Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.03.02 – Информационные системы и технологии
Направленность (профиль) образовательной программы Безопасность информационных систем

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«_____»_____2022 г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка обфускатора языков семейства .NET с использованием метода обратной разработки

Выполнил
студент группы 855-об _____ (подпись, дата) М.Е. Пимкин

Руководитель
доцент, канд. физ.-мат. наук _____ (подпись, дата) В.В. Ерёмина

Консультант
по безопасности и экологичности
доцент, канд. техн. наук _____ (подпись, дата) А.Б. Булгаков

Нормоконтроль
доцент, канд. техн. наук _____ (подпись, дата) В.Н. Адаменко

Благовещенск 2022

Министерство науки и высшего образо*вания Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики

Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____А.В. Бушманов
«_____»_____

ЗАДАНИЕ

К выпускной квалификационной работе студента Пимкина Максима Евгеньевича

1. Тема выпускной квалификационной работы: Разработка обфускатора языков семейства .NET с использованием метода обратной разработки
(утверждена приказом от 05.04.2022 №679-уч)

2. Срок сдачи студентом законченной работы 21.06.2022 г.

3. Исходные данные к выпускной квалификационной работе: отчет о прохождении преддипломной практики, нормативная документация, специальная литература

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов): описание предметной области, проектирование программы, программная реализация, информационная безопасность, безопасность жизнедеятельности

5. Перечень материалов приложения: техническое задание на разработку программного обеспечения

6. Консультанты по выпускной квалификационной работе
по безопасности и экологичности – Булгаков А.Б., доцент, кандидат технических наук

7. Дата выдачи задания 07.02.2022 г.

Руководитель выпускной квалификационной работы: Ерёмина Виктория Владимировна, доцент, канд. физ.-мат. наук
(фамилия, имя, отчество, должность, ученая степень, ученое звание)

Задание принял к исполнению _____

РЕФЕРАТ

Выпускная квалификационная работа содержит 49 с, 16 рисунков, 1 приложение, 25 источников, 8 нормативных ссылок.

IL, C#, .NET, MVVM, AVALONIAUI, ОБФУСКАТОР, АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ, РАЗРАБОТКА, ПРОЕКТИРОВАНИЕ

В работе произведены проектирование и разработка приложения «Обфускатор языков семейства .NET».

Цель работы: проектирование и разработка приложения «Обфускатор языков семейства .NET», в функционал которого входит обфускация кода, написанного на языке программирования семейства .NET. Помимо стандартных функций, приложение оснащено возможностью сокрытия отображения найденных идентификаторов. Данное приложение разработано для обеспечения противодействия злоумышленнику в попытке взлома другого приложения методом обратной разработки.

В ходе работы было выполнено:

− анализ существующих решений;

− разработан программный продукт на основании выбранных архитектуры и средств разработки;

− оценены информационная безопасности и безопасность жизнедеятельности.

Результатом выпускной квалификационной работы является приложение «Обфускатор языков семейства .NET».

НОРМАТИВНЫЕ ССЫЛКИ

В настоящей бакалаврской работе использованы ссылки на стандарты и нормативные документы:

ГОСТ Р ИСО/МЭК 27002-2021. Информационная технология. Методы и средства обеспечения безопасности. Свод норм и правил применения мер обеспечения информационной безопасности.

ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.

ГОСТ 30772-2001. Ресурсосбережение. Обращение с отходами. Термины и определения.

ГОСТ Р 51275-2006. Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положения.

СП 2.2.3670-20. Санитарно-эпидемиологические требования к условиям труда.

СанПиН 1.2.3685-21. Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания.

СП 52.13330.2016. Естественное и искусственное освещение.

НПБ 105-03. Нормы пожарной безопасности. Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ОЗУ	Оперативное запоминающее устройство
ПЗУ	Постоянное запоминающее устройство
ПО	Программное обеспечение
ЭВМ	Электронная вычислительная машина
ЯП	Язык программирования
IL	Intermediate Language

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире происходит большое количество взломов различного программного обеспечения. Существуют группы единомышленников, которые занимаются взломом средств защиты и программ, защищенных с помощью этих средств защиты, таких как, Denuvo.

Далеко не всегда то или иное средство защиты ПО может обеспечить должный уровень защиты, не ударив по производительности работы программы. В случае, если разработчику или заказчику ПО не обязателен высокий уровень защиты собственного программного продукта или есть требование высокой производительности и минимальной защиты программного продукта, прибегают к обфускации кода.

Обфускация кода подразумевает под собой запутывание кода, что усложнит работу злоумышленника по получению доступа к логике программы и данным, содержащихся в программе в константном виде, а также усложнит процесс внесения каких-либо модификаций в программу.

Существует достаточно большое количество решений для произведения обфускации, но они направлены на исходный код, который пишет разработчик, а также на один или два ЯП.

В ходе данной работы будет рассмотрена разработка компьютерного приложения, произведен анализ уже существующих решений, рассмотрена безопасность жизнедеятельности, даны рекомендации по работе за ЭВМ.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предприятия

1.1.1 Функции, организационная структура и общие сведения о предприятии

ООО «Стожары» – торгово-монтажная компания, занимающаяся комплексными системами безопасности. Ее обязанностями является продажа противопожарного оборудования и поставка различного программного обеспечения.

Главными функциями ООО «Стожары» являются продажа современного программного обеспечения из области информационной безопасности, предоставление услуг по пожарной и транспортной безопасности.

Организационная структура ООО “Стожары” показана на рисунке 1.1.

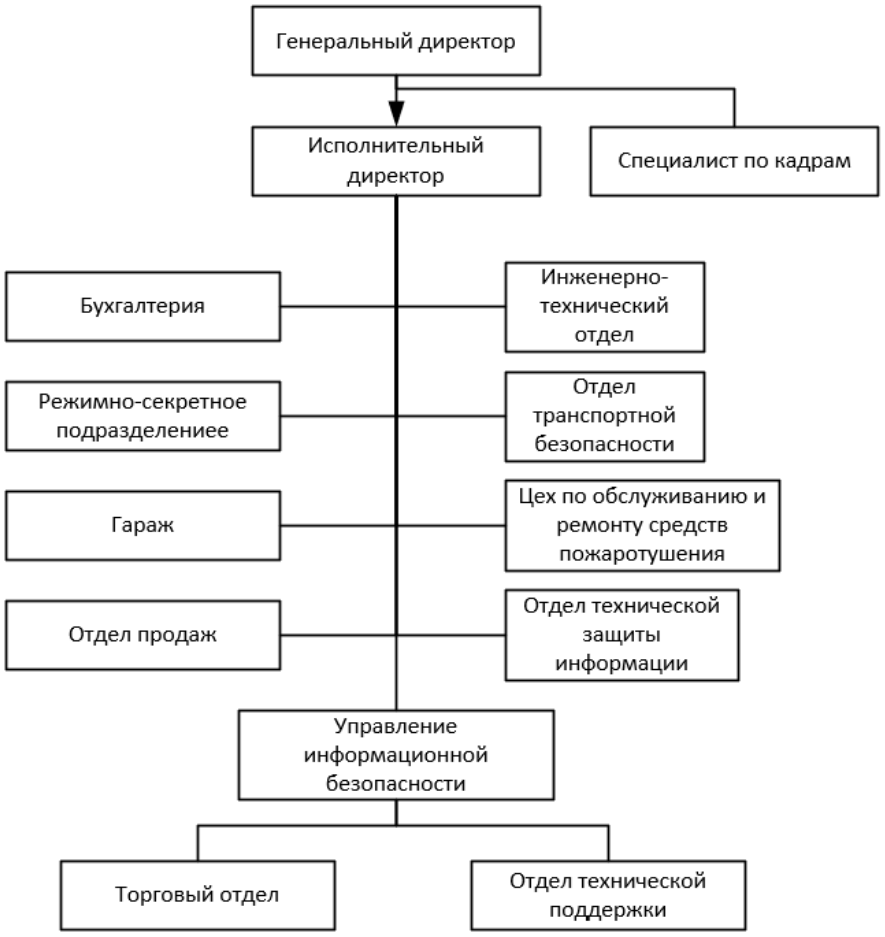

Рисунок 1.1 – Организационная структура ООО “Стожары”

Деятельность каждого отдела можно описать следующим образом:

– Дирекция – занимаются управлением ООО «Стожары»;

– Бухгалтерия предназначена для сбора данных об имуществе и обязательствах предприятия;

– Инженерно-технический отдел занимается техническим обслуживанием турникетов, составлением планов эвакуации, систем видеонаблюдения, пожарно-охранной сигнализацией, системами пожаротушения;

– Отдел продаж занимается продажей различного противопожарного оборудования;

– Цех по облуживанию и ремонту средств пожаротушения занимается заправкой, техническим обслуживанием и ремонтом огнетушителей.

– Отдел транспортной безопасности занимается аттестацией людей, обеспечивающих транспортную безопасность.

– Отдел технической защиты информации занимается проектами по защите информации от утечки по техническим каналам на объектах информатизации;

– Управление информационной безопасности занимается поставками различного программного обеспечения, операционных систем, выпуском ЭЦП, аттестацией объектов ИСПДн (Информационная система персональных данных), продажей, настройкой и сопровождением СЗИ;

– Режимно-секретное подразделение занимается секретной информацией;

– Лаборатория специальных исследований занимается проверкой различного оборудования на утечки информации по различным каналам связи.

1.1.2 Внешний и внутренний документообороты предприятия

Внешний документооборот – это все входящие и исходящие документы компании, которыми она обменивается с контрагентами, клиентами и контролирующими органами. К ним относятся счета-фактуры, накладные, акты выполненных работ, электронная отчётность и другие виды документов. Внешний документооборот представлен на рисунке 2.

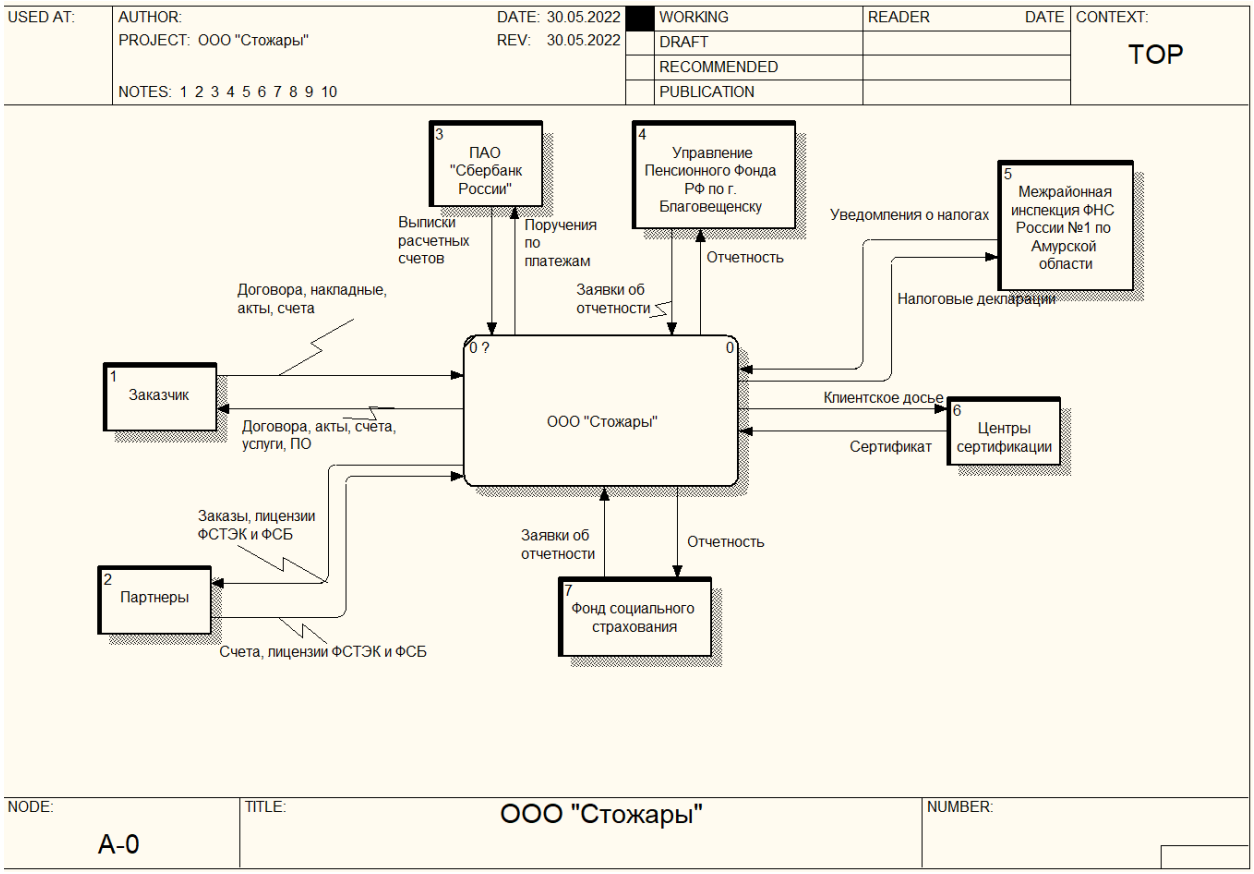

Рисунок 1.2 – Внешний документооборот ООО «Стожары»

