На правах рукописи

Михайлов Андрей Анатольевич

Методы декомпиляции объектного кода Delphi

05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Иркутск – 2017

Работа выполнена в Федеральном государственном бюджетном учреждении науки Институте динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук (ИДСТУ СО РАН)

Научный руководитель:	кандидат технических наук, доцент, **Хмельнов Алексей Евгеньевич,** ИДСТУ СО РАН, первый заместитель директора по информатизации
Официальные оппоненты:	чл.-к. РАН, доктор физико-математических наук, профессор **Аветисян Арутюн Ишханович**, ИСП РАН, директор
	кандидат технических наук, **Дордопуло Алексей Игоревич**, НИЦ Супер ЭВМ и нейрокомпьютеров, начальник отдела
Ведущая организация:	**Институт систем информатики им. А.П. Ершова СО РАН (г. Новосибирск)**

Защита состоится 12 декабря 2017 г. в 14:00 часов на заседании диссертационного совета Д 003.021.01 при Федеральном государственном бюджетном учреждении науки Институте динамики систем и теории управления имени В.М. Матросова СО РАН, расположенном по адресу: г. Иркутск, ул. Лермонтова, д. 134.

С диссертацией можно ознакомиться в библиотеке и на официальном сайте www.idstu.irk.ru ИДСТУ СО РАН.

Отзывы и замечания по автореферату в двух экземплярах, заверенные печатью, просьба высылать по вышеуказанному адресу на имя ученого секретаря диссертационного совета.

Ученый секретарь
диссертационного совета,

к.ф.-м.н., доцент  Груздева Т.В.

Общая характеристика работы

Актуальность темы исследования. Для разработки большинства сложных программных систем часто используются готовые компоненты, предоставляемые в виде скомпилированных модулей. Такой подход существенно сокращает время и стоимость создания программного обеспечения. С другой стороны, наличие сторонних модулей уменьшает надежность программного обеспечения и его информационную безопасность из-за возможного наличия уязвимостей, способствующих успешным атакам на информационную систему. Кроме того, сторонние компоненты могут содержать ошибки, устранение которых может быть затруднено из-за невозможности связаться с разработчиком, утраты разработчиком исходных кодов и т.д. В некоторых случаях может потребоваться доработка сторонних модулей, исходные тексты которых отсутствуют.

Среди объектных файлов особое место занимают файлы DCU, используемые компиляторами различных версий Delphi. С одной стороны, такие файлы технически можно отнести к объектным файлам, поскольку в дальнейшем с использованием редактора связей из них собирается загрузочный модуль. С другой стороны, файлы DCU содержат больше сведений, чем типичные объектные файлы. Файл DCU может полностью заменить исходный текст для той версии компилятора, при помощи которой он был создан. Этой особенностью активно пользуются разработчики программных модулей, которые часто распространяют их в формате DCU без предоставления исходных текстов, в особенности тогда, когда это делается на коммерческой основе.

В том случае, когда разработчик прекращает развитие своих программных модулей, отсутствие исходных текстов не позволяет применить эти модули с новыми версиями компилятора. Также становится невозможным исправить обнаруженные ошибки, проанализировать качество кода модуля, не говоря уже о его доработке. В связи с этим проблема разработки методов декомпиляции объектных файлов Delphi является актуальной.

Цель диссертационного исследования – разработка методов декомпиляции и анализа объектного кода Delphi.

Для достижения поставленной цели были решены следующие **задачи**:

1. Проведен анализ существующих методов декомпиляции объектного кода.
2. Впервые разработан метод декомпиляции объектного кода Delphi.
3. Реализовано инструментальное средство для анализа и декомпиляции объектного кода Delphi.
4. Проведена апробация созданной технологии на задачах автоматизации анализа объектного кода Delphi.

Научная новизна полученных в диссертации результатов состоит в следующем:

1. Впервые разработан метод декомпиляции объектного кода Delphi, скомпилированного под платформу .NET, позволяющий восстанавливать программу на языке CIL в программу на языке Delphi, включающий следующие этапы: синтаксический анализ объектного кода Delphi, генерация уграфа, генерации промежуточного представления, структурирование графа потоков управления, анализ потоков данных, улучшение и генерации кода Delphi.

2. На основе предложенных методов реализован оригинальный декомпилятор объектных файлов Delphi, скомпилированных под платформу .NET.

3. Разработан оригинальный метод визуализации управляющего графа на плоскости. Основной особенностью разработанного метода визуализации является возможность использования изобразительных соглашений, принятых при проектировании блок-схем, что позволяет эффективнее визуально выделять в графе узлы, соответствующие высокоуровневым операторам языков программирования.

Теоретическая и практическая значимость. Разработанные в рамках диссертационной работы методы и инструментальное средство позволяют повысить эффективность анализа исполняемого кода за счет снижения трудозатрат, сокращения времени анализа, а также повышения наглядности представления результатов. Практическая значимость результатов связана с возможностью их применения для поддержки, анализа и изучения унаследованного программного обеспечения, имеющего в своем составе компоненты, представленные в виде скомпилированных модулей Delphi. Материалы диссертации могут быть использованы при разработке спецкурсов для студентов математиков, а также при написании курсовых и дипломных работ, магистерских диссертаций.

