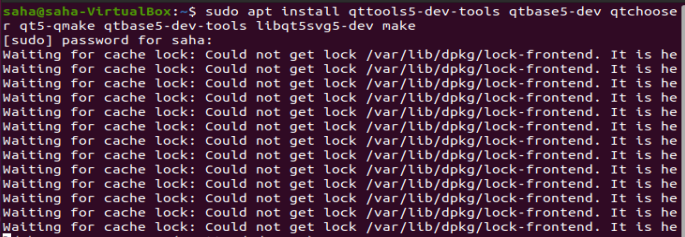


Рис. 31

Открываем папку и открываем файл



Рис. 32

Устанавливаем графический интерфейс «iaito»

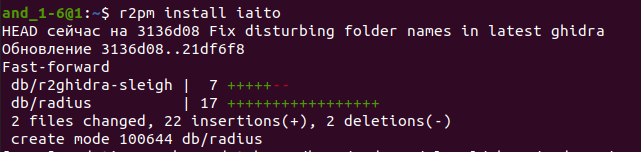


Рис. 34

Устанавливаем Декомпилятор «r2ghidra»

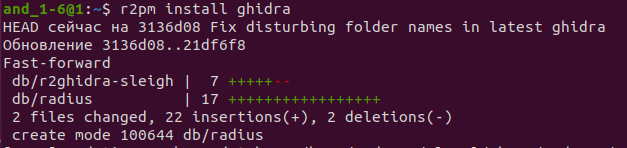


Рис. 35

## **5. Дизассемблирование**

1) Запускаем «Iaito»

