ОТЛАДКА ПРОГРАММЫ

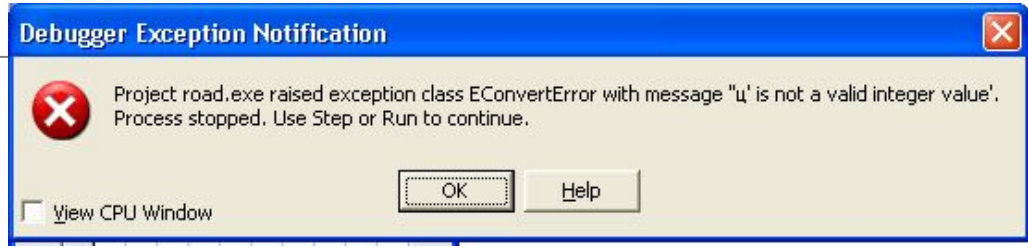
ПРЕДМЕТ: ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ПРЕПОДАВАТЕЛЬ: *КУМСКОВА И.А.*

ОПРЕДЕЛЕНИЯ

...Возмездье
Рукой бесстрастной чашу с нашим ядом
Подносит нам же..,

(Шекспир. Макбет)



Программа, свободная от ошибок, есть абстрактное теоретическое понятие .

ОТЛАДКА ПРОГРАММЫ (*program debugging*) - этап разработки программы, состоящий в локализации, выявлении и устранении программных ошибок, факт существования которых уже установлен.

Отладка имеет место тогда, когда очевидно, что программа либо не компилируется, либо работает неправильно.

Отладка программы предполагает обязательное наличие той или иной ошибки, в противном случае речь идет о тестировании.

СЛОЖНОСТЬ ОТЛАДКИ

ПРИЧИНЫ:

требует от программиста глубоких знаний специфики упр используемыми техническими средствами, операционно системы, среды и языка программирования, реализуемы процессов, природы и специфики различных ошибок, методик отладки и соответствующих программных средств;

психологически дискомфортна, так как необходимо искать собственные ошибки и, как правило, в условиях ограниченного времени;

возможно взаимовлияние ошибок в разных частях программы, например, за счет затирания области памяти одного модуля другим из-за ошибок адресации;

отсутствуют четко сформулированные методики отладки.

ОШИБКИ

Из истории ошибок: ***первая программная ошибка была обнаружена на заре развития ЭВМ, когда в Массачусетском технологическом институте окончилась неудачей попытка запуска машины Whirlwind I.***

В соответствии с этапом обработки, на котором появляются ошибки, различают:

синтаксические ошибки **- ошибки, фиксируемые компилятором (транслятором, интерпретатором) при выполнении синтаксического и частично семантического анализа программы;**

ошибки компоновки **- ошибки, обнаруженные компоновщиком (редактором связей) при объединении модулей программы;**

ошибки выполнения **- ошибки, обнаруженные операционной системой, аппаратными средствами или пользователем при выполнении программы.**

ОШИБКИ ВЫПОЛНЕНИЯ

Способы проявления:

