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

Е.В. Кружик

\_\_\_\_\_\_\_\_\_\_\_дата

2020-2021 уч.г.

План прохождения учебной практики

по ПМ.03 «Ревьюирирование программных продуктов»

Студент Кружик Егор Владимировиич

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.\_\_\_\_\_\_\_

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

**Отзыв руководителя**

Студент Кружик Егор Владимировиич

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 | | |  |

Критерии оценивания:

35-42 б. – «отлично»;

30-35 б. – «хорошо»;

26-28 б. – «удовлетворительно»;

Меньше 26 б. работа не оценивается.

Подпись руководителя:

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

\_\_\_\_\_\_\_\_\_\_\_\_\_дата

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc89724607)

[**1. Список терминов и определений** 7](#_Toc89724608)

[**2. Функциональные требования к Iaito.** 8](#_Toc89724609)

[**3. Измерение производительности приложения посредством анализа использования ЦП** 9](#_Toc89724610)

[**4. Установка ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования** 12](#_Toc89724611)

[**5. Дизассемблирование** 18](#_Toc89724612)

[**6. Обратное проектирование** 22](#_Toc89724613)

[**ЗАКЛЮЧЕНИЕ** 23](#_Toc89724614)

[**ЛИТЕРАТУРА** 24](#_Toc89724615)

**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.

И т.д найти все определения, которые используете при описании своей системы

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

**Дизассемблер** —это компьютерная программа, которая переводит машинный язык на язык ассемблера-операция, обратная операции ассемблера.

**Radare2** (также известный как r2)-это полный фреймворк для обратного проектирования и анализа двоичных файлов, состоящий из набора небольших утилит, которые могут использоваться вместе или независимо от командной строки.

**Точка останова** (англ. breakpoint) — это преднамеренное прерывание выполнения программы, при котором выполняется вызов отладчика (одновременно с этим программа сама может использовать точки останова для своих нужд).

**Компилятор** - это программа, которая выполняет преобразование текста программы в другое представление, обычно машинный код, без его запуска, статически.

**GCC** - это свободно доступный оптимизирующий компилятор для языков C, C++, Ada 95, а также Objective C. Его версии применяются для различных реализаций Unix (а также VMS, OS/2 и других систем PC), и позволяют генерировать код для множества процессоров.

**Программное обеспечение** (ПО) — программа или множество программ, используемых для управления компьютером (ISO/IEC)

**Linux** — семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты.

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

**Обратная разработка** — исследование некоторого готового устройства или программы, а также документации на него с целью понять принцип его работы; например, чтобы обнаружить недокументированные возможности (в том числе программные закладки), сделать изменение или воспроизвести устройство, программу или иной объект с аналогичными функциями, но без прямого копирования.

**Iaito** – графический интерфейс для radare2, написанного на С++ с использованием графического тулкита Qt 5.3

# **2. Функциональные требования к Iaito.**

**Iaito** – графический интерфейс для radare2, написанного на С++ с использованием графического тулкита Qt 5.3

**2. Функциональные требования к Iaito.**

1. Установка ПО (Linux, компилятор GCC, radare2, iaito)

2. Системные требования:

64-разрядная операционная система, процессор x64

2048 Мб оперативной памяти

200Мб места на жестком диске для инсталляции

Процессор с частотой, согласно требования ОС, 2 ядра

# **3. Измерение производительности приложения посредством анализа использования ЦП**

Запустили проект в программе Visual studio и разместили две точки остановы где хотим проверить нагрузку на ЦП и начали отладку программы.

