ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
СПбГУТ))) Декабрь 2016 Том 4 № 4 ISSN 2307-1303 http://www.itt.sut.ru
TELECOM IT
December 2016 Vol. 4 Iss. 4 ISSN 2307-1303 http://www.itt.sut.ru

УТИЛИТА ДЛЯ ПОИСКА УЯЗВИМОСТЕЙ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ ТЕЛЕКОММУНИКАЦИОННЫХ УСТРОЙСТВ МЕТОДОМ АЛГОРИТМИЗАЦИИ МАШИННОГО КОДА. ЧАСТЬ 3. МОДУЛЬНО-АЛГОРИТМИЧЕСКАЯ АРХИТЕКТУРА

К. Е. Израилов[1*], В. В. Покусов[2]

[1] СПбГУТ, Санкт-Петербург, 193232, Российская Федерация
[2] Санкт-Петербургский университет ГПС МЧС России, 196105, Российская Федерация
* Адрес для переписки: konstantin.izrailov@mail.ru

Аннотация
Предмет исследования. Статья продолжает цикл статей, описывающих различные стороны архитектуры авторской утилиты алгоритмизации машинного кода. **Метод.** Рассматривается возможность алгоритмической реализации заявленного функционала модулей. **Основные результаты.** В статье приводятся принципы работы алгоритмов основных модулей функциональной архитектуры и их блок-схемы. **Практическая значимость.** Указываются технические детали осуществленной программной реализации прототипа Утилиты, выполненной по приведенным алгоритмам.
Ключевые слова
телекоммуникационные устройства, поиск уязвимостей, утилита алгоритмизации, модульно-алгоритмическая архитектура, алгоритмы.

Информация о статье
УДК 004.4'422
Язык статьи – русский.
Поступила в редакцию 09.11.16, принята к печати 25.11.16.

Ссылка для цитирования: Израилов К.Е., Покусов В.В. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Часть 3. Модульно-алгоритмическая архитектура // Информационные технологии и телекоммуникации. 2016. Том 4. № 4. С. 104–121.

UTILITY FOR VULNERABILITY SEARCH IN A SOFTWARE OF TELECOMMUNICATION DEVICES BY METHOD ALGORITMIZATION OF MACHINE CODE. PART 3. MODULAR-ALGORITHMIC ARCHITECTURE

K. Izrailov[1], V. Pokusov[2]

[1] SPbSUT, St. Petersburg, 193232, Russian Federation
[2] Saint-Petersburg University of State Fire Service of EMERCOM of Russia, 196105, Russian Federation
* Corresponding author: konstantin.izrailov@mail.ru

Abstract—Research subject. The article continues a series of articles describing various aspects of the architecture of the author's utility of computer code algorithmization. **Method.** The possibility of algorithmic realization of the declared functional of modules is considered. **Core results.** The article describes the operation principles of algorithms for the main modules of the functional architecture and their block diagrams. **Practical relevance.** Specify the technical details of the implemented software implementation of the utility prototype, performed according to the above algorithms.
Keywords—telecommunication devices, vulnerability search, algorithmization utility, modular-algorithmic architecture, algorithms/

Article info
Article in Russian.
Received 09.11.16, accepted 25.11.16.

For citation: Izrailov K., Pokusov V.: Utility for Vulnerability Search in a Software of Telecommunication Devices by Method Algoritmization of Machine Code. Part 3. Modular-Algorithmic Architecture // Telecom IT. 2016. Vol. 4. Iss. 4. pp. 104–121 (in Russian).

Введение

Функциональная и информационная архитектура программного средства алгоритмизации (далее – Утилиты), приведенные в предыдущих авторских статьях [1, 2], описывают его организацию с достаточно абстрактной позиции, не позволяющей перейти к непосредственной реализации. Модульно-алгоритмическая архитектура средства, включающая в себя алгоритмы работы основных модулей, может быть использована для кодирования и сборки работающего экземпляра программного средства. По результатам тестирования последнего можно будет проверить практически работоспособность и реальные возможности, как самого средства, так и метода алгоритмизации [3, 4, 5, 6, 7, 8], основные назначением которых является обеспечение безопасности информационно-телекоммуникационных систем [9, 10, 11, 12, 13, 14, 15, 16]. Описанию такой архитектуры и посвящена данная статья.

Функциональная архитектура представляет Утилиту в виде шести последовательно выполняемых стадий, сгруппированных по трем фазам. Каждая стадия содержит несколько модулей (включая анализаторы первой стадии); взаимодей-

ствие последних с модулями противоположных стадий осуществляется посредством данных [17], представленных информационной архитектурой и поддерживающих S-модель [18, 19]. Также, есть модули, сквозные для стадий – обмен данными с ними происходит на протяжении всей работы. Основные 23 алгоритма, соответствующие модулям функциональной архитектуры, приведены далее. В последних проведенных исследования стадии содержат новые модуль, отсутствующий в предыдущей функциональной архитектуре, а именно – модуль генерации архитектуры восстанавливаемого кода, модуль генерации текстового описания корректировок алгоритмизации и модуль генерации текстового описания архитектуры. Для каждого приводятся принцип работы и блок-схема алгоритма. Нумерация модулей имеет формат: *M_X_Y*, где *X* – номер стадии, а *Y* – номер модуля в стадии; для сквозных модулей номер стадии обозначается, как *C*.

Алгоритмы, соответствующие модулям функциональной архитектуры

M_1_1. Лексический анализатор

Принцип работы модуля основан на разбиении входного текста ассемблерного кода в виде потока символов на лексемы согласно заданным регулярным выражениям. Блок-схема алгоритма представлена на рис. 1.

M_1_2. Синтаксический анализатор

Принцип работы модуля основан на комбинировании потока лексем согласно заданным правилам – так называемая *свертка*. Схема работы модуля соответствует типичному конечному автомату, переходящему между состояниями согласно получаемым лексемам. Блок-схема алгоритма представлена на рис. 2.

M_1_3. Семантический анализатор

Принцип работы модуля основан на построении дерева абстрактного синтаксиса (англ. *Abstract Syntax Tree* или AST) входного кода согласно произведенным сверткам и синтаксическому значению правил. В случае свертки правила корректировки алгоритмизации (заданной через расширение синтаксиса входного ассемблерного кода) соответствующая информация сохраняется и используется последующими модулями. Блок-схема алгоритма представлена на рис. 3.

M_2_1. Модуль выделение подпрограмм

Принцип работы модуля основан на анализе AST и выделении в нем поддеревьев, относящихся к отдельным подпрограммам. Также создаются глобальные переменные, указанные явно в ассемблерном коде. Блок-схема алгоритма представлена на рис. 4.

M_2_2. Модуль построения графа потока управления

Принцип работы модуля основан на анализе AST каждой из выделенных подпрограмм и построении графа потока управления (англ. *Control Flow Graph* или CFG). Узлы графа содержат собственные AST, соответствующие элементарным выполняемым операциям. Также осуществляется выделение входных и выходных аргументов, заданных корректировками алгоритмизации. Параллельно