Внутренний документооборот – все входящие и исходящие документы компании, перемещающиеся строго внутри компании между отделами и сотрудниками. К внутреннему документообороту относятся приказы, уставы, заявления, протоколы совещаний, инструкции, положения о структурных подразделениях, служебные записки и другие документы.

Внутренний документооборот представлен на рисунке 1.3.

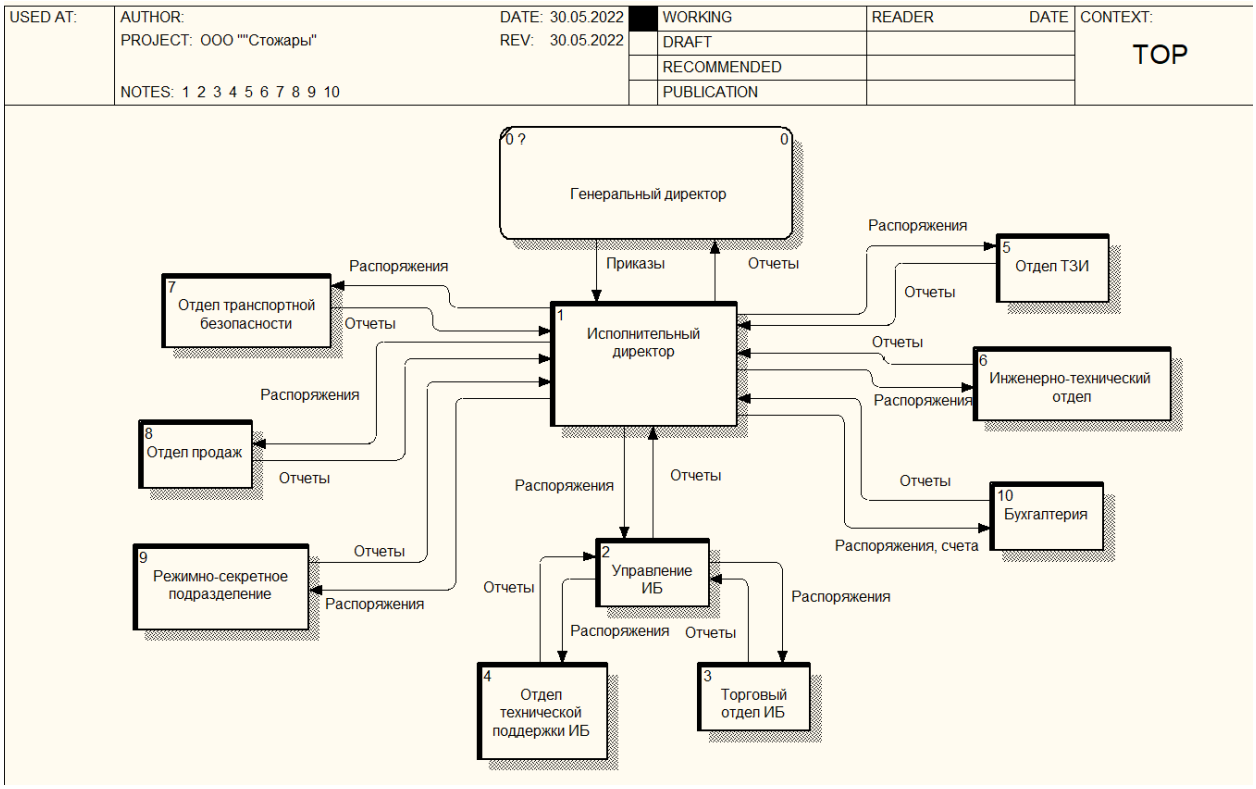

Рисунок 1.3 – Внутренний документооборот ООО «Стожары»

1.2 Обзор и анализ существующих решений

Существует относительно небольшое количество существующих действующих обфускаторов для языков .NET. Стоит отметить главный недостаток всех далее рассматриваемых решений, который заключается в обфускации исходного кода, который пишет программист во время выполнения своей работы.

По причине того, что в ООО «Стожары» не используется в данный момент какой-либо обфускатор, рассмотрим сторонние готовые решения. Для дальнейшего рассмотрения выбраны такие обфускаторы как NET Reactor, {SmartAssembly}, Dotfuscator, CodeVeil.

{SmartAssembly} является собственностью Redgate и предоставляет функцию обфускации кода языков программирования C# и VB. Стоит 1145 долларов за одну лицензию на 1 год использования. Поставляется в 3 вариациях: Standard, Pro и Personal. Пример интерфейса представлен на рисунке 1.5.

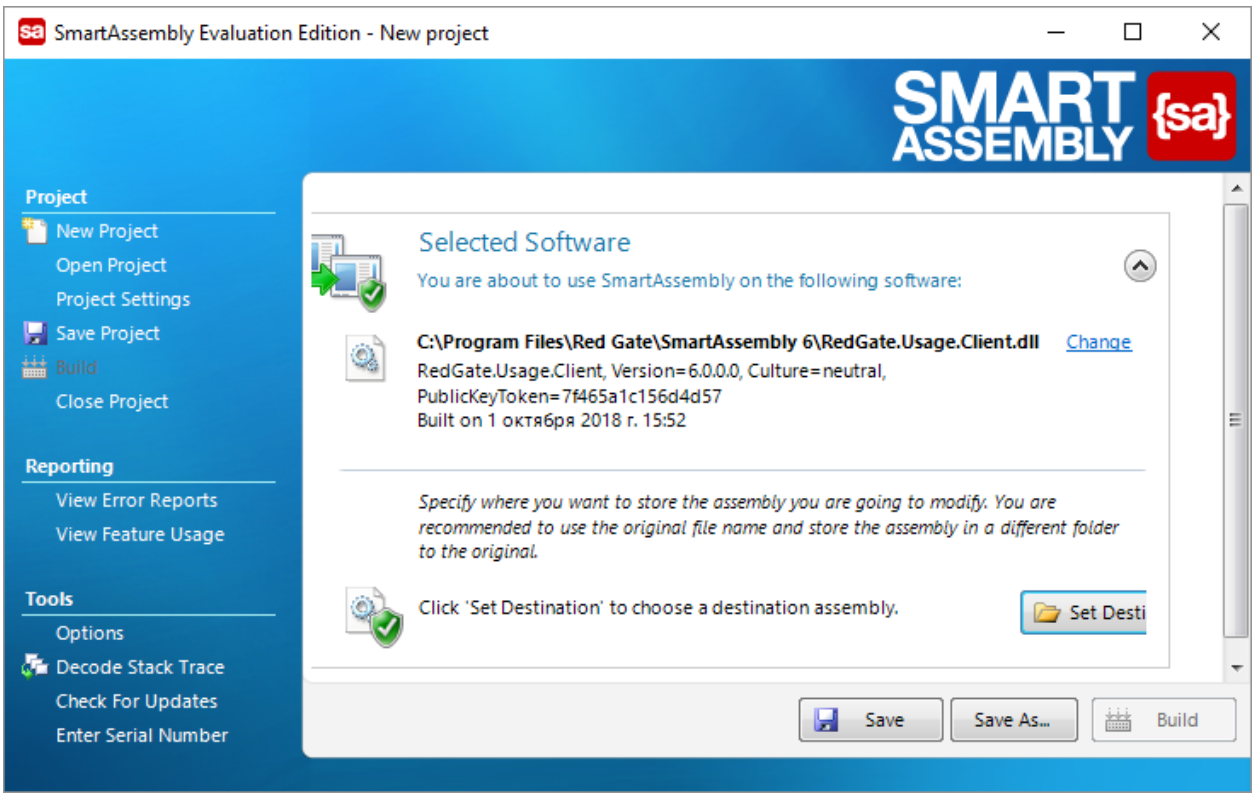

Рисунок 1.4 – Вид интерфейса {SmartAssembly} выбора решения для обфускации

Данный обфускатор помимо непосредственно обфускации предлагает клиенту возможность зашифровать и сжать при необходимости ресурсы программы для снижения занимаемого объема на носителе информации, а также удалить ненужные по версии программы.

.NET Reactor – обфускатор от компании Eziriz. Лицензия на 1 год на 1 компьютер для частного использования стоит 199 долларов, а для компании 299 долларов. Вид интерфейса представлен на рисунке 1.5.

Несмотря на то, что на официальном сайте разработчика данная разработка позиционируется как обфускатор для .NET, данная программа не способна провести обфускацию программ, написанных на языках семейства .NET, отличных от языка C#.

