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\_\_

\_\_\_\_\_\_\_\_\_\_\_\_В.А. Морозова

\_\_\_\_\_\_\_\_\_\_\_дата

2021-2022 уч.г.

План прохождения учебной практики

по ПМ.03 «Ревьюирирование программных продуктов»

Студент \_\_Морозова Виктория Алексеевна\_

Группа\_\_ИС-32\_\_

Специальность \_\_«09.02.07» Информационные системы и программирование

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

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

Сроки проведения практики\_\_\_\_22.11.-27.112021\_\_\_\_\_\_\_

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 | 30.12 |  |
| Обратное проектирование. | 01.12 |  |
| Дизассемблирование. | 02.12 |  |
| Разработать сопроводительную документацию | 03.12 |  |
| Защита практики | 04.12 |  |

**Отзыв руководителя**

Студент \_\_\_Морозова Виктория Алексеевна\_

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 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 | | |  |

Подпись руководителя:

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

\_\_\_\_\_\_\_\_\_\_\_\_\_дата

**Содержание**

Введение..............................................................................................5

1. Список терминов и определений....................................................7
2. Функциональные требования..........................................................9
3. Измерение производительности приложения посредством анализа использования ЦП..........................................................................10
4. Установка ПО.................................................................................14
5. Дизассемблирование......................................................................18
6. Обратное проектирование.............................................................19

Заключение..........................................................................................24

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

1. **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.

**Внешний код —** это функции в компонентах системы и платформы, которые исполняются вашим кодом.

**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.

1. **Функциональные возможности**

Характеристики ПК для выполнения практики:

* 64-разрядная операционная система, процессор x64
* От 2048 Mб оперативной памяти
* 200 Mб места на жестком диске для инсталляции
* процессор с частотой, согласно требованиям самой ОС, например в случае с Windows ХР это Pentium не менее 300 МГц или его аналог от AMD

1. **Измерение производительности приложения посредством анализа использования ЦП**

Ссылка на GutHub: https://github.com/Viktoria00069/Praktika.git

3.1 Сбор данных об использовании ЦП

3.1.1 Запускаем Visual Studio и открываем проект, который хотим проанализировать на нагрузку ЦП.

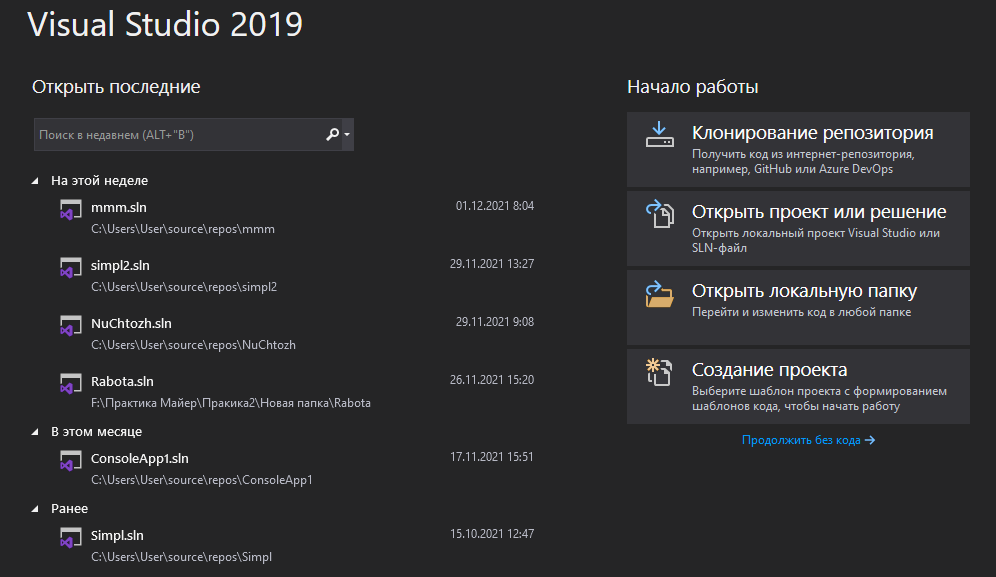


Рис.1 «Visual Studio»

3.1.2 Устанавливаем точку останова в месте, где хотим проанализировать код

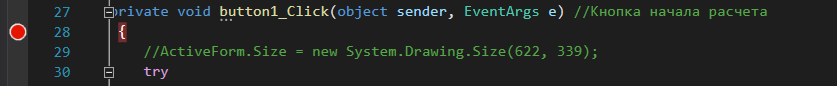


Рис. 2 «Первая точка останова»

3.1.3 Устанавливаем вторую точку останова в конце функции, которой требуется проанализировать



Рис. 3 «Вторая точка останова»

3.1.4 Откроем окно Средства диагностики. Что бы открыть щелкните Отладка > Окна > Показать средства диагностики

3.1.5 Нас главным образом интересует загрузка ЦП, поэтому убедитесь, что средство Загрузка ЦП включено (оно включено по умолчанию).