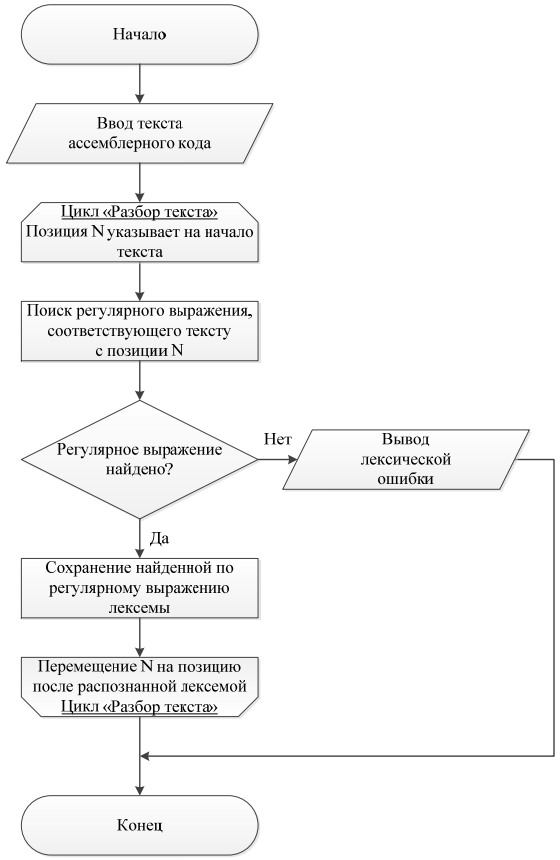

Рис. 1. Блок-схема алгоритма лексического анализатора

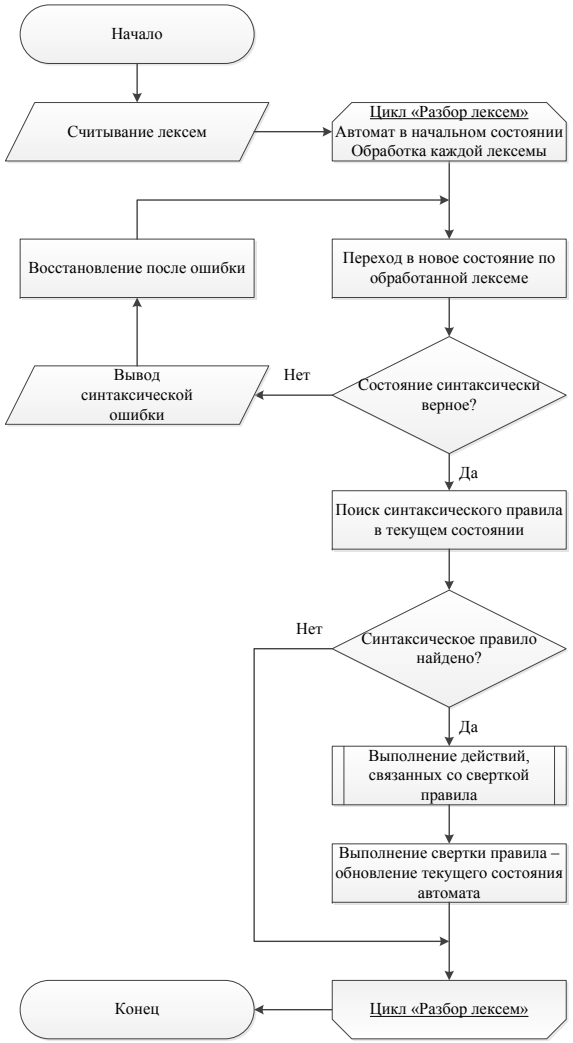

Рис. 2. Блок-схема алгоритма синтаксического анализатора

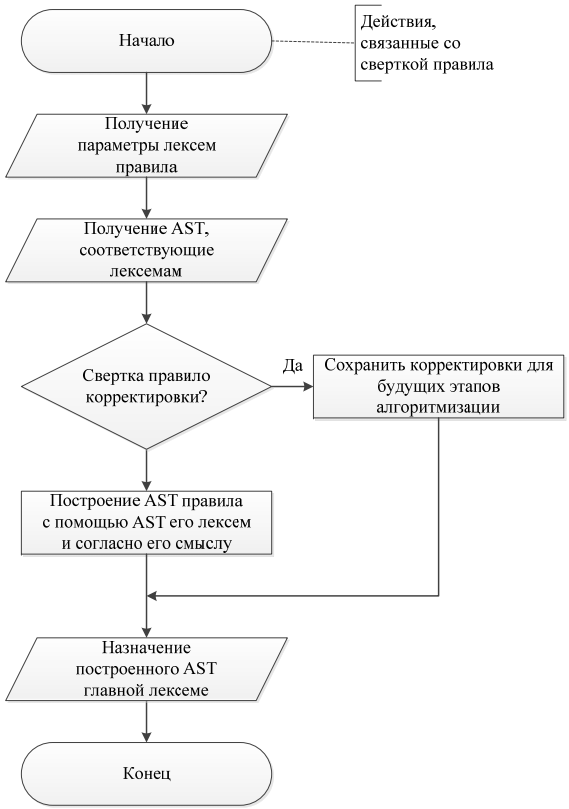

Рис. 3. Блок-схема алгоритма семантического анализатора

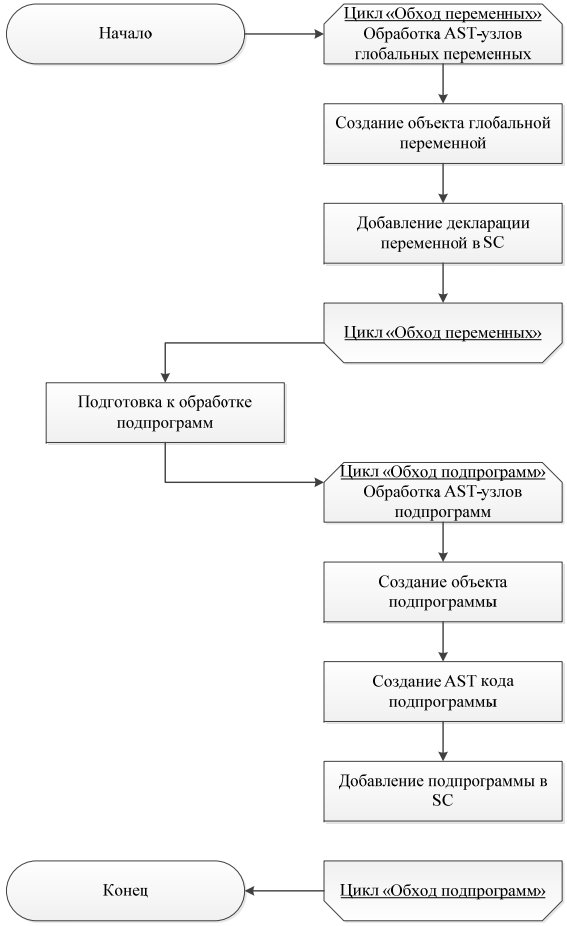

Рис. 4. Блок-схема модуля выделения подпрограмм

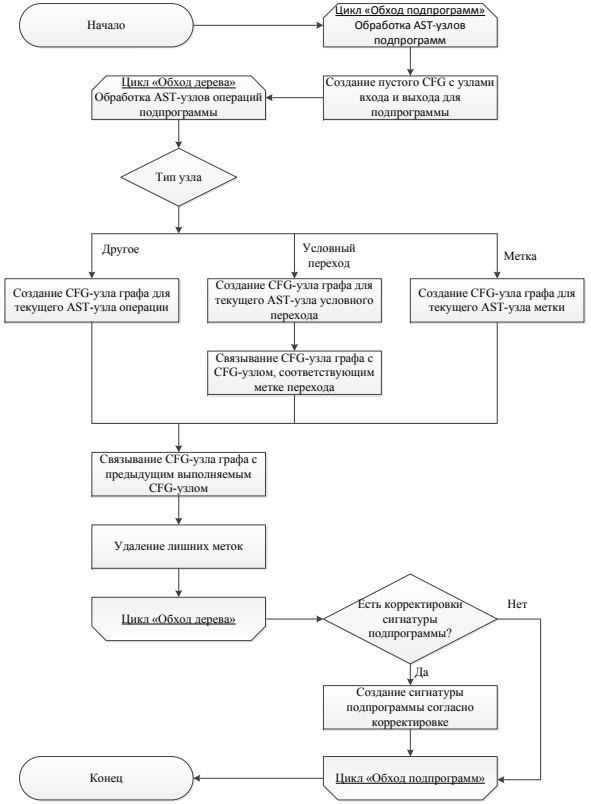

Рис. 5. Блок-схема модуля построения CFG

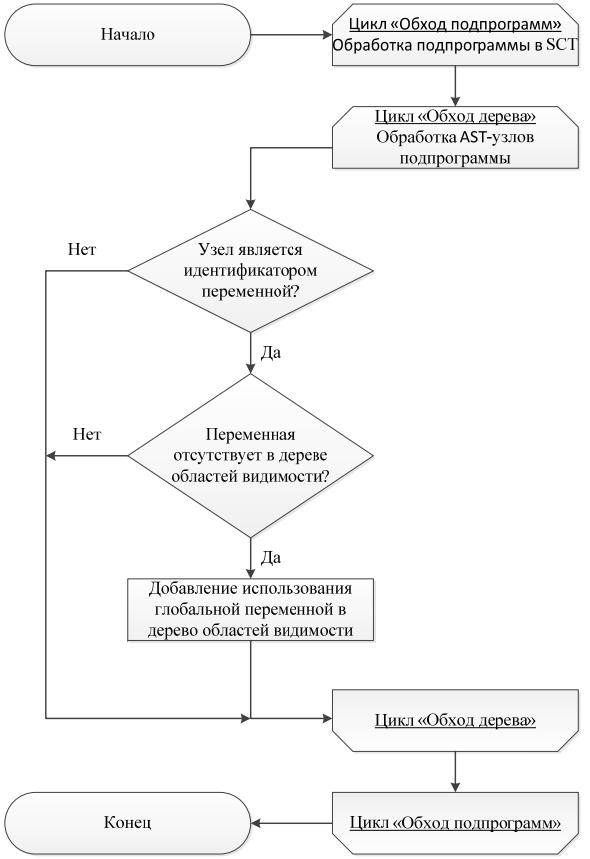

Рис. 6. Блок-схема модуля выделения глобальных переменных

строится граф вызовов подпрограмм по соответствующим операциям. Блок-схема алгоритма представлена на рис. 5.

