Титульный лист

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\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_Е.М.Пипоть

\_\_\_\_\_\_\_\_\_\_\_дата

2021-2022 уч.г.

**Содержание**

[**Введение** 3](#_Toc88825072)

[**1. Список терминов и определений** 5](#_Toc88825073)

[**2. Функциональные требования** 6](#_Toc88825074)

[**3. Измерение производительности приложения посредством анализа ЦП** 7](#_Toc88825075)

[**4. Установка ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования** 12](#_Toc88825076)

[Установка Ubuntu 12](#_Toc88825077)

[Настройка прокси на компьютерах колледжа 13](#_Toc88825078)

[Дизассемблирование 15](#_Toc88825079)

[**5. Дизассемблирование** 17](#_Toc88825080)

[**6. Обратное проектирование** 20](#_Toc88825081)

[**ЗАКЛЮЧЕНИЕ** 21](#_Toc88825082)

[**ЛИТЕРАТУРА** 22](#_Toc88825083)

# **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.

## **1. Список терминов и определений**

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

*Дизассемблер* – это программа, тесно взаимодействующая с пользователем в процессе дизассемблирования.

*Обратное проектирование* – это исследование некоторого готового устройства или программы, а также документации на него с целью понять принцип его работы.

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

*Виртуальная машина* – это эмулятор компьютера.

*Терминал* – это графическая программа эмулирующая консоль.

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

## **2. Функциональные требования**

Рекомендуемое ПО для корректного выполнения работы:

- 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 ЦП**

Для хранения информации в репозитории нам будут нужен GitHub

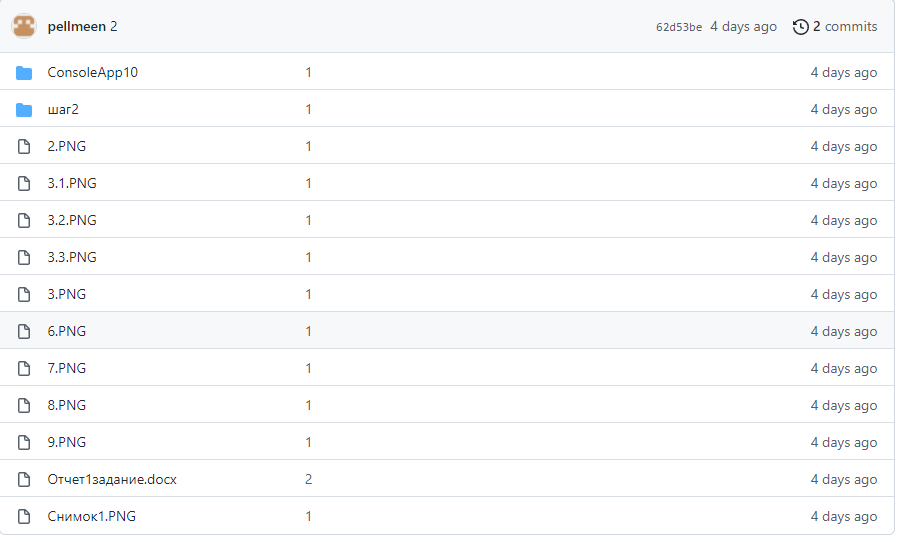


Рис. 1 GitHub

Для измерения производительности нам будет нужно Visual Studio 2019



Рис. 2 VS2019

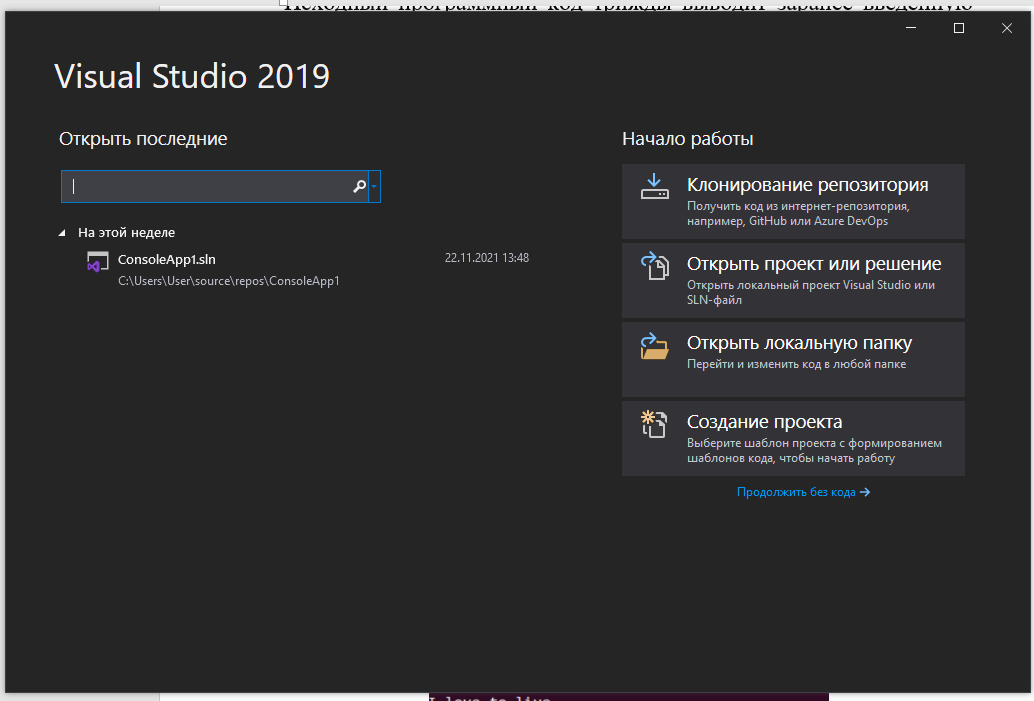
Запускаем полнофункциональную среду разработки для написания, отладки, тестирования и развёртывания кода Visual Studio 2019