3.1.6 Щелкните Отладка > Начать отладку (Запустить на панели инструментов или F5).По завершении загрузки приложения отображается представление "Сводка" средств диагностики. Если вам нужно открыть окно, щелкните Отладка > Окна > Показать средства диагностики

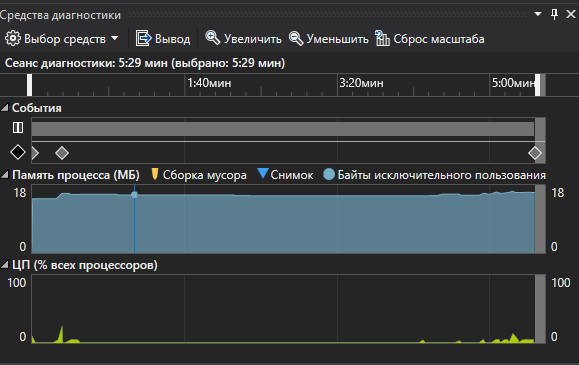


Рис. 4 «Средства диагностики»

3.1.7 Запустите сценарий, который вызвал срабатывание первой точки останова

3.1.8 Приостановив отладчик, включите сбор данных о загрузке ЦП, а затем откройте вкладку Загрузка ЦП



Рис. 5 «Запись профиля ЦП»

3.1.9 Нажмите клавишу F5, чтобы запустить приложение до второй точки останова. Профилировщик начинает подготавливать данные потока. Дождитесь завершения этой операции

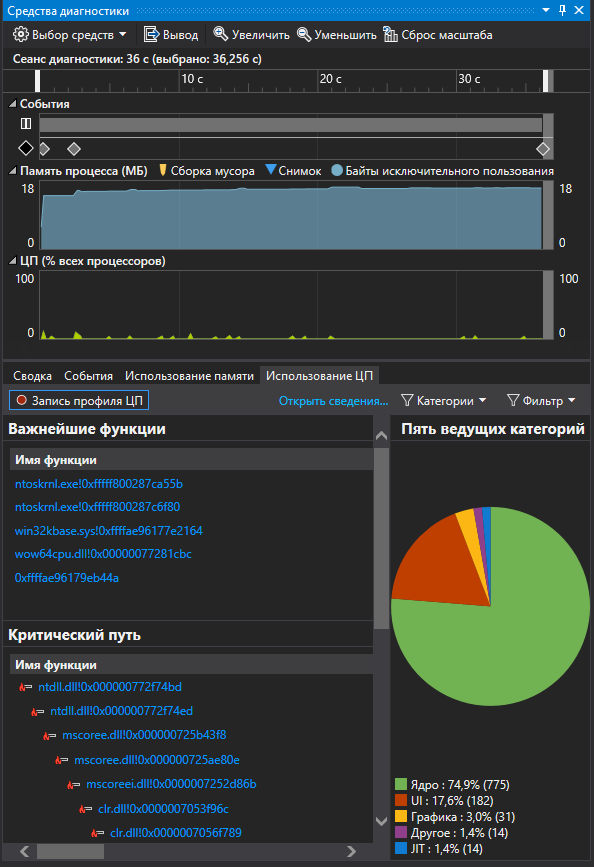


Рис. 6 «Загрузка профиля ЦП»

3.2 Анализ данных о загрузке ЦП

3.2.1 В списке функций изучите функции, которые выполняют большую часть работы

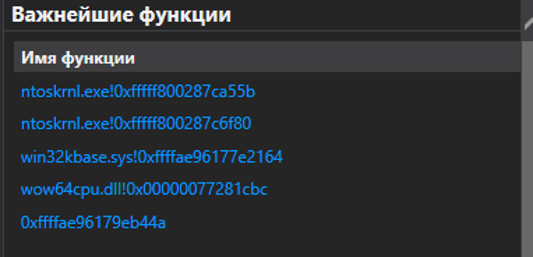


Рис. 7 «Список функций»

3.2.2 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

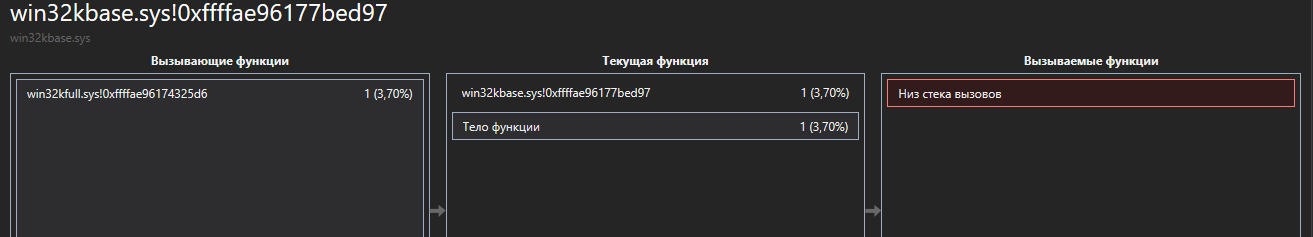


Рис. 8 «Вызывающий/вызываемый»