M_2_3. Модуль выделение глобальных переменных

Принцип работы модуля основан на анализе AST каждой из выделенных подпрограмм и определении глобальных переменных по идентификаторам, параллельно строя дерево областей видимости (англ. *Scope Tree* или SCT). Блок-схема алгоритма представлена на рис. 6.

M_3_1. Модуль построения графа потока данных

Принцип работы модуля основан на анализе использовании переменных в каждом узле CFG и сборе следующей информации: первое и последнее использование, хранимое значение, его связь со значениями других переменных. Это позволяет, как получить информацию о временах жизни переменных, так и построить граф поток данных (англ. *Data Flow Graph* или DFG) – по которому определяема зависимость между вычислениями. Блок-схема алгоритма представлена на рис. 7.

M_3_2. Модуль уточнения сигнатуры подпрограмм

Принцип работы модуля основан на учете информации о временах жизни переменных, а именно первому и последнему чтению их значения. Используемые без инициализации переменных считаются входными аргументами подпрограммы, а не используемые после инициализации – выходными. Блок-схема алгоритма представлена на рис. 8.

M_3_3. Модуль выделения структурных метаданных

Метод является наиболее существенным с точки зрения эффективности алгоритмизации, поскольку результаты его работы отражаются на конечной структурированности алгоритмизированного представления [20].

Принцип работы модуля основан на раскраске CFG и определении ветвлений (условных переходов) и циклов в нем; их замена на отдельные управляющие структуры «размыкает» цикличности и преобразовывает граф в дерево потока управления (англ. *Control Flow Tree* или CFT) – аналогичное представлению Насси-Шнейдермана [21]. Блок-схема алгоритма представлена на рис. 9.

M_4_1. Модуль оптимизации структурных метаданных

Принцип работы модуля основан на выполнении специализированных оптимизационных действий. Последние производят сопоставление топологии CFT заданным шаблонам и его перестроение по связанным правилам. Блок-схема алгоритма представлена на рис. 10.

M_4_2. Модуль оптимизации дерева потока управления

Принцип работы модуля основан на выполнении специализированных оптимизационных действий. Последние производят сопоставление областей CFT заданным шаблонам и их замене на более короткие (рис. 11).