Рис. 3 Запуск VS

Открываем проект, производительность которого нам необходимо измерить и ставим первую точку остановы на участке исследования.

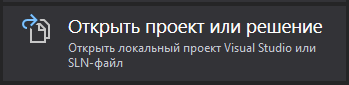


Рис. 4 Открытие проекта

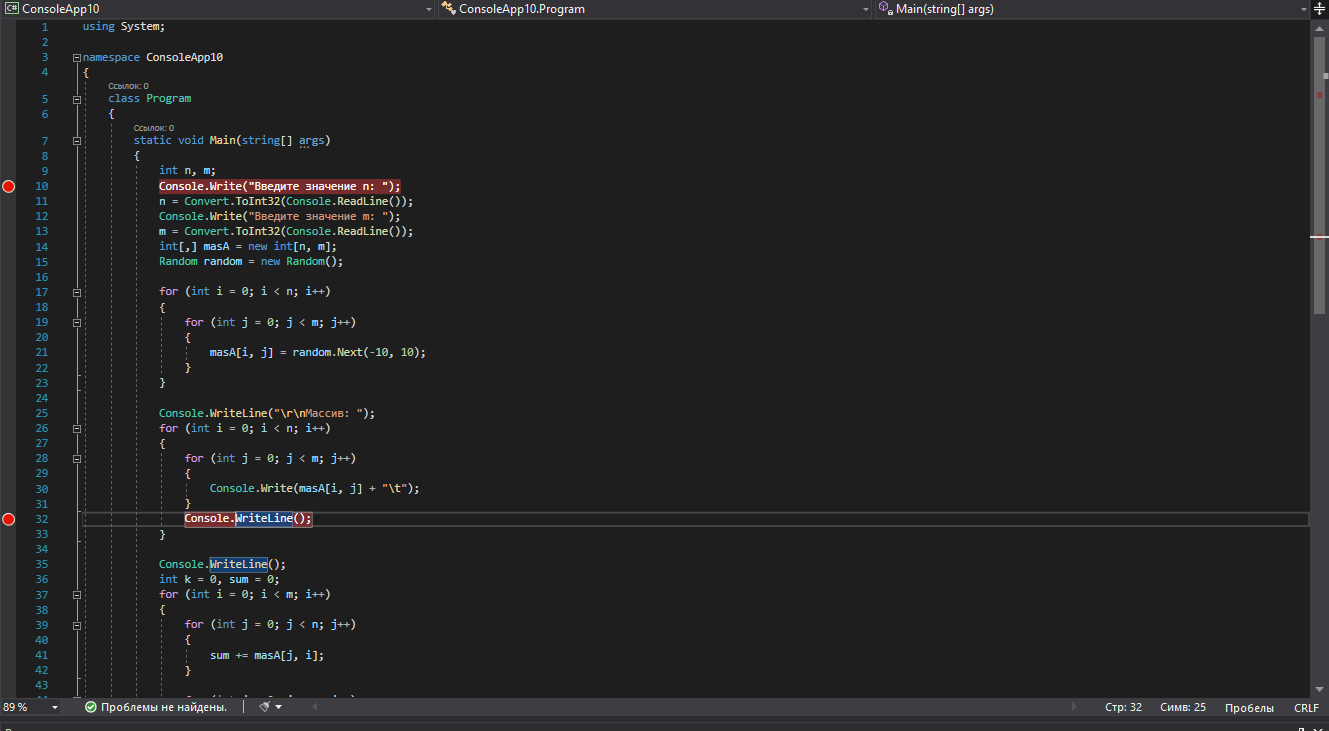


Рис. 5 Открытие проекта

Таким же образом располагаем вторую точку (она нужна для преднамеренного прерывания выполнения программы, при котором выполняется вызов отладчика), так мы отделяем нужный участок кода, чтобы более точечно провести измерения.

