VB Decompiler

Руководство пользователя

Русский

Visual Basic 5.0/6.0 · .NET

DotFix Software

Содержание

1 Описание 4

2 Быстрый старт 6

3 Главное окно 7

4 Просмотр кода 9

5 Улучшение кода с помощью ИИ 12

6 Открытие и сохранение 16

6.1 Открытие файла 16

6.2 Работа с базой данных 18

6.3 Сохранение списка процедур 19

7 Функции программы 20

7.1 Переход по адресу 20

7.2 Поиск строк 21

7.3 Ссылки на строки 22

7.4 Декомпиляция с заданного адреса 24

7.5 Редактирование данных 25

7.6 Обфускация 27

8 Параметры программы 28

8.1 Параметры программы 28

8.2 Формы 29

8.3 Декомпиляция 30

8.4 Дизассемблер 32

8.5 Цветовая схема 33

8.6 Расширенные возможности 35

8.7 Выбор языка 39

9 Плагины 40

10 Визуальное восстановление форм 42

11 Трассировка 46

12 Аналитические возможности 48

13 Ограничения обфускации 50

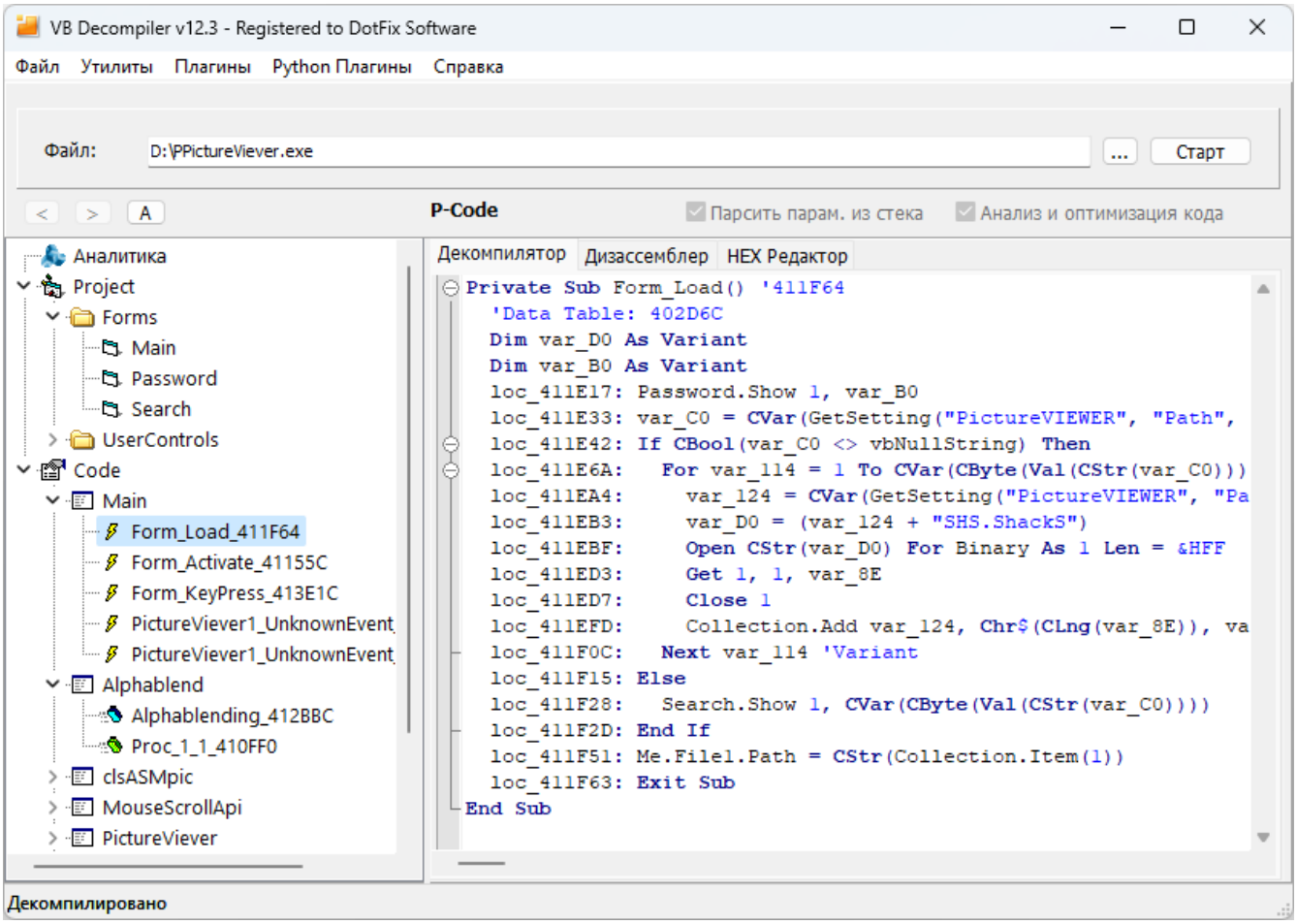
14 Техническая поддержка 52

15 Регистрация 52

16 О разработчике 53

1 Описание

VB Decompiler — это инструмент для реверс-инжиниринга и восстановления исходного кода программ Visual Basic 5.0/6.0 и .NET-сборок. Он объединяет высокоуровневую декомпиляцию, низкоуровневое дизассемблирование, анализ метаданных, встроенный HEX-редактор и восстановление визуального интерфейса. Программа применяется для таких задач, как поддержка устаревшего программного обеспечения, реагирование на инциденты, анализ вредоносного ПО и исследование недокументированного поведения приложений.



Поддерживаемые виды кода

- **VB5/6 P-Code:** восстанавливает операторы Visual Basic на основе инструкций P-Code и операций среды выполнения.
- **VB5/6 Native Code:** дизассемблирует инструкции x86 и использует внутренний эмулятор, сигнатуры функций среды выполнения и эвристики для восстановления кода, похожего на Visual Basic.
- **.NET:** дизассемблирует Common Intermediate Language (CIL, также называемый MS IL) и декомпилирует поддерживаемые шаблоны IL в код, похожий на C#, независимо от исходного языка .NET. Также может быть дизассемблирован нативный код x86 и x64 в .NET-сборках.

Движок анализа .NET не требует среды выполнения .NET или Reflection для анализа кода или визуального восстановления форм.

Инструменты анализа

Используйте поиск процедур и текста, перекрестные ссылки и переименование переменных, а также «Все строки программы» (String references) для исследования поведения. Визуальный просмотрщик восстанавливает классические формы VB, .NET Windows Forms и окна WPF, страницы и пользовательские элементы управления, хранящиеся в скомпилированных ресурсах BAML.

Функции трассировки, доступные по соответствующей лицензии, обеспечивают пошаговую эмуляцию Native Code VB5/6 и P-Code. Аналитические функции, доступные по соответствующей лицензии, формируют отчёты с группировкой операций по категориям для приложений VB5/6 и .NET. «ИИ Помощник» (AI Helper) может обрабатывать восстановленный код VB5/6 или C# через Ollama или OpenAI API.

Интерпретация восстановленного кода

Компиляция удаляет информацию, такую как комментарии, многие имена локальных переменных и некоторые метаданные типов и процедур. Оптимизация и обфускация могут дополнительно изменить структуру кода. Восстановленный листинг предназначен для анализа и восстановления кода. Подтверждайте важные выводы, сверяясь с дизассемблированным кодом, метаданными, ссылками на строки и кодом связанных процедур.

2 Быстрый старт

1. Выберите **«Файл > Открыть программу» (File > Open program)** и откройте файл EXE, DLL или OCX для анализа. Обработка начнется автоматически. Дождитесь заполнения дерева проекта и завершения обработки.
2. Откройте процедуру в **«Обозреватель решений» (Solution explorer)**. Сравните представления **«Декомпилятор» (Decompiler)** и **«Дизассемблер» (Disassembler)**; используйте **«HEX Редактор» (HEX Editor)** для проверки соответствующих байтов.
3. Нажмите **Ctrl+T**, чтобы найти процедуру по её имени или части имени. Используйте **Ctrl+F** для поиска отображаемого текста, или **«Утилиты > Все строки программы» (Tools > String references)** для исследования строк и процедур, использующих их.
4. Откройте восстановленную форму и выберите **«Дизайнер (только чтение)» (Designer (read only))** для визуального предпросмотра. Для классического VB, Windows Forms и WPF используйте восстановленные события для навигации к соответствующему обработчику события.
5. Сохраните анализ через **«Файл > База Данных» (File > Data Base)**, когда необходимо возобновить его позже или поделиться результатами с другим аналитиком.

Выбор режима декомпиляции

Быстрая декомпиляция обрабатывает процедуры при их открытии. Включите **«Анализ прототипов» (Analyze prototypes)** для дополнительного предварительного анализа сигнатур процедур, объявленных как Private. Для поиска всего декомпилированного текста или экспорта полного проекта отключите быструю декомпиляцию и обработайте файл повторно.

Сохранение восстановленного исходного кода

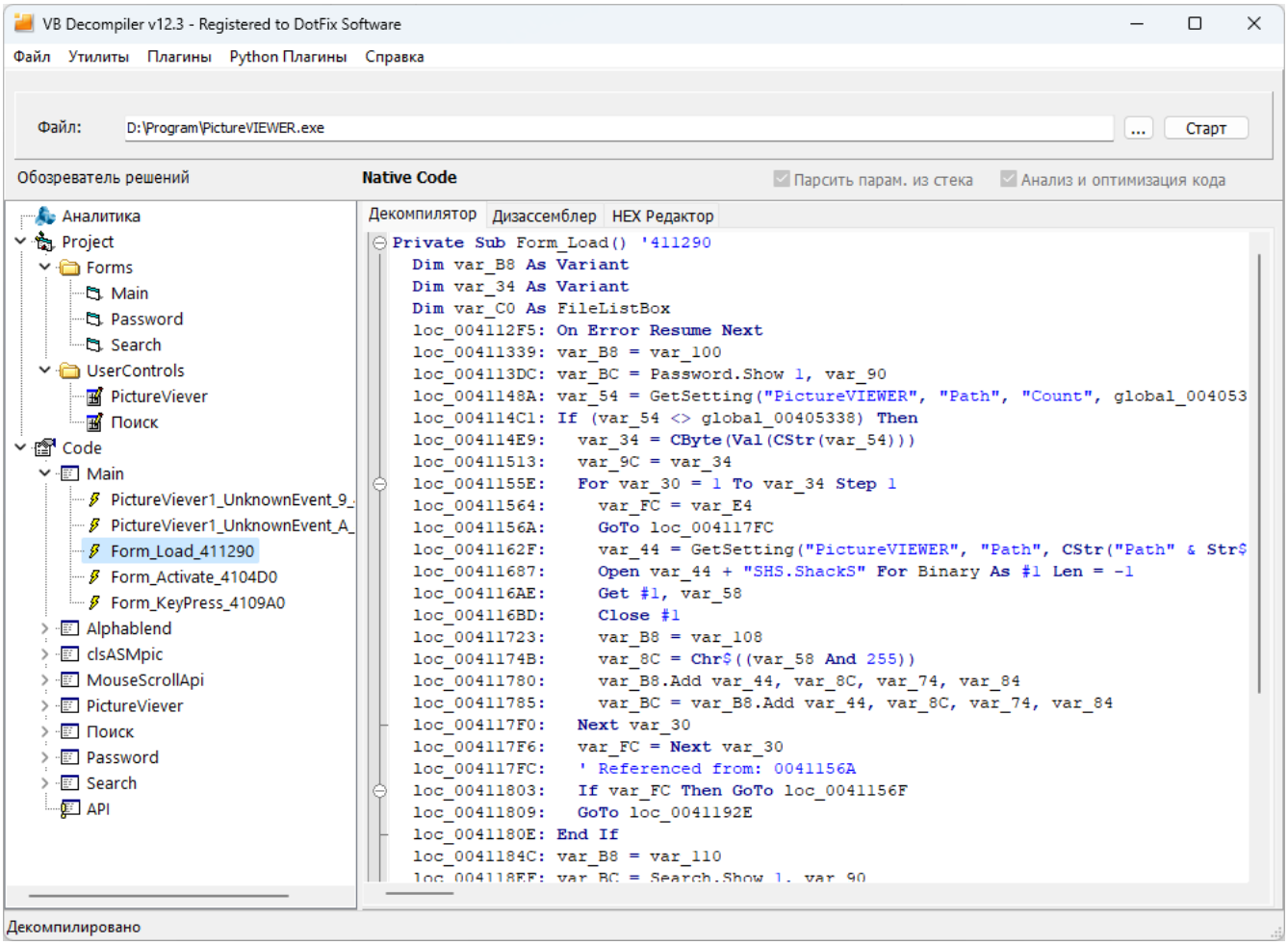
Для экспорта результатов декомпиляции VB5/6 выберите **«Файл > Сохранить декомпилированный проект» (File > Save decompiled project)** и укажите папку для сохранения. Экспорт включает восстановленные формы и модули; файлы FRX содержат связанные изображения и другие бинарные данные форм.

Перед открытием экспортированного кода в IDE вручную исправьте все неточности декомпиляции и несовместимости в восстановленном исходном коде.

См. Визуальное восстановление форм, Просмотр кода и Улучшение кода с помощью ИИ с подробным описанием этих операций.

3 Главное окно

«Обозреватель решений» (Solution explorer) отображает структуру восстановленного проекта: формы, модули, классы и процедуры. Область просмотра рядом с деревом отображает выбранный код, форму или метаданные. Опция **«Стиль окна как в Visual Studio» (Window style as in Visual Studio)** размещает дерево справа, а код — слева.



Представления кода и данных

- **«Декомпилятор» (Decompiler):** высокоуровневый восстановленный код.
- **«Дизассемблер» (Disassembler):** соответствующие инструкции x86/x64, P-Code VB или .NET IL с адресами.
- **«HEX Редактор» (HEX Editor):** исходные байты файла с навигацией по виртуальному адресу или смещению в файле.
- **«Дизайнер (только чтение)» (Designer (read only)):** восстановленная форма и её свойства, где это поддерживается.
- **«ИИ Помощник» (AI Helper):** дополнительная обработка с помощью ИИ для текущей декомпилированной процедуры.

Доступные вкладки зависят от выбранного объекта. Индикатор типа кода различает **P-Code**, **Native Code** и **.NET Assembly**.

Парсить парам. из стека (Parse stack parameters)

Для P-Code этот элемент управления объединяет операции со стеком и вызовы в выражения с аргументами. Отключите этот параметр, чтобы увидеть промежуточные операции. Это полезно, если сигнатура вызываемой процедуры неизвестна и аргументы со стека определяются неверно. Для Native Code этот параметр включает использование эмулятора для восстановления выражений более высокого уровня; при его отключении отображается низкоуровневое представление на ассемблере. Для .NET обработка стека способствует восстановлению IL в виде выражений, похожих на C#.

Анализ и оптимизация кода (Procedure analyzer and optimizer)

Этот элемент управления упрощает восстановленный листинг за счёт объединения выражений, восстановления распознанных управляющих конструкций и удаления распознанного служебного кода компилятора. Он применяется к коду VB5/6 и .NET. Оптимизатор .NET может восстановить поддерживаемые шаблоны `while`, `do...while` и `foreach`, вложенные условия, выражения операторов и шаблоны обработчиков интерполированных строк. Эти преобразования выполняются самим декомпилятором и не требуют помощника ИИ.

Если выражение или ветвь непонятны, сравните вывод с отключенной оптимизацией и изучите дизассемблированный код. В режиме **«Быстрая декомпиляция» (Fast decompilation)** без кэширования эти элементы управления можно изменять при просмотре процедуры. При полной декомпиляции или использовании кэшированных результатов обработайте файл заново после изменения настроек.

Навигация и доступ с клавиатуры

Кнопки со стрелками над деревом позволяют перемещаться по ранее просмотренным процедурам; **Esc** возвращает к предыдущей процедуре. Кнопки Back и Forward мыши используют ту же историю, включая запомненную позицию внутри процедуры.

Нажмите **Alt+1**, чтобы переключить фокус между «Обозреватель решений» и активной областью просмотра. Нажмите **Enter** в дереве, чтобы открыть выбранный объект. В «HEX Редактор» клавиша **Tab** переводит фокус между полями шестнадцатеричного и текстового представления, а затем к следующему элементу управления. Подписи элементов интерфейса связаны со своими полями ввода для навигации с помощью программы чтения с экрана.

См. Просмотр кода для контекстно-зависимых горячих клавиш, перекрёстных ссылок и переименования, а также Аналитические возможности для представления «Аналитика» (Analytics).

4 Просмотр кода

Представления кода объединяют подсветку синтаксиса, сворачиваемые блоки, навигацию по адресам и перекрестные ссылки. Используйте высокоуровневый листинг для понимания процедуры; проверяйте полученный результат по дизассемблированному коду и байтам файла.

Подсветка и навигация по идентификаторам

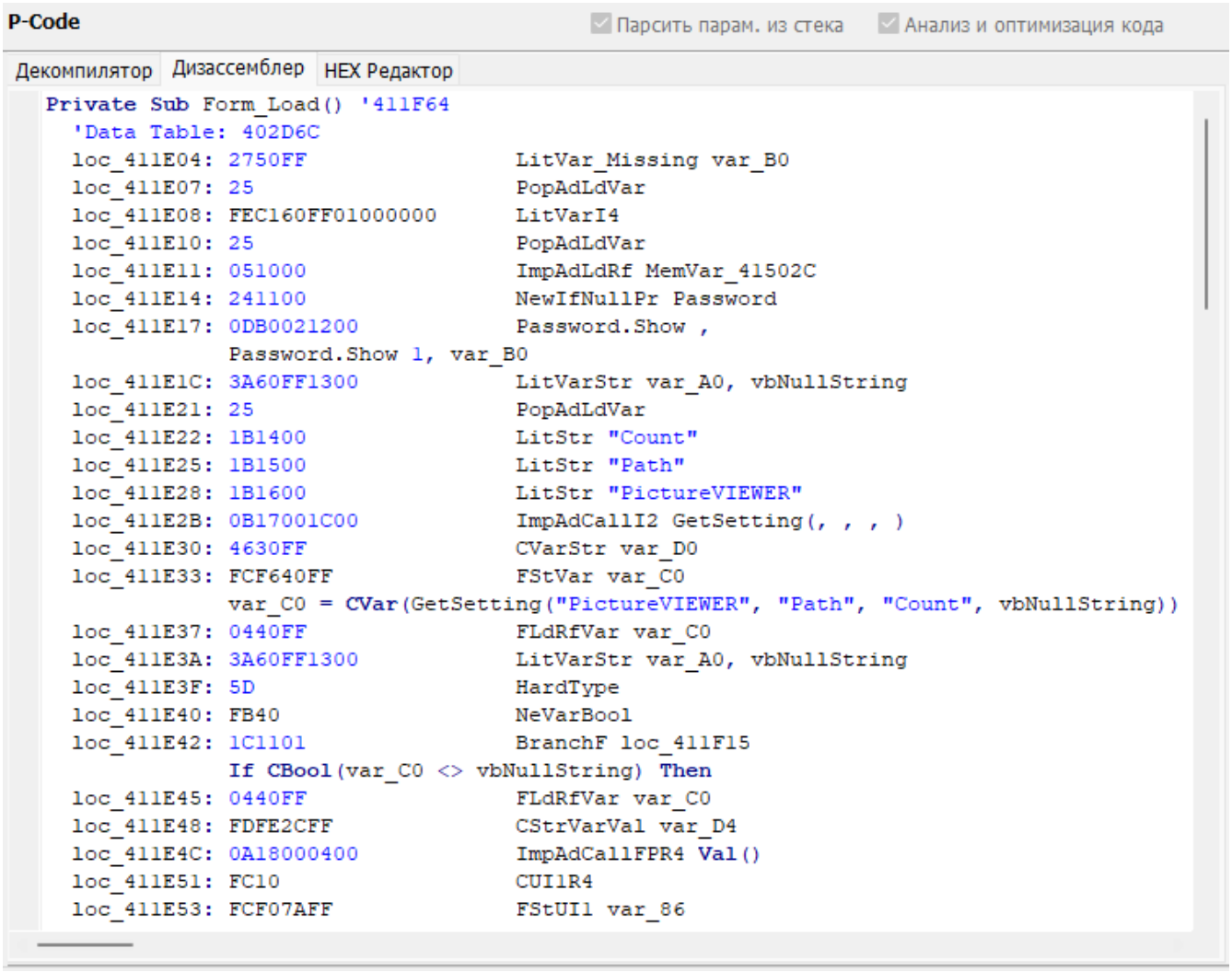
Поставьте курсор на регистр, переменную или имя функции, чтобы выделить соответствующие вхождения в текущем листинге. Локальные переменные, такие как `var_XX` и `str_XX`, также выделяются при использовании в качестве объекта в выражении обращения к члену, например `str_1.Value`. Выделение идентификатора `global_XX` подсвечивает его вхождения в текущей процедуре. Для поиска ссылок по всему проекту используется отдельная команда.