Созданное программное обеспечение зарегистрировано в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам (№2014617137, №2016612670).

Отдельные результаты диссертационной работы получены в рамках проекта СО РАН IV.38.2.3. «Новые методы, технологии и сервисы обработки пространственных и тематических данных, основанные на декларативных спецификациях и знаниях», а также научного проекта РФФИ № 15-37-20042 мол_а_вед.

Соответствие специальности. В соответствии с паспортом специальности 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей диссертационная работа охватывает решение задач повышения эффективности процессов анализа, сопровождения и создания программ и программных систем, в частности, унаследованного программного обеспечения, имеющего в своем составе компоненты, представленные в виде скомпилированных модулей Delphi. Отраженные в диссертации положения соответствуют пунктам 1, 2, 7 и 10 области исследований.

Методология и методы исследования. Для решения поставленной задачи в данной работе использованы методы теории множеств, теории компиляторов,

теории графов, понятия и методы теории сложности.

Результаты, выносимые на защиту:

1. Метод декомпиляции объектного кода Delphi, скомпилированного под платформу .NET.

2. Программная реализация декомпилятора объектных файлов Delphi скомпилированных под платформу .NET, позволяющего восстанавливать программы на низкоуровневом языке CIL в программы на языке Delphi.

3. Метод визуализации управляющего графа на плоскости, позволяющий использовать изобразительные соглашения, принятые при проектировании блок-схем.

Степень достоверности и апробация результатов обеспечивается обоснованным использованием методов и технологий декомпиляции информационных систем и визуализации уграфов, опубликованных в открытой печати; согласованностью с результатами исследований других авторов, представленных в печатных изданиях; работоспособностью разработанного декомпилятора и модуля визуализации графов, адекватностью полученных данных в результате их тестирования и сравнением с аналогичными средствами.

Основные результаты диссертации и её отдельные положения, а также результаты конкретных прикладных исследований и разработок, обсуждались на научных семинарах ИДСТУ СО РАН, ИСИ СО РАН, ИСП РАН, докладывались на отечественных и международных научных конференциях: The 39th International ICT Convention – MIPRO (г. Опатия, Хорватия, 2016 г.); 5th International Workshop on Computer Science and Engineering (г. Москва, 2015 г.); III Российско-монгольской конференции молодых ученых по математическому моделированию, вычислительно-информационным технологиям и управлению (п. Ханх, Монголия, 2015 г.); «Ляпуновские чтения» (г. Иркутск, 2012, 2013, 2014 гг.); «Малые Винеровские чтения» (г. Иркутск, 2013); XVIII Байкальская Всероссийская конференция «Информационные и математические технологии в науке и управлении» (г. Иркутск, 2013 г.); II Российско-Монгольской конференции молодых ученых (п. Ханх, Монголия, 2013 г.); XIII Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям (г. Новосибирск, 2012 г.).

Публикации. Материалы диссертации опубликованы в 16 печатных работах [1–16], из них 3 статьи в журналах, рекомендованных ВАК РФ для опубликования результатов диссертаций [1–3], 1 статья из списка WOS [4], 2 авторских свидетельства [15, 16].

Личный вклад автора. Все выносимые на защиту научные положения получены соискателем лично. В основных научных работах по теме диссертации, опубликованных в соавторстве, лично соискателем разработаны: методы декомпиляции объектного кода Delphi, скомпилированного под платформу .NET [2, 5,

6, 8, 9, 12]; программная реализация декомпилятора объектных файлов Delphi, скомпилированных под платформу .NET [3, 13–15]; метод визуализации управляющего графа на плоскости [1, 4, 7, 10, 11, 16].

Структура и объем диссертации. Диссертация состоит из введения, четырёх глав, заключения и библиографии. Общий объём диссертации 155 страниц, из них 108 страниц текста, включая 19 рисунков. Библиография включает 98 наименований на 10 страницах.

Содержание работы

Во введении обоснована актуальность диссертационной работы, сформулирована цель и аргументирована научная новизна исследований, показана практическая значимость полученных результатов, представлены выносимые на защиту научные положения.

Первая глава посвящена изучению задач декомпиляции как составной задачи обратной инженерии.

Декомпиляция – это процесс автоматического восстановления программы на языке высокого уровня из программы на языке низкого уровня.

Декомпилятор – это программа, получающая на вход программу на языке низкого уровня и выдающая на выход эквивалентную ей программу на некотором языке высокого уровня.

В общем случае для произвольного исполняемого файла задача декомпиляции чрезвычайно сложна в силу того, что в процессе компиляции утрачивается много информации о высокоуровневой программе. Часть информации утрачивается безвозвратно: комментарии; имена импортируемых модулей, подпрограмм, переменных, констант; сведения о типах данных. Программа на языке низкого уровня представляет собой последовательность инструкций с условными и безусловными переходами, в которой отсутствует информация об используемых типах данных, высокоуровневых операторах и т д. Однако некоторые виды утраченной информации можно частично или полностью восстановить. В то же время существует ряд в общем случае алгоритмически неразрешимых проблем, таких как отделение кода от данных, отделение констант от адресов и т. д.