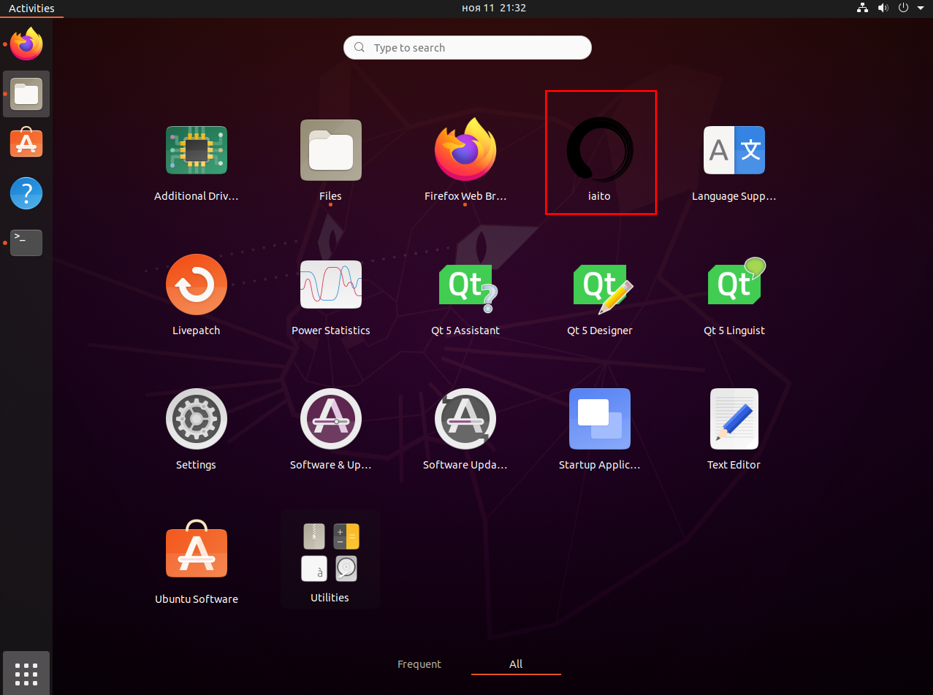


Рис. 36 – меню приложений

2) Запускаем и проверяем тестовый файл

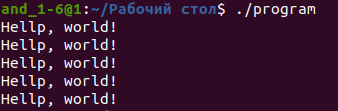


Рис. 37 – проверка файла

3) Открываем тестовый файл

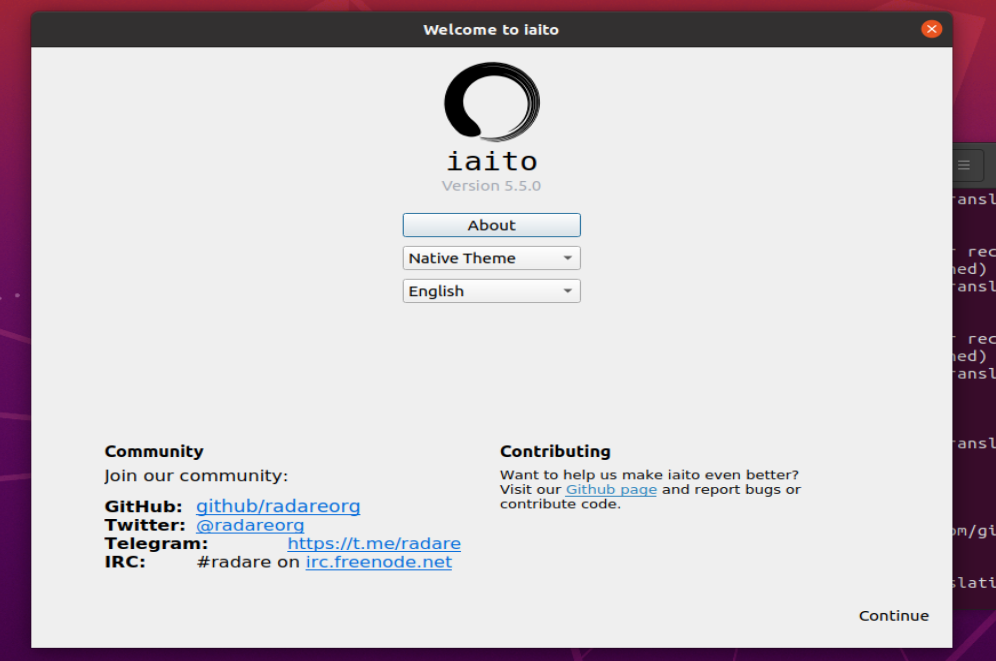


Рис. 38 – запуск iaito

4) Открываем раздел main

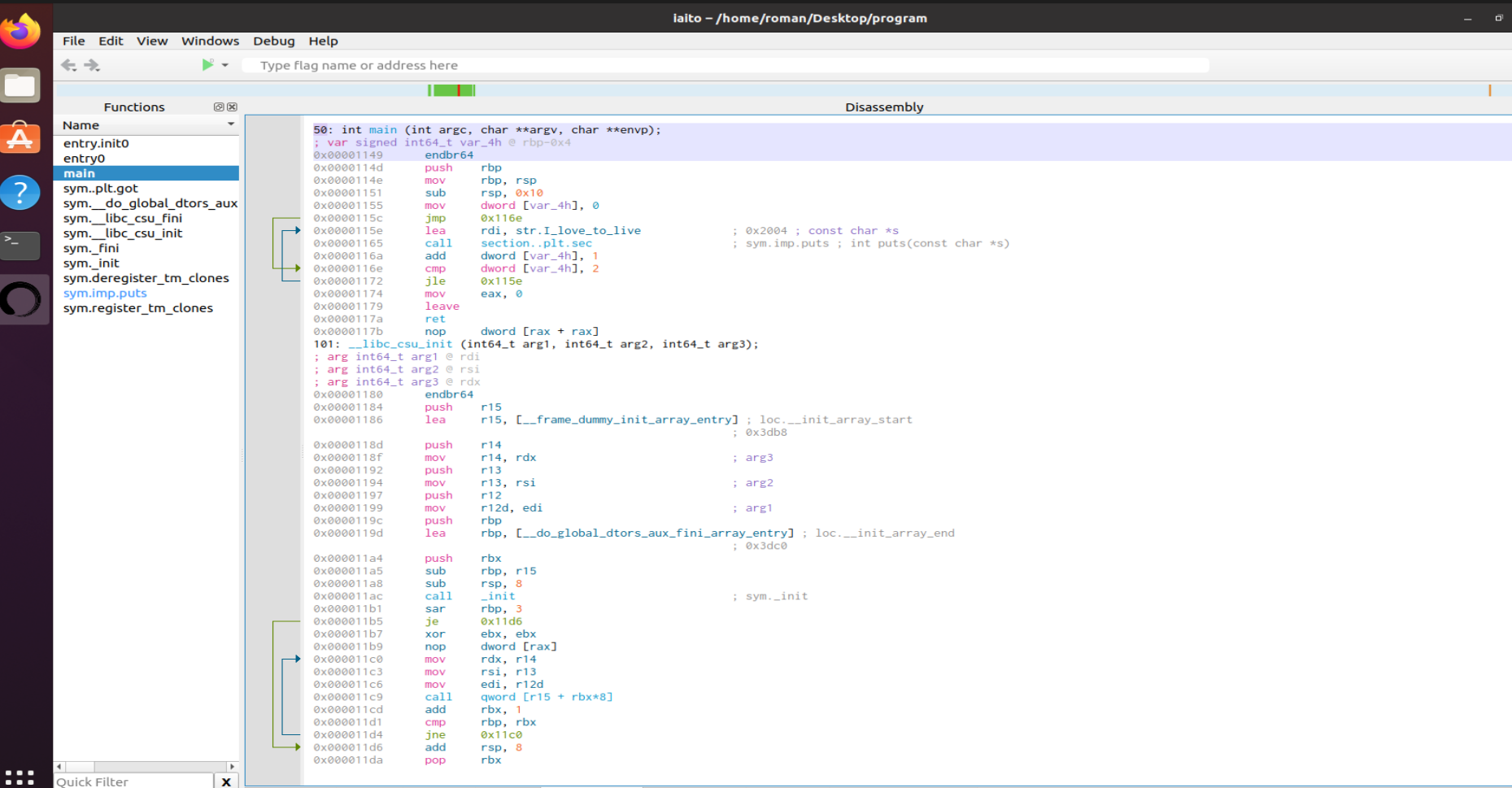


Рис. 39 – раздел Main

5) Заходим в раздел Graph

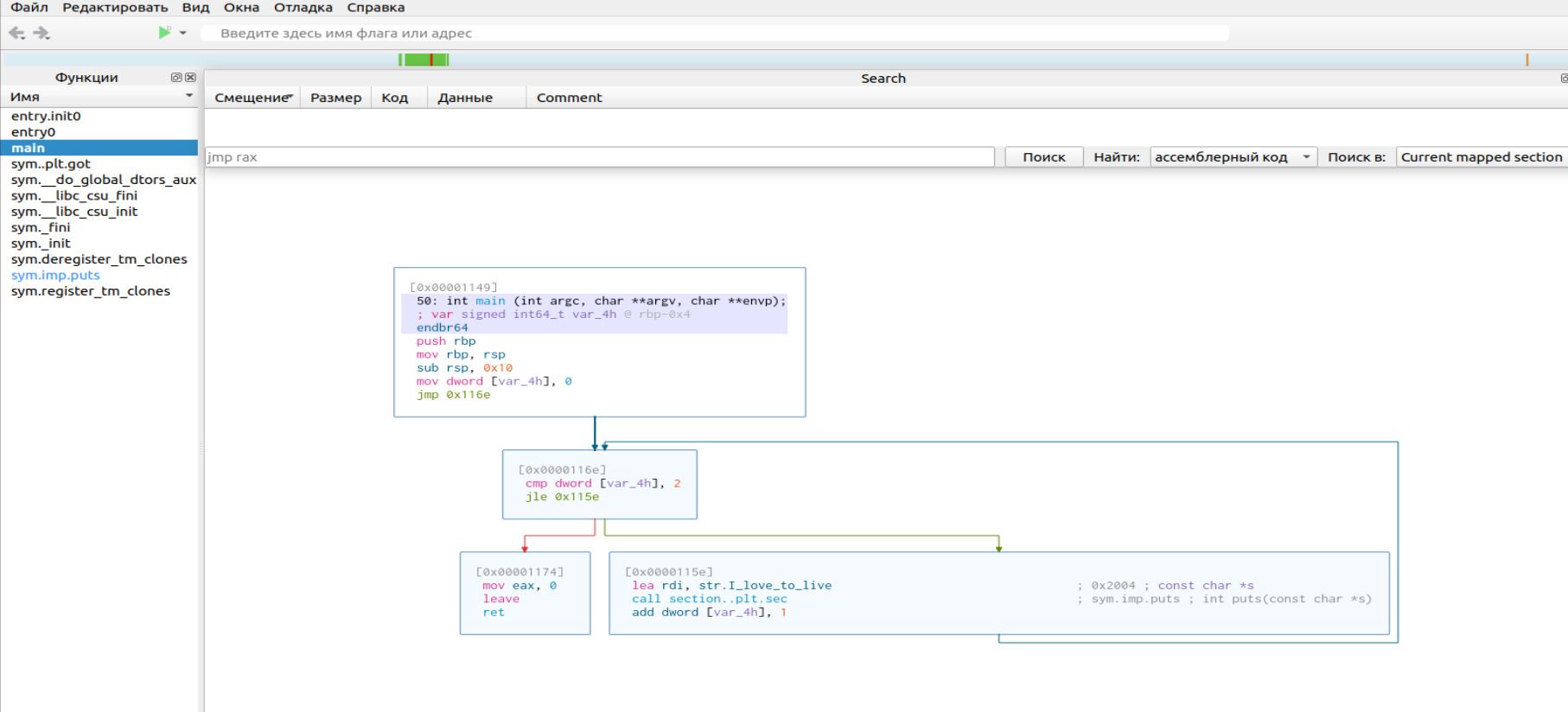


Рис. 40 – раздел Graph

6) Выделяем нужную переменную и переходим в дизассемблер, заменяем её на новое значение



Рис. 41 - находим переменную



Рис. 42 – новое значение

## **6. Обратное проектирование**

1) Был создан файл program.c с написанным кодом

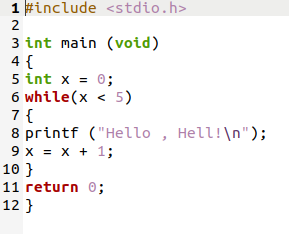


Рис. 43 – код файла

2) В результате выполнения программы проводилось 3 итерации

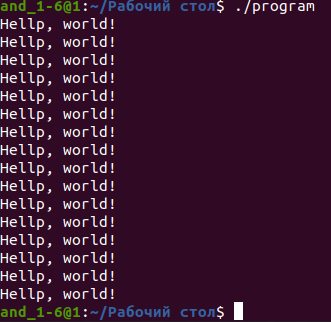


Рис. 44 – Вывод 3 итераций

3) После дизассемблирования в файл были внесены изменения. Количество итераций увеличилось до 10

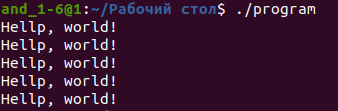


Рис. 45 – 10 итераций