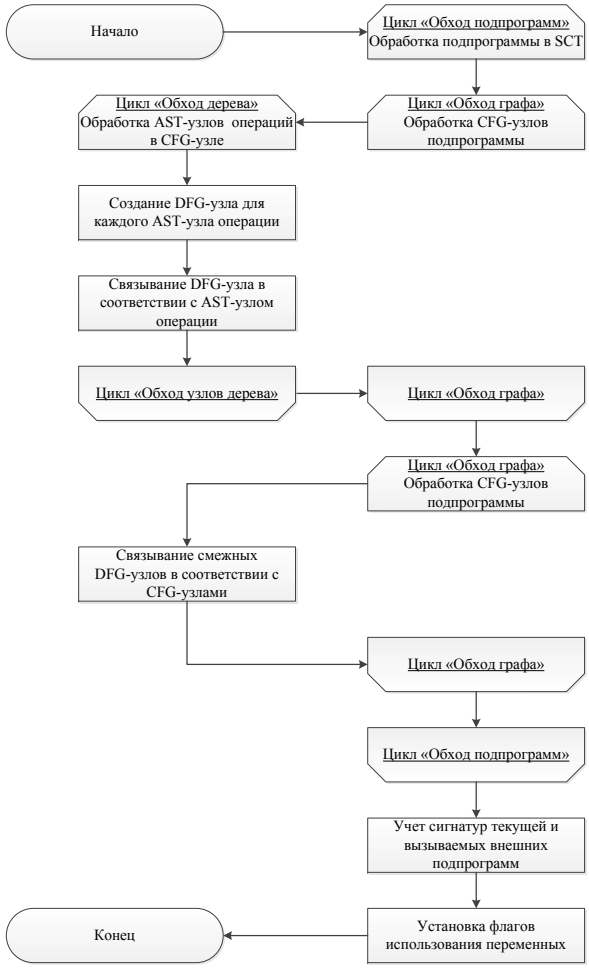

Рис. 7. Блок-схема модуля построения DFG

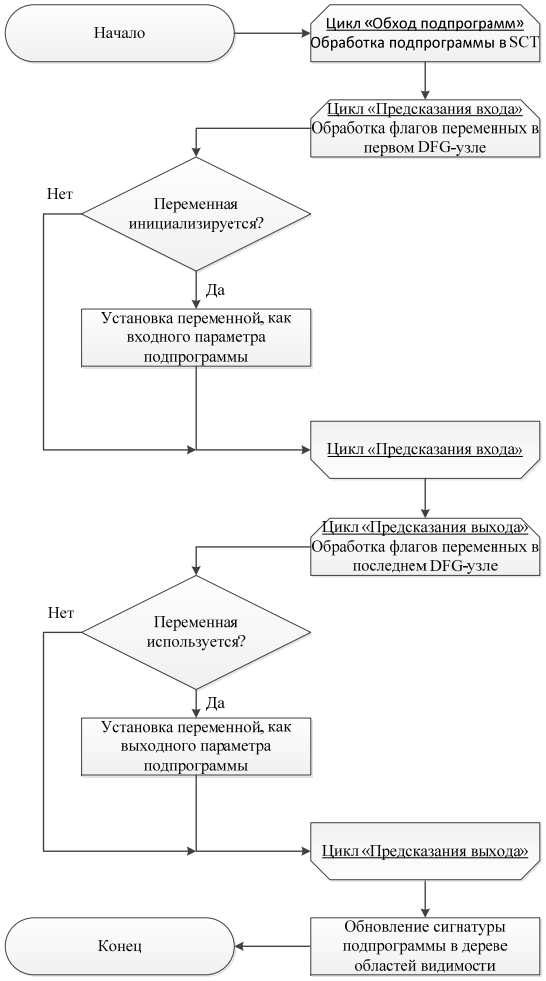

Рис. 8. Блок-схема модуля уточнения сигнатуры подпрограмм

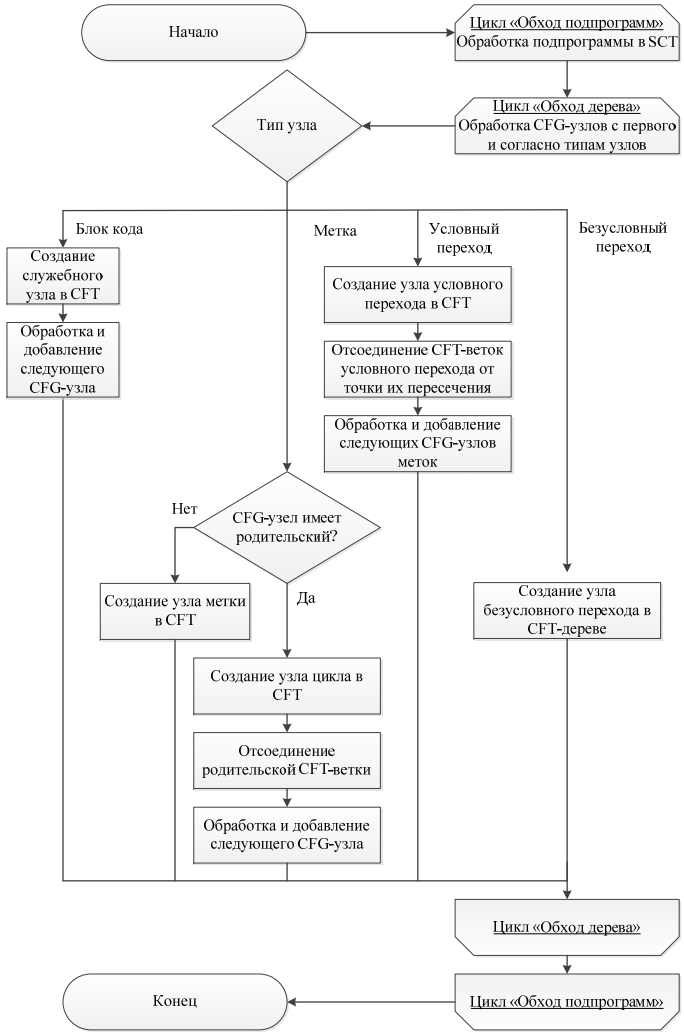

Рис. 9. Блок-схема модуля выделения структурных метаданных

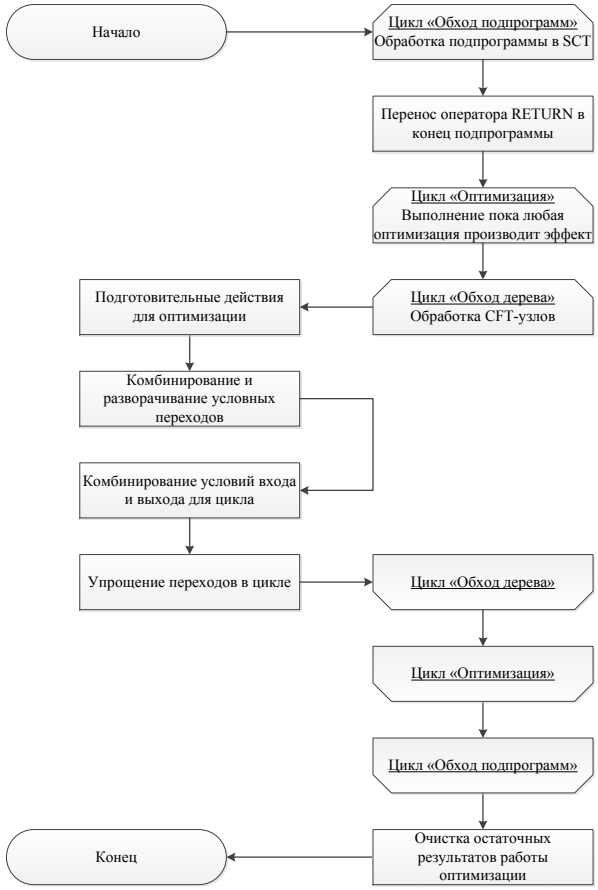

Рис. 10. Блок-схема модуля оптимизации структурных метаданных

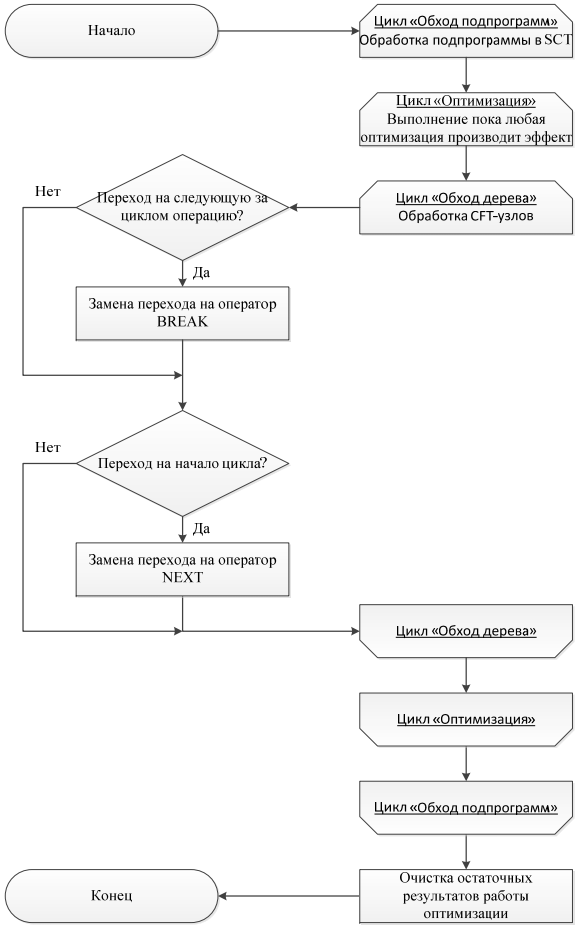

Рис. 11. Блок-схема модуля оптимизации CFT

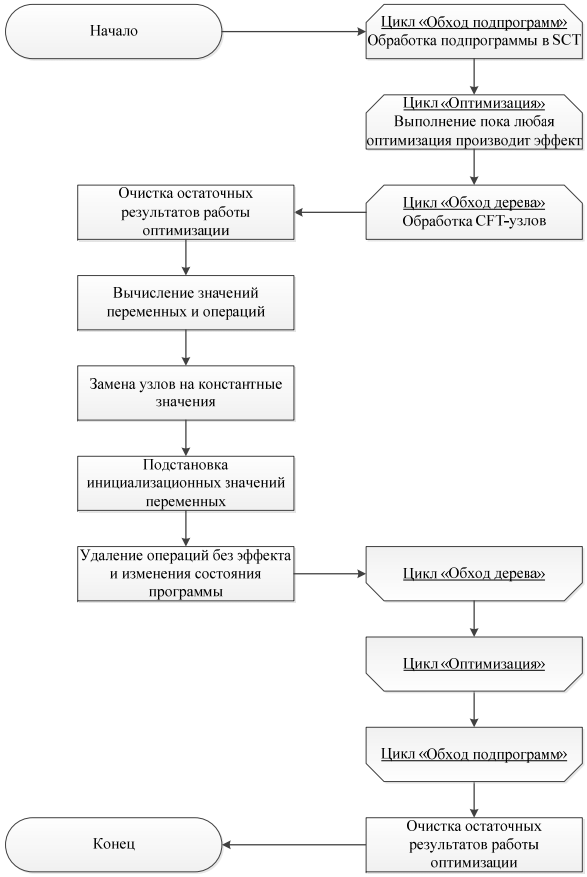

Рис. 12. Блок-схема модуля оптимизации вычислений

M_4_3. Модуль оптимизации вычислений

Принцип работы модуля основан на выполнении специализированных оптимизационных действий. Последние производят анализ операций кода в узлах CFT, вычисление значений переменных, оценку эффекта от выполнения операции с последующим изменением операций. Блок-схема алгоритма представлена на рис. 12.

M_5_1. Модуль генерации псевдокода глобальных переменных

Принцип работы модуля основан на обходе ветки дерева областей видимости, содержащей глобальные переменные, и создании соответствующих узлов в дереве псевдокода (англ. *Pseudo Code Tree* или PCT). Блок-схема алгоритма представлена на рис. 13.

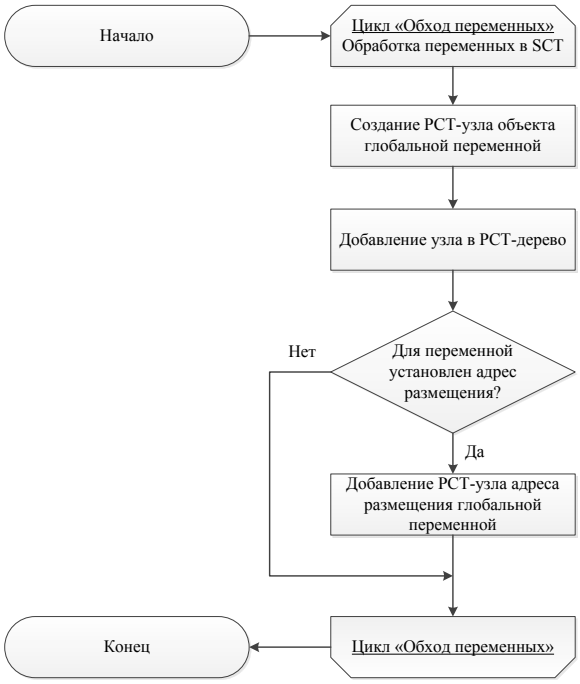

Рис. 13. Блок-схема модуля генерации псевдокода глобальных переменных

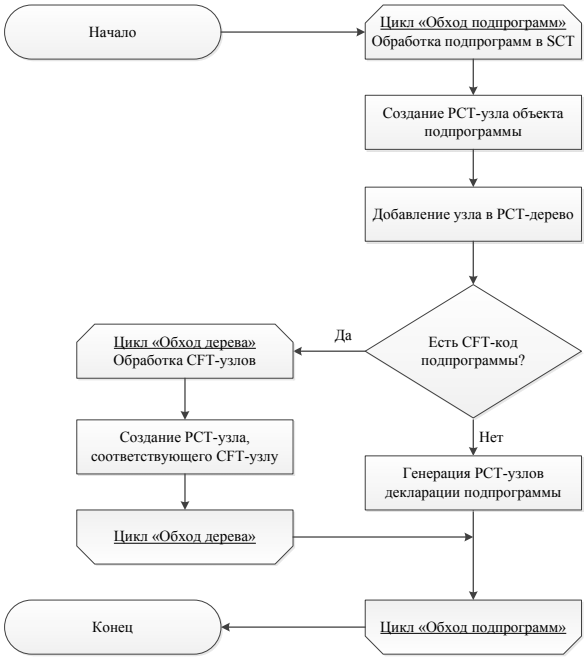

Рис. 14. Блок-схема модуля генерации псевдокода подпрограмм

M_5_2. Модуль генерации псевдокода подпрограмм

Принцип работы модуля основан на обходе ветки дерева областей, содержащей подпрограммы, и создании соответствующих узлов в PCT. Блок-схема алгоритма представлена на рис. 14.