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

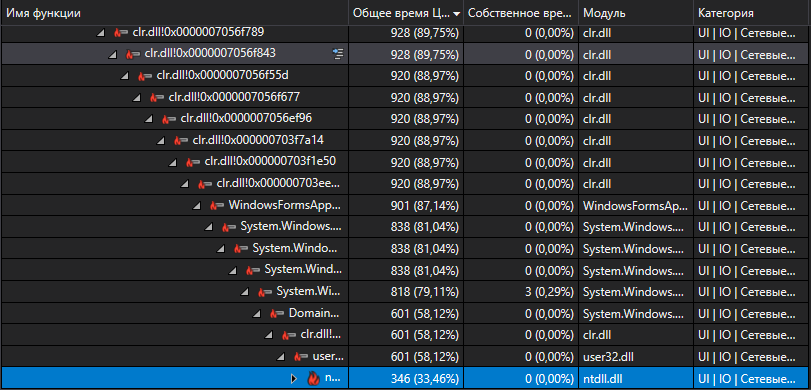


Рис. 9 «Дерево вызовов»

3.2.4 Чтобы увидеть вызовы функций, которые используют самый высокий процент ЦП в представлении дерева вызовов, нажмите Развернуть критический путь

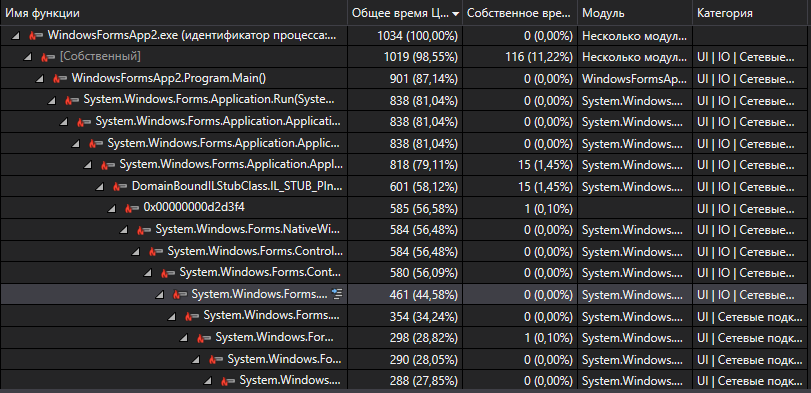


Рис. 10 «Критический путь»

3.2.5 Если вы захотите посмотреть пути к вызовам внешнего кода, выберите Показать внешний код в списке Представление фильтра и выберите Применить.

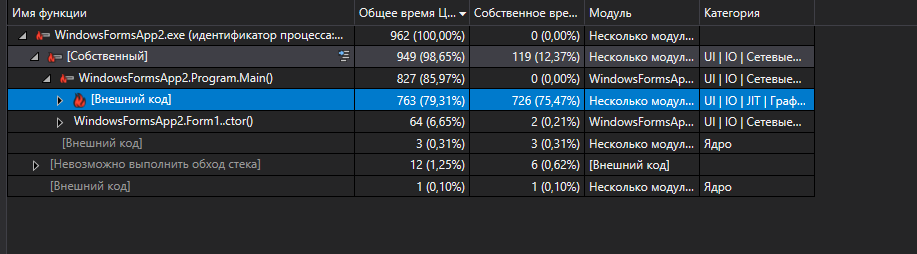


Рис. 11 «Внешний код»

Вывод: мы провели анализ программы на нагрузку ЦП несколько раз и увидели, как меняется нагрузка при упрощении кода

1. Установка ПО

4.1 Запускаем Virtual box

4.2 Нажимаем создать

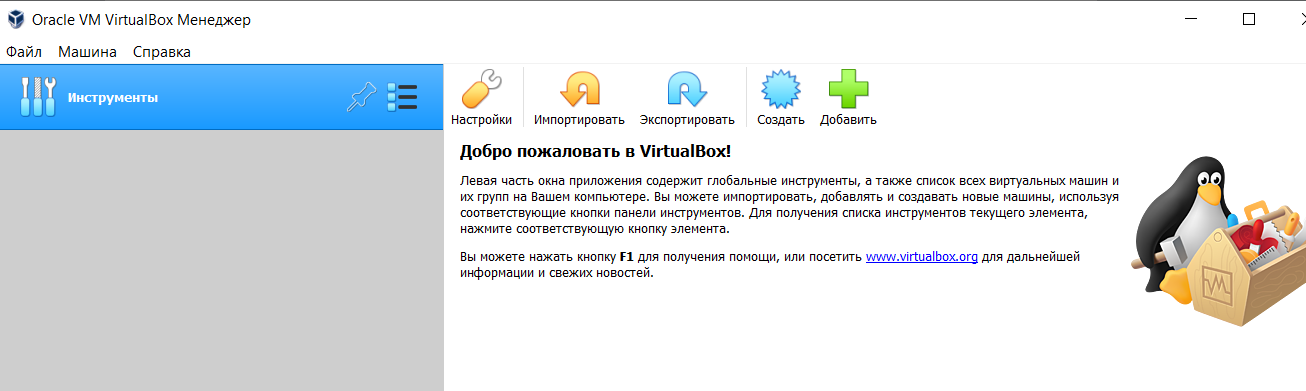


Рис. 12 «Кнопка создать»

