УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные тезисы, дайте ответы на контрольные вопросы.

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 20.02.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 32

Тема «Отладка программы»

Цель: Ознакомится с основными аспектами, средствами и методами отладки программ.

План лекции:

1. Трудности отладки

2. Средства и методы отладки

3. Рекомендации по отладке

1. Трудности отладки

Отладка (debugging) — процесс нахождения местоположения ошибок в программе и их исправление. Процесс отладки начинается после обнаружения факта ошибки в результате тестирования и осуществляется в два этапа.

Первый этап — установление причины ошибки и ее локализация (определение ее местоположения в программе); второй — исправление ошибки и проверка правильности работы программы.

Таким образом, основным средством обнаружения ошибок при отладке является тестирование. Тестирование и отладка при этом обычно проводятся одновременно. Принято выделять три стадии тестирования для:

– обнаружения ошибки в программе;

– установления местонахождения ошибки;

– подтверждения правильности работы программы после проведенной корректировки.

Как показывает опыт разработки программного обеспечения, трудоемкость отладки превышает суммарную трудоемкость разработки алгоритма, программирования (кодирования) и тестирования. Затраты времени на отладку составляют (по результатам ряда исследователей) от 50 до 80% общего времени разработки программы, поэтому отладку иногда называют искусством обнаружения местоположения ошибок в программе.

В результате разработки многочисленных программных изделий было установлено, что даже опытные программисты испытывают серьезные трудности в обнаружении причины ошибки. Они не знают, какие из ошибок наиболее вероятны, какие участки программ подвержены ошибкам в наибольшей степени, какие стратегии поиска ошибок наиболее эффективны и т.п. Это обусловлено тем, что обычно учат алгоритмизации и программированию задач, но не учат отладке и тестированию. Кроме этого, программисты относятся к процессу отладки, как правило, негативно, предпочитая ему более творческий процесс разработки алгоритма.

К причинам, определяющим значительную трудоемкость процесса отладки, следует отнести и нарушение программистами дисциплины структурной методологии, т.е. принципа формальности, который требует следования при проектировании и кодировании формальным правилам и методам разработки структурных программ. Выполнение структурных требований позволяет избежать большинства ошибок и заметно упростить процедуру отладки программы. К сожалению, многие программисты стремятся как можно быстрее написать программу, а затем обнаруживать ошибки путем многократного ее выполнения с разнообразными тестовыми данными без их внимательного анализа и тщательного проектирования. Такой подход, обусловленный во многом леностью разработчика, приводит к значительным затратам времени при установлении причин ошибок и их локализации, а также к заметному снижению надежности программ, что в дальнейшем проявляется на этапе сопровождения программного изделия.

Машинные методы отладки для опытных и творческих программистов оказываются наименее эффективными. Как показывают исследования, наиболее существенными для повышения эффективности отладки являются априорные знания о статистике ошибок и их наиболее вероятных типах, а также знания о структуре программы и об участках программы, в наибольшей степени предрасположенных к ошибкам. Такими участками, как показывает опыт, являются прежде всего участки с высокой сложностью, поэтому высокое цикломатическое число какого-либо модуля программы позволяет предполагать более высокую вероятность появления в нем ошибок.

С точки зрения организации работ по отладке программы наиболее эффективным считается групповой метод отыскания ошибок (их причин и местоположения в программе), когда два программиста сначала независимо, а затем совместно осуществляют этот процесс. Такой подход более предпочтителен, чем индивидуальная работа одного программиста. Повышение эффективности работы по обнаружению и локализации ошибок может быть достигнуто путем попеременного внимательного анализа программы за столом и машинного тестирования.

К трудностям отладки программ следует также отнести и разнообразие ситуаций, возникающих перед началом ее выполнения. Ситуации могут быть следующие:

1. Компилятор не выдает сообщений об ошибках, но программа не компилируется.

2. Программа откомпилирована, но при выполнении не выдает никаких результатов.

3. Программа откомпилирована, но при выполнении происходит преждевременный останов.

4. Программа зацикливается.

5. Программа выдает неверные результаты.

2. Средства и методы отладки

Отладка начинается после обнаружения ошибки. Существуют следующие средства обнаружения причины ошибки и ее локализации:

1. Листинг исходного кода.