M_5_3. Модуль лаконизации псевдокода

Принцип работы модуля основан на выполнении специализированных оптимизационных действий в интересах повышения восприятия кода человеком. Модуль обрабатывает операции в PCT подпрограмм, сопоставляет их с шаблонами и заменяет узлы на более информативные (с позиции человека). Блок-схема алгоритма представлена на рис. 15.

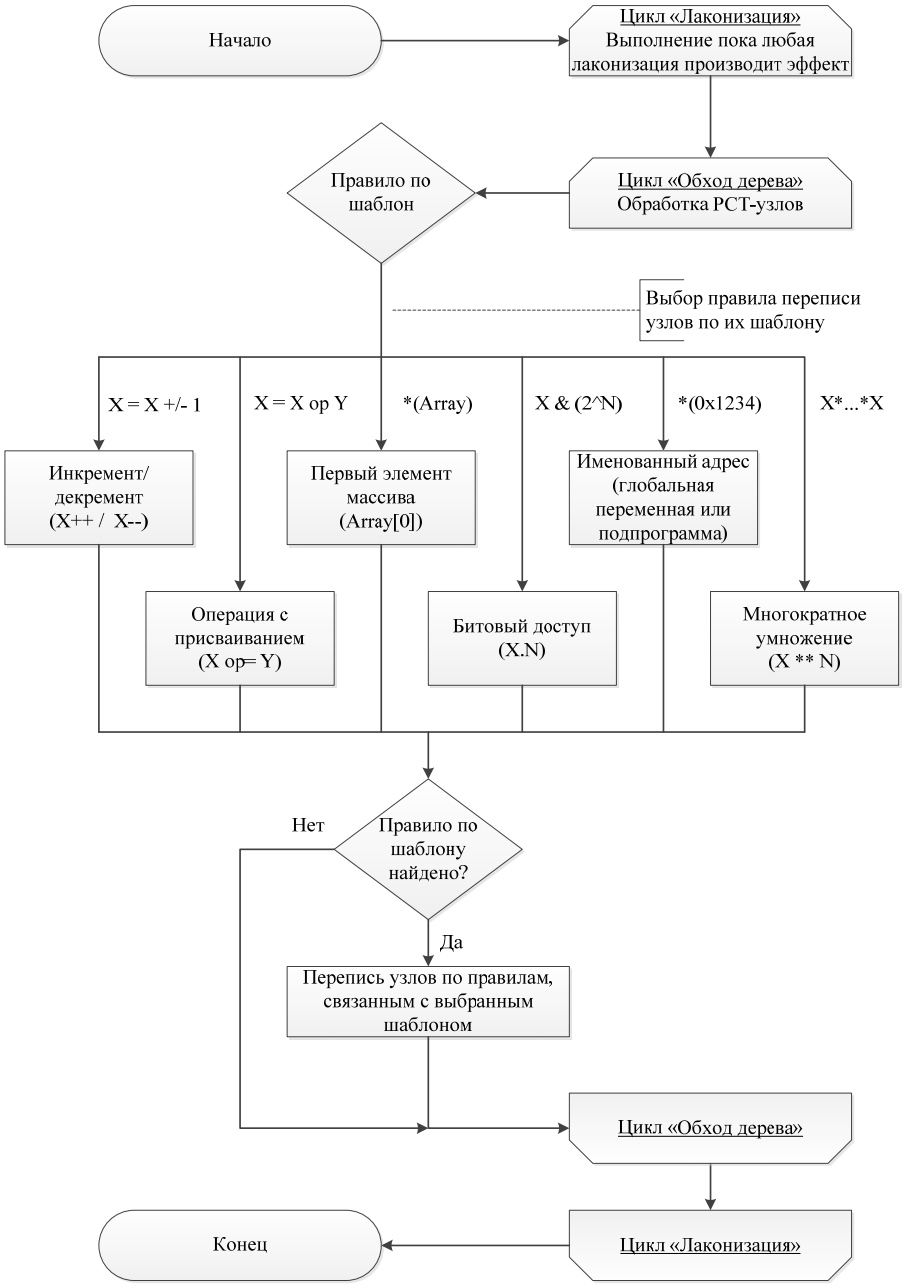

Рис. 15. Блок-схема модуля лаконизации псевдокода

M_5_4. Модуль генерации архитектуры

Принцип работы модуля основан на учете взаимосвязи глобальные переменных, используемых различными подпрограммами. Для этого по PCT строится промежуточный граф, отражающий все такие взаимосвязи, и производится его *разноцветная* раскраска так, чтобы только смежные узлы имели одинаковый цвет. Если несколько подпрограмм используют одну переменную, то считается, что они объединены в один модуль – на графе их цвет будет одинаковым. Информация о модулях добавляется в общее дерево псевдокода. Вызовы подпрограмм одним модулем из другого осуществляются посредством интерфейсов последнего. Блок-схема алгоритма представлена на рис. 16.

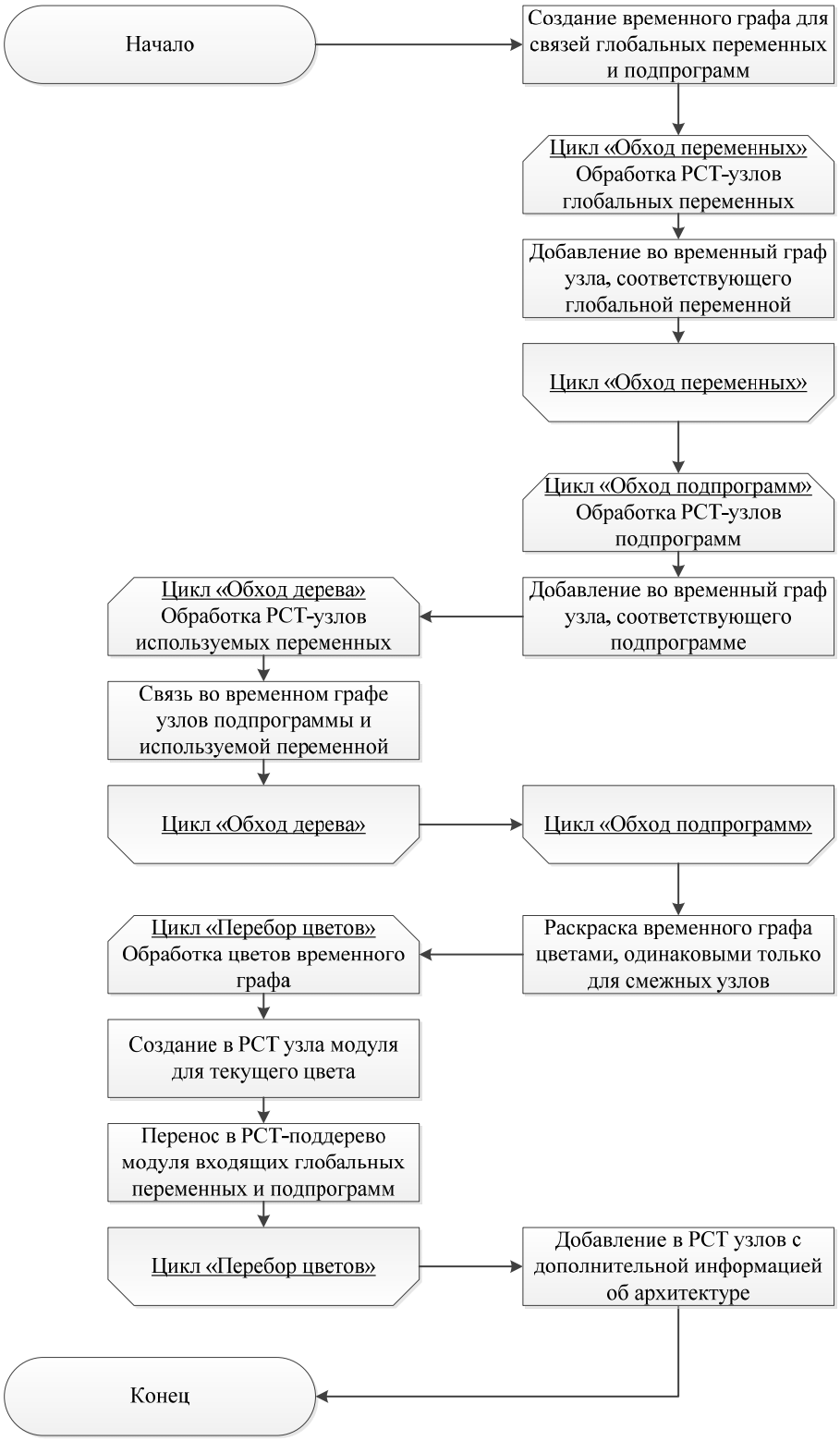

Рис. 16. Блок-схема модуля генерации архитектуры

M_6_1. Модуль генерации текстового описания корректировок алгоритмизации

Принцип работы модуля основан на генерации корректировок алгоритмизации, введенных пользователем в ассемблерном коде. Генерируемый формат идентичен понимаемому Утилитой на входе. Блок-схема алгоритма представлена на рис. 17.