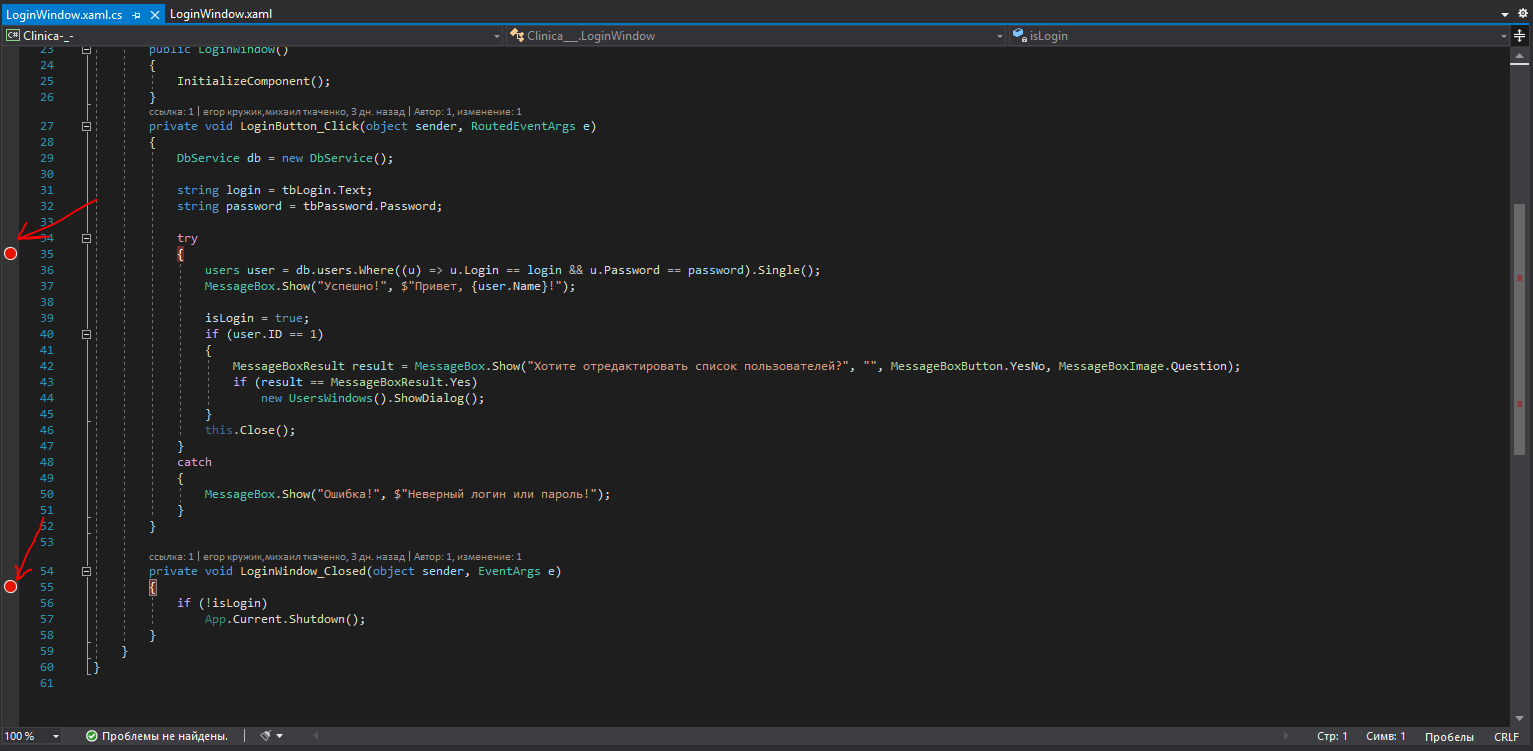


Рис.1 Установка точек остановы

Ставим галочку на кнопке «Запись профиля ЦП». После этого, программа дойдёт до второй точки остановы, ждём пока закончится обработка данных.

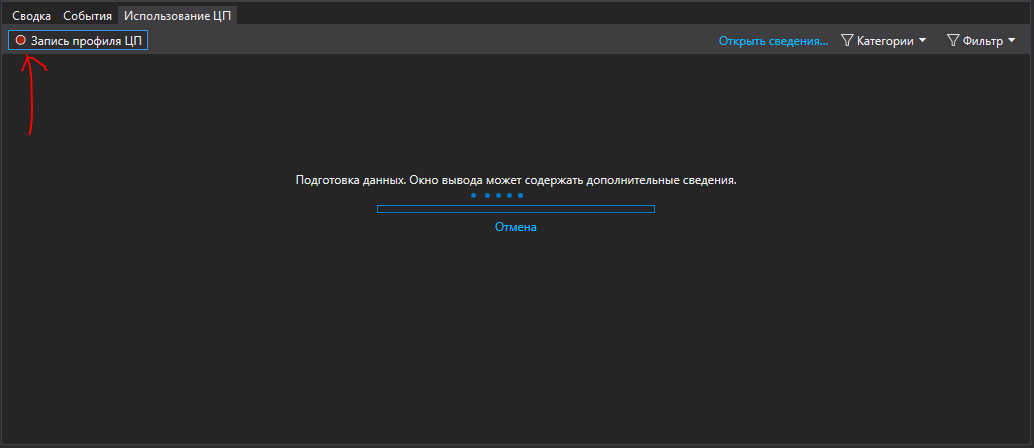


Рис.2 Использование ЦП

После того как обработка данных завершена, мы видим, все данные по загрузке ЦП

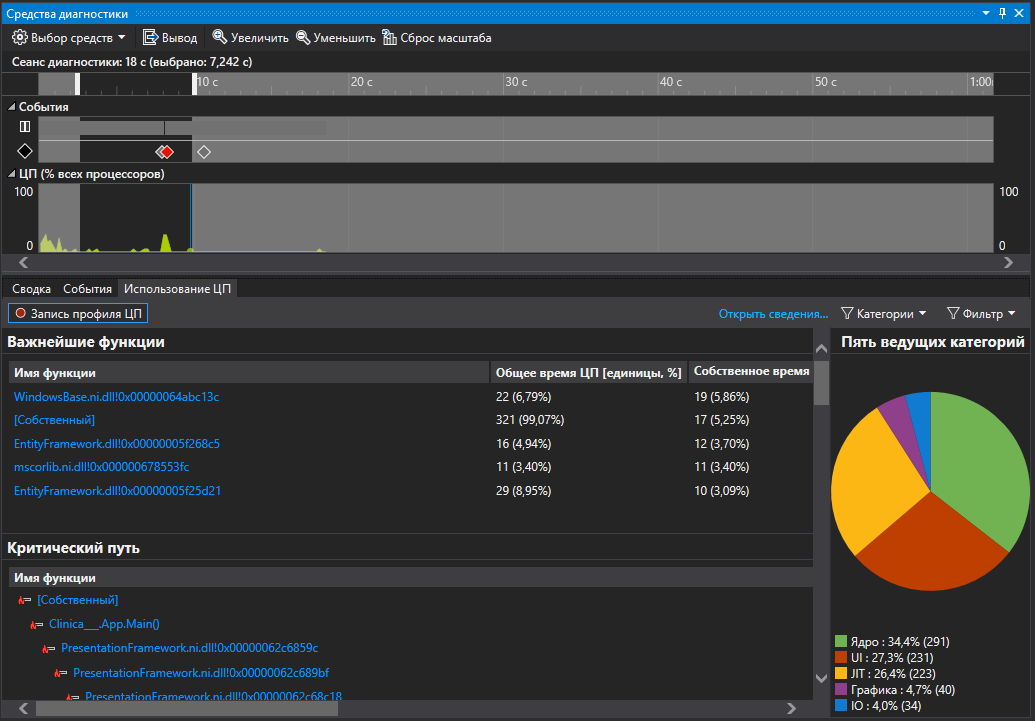


Рис.3 Результаты нагрузки ЦП

На данном скрине мы видим список функций, которые выполняют в большую часть работы.

