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 ИС-32

Ситкин Е.В.

\_\_\_\_\_\_\_\_\_\_дата

2021-2022 уч.г.

План прохождения учебной практики

по ПМ.03 «Ревьюирирование программных продуктов»

Студент Е.В.Ситкин

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 | 1.12 |  |
| Дизассемблирование | 2.12 |  |
| Разработать сопроводительную документацию | 3.12 |  |
| Защита практики | 4.12 |  |

**Отзыв руководителя**

Студент

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](#_Toc88849630)

[1. Список терминов и определений 7](#_Toc88849631)

[2. Функциональные требования 8](#_Toc88849632)

[3. Измерение производительности приложения посредством анализа использования ЦП 9](#_Toc88849633)

[3.1. Сбор данных профилирования 9](#_Toc88849634)

[3.2. Анализ данных о загрузке ЦП 12](#_Toc88849636)

[4. Установка ПО 14](#_Toc88849637)

[4.1. Установка Linux Ubuntu 14](#_Toc88849638)

[4.2. Установка остального ПО через терминал 20](#_Toc88849639)

[5. Обратное проектирование 22](#_Toc88849640)

[6. Дизассемблирование 23](#_Toc88849641)

[ЗАКЛЮЧЕНИЕ 28](#_Toc88849642)

[ПРИЛОЖЕНИЕ 1 29](#_Toc88849643)

**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 ЦП,
* установить ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования,
* выполнить обратное проектирование,
* выполнить дизассемблирование.

# **Список терминов и определений**

**Пользователь**— человек, который имеет, имел, или, возможно, будет иметь доступ в систему для совершения операций.

**Операция** — совокупность действий, составляющих содержание одного акта бизнес-деятельности. Операция должна соответствовать требованиям ACID (Atomicity, Consistency, Isolation, Durability). Совокупность операций одного модуля представляет интерфейс взаимодействия клиент-сервер этого модуля.

**Центральный процессор** (ЦП) – главная часть аппаратного обеспечения компьютера или программируемого логического контроллера. Иногда называют микропроцессором или просто процессором.

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

**Прокси** – это удаленный компьютер, который, при подключении к нему вашей машины, становится посредником для выхода абонента в интернет. Прокси передает все запросы программ абонента в сеть, и, получив ответ, отправляет его обратно абоненту.

**Репозиторий**(от англ. **repository** – склад, хранилище) – это профессиональный участник рынка ценных бумаг, осуществляющий ведение реестров договоров РЕПО и договоров с деривативами, заключенных на организованных торгах.

**GitHub** — это платформа, хранящая различные Git-репозитории на своих многочисленных серверах. Также GitHub называют крупнейшим веб-сервисом для хостинга и совместной разработки IT-проектов.

**Дизассемблирование** – это получение из исполняемого кода программы код на языке ассемблера.

**Дизассемблер** - программа, осуществляющая дизассемблирова­ние.

**Компилятор** — программа, переводящая текст, написанный на языке программирования, в набор машинных кодов.

**Декомпиляция** – это процесс восстановления исходного кода программы из машинного кода.

**Декомпилятор** – это программа, которая может совершить процесс декомпиляции.

**Обратное проектирование** – исследование некоторого готового устройства или программы, а также документации на него с целью понять принцип его работы; например, чтобы обнаружить недокументированные возможности, сделать изменение или воспроизвести устройство, программу или иной объект с аналогичными функциями, но без прямого копирования.

# **Функциональные требования**

ПО, используемое в ходе выполнения практики:

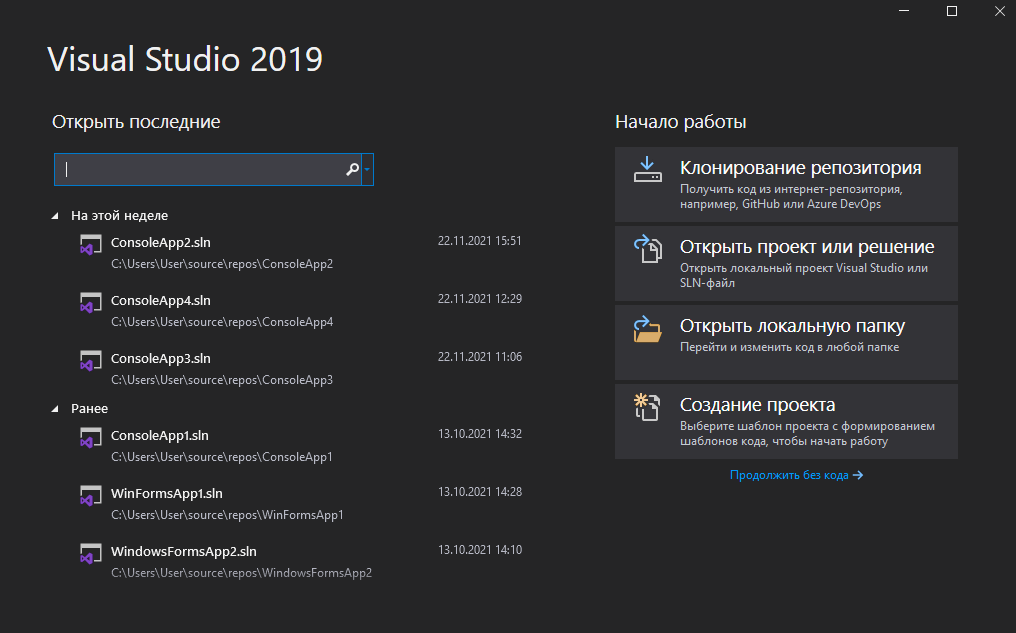
* Visual Studio 2016
* Oracle VM Virtual Box – программа для виртуализации ОС, в которой нужно создать виртуальную ОС Linux с дистрибутивом Ubuntu 64x
* Компилятор GCC - компилятор для различных языков программирования, используемый для свободных Unix-подобных операционных систем.
* Radare2 - свободный кроссплатформенный фреймворк для реверс-инжиниринга, который используется при реверсе, отладке вредоносного ПО и прошивок.
* Iaito - это одна из оболочек Radare2, предназначенная для платформы Linux и устанавливаемая при помощи терминала с веб-сайта GitHub.
* Декомпилятор r2ghidra

1. **Измерение производительности приложения посредством анализа использования центрального процессора**

### Сбор данных профилирования

* + 1. Запускаем Visual Studio, открываем любой проект

Рис. 1 «Visual Studio»



3.1.2. Установите точку останова в приложении в точке, где вы хотите проверить загрузку центрального процессора.