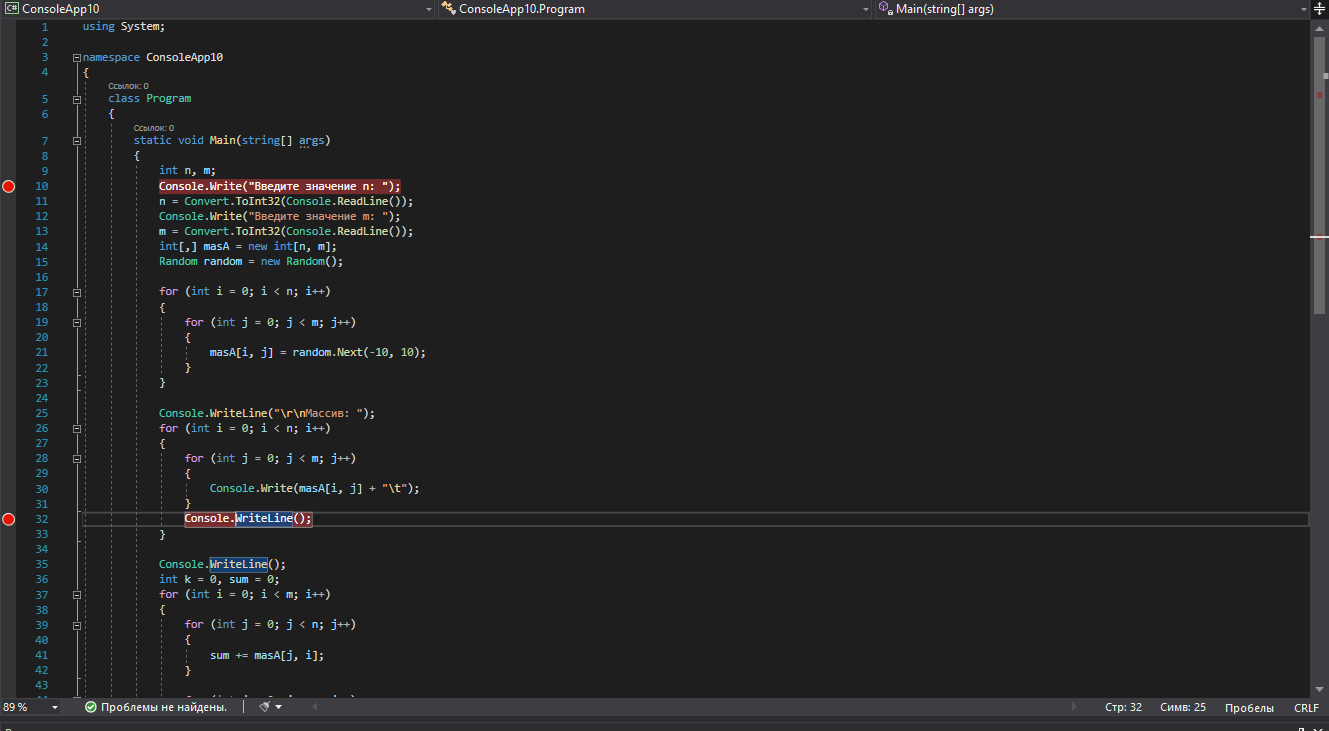


Рис. 6 Отделение участка

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

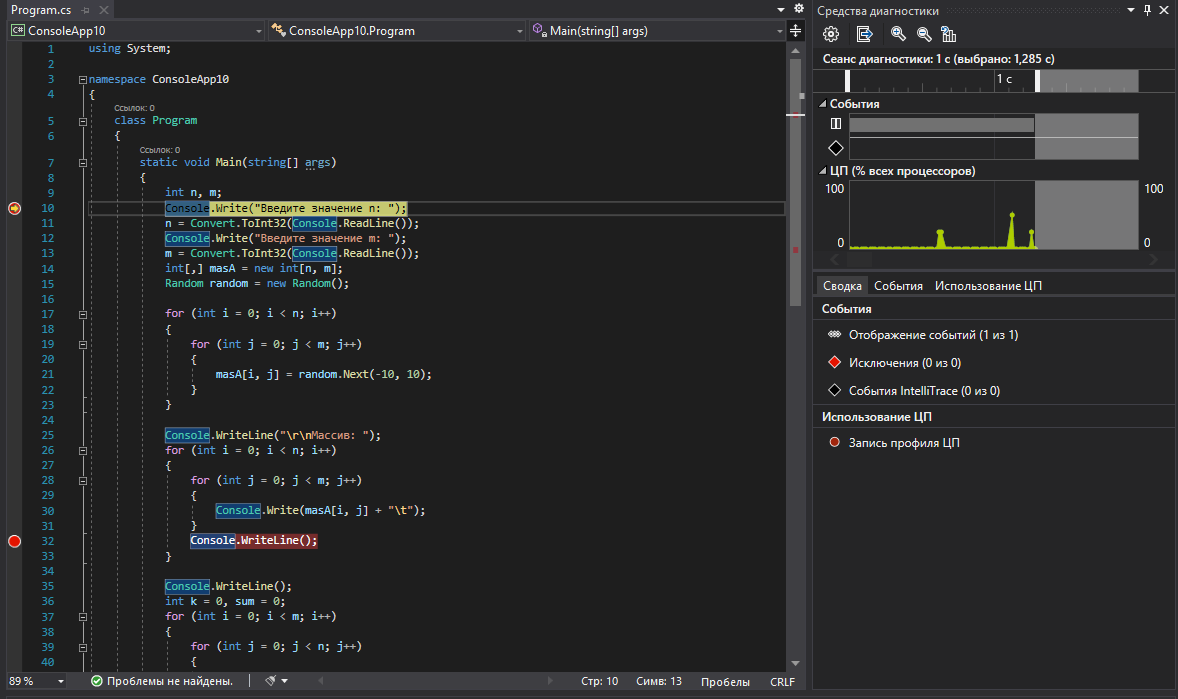


Рис. 7 Запуск программы

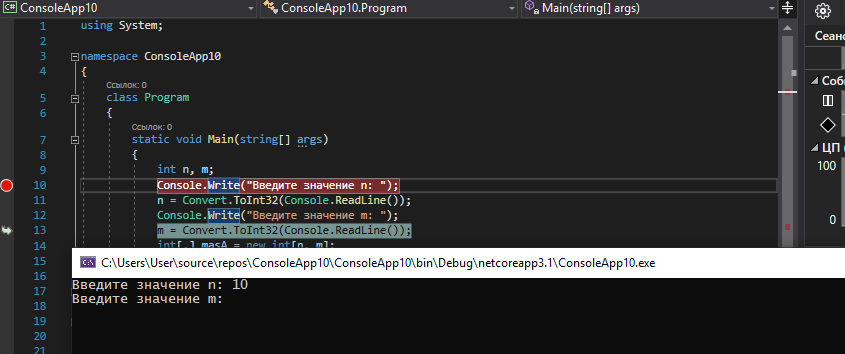


Рис. 8 Запуск программы

После запуска программы, в правой части отладчика у нас появляется окно средств диагностики (включено по умолчанию, в случае отсутствия вызывается комбинацией клавиш ctrl+alt+F2 или открывается во вкладке