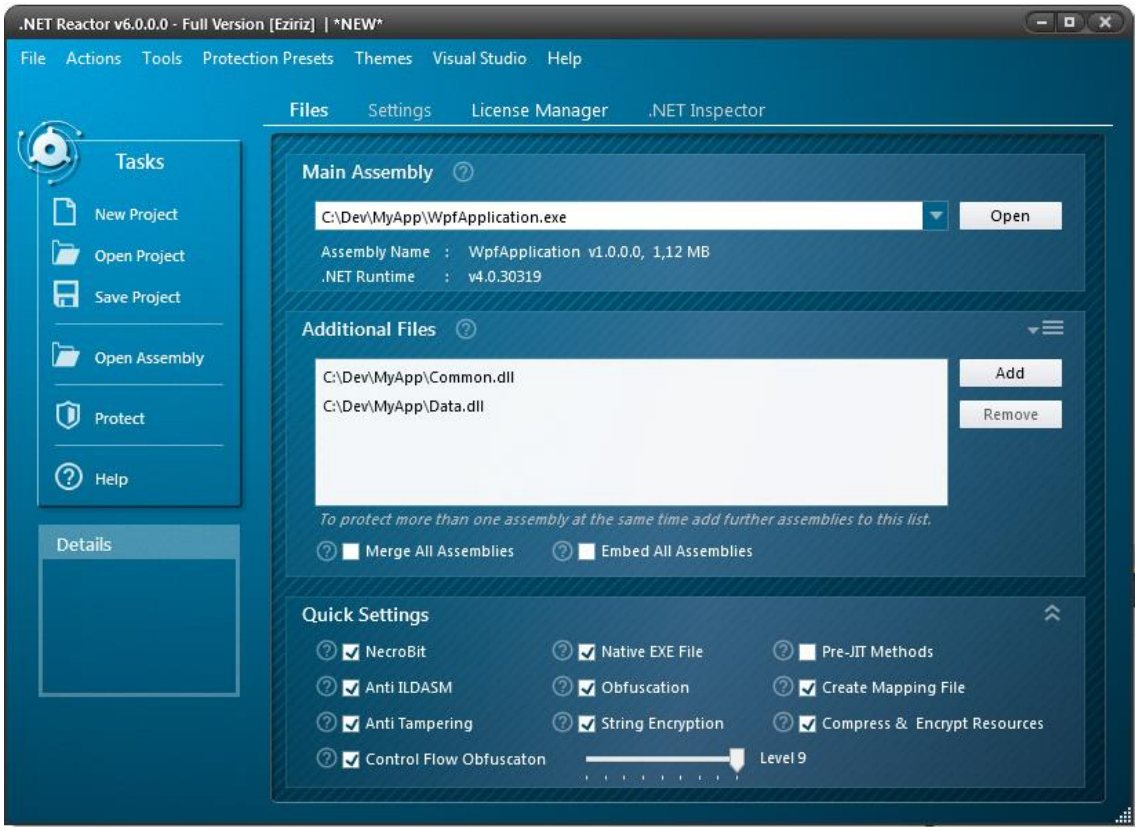

Рисунок 1.5 – Вид интерфейса .NET Reactor

Dotfuscator – обфускатор от PreEmptive. При заказе данного обфускатора цена будет уточняться и будет зависеть от количества разработчиков и машин, которые собирают проекты, а также от количества лет, на которое заказчик собирается купить лицензию. Цена же будет от 1900 долларов. Присутствует бесплатная пробная версия, для которой необходимо зарегистрироваться на официальном сайте и оставить заявку на получение данной лицензии. Существует также Community версия, которая устанавливается в среду разработки Visual Studio.

Отличительной чертой данного обфускатора является только защита от вмешательства в уже запущенную программу. При попытке вмешательства будет вызвана остановка программы со случайной ошибкой для сокрытия проверки на вмешательство.

CodeVeil – бесплатный обфускатор от XHEO INC, который более не обновляется разработчиком. Данный обфускатор встраивался в среду разработки Visual Studio 2005, 2008, 2010. Позволял шифровать программу, работал с

такими проектами как Windows Applications, ASP.NET, Windows Services, EXE, DLL. Позволял запрещать декомпилировать файлы.

Исходя из всего вышеописанного, можно сказать, что данные обфускаторы отличаются друг от друга незначительно с точки зрения функционала. Также стоит отметить то, что данные языки направлены на обфускацию именно проекта, а не исполняемого файла, что подразумевает обфускацию исходного кода, который пишет разработчик. Программы, написанные на языке семейства .NET перед тем, как стать исполняемым файлом на этапе сборке транслируются в IL-код, из которого уже будет собрана непосредственно программа. В случае обфускации IL-кода можно не обращать внимание на то, на каком языке изначально была написана программа.

2 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ

2.1 Цели и функции программы

Целью программы является противодействие злоумышленнику при модификации или попытке получения доступа к логике программы, написанной на языке семейства .NET.

Функциями программы является запутывание IL-кода, путём переименования всех пользовательских идентификаторов классов, переменных, методов и других типов данных на случайно сгенерированные, сохраняя возможность дальнейшего запуска программы и без внесений каких-либо существенных изменений в быстродействие защищаемой программы.

2.2 Требования к программе

2.2.1 Общие требования

Разработанная программа должна состоять из функциональных модулей, которые описаны в пункте 2.3. Модульность программы обеспечит дальнейшее упрощенное снесение изменений и добавление новых функций.

Программа должна корректно воспринимать IL-код, производить генерацию уникальных идентификаторов в рамках одного процесса обфускации.

Интерфейс программы не должен вводить в заблуждение, а интуитивность интерфейса должна быть высокой. Цветовая схема должна быть комфортной для работы, не должна быть раздражающей. Текст должен быть виден при расположении зрительного органа на рекомендуемом расстоянии от монитора.

2.2.2 Требования к математическому обеспечению

Требования к математическому обеспечению не предъявляются.

2.2.3 Требования к лингвистическому обеспечению

Интерфейс программы выполнен полностью русифицированным текстом, что позволяет увеличить интуитивность интерфейса и снизить сложность обучения работе с данной программой.

2.2.4 Требования к техническому обеспечению

Минимальные системные требования для работы с данной программой:

– операционная система: Windows, Linux;

– память ОЗУ: 128 МБ;

– память ПЗУ: 50 МБ;

– периферийные устройства: клавиатура, монитор, мышь.

2.3 Характеристика функциональных модулей программы

2.3.1 Модуль пользовательского графического интерфейса

Модуль пользовательского графического интерфейса предоставляет пользователю удобный интерфейс для взаимодействия с программой.

Данный модуль отвечает за отображение интерфейса, а также его перерисовку при изменении данных.

Перерисовка выполняется после добавления файлов для дальнейшей обфускации, а также после того, как выполнена полностью обфускация выбранных файлов. После нажатия кнопки, отвечающей за повторную обфускацию другой группы файлов, модуль пользовательского графического интерфейса перерисовывает отображение с учетом того, что список файлов становится пустым, а также выводит кнопку для запуска обфускации, скрывая результаты прошлой обфускации.

Модуль пользовательского графического интерфейса действует в рамках одного окна, меняя отображение лишь отдельных частей окна. AvaloniaUI не предоставляет возможности сделать многостраничное приложение, вследствие чего для сокращения количества окон было сделано одно окно, в котором перерисовываются отдельные части в соответствии с изменением данных или этапа работы программы.

2.3.2 Модуль генерации случайных идентификаторов

Модуль генерации случайных идентификаторов отвечает за создание уникальных идентификаторов по мере нахождения идентификаторов в тексте обфусцируемой программы.

Модуль генерации случайных идентификаторов работает следующим образом: в случае, если был найден в тексте программы идентификатор,

вызывается метод GenerateName, который вернет случайную строку, состоящую из случайного количества символов в пределах от 5 до 25, а допустимыми символами будут являться все цифры от 0 до 9 и заглавные символы латинского алфавита от A до Z.

Схема генерации случайного идентификатора представлена на рисунке 2.1.

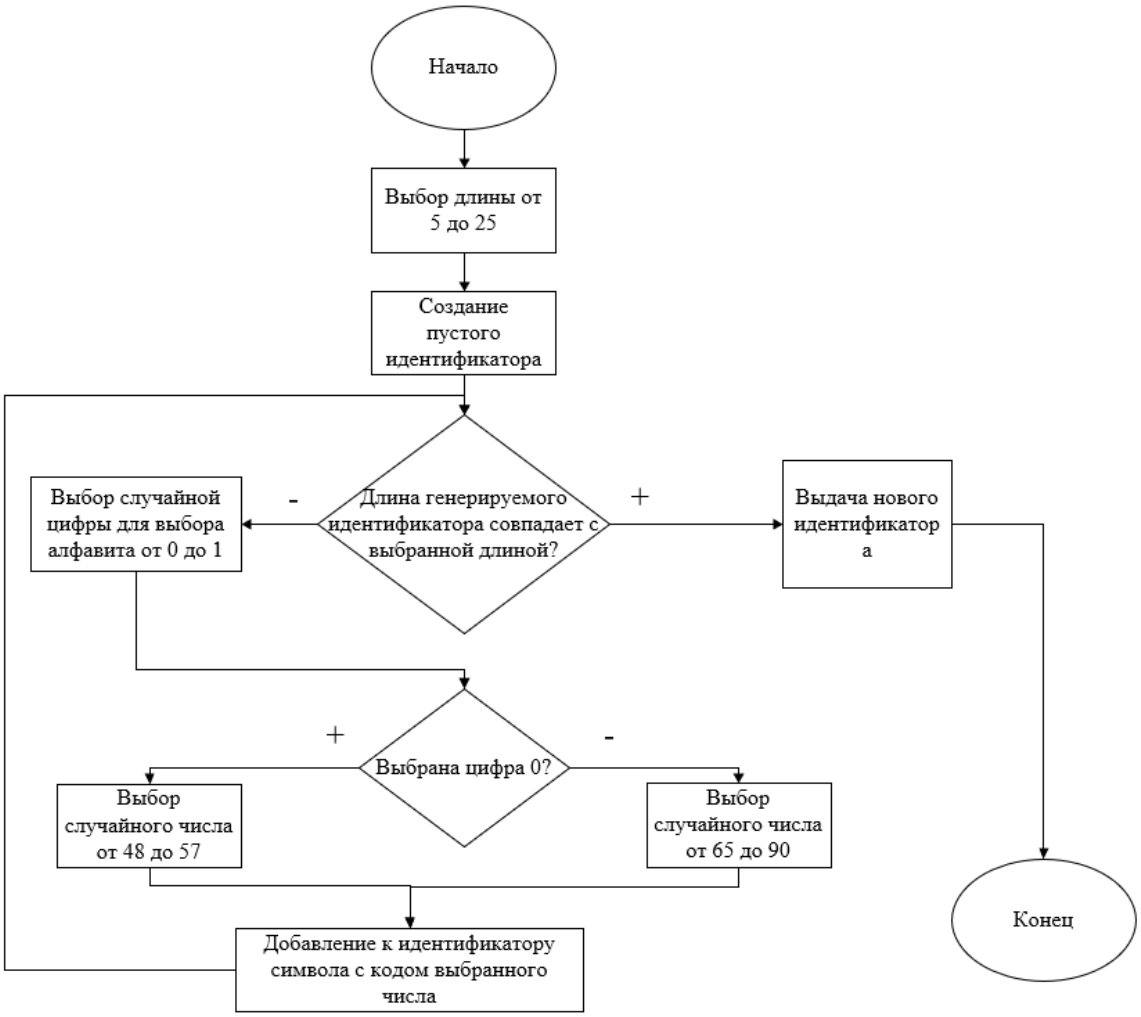

Рисунок 2.1 – Схема генерации случайного идентификатора

Генерация нового идентификатора начинается с выбора длины будущего идентификатора. Длина нового идентификатора является псевдослучайным числом от 5 до 25, зависящим от времени системы. После выбора длины будет запущен цикл, повторяющийся столько раз, сколько выбрана длина нового идентификатора. Во время работы данного цикла, случайно выбирается

набор допустимых символов. После выбора конкретного набора, выбирается случайно элемент из выбранного набора. Данный элемент добавляется в новый идентификатор, после чего цикл повторяется до тех пор, пока не будет добавлено столько символов, сколько выбрана длина.

Код, отвечающий за генерацию представлен на рисунке 2.2.