4.3 Называем виртуальную машину (рекомендуется назвать Ubuntu так она сразу выберет, какую систему устанавливать)

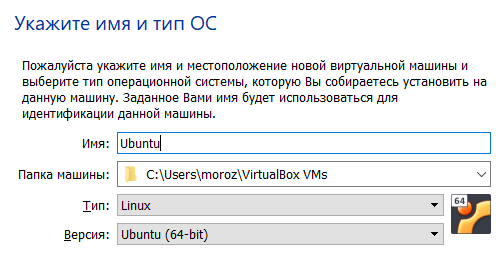


Рис. 13 «Указываем имя и тип»

4.4 Указываем объем оперативной памяти, которую хотим выделить для виртуальной машины (чем больше оперативной памяти, тем быстрее будет работать виртуальная машина)

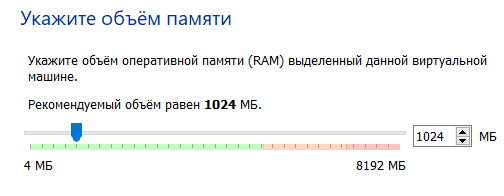


Рис. 14 «Выбор количества оперативной памяти»

4.5 Создаем новый виртуальный диск

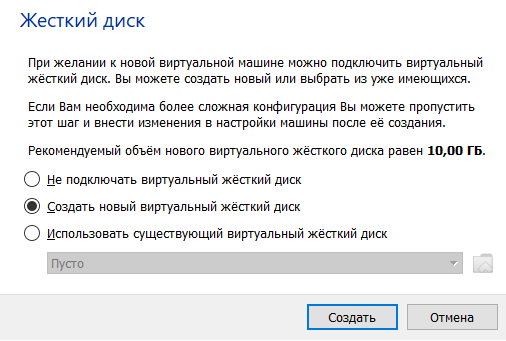


Рис. 15 «Жесткий диск»

4.6 Указываем тип

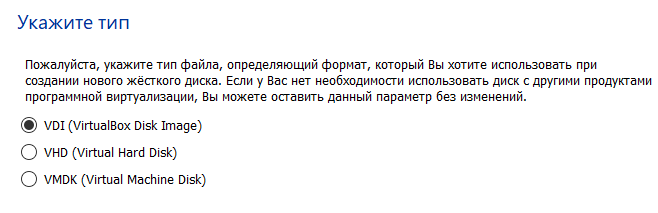


Рис. 16 «Тип файла»

4.7 Указываем формат хранения, выбираем «динамический виртуальный жесткий диск»

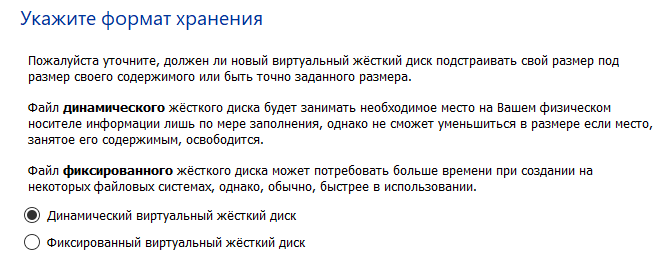


Рис. 17 «Формат хранения»

4.8 Указываем размер файла (оставляем как стоит)

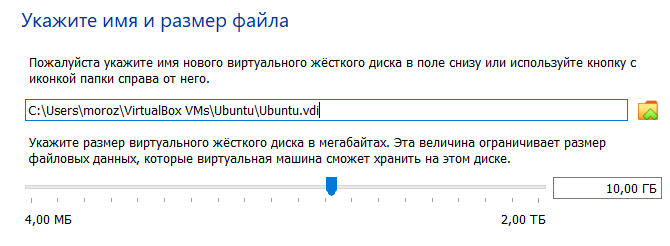


Рис. 18 «Размер файла»