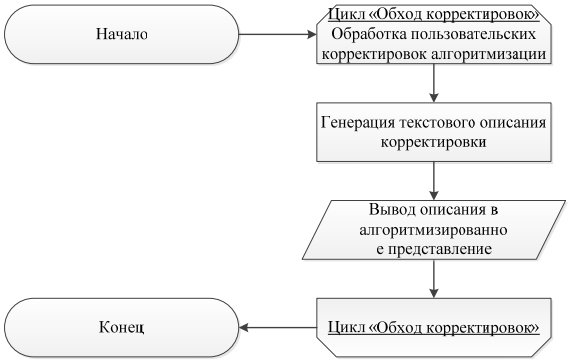

Рис. 17. Блок-схема модуля генерации текстового описания корректировок алгоритмизации

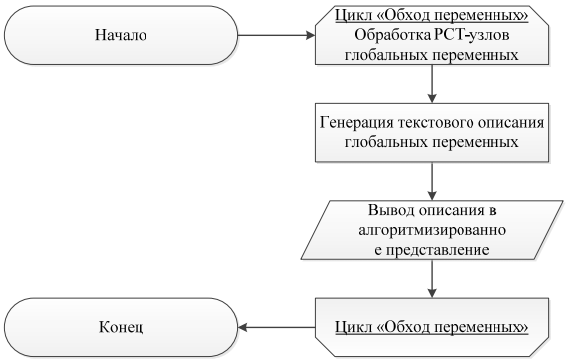

Рис. 19. Блок-схема модуля генерации текстового описания глобальных переменных

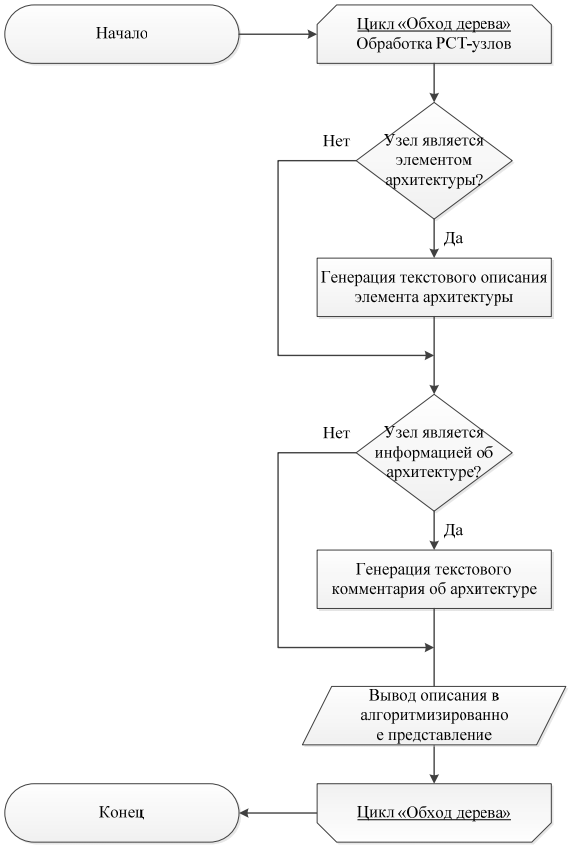

Рис. 18. Блок-схема модуля генерации текстового архитектуры

M_6_2. Модуль генерации текстового описания архитектуры

Принцип работы модуля основан на обходе поддерева псевдокода элементов архитектуры и генерации для каждого узла его текстового описание в алгоритмизированном. Блок-схема алгоритма представлена на рис. 18.

M_6_3. Модуль генерации текстового описания глобальных переменных

Принцип работы модуля основан на обходе поддерева псевдокода глобальных переменных, объединенных в элементы архитектуры, и генерации для каждого узла его текстового описание в алгоритмизированном. Блок-схема алгоритма представлена на рис. 19.

M_6_4. Модуль генерации текстового описания подпрограмм

Принцип работы модуля основан на обходе поддерева псевдокода подпрограмм, объединенных в группирующие элементы архитектуры, и генерации для каждого узла ее текстового описание в алгоритмизированном. Подпрограмма описывается сигнатурой, блоком операций и опциональными комментариями. Блок-схема алгоритма представлена на рис. 20.

M_6_5. Модуль генерации информации об уязвимостях

Модуль работает, как составная часть модуля генерации текстового описания подпрограмм, добавляя в описание информацию об уязвимостях, хранимую в специальных узлах псевдокода. Блок-схема алгоритма представлена на рис. 21.

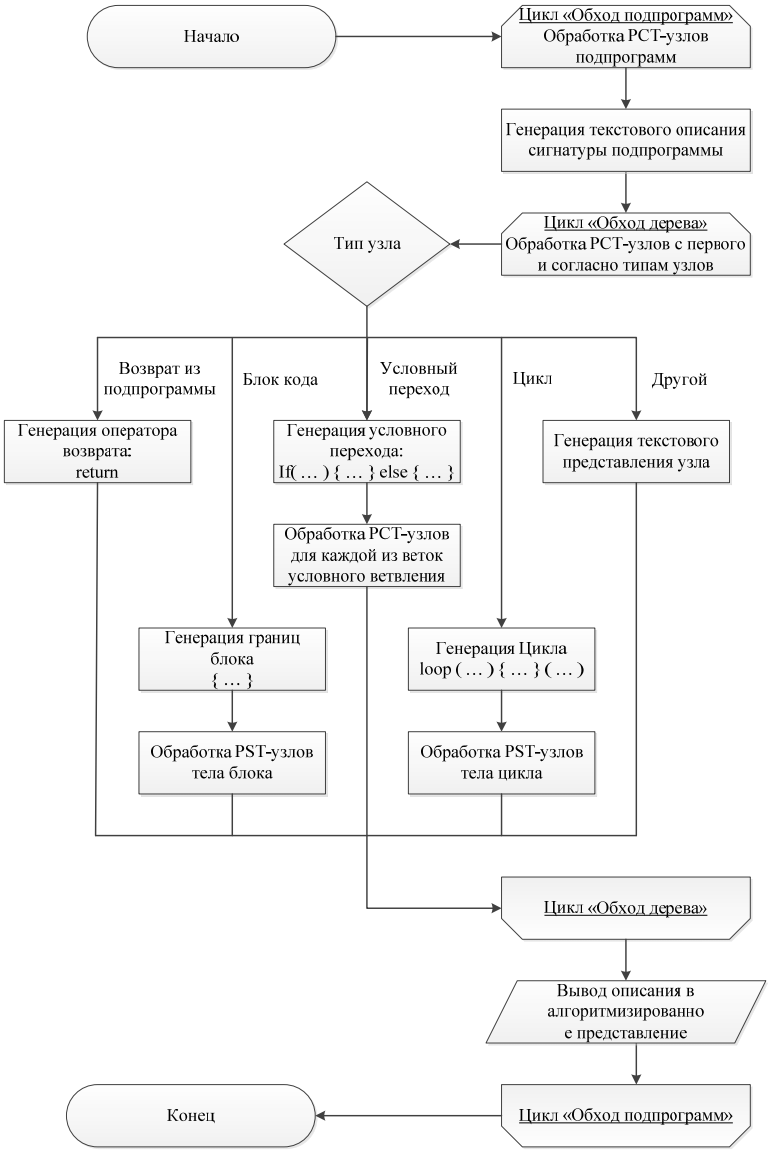

Рис. 20. Блок-схема модуля генерации текстового описания подпрограмм

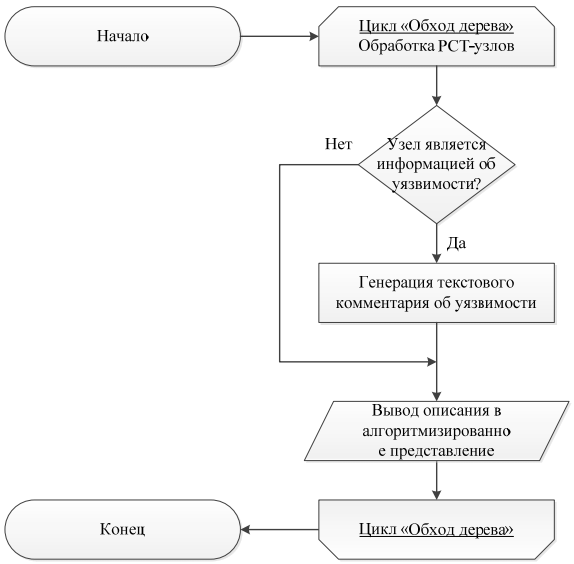

Рис. 21. Блок-схема модуля генерации информации об уязвимостях

M_C_1. Модуль поиска уязвимостей

Принцип работы модуля основан на эвристическом подходе и состоит из применения обобщенных шаблонов, систем критериев и оценок внутреннего представления Утилиты на предмет поиска потенциальных уязвимостей. Модуль может определить разрушение структуры программном коде и попытку записи в защищенную область памяти. В результате, будут добавлены специальные узлы в деревья и графы различных стадий, сигнализирующие о найденных уязвимостях. Блок-схема алгоритма представлена на рис. 22.