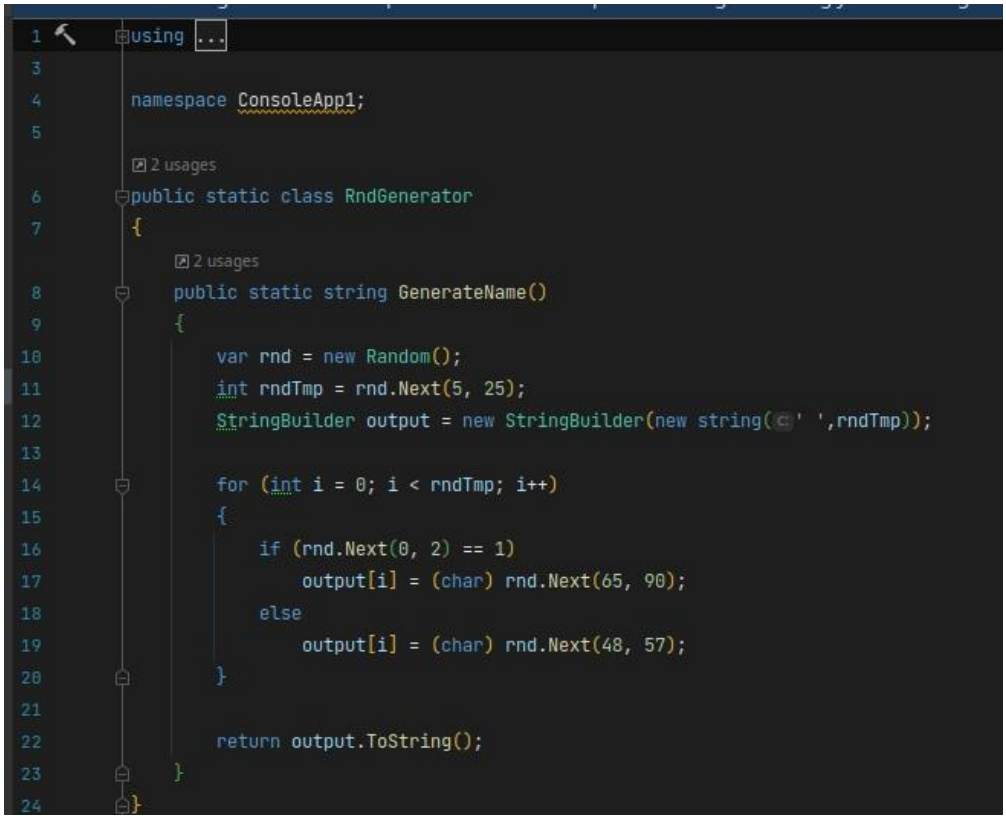

Рисунок 2.2 – Код генерации случайного идентификатора

2.3.3 Модуль временного хранения информации

Модуль временного хранения информации служит для хранения во время работы программы служебной информации такой как сгенерированный идентификатор, найденный тип идентификатора и найденное имя идентификатора.

Данная информация хранится в строго структурированном списке, а элементами данного списка являются объекты класса, содержащие информацию для каждого отдельного идентификатора, найденного в ходе работы программы, а также сгенерированное имя, полученное по запросу из модуля генерации случайных идентификаторов.

Данный модуль помимо хранения информации предоставляет методы для работы с этой информацией: метод поиска информации, а также метод модификации уже имеющейся информации.

Доступ пользователю к данному модулю осуществляется путем ограниченного просмотра информации, выводимой в модуль пользовательского графического интерфейса.

2.3.4 Модуль работы с файлами

Модуль работы с файлами представляет из себя статический класс, служащий для ввода информации в программу, а также вывода обфусцированной информации в выходной файл. Входными данными для программы являются исполняемые файлы, а также файлы библиотек, написанные на языках семейства .NET. Также программа может выполнять обфускацию текстового файла, содержащего код программы, написанный на языке IL.

IL-код может быть получен пользователем с помощью любого дизассемблера, после чего сохранен в текстовом файле и подвергнут обфускации.

В случае работы с исполняемыми файлами или файлами библиотек, обфускатор использует дизассемблер Microsoft ildasm. Для компиляции файлов, прошедших обфускацию, используется ассемблер IL-кода, предоставляемый Microsoft ilasm.

Модуль работы с файлами создает папку Output, находящуюся рядом с каждым отдельно выбранным файлом, в которую помещается выходной файл IL-кода или другого типа, который зависит от исходного типа файла.

Во время работы модуля поиска идентификаторов и обфускации, модуль работы с файлами предоставляет возможность дописывания в выходной файл обфусцированной строки.

2.3.5 Модуль поиска идентификатора и обфускации

Модуль поиска идентификатора и обфускации позволяет программе найти тип идентификатора, непосредственно имя идентификатора, используемое во входной программе, а также создать объект, включающий найденную

информацию и сгенерированный новый идентификатор, необходимый для дальнейшей замены во входной программе.

Схема работы модуля поиска идентификатора и обфускации представлена на рисунке 2.3.

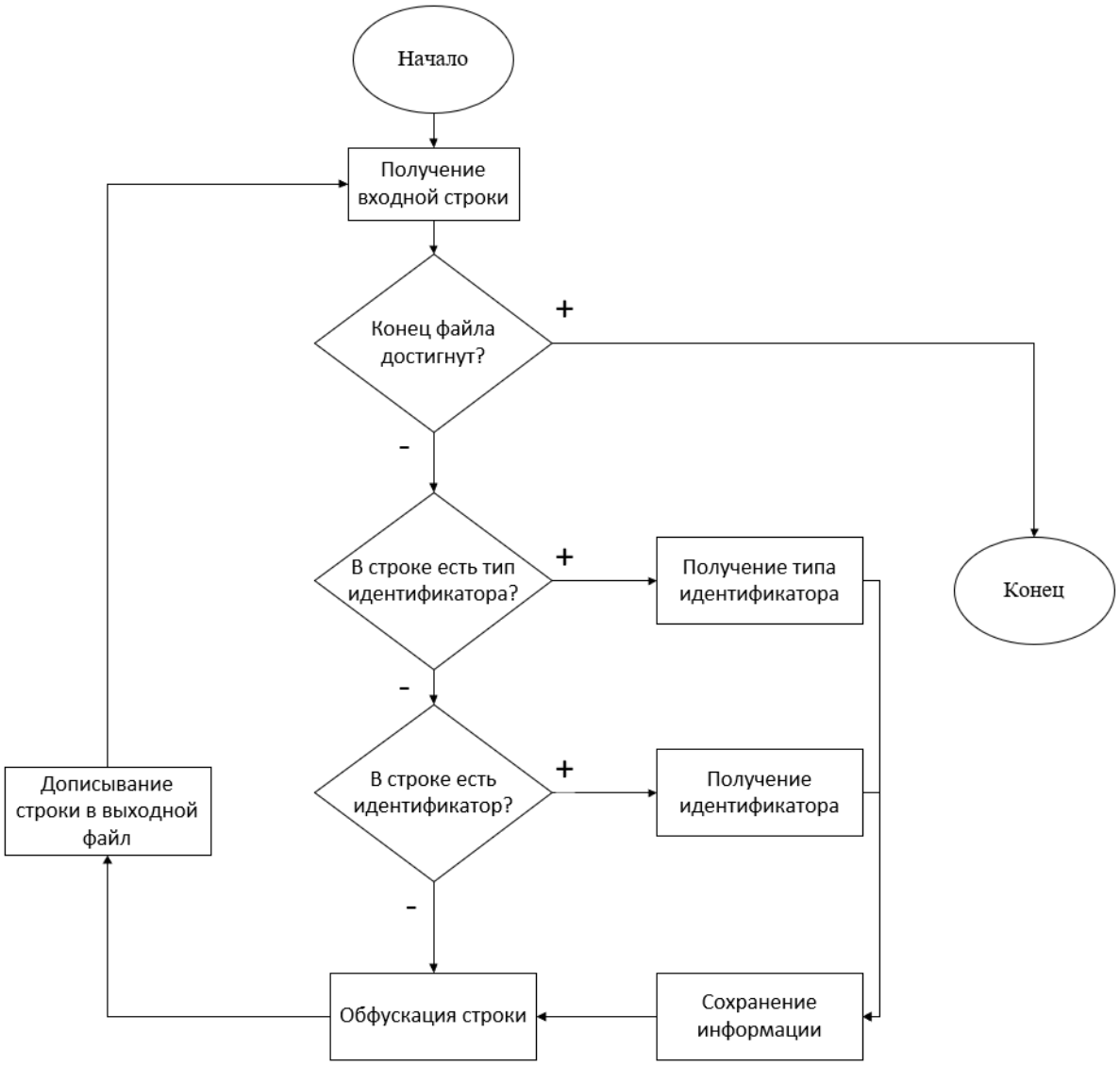

Рисунок 2.3 – Схема работы модуля поиска идентификатора и обфускации

Обфускация производится построчно: полученная входная строка проверяется на наличие типа идентификатора, в случае отсутствия которого, производится проверка на наличие самого идентификатора. Независимо от результата поиска идентификатора и его типа, производится обфускация входной строки путём замены любого ранее найденного идентификатора на сгенерированный. В случае, если был найден тип или идентификатор, они заносятся

в модуль временного хранения информации, а также вызывают модуль генерации случайных идентификаторов для генерации идентификатора. После чего строка, прошедшая обфускацию, отправляется в модуль работы с файлами для дальнейшей записи в выходной файл.

Стоит отметить, что обфусцировать точку входа в программу нельзя, потому что это может привести к непредвиденным последствиям, начиная от невозможности скомпилировать программу, заканчивая сбоями при работе, если случайно изменить точку входа.

Для того, чтобы обфускатор не изменил точку входа, были добавлены исключения, а именно класс Program и метод Main, содержащийся в классе Program. В случае, если модуль поиска идентификатора и обфускации натыкается на строку, содержащую данный класс и метод, он игнорирует их, не занося в модуль временного хранения информации данных о найденном классе и методе, что позволяет проигнорировать их, не нанося ущерба работе программы.

Стоит отметить также ошибки именования, совершаемые некоторыми программистами. Иногда программисты называют объекты классов именем класса, что может ввести в заблуждение человека, работающего с данным кодом в дальнейшем по причине того, что вместо вызова метода объекта, программист, работающий с данным кодом, написанным другим человеком, может посчитать это вызовом метода класса, что может вызвать недопонимание, а порой и заставить человека тратить время на поиск ошибки, которой нет.

Обфускатор при работе с подобным кодом применяет одно именование для всех одинаковых названий. С одной стороны, это дает злоумышленнику понимание того, что какие-то идентификаторы были изначально названы одинаково, с другой же стороны, это всё так же может привнести путаницу при работе злоумышленника не с IL-кодом, а с кодом, например, C#.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 Архитектура программы

При разработке используется архитектурный шаблон MVVM (Model – View – ViewModel) по причине того, что в дальнейшем он обеспечит простое внесение изменений и дополнений в программу, потому что разграничивает логику, отображение и логику отображения.

Согласно данному шаблону, при разбиении добавляются директории, соответствующие той или иной части шаблона, показанные на рисунке 3.1.

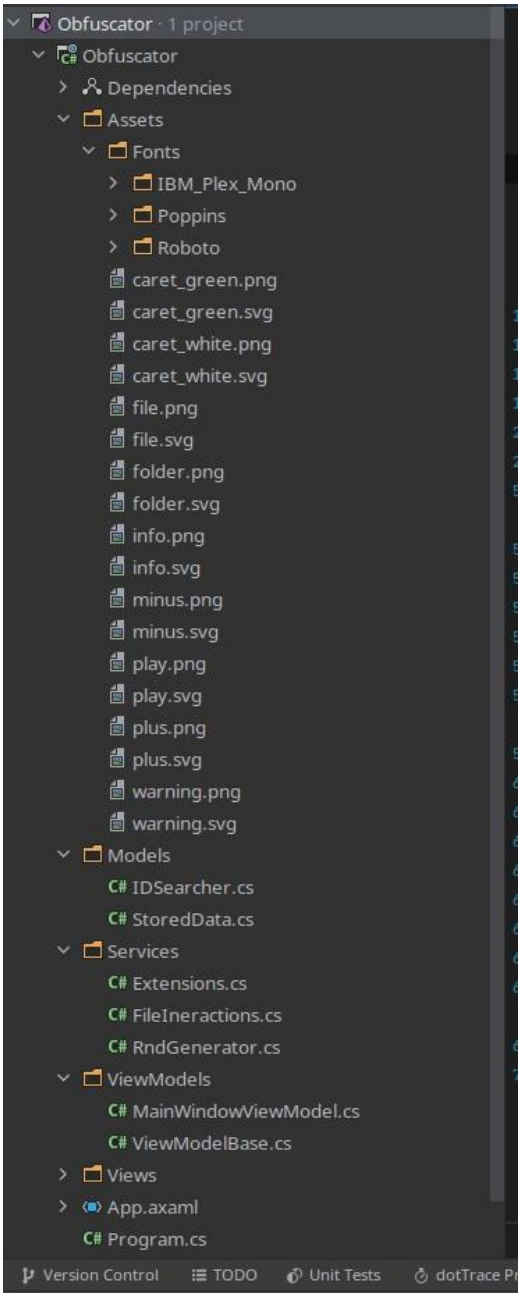

Рисунок 3.1 – Структура проекта

3.2 Средства разработки программы

Во время разработки используется студенческая лицензия для JetBrains Rider, Manjaro Linux, AvaloniaUI.