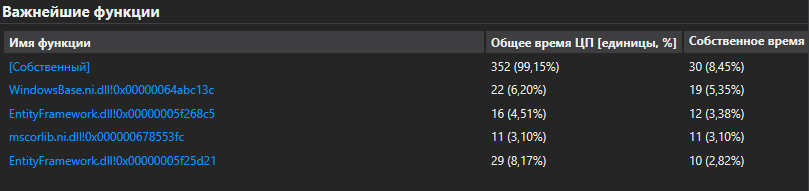


Рис.4 Список функций

После нажатия на одну из функций в левой панели откроется представление «Вызывающий/вызываемый». В списке функций изучите функции, которые выполняют большую часть работы.

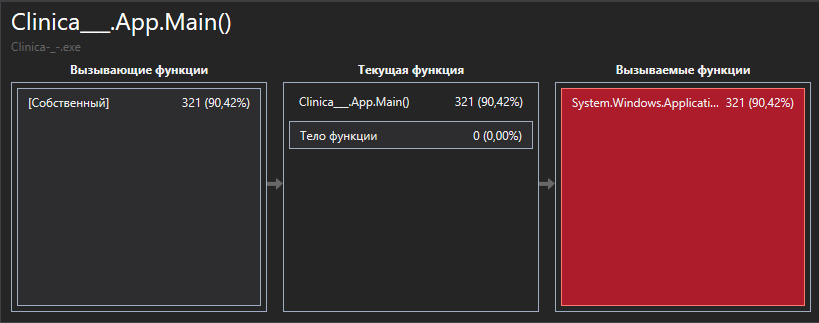


Рис.5 Представление вызывающий/вызываемый

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

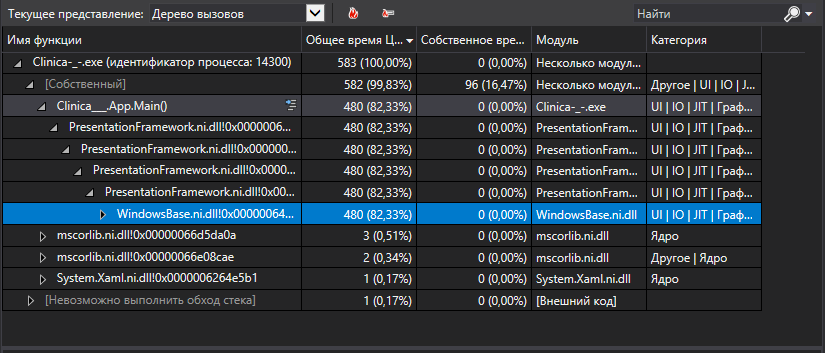


Рис.6 Дерево вызовов

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

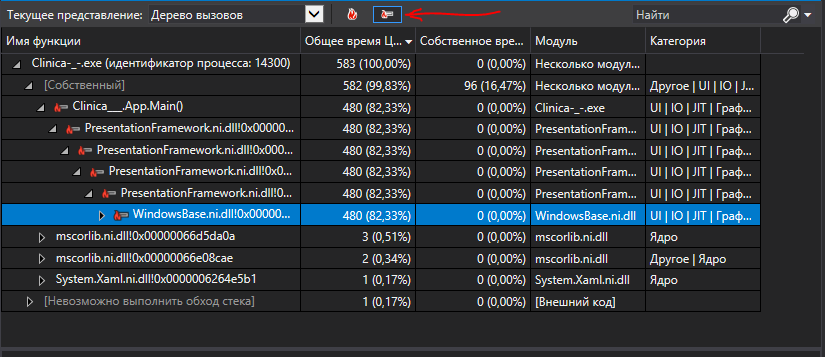


Рис.7 Критический путь

Поставили галочку «Показать внешний код» (**Средство диагностики> Фильтр**)

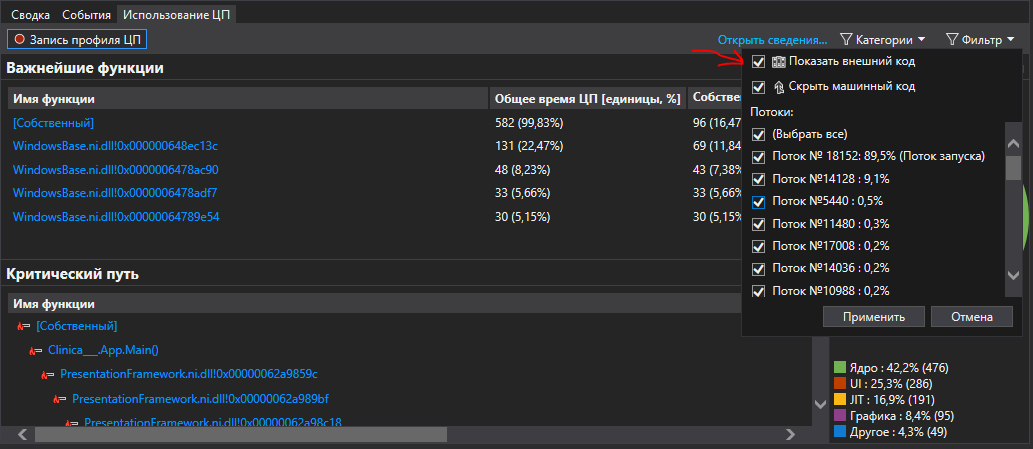


Рис.8 Установка галочки

Вывод: Данная программа не сильно нагружает ЦП и, следовательно, не нуждается в оптимизации.