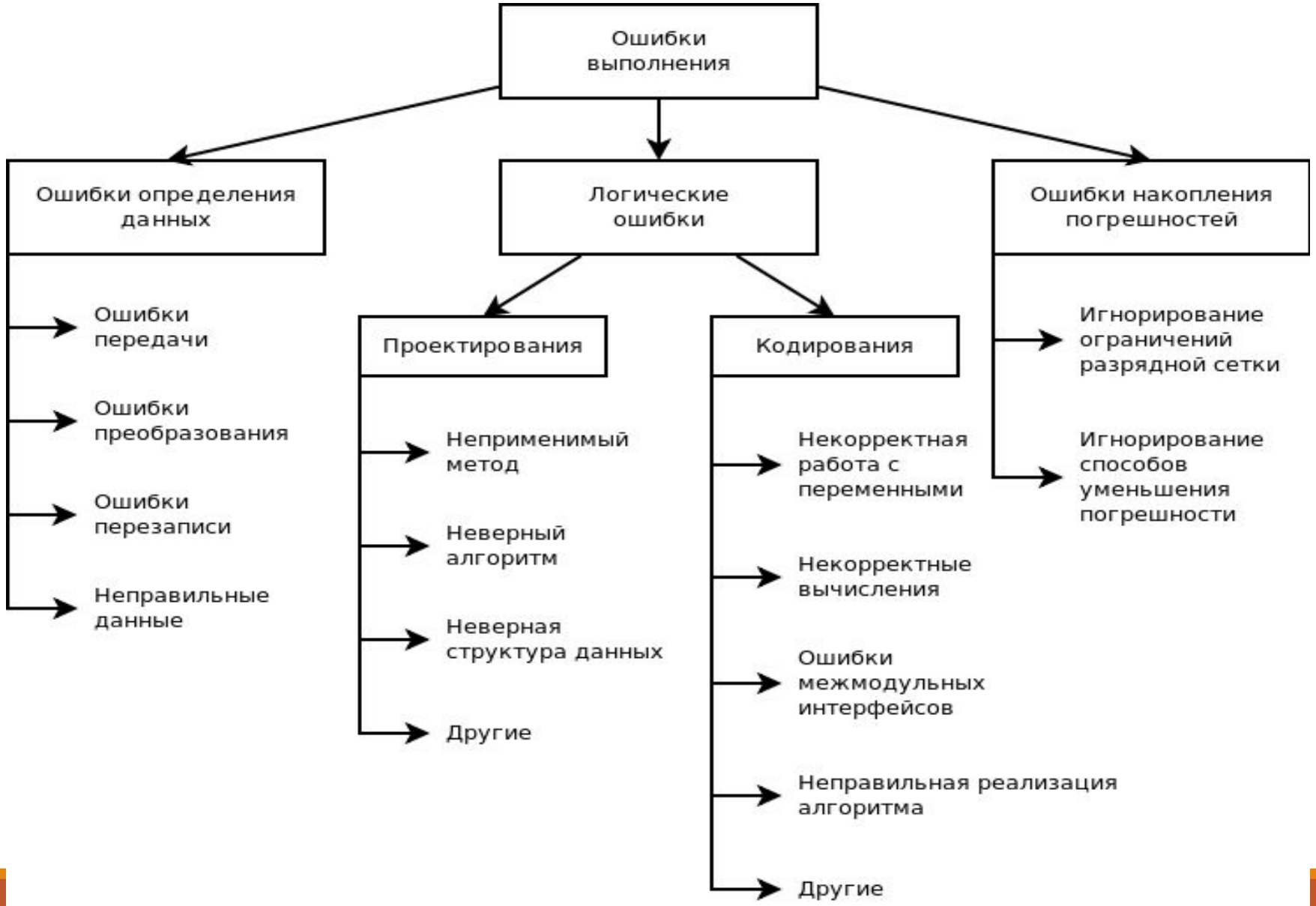
появление сообщения об ошибке, зафиксированной схемами контроля выполнения машинных команд, например, переполнении разрядной сетки, ситуации "деление на ноль", нарушении адресации и т.п.;

появление сообщения об ошибке, обнаруженной операционной системой, например, нарушении защиты памяти, попытке записи на устройства, защищенные от записи, отсутствии файла с заданным именем и т.п.;

"зависание" компьютера, как простое, когда удается завершить программу без перезагрузки операционной системы, так и "тяжелое", когда для продолжения работы необходима перезагрузка;

несовпадение полученных результатов с ожидаемыми.

ОШИБКИ ВЫПОЛНЕНИЯ

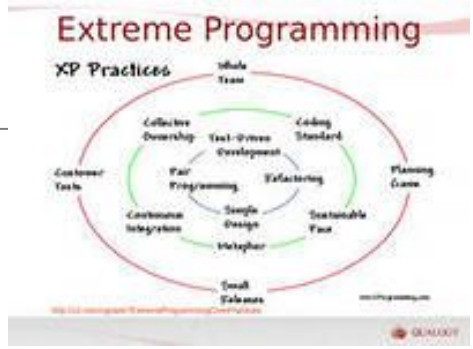


ОШИБКИ

ВИД ОШИБОК	ПРИМЕР
1. Неправильная постановка задачи	Правильное решение неверно сформулированной задачи
2. Неверный алгоритм	Выбор алгоритма, приводящего к неточному или неэффективному решению задачи
3. Ошибки анализа	Неправильное программирование алгоритма
4. Семантические ошибки	Непонимание порядка выполнения команды
5. Синтаксические ошибки	Нарушение правил, определяемых языком программирования
6. Ошибки при выполнении операций	Отсутствие, указаний на ограничивающие условия вычислений (деление на нуль и т.д.)
7. Ошибки в данных	Неудачное определение возможного диапазона изменения данных
8. Ошибки в документации	Документация пользователя не соответствует действующему варианту программы

СЛОЖНОСТЬ ОТЛАДКИ

Сложность отладки увеличивается также вследствие влияния следующих факторов:



опосредованного проявления ошибок;

возможности взаимовлияния ошибок;

возможности получения внешне

одинаковых проявлений разных ошибок;

отсутствия повторяемости проявлений некоторых ошибок от запуска к запуску - так называемые стохастические ошибки;

возможности устранения внешних проявлений ошибок в исследуемой ситуации при внесении некоторых изменений в программу, например, при включении в программу диагностических фрагментов может аннулироваться или измениться внешнее проявление ошибок;

написания отдельных частей программы разными программистами.

