**Отчёт обучающегося по практике**

Студент: Федотов Алексей Александрович Группа: ПИН-41

Направление: 09.03.04 Программная инженерия

Образовательная программа: Программные технологии распределенной обработки\_ информации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_

### Вид и тип практики: Производственная практика – преддипломная практика \_

Весенний семестр 2021 учебного года

Место прохождения практики: ООО «С-Терра СиЭсПи»

Практика проходила на предприятии ООО «С-Терра СиЭсПи» в отделе агента, мобильных решений и систем управления. Этот отдел занимается разработкой, тестированием и сопровождением программных продуктов кампании.

Продукция компании «С-Терра СиЭсПи» используется как в государственных учреждениях, так и в коммерческих организациях в соответствии с отраслевыми стандартами и требованиями по защите информации, в том числе:

защите конфиденциальной информации органов государственной власти в соответствии с СТР‑К, а также совместным приказом ФСБ и ФСТЭК от 31 августа 2010 года №416/489

защите подключения информационных систем государственных органов к Интернет в соответствии с Указом Президента РФ от 17 марта 2008 года №351, Постановлением Правительства РФ от 18 мая 2009 года №424, Приказом ФСО от 7 августа 2009 года №487

защите персональных данных в соответствии с методическими рекомендациями ФСБ по защите персональных данных, а также в соответствии с отраслевыми стандартами Банка России, НАУФОР, НАПФ, операторов связи и др.

защите объектов инфраструктуры в соответствии нормативными документами ФСТЭК России по защите ключевых систем информационной инфраструктуры

защите систем управления технологическими процессами (АСУ ТП)

защите крупных территориально-распределенных сетей и т.п.

В ООО «С-Терра СиЭсПи» была поставлена задача по разработке программного модуля, обеспечивающего упрощение конфигурирования сетевого сервиса, использующегося в компании.

# Отладка по точкам останова

Отладка – это процесс, с помощью которого можно найти и исправить ошибки в коде программы. Самым популярным способом отладки считается отладка с использованием точек останова.

Точка останова – это точка, которая обозначает место, в котором запланировано прерывание программы во время выполнения. Рассмотрим подробнее действия разработчика при таком способе отладки. Сначала устанавливаются точки в разных местах программы, затем программа запускается в режиме отладки. При достижении точки останова, программа прерывается и на экран выводятся все данные, необходимые для отладки. Благодаря этому, разработчик может понять, правильно ли ведет себя программа. В отличие от остановки программы, в данном случае выполнение можно продолжить с того же места, где была точка.

Отладчик – программа, которая позволяет в автоматическом режиме совершать поиск ошибок в программном коде, а также предоставлять дополнительные мощные инструменты контроля его выполнения, а именно:

пошаговое выполнение кода;

выполнение кода до определенных, поставленными на свое усмотрение программистом точек останова, называемых breakpoint-ами;