4.9 Заходим в настройки

4.10 Заходим в носители и удаляем контроллер, где написано пусто

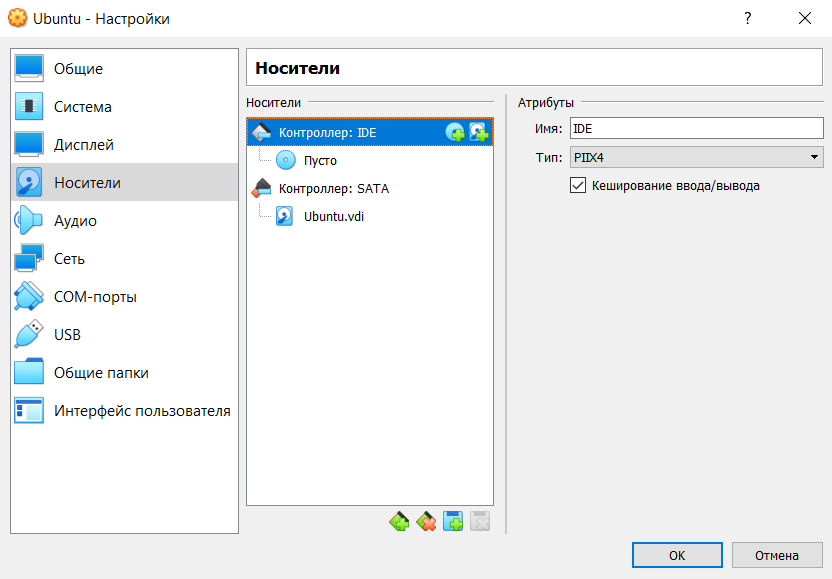


Рис. 19 «Носители»

4.11 Нажимаем на диск и выбираем образ системы

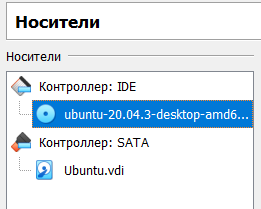


Рис. 20 «Выбранный образ»

4.12 Запускаем (пойдет установка Linux)

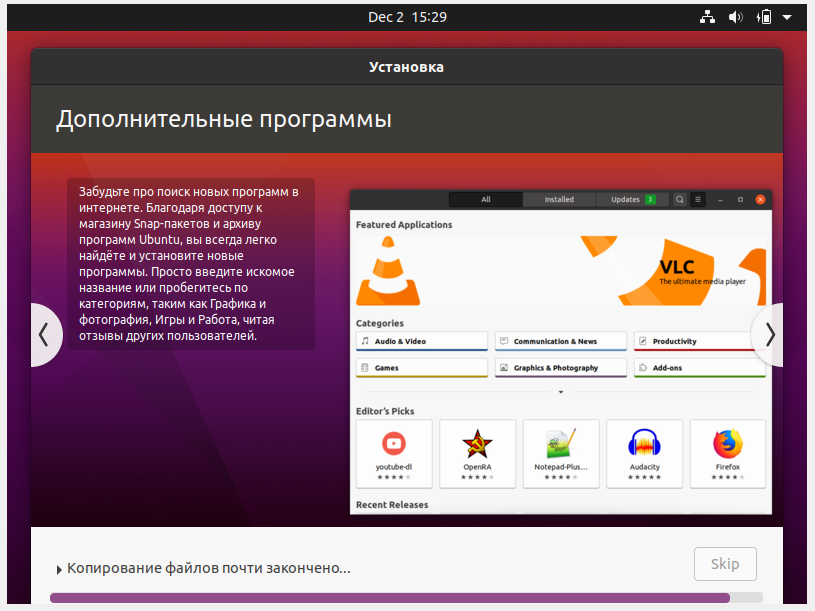


Рис. 21 «Установка»

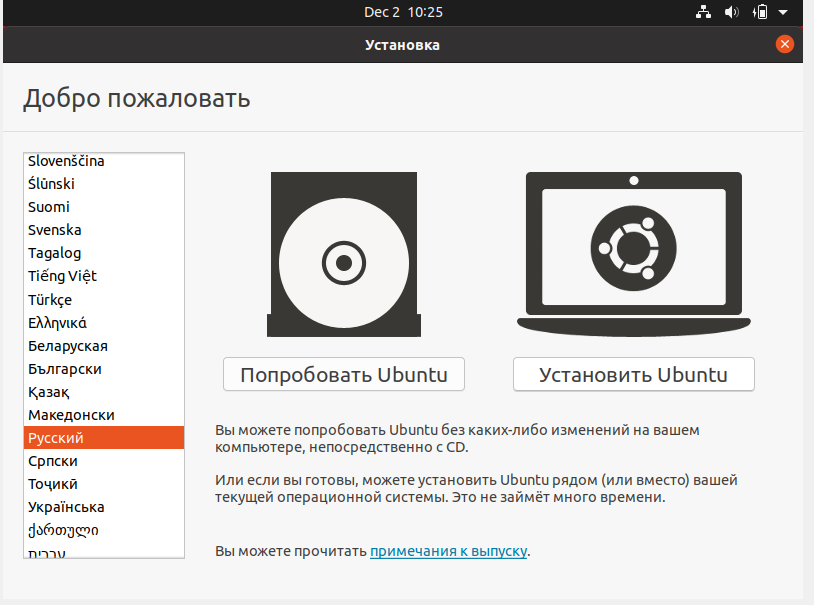


Рис. 22 «Настройка»