Дважды щелкните по распознанному вызову процедуры, чтобы открыть вызванную процедуру. Константы, похожие на адреса, могут вести к соответствующим байтам в HEX-редакторе. Используйте **Esc**, стрелки навигации или кнопки Назад/Вперед мыши для перемещения по истории просмотра.

Объединение декомпилированного и дизассемблированного кода

Щелкните правой кнопкой мыши листинг декомпилированного кода и выберите **«Скопировать в дизассемблер» (Copy to disassembler)**, или нажмите **D** в представлении кода. Это объединяет восстановленные операторы с низкоуровневым листингом для текущей процедуры. Чтобы сохранить текущую позицию просмотра, поставьте курсор на соответствующий адрес `loc_` перед вызовом команды.

Оставьте включенным **«Добавлять адреса» (Add Addresses)** в настройках декомпиляции при использовании этого рабочего процесса. Дизассемблер всегда отображает адреса; «Добавлять адреса» управляет видом «Декомпилятор» (Decompiler).



Переименование переменных

Для VB5/6 P-Code и Native Code щелкните правой кнопкой мыши переменную и выберите **«Переименовать переменную» (Rename variable)**, или поставьте курсор на переменную и нажмите **N**. Введите имя, которое отражает установленное при анализе назначение переменной. Декомпилятор проверяет конфликты имен и зарезервированные ключевые слова. Чтобы отменить переименование, используйте **«Отменить переименование» (Undo renaming)** или **Ctrl+Shift+Z**. Эта команда независима от обычной отмены редактирования текста.

Переименование облегчает анализ восстановленного листинга; оно не восстанавливает имя, которое было удалено компиляцией. Рассматривайте имена, назначенные аналитиком, как аннотации при описании результатов анализа.

Поиск вызывающих процедур и ссылок на глобальные переменные

Чтобы найти вызывающие процедуры, щелкните правой кнопкой мыши процедуру в «Обозреватель решений» (Solution explorer) и выберите **«Найти ссылки» (Find references)**, или нажмите **X** после выбора этой процедуры в дереве. В результатах перечислены процедуры, содержащие распознанные вызовы выбранной процедуры. Дважды щелкните по результату, чтобы проверить вызывающую процедуру. Поиск поддерживается для VB5/6 и .NET; поддержка .NET была добавлена в версии 12.4.

Чтобы исследовать глобальную переменную, поставьте курсор на её идентификатор в представлении кода и нажмите **X**. Таким образом, эта же клавиша выполняет поиск ссылок на процедуры в дереве и на глобальные переменные в коде. Отсутствие ссылки в списке не исключает косвенных, динамически определяемых или скрытых обфускацией обращений.

Контекстно-зависимые сочетания клавиш

Фокус	Сочетание клавиш	Действие
Код	N	Переименовать переменную под курсором (VB5/6).
Код	X	Найти ссылки на глобальную переменную под курсором.
Код	D	Скопировать текущий декомпилированный код в представление «Дизассемблер» (Disassembler).
Код	F4	Выполнить трассировку до адреса в текущей строке; требует функций трассировки.
Код	Ctrl+Shift+Z	Отменить переименование переменной под курсором.
Дерево	X	Найти ссылки на выбранную процедуру.
Дерево	G	Открыть диалог перехода к разделу.
Дерево	Ctrl+C	Скопировать полный путь к выбранному объекту в дереве.
Дерево	Space or Insert	Отметить или снять отметку с выбранного элемента.

Однобуквенные сочетания клавиш зависят от представления, в котором находится фокус, и не заменяют обычный ввод в полях ввода. Используйте **Alt+1** для изменения фокуса. См. Поиск строк для Ctrl+T и Ctrl+F, Переход к адресу для путей дерева и адресов, и Трассировка для ограничений эмулятора.

5 Улучшение кода с помощью ИИ

«ИИ Помощник» (AI Helper) обрабатывает код, уже восстановленный VB Decompiler: C# из сборок .NET и Visual Basic 5.0/6.0 из P-Code или Native Code. Для обработки можно использовать локальную модель через Ollama или сервис, подключённый через интеграцию с OpenAI API.

ИИ может предложить более понятные управляющие конструкции, имена переменных, объявления и выражения. Он не может достоверно восстановить информацию, отсутствующую в двоичном файле, или гарантированно исправить фрагменты, неверно интерпретированные декомпилятором. Перед использованием результата при восстановлении исходного кода или в отчёте о расследовании сравните его с исходным листингом декомпилятора, дизассемблированными инструкциями и кодом связанных процедур.

Обработка процедуры

1. Откройте нужную процедуру в «Обозреватель решений» (Solution explorer) и изучите её код в представлении **«Декомпилятор» (Decompiler)**.
2. Переключитесь на **«ИИ Помощник»** и настройте поставщика сервиса, адрес сервера и модель, как описано ниже.
3. Запустите обработку кнопкой рядом со списком моделей.
4. Проверьте полученный код в «ИИ Помощник» и сравните его с исходным результатом декомпиляции.

Обработка с помощью ИИ выполняется в отдельном потоке. Если модель слишком долго отвечает, нажмите **«Отмена» (Cancel)**. Дополнительные параметры подключения можно свернуть, чтобы освободить место для кода.

Локальная модель через Ollama

1. Установите Ollama с официальной страницы загрузки и запустите её.
2. Загрузите рекомендуемую модель командой `ollama pull qwen3:8b` в терминале.
3. В «ИИ Помощник» выберите локального поставщика LLM/Ollama. В поле **«Ollama хост» (Ollama host)** укажите адрес сервера. Для локальной установки по умолчанию используется `http://localhost:11434`.
4. Выберите установленную модель в поле **«Модель» (Model)** и запустите обработку.

Рекомендуется `qwen3:8b`: встроенные промпты «ИИ Помощник» адаптированы для этой легковесной модели. Более новые модели могут работать медленнее, не улучшая результат обработки кода.

Выбор модели не ограничен: можно использовать любую модель, установленную в выбранном экземпляре Ollama. Если формат её ответа неизвестен декомпилятору, в результате могут остаться служебные теги или другой вспомогательный текст.

Оборудование для локального запуска Ollama

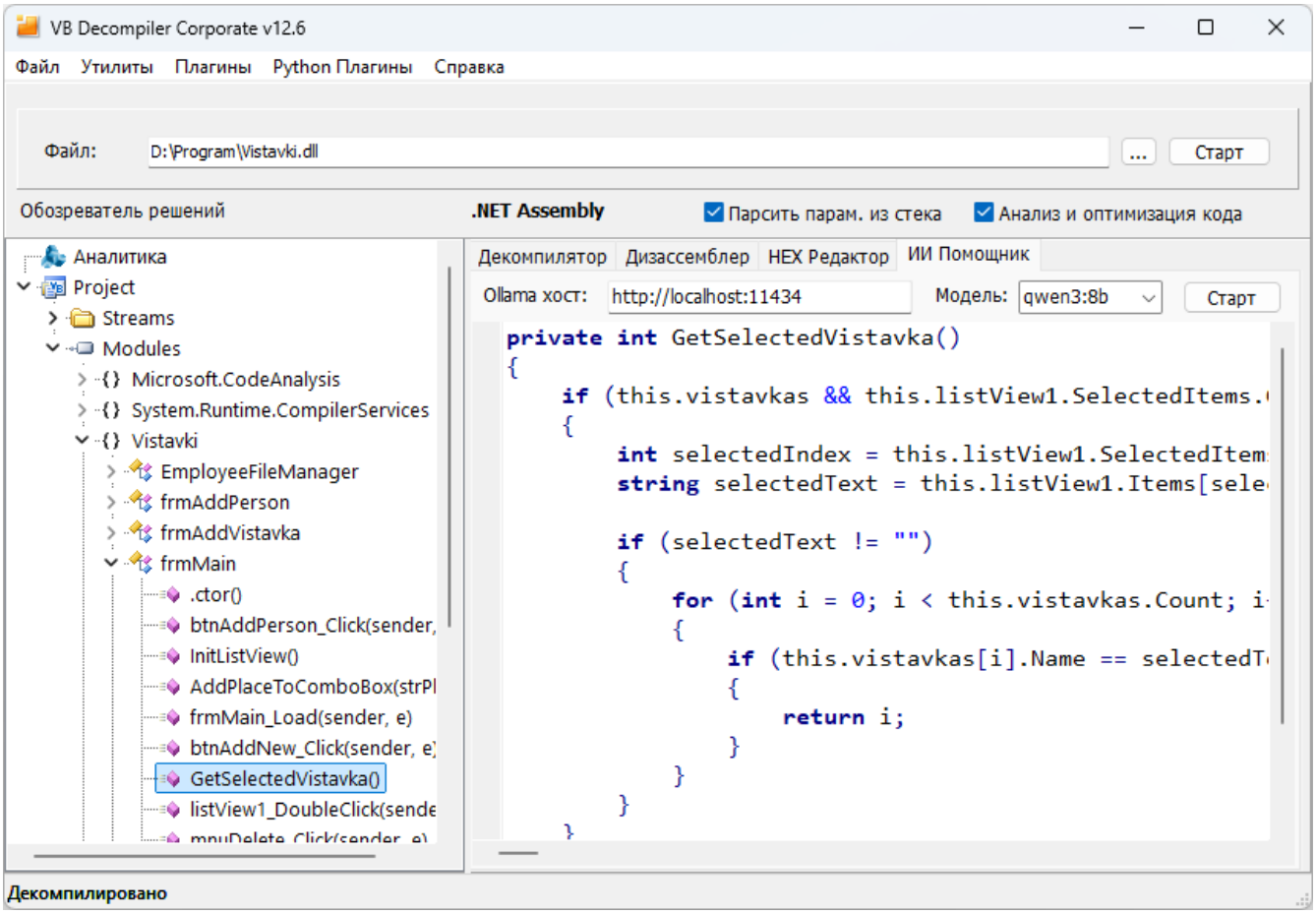
Локальная обработка ИИ выполняется на компьютере, где запущена Ollama, и требует значительно больше ресурсов, чем декомпиляция. Ниже приведены практические ориентиры для `qwen3:8b` с квантованием весов `Q4_K_M`, стандартным кэшем контекста `f16` и обработкой одного запроса за раз. Контекст измеряется в токенах: 4K — это 4096 токенов, а 128K — 131072 токена. В этот объём входят инструкции модели, исходный код и сформированный ответ.

Компонент	Практический минимум: контекст 4K	Рекомендуется: контекст 128K
Операционная система	64-разрядная Windows 10 (22H2) или Windows 11	64-разрядная Windows 10 (22H2) или Windows 11
Процессор	Современный x64-процессор с AVX2 и не менее чем 4 физическими ядрами	Современный x64-процессор с AVX2 и не менее чем 8 физическими ядрами
Оперативная память (RAM)	16 ГБ	64 ГБ
Видеокарта	Совместимая с Ollama видеокарта с выделенной видеопамятью не менее 8 ГБ	Совместимая с Ollama видеокарта с выделенной видеопамятью не менее 32 ГБ
Накопитель	SSD, не менее 20 ГБ свободного места для Ollama и этой модели	SSD, не менее 20 ГБ свободного места для Ollama и этой модели

Выберите размер контекста ползунком в настройках Ollama. Видеокарты с 8 ГБ памяти, подходящей для 4K, недостаточно для приведённой конфигурации с контекстом 128K. Размер загружаемой модели — около 5.2 ГБ, но это не весь необходимый объём памяти. При 128K один только кэш контекста `f16` занимает примерно 18 ГиБ; дополнительно нужны память для весов модели и рабочих буферов. Оставляйте запас памяти для Windows и других приложений.

При достаточном объёме RAM Ollama может выполнять вычисления на процессоре, однако обработка кода может быть очень медленной. Компьютеры на Pentium M и сопоставимые устаревшие системы не подходят для локального запуска `qwen3:8b`. Общая память старых встроенных Intel HD Graphics не заменяет указанную в таблице выделенную видеопамять. Эти требования относятся к компьютеру с Ollama; при подключении к удалённому серверу или внешнему API вычисления выполняются на их оборудовании.

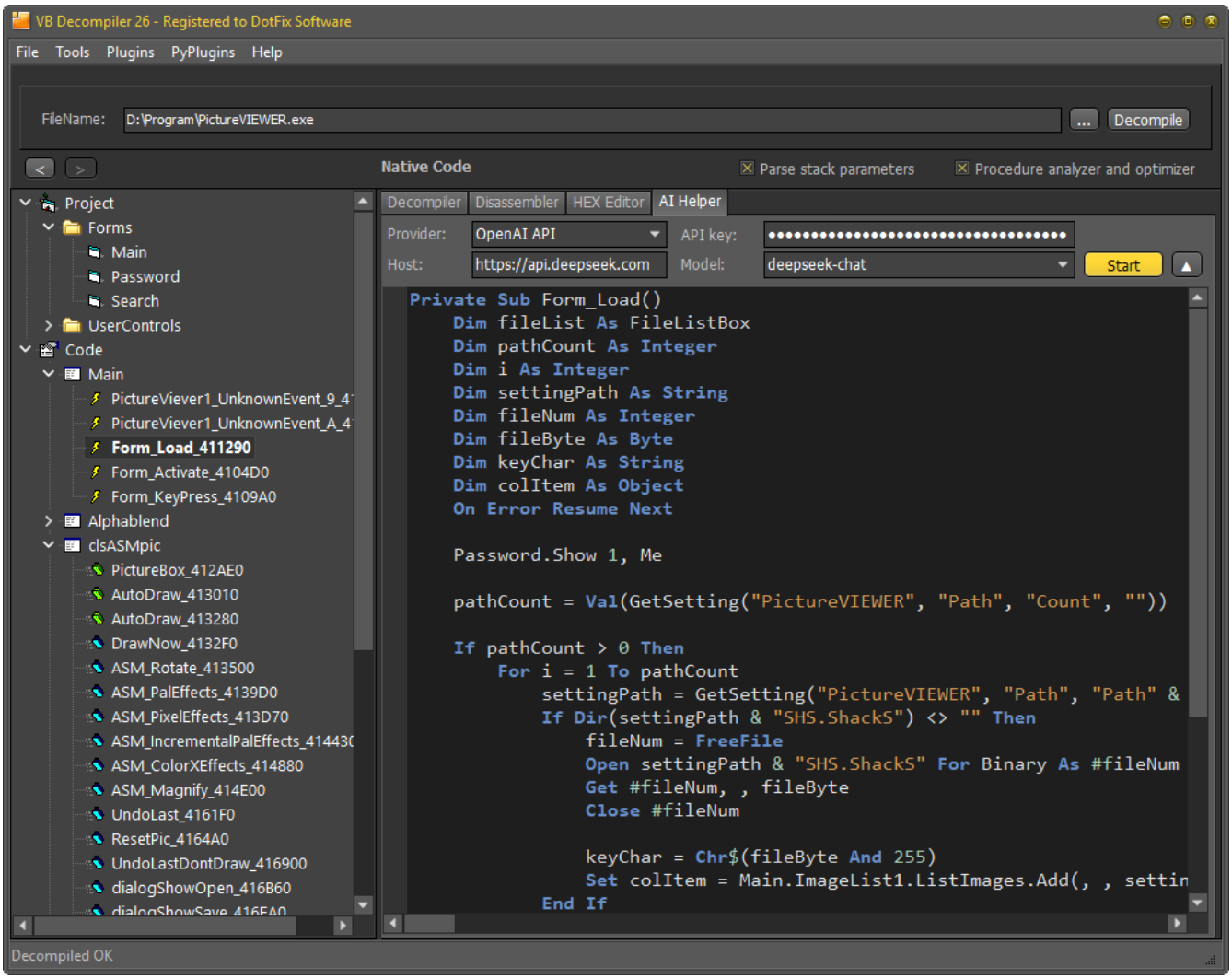
Для небольших процедур начните с 4K. Увеличение контекста повышает расход памяти и время обработки больших входных данных. Если обработка слишком медленная, отмените её, уменьшите контекст и проверьте использование GPU командой `ollama ps`. Фактический расход памяти зависит от версии Ollama, вычислительного бэкенда и других запущенных задач.



Подключение через OpenAI API

В поле **«Провайдер» (Provider)** выберите **OpenAI API**. Заполните поля **«Хост» (Host)**, **«API-ключ» (API key)** и **«Модель»** для нужного сервиса. Укажите действующий ключ API и доступную по этому ключу модель. Не включайте ключи API в скриншоты и материалы для совместного анализа.

Через поставщика **OpenAI API** также можно подключить DeepSeek. Укажите адрес сервера DeepSeek, ключ API и идентификатор модели в соответствующих полях. Пример настройки приведён ниже.



Где обрабатывается код

«ИИ Помощник» отправляет текущий декомпилированный код выбранному сервису. Для полностью локальной обработки используйте сервер Ollama на своей рабочей станции и модель, выполняемую на ней же. При подключении к Ollama на другом компьютере, облачной модели Ollama или внешнему API код может передаваться за пределы рабочей станции.

При восстановлении конфиденциального исходного кода и расследовании инцидентов выбирайте сервер и сервис с учётом требований организации к обработке данных. Наличие Ollama или адреса localhost само по себе не означает, что код обрабатывается без доступа к внешним сервисам.