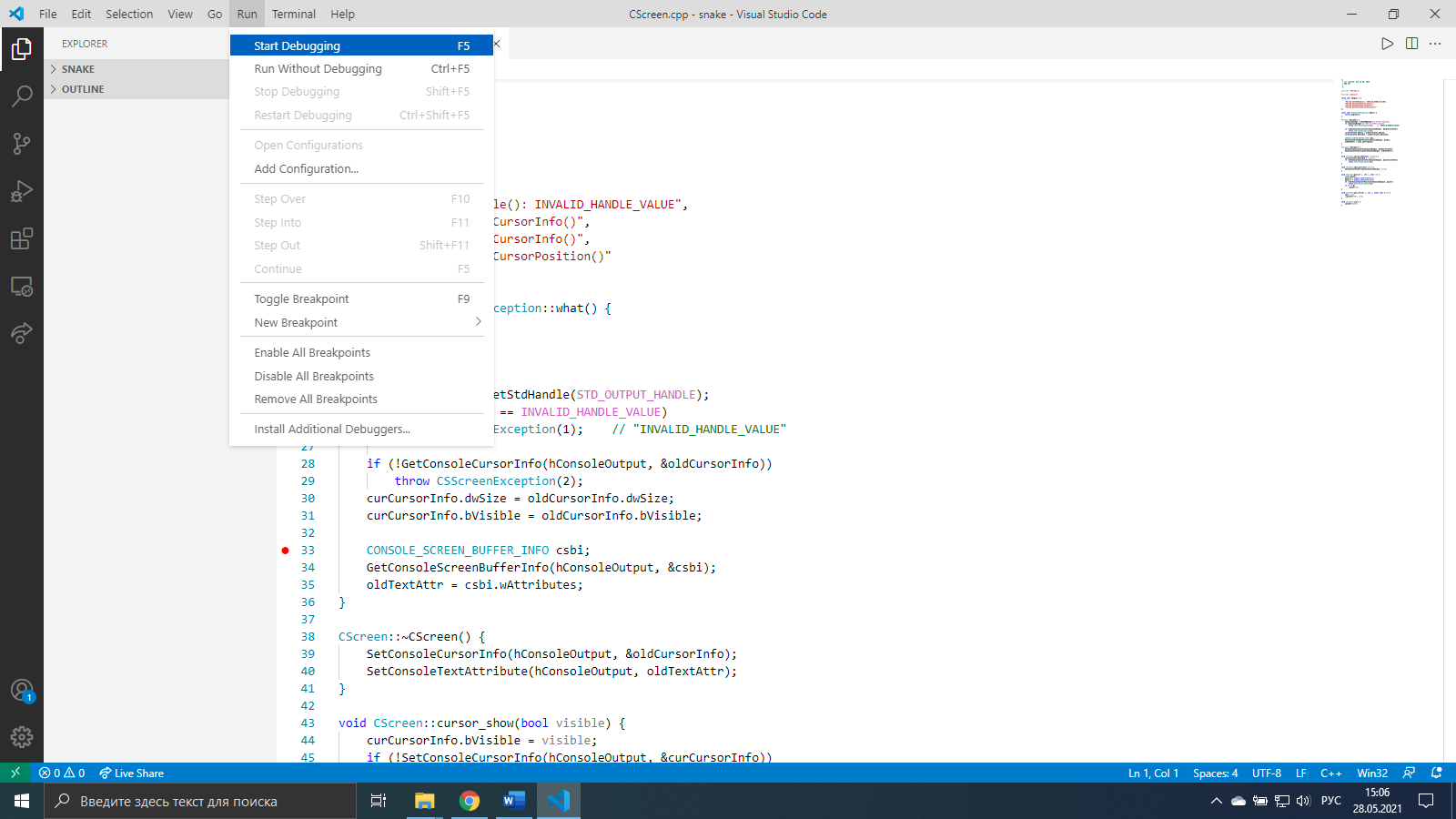
отслеживание значений любых инициализированных на данный момент переменных.

В процессе отладки ПМ НКСС активно использовался стандартный отладчик, встроенный в Visual Studio Code.

# Отладка ПМ НКСС

Отладка написанного программного кода производилась с помощью стандартного отладчика VS Code. Данный отладчик имеет графический интерфейс, что значительно упрощает его использование.

Для начала, чтобы начать его использовать, нужно подключить специальный плагин, предназначенный для компиляции кода на языке С++. Далее достаточно просто расставить точки останова и запустить код в режиме отладки.



*Рисунок 6 Запуск отладчика*

Данный отладчик обладает многими инструментами для поиска ошибок. Разработчик может всегда посмотреть, какие у него расставлены точки останова благодаря вкладке breakpoints.



Рисунок Вклдка breakpoints

Также есть возможность отслеживать значения любых переменных, даже тех, что находятся вне текущей функции. Их можно увидеть на вкладке watch

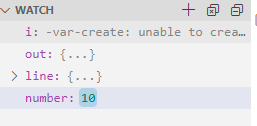


Рисунок Вкладка watch

Ошибки, возникающие при разработке программного обеспечения, можно разделить на синтаксические и логические.

Синтаксические – это такие ошибки, которые не позволяют транслятору или интерпретатору однозначно интерпретировать написанный исходный код программы. Иначе говоря, это ошибки в записи конструкций языка программирования. Зачастую такие ошибки вызваны обычными опечатками.