Рис. 2 «Первая точка остановы»

3.1.3. Устанавливаем вторую точку останова в конце функции или области кода, который требуется проанализировать.

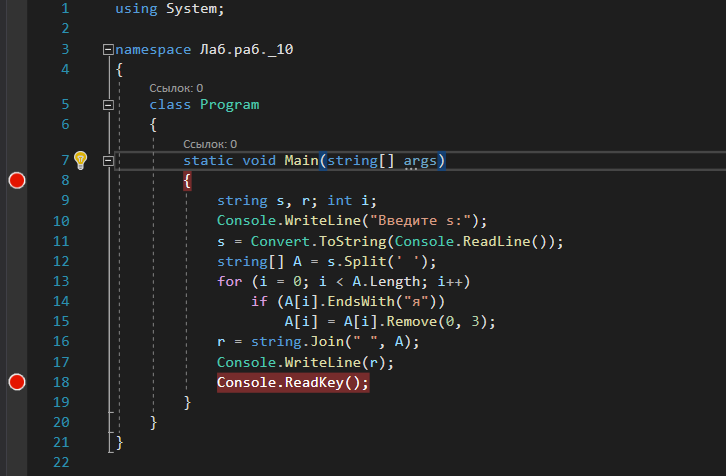


Рис. 3 «Вторая точка остановы»

3.1.4. Окно **Средства диагностики** появится автоматически, если вы не отключали эту функцию. Чтобы снова открыть окно, щелкните **Отладка > Окна > Показать средства диагностики.**

3.1.5. Вы можете выбрать, что следует просмотреть, Использование памяти или Использование ЦП (либо оба средства), с помощью параметра Выбор средств на панели инструментов.

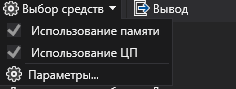


Рис. 4 «Выбор средств»

3.1.6. По завершении загрузки приложения отображается представление "Сводка" средств диагностики.

3.1.7. Запустите сценарий, который вызвал срабатывание первой точки останова.

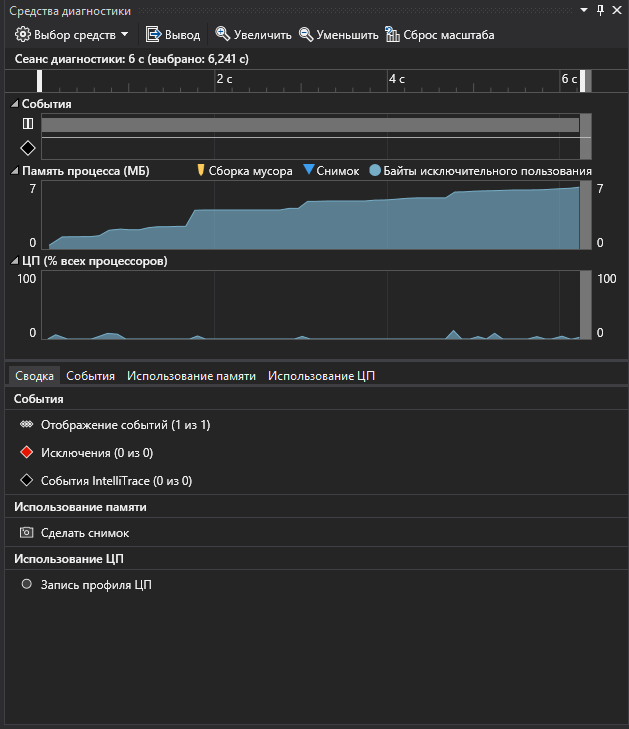


Рис. 5 «Окно диагностики во время отладки»

3.1.8. Приостановив отладчик, включите сбор данных о загрузке центрального процессора, а затем откройте вкладку **Загрузка ЦП**. При выборе пункта Запись профиля ЦП Visual Studio начнет записывать функции и сведения о времени их выполнения. Эти собранные данные можно просматривать только в том случае, если приложение останавливается в точке останова.

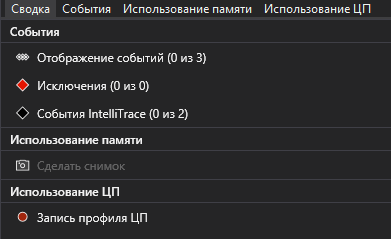


Рис. 6 «Запись профиля ЦП»

3.1.9. Нажмите клавишу F5, чтобы запустить приложение до второй точки останова.

Теперь у вас есть данные о производительности приложения именно для той области кода, которая выполняется между двумя точками останова.

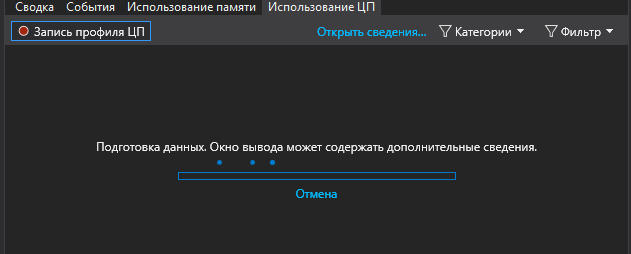
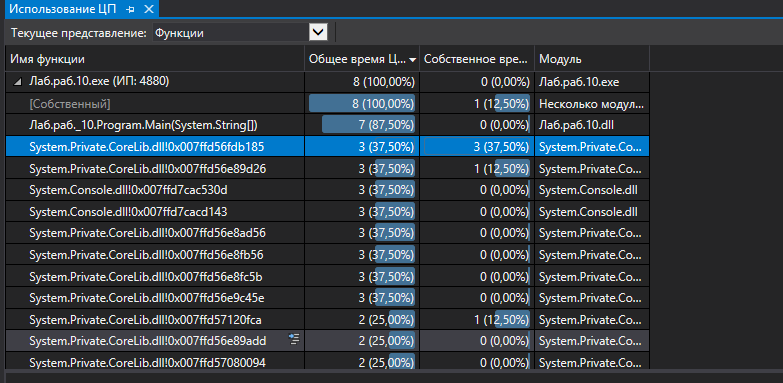
Профилировщик начинает подготавливать данные потока. Дождитесь завершения этой операции.