Устранение неполадок

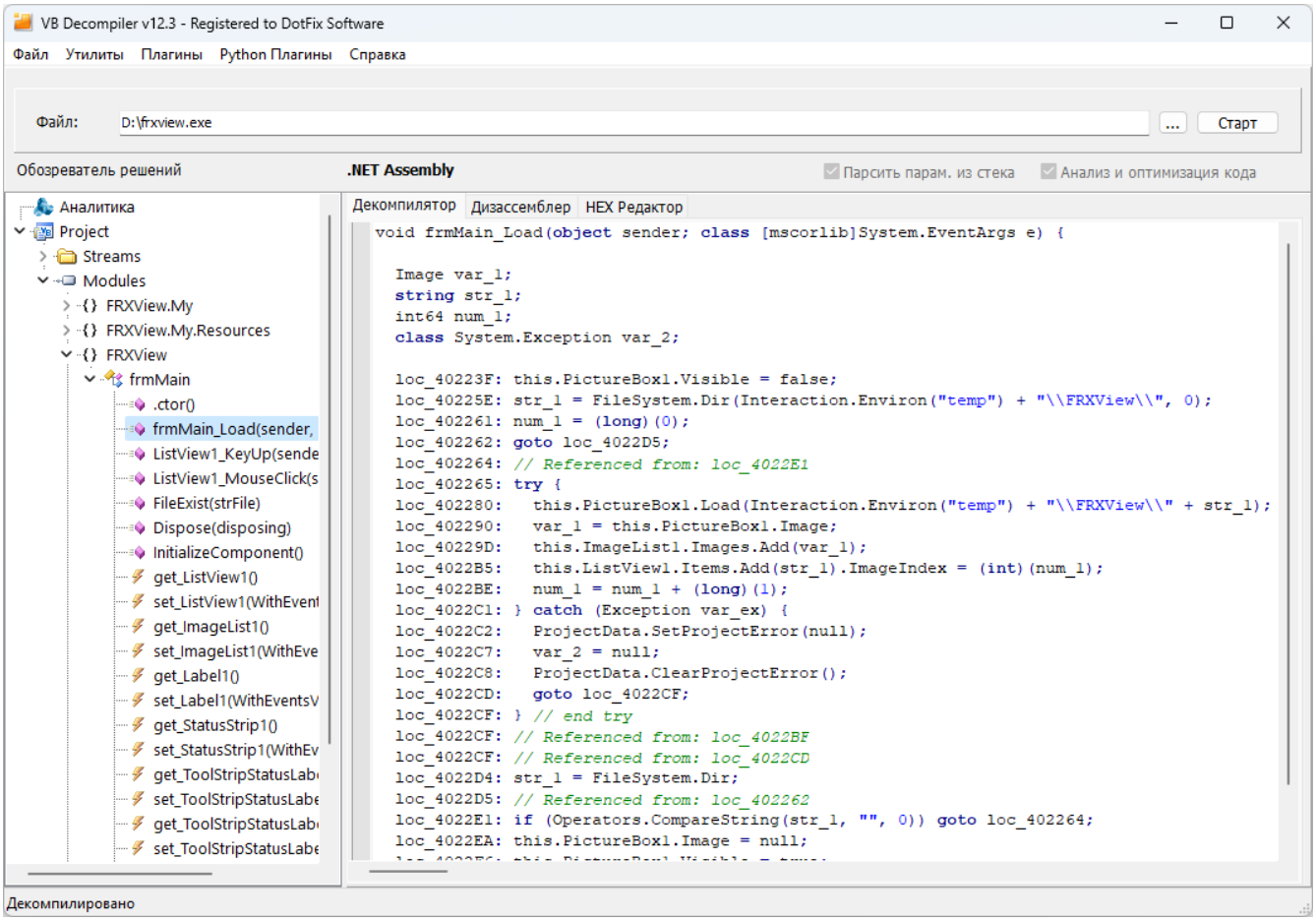
- Если локальная модель не отображается, убедитесь, что она установлена именно в том экземпляре Ollama, адрес которого указан в настройках.
- Если сервис не отвечает, проверьте, что он запущен и доступен. Адрес сервера, учётные данные и идентификатор модели должны соответствовать одному и тому же сервису.
- Если обработка слишком медленная, отмените её. При использовании другой локальной модели попробуйте рекомендуемую `qwen3:8b`.
- Если в ответе незнакомой модели остаются служебные теги или вспомогательный текст, попробуйте `qwen3:8b` или удалите этот текст вручную при проверке результата.
- Если в результате неожиданно изменились условие, вызов, константа или тип, сверьтесь с низкоуровневым листингом. Более читаемый код не обязательно сохраняет исходное поведение.

Ollama и сервисы внешних поставщиков моделей — сторонние продукты. Их доступность, поведение моделей и правила работы не зависят от VB Decompiler.

6 Открытие и сохранение

6.1 Открытие файла

Перед началом процесса декомпиляции необходимо выбрать файл для анализа. Для этого в меню **«Файл» (File)** имеется раздел **«Открыть программу» (Open program)**. **VB Decompiler** поддерживает открытие файлов EXE/OCX/DLL, созданных в версиях **Visual Basic** 5.0 или 6.0, а также приложений, скомпилированных в среде **Visual Studio .NET**. В последнем случае язык исходного кода не имеет значения, поскольку Visual Studio .NET компилирует приложения независимо от исходного языка в единый внутренний формат — **MS IL**.



После открытия файла декомпилятор определяет тип кода и компилятор, которым создана программа. Если открытая программа была разработана в неподдерживаемой среде (например, **Delphi** или **Visual C++**), появится сообщение о том, что такая среда не поддерживается и декомпилировать программу невозможно. В случае появления этого сообщения вы можете обратиться к нам, и мы сможем помочь с декомпиляцией вашей программы. У нас есть внутренние разработки в области декомпиляции приложений **Delphi** и **FreePascal**, и мы можем предложить декомпиляцию таких файлов как отдельную услугу.

Если открытая программа упакована с помощью **упаковщика EXE**, обфускатора или защищена — декомпилятор предложит попробовать распаковать этот файл. Эта функция доступна только в 32-разрядных версиях Windows и предоставляется без каких-либо гарантий. Если распаковка возможна — файл будет запущен, распакован в памяти, сохранён на диск под другим именем и открыт для декомпиляции. **Внимание!** Никогда не пытайтесь распаковывать с помощью встроенного распаковщика вредоносные программы или любой другой код, который может нанести вред вашей системе, поскольку при распаковке открываемая программа будет выполняться, и это может привести к повреждению системы или данных.

Важное примечание: ни в коем случае не следует пытаться декомпилировать или распаковывать программное обеспечение без наличия прав на него, а также если правообладатель запрещает декомпрессию и анализ этого файла по лицензии. Это может нарушать законы вашей страны.

*Visual Studio .NET, Visual C++ и Visual Basic являются зарегистрированными товарными знаками Microsoft Corporation. Delphi является зарегистрированным товарным знаком Embarcadero.

Другие способы открытия файлов

Вы можете перетащить файл из Проводника в окно VB Decompiler. Файлы с неизвестным расширением или без расширения также можно передать таким образом; распознавание всё ещё зависит от фактического содержимого файла. При открытии поддерживаемого ярлыка Windows **.lnk** декомпилятор определяет путь к целевому файлу. Сохранённые базы данных VDD также можно открыть перетаскиванием или из командной строки.

6.2 Работа с базой данных

Эта функция доступна через раздел **«База Данных» (Data Base)** в меню **«Файл» (File)** и предназначена для сохранения результатов декомпиляции в отдельный сжатый файл. Это может быть полезно для более быстрой загрузки результатов декомпиляции в будущем без необходимости повторно выполнять декомпиляцию кода.

Если вам потребуется вернуться к анализу программы или поделиться этими результатами с коллегой, повторная декомпиляция не требуется. В случае полной декомпиляции (с отключённым параметром **«Быстрая декомпиляция» (Fast decompilation)**) этот процесс может занять больше времени и потребовать дополнительных ресурсов. Гораздо удобнее заранее сохранить декомпилированный код в **Database**.

VB Decompiler использует внутренний формат сжатой базы данных, что позволяет значительно оптимизировать размер сохраняемых данных. Кроме того, обеспечивается совместимость с базами данных, сохраненными большинством предыдущих версий VB Decompiler, что позволяет возвращаться к сохранённым результатам анализа после обновления продукта.

Открытие и перенос баз VDD

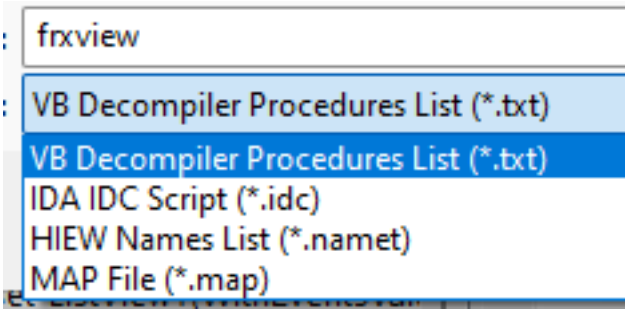
Сохраненные базы анализа используют расширение **.vdd**. Открыть базу VDD можно обычным способом открытия файлов, перетащив её в окно декомпилятора, или передав путь к ней в командной строке.

При поиске исходного анализируемого файла VB Decompiler проверяет как его сохраненный путь, так и папку с базой данных. Для совместной работы сохраняйте соответствующий двоичный файл рядом с файлом VDD, если его исходный путь недоступен. Сохраняйте именно ту версию двоичного файла, которую анализировали, чтобы адреса и байты оставались согласованными.

6.3 Сохранение списка процедур

Данная операция выполняется путем вызова опции **«Сохранить список процедур» (Save procedures list)** в меню **«Файл» (File)** и помогает анализировать программы во внешних инструментах совместно с **VB Decompiler**.

В настоящее время адреса и имена функций и процедур в обработанной программе могут быть сохранены в нескольких форматах. Вы можете выбрать любой из этих вариантов, указав тип файла в стандартном диалоге сохранения.



Сохранение в формате TXT позволяет сохранить список процедур в простом текстовом файле. Это может быть полезно для сравнения процедур между различными версиями программы, выявления изменений адресов и других подобных задач.

Сохранение в формате скрипта IDC позволяет загружать адреса функций и процедур в дизассемблер **IDA Pro**. В случаях, когда анализируемая программа скомпилирована в **Native Code**, эта функция может быть очень полезной.

Сохранение в формате NAMET позволяет упростить анализ с помощью утилиты **HIEW**. Скопируйте полученный файл в папку с анализируемой программой. Затем откройте программу в HIEW и нажмите кнопку F12. Появится окно со списком процедур, который будет пустым. Нажмите кнопку F12 еще раз, если путь к файлу NAMET соответствует созданному файлу — просто нажмите Enter. Адреса процедур будут загружены в HIEW, что позволит перемещаться по методам, используя список, нажимая клавишу F12.

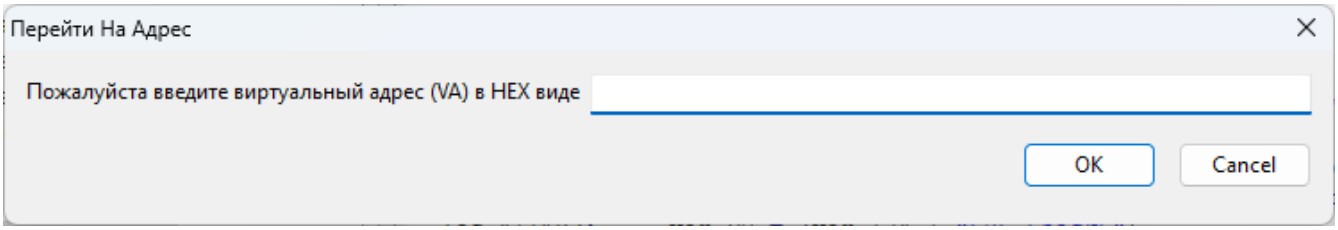
Сохранение в формате MAP генерирует стандартный файл карты адресов, который обычно создается компиляторами. Этот файл может использоваться с множеством отладчиков и дизассемблеров для обозначения начала функций программы. Поскольку программы, написанные на Visual Basic, часто неправильно дизассемблируются сторонним программным обеспечением, файл MAP, сгенерированный в **VB Decompiler**, значительно упростит процесс. Этот формат поддерживается через плагины в различных продуктах, таких как **OllyDbg**, **IDA Pro** и других.

7 Функции программы

7.1 Переход по адресу

Переход к виртуальному адресу

Выберите **«Утилиты > Перейти на виртуальный адрес» (Tools > Jump to virtual address)** или нажмите **Ctrl+G**. Введите виртуальный адрес кода для проверки. Вы также можете выбрать адрес в листинге и нажать Ctrl+G. Декомпилятор открывает код, связанный с этим адресом.



Виртуальный адрес (VA) задаёт расположение в адресном пространстве исполняемого образа и включает его базовый адрес. Смещение в файле задаёт позицию байта в файле на диске. Не заменяйте одно другим: HEX-редактор поддерживает обе формы навигации.

Возврат через историю навигации

Используйте кнопки со стрелками над деревом для перемещения назад и вперед по ранее просмотренным процедурам. **Esc** возвращает к предыдущей процедуре. Кнопки Back и Forward мыши также используют эту историю, сохраняя позицию внутри процедуры.

Переход по пути в дереве проекта

В «Обозреватель решений» (Solution explorer) щелкните правой кнопкой мыши требуемый объект и выберите **«Скопировать имя секции» (Copy the name of section)**, или нажмите **Ctrl+C**, когда фокус находится в дереве. Это копирует полный путь к выбранному узлу.

Чтобы вернуться к этому расположению, выберите **«Перейти используя путь секции» (Jump to the section)** в контекстном меню дерева или нажмите **G** в дереве и вставьте скопированный путь в диалоговое окно. Это полезно при передаче результатов анализа другому аналитику или возвращении к крупному проекту. Используйте сохранённый путь вместе с тем файлом, для которого он получен. В другой сборке проекта этот путь может указывать на другой код.

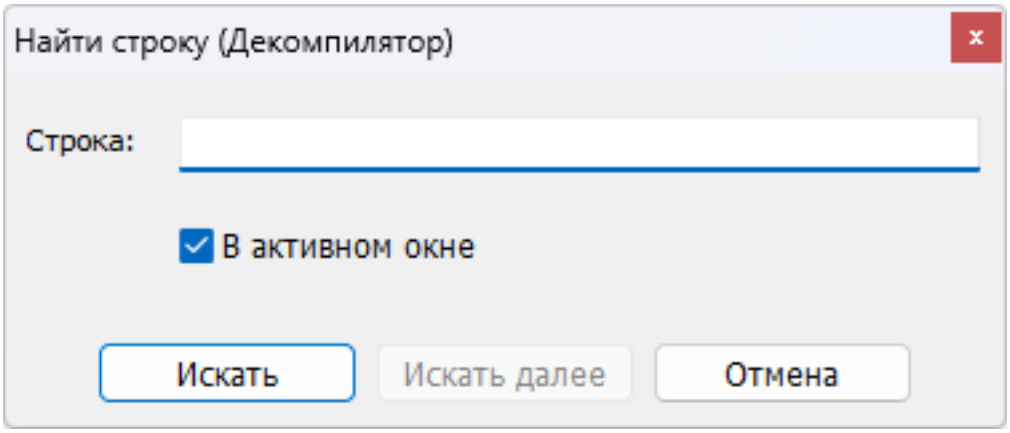
7.2 Поиск строк

Поиск процедуры по имени

Выберите **«Утилиты > Найти процедуру» (Tools > Find procedure)** или нажмите **Ctrl+T**. Введите полное имя процедуры или её отличительную часть. Поиск охватывает дерево проекта, поэтому не нужно знать, какой модуль или класс содержит процедуру. Если совпадает несколько имён, в списке результатов содержатся все соответствующие записи; дважды щёлкните по требуемой записи, чтобы открыть её.

Поиск текста в активном окне или во всём проекте

Выберите **«Утилиты > Find string» (Tools > Find string)** или нажмите **Ctrl+F**, чтобы открыть диалог поиска текста. При выборе **«В активном окне» (In active window)** поиск применяется к текущему отображаемому коду или данным формы. Снимите этот флажок, чтобы выполнить поиск по всему декомпилированному проекту.



Введите текст в **«Строка» (String)** и выберите **«Искать» (Find)**. Используйте **«Искать далее» (Find next)** для продолжения поиска того же текста. **«Отмена» (Cancel)** закрывает диалог.

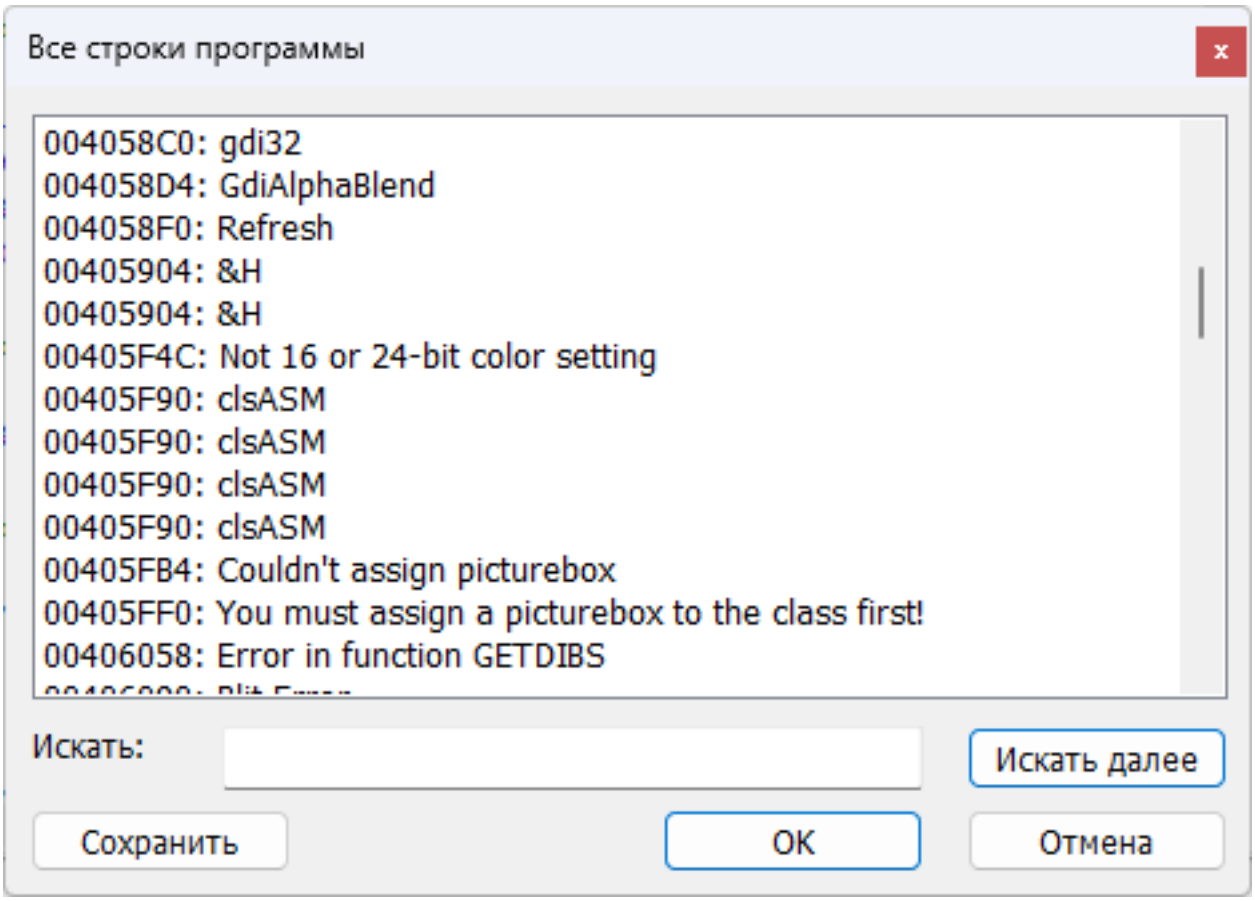
Поиск всего декомпилированного текста требует полной декомпиляции. Отключите **«Быстрая декомпиляция» (Fast decompilation)** в разделе Расширенные возможности и сначала обработайте файл заново. Поиск процедур по имени — это отдельная операция над структурой проекта.

Поиск ссылок на строки

Если вам нужны строковые литералы и код, использующий их, откройте Ссылки на строки. В этом окне поле **«Искать» (Search)** и кнопка **«Искать далее»** выполняют поиск в списке строк. Ссылки на строки можно собрать предварительным дизассемблированием в режиме быстрой декомпиляции; для этого не требуется предварительно декомпилировать все процедуры целиком.