В случае с разработкой ПМ НКССС подобные проблемы выявляет встроенный анализатор среды разработки VS Code еще до трансляции кода и сборки проекта. Среда разработки указывает программисту файл, в котором обнаружена ошибка, ее расположение в коде этого файла, выделяет проблемное место и показывает подробную информацию во всплывающей подсказке при наведении курсора на это место. В качестве проблемных мест, говоря о средах разработки, могут быть следующие ошибки:

ошибки подключения библиотек;

ссылки на ненайденные файлы;

использование необъявленных или неинициализированных переменных;

ошибки построения циклов, условий и прочих стандартных структур;

несоответствие типов данных;

В случае если данный анализатор что-либо пропустит или если программист проигнорирует его предупреждения, интерпретатор завершит свою работу с ошибкой.

Пример работы синтаксического анализатора в VS Code показан на рисунке ниже.

Таким образом, выявление синтаксических ошибок является тривиальной задачей и не требует подробного рассмотрения.

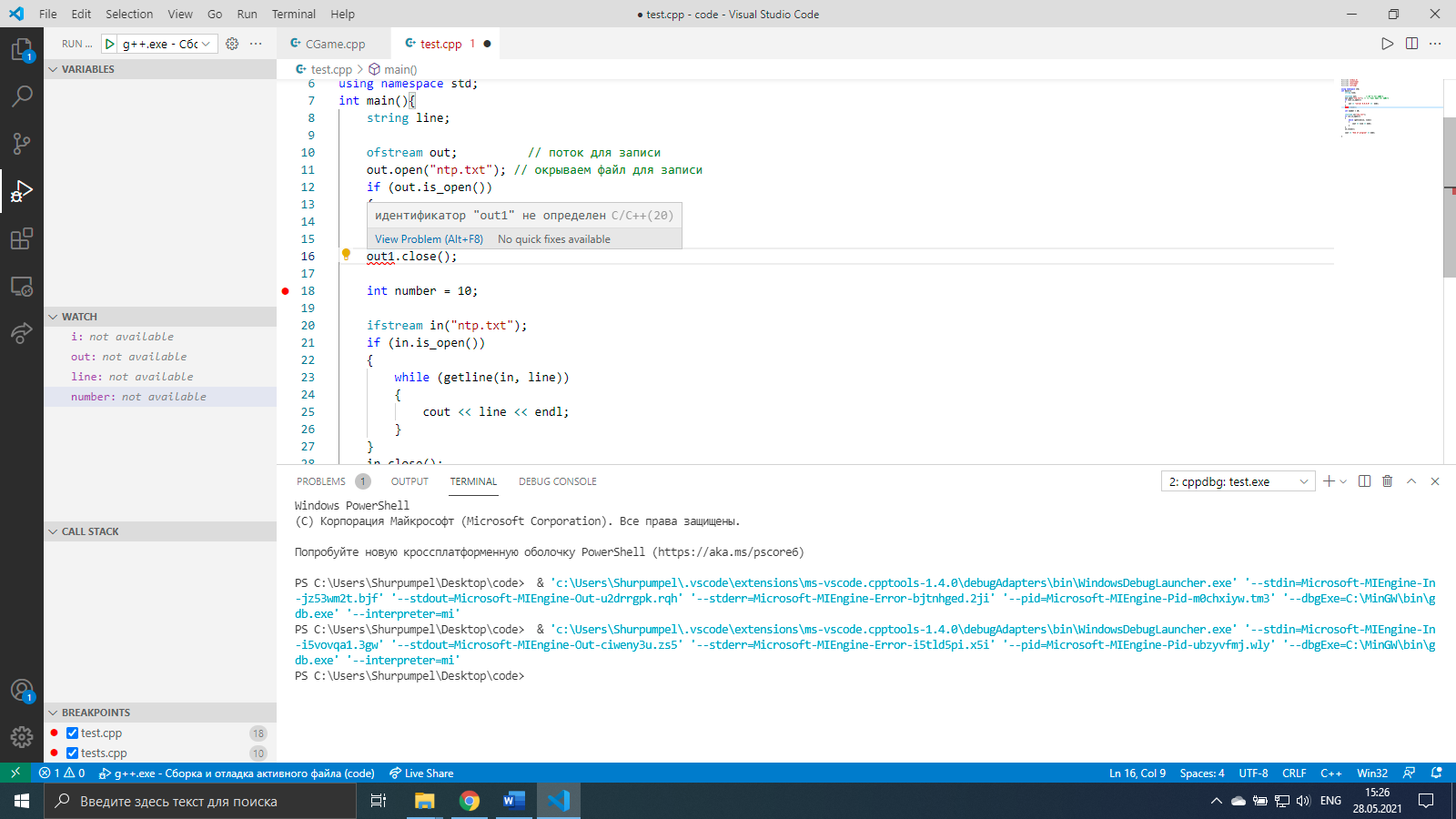


Рисунок Пример работы синтаксического анализатора

Логические ошибки – это любые ошибки, которые приводят к неверным результатам работы программы, при этом код может выполняться без сбоев и не приводить к аварийной остановке. Наличие именно таких ошибок в коде выявляется на этапе тестирования программного обеспечения. Для локализации ошибок и нахождения причин их возникновения применяется анализ путей выполнения программы и текущих значений переменных. Данный анализ производится при помощи отладочного вывода и специальных инструментов – отладчиков, которые позволяют пошагово выполнять программу и отслеживать состояние переменных в памяти.

# Методы тестирования

Тестирование ПО – это такой процесс, в котором проводятся испытания программного продукта. Оно проводится с целью проверить, насколько реальное поведение программы будет соответствовать ожидаемым прогнозам. Для этого проводится некоторое количество тестов, которые были выбраны по определенном принципам.

Существуют различные классификации тестирования:

По цели тестирования (функциональное, производительности, безопасности, локализации и т. д.)

По знанию внутреннего строения системы (черного/белого/серого ящика)

По позитивности сценария (позитивное, негативное)