Исходя из разницы уровня абстракции исходного и скомпилированного кода, все существующие декомпиляторы можно разделить по уровню представления программы, которое подается им на входе:

- *Декомпиляторы машинного кода исполняемых файлов и загрузочных модулей.* Декомпиляторы данного типа решают самые сложные задачи обратной инженерии, потому что в процессе компиляции утрачивается вся информация, которая не требуется процессору для исполнения. На текущий момент автору не известно успешных реализаций декомпиляторов данного типа. Из всех рассмотренных в данной работе инструментов (Boomerang, Dcc, REC,

Hex-Rays, SmartDec) самый современный и единственный применимый на практике – это плагин к интерактивному дизассемблеру IDA Pro Hex-Rays.

- *Декомпиляторы объектного кода.* В объектном коде присутствует дополнительная информация, предназначенная для работы редактора связей. Такая информация может включать в себя имена переменных, описания процедур и функций, импортируемые типы данных и т. д. Наличие такой информации может упростить процесс декомпиляции, тем самым повысив её качество.

- *Декомпиляторы байт-кода виртуальных машин.* Декомпиляторы данного типа являются наиболее успешными примерами в области обратной инженерии. Например, имея скомпилированную программу, написанную на языке программирования C#, можно восстановить ее исходный код с помощью декомпилятора ILSpy, который практически всегда с точки зрения семантики будет соответствовать исходной программе. Качество работы декомпиляторов данного типа объясняется тем, что на вход они получают файлы, содержащие дополнительные метаданные.

Высокоуровневые языки программирования одного типа предоставляют примерно схожие возможности. Однако иногда существуют различия, и в таких случаях трансляторам из одного языка высокого уровня в другой приходится моделировать код исходного языка средствами целевого. В процессе такого преобразования могут появляться артефакты трансляции, которые существенно затрудняют анализ. Таким образом задача декомпиляции становится не проще задачи трансляции.

Рассмотрим некоторые особенности языка Delphi и объектных файлов DCU в отличие от языков, компилирующихся в .NET:

- В процессе обработки объектных модулей Delphi линкером теряется информация о именах локальных переменных. Такая информация зачастую является очень ценной, поскольку позволяет специалисту быстрее понять назначение переменной, избавляя от необходимости изучать весь исходный код.

- Язык Delphi позволяет использовать вложенные процедуры и функции. Эта информация теряется в процессе компиляции, при этом вложенная процедура преобразуется в internal метод.

- Delphi хоть и является объектно ориентированным языком программирования, но позволят писать программы в процедурном стиле. Например, в отличие от C#, в нём присутствуют не только методы, а также процедуры и функции. К тому же функцию можно вызвать как процедуру.

- Delphi и C# поддерживают использование значений параметров по умолчанию. Вызовы методов с такими параметрами заменяются компилятором и эта информация уже не содержится в исполняемом файле. В объектных файлах Delphi такая информация присутствует.

- В Delphi классы и объекты – это разные типы данных. При компиляции в код виртуальной машины .NET эта информация также теряется, в отличие от объектного файла DCU.

Таким образом имеет смысл рассматривать задачу декомпиляции объектных файлов Delphi, а не исполняемых файлов, полученных в результате обработки их линкером, из-за того, что в объектных файлах содержится больше информации о исходной программе: имена локальных переменных, интерфейсы процедур и функций, объявление глобальных переменных и констант и т. д. Такая информация позволяет существенно повысить качество декомпиляции по сравнению с известными декомпиляторами .NET (ILSpy, NET Reflector) и в большинстве случаях избежать артефактов трансляции из-за несоответствия объектной структуры виртуальной машины с исходным языком программ Delphi.

Во второй главе рассмотрены задачи, которые необходимо решить для реализации декомпилятора объектных файлов DCUIL. Предлагаются и обосновываются используемые для этого алгоритмы и структуры данных. На рис. 1 представлена схема декомпиляции объектного кода Delphi.