7.3 Ссылки на строки

В коде любой программы вы часто встречаете строковые данные (ключи реестра, имена файлов, сообщения об ошибках и т. д.). При анализе полезно видеть список строк, используемых процедурами и функциями в декомпилированных приложениях. Именно для этого предназначена наша функция **«Все строки программы» (String references)**, доступная через меню **«Утилиты» (Tools)**. Чтобы использовать её, просто перейдите по пути **«Утилиты»** -> **«Все строки программы»**, и если ваша программа содержит какие-либо строковые данные, они будут отображены следующим образом:

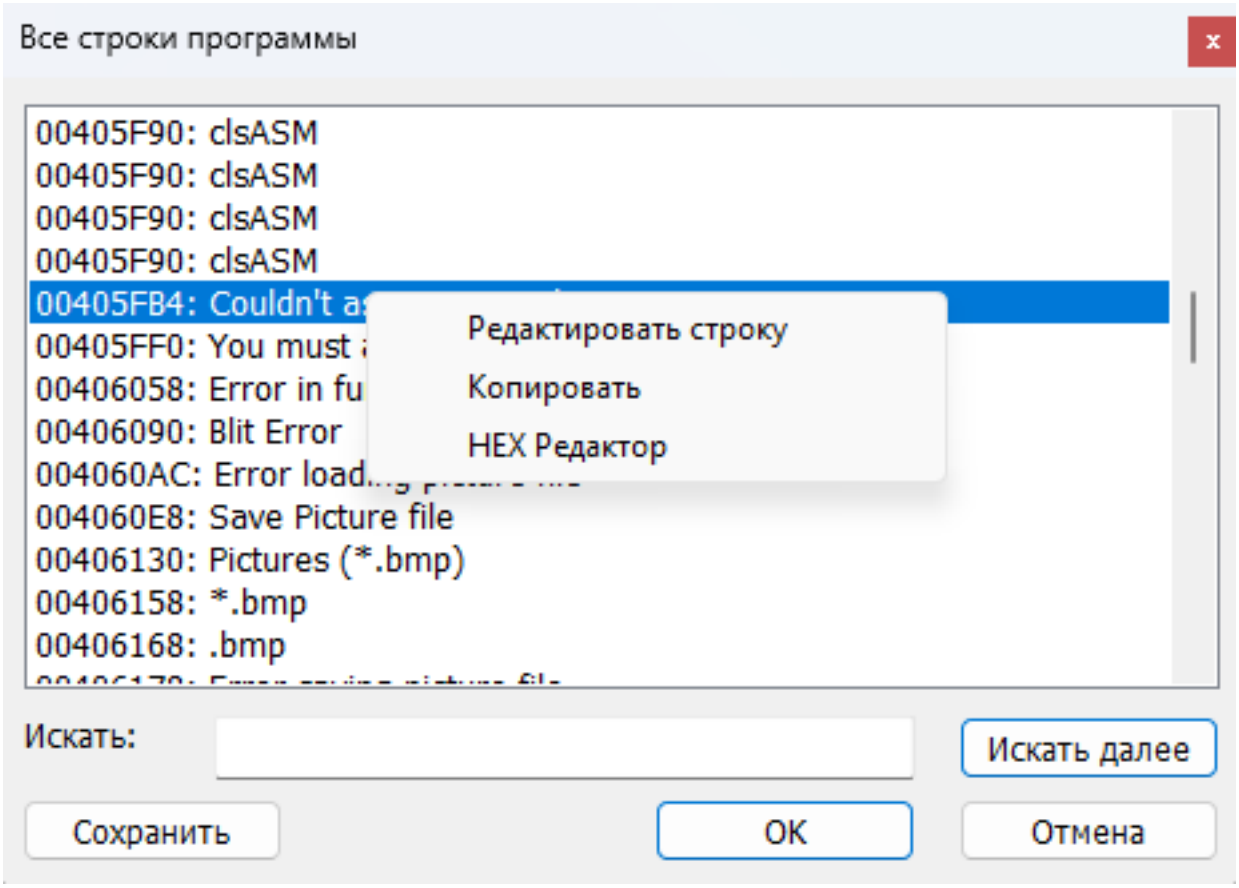


Найдите нужную строку в списке и дважды щёлкните по ней. Декомпилятор найдёт фрагмент кода, использующий эту строку, и откроет соответствующую процедуру. Повторный двойной щелчок по строке продолжит поиск этой строки внутри кода программы. Нажатие кнопки 'OK' найдёт выделенную строку в коде и закроет диалоговое окно выбора строк. Отмена просто закрывает диалоговое окно.

Функция для отображения **«Все строки программы»** доступна только в лицензионной версии программного обеспечения и недоступна для версии **Lite**.

Важное примечание: строки отображаются не только для активного модуля, как в программе 1.x, а для всей программы. Это изменение было сделано для упрощения анализа программ с большим количеством форм и модулей.

VB Decompiler также поддерживает редактирование строк. Чтобы вызвать диалоговое окно редактирования строки, щёлкните правой кнопкой мыши по строке, которую нужно изменить, и выберите **«Редактировать строку» (Edit string)** из появившегося контекстного меню. Появится диалоговое окно для ввода новой строки. После изменения строки нажмите кнопку **Ok**, и строка будет непосредственно изменена в анализируемом файле. Мы рекомендуем создать резервную копию анализируемого файла заранее, поскольку во время этого действия резервная копия не создаётся, а отмена этой операции невозможна.



Важное примечание: при редактировании строк помните, что длина новой строки не должна превышать длину старой. Это связано с форматом двоичных исполняемых файлов, где для вставки более длинной строки не выделяется место.

Вы также можете просматривать и частично редактировать строку в HEX-редакторе. Для этого щёлкните правой кнопкой мыши по строке и выберите «HEX Редактор» (HEX Editor) из появившегося меню. В HEX-редакторе есть функции для отмены и повторения изменений, что упрощает эксперименты при модификации строковых данных приложений.

Окно со списком строк также имеет текстовое поле **«Искать» (Search)** и кнопку **«Искать далее» (Find next)**. Это полезно для поиска строк внутри списка, особенно при работе с большими проектами, где ручной визуальный поиск по списку крайне затруднителен.

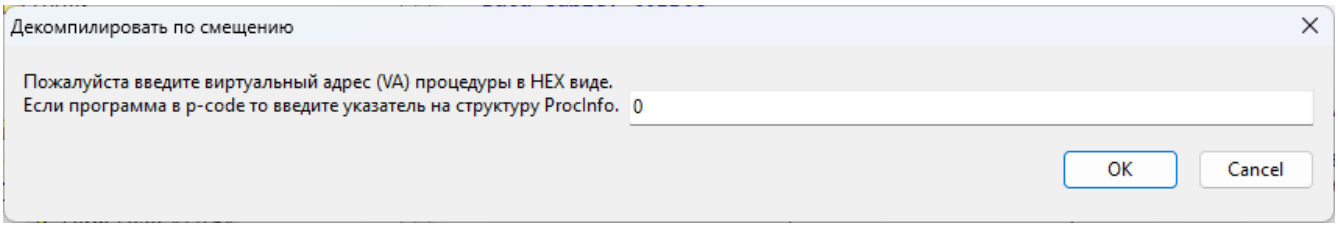
Быстрая декомпиляция и большие списки строк

Ссылки на строки можно собрать при предварительном дизассемблировании, включая режим **«Быстрая декомпиляция» (Fast decompilation)**. Для этого не требуется заранее формировать полный декомпилированный текст проекта. В версии 26.2 диалог поддерживает списки с более чем 32 000 ссылок; также улучшена работа с памятью при обработке больших списков.

Для .NET-сборок пользовательские строки из кучи #US можно просматривать вместе с их расположением в файле. В списке отображаются строки, доступные для статического анализа; зашифрованные или динамически генерируемые строки могут требовать отдельного исследования.

7.4 Декомпиляция с заданного адреса

В меню **«Утилиты» (Tools)** имеется очень полезная функция — **«Декомпилировать с VA» (Decompile from VA)** (декомпилировать любой код по указанному виртуальному адресу). Основное преимущество этой функции заключается в том, что она не требует корректной структуры файла и его предварительной обработки обычным способом. Конечно, в этом случае код будет декомпилирован с рядом ограничений. Например, если импортированные функции или объекты OLE не были предварительно обработаны, соответствующие фрагменты кода могут оказаться малоинформативными.



Для декомпиляции конкретного фрагмента кода внутри файла необходимо просто указать виртуальный адрес этого кода (если речь идет о **Native Code** или **сборке .NET**), либо указать адрес структуры **ProcInfo** (если речь идет о классическом **Visual Basic P-Code**). Особенность работы с **P-Code** заключается в том, что структура **ProcInfo** содержит таблицу данных, используемых кодом. Без этой таблицы невозможно определить, с какими функциями работает код, какие строки он использует и так далее. При работе со **сборками .NET** после ввода адреса декомпилятор предложит выбор между обработкой промежуточного кода **MS IL** и дизассемблированием машинного кода. Данный вариант обусловлен тем, что .NET-сборки могут содержать как промежуточный код MS IL (**управляемый код**), так и машинный код (так называемый **неуправляемый код**).

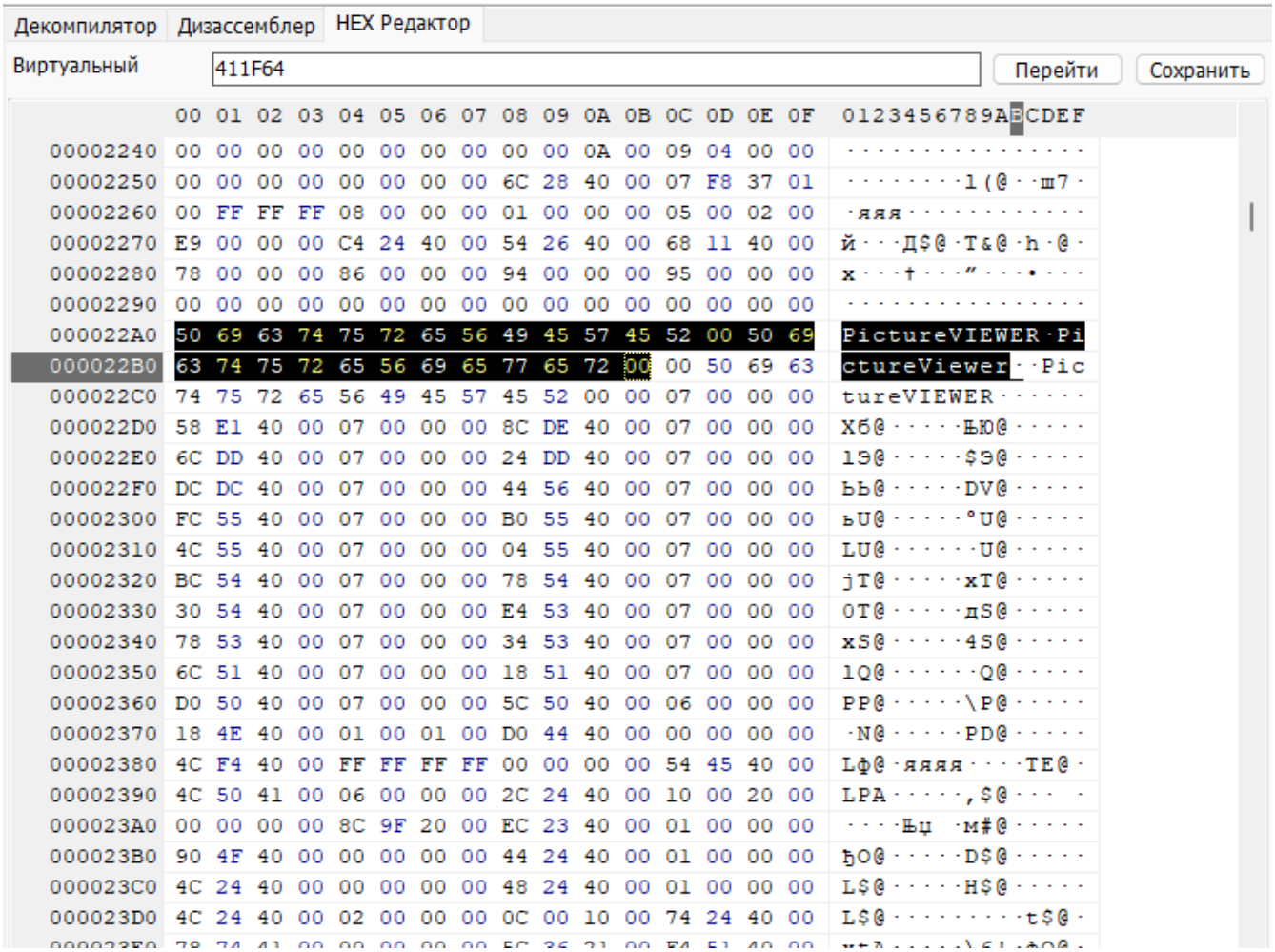
Если файл уже открыт в декомпиляторе, достаточно вызвать функцию **«Декомпилировать с VA»** и указать адрес для обработки. Если файл еще не открыт, это будет предложено автоматически, и файл откроется в упрощенном режиме.

Функция **«Декомпилировать с VA»** особенно полезна, когда нужный код не декомпилируется автоматически или если файл частично поврежден, что препятствует его обработке стандартным способом. Конечно, следует понимать, что результат декомпиляции в этом режиме может быть менее полным, чем при стандартной декомпиляции.

7.5 Редактирование данных

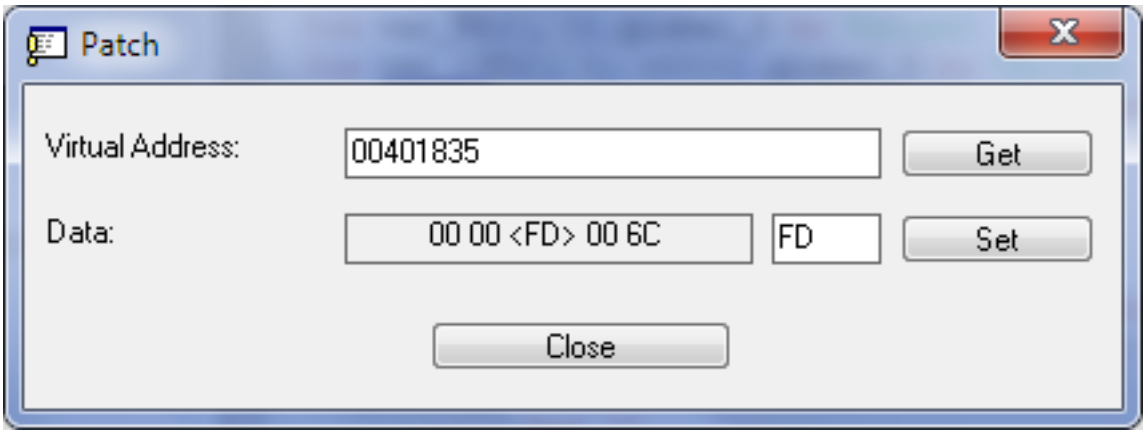
Начиная с версии 9.4, **VB Decompiler** поставляется с современным и удобным инструментом для редактирования двоичных данных внутри файлов — полноценным **«HEX Редактор» (HEX Editor)**. Он доступен из главного окна программы вместе с декомпилированным кодом во вкладке **«HEX Редактор»** для каждой процедуры или формы.

«HEX Редактор» позволяет редактировать байты файла по отдельности и блоками. Для навигации к нужному адресу в анализируемом файле можно использовать либо виртуальные адреса (включая **Image Base**), либо смещения (обычный адрес, указывающий смещение необходимых данных относительно начала файла).



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00002240	00	00	00	00	00	00	00	00	00	00	0A	00	09	04	00	00	····
00002250	00	00	00	00	00	00	00	00	6C	28	40	00	07	F8	37	01	····l(@··ш7·
00002260	00	FF	FF	FF	08	00	00	00	01	00	00	00	05	00	02	00	·яяя····
00002270	E9	00	00	00	C4	24	40	00	54	26	40	00	68	11	40	00	й···Д$@·T&@·h·@·
00002280	78	00	00	00	86	00	00	00	94	00	00	00	95	00	00	00	x···†···”···•···
00002290	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	····
000022A0	50	69	63	74	75	72	65	56	49	45	57	45	52	00	50	69	PictureVIEWER·Pi
000022B0	63	74	75	72	65	56	69	65	77	65	72	00	00	50	69	63	ctureViewer··Pic
000022C0	74	75	72	65	56	49	45	57	45	52	00	00	07	00	00	00	tureVIEWER·····
000022D0	58	E1	40	00	07	00	00	00	8C	DE	40	00	07	00	00	00	Xб@·····ЊЮ@·····
000022E0	6C	DD	40	00	07	00	00	00	24	DD	40	00	07	00	00	00	lЭ@·····$Э@·····
000022F0	DC	DC	40	00	07	00	00	00	44	56	40	00	07	00	00	00	ЬЬ@·····DV@·····
00002300	FC	55	40	00	07	00	00	00	B0	55	40	00	07	00	00	00	ьU@·····°U@·····
00002310	4C	55	40	00	07	00	00	00	04	55	40	00	07	00	00	00	LU@·····U@·····
00002320	BC	54	40	00	07	00	00	00	78	54	40	00	07	00	00	00	jT@·····xT@·····
00002330	30	54	40	00	07	00	00	00	E4	53	40	00	07	00	00	00	0T@·····дS@·····
00002340	78	53	40	00	07	00	00	00	34	53	40	00	07	00	00	00	xS@·····4S@·····
00002350	6C	51	40	00	07	00	00	00	18	51	40	00	07	00	00	00	lQ@·····Q@·····
00002360	D0	50	40	00	07	00	00	00	5C	50	40	00	06	00	00	00	PP@·····\P@·····
00002370	18	4E	40	00	01	00	01	00	D0	44	40	00	00	00	00	00	·N@·····PD@·····
00002380	4C	F4	40	00	FF	FF	FF	FF	00	00	00	00	54	45	40	00	Lф@·яяяя····TE@·
00002390	4C	50	41	00	06	00	00	00	2C	24	40	00	10	00	20	00	LPA·····,$@···· ·
000023A0	00	00	00	00	8C	9F	20	00	EC	23	40	00	01	00	00	00	····Њџ ·м#@·····
000023B0	90	4F	40	00	00	00	00	00	44	24	40	00	01	00	00	00	ђO@·····D$@·····
000023C0	4C	24	40	00	00	00	00	00	48	24	40	00	01	00	00	00	L$@·····H$@·····
000023D0	4C	24	40	00	02	00	00	00	0C	00	10	00	74	24	40	00	L$@·····t$@·

В более ранних версиях декомпилятора (до версии 9.4) для редактирования файлов использовался более простой инструмент: Data Patcher. Он работал следующим образом. В меню **«Утилиты» (Tools)** -> **«Патч данных» (Patch data)** присутствовал инструмент, называемый патчером, который можно было использовать в случаях, когда после анализа кода программы необходимо было изменить какую-то часть. Для изменения байтов без внешних инструментов **VB Decompiler** предоставлял встроенный редактор. Окно патчера выглядело следующим образом:



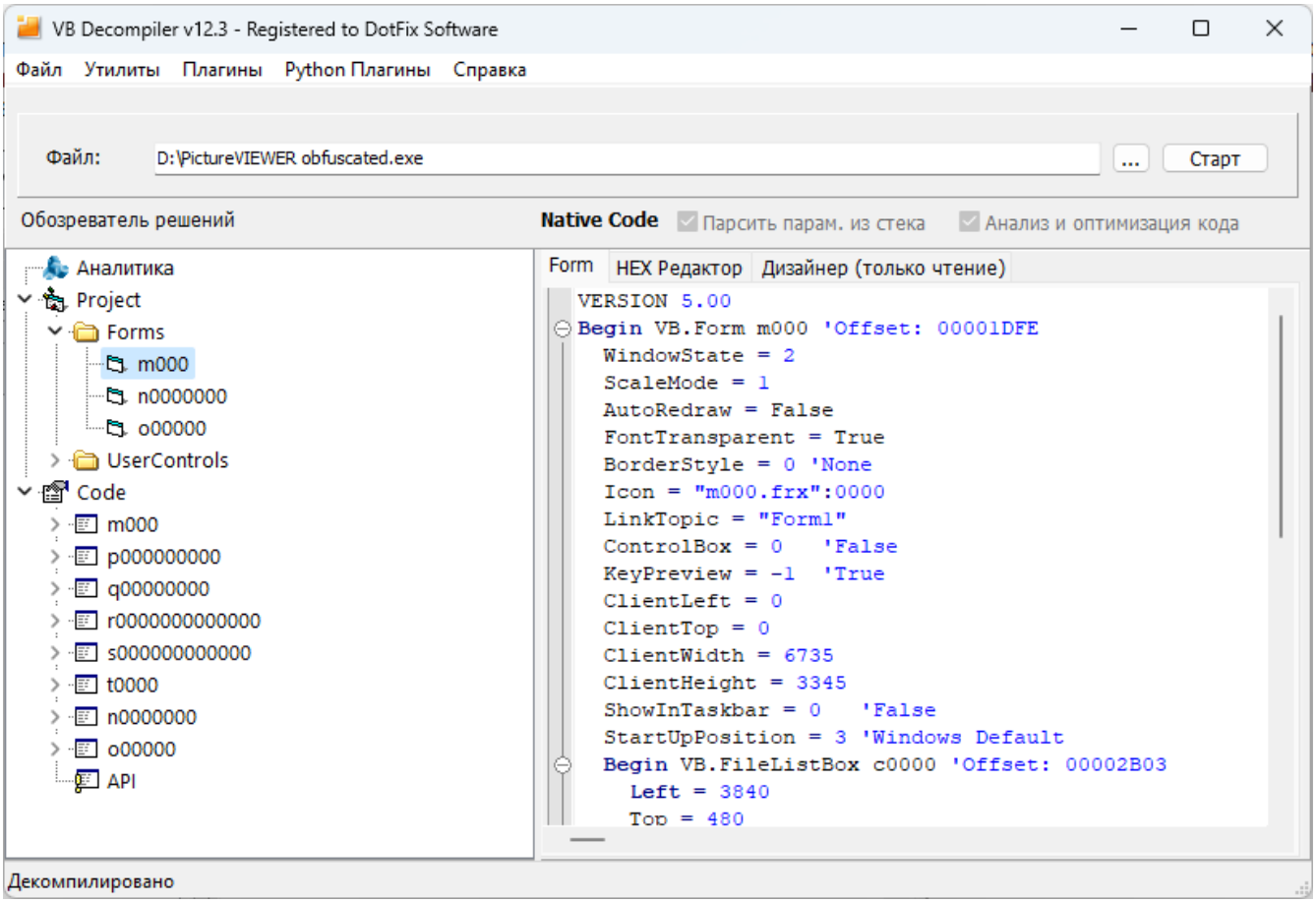
Работа с патчером была чрезвычайно проста. В поле **«Виртуальный адрес» (Virtual Address)** вы вводили адрес для редактирования байта, затем либо нажимали **Enter**, либо нажимали кнопку **«Считать» (Get)**. Байты данных, расположенные вблизи выбранного адреса, отображались в поле Data, и выбранный байт можно было изменить, после чего необходимо было нажать **«Записать» (Set)**, и введенный байт изменялся.