M_C_2. Модуль учета корректировок алгоритмизации

Принцип работы модуля основан на внесении поправок в работу всех других модулей на основании корректировок, заданных во входном ассемблерном коде. Модуль может указать другим точную сигнатуру подпрограммы, имя и адрес глобальной переменной и других идентификаторов, защиту от записи в область памяти (применяется при поиске уязвимостей). Блок-схема алгоритма представлена на рис. 23.

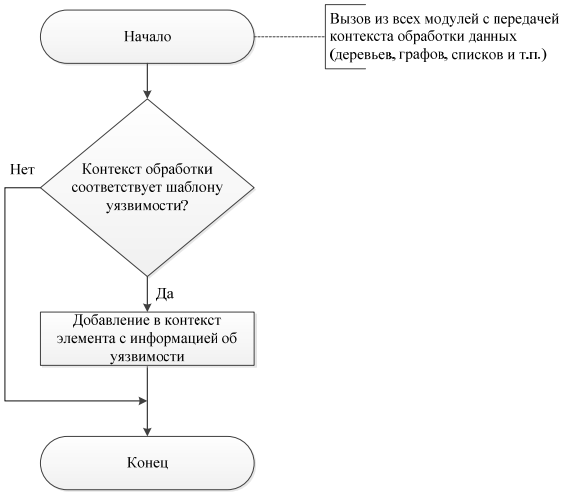

Рис. 22. Блок-схема модуля поиска уязвимостей

Рис. 23. Блок-схема модуля учета корректировок алгоритмизации

Реализация Утилиты

Приведенная архитектура описывает основные действия каждого модуля, условия и последовательность их выполнения, позволяя перейти непосредственно к кодированию. В результате такой реализации был получен исходный код прототипа Утилиты.

Опишем программную архитектуру разработанного прототипа. Основная часть исходного кода написана на языке C++ с четким делением на заголовочные и компилируемые файлы для каждого основного класса парадигмы объектно-ориентированного программирования или их групп. Также, активно используются конструкции namespace для логического объединения классов. Большинство ИК разработано вручную, хотя отдельные его части являются модификацией открытых реализаций требуемого функционала Утилиты (например, разборщик файлов формата JSON).

Весь исходный код делится на четыре следующие части. Проект статической библиотеки Common является общим для большой группы приложений и содержит вспомогательный функционал, такой, как конвертация различных типов данных, поддержка битовых последовательностей, расширенная поддержка работы с файлами и потоками, функции для работы с текстом, программно-языковые определения различных часто используемых типов, констант, макросов. Проект статической библиотеки Json реализует функционал по разбору и генерации данных в формате JSON; он используется в Утилите для отладочных целей. Проект статической библиотеки Core является главным, поскольку содержит весь основной функционал Утилиты. Использование библиотечного типа позволяет запускать процесс алгоритмизации не только из командной строки ОС, но и любыми другим способами (например, из графического приложения на C#). Проект консольного приложения Console является оболочкой для запуска библиотеки Core, производит разбор аргументов командной строки и передает их вместе с входным ассемблером в основную функцию библиотеки Core.

Утилита разработана в графической среде разработки Microsoft Visual Studio 2010 (MSVS2010) с подключением дополнительных средств построения кода; в ней же производится отладка кода. Среда разработки MSVS2010 была выбрана по причине широкого распространения, возможности бесплатного использования, наличия гибких настроек, удобной визуальной среды, полноценной поддержки современных языков программирования и возможности подключения дополнительного инструментария.

В качестве генераторов кода использовались стандартные компиляторы, ассемблеры и линкеры, входящие в состав MSVS2010. Также, для реализации первой стадии Утилиты производилась практически полная генерация кода ее модулей – лексического и синтаксического анализаторов, с помощью соответствующих подключенных утилит: генератора лексического анализатора Flex и генератора синтаксического анализатора Yacc, работающих совместно. Исходные лексика и грамматика входного ассемблера, используемые генераторами, создавались вручную на специальных языках; они соответствуют синтаксису ассемблера PowerPC, генерируемого продуктом IDA Pro.

Следующей статьей, логично завершающей их цикл, должно стать практическое тестирование прототипа Утилиты [22, 23, 24]. Получение положительного результата послужит основным доказательством применимости алгоритмизации машинного кода в интересах поиска уязвимости[1].

Окончание.
Начало – выпуски 1 и 2 2016 года.

Литература

1. Буйневич М. В., Израилов К. Е. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Часть 1. Функциональная архитектура // Информационные технологии и телекоммуникации. 2016. Т. 4. № 1. С. 115–130. URL: http://www.sut.ru/doci/nauka/review/20161/115-130.pdf

[1] On-line версия утилиты алгоритмизации машинного кода: сайт Израилова К. Е. [Электронный ресурс]. URL: http://demono.ru/online (дата обращения: 20.10.2016).

2. Израилов К. Е. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Часть 2. Информационная архитектура // Информационные технологии и телекоммуникации. 2016. Т. 4. № 2. С. 86–104. URL: http://www.sut.ru/doci/nauka/review/20162/86-104.pdf

3. Буйневич М. В., Израилов К. Е. Метод алгоритмизации машинного кода телекоммуникационных устройств // Телекоммуникации. 2012. № 12. С. 2–6.

4. Буйневич М. В., Израилов К. Е. Автоматизированное средство алгоритмизации машинного кода телекоммуникационных устройств // Телекоммуникации. 2013. № 6. С. 2–9.

5. Израилов К. Е. Утилита восстановления алгоритмов работы машинного кода // Свидетельство о государственной регистрации программы для ЭВМ № 2013618433. 09.09.2013.

6. Buinevich M., Izrailov K. Method and Utility for Recovering Code Algorithms of Telecommunication Devices for Vulnerability Search // 16th International Conference on Advanced Communication Technology (ICACT). 2014. pp. 172–176.

7. Buinevich M., Izrailov K., Vladyko A. Method for Partial Recovering Source Code of Telecommunication Devices for Vulnerability Search // 17th International Conference On Advanced Communications Technology (ICACT). 2015. pp. 76–80.

8. Buinevich M., Izrailov K., Vladyko A. Method and Prototype of Utility for Partial Recovering Source Code for Low-Level and Medium-Level Vulnerability Search // 18th International Conference on Advanced Communication Technology (ICACT). 2016. pp. 700–707.

9. Израилов К. Е. Анализ состояния в области безопасности программного обеспечения // II Международная научно-технической и научно-методической конференция «Актуальные проблемы инфотелекоммуникаций в науке и образовании». 2013. С. 874–877.

10. Буйневич М. В., Израилов К. Е., Мостович Д. И., Ярошенко А. Ю. Проблемные вопросы нейтрализации уязвимостей программного кода телекоммуникационных устройств // Проблемы управления рисками в техносфере. 2016. № 3 (39). С. 81-89.

11. Буйневич М. В., Израилов К. Е., Мостович Д. И. Сравнительный анализ подходов к поиску уязвимостей в программном коде // Актуальные проблемы инфотелекоммуникаций в науке и образовании: сборник научных статей V международной научно-технической и научно-методической конференции. 2016. С. 256–260.

12. Израилов К. Е. Модель прогнозирования угроз телекоммуникационной системы на базе искусственной нейронной сети // Вестник ИНЖЭКОНа. Серия: Технические науки. 2012. № 8 (59). С. 150–153.

13. Израилов К. Е., Васильева А. Ю. Язык описания модели безопасности телекоммуникационной сети // X Международная научно-технической конференция «Новые информационные технологии и системы» (НИТИС). 2012. С. 272–275.

14. Израилов К. Е. Алгоритмизация машинного кода телекоммуникационных устройств как стратегическое средство обеспечения информационной безопасности // Национальная безопасность и стратегическое планирование. 2013. № 2 (2). С. 28–36.

15. Buinevich M., Izrailov K., Vladyko A. The Life Cycle of Vulnerabilities in the Representations of Software for Telecommunication Devices // 18th International Conference on Advanced Communications Technology (ICACT). 2016. pp. 430–435.