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

Запускаем Virtual box

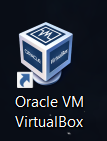


Рис.9 (VirtualBox)

Нажимаем создать.

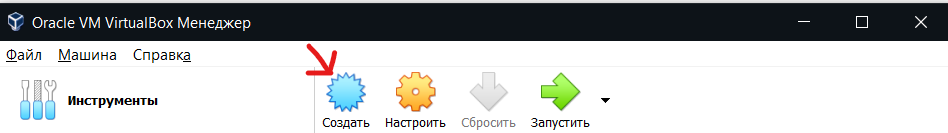


Рис.10 (кнопка создать)

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

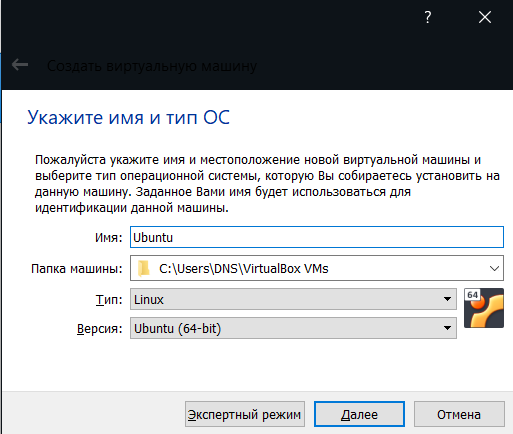


Рис.11 (называем и выбираем тип)

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

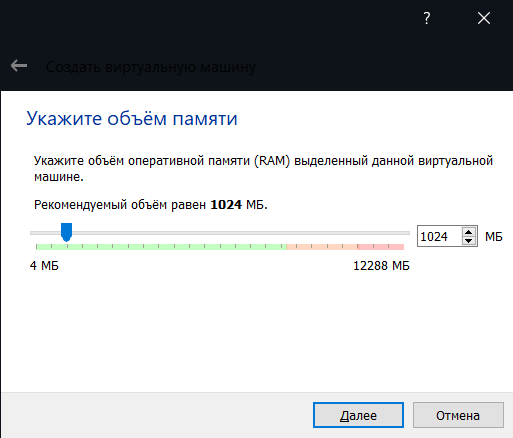


Рис.12 (выбор количества оперативной памяти)

Нажимаем создать

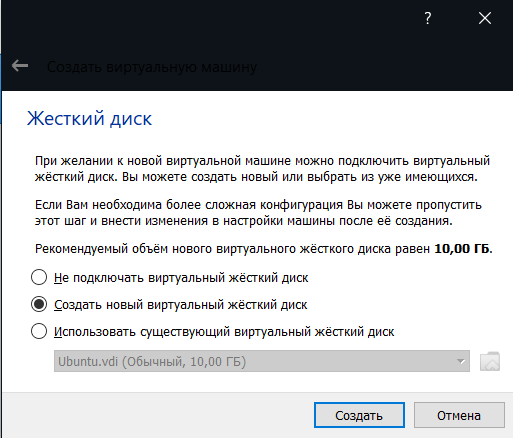


Рис.13 (выбираем жесткий диск)

Указываем тип и нажимаем далее

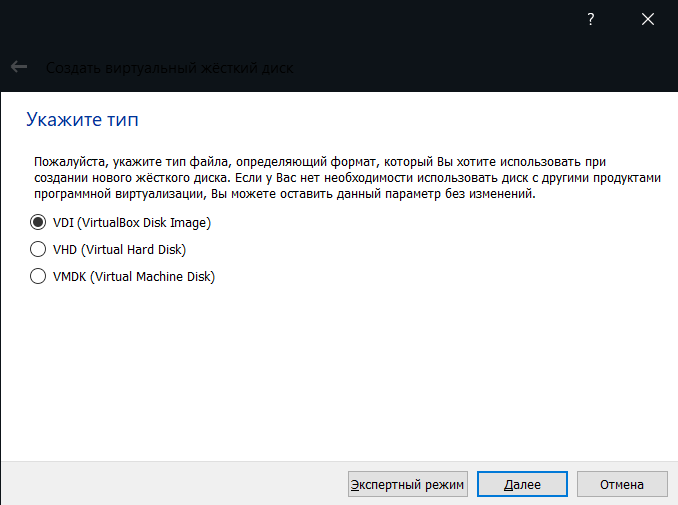


Рис.14 (тип файла)

Указываем формат хранения, выбираем «динамический виртуальный жесткий диск»

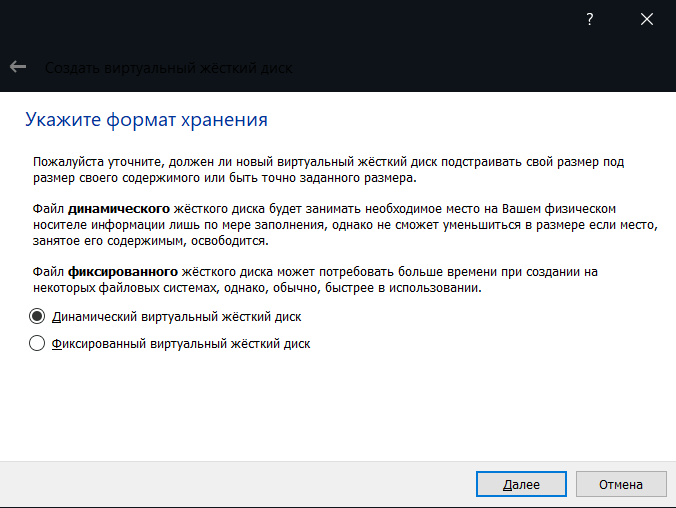


Рис.15 (формат хранения)

Указываем размер файла (оставляем так как стоит).