Если в Параметрах программы был включен параметр **«Быстрая декомпиляция» (Fast decompilation)** и отключен параметр **«Кэшировать декомпилированный код» (Cache decompiled code)**, то все изменения отражались в той функции, где были изменены байты. Для этого было достаточно повторно открыть ту же функцию двойным щелчком по её имени в дереве.

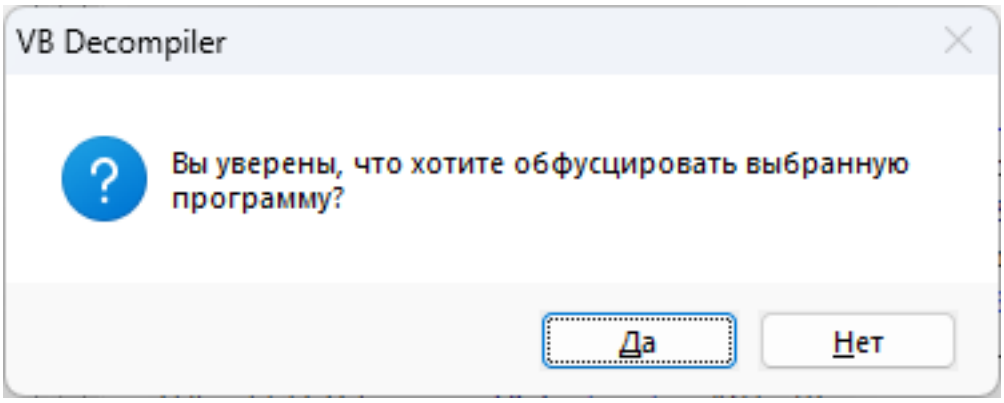
Важное примечание: эта возможность предназначена для редактирования только ваших файлов программ или тех, на которые у вас есть соответствующие права, а также исследования вредоносного кода в антивирусных лабораториях. Перед изменением чужого кода — убедитесь, что вы ознакомлены с лицензионным соглашением относительно наличия такого права.

7.6 Обфускация

Если вы разрабатываете программы в среде **Visual Basic 5.0/6.0**, вы можете повысить их защиту от декомпиляции. Для этого в декомпилятор встроен модуль **Obfuscator**, способный удалять все имена функций, форм, модулей и объектов из двоичного файла вашей программы, а также различные сигнатуры. Это затруднит анализ программы как с помощью VB Decompiler, так и других решений. Более того, большинство сторонних продуктов откажутся декомпилировать этот файл вообще.



Чтобы применить эту защиту, откройте свой проект в декомпиляторе и выберите **«Обфускация проекта» (Obfuscate project)** из меню **«Утилиты» (Tools)**. Этот процесс не занимает много времени, поэтому мы рекомендуем обфусцировать ваши проекты перед их публикацией.



Эта функция также интегрирована в наше комплексное решение по защите программного обеспечения — **DotFix NiceProtect**, которое можно загрузить с www.niceprotect.com.

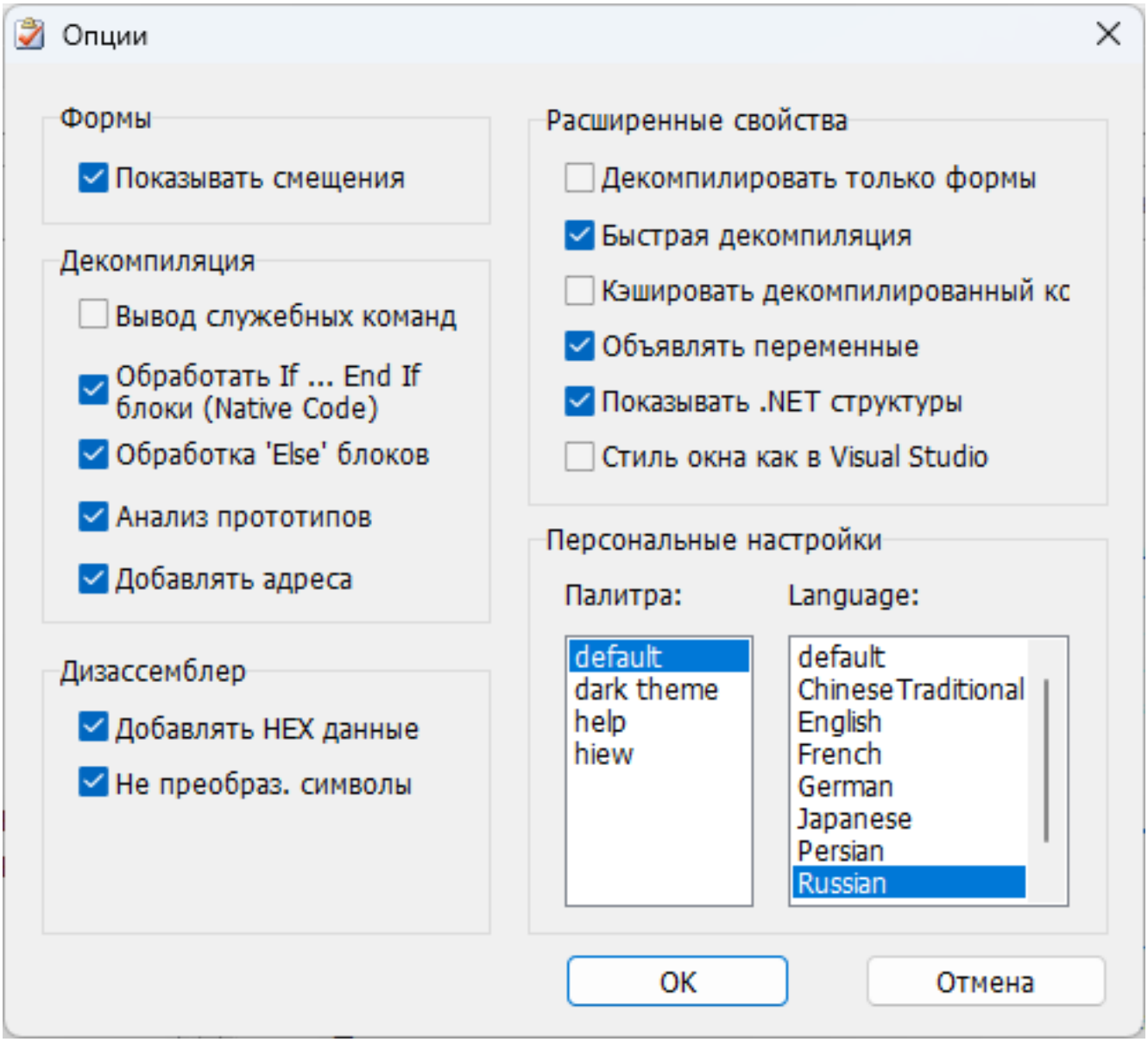
Обфускация доступна только владельцам лицензии Business продукта **VB Decompiler**.

Ограничения обфускации

8 Параметры программы

8.1 Параметры программы

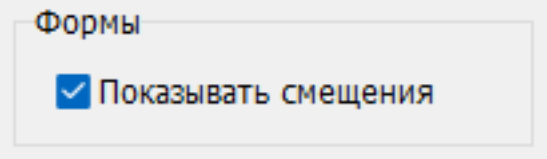
Чтобы изменить настройки программы, перейдите в меню **«Утилиты» (Tools)** > **«Опции» (Options)**. В появившемся диалоговом окне вы увидите следующие параметры:



Формы
Декомпиляция
Дизассемблер
Цветовая схема Расширенные возможности Выбор языка

8.2 Формы

Эти параметры управляют отображением данных форм.



В этом разделе настроек можно включить отображение смещений рядом с объектами форм. Это может быть необходимо, например, при переводе интерфейса программы на другой язык без доступа к исходному коду. **«Показывать смещения» (Show offsets)** показывает расположение объектов, их имён и свойств в файле.

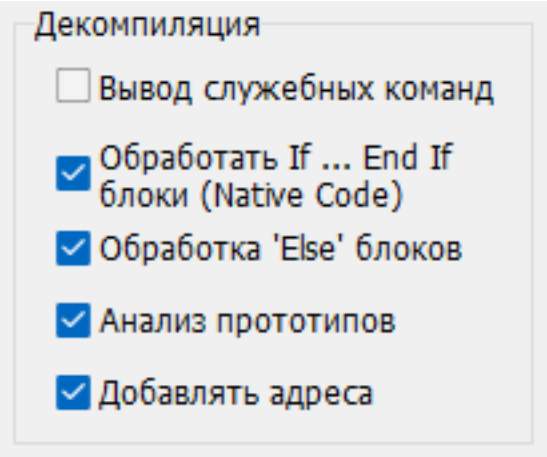
Затем, используя HEX-редактор, вы можете изменить эти значения. Включайте отображение смещений, если планируете исследовать или редактировать двоичные данные формы. Ошибочные изменения могут нарушить работу файла. Для обычного просмотра форм отображение смещений не требуется.

Визуальный просмотр и навигация по событиям

Эти параметры не управляют возможностями визуального представления **«Дизайнер (только чтение)» (Designer (read only))** . См. Визуальное восстановление форм с описанием классических форм VB, Windows Forms, WPF/BAML, переходов к обработчикам событий и ограничений восстановления.

8.3 Декомпиляция

Откройте **«Утилиты > Опции» (Tools > Options)** для настройки декомпиляции. При интерактивном сравнении настроек используйте Быструю декомпиляцию без кэширования; в противном случае обработайте файл повторно после изменения опции.



Вывод служебных команд (Show debug commands)

Для классического P-Code VB включает операции компилятора и отладчика, которые в противном случае были бы опущены из высокоуровневого листинга. Включите этот параметр для исследования низкоуровневой последовательности инструкций, а не только восстановленной логики программы.

Обработать If ... End If блоки (Native Code) (Process If ... End If blocks (Native Code))

Восстанавливает распознанные условные переходы Native Code в виде блоков `If...End If`. Отключите этот параметр, если требуется проверить исходную последовательность переходов или изучить процедуру, структура которой кажется некорректной.

Обработка 'Else' блоков (Process 'Else' blocks)

Использует дополнительные эвристики для отделения ветвей Else от вложенных условий и переходов GoTo. Для Native Code необходимо также включить обработку блоков If. Обычно этот параметр следует оставлять включённым. Если восстановленная ветвь неоднозначна, сравните результаты с включённым и отключённым параметром.

Анализ прототипов (Analyze prototypes)

Выполняет предварительный эвристический анализ аргументов и возвращаемых значений процедур. Это особенно полезно при использовании Быстрой декомпиляции: функции и процедуры, объявленные как Private, в стандартных модулях могут не иметь явных метаданных сигнатуры. Предварительный анализ занимает дополнительное время, но позволяет учитывать сигнатуру процедуры при анализе вызова, не полагаясь только на текущее содержимое стека. Эвристически определенная сигнатура остается предположением, особенно когда метаданные, внешние типы или код неполны.

Добавлять адреса (Add Addresses)

Отображает виртуальные адреса рядом с восстановленными операторами. Оставьте этот параметр включённым, чтобы сопоставлять код с дизассемблированными инструкциями, байтами файла и результатами других инструментов. Он также требуется команде **«Скопировать в дизассемблер» (Copy to disassembler)** для объединения высокоуровневых операторов с соответствующими низкоуровневыми инструкциями. Дизассемблер всегда отображает адреса независимо от этого параметра.

Особенности восстановления Native Code

Движок Native Code распознаёт поддерживаемые шаблоны `Select Case`, доступ к многомерным массивам, вызовы COM и операции среды выполнения VB. Обработка ошибок может быть восстановлена как `On Error Resume Next`, `On Error GoTo` и `On Error GoTo 0`. Оптимизация компилятора и отсутствие информации о типах всё ещё могут ограничивать результат; проверяйте дизассемблированный код, если высокоуровневый листинг неполон.

Версия 26.2 расширила типизацию Variant, чтение и запись многомерных SafeArray и восстановление `VarPtr` для элементов массива. В листинге ссылка на массив может быть обозначена как `SafeArray`, отражая структуру среды выполнения, распознанную движком анализа. Раздельное отслеживание состояния в ветвях и анализ временных переменных также улучшают восстановление сложного кода If/Else. Версия 26.5 дополнительно улучшает обработку Select Case и функций среды выполнения VB5/VB6.

Информация о типах COM и ActiveX

VB Decompiler сначала использует свою встроенную базу знаний для известных интерфейсов. Для неизвестного внешнего объекта COM он может найти сведения о библиотеке, классе и интерфейсе по их идентификаторам в системном реестре и разобрать соответствующую библиотеку типов DLL/OCX. Это предоставляет сигнатуры методов и свойств, включая типы возвращаемых интерфейсов, для ранней привязки и поддерживаемых вызовов с поздней привязкой.

Если требуемая библиотека типов отсутствует или сведения о её регистрации недоступны, некоторые вызовы могут остаться неидентифицированными, например как `UnkVCall`. Поэтому при сравнении результатов учитывайте наличие зависимостей на разных рабочих станциях анализа. Расширенная база знаний IID включает автоматизацию Office, ADO/DAO, XML и HTTP, сценарии, системную оболочку, обмен сообщениями и службы каталогов; это не подразумевает, что любой вызов COM будет распознан.

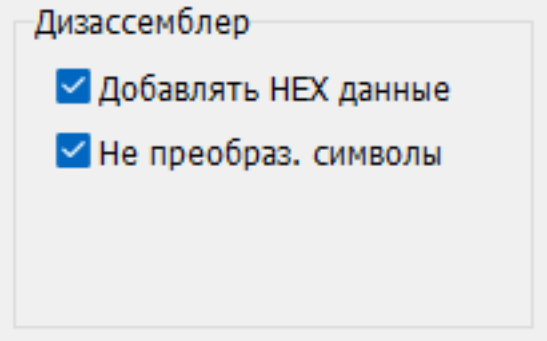
В некоторых двоичных файлах VB сохраняется лишь ограниченный объем метаданных UDT. Декомпилятор может восстановить информацию, такую как размер структуры и число строковых полей, без восстановления полной исходной структуры полей. Не интерпретируйте частичное объявление как полное определение исходной структуры.

Восстановление .NET и оптимизация

Из поддерживаемых инструкций IL восстанавливаются выражения и управляющие конструкции C#. Параметр **«Анализ и оптимизация кода» (Procedure analyzer and optimizer)** в главном окне может упростить поддерживаемые шаблоны циклов, условий, операторов и временных переменных. Если фрагмент не поддерживается или обфусцирован, проверяйте IL и метаданные. Эта обработка не зависит от «ИИ Помощник» (AI Helper).

8.4 Дизассемблер

Эта группа параметров управляет дизассемблированием машинного кода:



«Добавлять HEX данные» (Add HEX dump) отображает машинные коды перед каждой дизассемблированной инструкцией, что полезно при проверке и изменении двоичного файла в HEX-редакторе.

«Не преобраз. символы» (Ignore chars convertions) используется для отключения преобразования неанглоязычных символов строк в числовые коды Unicode. Включение этой опции делает отображаемые строки более удобочитаемыми. Отключите её, когда требуется числовое представление Unicode.

«Показывать польз. классы» (Show user classes) включает разбор пользовательских классов внутри приложений **Delphi** / **Free Pascal**. Эта опция не используется для приложений **Visual Basic** и **C#**.

«Показывать VCL классы» (Show VCL classes) включает декомпиляцию всех внутренних классов библиотеки **VCL**. Рекомендуется отключить эту опцию, чтобы декомпилятор не анализировал тысячи служебных классов, которые обычно не требуются при анализе программ. Эта опция используется только для декомпиляции приложений **Delphi** во внутренних целях и была создана для унификации с другими продуктами, но она не реализована в **VB Decompiler**.

Частичная декомпиляция приложений **Delphi** и **Free Pascal** в настоящее время предлагается исключительно как отдельная услуга (не как программное обеспечение).

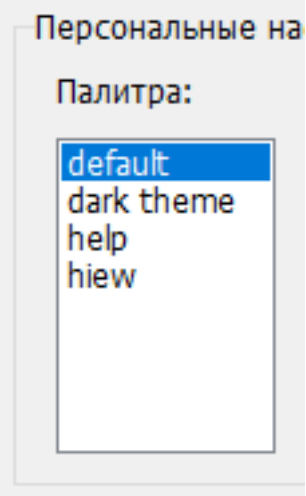
VB P-Code и .NET IL

«Добавлять HEX данные» также поддерживает низкоуровневый листинг P-Code. Чтобы сопоставить восстановленные операторы с P-Code или машинными инструкциями в одном листинге, используйте **«Скопировать в дизассемблер» (Copy to disassembler)** в представлении «Декомпилятор» (Decompiler).

Для сборок .NET дизассемблер может отображать IL или нативные методы x86/x64 по мере необходимости. Восстановленные сигнатуры локальных переменных могут включать массивы, обобщённые типы, типы значений и ссылки TypeSpec. Включите **«Показывать .NET структуры» (Show .NET structures)**, когда необходимо проверить таблицы и кучи метаданных вместе с кодом. Поддерживается анализ обфусцированных метаданных, в том числе таблицы FieldPtr; это не гарантирует успешное восстановление любой поврежденной или обфусцированной сборки.

8.5 Цветовая схема

В этом разделе описывается цветовая схема и подсветка синтаксиса при просмотре кода в декомпиляторе:



Наборы настроек хранятся в папке "colors" внутри файлов .ini, расположенных в директории установки VB Decompiler. Имена файлов определяют названия наборов в разделе настроек программы. Эти файлы имеют простой формат, и вы можете легко создать их самостоятельно или отредактировать существующие.

