Место отладки в цикле разработки ИС. Инструменты отладки ИС. Принципы и виды отладки ИС

Отладка (debug, debugging) — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится: узнавать текущие значения переменных; выяснять, по какому пути выполнялась программа.

Процесс отладки начинается с попытки воспроизвести проблему, что может оказаться не простой задачей при программировании параллельных процессов или при некоторых необычных ошибках, известных как гейзенбаги.

Технологии отладки.

1) Использование ***отладчиков*** — программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.

2) Вывод текущего состояния программы с помощью расположенных в *критических точках* программы *операторов вывода* — на экран, принтер, громкоговоритель или в файл. Вывод отладочных сведений в файл называется журналированием.

Инструменты отладки.

1. ***Отладчик*** – программный инструмент, позволяющий программисту наблюдать за выполнением исследуемой программы, останавливать и перезапускать её, прогонять в замедленном темпе, изменять значения в памяти и даже, в некоторых случаях, возвращать назад по времени.

2. ***Профилировщики*** – позволяют определить сколько времени выполняется тот или иной участок кода, а анализ покрытия позволит выявить неисполняемые участки кода.

3. ***API логгеры*** – позволяют программисту отследить взаимодействие программы и Windows API при помощи записи сообщений Windows в лог.

4. ***Дизассемблеры*** позволят программисту посмотреть ассемблерный код исполняемого файла

5. ***Сниферы*** помогут программисту проследить сетевой трафик генерируемой программой

6. ***Сниферы*** *аппаратных интерфейсов* позволят увидеть данные которыми обменивается система и устройство.

7. ***Логи системы***.

Использование *языков программирования высокого уровня*, таких как Java, обычно упрощает отладку, поскольку содержат такие средства как *обработка исключений*, сильно облегчающие поиск источника проблемы. В некоторых низкоуровневых языках, таких как *Ассемблер*, ошибки могут приводить к незаметным проблемам — например, повреждениям памяти или утечкам памяти, и бывает довольно трудно определить, что стало первоначальной причиной ошибки. В этих случаях, могут потребоваться изощрённые приёмы и средства отладки.

Отладка = Тестирование + Поиск ошибок + Редактирование

Виды отладки **ПО, включая тестирование (в нашей стране).**

1.1. ***Автономная*** **отладка**. Последовательное раздельное тестирование различных частей программ, входящих в ПО, с поиском и исправлением в них фиксируемых при тестировании ошибок. Она фактически включает отладку каждого программного модуля и отладку сопряжения модулей.

1.2. ***Комплексная*** **отладка**. Тестирование ПО в целом с поиском и исправлением фиксируемых при тестировании ошибок во всех документах (включая тексты программ ПО), относящихся к ПО в целом. К таким доку-ментам относятся определение требований к ПО, спецификация качества ПО, функциональная спецификация ПО, описание архитектуры П.О. и тексты программ ПО.

2.1. ***Синтаксическая*** **отладка**. Синтаксические ошибки выявляет компилятор, поэтому исправлять их достаточно легко.

2.2. ***Семантическая*** **(*смысловая*) отладка**. Ее время наступает тогда, когда синтаксических ошибок не осталось, но результаты программа выдает неверные. Здесь компилятор сам ничего выявить не сможет, хотя в среде программирования обычно существуют вспомогательные средства отладки, о которых мы еще поговорим.

Взаимосвязь процессов тестирования и отладки **через алгоритм отладки.**

После того как написан рабочий код производятся тестовые запуски программы на различных наборах тестовых данных.

При этом тестер или программист заранее должны получить контрольный результат, с которым будет идти сверка работы проверяемого кода.

В случае обнаружения расхождений между контрольным и фактическим результатами, начинается поиск проблемного участка кода и выявление ошибок вышеуказанными способами.

Средства автоматического тестирования исходного кода программ.

Основной прием здесь это создание тестов исходного текста, которые будут применены к тестируемому участку кода, а система тестирования сообщит об их результатах.

Примерами таких систем могут быть: встроенный модуль doctest в Python и мультиязыковая библиотека тестирования xUnit, распространяемая на условиях GNU/GPL и LGPL. Основа применения всех этих средств и техник, это разбиение одной большой задачи на ряд четких и более маленьких задач.