JetBrains Rider выбран по причине того, что Microsoft Visual Studio не устанавливается на Linux, что усложняет разработку программы для Windows и Linux.

Manjaro Linux выбрана по причине дружелюбности системы к пользователю.

AvaloniaUI выбрана в связи с возможностью кроссплатформенной разработки под Windows и Linux без отдельной разработки под каждую платформу.

3.3 Описание графического пользовательского интерфейса

Графический пользовательский интерфейс сделан при помощи языка разметки XAML и AvaloniaUI.

Пользователю во время работы с программой доступны кнопки добавления файлов, начала обфускации и повторение, если необходимо провести обфускацию нескольких программ, что подразумевает выбор нового набора файлов и последующую генерацию новых идентификаторов. Графический интерфейс оконной формы представлен на рисунках 3.2, 3.3, 3.4.

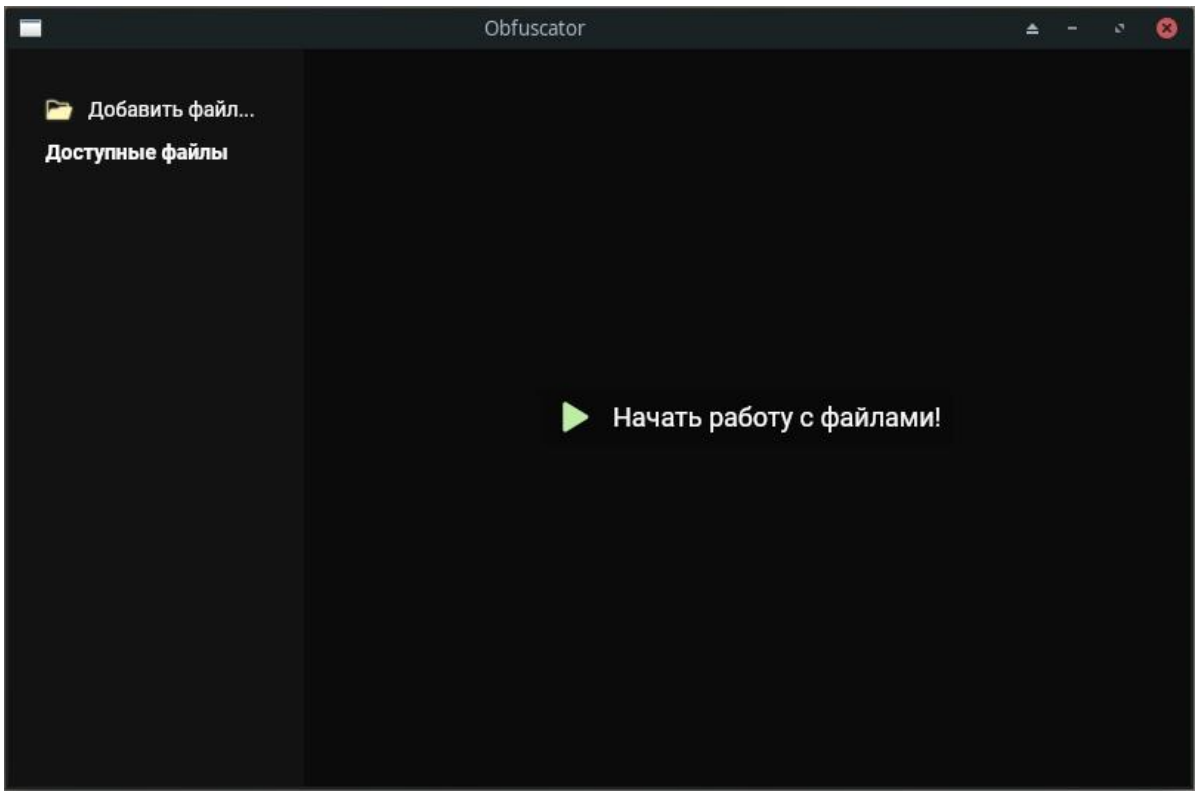

Рисунок 3.2 – Графический интерфейс программы при невыбранных файлах

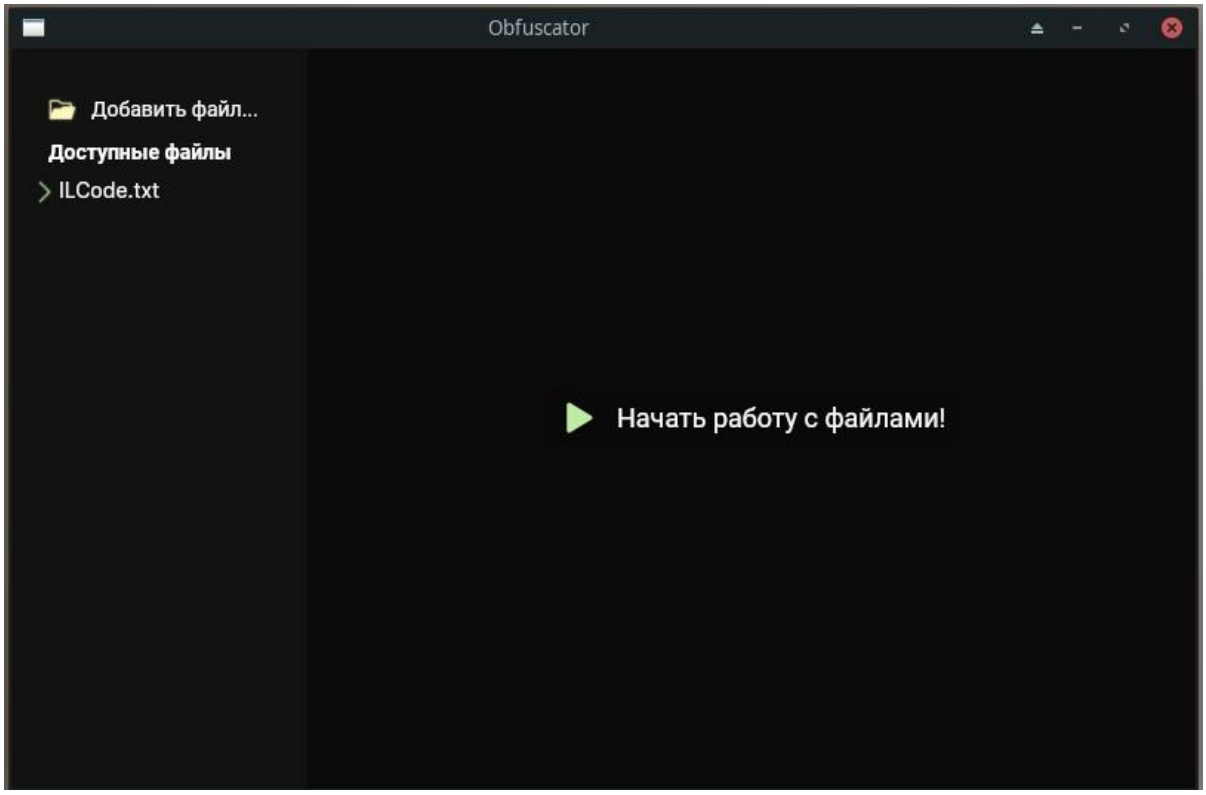

Рисунок 3.3 – Графический интерфейс программы после выбора файла

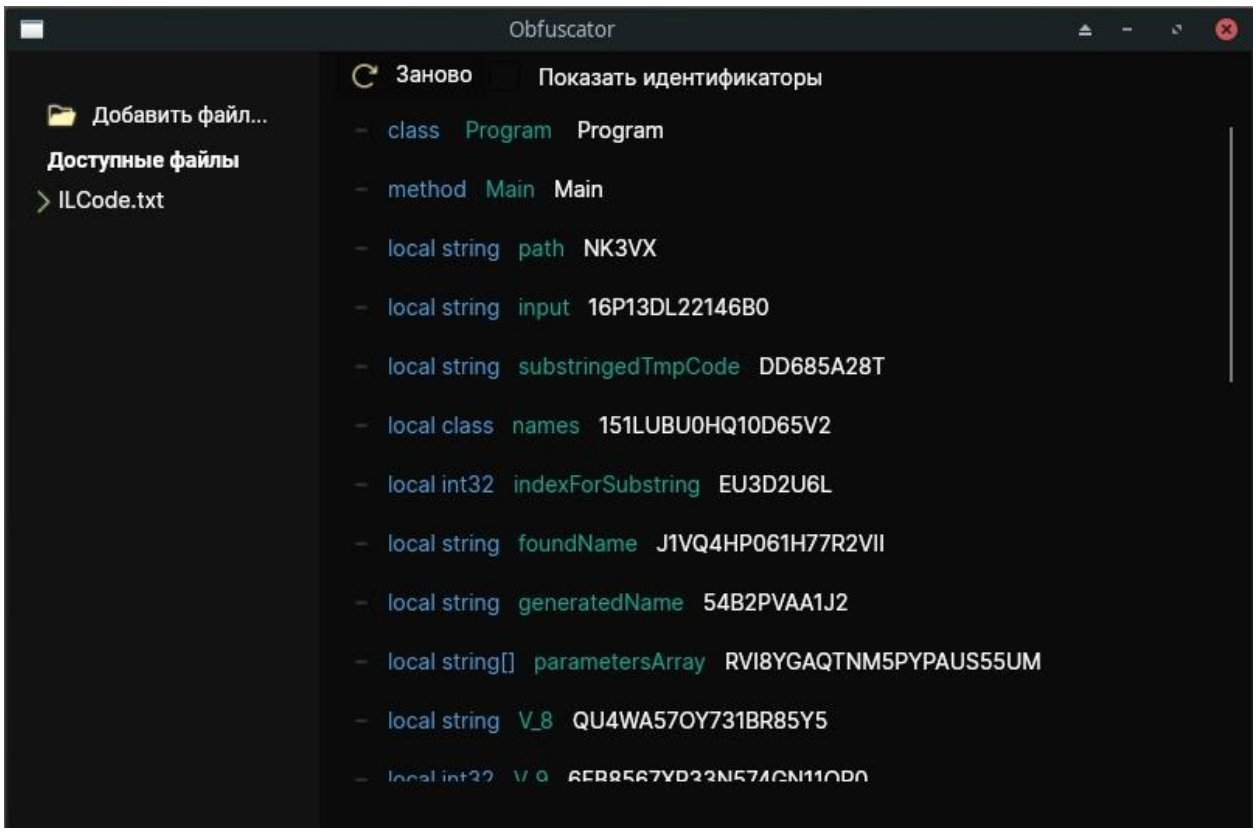

Рисунок 3.4 – Графический интерфейс программы после запуска процесса обфускации

3.4 Описание работы программы

Алгоритм работы программы можно описать следующим образом: пользователь выбирает один или более файлов, которые необходимо обфусцировать. После выбора файлов и запуска программы, программа рассматривает каждый файл в отдельности, начиная с первого выбранного. Во время обработки конкретного файла, программа построчно рассматривает весь IL-код, содержащийся в файле. Каждая строка проверяется на наличие типа и идентификатора. В случае, если тип был найден, то рассматривается два случая: идентификатор содержится в той же строке, что и тип, или идентификатор находится дальше. Независимо от того, есть ли в строке тип или идентификатор, строка подвергается обфускации, а затем дописывается в выходной файл.