Формат конфигурации следующий:

```
[Colors]
Background = decimal color code (for example: 1973790)
Foreground = decimal color code (for example: 14671840)
KeyWords = decimal color code (for example: 13471849)
LocationsAndArguments = decimal color code (for example: 11849128)
Procedures = decimal color code (for example: 5676246)
; next properties is only available in version 12 or later
UnderCursorBackground = decimal color code (for example: 6986069)
UnderCursorForeground = decimal color code (for example: 1973790)
UseDarkMode = 0 or 1 (0 for Light Theme, 1 for Dark Theme)
Colors2 = 0 or 1 (using extended colors at Colors2 section)
```

```
[Font]
Name = font name (for example: Consolas)
Size = font size (for example: 11)
KeyWordsBold = 0 or 1 (font bold)
LocationsAndArgumentsBold = 0 or 1 (font bold)
ProceduresBold = 0 or 1 (font bold)
```

```
; this section is only available in version 12 or later
[Colors2]
TVFontName = Solution Explorer treeview font name or list of font names, separated by comma (for example:
    Segoe UI, Tahoma, Arial)
TVFontSize = size of the font (for example: 10)
FontName = Decompiler and Disassembler font name or list of font names, separated by comma (for example:
    Courier New, Consolas, Liberation Mono, monospace)
FontSize = size of the font (for example: 11)
Background = BackColor in hexadecimal for Decompiler and Disassembler tab (for example: 1E1E1E)
Foreground = ForeColor in hexadecimal for Decompiler and Disassembler tab (for example: DFDFE0)
; code highllighting
Key = color in hexadecimal (for example: CD9069)
Comment = color in hexadecimal (for example: CD9069)
Char = color in hexadecimal (for example: B4CDA8)
Float = color in hexadecimal (for example: B4CDA8)
Hex = color in hexadecimal (for example: B4CDA8)
Number = color in hexadecimal (for example: B4CDA8)
String = color in hexadecimal (for example: 569CD6)
Symbol = color in hexadecimal (for example: DFDFE0)
; HEX Editor font and colors
HexFontName = font name or list of font names, separated by comma (for example: Courier New, Consolas,
    Liberation Mono, monospace)
HexFontSize = size of the font (for example: 11)
HexBackground = color in hexadecimal (for example: 1E1E1E)
HexForeground = color in hexadecimal (for example: DFDFE0)
HexGrid = color in hexadecimal (for example: 1E1E1E)
HexActiveFieldBackground = color in hexadecimal (for example: 1E1E1E)
HexOffsetBackground = color in hexadecimal (for example: 1E1E1E)
HexCurrentOffset = color in hexadecimal (for example: B4CDA8)
HexOddColumn = color in hexadecimal (for example: B4CDA8)
HexEvenColumn = color in hexadecimal (for example: B4CDA8)
HexOffset = color in hexadecimal (for example: 569CD6)
```

Цвет задаётся 32-битным значением True Color: в десятичном формате в разделе Colors (для совместимости со старыми версиями VB Decompiler) и в шестнадцатеричном формате в разделе Colors2.

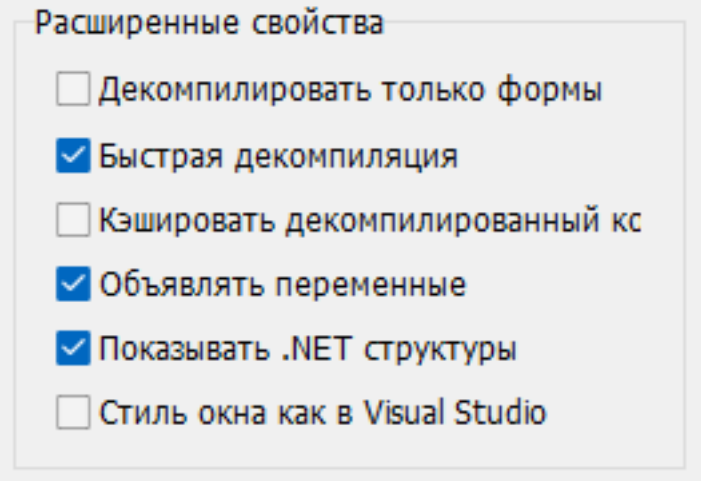
Смена тем при анализе кода

Система тем поддерживает тёмную тему и может применять выбранную тему без перезапуска VB Decompiler, в том числе после редактирования соответствующего файла INI. В расширенной конфигурации можно перечислить через запятую имена резервных шрифтов. Это полезно, если в Windows и среде совместимости доступны разные наборы шрифтов.

Настройки цветов для элемента под курсором управляют подсветкой идентификаторов. См. Просмотр кода для подсветки локальных переменных, членов объектов и глобальных переменных.

8.6 Расширенные возможности

Эти настройки управляют обработкой проекта и отображением восстановленной информации.



Декомпилировать только формы (Decompile only forms)

Отключает декомпиляцию кода, если вам нужны только определения восстановленных форм и UserControl. Возможности визуального просмотра описаны в разделе Визуальное восстановление форм.

Быстрая декомпиляция (Fast decompilation)

При включении VB Decompiler сначала обрабатывает структуру проекта и декомпилирует отдельные процедуры по мере их открытия. Это сокращает время первоначальной обработки при исследовании выбранных процедур. При отключенном **«Кэшировать декомпилированный код» (Cache decompiled code)** вы можете изменить настройки анализа и открыть процедуру заново для получения обновлённого листинга.

Включите **«Анализ прототипов» (Analyze prototypes)** в Настройках декомпиляции для проведения предварительного эвристического анализа сигнатур процедур, объявленных как Private. Это увеличивает время первоначальной обработки, но может улучшить восстановление аргументов без необходимости полной декомпиляции.

В режиме быстрой декомпиляции полный высокоуровневый листинг проекта не формируется заранее. Для поиска по всему декомпилированному тексту или экспорта полного проекта отключите этот режим и обработайте файл заново. Ссылки на строки собираются отдельно, при предварительном дизассемблировании, и доступны в режиме быстрой декомпиляции.

Полная декомпиляция

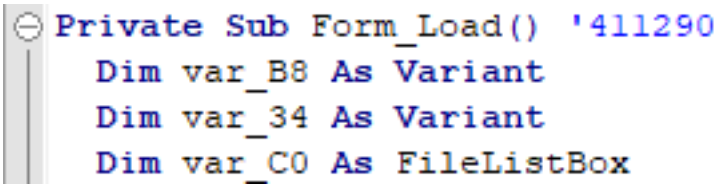
При отключенной быстрой декомпиляции декомпилятор обрабатывает процедуры проекта заранее. Это занимает больше времени, но позволяет использовать полный восстановленный листинг для поиска по тексту и экспорта. Изменения настроек декомпиляции требуют повторной обработки файла.

Кэшировать декомпилированный код

Сохраняет уже полученные листинги в режиме быстрой декомпиляции. Это позволяет не декомпилировать процедуру повторно при каждом открытии, но кэшированные результаты не отражают последующие изменения настроек анализа. При интерактивном сравнении настроек **«Парсить парам. из стека» (Parse stack parameters)** или **«Анализ и оптимизация кода» (Procedure analyzer and optimizer)** отключите кэширование.

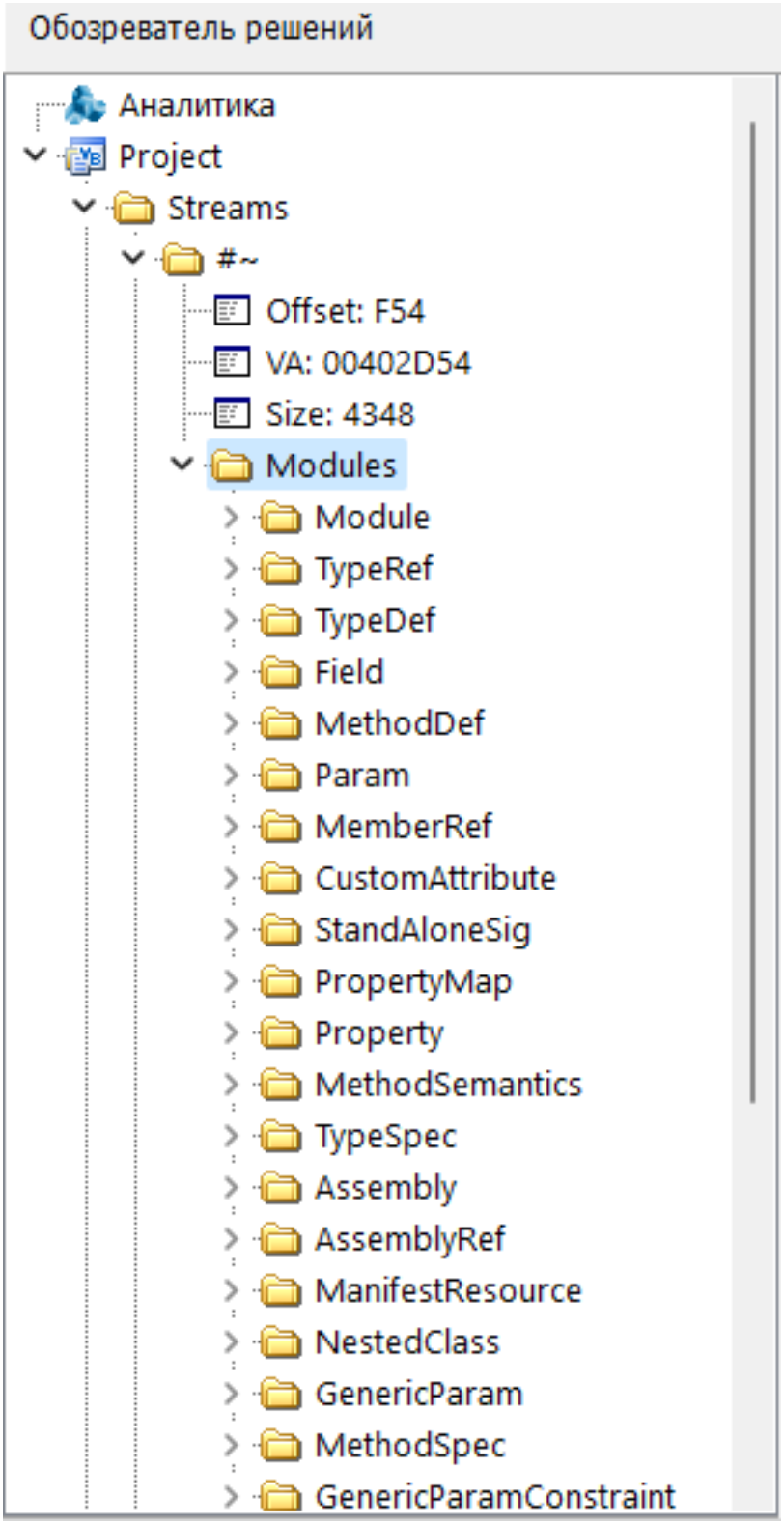
Объявлять переменные (Declare variables)

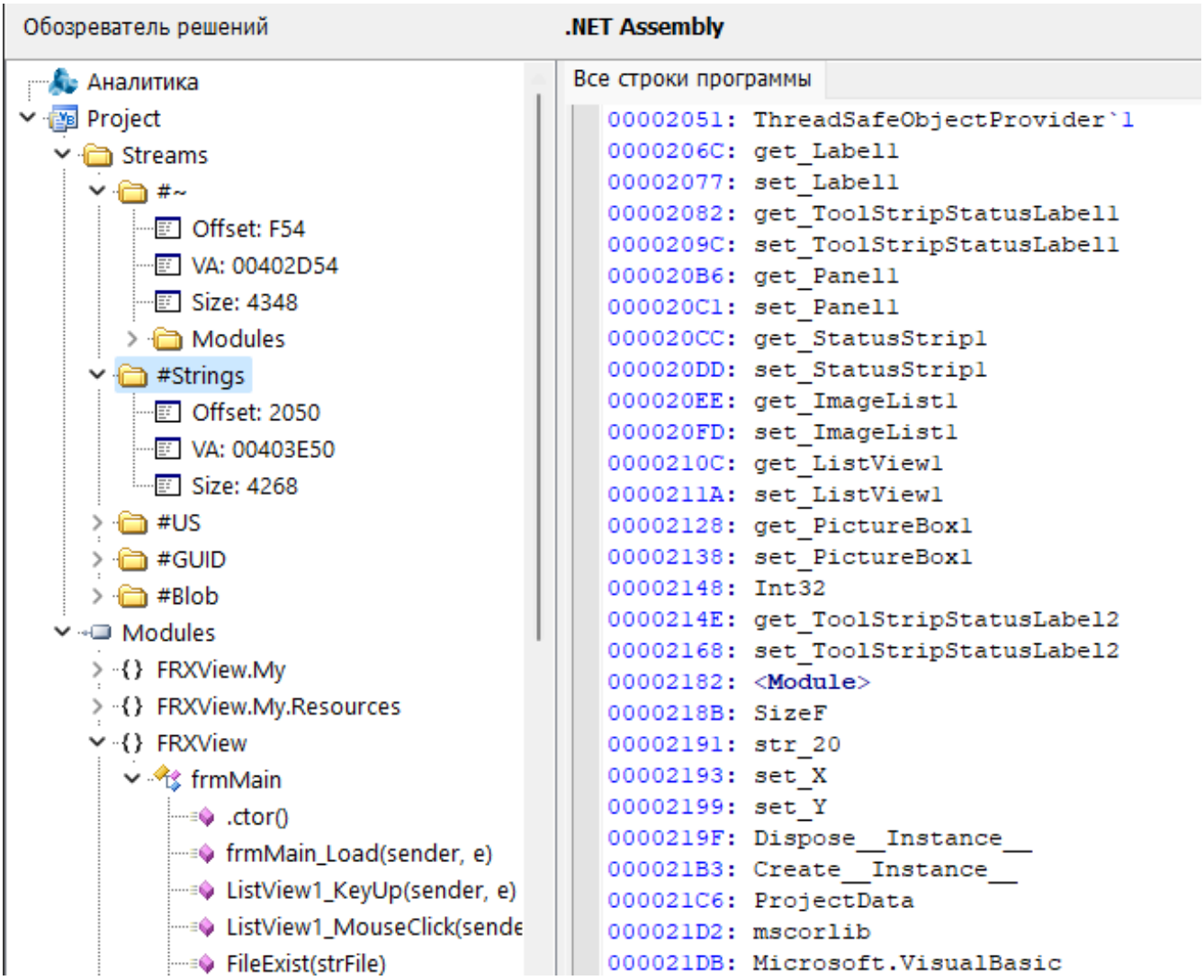
Добавляет блок объявлений для восстановленных переменных в начале каждой декомпилированной процедуры. Это полезно для восстановления исходного кода и проверки типов. Если вам нужен краткий обзор логики процедуры, объявления можно скрыть. Восстановленные типы зависят от метаданных и анализа, доступных декомпилятору.



Показывать .NET структуры (Show .NET structures)

Отображает таблицы и кучи метаданных .NET в дереве «Обозреватель решений» (Solution explorer). Используйте это для проверки структур и сигнатур вместе с кодом. При выборе таблицы отображается её содержимое; куча #US содержит пользовательские строки. Отключите опцию, если вам не нужна навигация по метаданным.


Обозреватель решений
Аналитика
Project
Streams
#~
Offset: F54
VA: 00402D54
Size: 4348
Modules
Module
TypeRef
TypeDef
Field
MethodDef
Param
MemberRef
CustomAttribute
StandAloneSig
PropertyMap
Property
MethodSemantics
TypeSpec
Assembly
AssemblyRef
ManifestResource
NestedClass
GenericParam
MethodSpec
GenericParamConstraint



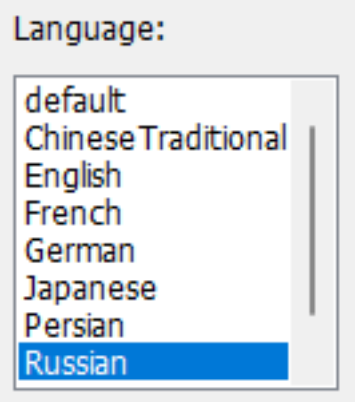
Стиль окна как в Visual Studio (Window style as in Visual Studio)

Размещает код слева, а «Обозреватель решений» справа. Это меняет компоновку рабочей области, не изменяя восстановленный код.

8.7 Выбор языка

VB Decompiler поддерживает более 20 языков интерфейса, используемых во всём мире. Документация также доступна на нескольких языках.

Вы можете выбрать предпочитаемый язык интерфейса программы в меню **«Утилиты» (Tools)** -> **«Опции» (Options)**, в разделе **«Персональные настройки» (Personalize)** настроек. После выбора языка необходимо перезапустить **VB Decompiler** для корректного применения языковых настроек во всей программе. Если перезапуск программы нежелателен, вы можете продолжить использование **VB Decompiler**; однако, часть текста может отображаться на предыдущем языке.



Язык справки выбирается автоматически на основе системного языка вашего компьютера. Для чтения руководства на другом языке откройте соответствующий файл справки из папки, где установлен **VB Decompiler**. PDF является основным форматом для локализованных руководств; файлы справки CHM остаются доступными для некоторых языков.

Английское руководство является источником для всех переводов. Если вы заметили ошибку или несоответствие в локализованном руководстве, пожалуйста, сообщите об этом технической поддержке.

Открытие руководства

Если локальный файл справки или требуемый просмотрщик PDF недоступны, VB Decompiler может открыть онлайн-документацию. Если анализ проводится без доступа к сети, заранее сохраните необходимые локальные файлы справки.

9 Плагины

Плагины расширяют возможности VB Decompiler с помощью нативных DLL. Установленные плагины доступны из меню **«Плагины» (Plugins)**. API, доступный через функцию обратного вызова, предоставляет доступ к восстановленной информации о проекте и коду, позволяя плагину проверять или изменять поддерживаемые внутренние структуры.

FRX Image Viewer

FRX Image Viewer 1.2.0 позволяет просматривать и извлекать изображения, зарегистрированные VB Decompiler или DE «Декомпилятор» (Decompiler). Для графики форм VB5/6 и .NET используйте VB Decompiler, для графики бинарных DFM Delphi — DE «Декомпилятор». Поддерживаются BMP, PNG, JPEG, GIF, ICO и CUR.