По уровню тестирования (модульное, системное и т. д.)

По разработке тестовых сценариев (на основе модели, требований)

По степени автоматизации (ручное, автоматическое, полуавтоматическое)

По времени проведения тестирования (альфа, бета)

По исполнению кода (статическое, динамическое)

И множество других классификаций.

Классификация по объекту тестирования.

Функциональное тестирование – это тестирование ПО, с помощью которого проверяют, способно ли ПО решать какие-либо задачи при определенных условиях. Обычно такие тесты отвечают на вопросы: «сможет ли пользователь сделать это?» или «работает ли эта функция?». Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

Тестирование производительности — это тестирование, которое проводится с целью определения, как быстро работает вычислительная система или её часть под определённой нагрузкой. Также может служить для проверки и подтверждения других атрибутов качества системы, таких как масштабируемость, надёжность и потребление ресурсов. Данный вид тестирования включает в себя несколько видов тестирования: нагрузочное тестирование, стресс-тестирование, тестирование стабильности, конфигурационное тестирование. Нагрузочное тестирование (его еще часто называют испытанием на выносливость) – это проверка того, сможет ли система работать при определенных нагрузках, таких как большой объем данных или большое количество пользователей и т. д. Стресс-тестирование – это способ проверить работу системы при неожиданных или же редких рабочих нагрузках. Тестирование стабильности проверяет, сможет ли программа непрерывно работать в течение какого-то периода времени.

Проверка эргономичности– это тестирование, с целью проверки интерфейса. Определяется, насколько он прост в понимании и использовании. В основном данный вид тестирования применяется при разработке приложений. Такой метод невозможно автоматизировать, необходимы реальные пользователи, за которыми следят опытные дизайнеры пользовательского интерфейса.

Тестирование безопасности — оценка уязвимости ПО к различным атакам, обычно применяется к ПО, содержащему или обрабатывающему конфиденциальные данные. В ходе тестирования испытатель играет роль взломщика. Он может использовать любые средства от попыток узнать пароль с помощью внешних средств до просмотра несекретных данных в надежде найти ключ для входа в систему.