В случае, если был найден идентификатор, который не должен быть обфусцирован, программа не запоминает найденный идентификатор, что позволяет сохранить точку входа в программу. Изменение точки входа в

программу может привнести непоправимые последствия в работу программы или не дать скомпилировать файл в исполняемый.

Для получения исходного кода программы может быть использован любой декомпилятор, позволяющий получить IL-код, а не исходный. В случае получения такого кода без помощи поставляемого ildasm, необходимо вручную сохранить текстовый файл с кодом, после чего его загрузить в программу.

Для реализации метода обратной разработки с обфускатором поставляется ildasm – декомпилятор Microsoft, который способен получить IL-код.

Вид исходного файла с IL-кодом представлен на рисунке 3.4.

File Edit Search View Document Help

ILCode.txt_Out

```
// Type: Program
// Assembly: ConsoleApp1, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// MVID: EE49D1D2-315B-42F9-A6FA-601932A2BDB8
// Location: /home/maxark/RiderProjects/ConsoleApp1/ConsoleApp1/bin/Release/net6.0/ConsoleApp1.dll
// Sequence point data from /home/maxark/RiderProjects/ConsoleApp1/ConsoleApp1/bin/Release/net6.0/ConsoleApp1.

.class private auto ansi beforefieldinit
  Program
    extends [System.Runtime]System.Object
{
  .custom instance void [System.Runtime]System.Runtime.CompilerServices.CompilerGeneratedAttribute::.ctor()
    = (01 00 00 00 )

  .method private hidebysig static void
    '<Main>$'(
      string[] args
    ) cil managed
  {
    .entrypoint
    .maxstack 4
    .locals init (
      [0] string path,
      [1] string input,
      [2] string substringedTmpCode,
      [3] class ConsoleApp1.StoredData names,
      [4] int32 indexForSubstring,
      [5] string foundName,
      [6] string generatedName,
      [7] string[] parametersArray,
      [8] string V_8,
      [9] int32 V_9,
      [10] string[] V_10,
      [11] string item,
      [12] string item_V_12,
      [13] valuetype [System.Runtime]System.Runtime.CompilerServices.DefaultInterpolatedStringHandler V_13
    )
```

Рисунок 3.4 – Вид файла с IL-кодом до обфускации

Вид того же исходного файла с IL-кодом представлен на рисунке 3.5.

Обфусцированный код становится непонятным, по причине того, что все идентификаторы были заменены на случайно сгенерированные, что не позволяет злоумышленнику легко понять логику работы защищаемой программы.

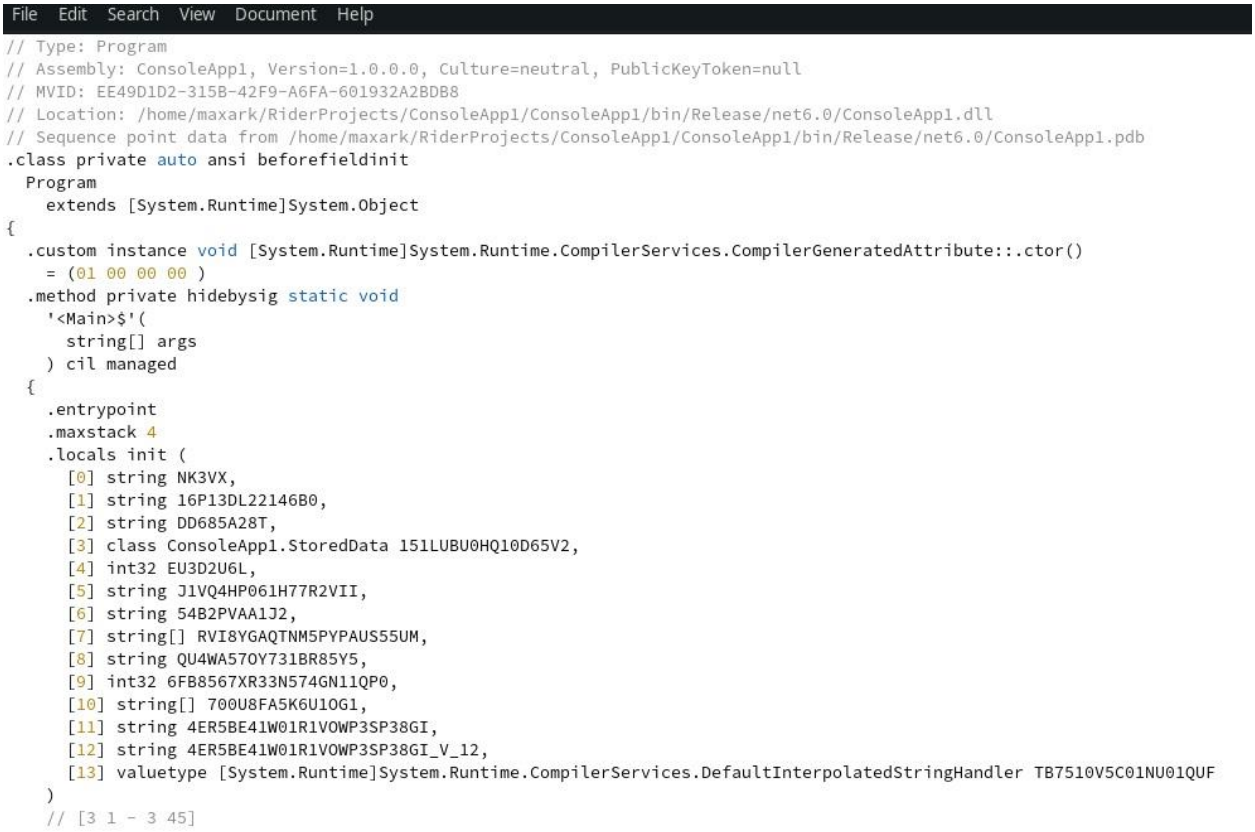

Рисунок 3.5 – Вид файла с IL-кодом, прошедшим обфускацию

3.5 Возможные дальнейшие доработки программы

Для развития программы можно добавить обфусцирование кода, написанного на языке Assembler, что позволит охватить очень большое количество программ.

Также можно добавить обфускацию исходного кода, написанного на разных языках программирования. Это позволит защитить исходный код, написанный разработчиком, но еще не прошедший транслирование в объектный код.

В данную разработку возможным вариантом дальнейшего развития окажется добавление шифрования скомпилированного файла, что позволит

обеспечить дополнительный уровень препятствования злоумышленнику при взломе.

Для обеспечения гарантии целостности обфусцированной программы, можно внедрять контрольную сумму в код в качестве комментария к коду. Однако, любое изменение файла повлечет неизбежное изменение контрольной суммы, отчего добавленный комментарий в код программы вызовет несоответствие добавленной контрольной суммы и контрольной суммы, полученной вследствие расчета для файла кода. Это может стать дополнительной трудностью для злоумышленника при попытке внедрения модифицированной программы, если он не знает логику сравнения контрольной суммы, добавленной в файл. Для сравнения контрольной суммы, необходимо будет убрать строку контрольной суммы, после чего сохранить файл и высчитать для него контрольную сумму, которую необходимо сравнить с убранной строкой. В случае, если контрольные суммы совпадают, это означает, что модификации кода не произошло, а в случае разных контрольных сумм будет явно видно, что были внесены какие-либо изменения в код.

4 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

По причине того, что генерация идентификаторов происходит случайно, смысла в чересчур серьезной защите программы нет. Для противодействия пониманию логики разрабатываемой программы, исполняемый файл будет обфусцирован.

Если злоумышленник знает всю логику программы, то вернуть идентификаторам исходный вид он не сможет по причине того, что генерация происходит с помощью псевдослучайных чисел, базирующихся на времени системы. Даже если злоумышленник узнает время, когда создавался тот или иной идентификатор, вернуть исходный идентификатор он не сможет, потому что он нигде не используется. Новые идентификаторы не привязаны к старым, а вид новых специально сделан так, чтобы казалось, что это не так.

Единственным способом получения данных с программы остается получение оперативной памяти в момент, когда работа программы была завершена, но программа не была закрыта, но для получения информации таким способом злоумышленнику требуется обладать необходимыми навыками и иметь в своем арсенале специализированные технические средства.

5 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

5.1 Безопасность

5.1.1 Анализ эргономики программы

Эргономичность – наибольшая производительность при наименьшей вероятности ошибки при работе с программой.

Обфускатор языков семейства .NET – программный продукт, разрабатываемый в рамках выпускной квалификационной работы бакалавра. Назначением данного продукта является обеспечение противодействия злоумышленнику в простом получении доступа к логике и алгоритмам защищаемых программ.

Критериями для оценки эргономичности программы можно считать:

- сложность обучения;

- интуитивность графического интерфейса, выражающуюся в совпадении между изображением в интерфейсе и предполагаемым действием;

- цена ошибки, принимаемую как стоимость ошибки, произошедшей по винее пользователя, как например, некорректный ввод или неверные действия пользователя.

Интуитивность графического интерфейса и скорость обучения являются связанными напрямую показателями. В случае, если графический интерфейс будет непонятен пользователю или не соответствовать назначению, пользователю придется потратить дополнительное время на обучение работы с программным продуктом.

Цена ошибки может быть определена как количество возникающих ошибок и их стоимость. Цену ошибки можно оценить как минимальную, по причине того, что в случае ввода некорректных данных, ошибочных действий со стороны пользователя, исходные данные никак не будут затронуты и не будут испорчены, что позволяет перезапустить программу или начать процесс обфускации заново. Стоит отметить, что в случае возникновения ошибки или при вводе некорректных данных, а также дальнейшем перезапуске

программы, сотрудник пребывает в состоянии бездействия, но время на перезапуск и выбор файлов повторно занимает достаточно малое время, что позволяет пренебрегать этими факторами.

Для минимизации возможных ошибок, интерфейс программы не подразумевает большое количество активных элементов, например, кнопки и переключатели. На стартовом окне, показанном на рисунке 4.1, можно отметить лишь кнопку добавления файлов и кнопку старта обфускации. В случае попытки обфускации пустого перечня файлов, программа не даст выполнить данное действие, предупредив пользователя о том, что необходимо добавить файлы.