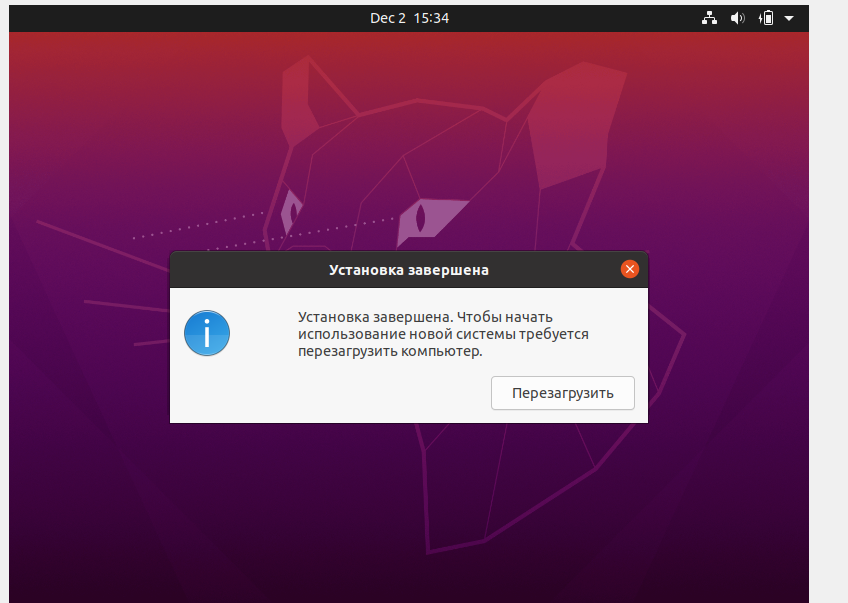


Рис. 23 «Перезагрузка»

4.13 Устанавливаем компилятор GCC и все необходимые для него компоненты

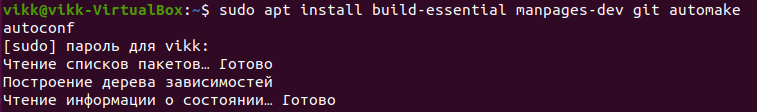


Рис. 24 «Установка компилятора»

1. Дизассемблирование

5.1 Устанавливаем компилятор GCC и все необходимые для него компоненты

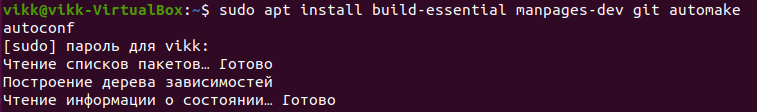


Рис. 25 «Установка GCC»

5.2 Устанавливаем необходимые пакеты для работы с radare2

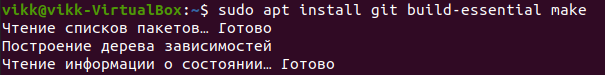


Рис. 26 «Установка пакетов для radare2»

5.3 Установка radare2

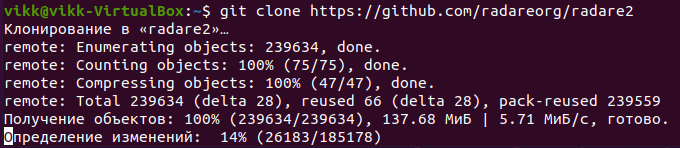


Рис. 27 «Клонирование с GitHud»



Рис. 28 «Установка radare2»

5.4 Устанавливаем необходимые компоненты для графической оболочки «Iaito»

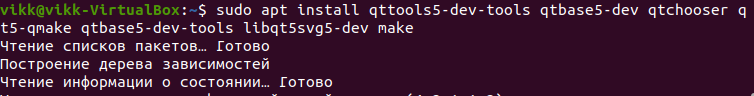


Рис. 29 «Установка пакетов для Iaito»

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



Рис. 30 «Установка графического интерфейса»

5.6 Устанавливаем декомпилятор «r2ghidra» (установка может выполняться из любого места)



Рис. 31 «Установка декомпилятора r2ghidra»

5.7 Запускаем "Iaito"

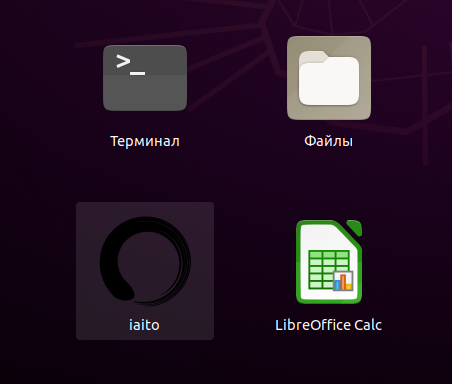


Рис. 32 «Иконка Iaito»