Окна), в котором мы можем увидеть данные о событиях. Начинается потребление мощности ЦП.

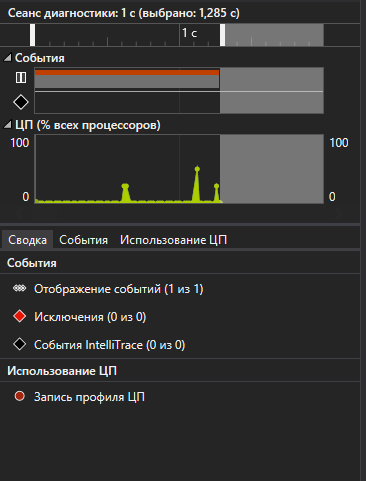


Рис. 9 Средства диагностики

Для получения информации о загрузке ЦП перейдём во вкладку Использование ЦП.

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

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

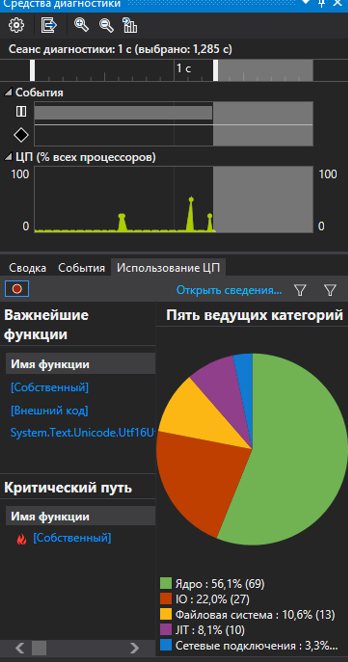


Рис. 10 Загрузка ЦП

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.