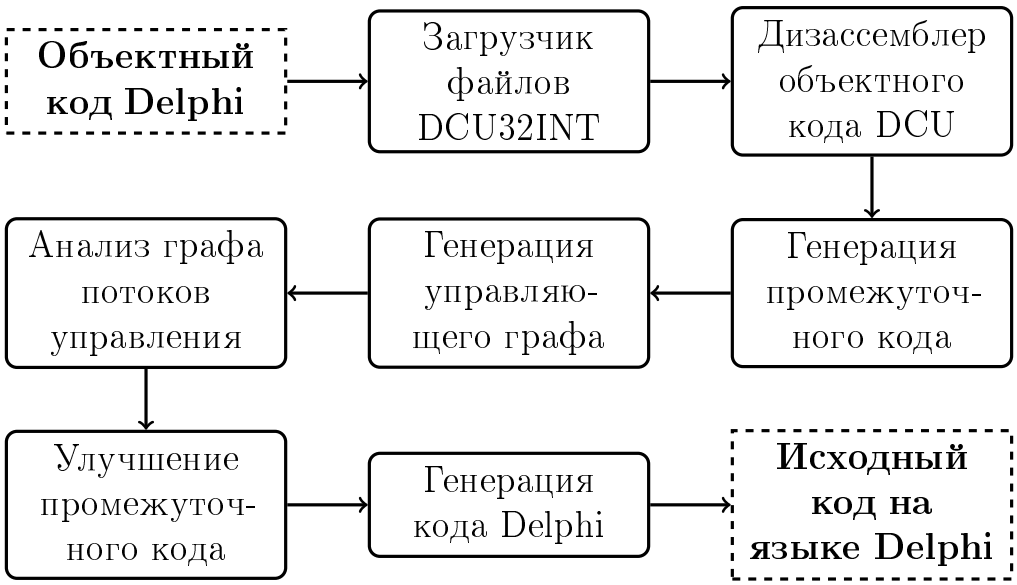


Рис. 1. Схема декомпиляции объектного кода Delphi

В объектных файлах Delphi, в отличие от исполняемого файла в формате PE, программа оказывается более структурированной. Например, выделены блоки памяти, соответствующие коду каждой процедуры; имеется информация о типах данных; может присутствовать отладочная информация. Такой информации нет в обычных исполняемых файлах. В общем виде формат файла скомпилированных модулей Delphi выглядит следующим образом. Сначала идет небольшой заголовок, в котором содержится общая информация о файле, такая как размер, время компиляции и т. д. После заголовка следует поток теговой информации. Для обобщения теги можно разделить на следующие группы: описания включаемых модулей и объектных файлов; импортируемых из этих модулей определений

(типов данных, процедур, и т. д.); описания определений (типов данных, процедур и функций, и т. д.) из данного модуля; блок памяти, составленный из блоков кода для процедур и функций, образов типизированных констант и т. д.; информация для редактора связей (в какие места блока памяти необходимо занести адреса, получаемые из других модулей); отладочная информация.

Таким образом, для реализации декомпилятора объектных файлов DCUIL основные задачи, которые необходимо решить (рис. 1), – это восстановление высокоуровневых операторов, реализация промежуточного представления, генерация кода, а также оптимизации, нацеленные на улучшение результата декомпиляции.

Рассмотрим более подробно задачу восстановления высокоуровневых операторов. На практике в большинстве работ, посвященных декомпиляции, используются два подхода к анализу потока управления отдельных процедур. Первый подход использует дерево доминаторов для поиска *естественных* циклов, и в дальнейшем использует их для оптимизации. Второй подход, называемый *интервальным анализом*, включает методы, которые позволяют анализировать структуру процедуры вцелом и разбивать её на вложенные участки, называемые *интервалами*. Теория интервалов была предложена Алленом[1] в начале 1970-х годов и использовалась для проведения оптимизаций при более тщательном анализе потоков данных. Наиболее глубокий вариант интервального анализа, называемый *структурным анализом*, был предложен Цифуентес[2]. Данный метод классифицирует абсолютно все структуры потока управления в процедуре. На первом этапе метод производит выделение и структурирование циклов. Далее, в порядке, обратном обходу в глубину, на граф накладываются шаблоны, соответствующие высокоуровневым операторам, и с помощью семантически эквивалентных преобразований граф сводится к одной абстрактной вершине, которая содержит в себе всю иерархию вложенных интервалов. В теории компиляции методы анализа потока данных на основе анализа интервалов называются *методами устранения*.

На основе анализа дерева доминирующих вершин разработан алгоритм структурирования управляющего графа, в основе которого лежит предложенный Джонсоном с коллегами[3] в 1994 году метод структурирования управляющего графа путем представления его в виде иерархии SESE-регионов (Single Entry Single Exit). В графе выделяются регионы, имеющие одну входную и одну выходную вершины, которые в дальнейшем классифицируются и заменяются одной абстрактной вершиной. Свертка продолжается до тех пор, пока граф не преобразуется в одну абстрактную вершину, содержащую в себе всю иерархию вложенных программных структур. После завершения преобразования алгоритм производит рекурсивное развертывание абстрактных вершин, применяя заранее предопределённые шаблоны генерации кода в зависимости от типа обрабатываемого региона.

---

[1] Allen F. E. Control flow analysis //ACM Sigplan Notices. ACM, 1970. Т. 5. №. 7. С. 1-19.

[2] Cifuentes C. Structuring decompiled graphs //Compiler Construction. Springer Berlin/Heidelberg, 1996. С. 91-105.

[3] Johnson R., Pearson D., Pingali K. The program structure tree: Computing control regions in linear time //ACM SigPlan Notices. ACM, 1994. Т. 29. №. 6. С. 171-185.

Ниже приведён псевдокод верхнего уровня работы предлагаемого алгоритма 1 структурирования управляющего графа:

Алгоритм 1: Алгоритм структурирования

Исходные параметры: G, D, P
Результат: Абстрактный узел, содержащий в себе иерархию вложенных регионов

```
1  для каждого v из D в обратном порядке выполнять
2      для каждого p ∈ (v) выполнять
3          если p pdom v тогда
4              S ← (v) \ p
5              C ← КлассифицироватьРегион(S)
6              если C ≠ неопределённый тогда
7                  НаложитьШаблон(C, S)
8              конец условия
9              иначе
10                 ВыделитьНеопределённыйРегион(S)
11             конец условия
12             Модифицировать(G, D, P)
13         конец условия
14     конец цикла
15 конец цикла
```

Построим дерево доминаторов D и постдоминаторов P для графа $G(E, V)$.

На каждой итерации алгоритма 1 выделяется регион, имеющий наибольший уровень вложенности и только одну входную и одну выходную вершины (2T-регион). Для этого используется правило $p\ pdom\ v$. Постдоминатор вершины v из множества S является терминальной вершиной, на которой сходится поток управления, прошедший через v.