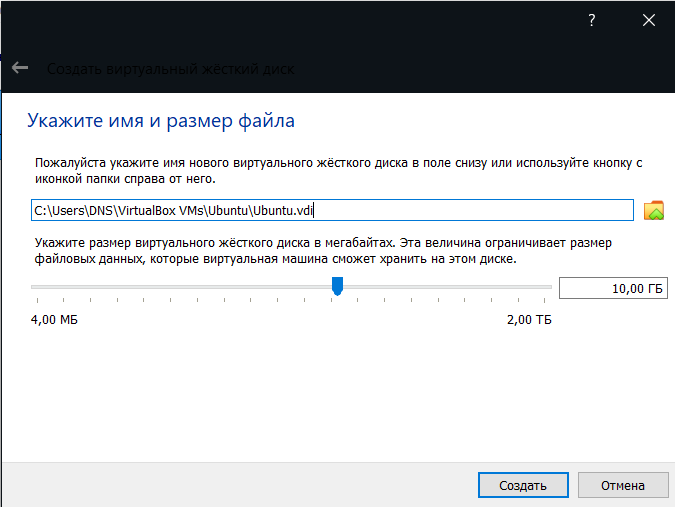


Рис.16 (размер файла)

Заходим в настройки.

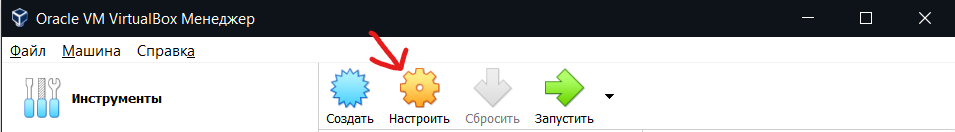


Рис.17 (кнопка настройка)

Заходим в «носители» и удаляем контроллер, где написано пусто

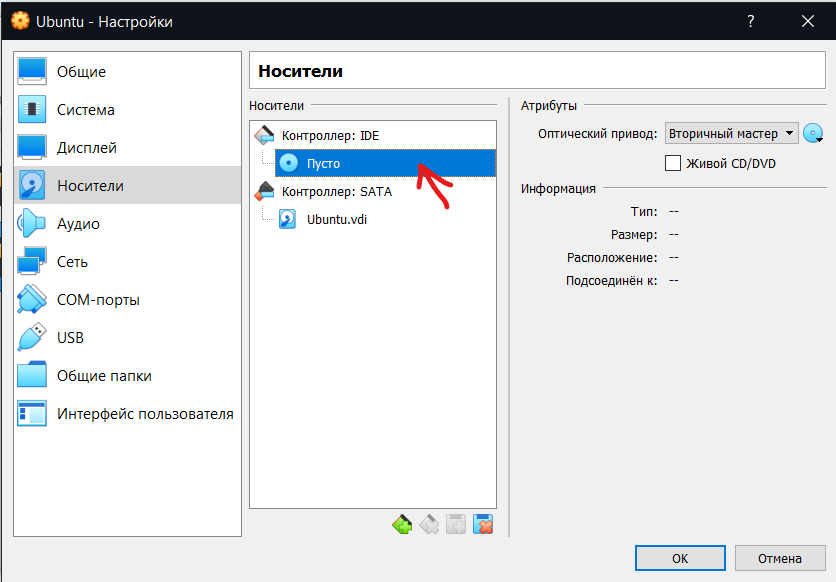


Рис.18 (удаления контроллера)

Нажимаем на диск и выбираем образ устанавливаем системы.

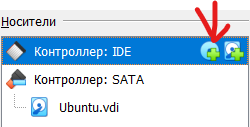


Рис.19 (выбор образа)



Рис.20 (кнопка добавления образа)

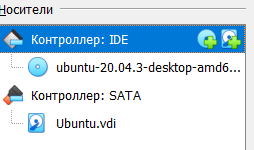


Рис.21 (выбранный образ)

Запускаем (пойдет установка Linux)

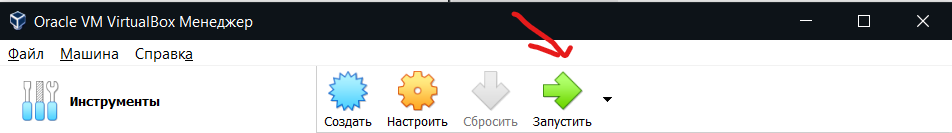


Рис.22 (кнопка запустить)

