МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Гжельский государственный университет»** (ГГУ)

Колледж ГГУ

Специальность 09.02.07 Информационные системы и программирование

**Реферат**

**По теме «**Обратное проектирование алгоритма**»**

ВЫПОЛНИЛ:

Студент группы ИСП-О-17

Попов Тарас Алексеевич

ПРОВЕРИЛА:

Прокуронова А.Ю.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

пос. Электроизолятор,

2019 год

**Алгоритм** — это точно определённая инструкция, последовательно применяя которую к исходным данным, можно получить решение задачи. Для каждого алгоритма есть некоторое множество объектов, допустимых в качестве исходных данных. Например, в алгоритме деления вещественных чисел делимое может быть любым, а делитель не может быть равен нулю.

      Алгоритм служит, как правило, для решения не одной конкретной задачи, а некоторого класса задач. Так, алгоритм сложения применим к любой паре натуральных чисел. В этом выражается его свойство массовости, то есть возможности применять многократно один и тот же алгоритм для любой задачи одного класса.

      Для разработки алгоритмов и программ используется алгоритмизация — процесс систематического составления алгоритмов для решения поставленных прикладных задач. Алгоритмизация считается обязательным этапом в процессе разработки программ и решении задач на ЭВМ. Именно для прикладных алгоритмов и программ принципиально важны детерминированность, результативность и массовость, а также правильность результатов решения поставленных задач.

**Форма алгоритмов**

      Алгоритм может быть записан словами и изображён схематически. Обычно сначала (на уровне идеи) алгоритм описывается словами, но по мере приближения к реализации он обретает всё более формальные очертания и формулировку на языке, понятном исполнителю (например, машинный код).

      Например, для описания алгоритма применяются блок-схемы. Другим вариантом описания, не зависимым от языка программирования, является псевдокод.

**Эффективность алгоритмов**

      Хотя в определении алгоритма требуется лишь конечность числа шагов, требуемых для достижения результата, на практике выполнение даже хотя бы миллиарда шагов является слишком медленным. Также обычно есть другие ограничения (на размер программы, на допустимые действия). В связи с этим вводят такие понятия как сложность алгоритма (временна́я, по размеру программы, вычислительная и др.).

      Для каждой задачи может существовать множество алгоритмов, приводящих к цели. Увеличение эффективности алгоритмов составляет одну из задач современной информатики. В 50-х гг. XX века появилась даже отдельная её область — быстрые алгоритмы. В частности, в известной всем с детства задаче об умножении десятичных чисел обнаружился ряд алгоритмов, позволяющих существенно ускорить нахождение произведения.

*Под обратным проектированием* (reverse engineering) понимают процесс исследования и анализа машинного кода, нацеленный на понимание общих механизмов функционирования программы, а также на его перевод на более высокий уровень абстракции (более высокий уровень языка программирования) вплоть до восстановления текста программы на исходном языке программирования.

Основными методами обратного проектирования являются отладка и дизассемблирование программ. При этом используются следующие средства (инструменты).

*Отладчики* – программные средства, позволяющие выполнять программу в пошаговом режиме, контролировать ее выполнение, вносить изменения в ход выполнения. Данные средства позволяют проследить весь механизм работы программы и являются средствами динамического исследования работы программ. В качестве примера одного из наиболее мощных отладочных средств можно привести отладчик SoftIce.

*Дизассемблеры* – программные средства, позволяющие получить листинг программы на языке ассемблера, с целью его дальнейшего статического изучения. Дизассемблеры являются средствами статического исследования.

По способу реализации интерфейса взаимодействия с пользователем, существующие дизассемблеры можно разделить на две категории – автономные и интерактивные.

Автономные дизассемблеры требуют от пользователя задания всех указаний до начала дизассемблирования и не позволяют вмешиваться непосредственно в сам процесс. Если же конечный результат окажется неудовлетворительным, пользователь либо вручную правит полученный листинг, либо указывает дизассемблеру на его ошибки и повторяет всю процедуру вновь и вновь, порой десятки раз. Такой способ общения человека с дизассемблером непроизводителен и неудобен, но его легче запрограммировать.

Интерактивные дизассемблеры обладают развитым пользовательским интерфейсом, благодаря которому приобретают значительную гибкость, позволяя человеку «вручную» управлять разбором программы, помогая автоматическому анализатору там, где ему самому не справиться – отличать адреса от констант, определять границы инструкций и т.д.

Примером автономного дизассемблера является SOURCER, а интерактивного – IDA Pro.

Мониторы событий – программные средства, позволяющие отслеживать определенные типы событий, происходящие в системе.

Редакторы кода занимают отдельное место среди средств обратного проектирования. Данные средства, как правило, включают функции дизассемблирования, но позволяют также вносить изменения в код программы. Данные средства предназначены, в основном, для исследования программ, занимающих небольшой объём.

Классификацию средств обратного проектирования программного обеспечения можно представить в виде схемы.

Таким образом, использование техники обратного проектирования позволяет злоумышленнику детально разобраться с принципами функционирования программы, в том числе и модулей защиты. Это может ему позволить в дальнейшем атаковать отдельные модули системы защиты. Производитель ПО должен стремиться не допустить этого, он должен противодействовать внутреннему исследованию кода своей программы злоумышленником. Таким образом, защита своих программ от внутреннего исследования (в особенности критических модулей, ответственных за безопасность), является одной из актуальных задач для производителя программного обеспечения. Производитель должен уметь противостоять всем средствам обратного проектирования, облегчающим злоумышленнику задачу взлома.