Также существует много тестов по объекту тестирования, но они узконаправлены, поэтому рассматриваться не будут.

Среди перечисленных выше методов тестирования самым важным является функциональное тестирование. Поэтому оно обязательно будет проводится для разрабатываемого ПМ НКСС. У модуля пользовательским интерфейсом является Cisco-подобная консоль, в которую пользователь вводит команды. Поэтому юзабилити-тестирования проводиться не будет. В рамках данной работы не поднимались вопросы безопасности приложения, так что этот параметр также не тестируется.

Классификация по уровню детализации

Модульное (unit) тестирование проверяет отдельные небольшие части приложения, которые могут функционировать отдельно от других частей. Как правило, тестируются отдельные функции или классы. Этот вид тестирования позволяет на начальном этапе разработки обнаружить многие ошибки.

Интеграционное тестирование уже направлено на проверку взаимодействия отдельных модулей друг с другом.

Системное тестирование проверяет приложение как единое целое, собранное из отдельных модулей.

Приемочное тестирование оценивает соответствие требований к программному продукту

Для ПМ НКСС будут использоваться все методы из данной классификации.

Классификация по знанию внутреннего строения системы

Тестирование методом черного ящика, также известное как функциональное тестирование – это такой способ тестирования ПО, при котором программа рассматривается как некий черный ящик. Тестировщики не знают о внутреннем устройстве программы, им известно лишь то, что программа может получать на вход и что должно в итоге получиться после ее выполнения.

Тестирование белого ящика, также тестирование стеклянного ящика, структурное тестирование — тестирование, которое учитывает внутренние механизмы системы или компонента (ISO/IEC/IEEE 24765). Обычно включает тестирование ветвей, маршрутов, операторов. При тестировании выбирают входы для выполнения разных частей кода и определяют ожидаемые результаты.

Тестирование серого ящика – это метод тестирования ПО, при котором тестировщик имеет представление об алгоритмах, применяемых в программе. При его выполнении нет сильной необходимости в доступе к коду, однако зачастую такой доступ есть. Тесты пишутся на основе знания алгоритма, архитектуры, внутренних состояний или других высокоуровневых описаний поведения программы.

В данном случае разработчик также является тестировщиком, поэтому будет использоваться метод белого ящика.

# Тестирование ПМ НКСС

При разработке ПМ ВКМ для тестирования был использован метод белого ящика, а также стратегия модульного тестирования (Unit-тестирования). Такое тестирование позволило на всех этапах разработки следить за правильностью выполнения кода.

Было написано несколько тестов, на рисунке ниже можно увидеть результат их выполнения.

Таблица . Список тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название теста | Выполняемые действия | Ожидаемый результат | Реальный результат | Результат прохождения |
| Правильное добавление сервера | Правильный ввод команды на добавление сервера | В файлах конфигурации должны появиться введенные настройки | В конфигурации появились введенные настройки | Тест пройден |
| Неправильное добавление сервера | Неправильный ввод команды на добавление сервера | Должно отобразиться сообщение об ошибке | Отобразилось сообщение об ошибке | Тест пройден |
| Правильное удаление сервера | Правильный ввод команды на удаление сервера | В файлах конфигурации не должны быть строки с введенными настройками | В конфигурации введенные настройки не найдены | Тест пройден |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Неправильное удаление сервера | Неправильный ввод команды на добавление сервера | Должно отобразиться сообщение об ошибке, файлы меняться не должны | Отобразилось сообщение об ошибке. Конфигурационные файлы не изменились | Тест пройден |
| Проверка на ввод show run | Вводится команда на добавление сервера  Вводится команда show run | Среди всех выведенных данных должны быть строки по введенным ранее настройкам | В выведенных данных есть строки с введенными серверами | Тест пройден |
| Проверка на show run | include ntp | Вводится команда на добавление сервера  Вводится команда show run | include ntp | Должны появиться строки только с введенными серверами | Появились строки с введенными серверами | Тест пройден |

Руководитель практики от МИЭТ / Касимов Р. А./

Руководитель практики   
от организации / Харитонов Р. Л./

Обучающийся / Федотов А.А./