Установка Linux

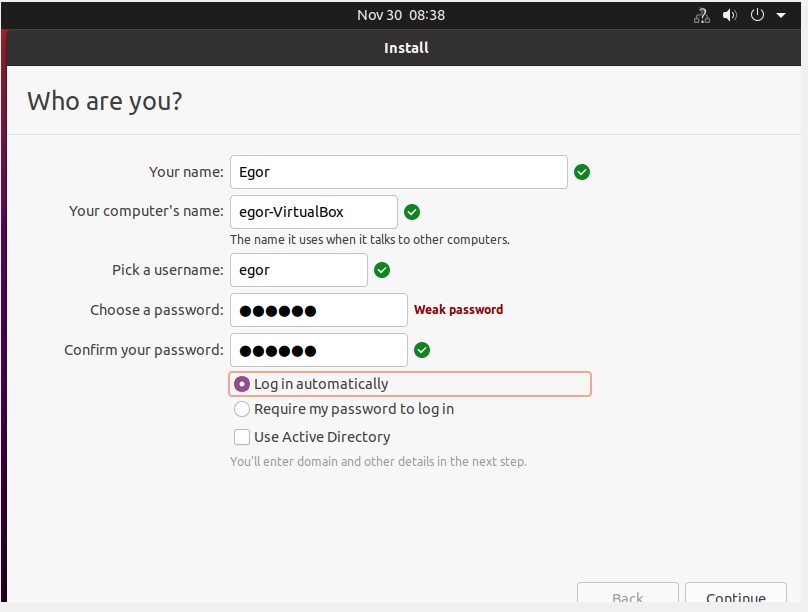


Рис.23 Создание профиля Linux

Процесс установки Linux

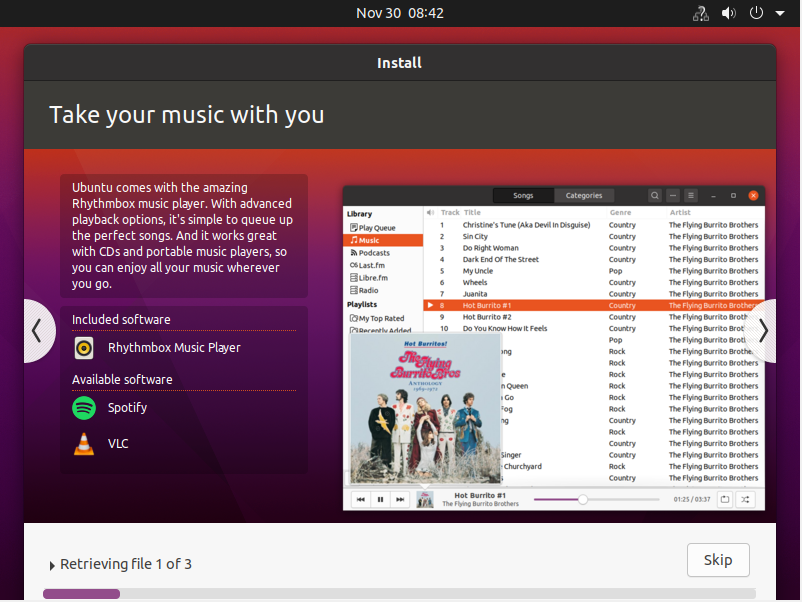


Рис. 24 Ожидание установки Linux

Настроили прокси на компьютере

C:\Users\Okkuro\Documents\ShareX\Screenshots\2021-12\ApplicationFrameHost_4F0HO9P9Ce.png