Рис. 7 «Подготовка данных»

Средство "Загрузка ЦП" выведет отчет на вкладке **Загрузка ЦП**.



Рис. 8 «Загрузка центрального процессора»



### 

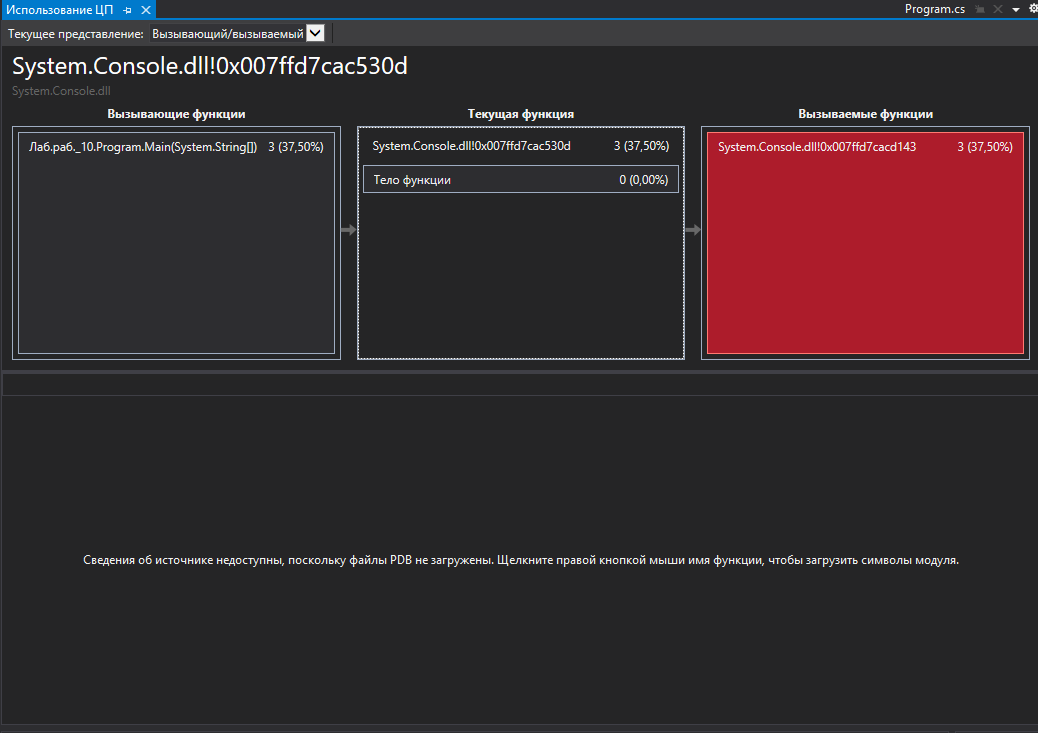
Рис. 9 «Список функций»

### 3.2. Анализ данных о загрузке центрального процессора

3.2.1. В списке функций дважды щелкните одну из функций вашего приложения, которая выполняет много работы.

При двойном щелчке функции в левой панели откроется представление **Вызывающий/вызываемый**.

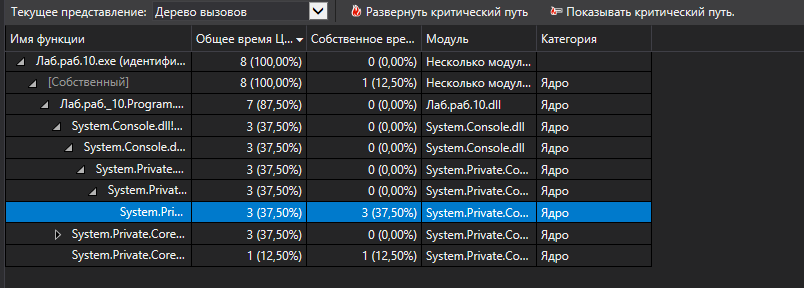
Рис. 10 «Представление **Вызывающий/вызываемый**»



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.

3.2.2. Чтобы увидеть более обобщенное представление, показывающее порядок, в котором вызываются функции, выберите в раскрывающемся списке в верхней части панели пункт **Дерево вызовов**.

Рис. 11 «Дерево вызовов»



Чтобы увидеть вызовы функций, которые используют самый высокий процент ЦП в представлении дерева вызовов, нажмите «**Развернуть критический путь»**.

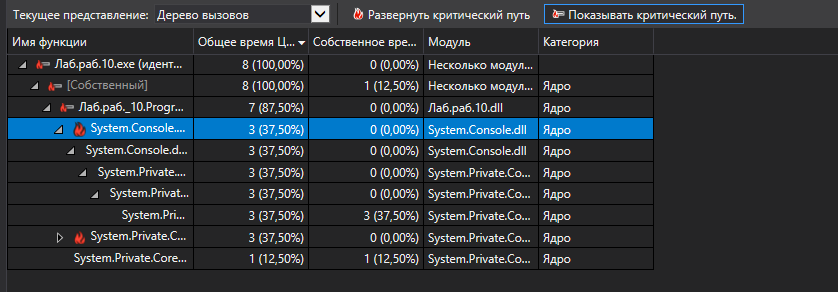
****

Рис. 12 «Развернули критически путь»

Просмотр внешнего кода

Чтобы посмотреть пути к вызовам внешнего кода, выберите **Показать внешний код** в списке **Представление фильтра** и выберите **Применить**.

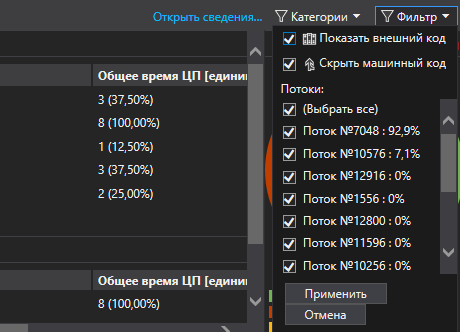


Рис.13 «Показать внешний код»

1. **Установка ПО**
   1. Установка Linux Ubuntu

Для начала нужно установить Virtual Box что бы на нём поставить Linux.