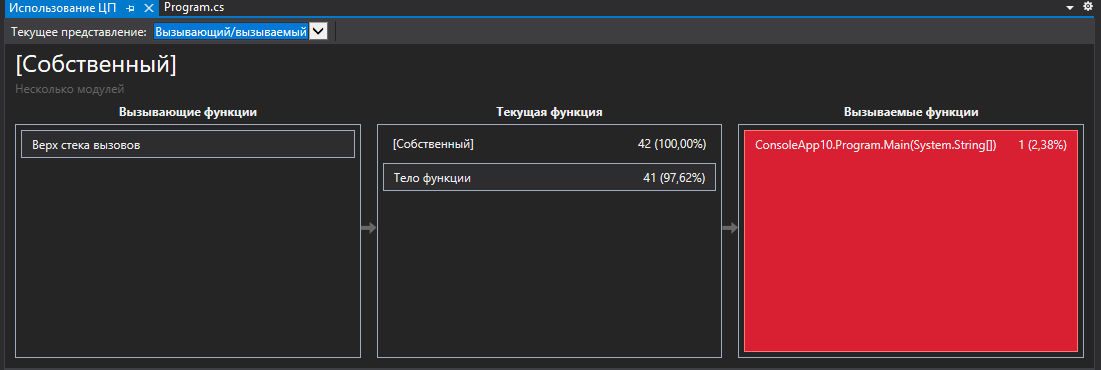
****

Рис. 11 Вызывающий\вызываемый

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

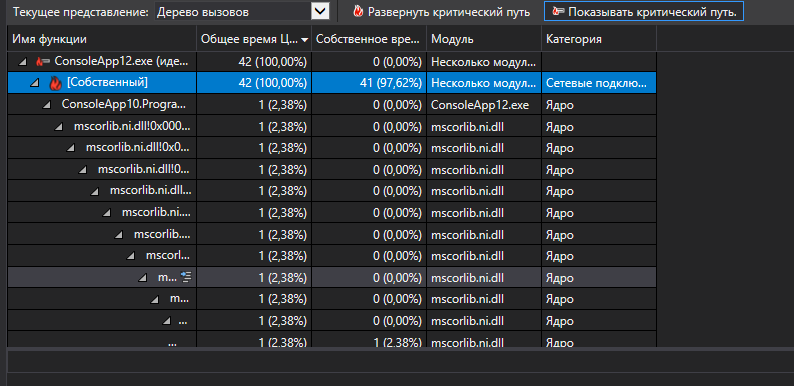
****

Рис. 12 Дерево вызовов

**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 VMware на оболочке Ubuntu.

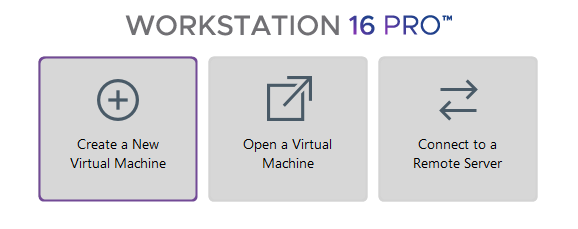


Рис. 13 Создание образа

После чего выбираем образ формата iso и подключаем образ Ubuntu

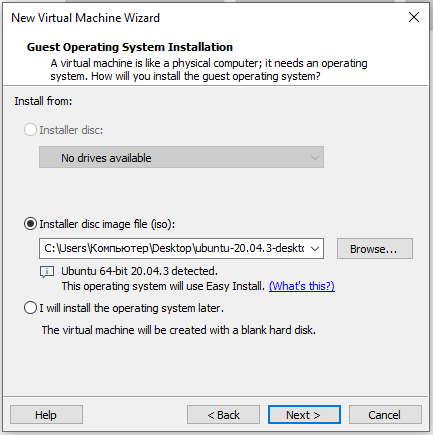


Рис.14 Подключение образа

Затем создаем профиль Ubuntu

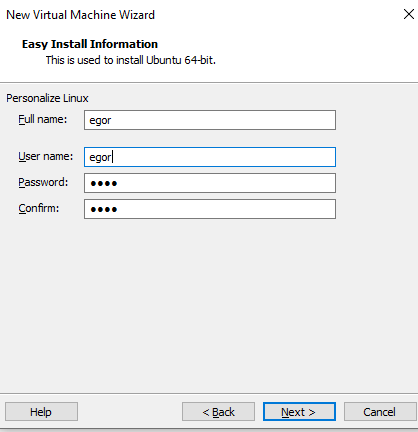


Рис. 15 Создание профиля

Указываем размер диска (20Гб)