После того, как выделено множество вершин 2T-региона, он классифицируется. На выделенный подграф последовательно накладываются заранее предопределенные шаблоны. Если регион соответствует одному из этих шаблонов, то он считается *определенным*, иначе *неопределенным*.

Для определенных регионов создается новый абстрактный узел, включающий в себя структуру содержащегося в нём подграфа. Неопределенный регион заменяется новым абстрактным узлом. В конечном счете останется один абстрактный регион, не имеющий входящих и исходящих дуг, который содержит в себе всю иерархию вложенности операторов подпрограммы.

Современные методы декомпиляции в большинстве своем основаны на методах построения компиляторов. Набор применяемых методов зависит от исходного и целевого языка высокого уровня. В данной главе рассмотрены основные методы, применимые к задаче анализа объектного кода Delphi. Рассмотрены методы и алгоритмы анализа потоков данных подпрограмм. На основе рассмотренных методов предложен алгоритм структурирования управляющего графа.

В третьей главе разработана архитектура и изложены основные принципы реализации декомпилятора DCUIL2PAS. На рис. 2 многоугольниками представлены компоненты декомпилятора, а стрелками отображается поток данных между ними.

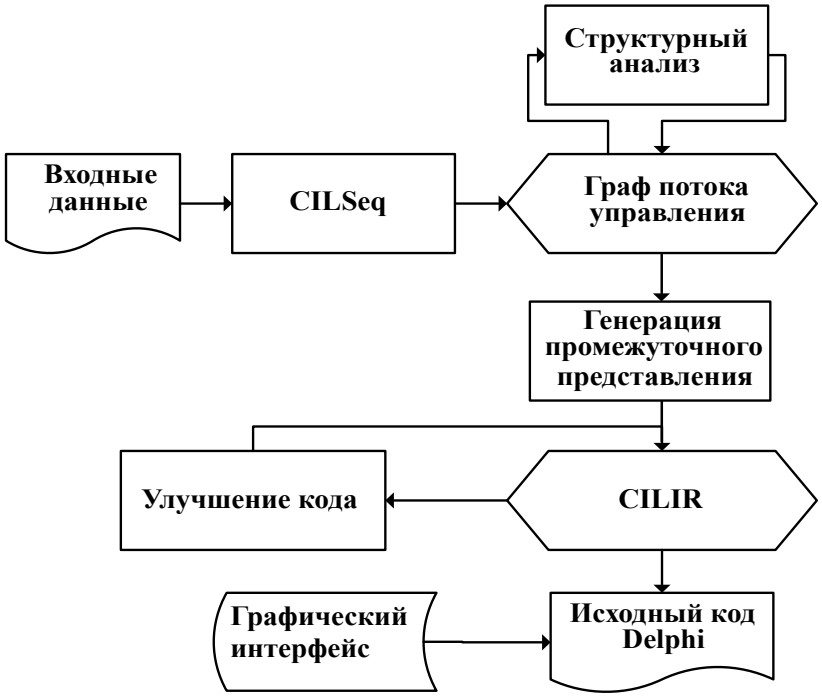


Рис. 2. Архитектура декомпилятора DCUIL2PAS.

CILSeq – это *внутреннее представление* исходной подпрограммы в виде последовательности CIL инструкций.

Модуль *структурного анализа* восстанавливает высокоуровневые конструкции языка посредством модификации графа потоков управления.

Модуль *генерации промежуточного представления* отвечает за генерацию промежуточного кода, который реализован в виде иерархии классов и строится для каждого базового блока графа потоков управления. Сначала определяется начальное состояние каждого блока, которое задаёт значения параметров и аргументов функции, а также состояние стека. Затем каждому CIL коду ставится в соответствие выражение (экземпляр класса), реализующее семантику опкода.

Иерархия классов промежуточного представления показана на рис. 15. Базовый класс `TCILExpr` реализует логику работы со счетчиками ссылок и задает набор обязательных методов: `Eval` – вычисляет значение выражения; `Eq(E: TCILExpr)` – возвращает «true», если переданное выражение эквивалентно текущему; `AsString(BrRq: boolean)` – возвращает текстовое представление выражения; `Show` – выводит на печать текстовое представление выражения.

Наследники класса `TCILUnOp` описывают все опкоды, аргументом которых является один единственный операнд. Аналогично все наследники `TCILBinOP` описывают семантику всех бинарных операций. Аргументы и локальные переменные наследуются от класса `TCILArgs`. Для описания оставшихся неописанных инструк-

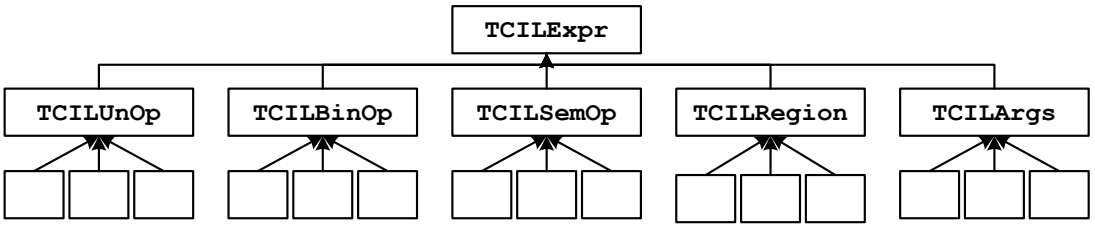


Рис. 3. Иерархия классов промежуточного представления.