Установка и запуск

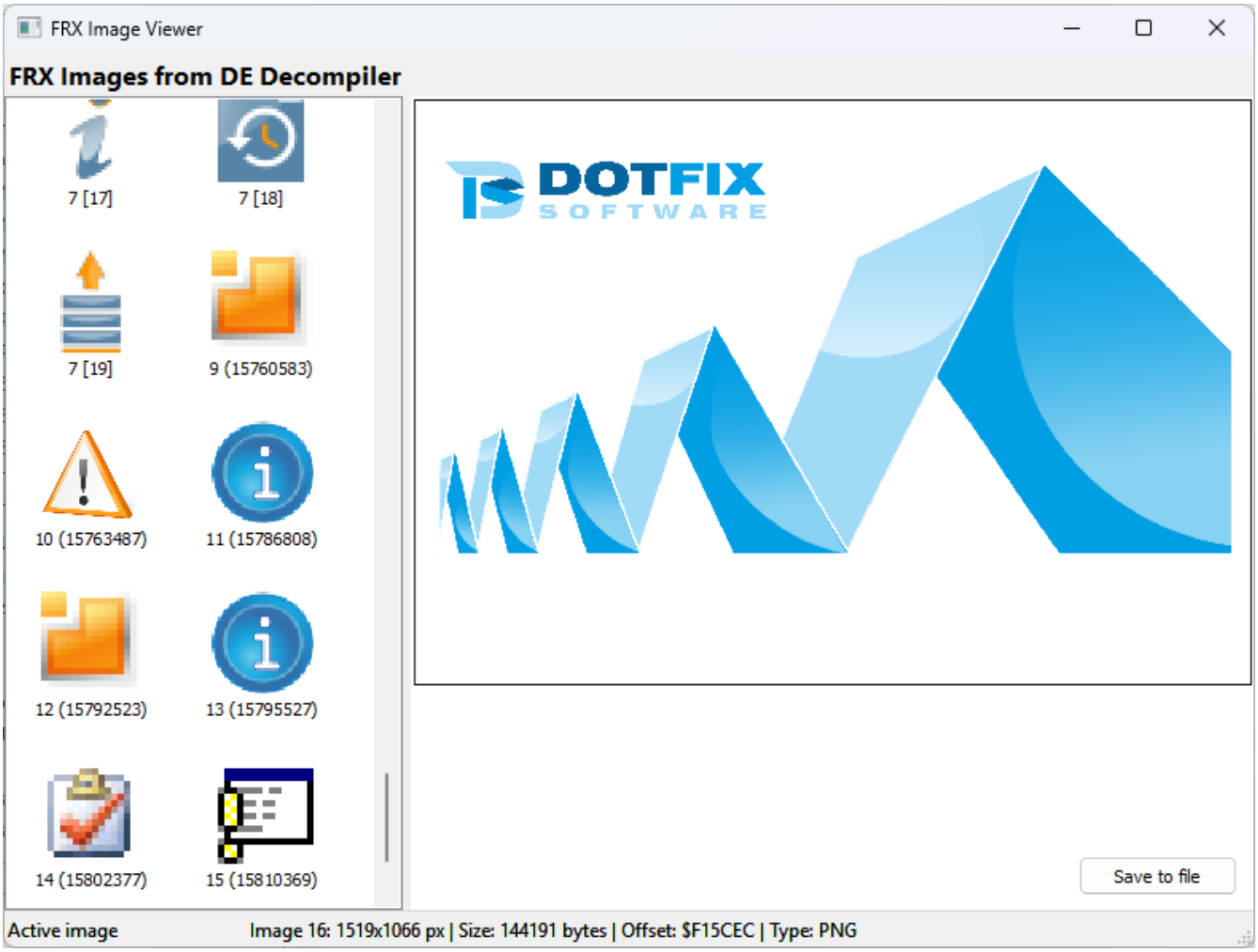
1. Скопируйте `FRXViewer.dll` в подпапку `plugins` установленного декомпилятора и перезапустите его. Разрядность DLL должна соответствовать самому декомпилятору, независимо от разрядности исследуемого файла.
2. Откройте EXE или DLL и выполните разбор форм, чтобы декомпилятор зарегистрировал доступные изображения.
3. Выберите **«Плагины»** > **FRX Image Viewer**, затем нажмите миниатюру нужного изображения.

Архив версии 1.2.0 содержит DLL для хоста x86. Для хоста x64 нужна сборка соответствующей разрядности. Плагин требует Windows 10/11; инструкции по сборке приведены в репозитории исходного кода.

Просмотр и экспорт

Галерея и область предварительного просмотра показывают доступные изображения, в том числе составные ICO/CUR. В строке состояния указаны выбранный элемент, размеры, объём данных или смещение исходного растра и формат. Плагин использует тему декомпилятора и поддерживает тёмное оформление.

Нажмите **Save to file**, чтобы экспортировать выбранный элемент. Отдельные изображения сохраняются с исходными байтами и в исходном формате. Элементы ImageList экспортируются в ICO с масками и альфа-прозрачностью; эти ICO формируются из исходных растров списка.



FRX Image Viewer в DE «Декомпилятор»: графика DFM Delphi и отдельные значки ImageList.

Нативные ImageList Delphi и совместимость

В совместимом декомпиляторе распознанные нативные ImageList Delphi отображаются как отдельные значки. Изображения, независимо закодированные в элементах коллекций, остаются обычными отдельными изображениями. Для неподдерживаемых потоков сохраняется просмотр целого изображения, если декомпилятор зарегистрировал соответствующие исходные данные.

Общая редакция SDK 26.6, поддерживаемая VB Decompiler 26.6 и DE «Декомпилятор» 26.5, добавляет необязательные команды `GetFrxImageListInfo` (63) и `GetProductName` (64). В старых хостах сохраняются исходный API изображений и просмотр целых изображений. Проверка возможностей и правила смещений описаны в справочнике SDK.

При просмотре и извлечении исследуемый файл не выполняется, его компоненты не загружаются. Для анимированных GIF отображается первый кадр. Доступность ресурсов зависит от успешного разбора и регистрации изображений декомпилятором.

Разработка плагина

SDK включает примеры для Visual C++, Rust, Delphi и Free Pascal. См. справочник VB Decompiler Plugin API с описанием экспортируемых точек входа, сигнатуры функции обратного вызова, идентификаторов команд, форматов данных и примеров проектов.

DLL-плагин предоставляет информацию о своем имени и авторе и получает указатель на функцию обратного вызова хоста через `VBDecompilerPluginLoad`. Функция обратного вызова используется для запроса или обновления поддерживаемых данных проекта. При реализации этого интерфейса соблюдайте требования SDK

к соглашениям о вызовах, указателям, кодировкам строк и буферам; при использовании управляемого языка обеспечьте совместимость с нативным интерфейсом через соответствующий слой взаимодействия.

VB Decompiler инициализирует доступные DLL-плагины при запуске. Собирайте плагины для целевой архитектуры хоста и среды Windows, а также проверяйте полученную DLL в этой среде. SDK предоставляет доступ как к коду активного представления, так и к коду проекта; операции, требующие всех декомпилированных процедур, требуют полной декомпиляции, а не только листинга, полученного по запросу.

10 Визуальное восстановление форм

Вкладка **«Дизайнер (только чтение)» (Designer (read only))** отображает восстановленное визуальное представление поддерживаемых интерфейсов из проанализированного двоичного файла. Она помогает связать элементы интерфейса с поведением приложения без ручного анализа каждого восстановленного свойства или оператора инициализации.

Открытие восстановленной формы

1 Откройте приложение и дайте VB Decompiler восстановить структуру проекта.

2 Выберите восстановленную форму в окне «Обозреватель решений» (Solution explorer) и откройте **«Дизайнер (только чтение)»**.

3 Выберите элемент управления, чтобы просмотреть его свойства и восстановленные сведения о событиях. Используйте события для навигации к соответствующему обработчику события.

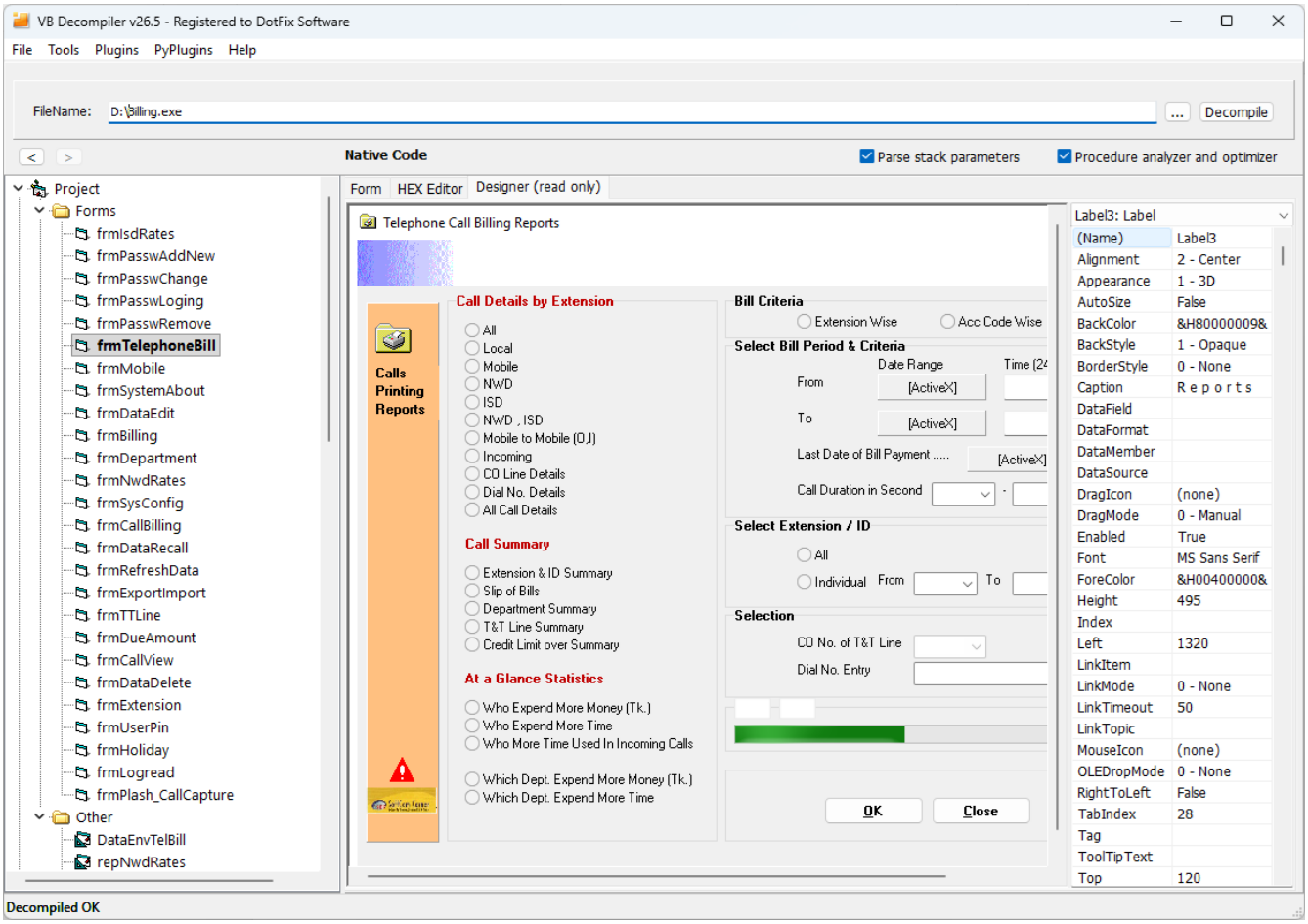
Предварительное представление строится по данным из файла. Целевое приложение при этом не запускается. Неподдерживаемые элементы управления, динамически создаваемый контент, отсутствующие ресурсы и пользовательская логика компоновки могут приводить к неполным или приблизительным результатам.

Visual Basic 5.0/6.0

Просмотрщик восстанавливает формы и UserControl, включая поддерживаемые свойства элементов управления, шрифты, цвета и изображения. Он считывает двоичные данные форм, используемые списками, выпадающими списками и другими элементами управления. Изображения могут включать значки, курсоры, растровые изображения, JPEG, PNG и GIF.

Визуальное меню восстанавливается в приложениях как P-Code, так и Native Code. Поддерживаемые элементы управления включают `SSTab`, `MSComctlLib.StatusBar`, `MSComctlLib.ListView`, `MSComctlLib.ProgressBar` и `MSComctlLib.TreeView`.

Для проверки обработчика элемента управления классического VB выберите элемент управления и дважды щелкните его поле **_Event** в списке свойств. Выберите требуемое событие в открывшемся списке, чтобы перейти к обработчику.

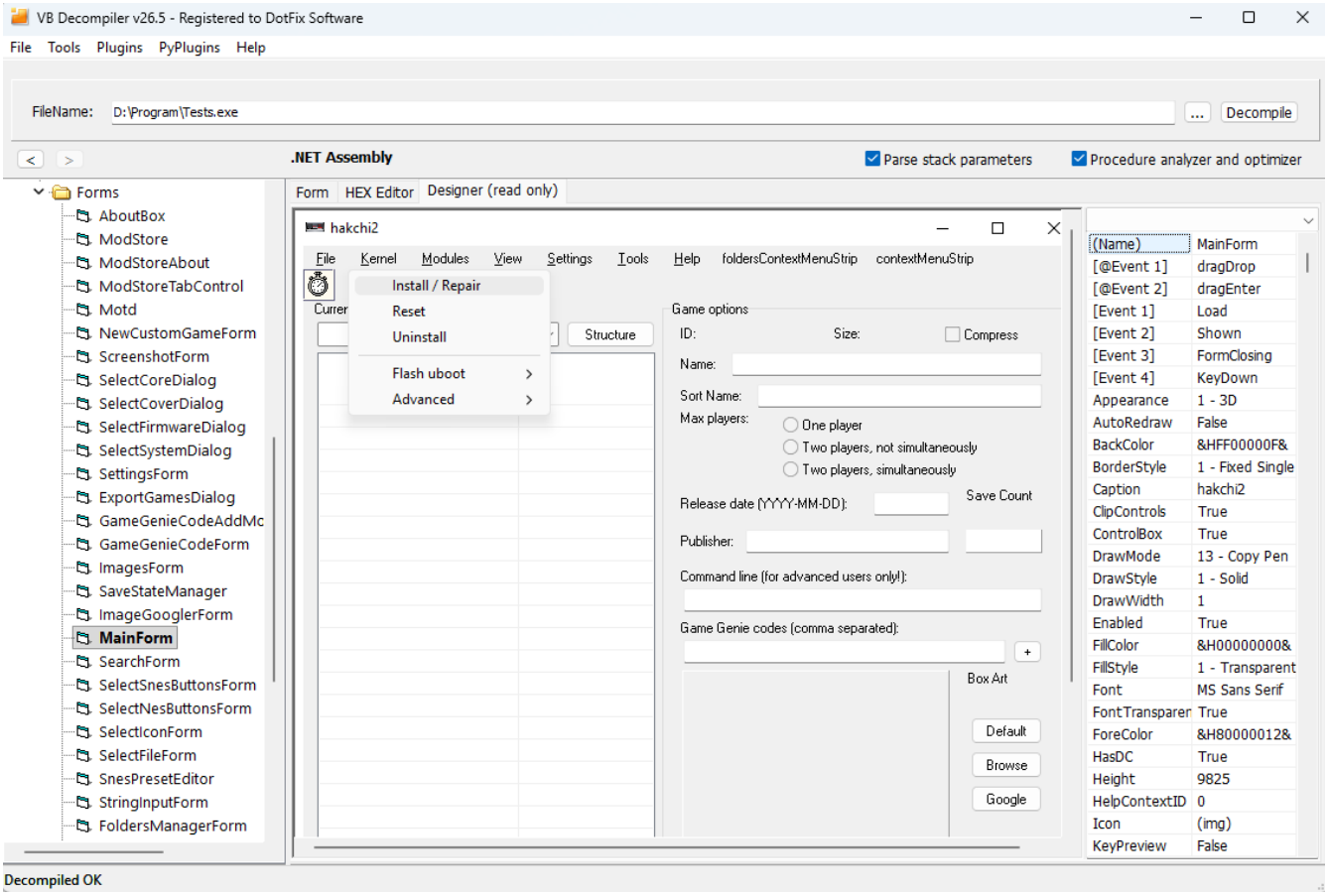


.NET Windows Forms

VB Decompiler статически анализирует `InitializeComponent` и восстанавливает поддерживаемые элементы управления Windows Forms и их компоновку. Разбор и визуализация не требуют среды выполнения .NET или запуска целевого приложения. Поддерживаемые восстановленные изображения также могут отображаться на элементах управления и фонах форм.

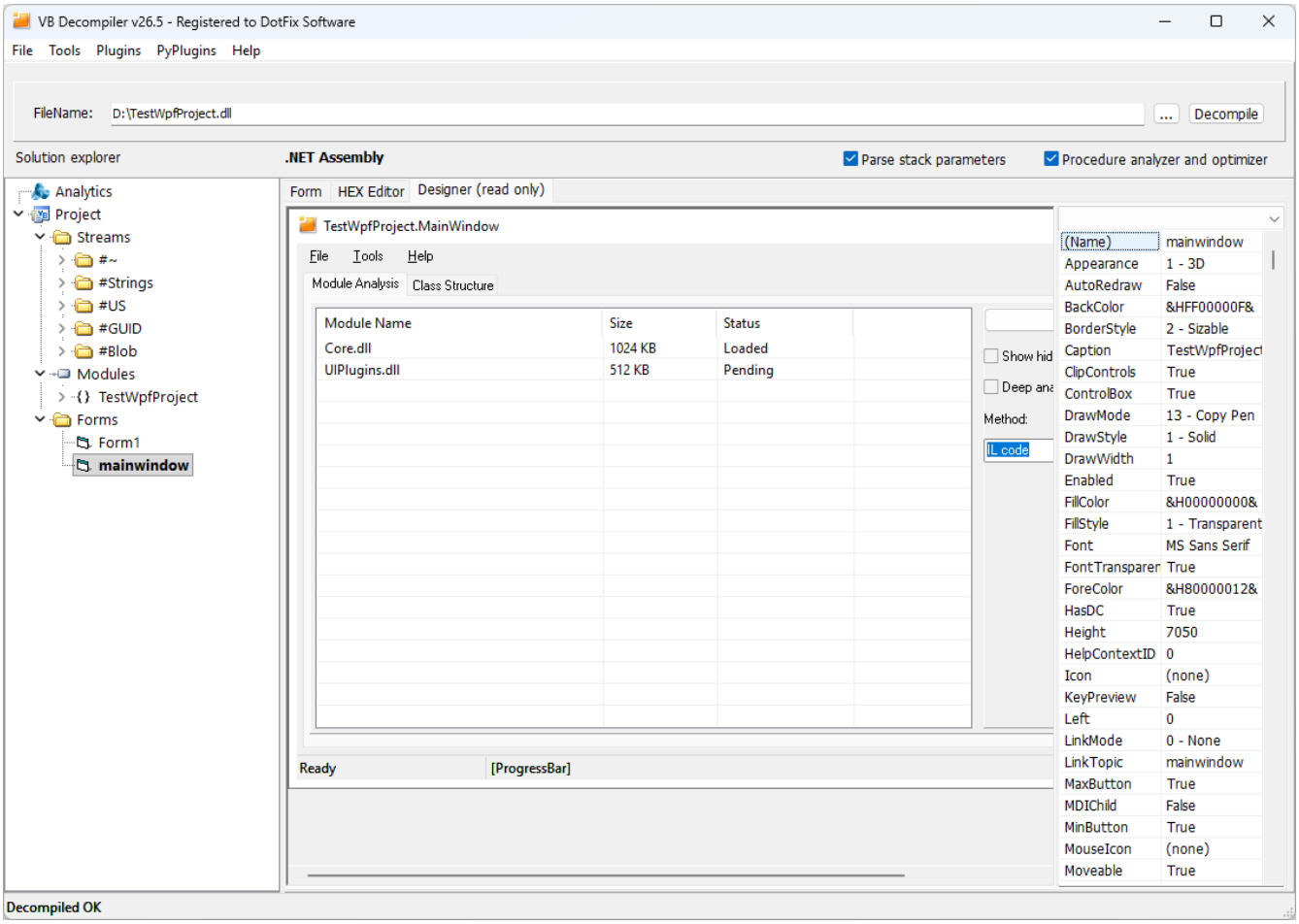
Выберите элемент управления для проверки его восстановленных событий, затем дважды щелкните событие для открытия соответствующего обработчика. Просмотрщик восстанавливает иерархии меню и контекстных меню; щелчок по восстановленному элементу меню открывает соответствующее событие Click. Пользовательские события Windows Forms отображаются с префиксом **@**.

Просмотрщик обрабатывает Z-порядок элементов управления, наследуемые цвета текста и фона, значения `Color.FromArgb`, числовые флаги `FontStyle`, свойства ресурсов типа Boolean и `MinimumSize` наряду с `ClientSize`.



Окна WPF, страницы и пользовательские элементы управления

Просмотрщик восстанавливает поддерживаемые окна WPF, страницы и пользовательские элементы управления из скомпилированных ресурсов XAML, хранящихся как **BAML**.



Поддерживаемые контейнеры компоновки включают **Grid** с фиксированными, Auto и пропорциональными (*) размерами строк и столбцов, **StackPanel**, **DockPanel** и **Canvas**, включая вложенные компоновки. При восстановлении учитываются поддерживаемые размеры, отступы, внутренние поля, выравнивание, ориентацию, закрепление и объединение строк и столбцов Grid.

Помимо обычных элементов ввода, просмотрщик может восстанавливать вкладки, группы, сворачиваемые панели, иерархии меню и разделители, панели состояния, столбцы ListView и поддерживаемые статические данные, узлы статического TreeView и элементы выпадающего списка. Интерфейсы Ribbon, панели навигации и закрепляемые панели могут иметь упрощенные представления.