2. Подробная спецификация программы.

3. Детальный алгоритм программы, представленный в виде блок-схемы, схемы действий, таблицы решений и т.п.

4. Выходной листинг.

5. Анализ последовательности выполнения операторов и оценка ожидаемых значений переменных.

6. Отслеживание обращений к подпрограммам.

7. Дампы памяти.

Эти средства предполагают проведение внимательного анализа программы за столом.

Существенную помощь в отладке оказывают инструментальные средства:

– отладочные компиляторы применительно к языку программирования;

– специальные средства расширения языка программирования для контроля типов и диапазонов значений данных, обработки исключительных ситуаций и т.д.;

– средства для печати значений используемых переменных при аварийном завершении программы, для трассировки значений переменных в процессе выполнения программы и т.п.;

– пакеты программ для прослеживания потоков управления и данных в программе, контроля индексов и регистрации вызовов подпрограмм;

– генераторы тестовых данных, формирующие тестовые наборы данных в соответствии со спецификациями, задаваемыми пользователем;

– специальные онлайновые отладчики, обеспечивающие автоматизацию рестартов, остановов и прерываний программы, просмотр работы отдельных операторов и т.п.;

– пакеты словарей/справочников данных, позволяющие контролировать имена и типы данных и их использование разными модулями программы;

– CASE-средства для построения схем потоков данных, моделей данных, схем алгоритмов и т.п.;

– автоматизированные рабочие места программистов, включающие большинство из перечисленных средств.

Для отладки применяется ряд методов, которые можно укрупненно представить в виде следующих групп.

1. Метод "грубой силы". Наиболее распространенный и традиционно используемый программистами подход; связан со всесторонним анализом за столом исходного кода и алгоритма программы, выходных результатов и сообщений компилятора. Прежде широко использовался анализ содержимого памяти (дампа памяти) обычно в шестнадцатеричной или восьмеричной форме.

Для повышения эффективности отладки в текст программы включают операторы отладочного кода. Наиболее просто осуществить вставку операторов, которые регистрируют результаты исполнения конкретного оператора. Часто — это распечатка выборочных значений переменных. Кроме того, может осуществляться проверка завершения логических участков программ, результаты выполнения условий и т.п. Такой метод вставок предполагает, что после проверяемых включаются операторы, регистрирующие результаты выполнения контролируемых. После завершения отладки программы отладочные операторы можно оставить в виде комментариев для возможного использования их в дальнейшем на этапе сопровождения программного изделия.

К сожалению, подобный подход требует использования большого объема тестовых данных, поскольку процедура локализации ошибок достаточно случайна. Хотя программисту предоставляется возможность работать за терминалом, анализируя работу программы в динамике и используя при этом разнообразные инструментальные отладочные средства, эффективность его работы методом проб и ошибок остается низкой.

Опыт разработки особенно сложных программ показывает, что более рационально искать местоположение ошибки не путем многократного выполнения программы со случайными тестовыми наборами данных, а путем систематического и тщательного обдумывания и анализа решаемой задачи.

2. Метод индукции. Большая часть ошибок может быть локализована в результате анализа алгоритма решаемой задачи, используя стратегию движения от частного к общему. В результате тестирования разработчик получает данные, отражающие как правильные, так и неверные действия программы- Данные

должны быть систематизированы и хорошо структурированы (например, представлены в табличной форме), с указанием симптомов ошибки, места и времени ее появления. Одновременно указываются тестовые наборы данных, приводящие к неверным результатам, и те, которые дают правильный результат. В результате анализа этих данных и взаимосвязей между различными признаками ошибки выявляются определенные закономерности и формулируется гипотеза о причинах ошибки.

Для доказательства правильности выдвинутой гипотезы необходимо показать, что она полностью объясняет все обнаруженные симптомы ошибки. В противном случае гипотеза отвергается и процесс необходимо повторить, собрав предварительно дополнительные данные для выдвижения новой гипотезы.