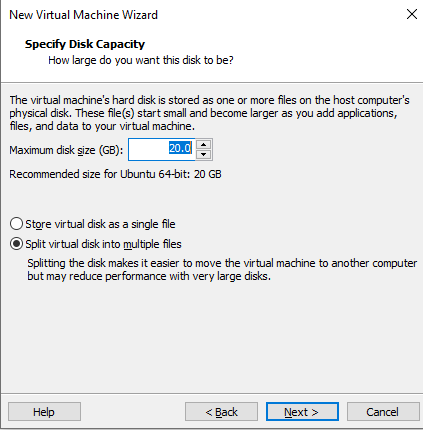


Рис. 16 Выбираем размер

### Настройка прокси на компьютерах колледжа

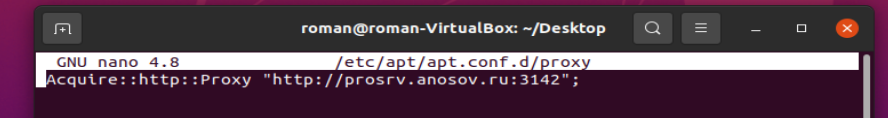
В учебном заведении ведётся политики цифровой безопасности. Не допускается возможность взаимодействия с ресурсами, не имеющими отношения к образовательному плану. Поэтому необходимо ввести имя прокси сервера, введённое информационным центром колледжа.

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



Рис. 17 Nano

Вписываем туда данные, необходимые для работы с прокси.

Рис. 18 Обход прокси

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. 19 Установка gcc

Скачиваем файл с программным кодом из репозитория.



Рис. 20 Создание файла

Открываем.

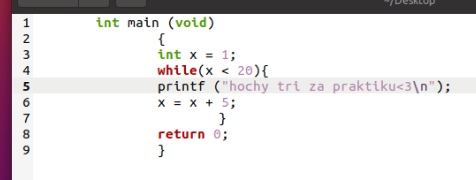


Рис. 21 Программный код

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

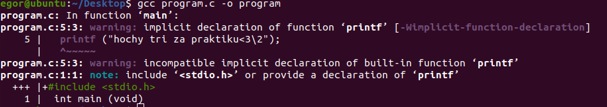


Рис. 22 Компиляция файла

В терминале запускаем скомпилированный файл и проверяем правильность выполнения.

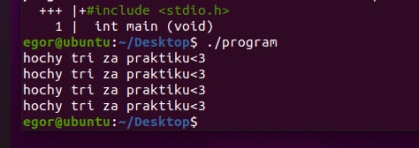


Рис. 23 Тест кода

### Дизассемблирование

Устанавливаем необходимые пакеты.



Рис. 24 Установка пакетов

Прописываем git config, чтобы настроить значения конфигурации Git на глобальном уровне.



Рис. 25 Настройка git

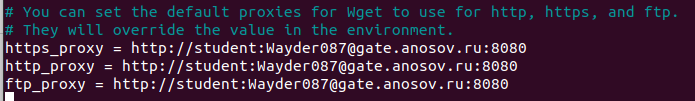


Рис. 26 Настройка git

Скачиваем radare2, копируя репозиторий с гитхаба на рабочий стол с помощью команды git clone.



Рис. 27 Копирование с git

Устанавливаем radare2.



Рис. 28 Установка

Устанавливаем необходимые компоненты для графической оболочки «Iaito».



Рис. 29 Установка пакетов для iaito

Открываем папку и запускаем файл.



Рис. 30 Запуск radare2

Запускаем утилиту, предназначенную для автоматизации преобразования файлов из одной формы в другую.



Рис. 31 Запуск утилиты

Устанавливаем графический интерфейс «iaito».



Рис. 32 Установка iaito

Устанавливаем Декомпилятор «r2ghidra».



Рис. 33 Установка r2ghidra

## **5. Дизассемблирование**

Запускаем Iaito (графический интерфейс radare2).