После нажимаем создать, указываем имя, тип, версию, объем памяти,

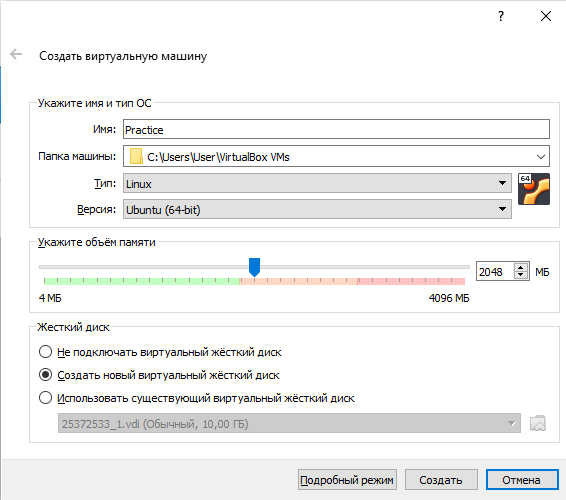


Рис. 14 «Настраиваем машину»

Создаём новый жесткий диск

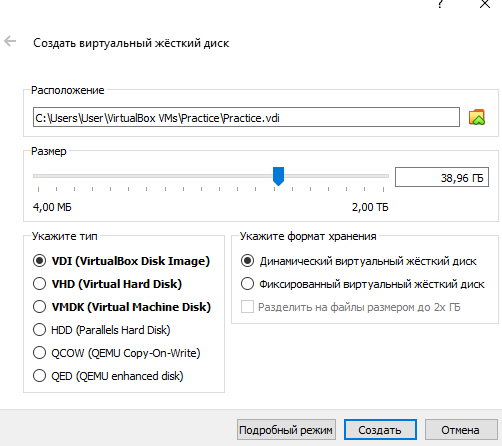


Рис. 15 «Новый жесткий диск»

Затем в настройках нужно поставить контроллер

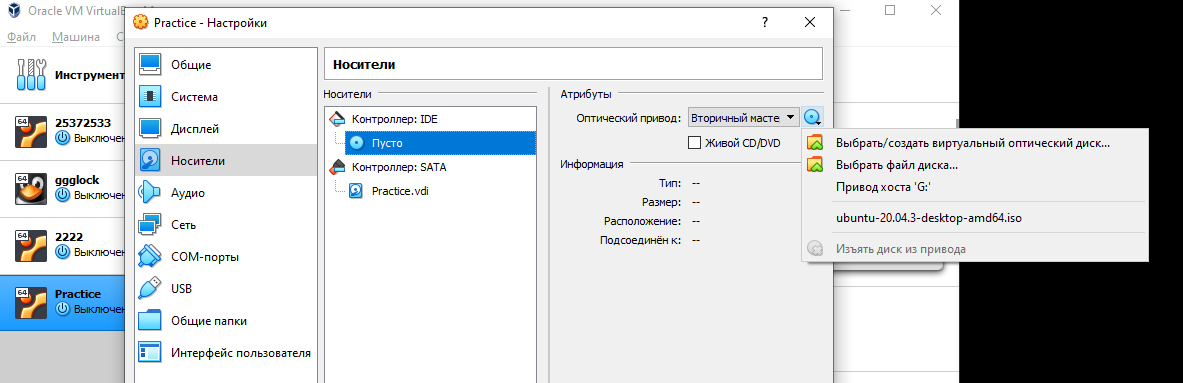


Рис. 16 «Выбираем привод диска»

Запускаем нашу машину

После загрузки можно выбрать язык и нажимаем «Install Ubuntu»

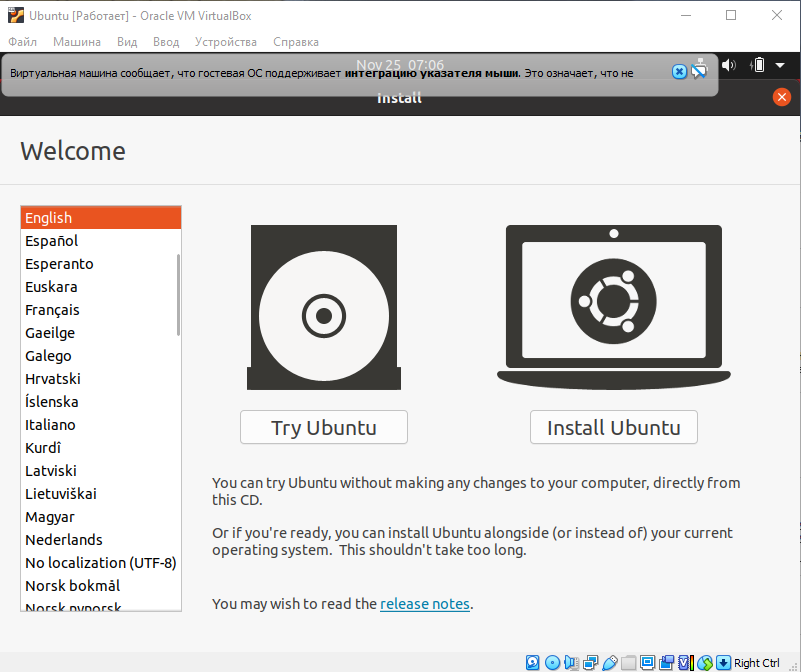


Рис. 17 «Выбираем Install Ubuntu»

Выбираем «Minimal installation» и нажимаем продолжить

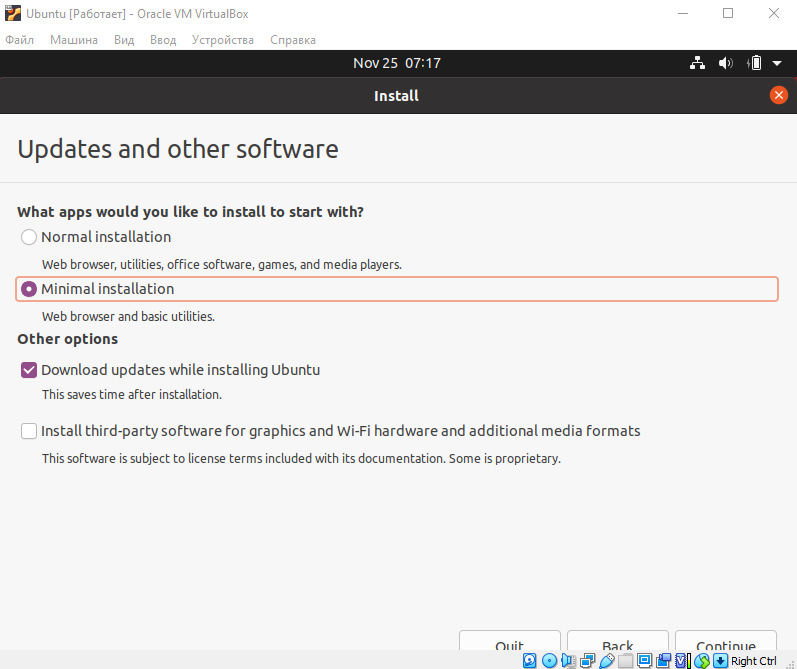


Рис.18 «Выбираем **Minimal installation**»