Если доступны данные ресурсов, просмотрщик восстанавливает имена, видимый текст, подсказки, шрифты, цвета фона, состояния доступности и видимости, а также встроенные изображения, включая поддерживаемую прозрачность PNG и непрозрачность фона. Неподдерживаемые пользовательские элементы управления могут отображаться в виде заглушек с указанием исходного типа WPF.

Просмотрщик WPF связывает восстановленные события элементов управления с их обработчиками. Используйте сведения о событиях в визуальном представлении, чтобы открыть соответствующий декомпилированный код.

Ограничения восстановления: визуальный предпросмотр является приблизительным. Шаблоны, стили, привязки, пользовательские элементы управления и динамически генерируемый контент могут быть представлены частично.

Проверка и извлечение данных форм

См. Формы для настройки «Показывать смещения» (Show offsets) и связанной проверки двоичного файла. Используйте Плагин FRX Image Viewer для проверки и экспорта поддерживаемых восстановленных графических ресурсов.

11 Трассировка

Функции **Tracing features** обеспечивают пошаговую эмуляцию Native Code и P-Code Visual Basic 5.0/6.0. Для их использования требуется лицензия с функциями трассировки. Трассировщик работает с внутренним представлением процедуры в эмуляторе; он не выполняет целевую процедуру на процессоре рабочей станции.

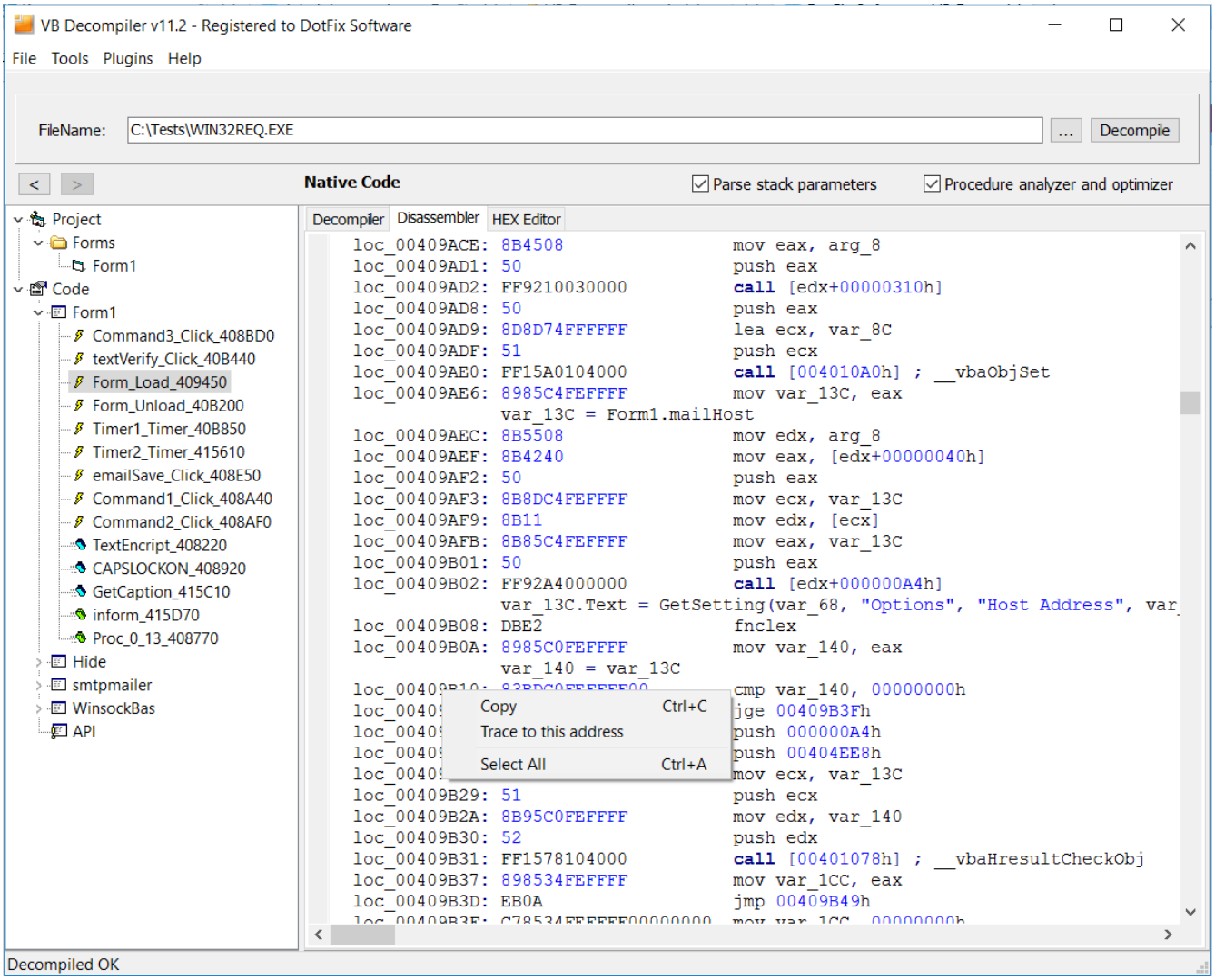
Запуск трассировки

1 Откройте процедуру VB5/6 и оставьте отображение адресов в листинге включенным.

2 На интересующей строке вызовите **«Трассировать до этого адреса» (Trace to this address)** из контекстного меню кода или нажмите **F4**.

3 Изучите код и состояние трассировщика. Для пошагового продолжения нажимайте **«Далее» (Next)**. Завершите трассировку после исследования.

С версии 26.4 команда может определять адрес `loc_` после удаления пробелов в начале строки. Вы можете поместить курсор в другое место на этой строке, а не точно на метку адреса.



Состояние при трассировке Native Code

Трассировщик отображает регистры CPU и FPU, стек, переменные и восстановленный код по мере выполнения эмуляции. Используйте эти представления для изучения того, как было получено значение или выражение. Поиск в списке переменных и отображение восстановленного типа текущего объекта могут помочь с большими процедурами.

Трассировка P-Code

Трассировка P-Code, представленная в версии 11, показывает, как в ходе обработки изменяются сведения о переменных и формируются декомпилированные операторы. Она может помочь объяснить различия между промежуточными операциями P-Code и оптимизированным высокоуровневым листингом.

Область действия и ограничения

- Трассировка применяется к отдельной процедуре. Она начинается с пустого состояния эмулируемых регистров, стека и переменных, а не с состояния запущенного процесса.
- Неподдерживаемые инструкции Native Code могут быть пропущены. Вызовы внешних функций не выполняются; обработка поддерживаемых соглашений о вызовах может корректировать моделируемый стек без воспроизведения поведения внешней функции.
- Трассировщик не отображает полное состояние флагов процессора. Не интерпретируйте его состояние как полную аппаратную трассу выполнения.
- Трассировка выполняется как прямая пошаговая эмуляция, без обратной трассировки или повторного выполнения итераций циклов.
- Описанные функции трассировки относятся к VB5/6. Возможность дизассемблирования .NET не означает наличия отладчика для выполнения кода .NET.

Используйте трассировку, чтобы исследовать промежуточные значения и понять, как декомпилятор восстанавливает код. Значение, полученное при эмуляции, само по себе не определяет результат выполнения реального процесса с инициализированными глобальными переменными, внешними зависимостями и входными данными.

12 Аналитические возможности

Функции **Analytic features**, доступные по соответствующей лицензии, формируют отчёты о распознанных операциях в анализируемой программе, группируя их по категориям. Эти отчеты помогают аналитикам безопасности выявлять процедуры, требующие дальнейшего исследования, до чтения всего кода. Поддерживаются приложения Visual Basic 5.0/6.0 и, начиная с версии 11.7, сборки .NET.

Формирование отчёта

1 Включите **«Анализ прототипов» (Analyze prototypes)** в **«Утилиты > Опции» (Tools > Options)**.

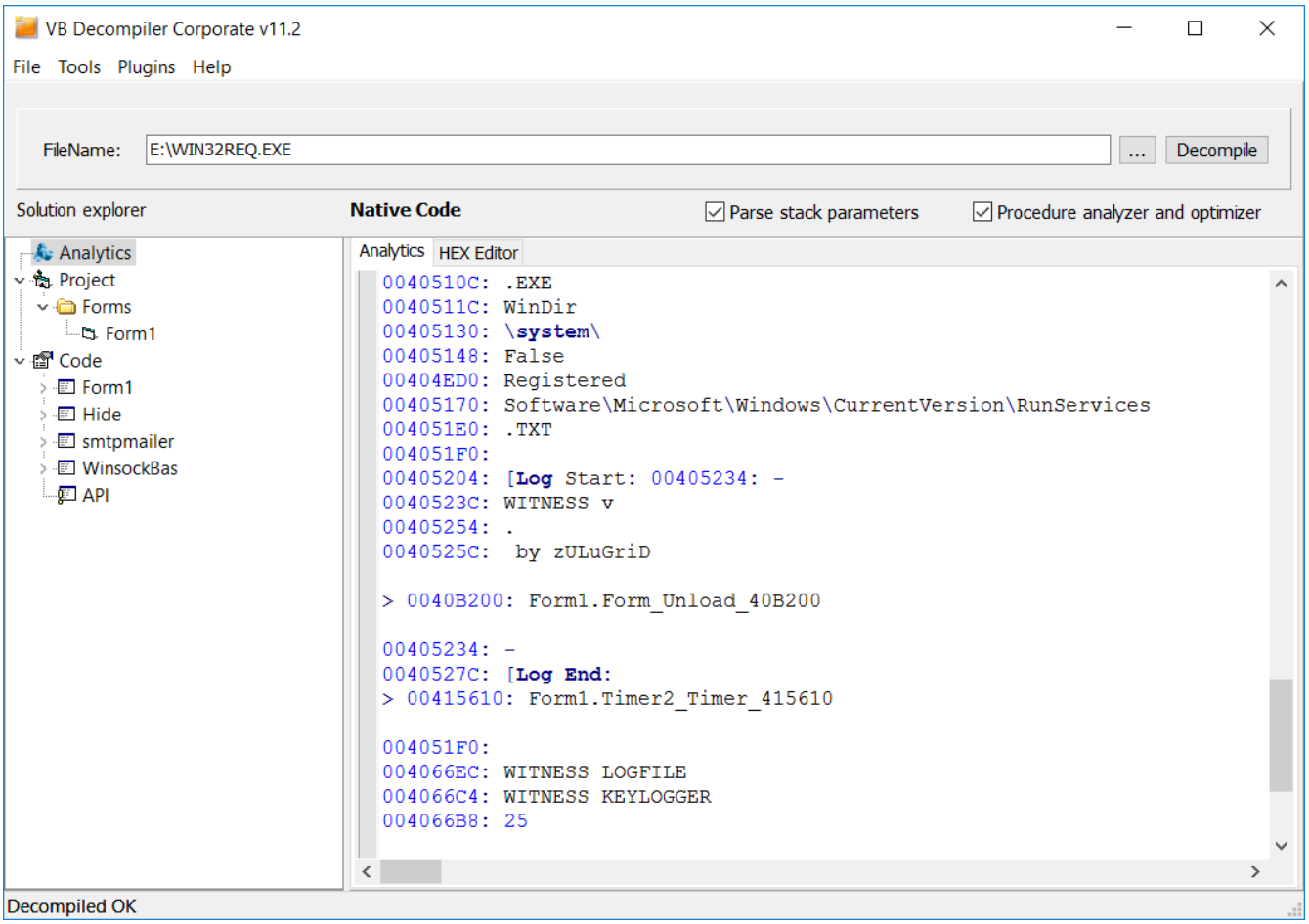
2 Откройте файл и дождитесь завершения анализа. При наличии требуемой лицензии и настроек отчёт формируется и отображается автоматически.

3 Изучите категории, затем откройте соответствующие процедуры и проверьте их код и ссылки на строки.

Категории охватывают распознанные операции, такие как доступ к файлам и томам, доступ к реестру, сетевая активность, создание процессов, работа с окнами и некоторые встроенные функции VB, такие как `CallByName` и `VarPtr`.

Быстрый отчёт и связанные строки

Быстрый отчет перечисляет адреса и имена процедур, содержащих распознанные операции. Расширенный отчет добавляет доступные ссылки на строки из этих процедур. Эти сведения помогают обнаружить возможные пути, ключи реестра, имена хостов или другие значения, заслуживающие изучения.



Строки, которые зашифрованы, обфусцированы или генерируются во время выполнения, могут отсутствовать. Связанная с процедурой строка помогает понять её назначение. Однако сама по себе эта связь не доказывает, что строка передаётся конкретной функции API или что соответствующая ветвь кода выполняется.

Навигация и сохранение результатов

Дважды щелкните имя процедуры в отчете, чтобы открыть ее код. Дважды щелкните адрес, предшествующий процедуре или строке, чтобы проверить его расположение в HEX-редакторе. Кнопка **A** над деревом объектов возвращает к прежней позиции в отчёте. Стрелки навигации и клавиша **Esc** позволяют вернуться к ранее просмотренным процедурам.

Нужный фрагмент отчёта можно выделить и скопировать. Для сохранения всего отчёта выберите **«Файл > Сохранить аналитический отчет» (File > Save analytics report)**.

Интерпретация результатов

Отчёт содержит статически распознанные шаблоны кода и ссылки. Это средство предварительного анализа, а не запись наблюдаемого выполнения или автоматический вердикт о том, что файл вредоносен. Косвенные вызовы, нераспознанные внешние интерфейсы, поврежденные метаданные, обфускация и значения, генерируемые во время выполнения, могут ограничивать полноту анализа. Подтверждайте значимые результаты по коду соответствующих процедур, дизассемблированным инструкциям и данным файла.

Чтобы подробнее изучить восстановление кода VB5/6, см. Трассировка. Для строковых литералов и их расположения в коде см. Ссылки на строки.

13 Ограничения обфускации

VB Decompiler включает обфускатор, который скрывает служебные данные, оставленные компилятором. Он разработан на основе исследований по декомпиляции **приложений Visual Basic 5.0 и 6.0**. Современные языки программирования часто хранят значительные объёмы метаданных и информации для отладки внутри скомпилированных файлов, что облегчает взломщикам поиск нужных функций в вашем приложении. Для противодействия этой угрозе мы разработали обфускатор, специально предназначенный для приложений **Visual Basic**.

Как работает обфускация

Обфускатор работает за счёт разбора структуры программы, включая формы, модули, классы и элементы управления, такие как кнопки и текстовые поля. Для каждой формы определяются её элементы управления и связанные события. Далее собирается информация о функциях и процедурах, а также о событиях, связанных с каждым объектом в программе. В ходе этого процесса обфускатор проверяет зависимости между объектами и согласованно изменяет их имена, чтобы затруднить определение того, к какому модулю или классу относится та или иная функция.

В результате поиск конкретных форм, таких как **frmTrial** или **frmRegistration**, внутри программы становится значительно сложнее.

Ограничения обфускации приложений Visual Basic 5.0/6.0

Обфускатор учитывает зависимости между объектами, однако некоторые способы обращения к ним не поддерживаются. В таких случаях перед обфускацией необходимо изменить исходную программу. Приведённые ниже пункты относятся исключительно к обфускации приложений Visual Basic 5.0 / 6.0.

1. Обфускатор не поддерживает ссылки на другие объекты, заданные в конструкторе форм, что может привести к проблемам во время выполнения.

Например, рассмотрите сценарий, в котором у вас есть элемент управления **TreeView**, который зависит от связанного с ним **ImageList**. Если имя связанного объекта **ImageList** сохранено в свойствах вашего элемента управления **TreeView**, любые попытки обфускации приведут к ошибкам, поскольку ссылка сохраняет исходное имя объекта. Это ограничение также распространяется на компоненты, связанные с базами данных, такие как **DataSets**

```
Begin ComctlLib.TreeView TreeView1
  Height          =   1932
  Left            =   3600
  TabIndex        =   3
  Top             =   1320
  Width           =   2412
  _ExtentX        =   4255
  _ExtentY        =   3408
  _Version        =   327682
  Style           =   7
  ImageList       =   "ImageList1"
  Appearance      =   1
End
```

Чтобы избежать этой проблемы, уберите такую связь из свойств, заданных в конструкторе, и установите её программно в обработчике **Form_Load**:

```
Private Sub Form_Load()
  TreeView1.ImageList = ImageList1
End Sub
```

Аналогично измените все такие связи. В большинстве проектов их немного.

2. Не поддерживаются объявления объектов через As New (вызовы LateCall)

В проектах VB6 это ограничение касается, в частности, объявления автоматически создаваемого экземпляра формы через As New:

```
Private Sub Form_Load()
  Dim oForm as New frmMain
  oForm.Show
End Sub
```

В таких случаях компилятор VB6 вставляет имя **frmMain** непосредственно в код функции **Form_Load**. Поскольку наш обфускатор изменяет только данные, а не код, это имя в коде остаётся неизменным после обфускации. К сожалению, это приводит к завершению программы с ошибкой.

Для поддержки таких вызовов обфускатору **VB Decompiler** потребовалось бы дизассемблировать и декомпилировать все функции программы. Несмотря на наличие такой возможности, полная декомпиляция требует значительного времени. В крупных проектах один только анализ всего кода может занять более получаса. Кроме того, нам также необходимо анализировать ссылки на объекты и переименовывать их, что добавляет ещё больше времени обработки.

Учитывая эти технические ограничения, мы решили не реализовывать полную декомпиляцию и анализ кода в обфускаторе **VB Decompiler**, особенно учитывая, что это решило бы проблему лишь в редких случаях для приложений Visual Basic 5.0 / 6.0 и не принесло бы никакой пользы большинству клиентов.

14 Техническая поддержка

Если у вас возникли вопросы по программе, и вы не можете найти ответы в этом руководстве, пожалуйста, обратитесь в нашу службу технической поддержки. Этот вариант доступен только зарегистрированным пользователям. Для незарегистрированных пользователей мы с радостью поможем с вопросами, касающимися покупки лицензии для **VB Decompiler**.

Кроме того, если вы столкнулись с ошибками в программе или у вас есть предложения по её улучшению, сообщите об этом в службу технической поддержки.

15 Регистрация

Для использования полного функционала **VB Decompiler** необходимо приобрести **Business License**. При покупке Лицензии для одиночного разработчика или Корпоративной лицензии вы получаете возможность:

1. Восстанавливать около 85% кода в типичных поддерживаемых приложениях **Visual Basic 5.0/6.0 P-Code**;
2. Восстанавливать до 85% кода в типичных поддерживаемых приложениях **Visual Basic 5.0/6.0 Native Code**;
3. Декомпилировать типичные поддерживаемые необфусцированные сборки **.NET** в код, близкий к **C#**, с восстановлением до 95%;
4. Получать обновления программы бесплатно в течение одного года после покупки лицензии;
5. Получать техническую поддержку по электронной почте в течение одного года после покупки лицензии;
6. Поддерживать разработку этого декомпилятора.

Кроме того, мы рекомендуем рассмотреть покупку специальных лицензий для аналитиков вредоносного ПО и цифровой криминалистики:

1. лицензия с функциями трассировки
2. лицензия с аналитическими функциями

Для покупки **лицензии** декомпилятора прямо сейчас выберите опцию **«Купить Pro версию» (Order Pro version)** в меню **«Справка» (Help)** программы. В данный момент вы должны быть подключены к интернету. Откроется окно браузера на веб-сайте программы, где можно купить лицензию.

Business License декомпилятора предназначена для использования только на одном компьютере и только одним разработчиком — владельцем лицензии **Single Developer**. Если вам необходимо использовать программу на нескольких компьютерах, вам потребуется приобрести отдельную лицензию для каждого или рассмотреть покупку **Business Enterprise license** (до 20 компьютеров).

16 О разработчике

Декомпилятор разрабатывается Чубченко Сергеем Георгиевичем (**DotFix Software**).

Сайт продукта: www.vb-decompiler.org
Электронная почта технической поддержки: admin@vb-decompiler.org