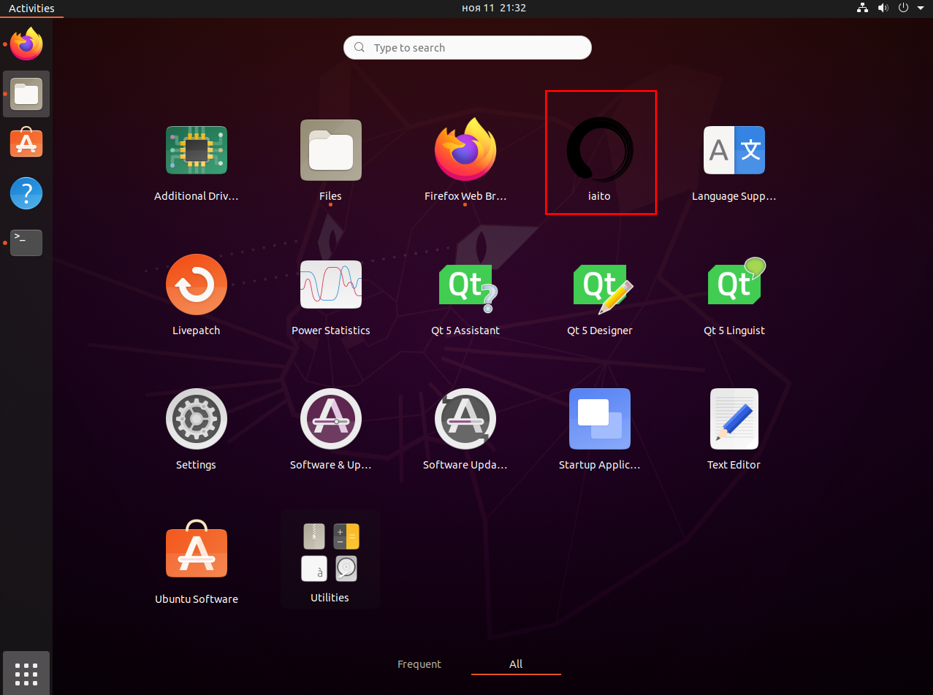


Рис. 34 Запуск iaito

Запускаем и проверяем тестовый файл. В указанном коде должно выполняется 4 итерации.

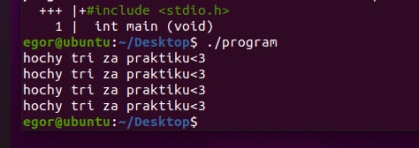


Рис. 35 Запуск тест файла

Запускаем графический интерфейс Iaito.

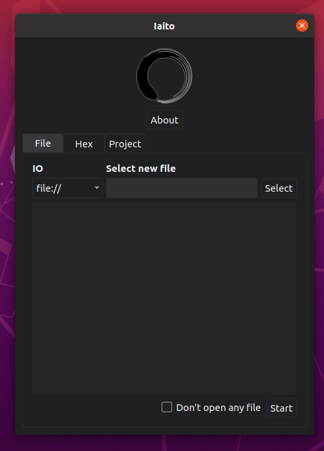


Рис. 36 Запуск iaito

Открываем раздел main в меню слева.

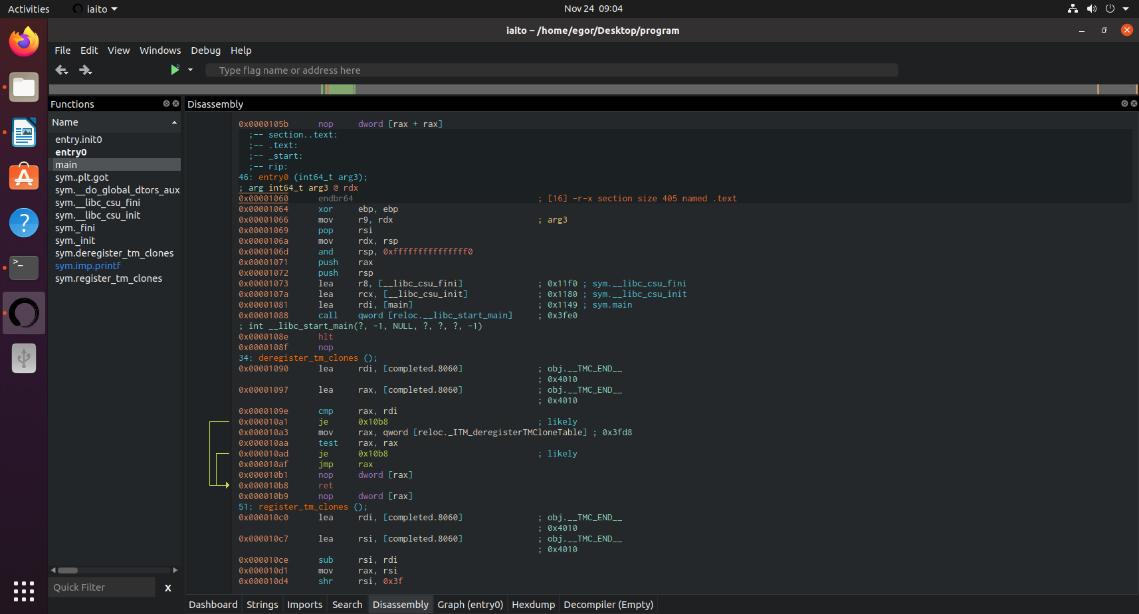


Рис. 37 меню main

Заходим в раздел Graph. Здесь в виде отдельных блоков описывается принцип работы кода.