Далее ничего не меняем и продолжаем

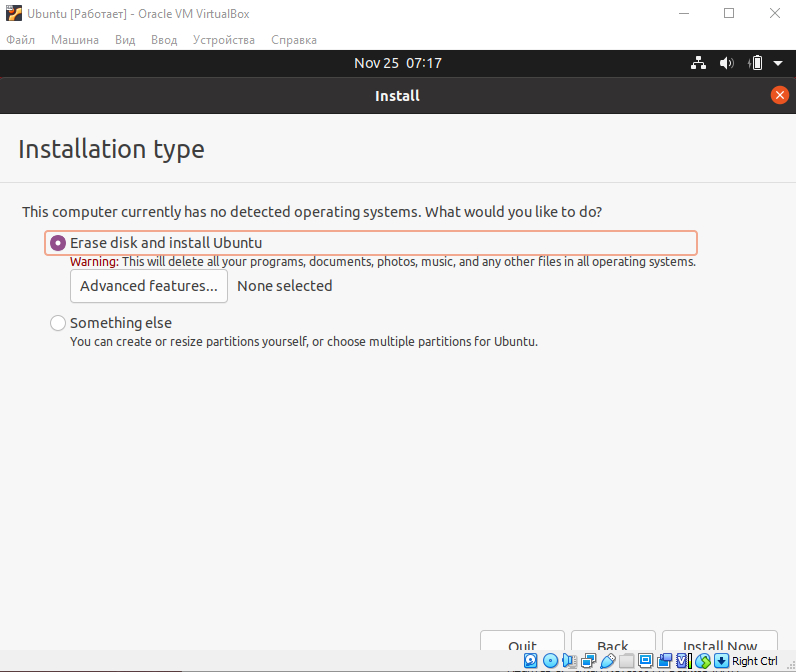


Рис.19 «Installation type»

Выбираем свой регион



Рис. 20 «Регион»

Далее вводим свой логин и пароль

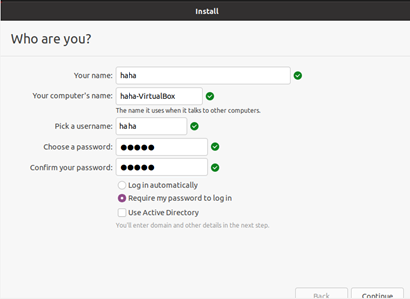


Рис. 21 «Логин и пароль»

Далее происходит установка

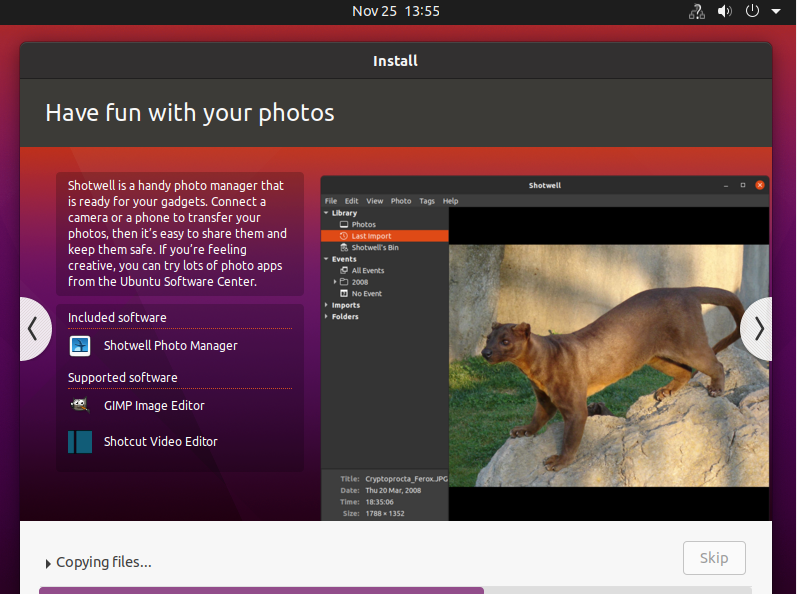
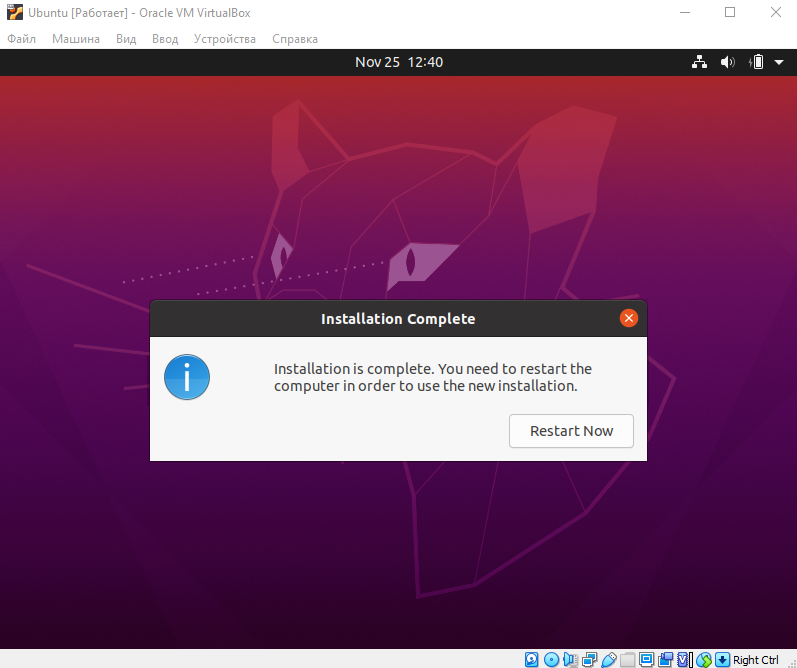