ций в качестве базового класса используется `TCILSemOp`.

Модуль *графа потоков управления* выполняет построение уграфа. Все условные и безусловные переходы в процессе разбора преобразуются в выражение вида – `CILCond(Next, Tgt, Cond)`, где `Next` и `Tgt` – это блоки назначения перехода, `Cond` – условие перехода. Все выделяемые регионы (`TCILRegion`) в процессе анализа потоков управления также являются наследниками базового класса `TCILExpr`. Каждому из выделяемых регионов соответствует свой класс, реализующий его семантику. Список классов реализующих регионы в декомпиляторе DCUIL2PAS: `TCILIfThenElse`, `TCILWhile`, `TCILRepeat`, `TCILCaseSt`, `TCILBasicBlock`.

Модуль *улучшения кода* производит некоторые изменения промежуточного представления для улучшения качества генерируемого кода. Одним из важных улучшений является объединение условий операторов ветвления и циклов. Объединение сложных выражений происходит в процессе их вычисления по заранее определенному набору правил:

- $A\ and\ B \Rightarrow if\ A\ then\ B\ else\ False$,
- $A\ or\ B \Rightarrow if\ A\ then\ True\ else\ B$.

Модуль *графического интерфейса* реализует графический интерфейс пользователя с поддержкой подсветки синтаксиса.

Разработанный декомпилятор был протестирован на специально подготовленном наборе процедур, взятых из модификации алгоритма LZW, написанного на языке Delphi. Для сравнения был выбран декомпилятор с открытым исходным кодом ILSpy, для оценки качества – *мера качества декомпиляции*[4], которая выражается формулой (1):

$$C_{decom} = \sum_{prog \in TS} \frac{max(0, K' - K)}{KLOC(prog)}, \tag{1}$$

где TS – тестовый набор программ; $prog$ – исходная программа; $KLOC(prog)$ – количество тысяч значимых строк кода программы prog; K – сумма штрафов исходной программы; K' – сумма штрафов восстановленной исходной программы.

[4] Трошина Е. Н. Исследование и разработка методов декомпиляции программ //М.: Моск. гос. ун-т им. МВ Ломоносова. – 2009.

Штрафы за артефакты трансляции и неполноту восстановления (табл. 1) были изменены в соответствии с требованиями декомпиляции объектного кода Delphi. Подсчет меры качества производился для каждой процедуры отдельно, при этом не учитывалось качество восстановления интерфейсной части модуля.

Таблица 1. Штрафы за артефакты трансляции и неполноту восстановления

Конструкции программы	**Назначаемые штрафы**
невосстановление имени переменной	1
оператор перехода goto	3
оператор выхода из середины цикла break	1
оператор прерывания витка цикла continue	1
невосстановление оператора for	1

Вычисление меры качества декомпиляции представлено в табл. 2. На всех примерах мера качества разработанного декомпилятора оказалась выше, чем у ILSpy. Это связано в первую очередь с тем, что в процессе обработки объектных модулей линкером теряется часть информации о исходной программе, а также с тем, что декомпилятор ILSpy изначально разрабатывался, исходя из соображений, что исходная программа была написана не на языке Delphi.

Помимо сравнительного анализа декомпилятором в пакетном режиме была разобрана стандартная библиотека VCL Delphi 8. Результаты, которые в большей степени демонстрируют производительность работы данного декомпилятора, приведены в табл. 3.

Для оценки качества декомпиляции стандартной библиотеки VCL Delphi 8 в автоматическом режиме было просчитано количество процедур и функций, восстановленных в структурном виде (без использования оператора `goto`). Тестирование показало (табл. 4), что в 98,7% случаях удается восстановить программу без операторов `goto`.

В работе приведён пример использования декомпилятора для восстановления исходного кода модуля программы сжатия LZRW.

Разработанный декомпилятор объектных файлов DCUIL позволяет восстанавливать исходный код на языке Delphi, который в большинстве случаев пригоден для дальнейшей его компиляции и полностью семантически эквивалентен исходному представлению программы. Данное программное средство позволяет существенно сократить время на решение задач, связанных с поддержкой и пе-

Таблица 2. Мера качества

Название	**DCUIL2PAS**	**ILSpy**
BitWise	62,5	133,3
Compression	18,6	146
LZRW1KHCompressor	75	140
GetMatch	0	166,6

Таблица 3. Результаты тестирования производительности

Название	**Кол-во файлов**	**Размер (мб)**	**Время обработки (с)**
Delphi 8 VCL	325	39	396

Таблица 4. Результаты тестирования качества

Название	**Кол-во процедур**	**Без** `goto`	**С** `goto`	**%**
Delphi 8 VCL	9003	8879	124	1,3

реработкой унаследованного и стороннего программного обеспечения, исходные тексты которого не предоставлялись или были утрачены; позволяет находить закладки в готовых модулях и компонентах Delphi под .NET; искать и исправлять ошибки.

На разработанный декомпилятор объектных файлов DCUIL получено авторское свидетельство о государственной регистрации программ для ЭВМ.