3. Метод дедукции. Метод предполагает, что на основе результатов тестирования выдвигается множество возможных гипотез о причине ошибки. Затем из общего списка исключаются предположения, которым противоречат данные тестирования. Если в результате анализа будут исключены все выдвинутые гипотезы, то необходимо с помощью тестирования собрать дополнительные данные и повторить процедуру выдвижения новых предположений о причине ошибки. Когда остается несколько предположений, все они тщательно анализируются, начиная с наиболее правдоподобного. Выбранная гипотеза всесторонне рассматривается и уточняется. Доказательство ее правильности осуществляется, как в предыдущем методе, и, если она оказывается верной, на ее основе находится ошибка.

4. Инверсное прослеживание логики программы. Для небольших программ анализ логики выполнения программы в обратном направлении оказывается довольно эффективным способом обнаружения ошибки. Отладка начинается с точки программы, где обнаружен неверный результат или произошел останов программы. На основе полученных в этой точке значений переменных необходимо определить, исходя из логики программы, какие результаты должны были быть при правильной работе программы. Последовательное продвижение к началу программы позволяет достаточно быстро и точно определить место (и причину возникновения) ошибки, т.е. место между оператором, где результат выполнения

программы соответствовал ожидаемому, и оператором, в котором появились расхождения

3. Рекомендации по отладке

1. Используйте систематический, продуманный заранее подход к отладке. Планируйте процесс отладки и тщательно проектируйте тестовые наборы данных, начиная с наиболее простых вариантов, вначале исключая наименее вероятные источники ошибок.

Для упорядочения процесса тестирования собирайте и анализируйте информацию:

• об особенностях и статистике ошибок;

• о специфике исходных данных и последовательности изменения переменных в программе и их взаимном влиянии;

• о структуре алгоритма и особенностях его программной реализации.

2. В каждый момент времени определяйте местоположение одной ошибки.

3. Используйте средства регистрации и отображения информации об ошибках, включая в программу специальный отладочный код для распечатки выборочных значений переменных, сообщений об окончании отдельных участков программы, трассировки логических условий и т.п.

4. Тщательно изучайте полученные выходные данные и сравнивайте их с ожидаемыми, заранее просчитанными результатами.

5. Обращайте особое внимание на данные, обрабатываемые программой, поскольку функционирование программы — это обработка потока данных. Тщательно анализируйте работу программы для граничных значений проверяемых условий, а также при неправильных входных данных, когда необходимо контролировать ошибки, относящиеся к данным. Контролируйте типы данных, диапазоны их значений, размеры полей и конкретные значения переменных и их точность.

6. Используйте совместный анализ потоков данных и потоков управления для проверки корректности в установлении областей определения данных для разных маршрутов выполнения программы.

7. Используйте одновременно различные средства отладки, не останавливайтесь на одной возможности. Привлекайте наиболее мощные автоматизированные средства и одновременно применяйте ручные методы отладки и тестирования за рабочим столом, проверяя текст программы (индивидуально или группой) с точки зрения ее функционирования и с учетом наиболее вероятных ошибок.

8. Документируйте все обнаруженные и исправленные ошибки, отмечая, где они были найдены, и указывая тип ошибки. Эта информация будет полезной для предсказания источников ошибок в будущем. (Каждый программист имеет свой профиль ошибок, т.е. склонность к ошибкам определенного типа.)

9. Измеряйте сложность программ. Программы (модули) с высоким цикломатическим числом имеют более высокую предрасположенность к ошибкам и будут, вероятно, требовать большего времени для их обнаружения и исправления. В программах с высокой сложностью более высока вероятность ошибок спецификаций и проектирования, а с низкой сложностью — кодирования и канцелярских ошибок.

10. Для повышения опыта и тренировки в отладке программ используйте программы с искусственно помещенными в них ошибками. После определенного периода отладки программисту следует указать на. оставшиеся необнаруженные ошибки.

+Подобное "засеивание" программы дополнительными ошибками широко используется для оценки числа необнаруженных реальных ошибок. Действительно, если предположить, что во время тестирования и отладки с равной вероятностью обнаруживаются и искусственные, и реальные ошибки, то по проценту соотношения обнаруженных внесенных и реальных ошибок можно высказать предположение о числе оставшихся необнаруженных ошибок в программе.

Контрольные вопросы:

1. Трудности отладки

2. Средства и методы отладки

3. Рекомендации по отладке