Рис. 22 «Установка»

После установки нужно перезапустить машину

Рис. 23 «Перезагрузка»



* 1. Установка остального ПО через терминал
     1. Устанавливаем компилятор GCC и все необходимые для него компоненты

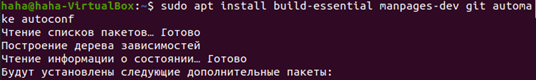
с

Рис. 24 «Установка GCC»

* + 1. Установка radare2

Устанавливаем необходимые пакеты для работы с radare2

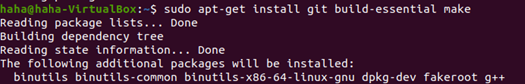


Рис. 25 «Установка пакетов»

Так как используется компьютер в колледже прописываем:

https://cdn.discordapp.com/attachments/890271394011697176/915984742325846026/erer.png

Рис. 26 «Прописываем прокси»

sudo nano /etc/wgetrc

Ищем прокси и меняем на:

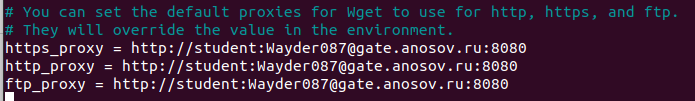
****

Рис. 27 «Меняем прокси»

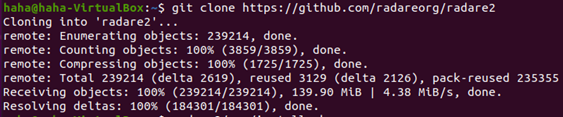
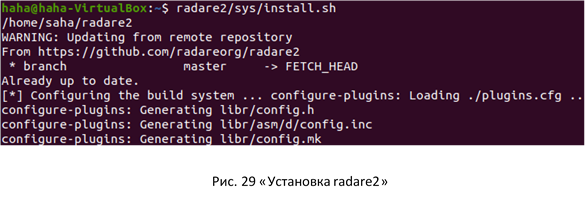
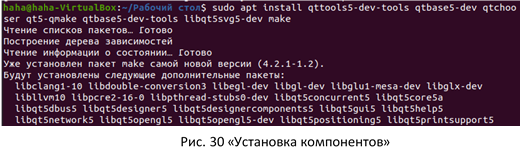


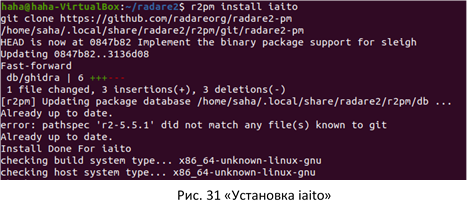
Рис. 28 «Установка radare2»



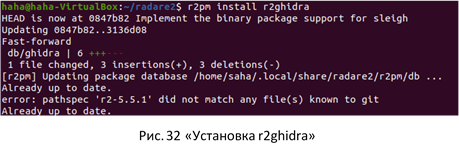
* + 1. Устанавливаем необходимые компоненты для граф.оболочки «Iaito»



Устанавливаем графический интерфейс «iaito» (установка может выполняться из любого места)



* + 1. Устанавливаем декомпилятор «r2ghidra»



# **Обратное проектирование**

* 1. Создаем файл с кодом для компиляции

https://cdn.discordapp.com/attachments/890271394011697176/915990109965393940/ererer.png

Рис. 33 «Создаём файл»

* 1. Открываем его и вставляет туда код для теста

include <stdio.h>

int main (void)

{

int x = 0;

while(x < 4){

printf ("Guten Morgen!\n");

x = x + 1;

}

return 0;

}

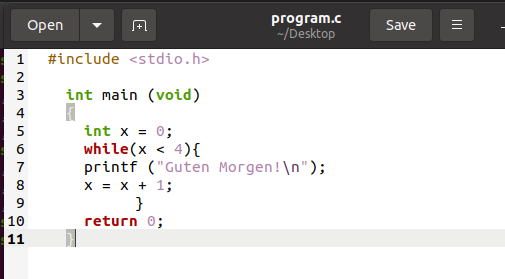
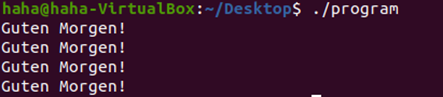


Рис. 34 «Код»

* 1. Компилируем

https://cdn.discordapp.com/attachments/890271394011697176/915990248960458862/trtrt.png

Рис. 35 «Компиляция»

* 1. Открываем и проверяем наш файл

# **6. Дизассемблирование**

Рис. 36 «Проверка файла»

6. 1. Открываем папку и открываем наш файл

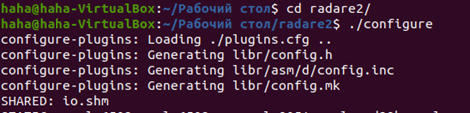


Рис. 37 «Открываем файл»

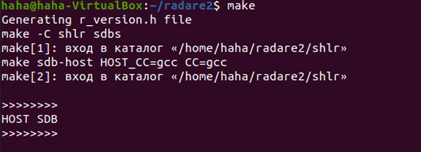


Рис. 38 «Открываем файл»

6.2. Запускаем «Iaito»

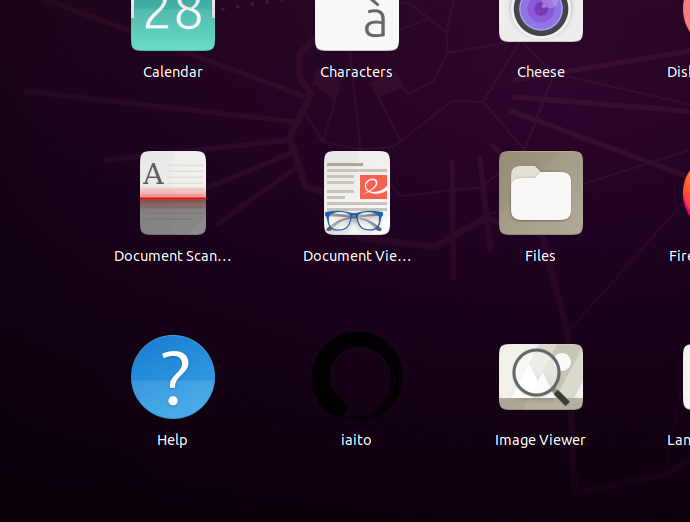


Рис. 39 «Приложение Iaito»