16. Карев А. С., Бирих Э. В., Сахаров Д. В., Виткова Л. А. Проблемы информационной безопасности в интернете вещей // 2-я Международная научно-техническая конференции студентов, аспирантов и молодых ученых «Интернет вещей и 5G» (INTHITEN). 2016. С. 66–70.

17. Израилов К. Е. Внутреннее представление прототипа утилиты для восстановления кода // Фундаментальные и прикладные исследования в современном мире. 2013. № 2. С. 79–90.

18. Буйневич М. В., Щербаков О. В., Израилов К. Е. Модель машинного кода, специализированная для поиска уязвимостей // Вестник Воронежского института ГПС МЧС России. 2014. № 2 (11). С. 46–51.

19. Буйневич М. В., Щербаков О. В., Израилов К. Е. Структурная модель машинного кода, специализированная для поиска уязвимостей в программном обеспечении автоматизированных систем управления // Проблемы управления рисками в техносфере. 2014. № 3 (31). С. 68–74.

20. Израилов К. Е. Расширение языка «С» для описания алгоритмов кода телекоммуникационных устройств // Информационные технологии и телекоммуникации. 2013. № 2 (2). С. 21–31. URL: http://www.sut.ru/doci/nauka/review/2-13.pdf

21. Израилов К. Е. Применение диаграммы Насси-Шнейдермана для анализа структурированных алгоритмов // Пятый научный конгресс студентов и аспирантов ИНЖЭКОН-2012. СПб.: СПбГИЭУ. 2012. С. 49–50.

22. Израилов К. Е., Васильева А. Ю., Рамазанов А. И. Укрупненная методика оценки эффективности автоматизированных средств, восстанавливающих исходный код в целях поиска уязвимостей // Вестник ИНЖЭКОНа. Серия: Технические науки. 2013. № 8 (67). С. 107–109.

23. Израилов К. Е. Методика оценки эффективности средств алгоритмизации, используемых для поиска уязвимостей // Информатизация и связь. 2014. № 3. С. 44–47.

24. Buinevich M., Izrailov K., Vladyko A. Testing of Utilities for Finding Vulnerabilities in the Machine Code of Telecommunication Devices // 19th International Conference on Advanced Communication Technology (ICACT). 2017. pp. 408–414.

References

1. Buinevich, M., Izrailov, K.: Utility for Vulnerability Search in a Software of Telecommunication Devices by Method Algorithmization of Machine Code. Part 1. Functional Architecture // Telecom IT. 2016. Vol. 4. Iss. 1. pp. 115–130. URL: http://www.sut.ru/doci/nauka/review/20161/115-130.pdf

2. Израилов К.Е.: Utility for Vulnerability Search in a Software of Telecommunication Devices by Method Algorithmization of Machine Code. Part 2. Information Architecture // Telecom IT. 2016. Vol. 4. Iss. 2. pp. 86–104. URL: http://www.sut.ru/doci/nauka/review/20162/86-104.pdf

3. Buinevich, M., Izrailov, K. Method Algorithmization of Machine Code of Telecommunication Devices // Telecommunikatsii. 2012. No. 12. pp. 2–6.

4. Buinevich, M., Izrailov, K. Automated Tool of Algorithmization Machine Code of Telecommunication Devices // Telecommunikatsii. 2013. No. 6. pp. 2–9.

5. Izrailov, K. Utility for Recovering Machine Code Algorithms // Certificate of State Registration of the Computer Program № 2013618433. 09.09.2013.

6. Buinevich, M., Izrailov, K. Method and Utility for Recovering Code Algorithms of Telecommunication Devices for Vulnerability Search // 16th International Conference on Advanced Communication Technology (ICACT). 2014. pp. 172–176.

7. Buinevich, M., Izrailov, K., Vladyko, A. Method for Partial Recovering Source Code of Telecommunication Devices for Vulnerability Search // 17th International Conference On Advanced Communications Technology (ICACT). 2015. pp. 76–80.

8. Buinevich, M., Izrailov, K., Vladyko, A. Method and Prototype of Utility for Partial Recovering Source Code for Low-Level and Medium-Level Vulnerability Search // 18th International Conference on Advanced Communication Technology (ICACT). 2016. pp. 700–707.

9. Izrailov, K. Analysis of the Security State of Software // II International Scientific-Technical and Scientific-Methodical Conference "Actual Problems of Education in Science and Education". 2013. pp. 874–877.

10. Buinevich, M., Izrailov K., Mostovich D., Yaroshenko A. Problematic Issues of Neutralization of Vulnerabilities in a Software Code of Telecommunication Devices // Problemy upravleniya riskami v tekhnosfere. 2016. No. 3 (39). pp. 81–89.

11. Buinevich, M., Izrailov K., Mostovich D. A Comparative Analysis of Approaches to Finding Vulnerabilities in Software Code // V International Scientific-Technical and Scientific-Methodical Conference "Actual Problems of Education in Science and Education". 2016. pp. 256–260.

12. Izrailov, K. Model of Forecasting the Telecommunication System Threats based on the Artificial Neural Network // Vestnik INGECONa. Seriya: Tekhnicheskie nauki. 2012. № 8 (59). pp. 150–153.

13. Izrailov, K., Vasilieva, A. Description Language of Telecommunications Network Security Model // X International Scientific and Technical Conference "New Information Technologies and Systems" (NITIS). 2012. pp. 272–275.

14. Izrailov, K. Algorithmization Machine Code of Telecommunication Device as a Means of Strategic Information Security // Natsional'naya bezopasnost' i strategicheskoe planirovanie. 2013. No. 2 (2). pp. 28–36.

15. Buinevich, M., Izrailov, K., Vladyko, A. The Life Cycle of Vulnerabilities in the Representations of Software for Telecommunication Devices // 18th International Conference on Advanced Communications Technology (ICACT). 2016. pp. 430–435.

16. Karev, A., Birih, E., Sakharov, D., Vitkova, L. The Problems of Information Security in the Internet of Things // 2nd Young Researchers International Conference on the Internet of Things and Its Enablers: "IoT and 5G" (INTHITEN). 2016. pp. 66–70.

17. Izrailov, K. The Internal Representation of a Prototype Utility for Code Recovery // Fundamental'nyye i prikladnyye issledovaniya v sovremennom mire. 2013. No. 2. pp. 79–90.

18. Buinevich, M., Sherbakov, O., Izrailov, K. Model of Machine Code Specialized for Vulnerabilities Search // Vestnik Voronezhskogo instituta GPS of EMERCOM of Russia. 2014. No. 2 (11). pp. 46–51.

19. Buinevich, M., Sherbakov, O., Izrailov, K. Structural Model of Machine Code Specialized for Search for Software Vulnerabilities of the Automated Control Systems // Problemy upravleniya riskami v tekhnosfere. 2014. No. 3 (31). pp. 68–74.

20. Izrailov, K. C-Language Extension for Algorithm Description of Telecommunication Devices Code // Telecom IT. 2013. Vol. 2 (2). pp. 21–31. URL: http://www.sut.ru/doci/nauka/review/2-13.pdf

21. Izrailov, K. The Use of Nassi-Shneiderman Diagrams for Structured Analysis Algorithms // The Fifth Scientific Congress of Students and Graduate Students of ENGECON-2012. SPb.: SPbGIEU. 2012. pp. 49–50.

22. Izrailov, K., Vasilieva, A., Ramazanov, A. Enlarged Assessment Method of Effectiveness of Automated Tools, Recovering Source Code in Search for Vulnerability // Vestnik INGECONa. Seriya: Tekhnicheskie nauki. 2013. No. 8 (67). pp. 107–109.

23. Izrailov, K. Methods of Assessing the Effectiveness of Medium of Algorithmization Used to Find Vulnerabilities // Informatizatsiya i svyaz'. 2014. No. 3. pp. 44–47.

24. Buinevich, M., Izrailov, K., Vladyko, A. Testing of Utilities for Finding Vulnerabilities in the Machine Code of Telecommunication Devices // 19th International Conference on Advanced Communication Technology (ICACT). 2017. pp. 408–414.

Израилов Константин Евгеньевич – аспирант, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, konstantin.izrailov@mail.ru

Покусов Виктор Владимирович – аспирант, Санкт-Петербургский университет ГПС МЧС России, Санкт-Петербург, 196105, Российская Федерация, v@victor.kz

Izrailov Konstantin – postgraduate, SPbSUT, St. Petersburg, 193232, Russian Federation, konstantin.izrailov@mail.ru

Pokusov Viktor – postgraduate, Saint-Petersburg University of State Fire Service of EMERCOM of Russia, St. Petersburg, 196105, Russian Federation, v@victor.kz