ОШИБКИ ОБЩЕГО ХАРАКТЕРА

Логические ошибки

Ошибки в циклах

Ошибки при работе с данными

Ошибки в описании переменных

Ошибки при работе с массивами

Ошибки арифметических операций

Ошибки в подпрограммах

Ошибки ввода-вывода

Ошибки логических операций

МЕТОДИКА ОТЛАДКИ ПО

1 этап **- изучение проявления ошибки:** **если выдано какое-либо сообщение или выданы неправильные или неполные результаты, то необходимо их изучить и попытаться понять, какая ошибка могла так проявиться версии о характере ошибки, которые необходимо проверить.**

2 этап **- локализация ошибки:** **определение конкретного фрагмента, при выполнении которого произошло отклонение от предполагаемого вычислительного процесса.**

3 этап **- определение причины ошибки:** **изучение результатов второго этапа и формирование версий возможных причин ошибки.**

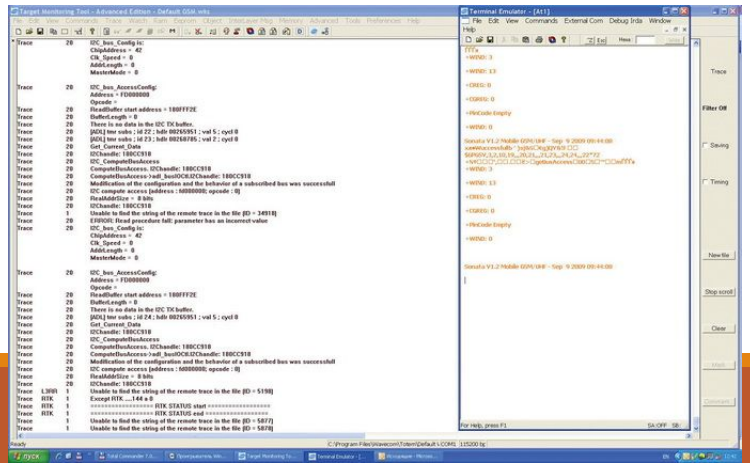
4 этап **- исправление ошибки:** **внесение соответствующих изменений во все операторы, совместное выполнение которых привело к ошибке.**

5 этап **- повторное тестирование:** **повторение всех тестов с начала, так как при исправлении обнаруженных ошибок часто вносят в программу новые.**

СРЕДСТВА ОТЛАДКИ

Типы отладочных средств, применяемых при программировании:

1. **Распечатывание содержимого памяти.**
2. **~~Отслеживание хода выполнения алгоритма.~~**
3. **Отслеживание обращений к переменным.**
4. **Отслеживание обращений к подпрограммам.**
5. **Проверка индексов.**
6. **Воспроизведение значений переменных.**



МЕТОДЫ ОТЛАДКИ

1. **Запуск программы из под отладчика** с пошаговой отладкой, просмотром состояний (переменных, стека, памяти, регистров и т.п.) в требуемых точках исполнения программы.
2. **Логирования кода** – вывод в файл (или консоль и т.п.) входных, выходных аргументов функций, промежуточных состояний (переменных, стека, памяти, передаваемых или получаемых каким-либо образом данных и т.п.) в процессе исполнения программы. При сложностях с воспроизведением сценария дефекта, логирование становится основной методикой отладки.

МЕТОДЫ ОТЛАДКИ

3. **Анализ кода без исполнения программы** – поиск причин возникновения дефекта с помощью анализа исходного кода программы, проблемного контента, конфигурации, состояния базы данных и т.п.

4. **Анализ поведения системы или её части** – изолирование проблемы, путём упрощения сценария (используя ручное или автоматическое тестирование). Аксиома звучит так: чем проще сценарий, тем проще отладить проблему. Если найти более простой сценарий, то отладка может упроститься.