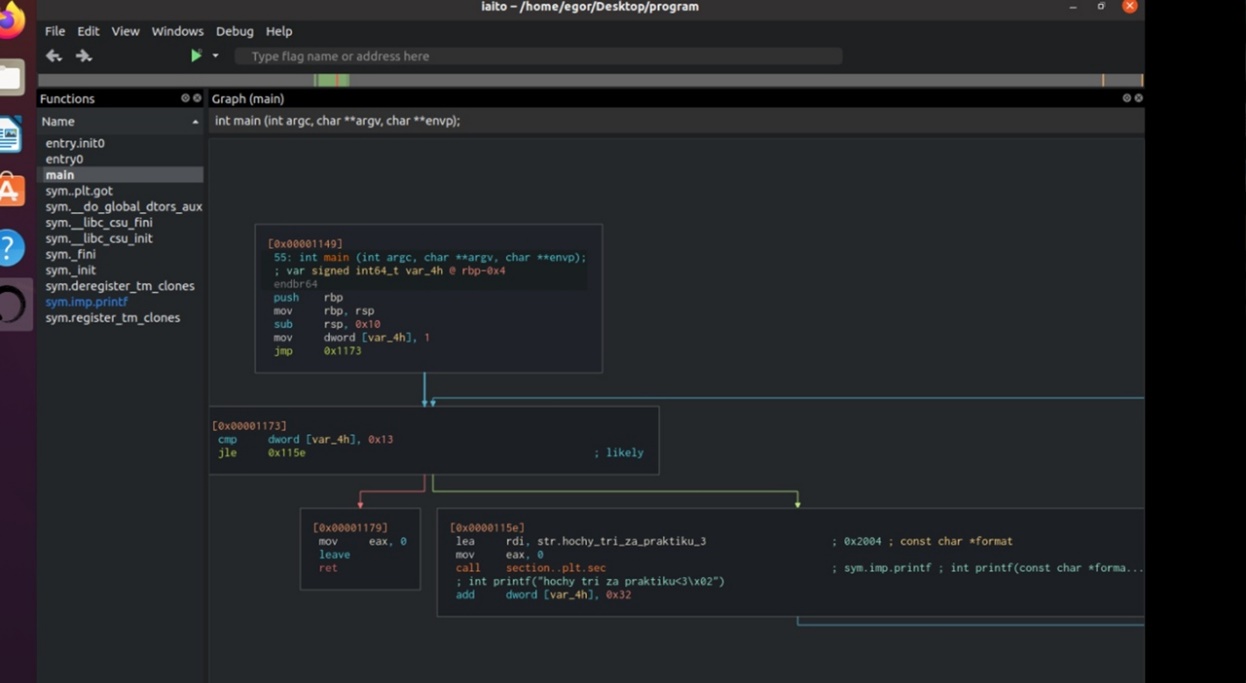


Рис. 38 раздел Graph

Выделяем переменную, которую хотим заменить и переходим в дизассемблер, заменяем её на новое значение.

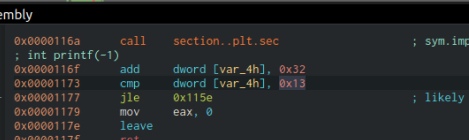


Рис. 39 Исходное значение

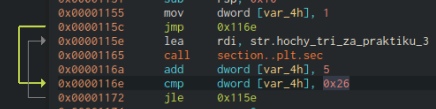


Рис. 40 Новое значение

## 

## **6. Обратное проектирование**

Исходный программный код четыре раза выводит заранее введённую фразу.

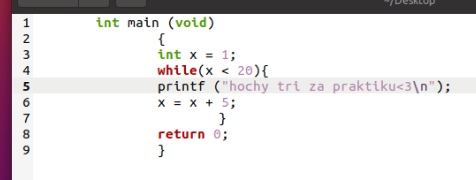


Рис. 41 Программный код

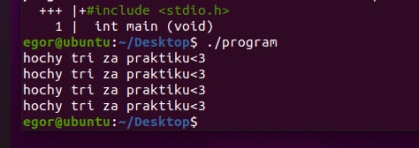


Рис. 42 Тест кода

После дизассемблирования в файл были внесены изменения. Число 4, участвующее в цикле while было заменено на 8, ввиду чего количество итераций увеличилось до 8.

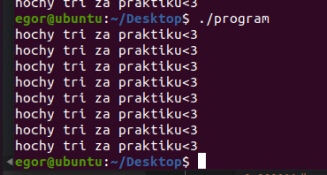


Рис. 43 Тест кода после изменения

**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 были выполнены следующие задачи:

* измерили производительность приложения посредством анализа использования ЦП,
* установили ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования,
* выполнили обратное проектирование,
* выполнили дизассемблирование.

# **ЛИТЕРАТУРА**

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

Приложение 1