Четвертая глава посвящена описанию применения методов декомпиляции в задаче визуализации управляющего графа. Декомпиляция кода, скомпилированного под платформу x86 не всегда выполнима. Для анализа такого кода реализован импорт из программы DCUIL2PAS в формат GRAPHML. На основе описанного во второй главе метода структурирования разработан алгоритм раскладки графа потоков управления на плоскости. Данный алгоритм реализован в виде отдельного раскладчика для системы визуализации сложно структурированной информации большого объема на основе графовых моделей Visual Graph.

Процесс раскладки происходит рекурсивно сверху вниз, начиная с региона верхнего уровня. Для этого региона задаются начальные координаты. Далее, если региону был сопоставлен шаблон, применяются правила отображения, определенные в нем. Приведем пример для шаблона `if-then-else` (рис. 4).

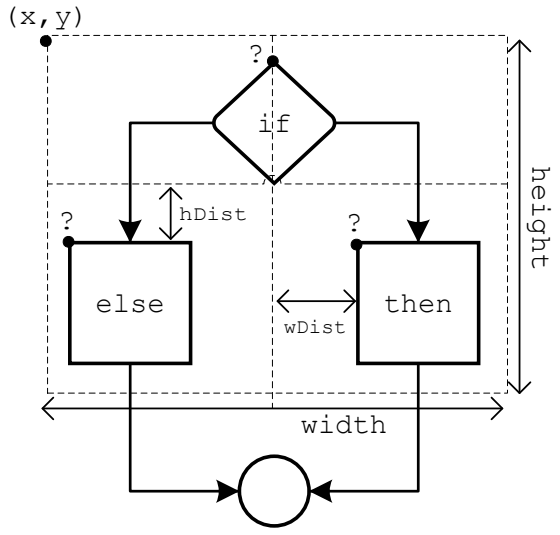
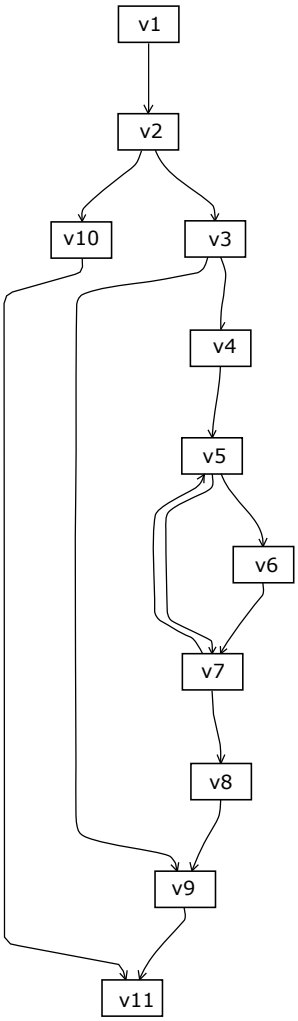


Рис. 4. Шаблон отображения if-then-else

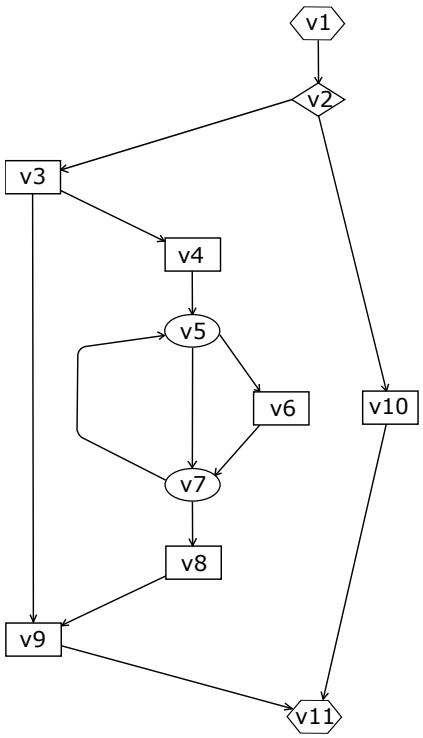
Определим правила вычисления координат для узлов `if`, `then`, `else`:

- `if.x = x + abs(width - if.width) / 2;`
- `then.x = x + wDist / 2, then.y = y + if.height + hDist;`
- `else.x = x - else.width - wDist / 2, else.y = y + if.height + hDist;`

где `wDist` – вертикальное и `hDist` – горизонтальное расстояния между узлами. Данные параметры задаются вручную, исходя из эстетических соображений.



а) Иерархический раскладчик



б) Структурный раскладчик

Рис. 5. Результат раскладки управляющего графа.

Для каждого шаблона аналогичным образом определяются правила визуализации. Для неопределенных регионов используется иерархический раскладчик. Таким образом, процесс раскладки сводится к последовательному применению правил отображения для вложенных регионов.

Рассмотрим результат работы алгоритма в сравнении с результатом раскладки, полученным при помощи стандартного иерархического раскладчика. На рис. 5: б) обратная дуга из $v7$ в $v5$, которая соответствует циклу в графе, изображена специальным образом и явно отличается от других дуг. К тому же данный алгоритм позволяет специальным образом выделять терминальные узлы ($v1$, $v11$,

$v2$, $v5$, $v7$). Изображение дуг и узлов с учётом семантики исходной программы позволяет быстрее визуально выделять в графе наиболее важные участки для их более подробного анализа. При этом остальные дуги, изображены прямой линией, что значительно облегчает анализ путей передачи управления в отличие от результата визуализации иерархического раскладчика рис. 5: а).