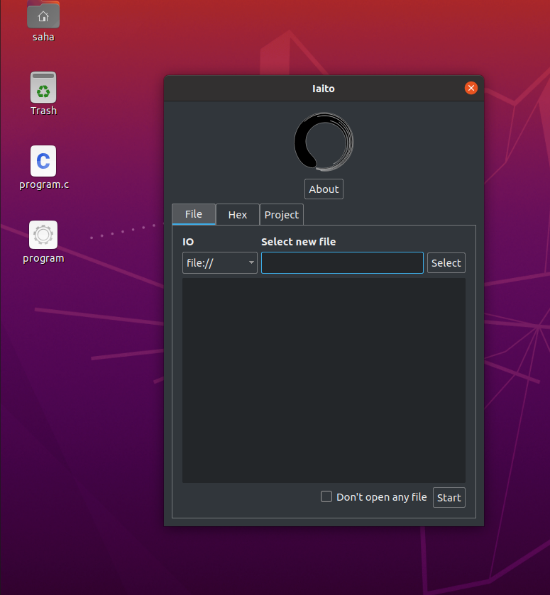


Рис. 40 «Интерфейс»

6.3. Открываем тестовый файл в «radare2». Настройки оставляем стандартные.

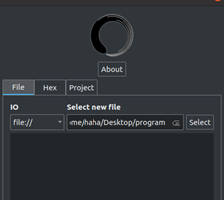


Рис. 41 «Выбираем файл»

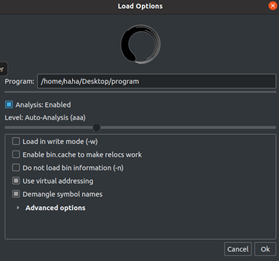


Рис. 42 «Настройки»

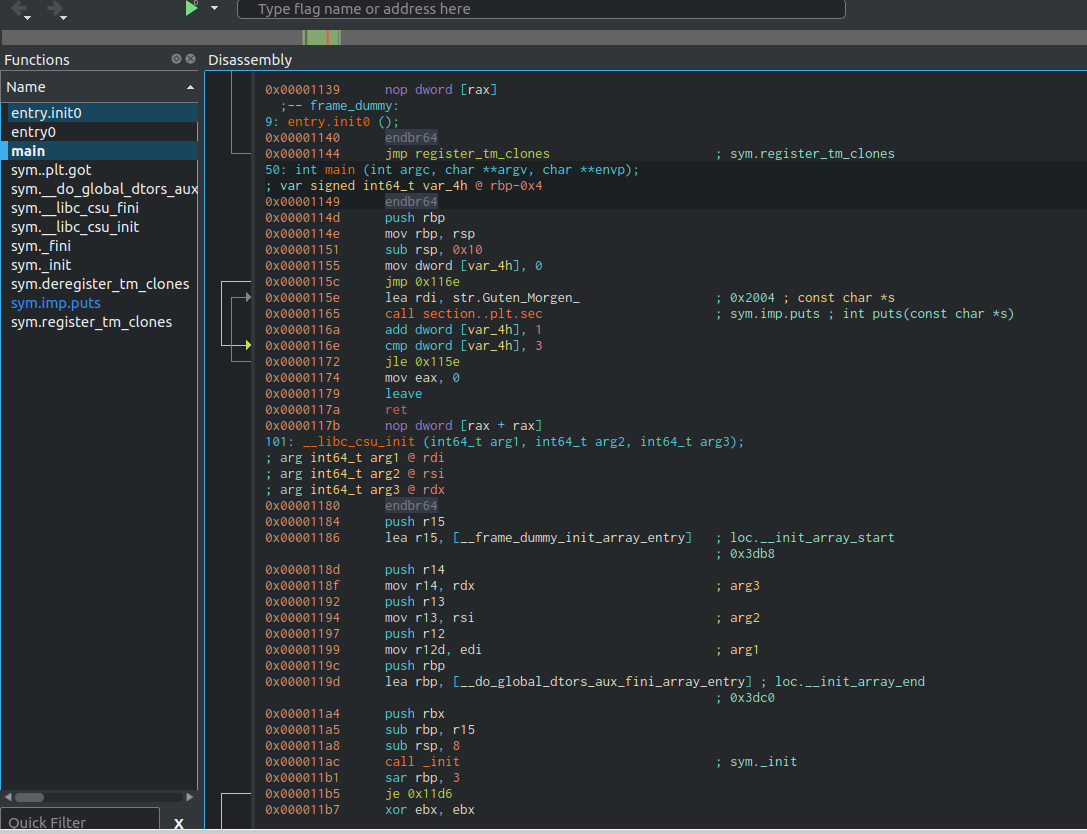
6.4. Открываем слева «main»

Рис .43 «Код»

6.5. Открываем вкладку декомпилятора. Видим, что код практически читаем, кроме переменных