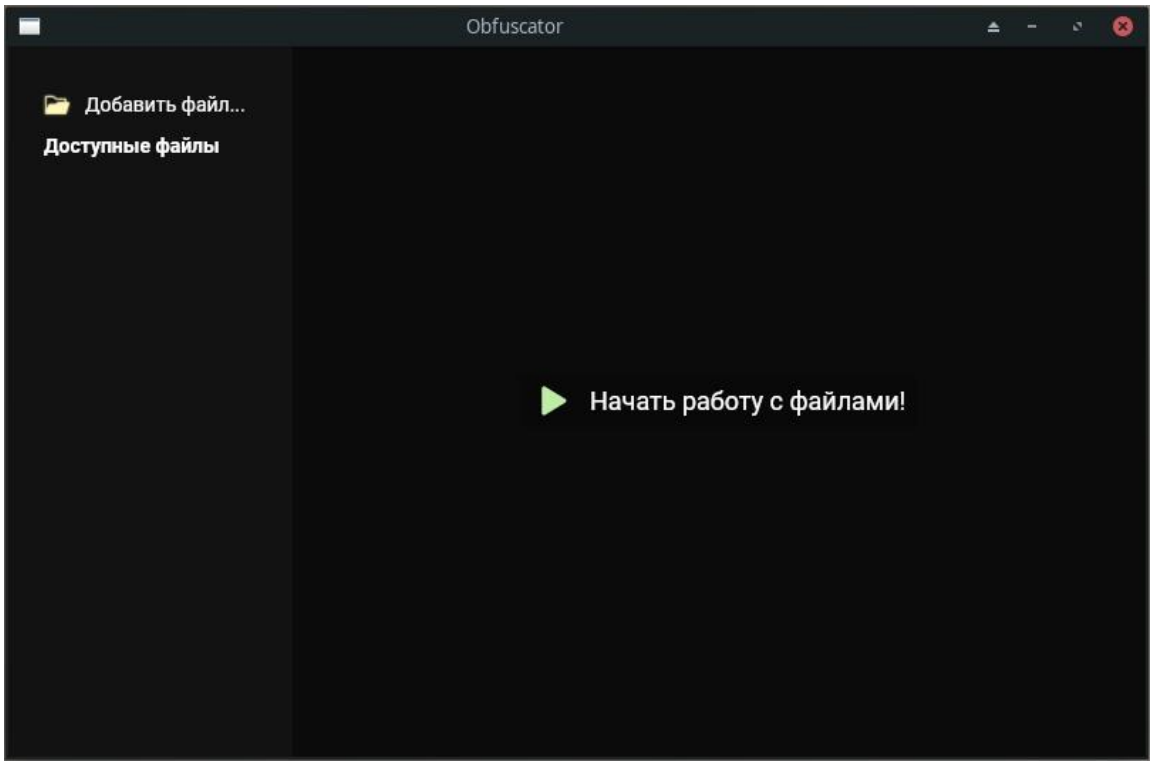

Рисунок 4.1 – Стартовое окно обфускатора

Кнопки подписаны и выполняют именно то, что объявлено на кнопке, что позволяет сделать интуитивно понятный интерфейс программы.

Из всего вышеперечисленного можно утверждать, что *цена ошибки* для данного программного продукта является минимальной, *интуитивность графического интерфейса* оценивается как интуитивно понятна, по причине того, что в интерфейсе нет несоответствия между изображением и назначением, а

сложность обучения работе с данным программным продуктом является очень низкой в связи с тем, что пользователю доступно минимальное количество элементов управления, а интуитивность данных элементов является очень высокой.

5.1.2 Анализ опасных и вредных факторов на рабочем месте пользователя ЭВМ

Данная программа рассчитана на использование только на ЭВМ, что подразумевает необходимость в анализе рабочего места пользователя с ЭВМ.

Работа за ЭВМ подразумевает работу в сидячем положении. Рабочее место для работы в сидячем положении должно соответствовать требованиям, описанным пунктом 6.3 СП 2.22.3670-20, а именно:

– пространство для размещения ног высотой не менее 600 мм;

– на уровне колен глубина не менее 450 мм;

– на уровне стоп глубина не менее 600 мм;

– шириной не менее 500 мм.

Площадь одного рабочего места пользователя ПЭВМ с использованием плоских дискретных экранов обозначена в пункте 249 СП 2.2.3670-20 и составляет не менее 4,5 кв.м.

ПЭВМ необходимо размещать таким образом, чтобы показатели освещенности соответствовали значениям, установленным в СанПиН 1.2.3685-21 таблице 5.25 норм, согласно пункту 251 СП 2.2.3670-20.

5.2 Экологичность

Экологичность – качество чего-либо, отражающее его способность не наносить вреда окружающей природе. Отходы производства – остатки любого сырья, отработанных материалов, любых других продуктов, произведенных или полученных в ходе любой деятельности предприятия. Для увеличения экологичности, предприятие должно обеспечить утилизацию должным образом своих отходов производства. Утилизация отходов – главная задача, поставленная перед предприятием для обеспечения экологической безопасности окружающей среды. Утилизация отходов является трудоемким и сложным

процессом, для чего необходимы специально обученный и подготовленный персонал, состоящий из специалистов, способных должным образом утилизировать тот или иной вид отходов.

Каждое предприятие должно обеспечить уничтожение отходов. Утилизация отходов должна соответствовать необходимым нормам и стандартам, несоблюдение которых может повлечь за собой получение предприятием серьезных штрафов и санкций, а также закрытие работы предприятия.

Организация, производящая какой-либо продукт, производит отходы, указанные в ГОСТ 30772-2001:

− вторичная продукция – вещества, материалы, комплектующие изделия, детали, функциональные узлы, блоки, агрегаты от различных объектов, утратившие свои потребительские свойства и не пригодные для дальнейшей эксплуатации в соответствии с директивными требованиями и/или нормативной документацией, но представляющие собой товарную продукцию.

− отходы производства – остатки сырья, материалов, веществ, изделий, предметов, образовавшиеся в процессе производства продукции, выполнения работ (услуг) и утратившие полностью или частично исходные потребительские свойства.

Ко вторичной продукции, в организации использующей ЭВМ, можно отнести комплектующие и периферию ЭВМ.

Возможна переработка объектов, состоящих преимущественно из пластмассы. Объекты микроэлектроники, которые не подлежат переработке, отправляются на вторичное использование.

Возможна продажа вторичной продукции как товаров бывших в употреблении. Тогда для организации появляется возможность окупить некоторые объекты производства, не затрачивая ресурсов на утилизацию.

Отходы производства подлежат только утилизации. Для утилизации отходы производства сортируются по типу материала, например, стекло, металлы и другие типы, после чего отправляются в центры утилизации, которые утилизируют отходы в соответствии с необходимыми требованиями.

5.3 Безопасность при возникновении чрезвычайных ситуаций

Пожар является одной из самых распространенных чрезвычайных ситуаций на предприятиях. Основными причинами возникновения пожара можно считать: неосторожное обращение с огнем или легко-воспламеняемыми веществами и предметами вблизи открытых источников огня, курение в неустановленных для этого местах, оставленные без присмотра электроприборы и использование электроприборов с какой-либо физической неисправностью, пренебрежительное отношение к правилам пожарной безопасности при проведении любых работ и т.д.

Согласно НПБ 105-03, помещения с ЭВМ являются пожароопасными в категории В1 – В4, в которых могут содержатся материалы, способные гореть при взаимодействии с водой или друг другом. Все провода должны быть спрятаны в кабель-каналы или стены, а путь эвакуации не должен быть загорожен мебелью или другими объектами.

Для предотвращения возможного пожара необходимо соблюдать следующие правила:

– не хранить и не применять горючие жидкости, взрывчатые вещества, баллоны с газами и рядом с ЭВМ;

– не использовать электронагревательные приборы;

– не эксплуатировать провода электроприборов с поврежденной изоляцией;

– не пользоваться поврежденными розетками, и прочим электрооборудованием;

– не накрывать светильники, бытовые приборы бумагой, тканью и другими горючими материалами;

– не курить в помещении;

– оставлять без наблюдения включенную в сеть радиоэлектронную ПЭВМ;

– не пользоваться неисправной аппаратурой;

– не разрешается ремонтировать блоки ЭВМ непосредственно в помещениях, где они располагаются;

– не нарушать правила эксплуатации ПЭВМ;

– раз в 3 месяца необходимо проводить санитарную очистку.

По окончании работы необходимо обесточить все электроприборы, осмотреть помещения на наличие признаков возгорания, а также необходимо выключить автомат питания в распределительном щите, если такой имеется.

Если случилось возгорание, необходимо позвонить в пожарную службу, сообщить всю необходимую информацию, подготовить к эвакуации материальные ценности, документацию и покинуть здание через запасные выходы. Если нет возможности покинуть здание, то необходимо закрыться в менее задымлённой комнате, не дать дыму попадать в комнате любыми подручными средствами и открыв все окна ожидать помощи спасательной бригады.

5.4 Физические упражнения и рекомендации при работе за ЭВМ

Люди стали проводить слишком большое количество времени за компьютером, что приводит к высокой нагрузке как на опорно-двигательный, так и на зрительный аппараты. При незначительных изменениях проблема может быть незаметна, однако постепенно формируются серьезные нарушения, такие как: снижения зрения, искривление осанки и артриты. В таком случае рекомендуется выполнять комплекс упражнений для снятия усталости за компьютером, а также соблюдать рекомендации, предназначенные для комфортной работы за ЭВМ и сохранения здоровья пользователя ЭВМ. Благодаря регулярным тренировкам можно предотвратить появление многих проблем.

В качестве рекомендаций при работе с ЭВМ стоит отметить:

– Дистанция между монитором и глазами не менее 45 см и не более 70 см.

– Клавиатура и экран стоит располагать прямо перед собой, чтобы минимизировать повороты.

– Голову необходимо держать прямо, а руки необходимо располагать на клавиатуре так, чтобы запястья были расслаблены.

– Спину стоит необходимо держать ровной, однако позволительно немного откинуть спинку стула.

– Необходимо делать периодический отдых. Раз в час стоит вставать с рабочего места и делать для себя перерыв. Не лишним будет выполнение в это время гимнастики, простой прогулки по помещению или выход на улицу для прогулки. Стоит снимать напряжение с глаз, по причине того, что работа за компьютером подразумевает частое напряжение зрительного органа человека. Для снятия напряжения с глаз стоит делать зрительную гимнастику.

Упражнения для снятия усталости с кистей рук и плечевого пояса.

а) Упражнение можно выполнять как сидя, так и стоя. Левую руку вытягиваем вперед, правую поднимаем вверх. Меняем положения рук, поочередно. Темп выполнения средний.

б) Положение стоя. Руки тыльной стороной кисти прижать к поясу. Свести локти вместе голову наклонить вперед. Локти развести в стороны и пытаться свести за спиной, голову наклонить назад.

в) Выполнять сидя на стуле. Поднять руки вверх сжимать и разжимать поочередно кисти рук.

Упражнения для снятия напряжения с туловища.

а) Исходное положение, стоя, руки за голову, ноги чуть шире плеч. Поворачивать таз влево и вправо. Плечевой пояс должен быть неподвижен.

б) Положение аналогично первому упражнению. Тазом делаем круговые вращения почасовой стрелки и против часовой стрелки, поочередно.

в) Стойка – ноги врозь. Наклоняемся вперед, правая рука скользит по ногам вниз, а левая поднимается вдоль тела. Далее проделываем то же самое, но меняем положение рук.

Упражнения для улучшения кровообращения в мозговой области.

а) Исходное положение на стуле, руки свесить, расслабиться. Медленно наклоните голову назад. Считаем до трех, медленно. Затем занимаем исходное положение. Затем медленно наклоняем голову вперед. Считаем до трех и возвращаемся в исходное положение.

б) Исходное положение, сидя на стуле, руки на поясе. Делаем все как в первом упражнении, но голову наклоняем сначала к левому плечу, потом к правому.

в) Можно сидя или стоя. Левую руку заносим за голову и тянемся к правому плечу, поворачиваем голову на лево. Считаем до трех и проделываем все то же самое, но с правой рукой и голову поворачиваем на право.