Предложен новый подход к раскладке графов потоков управления на плоскости с использованием методов структурного анализа. На основе разработанных методов реализован структурный раскладчик атрибутивных графов потоков управления. Произведено тестирование структурного раскладчика на тестах SPEC CPU2000:

- 197.parser – синтаксический разбор для естественного языка;
- 252.eon – трассировка лучей.

В результате около 76% графов удалось структурировать полностью (не содержат «неопределенных» регионов). Около 96% всех выделенных регионов являются структурными.

В Заключении перечислены основные результаты, полученные в ходе выполнения диссертационной работы, приводится ряд возможных направлений дальнейших исследований.

В приложениях содержатся акт внедрения, подтверждающий практическое применение полученных результатов, фрагменты исходного кода, примеры результатов декомпиляции объектных файлов DCUIL и результатов раскладки графов потоков управления на плоскости.

Список публикаций

1. Михайлов А. А. Анализ графа потоков управления в задаче декомпиляции подпрограмм объектных файлов dcuil // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2014. Т. 12, № 2. С. 74–79.
2. Михайлов А. А. Промежуточное представление подпрограмм в задаче декомпиляции объектных файлов dcuil // Вестник Бурятского государственного университета. 2014. № 9-3. С. 32–38.
3. Хмельнов А. Е., Бычков И. В., Михайлов А. А. Декларативный язык FlexT—инструмент анализа и документирования бинарных форматов данных // Труды института системного программирования РАН. 2016. Т. 28, № 5. С. 239–268.
4. Mikhailov A., Hmelnov A., Cherkashin E. et al. Control flow graph visualization in compiled software engineering // Information and Communication Technology, Electronics and Microelectronics (MIPRO) / IEEE. 2016. P. 1313–1317.

5. Михайлов А. А. Анализ объектных файлов Delphi с использованием спецификации семантики машинных команд // Прикладная дискретная математика. 2012. Т. 5. С. 108–110.
6. Михайлов А. А. Анализ потоков данных подпрограмм в объектных файлах dcu // Материалы конференции «Малые Винеровские чтения». 2013. С. 23–28.
7. Михайлов А. А. Анализ потоков данных подпрограмм в объектных файлах DCU // Тезисы конференции «Ляпуновские чтения». 2012. С. 24.
8. Михайлов А. А. Анализ потоков данных подпрограмм объектных файлов Delphi // Труды XVIII Байкальской Всероссийской конференции "Информационные и математические технологии в науке и управлении". Т. 2. 2013. С. 151–156.
9. Михайлов А. А. Алгоритм анализа потоков данных подпрограмм объектных файлов DCU // Тезисы II Российско-Монгольской конференции молодых ученых. 2013. С. 43.
10. Михайлов А. А. Визуализация управляющего графа // Тезисы доклада III Российско-монгольской конференции молодых ученых по математическому моделированию, вычислительно-информационным технологиям и управлению. 2015. С. 59.
11. Михайлов А. А., Хмельнов А. Е. Анализ программного кода в объектных файлах Delphi, скомпилированных под платформу .NET // Труды конференции «Языки программирования и компиляторы». 2017. С. 202–204.
12. Михайлов А. А. Анализ программного кода объектных файлов Delphi с использованием спецификации семантики машинных команд. 2012. URL: http://conf.nsc.ru/ym2012/ru/reportview/139230 (дата обращения: 2017-09-01).
13. Hmelnov A. E., Mikhaylov A. A., Burlakov A. S. Delphi .NET object file decompiler // Proc. of the 5th International Workshop on Computer Science and Engineering — Russia, Moscow: Bauman Moscow State Technical University. 2015. P. 202–208.
14. Burlakov A. S., Mikhaylov A. A. The Computer Architecture and Hardware Descriptive Language // Proc. of the 5th International Workshop on Computer Science and Engineering — Russia, Moscow: Bauman Moscow State Technical University. 2015. P. 148–154.
15. Михайлов А. А., Хмельнов А. Е. DCUIL2PAS - декомпилятор объектных модулей Delphi, скомпилированных по платформу .NET (файлов *.DCUIL). 2014. Свидетельство о государственной регистрации программ для ЭВМ № 2014617137 М.: Федеральная служба по интеллектуальной собственности, патентам и товарным знакам.
16. Михайлов А. А., Хмельнов А. Е. Модуль структурной раскладки графов потоков управления на плоскости для программы визуализации графов Visual Graph. 2016. Свидетельство о государственной регистрации программ для ЭВМ № 2014617137 М.: Федеральная служба по интеллектуальной собственности, патентам и товарным знакам.

Редакционно-издательский отдел
Федерального государственного бюджетного учреждения науки
Института динамики систем и теории управления имени В.М. Матросова СО РАН
664033, Иркутск, ул. Лермонтова, д. 134
e-mail: rio@icc.ru

Подписано к печати 15.09.2017
Формат бумаги 60 x 84 1/16, объем 1 п.л. Заказ №16. Тираж 100 экз.

Отпечатано в ИДСТУ СО РАН