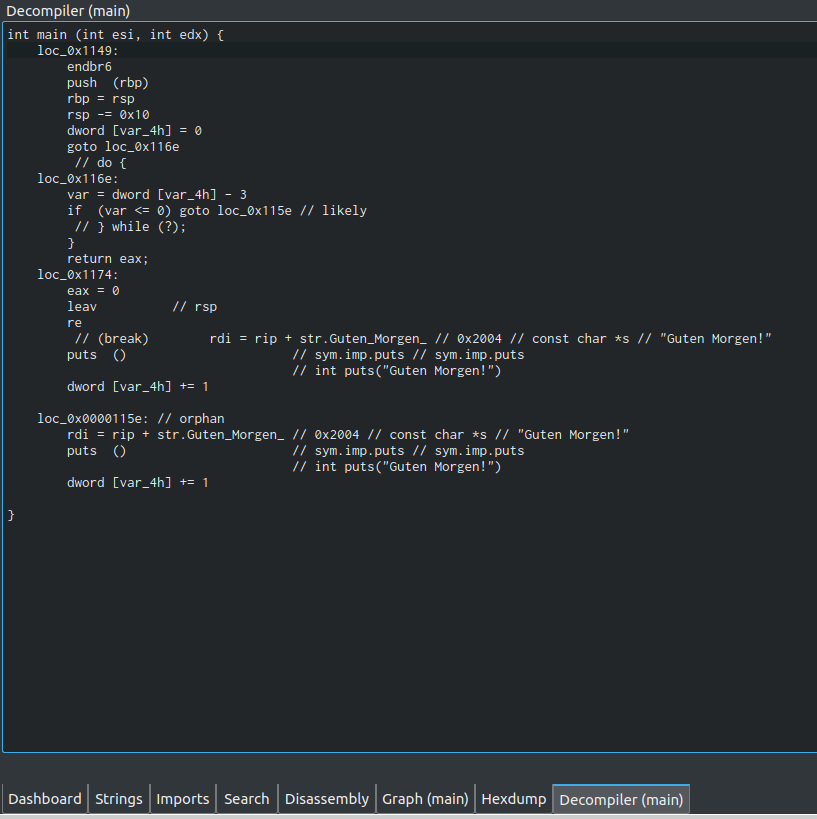


Рис. 44 «Декомпилятор»

6.6. Меняем режим работы программы на Cache mode

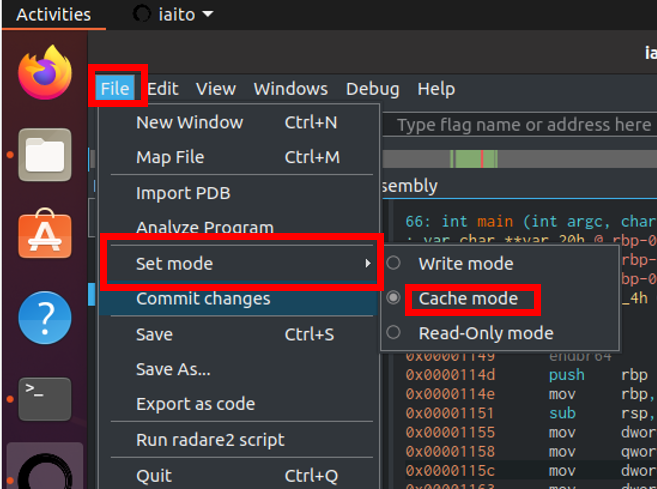


Рис. 45 «Cache mode»

6.7. Так как фраза «Guten Morgen!» выводится 4 раза, а нам необходимо увеличить вывод до 10 раз, то смотря на код, можно увидеть, что используется цикл «while» по значению переменной [var4\_h] равной 3 (т.к. While <4, а цикл начинается с 0). Выделяем цифру 3 и нажимаем «Edit» затем «Instruction» меняя значение переменное с 3 до 9.

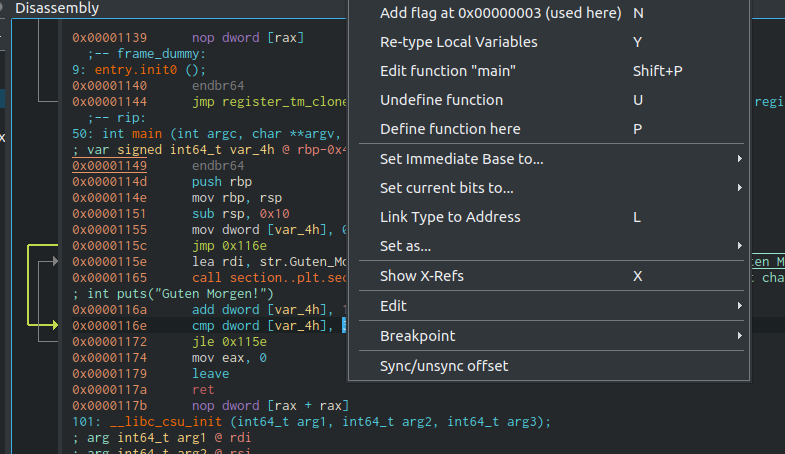


Рис. 46 «Edit>Instruction»

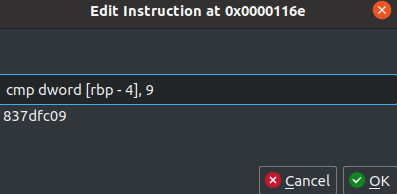
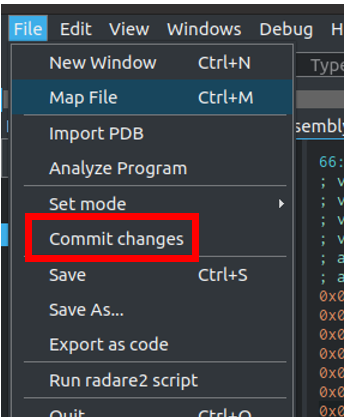


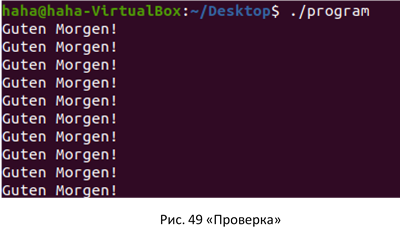
Рис. 47 «Изменяем значение»

6.8. Сохраняем наши изменения, нажав на кнопку «Commit changes»

Рис. 48 «Применяем изменения»



6.9. Запускаем файл для проверки



# **ЗАКЛЮЧЕНИЕ**

Целью учебной практики по ПМ.03 «Ревьюирирование программных продуктов» являлось, 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 выполнения заданий на учебную практику было выполнено:

1. Изучены такие термины как, обратное проектирование, репозиторий, дизассемблирование и т.д.
2. Измерена производительность приложения посредством анализа использования ЦП. Здесь мы узнали, как собирать и анализировать данные об использовании ЦП.
3. Выполнено обратное проектирование.
4. Выполнено дизассемблирование, где было произведено изменение кода с помощью дизассемблера.

# **ПРИЛОЖЕНИЕ 1**

**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. 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. Проектирование информационных систем. Лекция 1. Презентация [Электронный ресурс]. - Москва: Национальный Открытый Университет «ИНТУИТ», 2014. - 27 с.
7. 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)
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. Страбыкин, Дмитрий Алексеевич. Организация ЭВМ: лабораторный практикум на компьютерах: учеб. пособие для студентов направления подготовки 09.03.01 (230100.62) / Д. А. Страбыкин; ВятГУ, ФАВТ, каф. ЭВМ. - 3-е изд., перераб. и доп. - Киров: [б. и.], 2013. - 62 с.