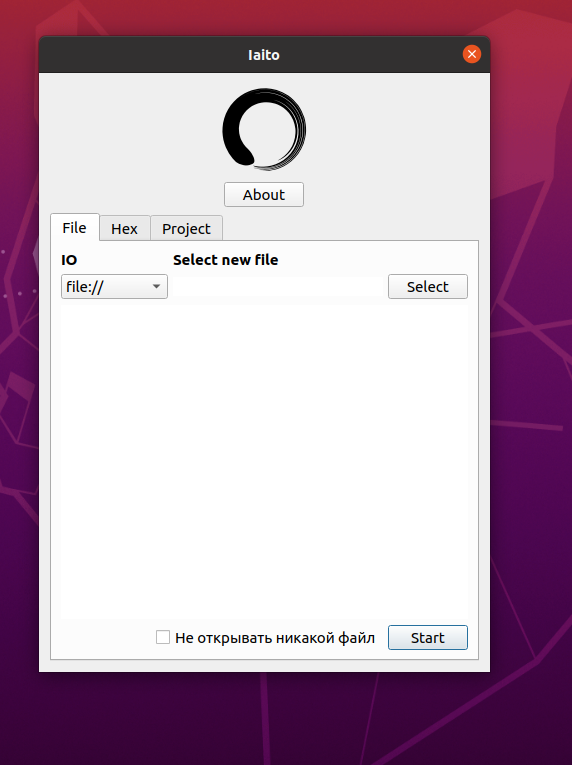


Рис. 33 «Интерфейс»

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

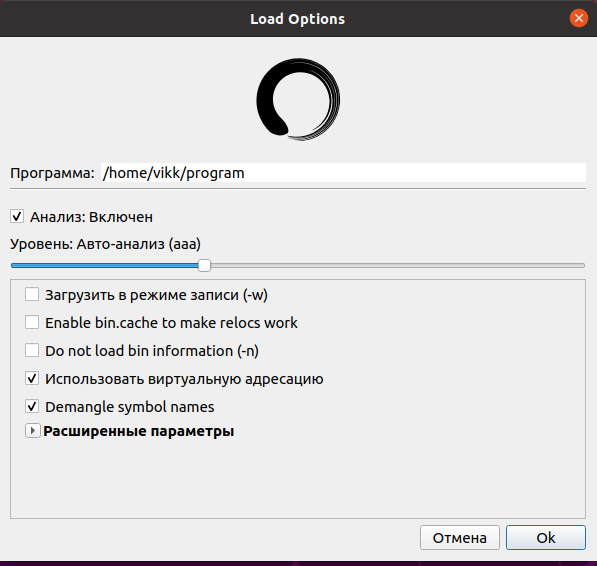


Рис. 34 «Настройки»

5.9 Открываем "main"

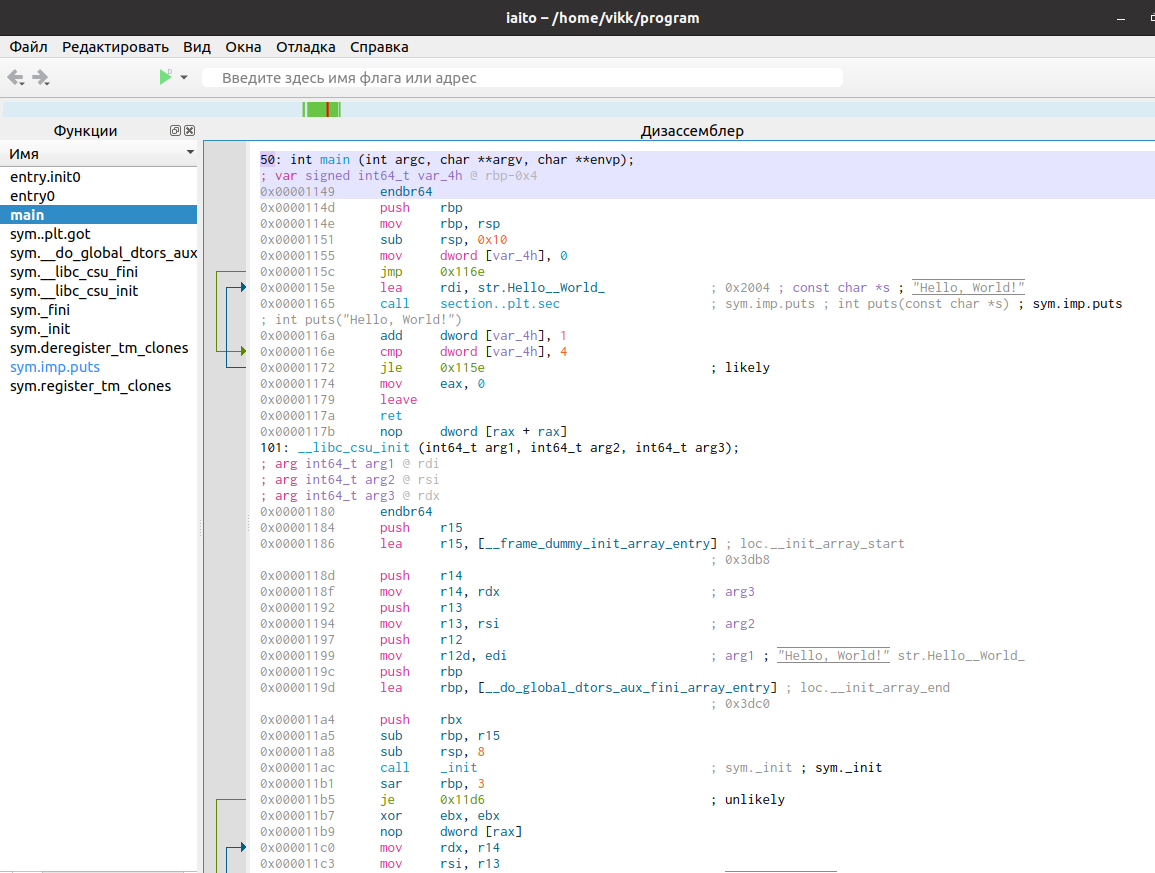


Рис. 35 «Main»