Рис. 25 Код для настройки прокси

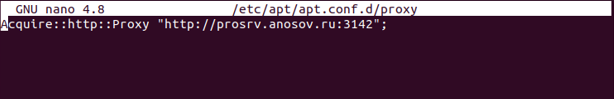


Рис.26 Код для прокси

Обновление убунту



Рис. 27 Код для обновления убунту

Устанавливаем компилятор GCC все необходимые для него компоненты



Рис.28 Код для установки

Установили пакеты для работы Radare2



Рис. 29 Код для установки пакетов

Установка Radare2



Рис.30 Код

Настройка прокси для колледжа



Рис.31 Код

Установили компоненты для граф. Оболочки iaito



Рис.32 Код

Установили граф. Оболочку



Рис.33 Код

Установили Декомпилятор r2ghidra



Рис.34 Код

# **5. Дизассемблирование**

Создали программу, которую будем дизассемблировать



Рис.35 Код



Рис.36 Код

Компиляция кода



Рис.37 Код

После установки появилась иконка iaito

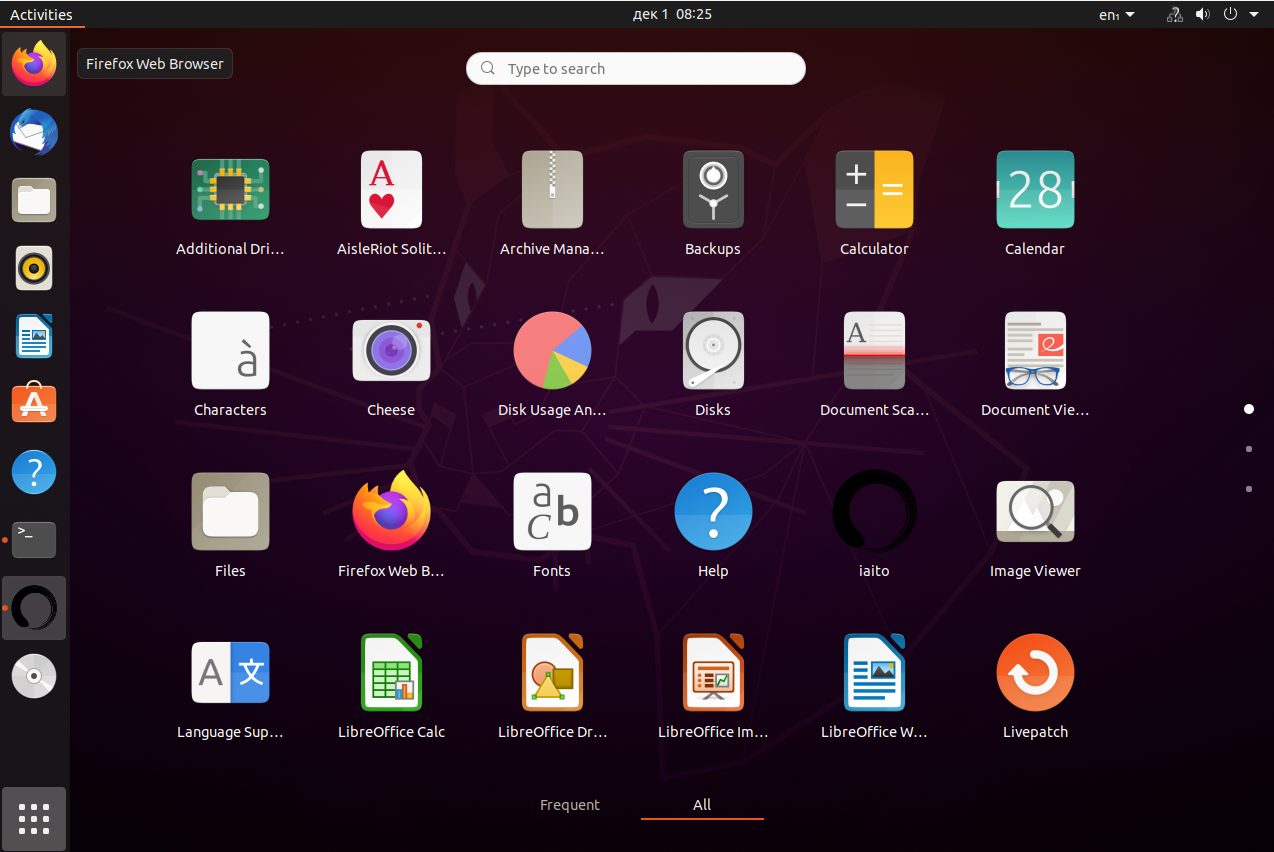


Рис. 38 Иконка Iaito

Запустили Iaito и ввели путь к программе

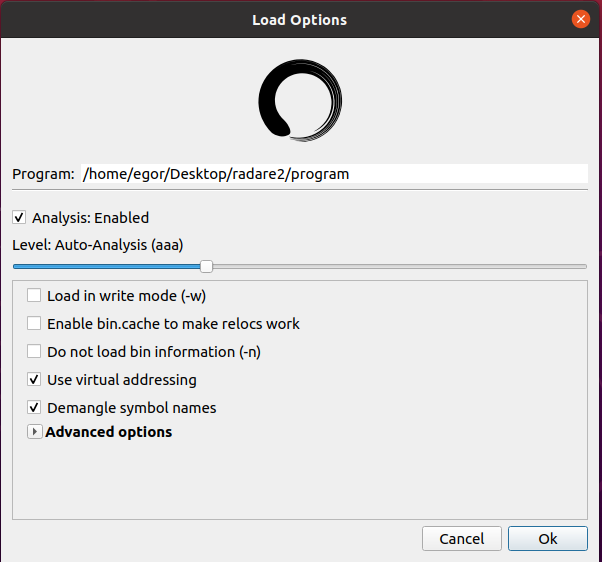


Рис.39 Интерфейс программы

Открыли слева в меню “main”

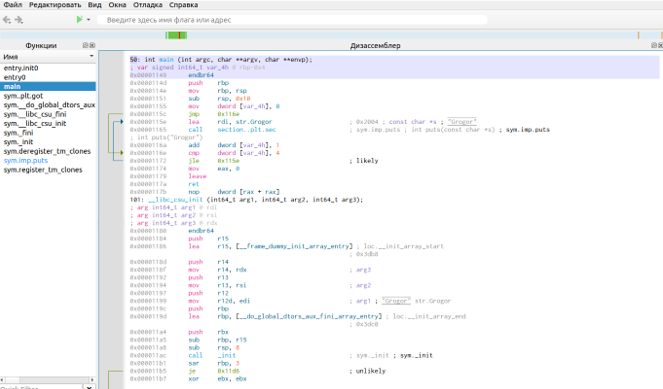


Рис.40 Интерфейс программы

Выбрали другой режим работы (Файл> Set mode> Cache mode)

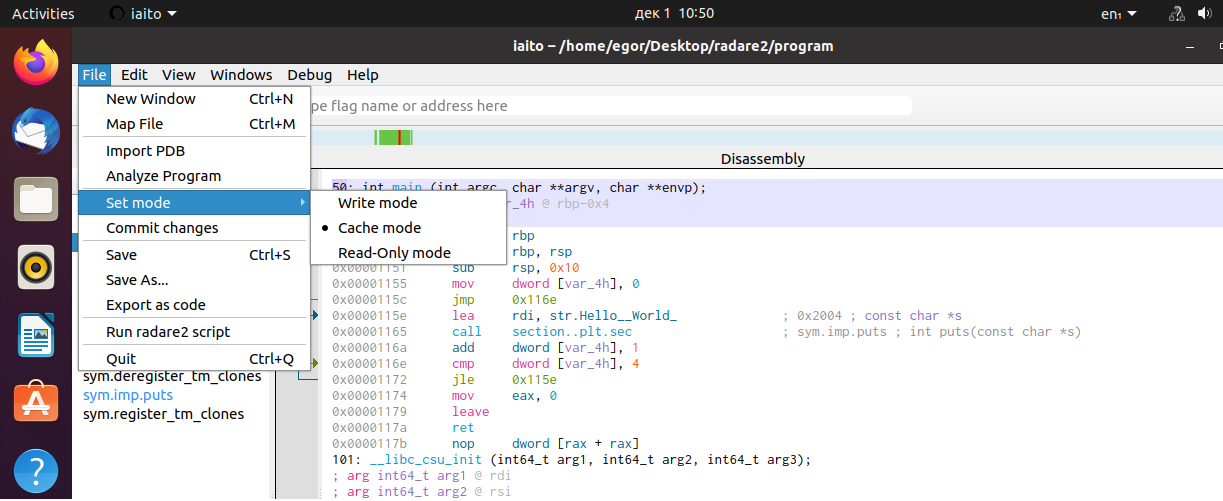


Рис.41 Интерфейс программы

# **Обратное проектирование**

Так как фраза «DAROVA!» выводится 4 раза, а нам необходимо увеличить вывод до 10 раз, то смотря на код, можно увидеть, что используется цикл «while» по значение переменной [var4\_h] равной 4. Выделяем цифру 4 и нажимаем (Редактировать> Инструкция) меняя значение переменное с 4 до 10.