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения заданий на учебную практику было выполнено измерение Центральный Процессор, провели анализ и пришли к выводу то, что упрощение кода может разгрузить работу Центральный Процессор. Провели процесс дизассемблирования и обратного проектирования при этом была установлена виртуальная машина с операционной системой Ubuntu. При выполнение практических работ пользовались репозиторием GIT для взаимодействия с файлами.

В ходе выполнения заданий на учебную практику были выполнены следующие задачи:

измерили производительность приложения посредством анализа использования Центральный Процессор,

установили ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования,

выполнили дизассемблирование,

выполнили обратное проектирование.

**ЛИТЕРАТУРА**

1. (бакалавриат), 38.03.05 (бакалавриат) и 10.05.02 (специалитет) всех профилей подготовки / Юрий Владимирович Ланских ; ВятГУ, ФАВТ, каф. АТ. - Киров: [б. и.], 2015. - 138 с.
2. 3.Золотов, С. Ю. Проектирование информационных систем [Электронный ресурс] / С.Ю. Золотов. - Томск: Эль Контент, 2013. - 88 с.
3. Карпенков, С. Х. Технические средства информационных технологий [Электронный ресурс] / С.Х. Карпенков. - 3-е изд., испр. и доп. - М.|Берлин: Директ-Медиа, 2015. - 376 с.
4. Коноплева, И. А. Информационные технологии [Электронный ресурс] / И.А. Коноплева. - 2-е изд., перераб. и доп. - Москва: Проспект, 2014. - 328 с.
5. Корячко, В. П. Процессы и задачи управления проектами информационных систем [Электронный ресурс] / В.П. Корячко. - Москва: Горячая линия - Телеком, 2014. - 376 с.
6. Ланских, Юрий Владимирович Предметно-ориентированные информационные системы [Электронный ресурс] : учеб. пособие для студентов направления 09.03.02, 10.03.01,
7. Проектирование информационных систем. Лекция 1. Презентация [Электронный ресурс]. - Москва: Национальный Открытый Университет «ИНТУИТ», 2014. - 27 с.
8. Советов, Борис Яковлевич. Информационные технологии [Электронный ресурс]: учебник / Б. Я. Советов, В. В. Цехановский. - 6-е изд. - Москва: Юрайт, 2015. - х эл. опт. диск (CD-ROM)
9. Советов, Борис Яковлевич. Информационные технологии [Электронный ресурс]: учебник / Б. Я. Советов, В. В. Цехановский. - 6-е изд. - Москва: Юрайт, 2015. - х эл. опт. диск (CD-ROM)