Частное учреждение образования

«Колледж бизнеса и права»

УТВЕРЖДАЮ

Ведущий

методист колледжа

\_\_\_\_\_\_\_\_\_\_\_ Е.В. Паскал

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_

|  |  |
| --- | --- |
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Технология разработки программного обеспечения» |

**Лабораторная работа № 28**

**Инструкционно-технологическая карта**

Тема: Проверка на наличие вторичных ошибок, осуществление корректировок, отладка программы

Цель: научиться проверять приложение на наличие вторичных ошибок.

Время выполнения: 2 часа

**Контрольныевопросы**

1. Перечислите и опишите виды ошибок.

2. Дайте определение понятию «отладка».

3. Дайте определение понятию «тестирование».

4. Опишите, какие принципы нужно соблюдать для повышения качества тестирования.

5. Перечислите и опишите виды ошибок.

**Теоретические сведения для выполнения работы**

Недостаточно выполнить проектирование и кодирование программного продукта, также необходимо обеспечить его соответствие требованиям и спецификациям. Многократно проводимые исследования показали, что чем раньше обнаруживаются те или иные несоответствия или ошибки, тем больше вероятность их исправления и ниже его стоимость. Современные технологии разработки ПО предусматривают раннее обнаружение ошибок за счет выполнения контроля результатов всех этапов и стадий разработки. На начальных этапах контроль осуществляют вручную или с использованием CASE-средств, на последних – он принимает форму тестирования.

Тестирование – это процесс выполнения программы, целью которого является выявление ошибок. Никакое тестирование не может доказать отсутствие ошибок в сложном ПО, поскольку выполнение полного тестирования становится невозможным и имеется вероятность, что остались невыявленные ошибки. Соблюдение основных правил тестирования и научно обоснованный подбор тестов может уменьшить их количество. Процесс разработки согласно современной модели жизненного цикла ПО предполагает три стадии тестирования:

- автономное тестирование компонентов ПО;

- комплексное тестирование разрабатываемого ПО;

- системное или оценочное тестирование на соответствие основным критериям качества.

Для повышения качества тестирования рекомендуется соблюдать следующие основные принципы:

- предполагаемые результаты должны быть известны до тестирования;

- следует избегать тестирования программы автором;

- необходимо досконально изучать результаты каждого теста;

- необходимо проверять действия программы на неверных данных;

- необходимо проверять программу на неожиданные побочные эффекты на неверных данных.

Вероятность наличия необнаруженных ошибок в части программы пропорциональна количеству ошибок уже найденных в этой части. Удачным считают тест, который обнаруживает хотя бы одну ошибку. Формирование набора тестов имеет большое значение, поскольку тестирование является одним из наиболее трудоемких этапов создания ПО. Доля стоимости тестирования в общей стоимости разработки возрастает при увеличении сложности ПО и повышении требований к их качеству.

Существуют два принципиально различных подхода к формированию тестовых наборов: структурный и функциональный. Структурный подход базируется на том, что известна структура тестируемого ПО, в том числе его алгоритмы («стеклянный ящик»). Тесты строятся для проверки правильности реализации заданной логики в коде программы. Функциональный подход основывается на том, что структура ПО не известна («черный ящик»). В этом случае тесты строят, опираясь на функциональные спецификации. Этот подход называют также подходом, управляемым данными, так как при его использовании тесты строят на базе различных способов декомпозиции множества данных. Наборы тестов, полученные в соответствии с методами этих подходов, объединяют, обеспечивая всестороннее тестирование ПО.

Ручной контроль используют на ранних этапах разработки. Все проектные решения анализируются с точки зрения их правильности и целесообразности как можно раньше, пока их можно легко пересмотреть. Различают статический и динамический подходы к ручному контролю. При статическом подходе анализируют структуру, управляющие и информационные связи программы, ее входные и выходные данные. При динамическом - выполняют ручное тестирование (вручную моделируют процесс выполнения программы на заданных исходных данных). Исходными данными для таких проверок являются: техническое задание, спецификации, структурная и функциональная схемы программного продукта, схемы отдельных компонентов, а для более поздних этапов - алгоритмы и тексты программ, а также тестовые наборы. Доказано, что ручной контроль способствует существенному увеличению производительности и повышению надежности программ и с его помощью можно находить от 30 до 70 % ошибок логического проектирования и кодирования. Основными методами ручного контроля являются: инспекции исходного текста, сквозные просмотры, проверка за столом, оценки программ.

В основе структурного тестирования лежит концепция максимально полного тестирования всех маршрутов, предусмотренных алгоритмом (последовательности операторов программы, выполняемых при конкретном варианте исходных данных). Недостатки: построенные тестовые наборы не обнаруживают пропущенных маршрутов и ошибок, зависящих от заложенных данных; не дают гарантии, что программа правильна.

Другим способом проверки программ является функциональное тестирование: программа рассматривается как «черный ящик», целью тестирования является выяснение обстоятельств, когда поведение программы не соответствует спецификации. Для обнаружения всех ошибок необходимо выполнить исчерпывающее тестирование (при всех возможных наборах данных), что для большинства случаев невозможно. Поэтому обычно выполняют «разумное» или «приемлемое» тестирование, ограничивающееся прогонами программы на небольшом подмножестве всех возможных входных данных. При функциональном тестировании различают следующие методы формирования тестовых наборов:

- эквивалентное разбиение;

- анализ граничных значений;

- анализ причинно-следственных связей;

- предположение об ошибке.