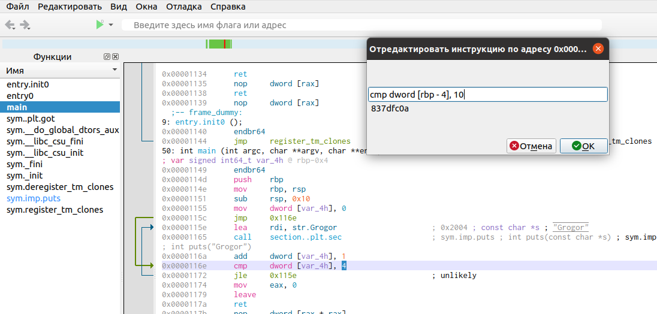


Рис.42 Интерфейс программы

Сохранили изменения (Файл> Commit changes)

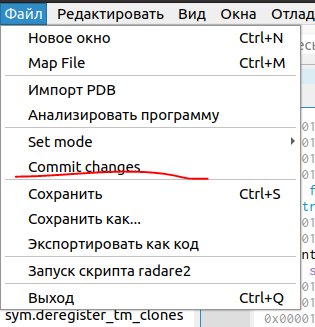


Рис. 43 Интерфейс программы

Проверили нашу программу

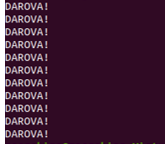


Рис.44 Код

Проверка кода на использование ЦП

Перевели код на язык C# и установили точки остановы

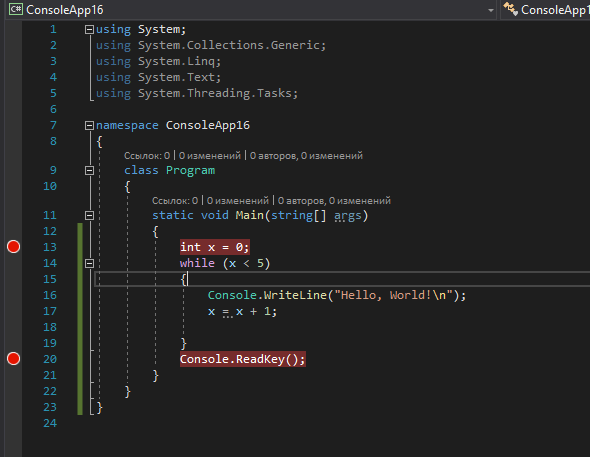


Рис.45 Код

Провели анализ окна средства диагностики

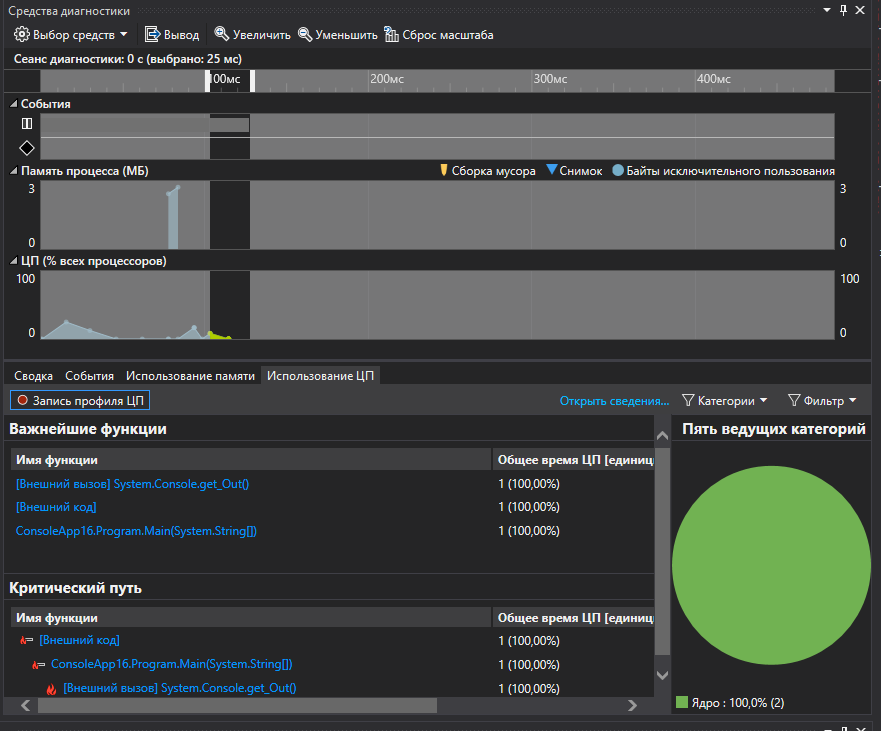


Рис.46 окно «средства диагностики»

# **ЗАКЛЮЧЕНИЕ**

Целью учебной практики по ПМ.03 «Ревьюирирование программных продуктов» являлось, научиться разрабатывать функциональные требования к информационной системе автоматизации бизнес-процессов.

В ходе выполнения заданий на учебную практику было выполнено измерение производительности приложения посредством анализа использования ЦП в программе Visual studio на основе своей собственной программы, установили ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования, обратное проектирование, дизассемблирование на системе Linux Ubuntu x64 программой Iaito и дизассемблером r2ghidro.

# **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.