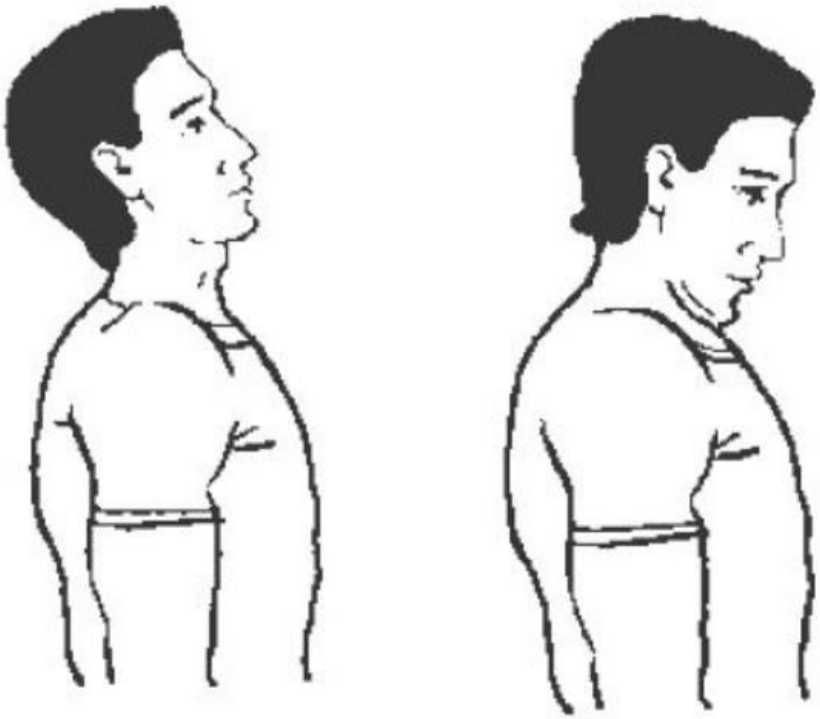

Рисунок 5.1 – Иллюстрация упражнений для шейного отдела

ЗАКЛЮЧЕНИЕ

На сегодняшний день с помощью метода обратной разработки производят взломы программного обеспечения в достаточно больших масштабах. В особенности этому подвержены игры, распространяемые не бесплатно, а также дорогостоящее программное обеспечение.

С помощью обратной разработки можно получить доступ к логике программы, зачастую в понятном виде, если не было произведено обфускации исходного кода, а также доступ к константным переменным, которые могут хранить информацию для подключения к серверу, например, порт для подключения, IP-адрес, а также такие переменные могут быть служебными и содержать логин администратора для подключения.

Обфускация не позволит полностью защитить программный продукт от взлома и модификации, но позволит замедлить злоумышленника, что даст время разработчику, например, для закрытия возможностей проникновения нарушителя.

В ходе выполнения бакалаврской работы был осуществлен анализ существующих решений обфускаторов .NET, выявлен их главный недостаток. На основании данного недостатка были определены требования к разработке.

В рамках темы были рассмотрены и описаны необходимые функциональные модули системы, описана структура приложения и описан алгоритм работы приложения.

Для разработанного приложения произведен анализ информационной безопасности, безопасности жизнедеятельности и экологичности в рамках проекта, были даны рекомендации при работе за ЭВМ.

Разработанное приложение позволяет проводить обфускацию IL-кода и программ, написанных на языках семейства .NET.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Гаврилов, М. В. Информатика и информационные технологии : учебник для вузов / М. В. Гаврилов, В. А. Климов. — 4-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2022. — 383 с. — (Высшее образование). — ISBN 978-5-534-00814-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/488708 (дата обращения: 13.02.2022).

2 Гунько, А. В. Системное программирование в среде Linux : учебное пособие / А. В. Гунько. — Новосибирск : Новосибирский государственный технический университет, 2020. — 235 с. — ISBN 978-5-7782-4160-2. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/98735 (дата обращения: 14.03.2022). — Режим доступа: для авторизир. пользователей.

3 Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для вузов / С. В. Зыков. — Москва : Издательство Юрайт, 2022. — 155 с. — (Высшее образование). — ISBN 978-5-534-00850-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/490423 (дата обращения: 01.04.2022).

4 Костин, В. Н. Методы и средства защиты компьютерной информации: аппаратные и программные средства защиты информации : учебное пособие / В. Н. Костин. — Москва : Издательский Дом МИСиС, 2018. — 21 с. — ISBN 978-5-906953-22-3. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/98199 (дата обращения: 5.10.2021). — Режим доступа: для авторизир. пользователей.

5 Логанов, С. В. Объектно-ориентированное программирование / С. В. Логанов, С. Л. Моругин. — Саратов, Москва : Профобразование, Ай Пи Ар Медиа, 2022. — 215 с. — ISBN 978-5-4488-1355-9, 978-5-4497-1586-9. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО

PROFобразование : [сайт]. — URL: https://profspo.ru/books/118969 (дата обращения: 12.02.2022). — Режим доступа: для авторизир. пользователей.

6 Макаров, А. В. Common Intermediate Language и системное программирование в Microsoft.NET : учебное пособие / А. В. Макаров, С. Ю. Скоробогатов, А. М. Чеповский. — 3-е изд. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 397 с. — ISBN 978-5-4497-0293-7. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/89403 (дата обращения: 7.03.2022).

7 Маклаков С.В., BPWin и ERWin CASE - средства разработки информационных систем / Маклаков С.В. — М: ДИАЛОГ МИФИ, 2001. — 256с.;

8 Маляров, А. Н. Объектно-ориентированное программирование : учебник для технических вузов / А. Н. Маляров. — Самара : Самарский государственный технический университет, ЭБС АСВ, 2017. — 332 с. — ISBN 978-5-7964-1952-6. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/91772 (дата обращения: 14.06.2022). — Режим доступа: для авторизир. пользователей.

9 Моренкова, О. И. Операционные системы. Linux / О. И. Моренкова. — Саратов : Профобразование, 2021. — 104 c. — ISBN 978-5-4488-1173-9. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/106624 (дата обращения: 01.06.2022). — Режим доступа: для авторизир. пользователей.

10 Троелсен, Э., Джепикс Ф. Язык программирования C# 7 и платформы .NET и .NET Core / Э. Троелсен ; [пер. с англ. Ю.Н. Артеменко] . – 8-е изд. – Вильямс, 2018. – 1328 с.

11 Тузовский, А. Ф. Объектно-ориентированное программирование : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2022. — 206 с. — (Высшее образование). — ISBN 978-5-534-00849-4.

— Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/490369 (дата обращения: 15.03.2022).

12 Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Рихтер Дж. ; [пер. с англ. Е. Матвеев]. - 4-е изд. - Санкт-Петербург : Питер, 2013. - 895 с. : ил.

13 Фомин, Д. В. Информационная безопасность : учебное пособие для СПО / Д. В. Фомин. — Саратов, Москва : Профобразование, Ай Пи Ар Медиа, 2022. — 218 с. — ISBN 978-5-4488-1351-1, 978-5-4497-1565-4. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/118458 (дата обращения: 3.01.2022). — Режим доступа: для авторизир. пользователей. - DOI: https://doi.org/10.23682/118458

14 AvaloniaUI Docs, Avalonia documentation [Электронный ресурс] // URL : https://docs.avaloniaui.net/

15 Habr: Avalonia Tutorial: Реализация MVVM по шагам с примерами // URL : https://habr.com/ru/post/505036/

16 Habr: C#: Внутреннее строение инициализаторов массивов // URL : https://habr.com/ru/post/247047/

17 Habr: MVVM: полное понимание [Электронный ресурс] // URL : https://habr.com/ru/post/338518/

18 Habr: Авалония для самых маленьких [Электронный ресурс] // URL : https://habr.com/ru/company/skbkontur/blog/524518/

19 Habr: Навигация в кроссплатформенном приложении на .NET Core с сохранением состояния на диск на примере ReactiveUI и Avalonia // URL : https://habr.com/ru/post/457164/

20 Habr: Оптимизация .NET приложений: большой результат маленьких правок // URL : https://habr.com/ru/company/pvs-studio/blog/572310/

21 Habr: От WPF к Авалонии // URL : https://habr.com/ru/company/skbkontur/blog/542532/

22 Michael Sikorski, Andrew Honig, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software / Michael Sikorski. – New York: Apress, 2012. – 800 p.

23 Microsoft Docs, Microsoft technical documentation [Электронный ресурс] // URL : https://docs.microsoft.com/

24 METANIT.COM, Сайт о программировании [Электронный ресурс] // URL : https://metanit.com/

25 Serge Lidin, .NET 2.0 IL Assembler / Serge Lidin. – New York: Apress, 2006. – 530 p.

ПРИЛОЖЕНИЕ А

Техническое задание

УТВЕРЖДАЮ
Исполнительный директор
ООО «СТОЖАРЫ»
_____Ю.А. Кравчина
«____ __»_____2022 г.

Техническое задание

1 Введение

1.1 Наименование программы

Полное название: «Обфускатор языков программирования семейства .NET»

Краткое название: «Обфускатор»

1.2 Назначение и область применения

Программа предназначена для оказания противодействия злоумышленнику в попытках получения доступа к бизнес-логике защищаемой программы, транслируемой в IL-код, или внесению каких-либо изменений в код защищаемой программы.

Программа предполагает использование на защищенном от несанкционированного доступа рабочем месте.

2 Требования к программе

2.1 Требования к функциональным характеристикам

Программа должна обеспечивать выполнение следующих функций:

- поиск идентификаторов различных типов данных;
- генерация случайных идентификаторов;
- обработка текстовых файлов кода, написанного на языке IL и получаемого любым доступным декомпилятором.

2.2 Требования к надежности

2.2.1 Требования к обеспечению надежного функционирования программы

Надежное функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий:

а) использованием лицензионного программного и технического обеспечения;

б) организацией бесперебойного питания технических средств;

в) регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;

г) регулярным выполнением требований ГОСТ 51188-98. «Защита информации. Испытания программных средств на наличие компьютерных вирусов».

2.2.2 Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, нее должно превышать 15-ти минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

2.2.3. Отказы из-за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой.

Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

3 Условия эксплуатации

3.1 Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должна обеспечиваться исправная работа программы, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

3.2 Требования к численности и квалификации персонала

Необходимое и достаточное количество сотрудников для работы с программой – 1 человек – оператор ЭВМ.

Требования к квалификации персонала не предъявляются.

3.3 Требования к составу и параметрам технических средств

В состав технических средств должен входить персональный компьютер (ПЭВМ), включающий в себя:

– операционная система: Windows, Linux;

– память ОЗУ: не менее 128 МБ;

– память ПЗУ: не менее 50 МБ;

– периферийные устройства: клавиатура, монитор, мышь.

3.4 Требования к информационной и программной совместимости

3.4.1 Требования к языкам программирования

Программа должна быть выполнена на языке C#

3.4.2 Требования к защите информации и программ

Программа должна быть обфусцирована. Программа не должна сохранять данные защищаемых программ после работы.

3.5 Специальные требования

Специальные требования не предъявляются

4 Требования к программной документации

Состав программной документации должен включать в себя:

а) техническое задание;

б) программу;

в) руководство пользователя.

5 Технико-экономические показатели

5.1. Экономические преимущества разработки

Ориентировочная экономическая эффективность не рассчитываются.

Аналогия не проводится ввиду уникальности предъявляемых требований к разработке.

6 Стадии и этапы разработки

6.1 Стадии разработки

Разработка должна быть проведена в три стадии:

а) разработка технического задания;

б) рабочее проектирование;

в) внедрение.

6.2. Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

а) разработка программы;

б) разработка программной документации;

в) испытания программы.

На стадии внедрения должен быть выполнен этап разработки подготовка и передача программы

6.3. Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

а) постановка задачи;

б) определение и уточнение требований к техническим средствам;

в) определение требований к программе;

г) определение стадий, этапов и сроков разработки программы и документации на неё;

д) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями к составу документации.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

а) разработка, согласование и утверждение методики испытаний;

б) проведение приемо-сдаточных испытаний;

в) корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах Заказчика.

7. Порядок контроля и приемки

7.1. Виды испытаний

Приемо-сдаточные испытания должны проводиться на объекте Заказчика в оговоренные сроки.

Приемо-сдаточные испытания программы должны проводиться согласно разработанной Исполнителем и согласованной Заказчиком Программы и методик испытаний.

Ход проведения приемо-сдаточных испытаний Заказчик и Исполнитель документируют в Протоколе проведения испытаний

7.2. Общие требования к приемке работы

На основании Протокола проведения испытаний Исполнитель совместно с Заказчиком подписывает Акт приемки-сдачи программы в эксплуатацию.

Подпись Исполнителя_____