5.10 Нам необходимо увеличить вывод до 10 раз, то смотря на код, можно увидеть, что используется цикл «while» по значение переменной [var4\_h] равной 4 (т.к While <=4, а цикл начинается с 0). Выделяем цифру 4 и нажимаем «Edit» меняя значение переменное с 4 до 10.

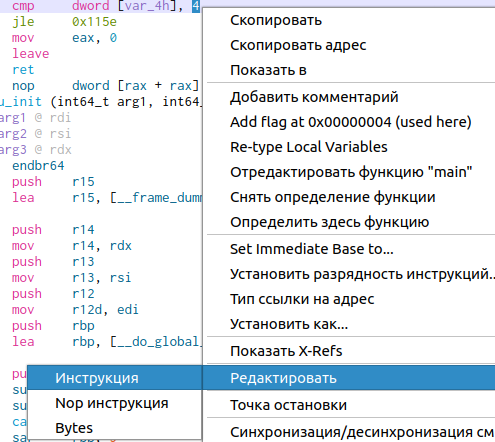


Рис. 36 «Редактируем»

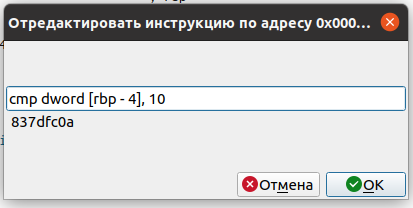


Рис. 37 «Настраиваем количество»

5.11 Сохраняем наши изменения, нажав на кнопку «Commit changes»

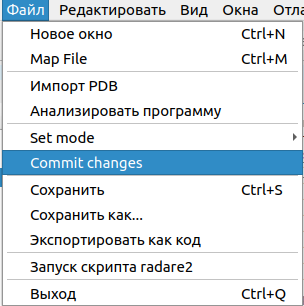


Рис. 38 «Коммитим»

5.12 Запускаем файл для проверки

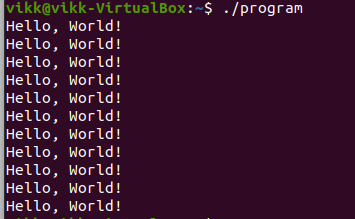


Рис. 39 «Результат»

1. Обратное проектирование

6.1 Создаем файл с кодом для компиляции



Рис. 40 «Файл»

6.2 Открываем его и вставляем туда код для теста

Изображение выглядит как текст

Автоматически созданное описание

Рис. 41 «Файл с кодом для теста»

6.3 Компилируем, открываем и проверяем файл

Изображение выглядит как текст

Автоматически созданное описание

Рис. 42 «Компиляция и открытие файла»

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 и инспекцией кода в Visual Studio, совместно работать над проектом в системе контроля версий GIT.

В ходе выполнения заданий на учебную практику я научилась выполнять прямое и обратное проектирование ПО, овладела навыками приостановки и возобновления работы с рабочими задачами иинспекцией кода в Visual Studio, совместно работала над проектом в системе контроля версий GIT.

А также в ходе учебной практики для достижения цели, выполнила задачи:

* измерила производительность приложения посредством анализа использования ЦП,
* установила ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования,
* выполнила дизассемблирование,
* выполнила обратное проектирование.

**ЛИТЕРАТУРА**

1. 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.
2. 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)
3. 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)
4. Проектирование информационных систем. Лекция 1. Презентация [Электронный ресурс]. - Москва: Национальный Открытый Университет «ИНТУИТ», 2014. - 27 с.
5. Ланских, Юрий Владимирович Предметно-ориентированные информационные системы [Электронный ресурс] : учеб. пособие для студентов направления 09.03.02, 10.03.01,
6. Корячко, В. П. Процессы и задачи управления проектами информационных систем [Электронный ресурс] / В.П. Корячко. - Москва: Горячая линия - Телеком, 2014. - 376 с.
7. Коноплева, И. А. Информационные технологии [Электронный ресурс] / И.А. Коноплева. - 2-е изд., перераб. и доп. - Москва: Проспект, 2014. - 328 с.
8. 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.
9. 3.Золотов, С. Ю. Проектирование информационных систем [Электронный ресурс] / С.Ю. Золотов. - Томск: Эль Контент, 2013. - 88 с.
10. (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.