5. **Unit тестирование** – выполнение автоматических unit test-ов в основном изолировано (т.е. в более простых сценариях) для функций (модулей, компонентов и т.п.), и таким образом автоматическое выявление проблемных участков кода. Unit тестирование в каком-то смысле одна из разновидностей отладки путём «анализа поведения системы».

МЕТОДЫ ОТЛАДКИ

6. **Прототипирование** – проверка функций (модулей, библиотек, и т.п.) в изоляции с помощью небольших примеров кода (прототипов). Прототип легче отлаживать, чем целевую систему. Если проблема воспроизводиться с помощью прототипа, отладка упрощается.

7. **Отладка с помощью memory-dump-ов** – разновидность логирования кода, только здесь логируется не просто некая структура памяти, а целиком вся память процесса и состояния регистров, когда возникает exception. По такому дампу памяти можно «раскрутить» состояние программы (стеков, очередей, переменных и т.п.), в котором она находилась во время паники. Достаточно много существует инструментальных средств для выполнения этой операции.

МЕТОДЫ ОТЛАДКИ

8. **Отладка с помощью перехватов** – в основном используется в случаях утечки ресурсов, разновидность логирования кода. Основная идея: перехват и логирование вызова функций выделения и освобождения ресурса, а также анализ состояния ресурсов (например, памяти) в требуемый момент времени или в нужной точке исполнения программы.

9. **Профилирование кода** (если необходима оптимизация производительности) – разновидность логирования кода, хотя часто выполняется с использованием специализированных инструментальных средств (профилировщиков). Этот метод отладки позволяет получить *профиль исполнения* программы – сколько и какая функция, строчка кода, модуль, и т.п. отнимают процессорного времени, и таким образом найти узкие места.

МЕТОДЫ ОТЛАДКИ

10. **Выполнения программы (или её части) в другой среде** (операционной системе, эмуляторе, симуляторе) – основная идея в том, что если нет инструментальных средств на целевой платформе, то можно спортировать код на другую платформу, где они есть. Также можно изначально писать кросс-платформенный код системы или какой-то её части, и таким образом, при необходимости практически без портирования отлаживать код на другой платформе.

11. **Отладка методом RPC** (remote procedure call) – применимо в основном для встроенного программирования. Суть метода в возможности вызвать любую функцию (модуль и т.п.) передавая аргументы и получая результаты исполнения удалённо с одного хоста на другом вместо того, чтобы тратить время на компиляцию или обновление софта на удалённом хосте.

МЕТОДЫ ОТЛАДКИ

12. **Отладка путём анализа документации, дизайна, требований или ограничений модулей** (программных или аппаратных) – применимо в основном для сложных и крупных проектов. Основная идея понять по имеющейся документации допустимо ли поведение, происходящее в дефекте.

13. **Отладка трансляцией кода** – сложный алгоритм пишется или прототипируется на одном языке программирования) с наличием всех доступных инструментальных средств), а потом исходный код отлаженного алгоритма транслируется в ручную или автоматически в другой язык программирования (целевой системы), для которого отсутствуют необходимые инструментальный средства.

Возможны и другие варианты, например, дисассемблерование с целью более низкоуровневого понимания, что происходит при выполнении программы. Т.е. анализируется некий промежуточный вариант кода, который в некоторых ситуациях легче отладить или понять.

МЕТОДЫ ОТЛАДКИ

14. **Отладка разработкой интерпретатора** - метод отладки, корый используется, когда модуль требует частых изменений, а время построения приложения очень большое.

Для ускорения процесса и гибкости пишется небольшой интерпретатор кода с наличием управляющих конструкций. При наличии такого интерпретатора разработчик сравнительно не сложно создаёт скрипты, которые можно быстрее исправить и отладить.

ПРЕДОТВРАЩЕНИЕ ОШИБОК

Не применяйте непроверенных способов программирования.

Старайтесь не использовать принцип умолчания.

Никогда не допускайте зависимости работы программы от достоверности данных.

Добивайтесь полноты логических решений.

Стремитесь минимизировать число обращений к оператору ЭВМ.

СОВЕТЫ ПРОГРАММИСТУ

Применяйте отладочный компилятор.
Первым делом проверяйте программу за столом.
Выполняйте эхо-проверку вводимых данных.
Вводите средства отладки как можно раньше.
Контролируйте правдоподобность вводимых данных.
Используйте доступные для вас средства отладки.
Делайте программу правильной с самого начала.