При комплексном тестировании используют тесты, построенные по методам эквивалентных классов, граничных условий и предположении об ошибках, поскольку структурное тестирование для него не применимо. Одним из самых сложных является вопрос о завершении тестирования, так как невозможно гарантировать, что в программе не осталось ошибок. Часто тестирование завершают потому, что закончилось время, отведенное на его выполнение. Его сворачивают, обходясь минимальным тестированием , которое предполагает: тестирование граничных значений, тщательную проверку руководства, тестирование минимальных конфигураций технических средств, возможности редактирования команд и повторения их в любой последовательности, устойчивости к ошибкам пользователя.

Отладка – это процесс локализации (определения оператора программы, выполнение которого вызвало нарушение вычислительного процесса) и исправления ошибок, обнаруженных при тестировании ПО. Для исправления ошибки необходимо определить ее причину. Отладка требует от программиста глубоких знаний специфики управления используемыми техническими средствами, операционной системы, среды и языка программирования, реализуемых процессов, природы и специфики ошибок, методик отладки и соответствующих программных средств; психологически дискомфортна (нужно искать собственные ошибки в условиях ограниченного времени); оставляет возможность взаимовлияния ошибок в разных частях программы. Четко сформулированные методики отладки отсутствуют. Различают:

- синтаксические ошибки – сопровождаются комментарием с указанием их местоположения, фиксируются компилятором (транслятором) при выполнении синтаксического и частично семантического анализа;

- ошибки компоновки - обнаруживаются компоновщиком (редактором связей) при объединении модулей программы;

- ошибки выполнения - обнаруживаются аппаратными средствами, операционной системой или пользователем при выполнении программы, проявляются разными способами и в свою очередь делятся на группы:

- ошибки определения исходных данных (ошибки передачи, ошибки преобразования, ошибки перезаписи и ошибки данных);

- логические ошибки проектирования (неприменимый метод, неверный алгоритм, неверная структура данных, другие) и кодирования (ошибки некорректного использования переменных, вычислений, межмодульного интерфейса, реализации алгоритма, другие);

- ошибки накопления погрешностей результатов вычислений (игнорирование ограничений разрядной сетки и способов уменьшения погрешности).

Отладка программы в любом случае предполагает обдумывание и логическое осмысление всей имеющейся информации об ошибке. Большинство ошибок можно обнаружить по косвенным признакам посредством тщательного анализа текстов программ и результатов тестирования без получения дополнительной информации с помощью следующих методов:

1. ручного тестирования (при обнаружении ошибки нужно выполнить тестируемую программу вручную, используя тестовый набор, при работе с которым была обнаружена ошибка);

2. индукции (основан на тщательном анализе симптомов ошибки, которые могут проявляться как неверные результаты вычислений или как сообщение об ошибке);

3. дедукции (вначале формируют множество причин, которые могли бы вызвать данное проявление ошибки, а затем анализируя причины, исключают те, которые противоречат имеющимся данным);

4. обратного прослеживания (для точки вывода неверного результата строится гипотеза о значениях основных переменных, которые могли бы привести к получению данного результата, а затем, исходя из этой гипотезы, делают предположения о значениях переменных в предыдущей точке).

Для получения дополнительной информации об ошибке выполняют добавочные тесты и используют специальные методы и средства: отладочный вывод; интегрированные средства отладки; независимые отладчики.

Общая методика отладки программных продуктов, написанных для выполнения в операционных системах MS DOS и Win32:

- 1 этап – изучение проявления ошибки;

- 2 этап – определение локализации ошибки;

- 3 этап – определение причины ошибки;

- 4 этап – исправление ошибки;

- 5 этап – повторное тестирование.

Процесс отладки можно существенно упростить, если следовать основным рекомендациям структурного подхода к программированию:

1. программу наращивать «сверху-вниз», от интерфейса к обрабатывающим подпрограммам, тестируя ее по ходу добавления подпрограмм;

2. выводить пользователю вводимые им данные для контроля и проверять их на допустимость сразу после ввода;

3. предусматривать вывод основных данных во всех узловых точках алгоритма (ветвлениях, вызовах подпрограмм).

**Порядок выполнения работы**

1. Провести тестирование разработанного программного продукта с помощью отладки и выявить ошибки.
2. Используя теоретический материал, проанализировать, классифицировать имеющиеся ошибки.
3. Осуществить корректировку выявленных ошибок.

**Литература**

Орлов С.А Технологии разработки программного обеспечения: Учебник, 5-е изд. - СПб.: Питер, 2016 с. 442-471

Преподаватель \_\_\_\_\_\_\_\_ А.С. Комаровский

Рассмотрено на заседании цикловой

комиссии ПОИТ №

Протокол № от «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 202\_

Председатель ЦК \_\_\_\_\_\_\_\_ В.Ю. Михалевич